COMP 5334 Advanced Network and Security

Virtual Private Network (VPN) Lab

Final Project

Ali Ustunkol

austunkol@na.edu

1 Introduction

A Virtual Private Network (VPN) is used for creating a private scope of computer communications or providing a secure extension of a private network into an insecure network such as the Internet. VPN is a widely used security technology. VPN can be built upon IPSec or TLS/SSL (Transport Layer Security/Secure Socket Layer). These are two fundamentally different approaches for building VPNs. In this lab, we focus on the TLS/SSL-based VPNs. This type of VPNs is often referred to as TLS/SSL VPNs. The learning objective of this lab is for students to master the network and security technologies underlying VPNs. To achieve this goal, students will be asked to implement a simple TLS/SSL VPN. Although this VPN is simple, it does include all the essential elements of a VPN. The design and implementation of TLS/SSL VPNs exemplify a number of security principles, including the following:

- Virtual Private Network
- TUN/TAP, and IP tunneling
- Routing
- Public-key cryptography, PKI, and X.509 certificate

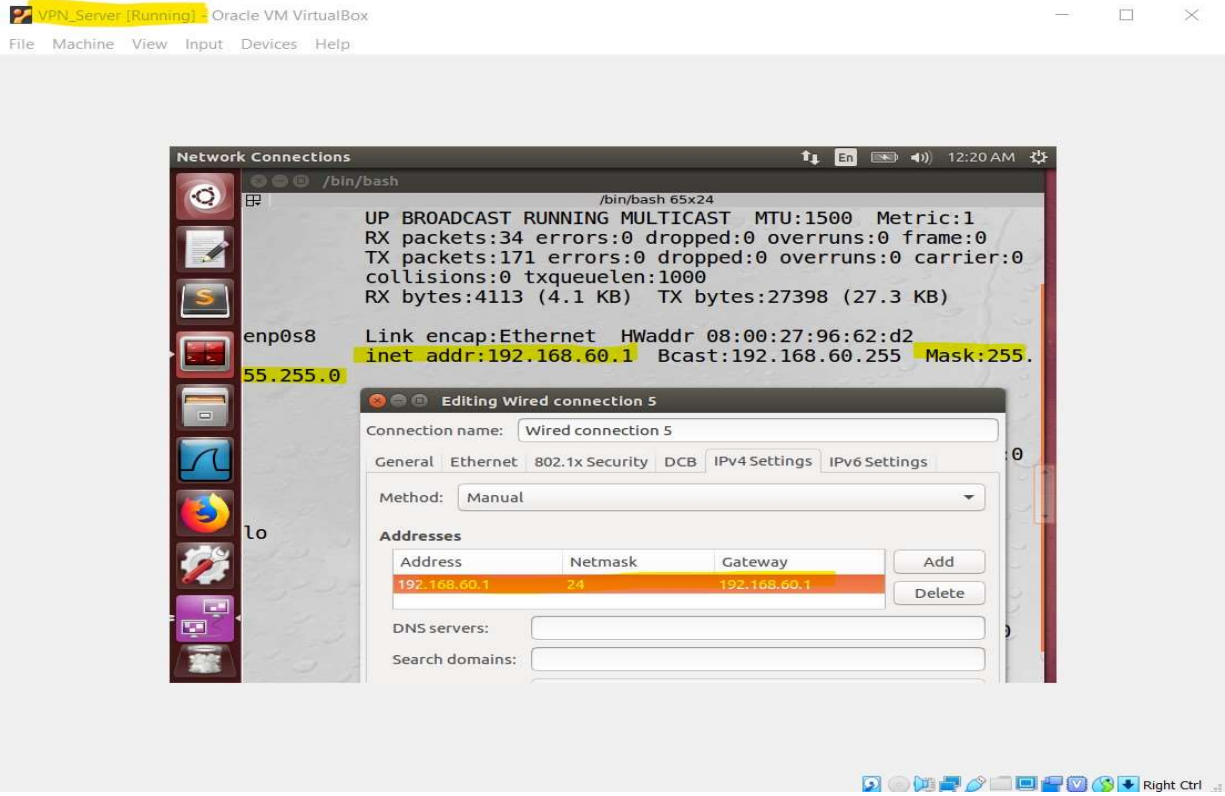
• TLS/SSL programming

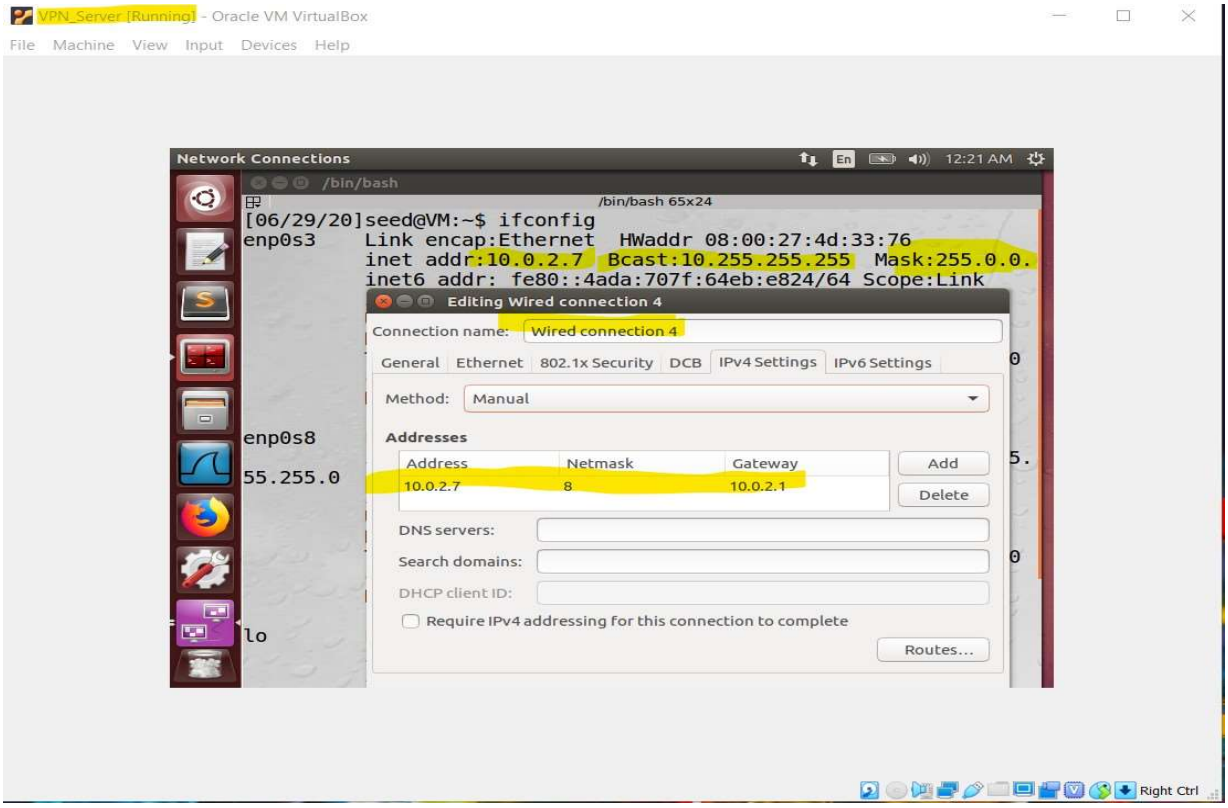
• Authentication

2 Lab Tasks

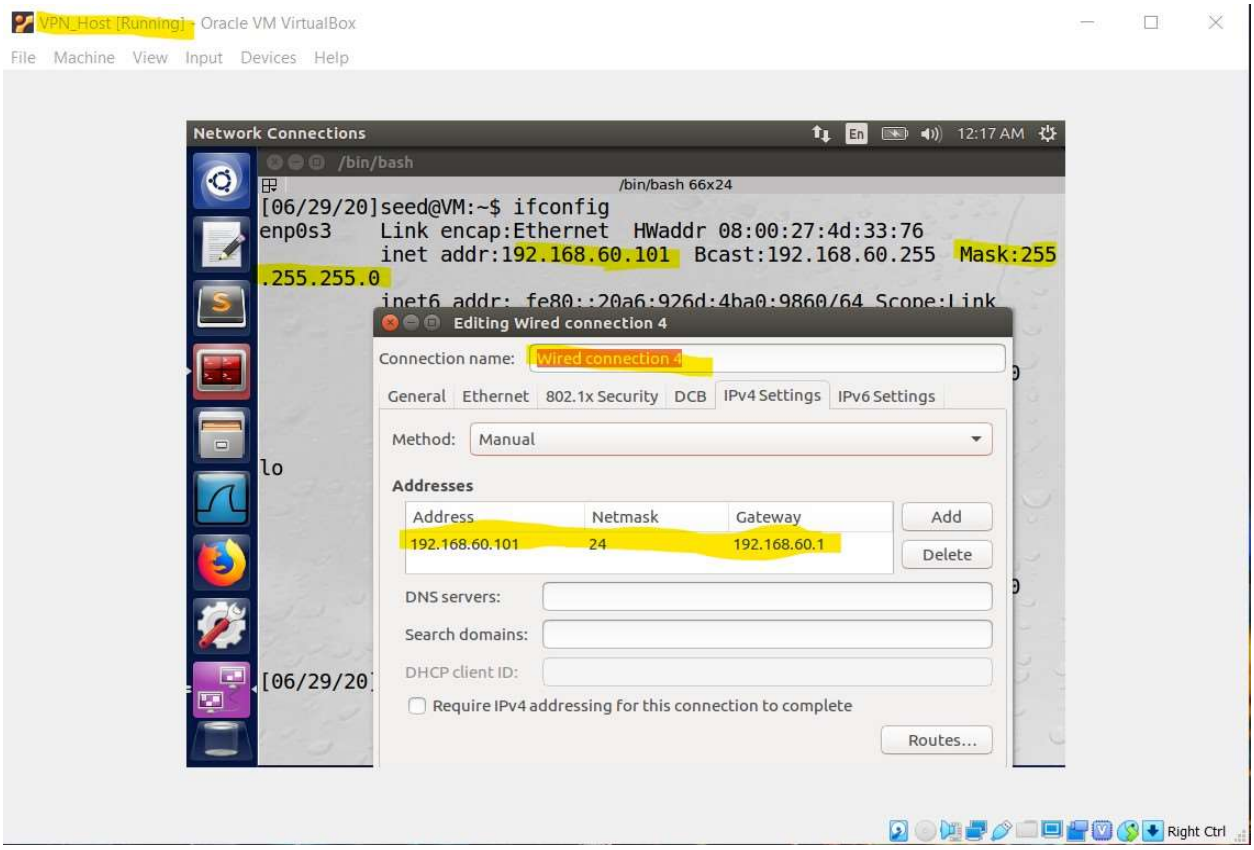
2.1 Task 1: VM Setup

VPN Server: We manually set up the IP address for the "Internal Network" adaptor on VPN Server:

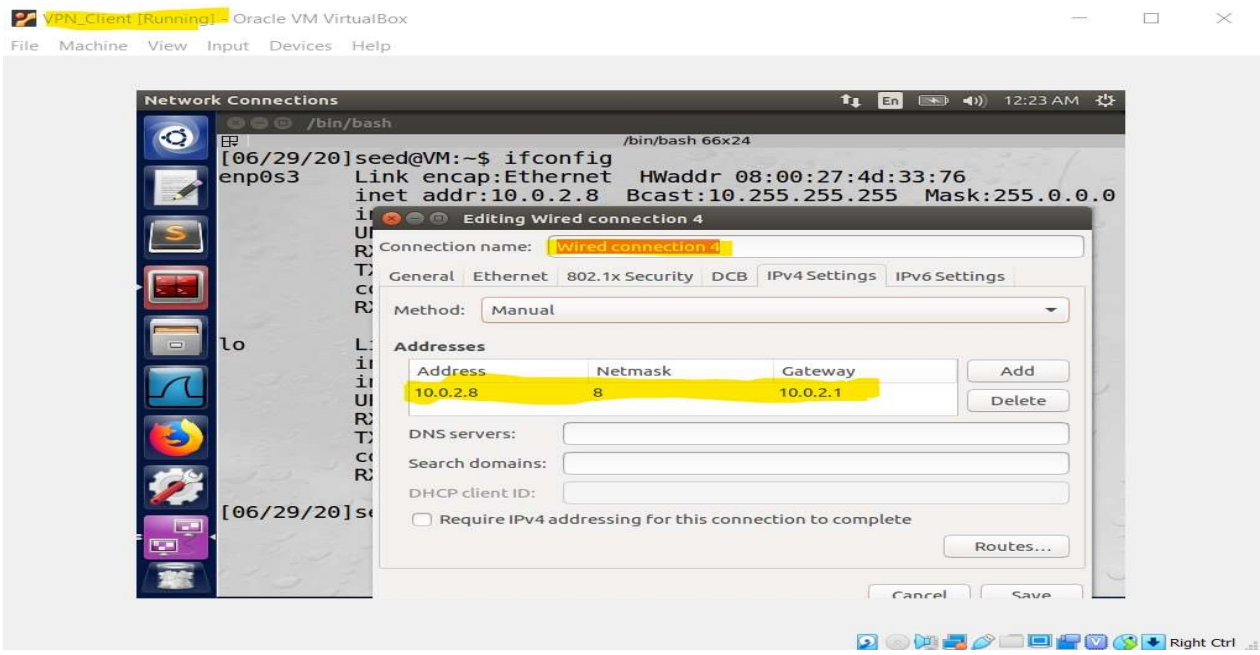




Host V: We set up the Host V by giving the gateway as the server's IP, both of which are connected through internal network.



Also, I created manual IP on the client machine.



2.2 Task 2: Creating a VPN Tunnel using TUN/TAP

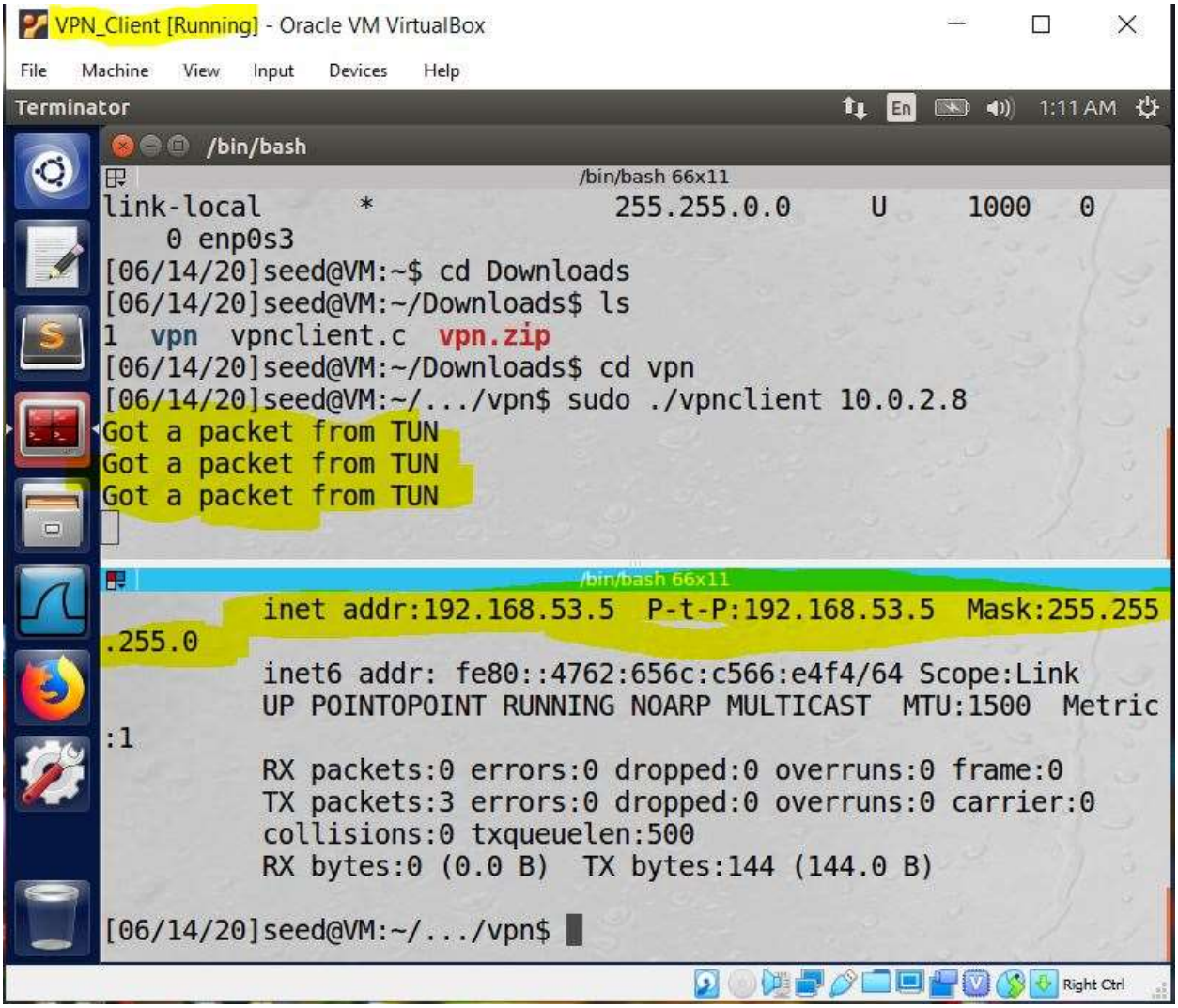
Step 1 Run VPN Server

Assigning IP address 192.168.53.1 to the tun0 device and bringing it up.

I also enable the IP forwarding on the VPN server so that it behaves like a gateway:

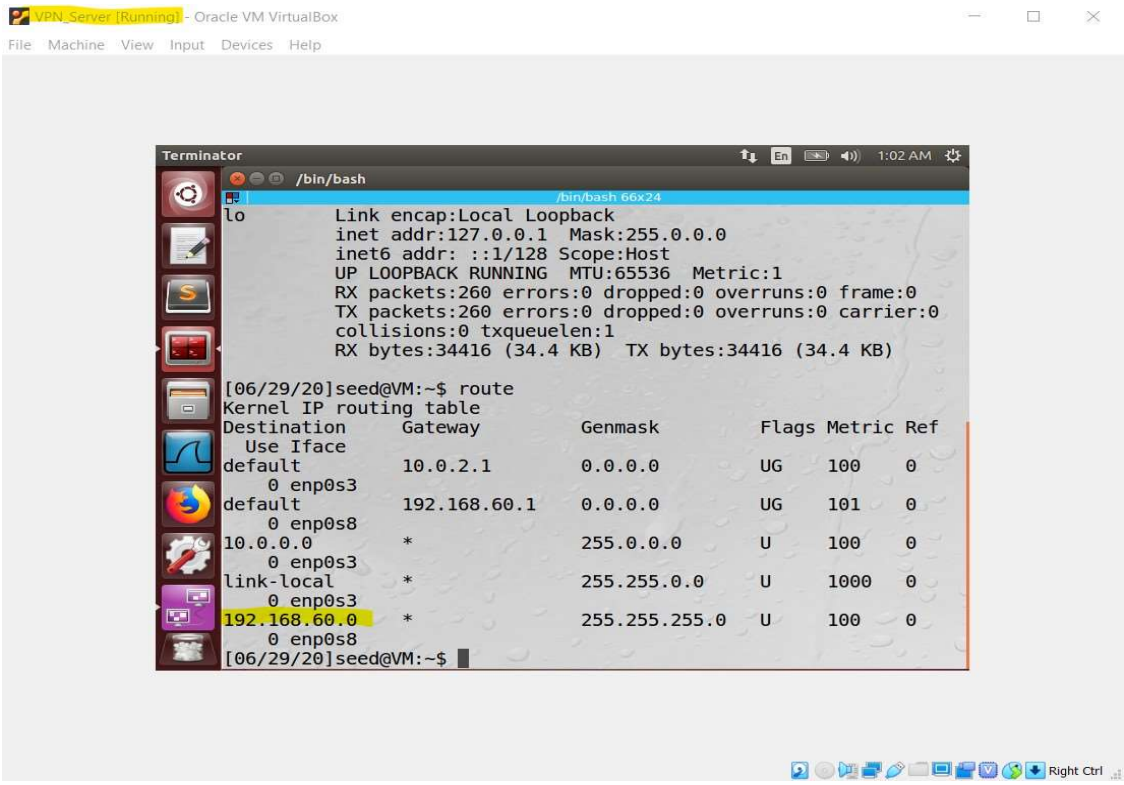
```
/bin/bash
/bin/bash 66x11
          TX packets:1530 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:133829 (133.8 KB)  TX bytes:133829 (133.8 KB)

[06/14/20]seed@VM:~$ cd Downloads
[06/14/20]seed@VM:~/Downloads$
[06/14/20]seed@VM:~/Downloads$ ls
vpn  vpn.zip
[06/14/20]seed@VM:~/Downloads$ cd vpn
[06/14/20]seed@VM:~/.../vpn$ sudo ./vpnserver

/bin/bash 66x11
          TX packets:1588 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:138306 (138.3 KB)  TX bytes:138306 (138.3 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-
00-00-00-00-00-00-00-00
          inet addr:192.168.53.1  P-t-P:192.168.53.1  Mask:255.255
.255.0
          inet6 addr: fe80::c824:b754:27a9:c1a4/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric
:1
```

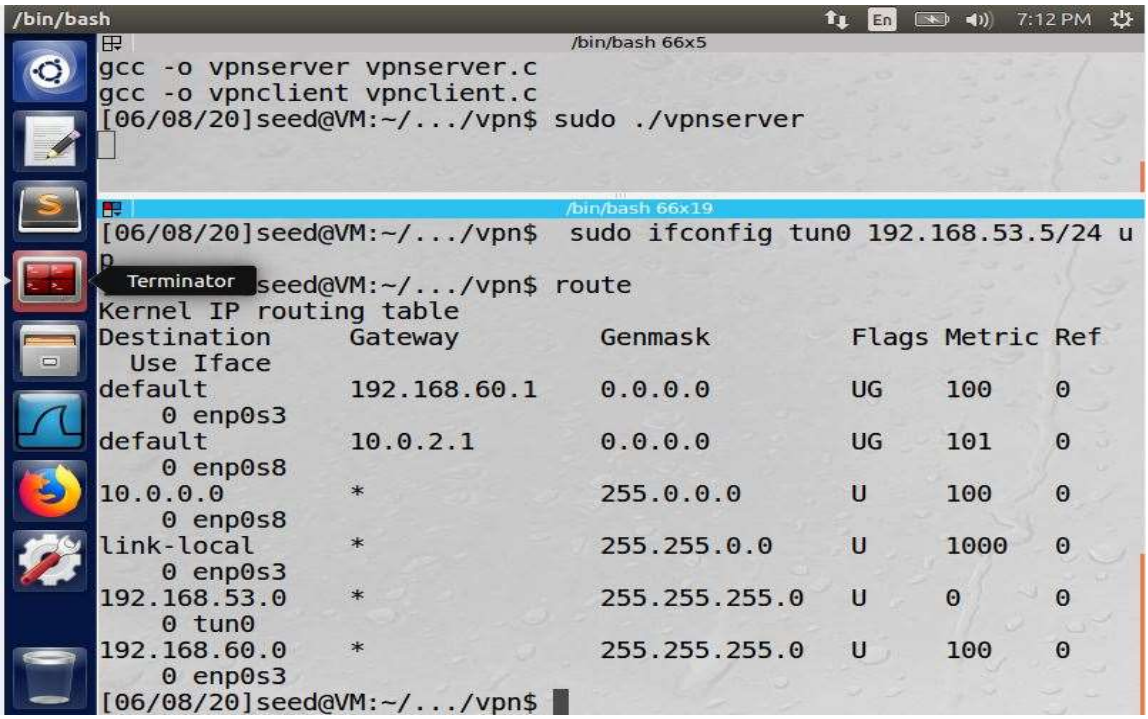
Step 2: Running the VPN Client:



Step 3: Set Up Routing on Client and Server VMs:

Add 192.168.60.0/24 on the client so that it forwards the packet directed to the host V on the network, 192.168.60.0 towards the tun interface.





Step 4: Set Up Routing on Host V

I add the following routing entry in the Host V so that the machine knows that the traffic directed to the network 192.168.53.0/24 goes through the VPN server acting as the gateway.

```
192.168.53.0    192.168.60.1    255.255.255.0   UG    0      0        0
192.168.60.0    0.0.0.0         255.255.255.0   U     100    0        0
[04/25/2018 23:45] () seed@HostV$
```

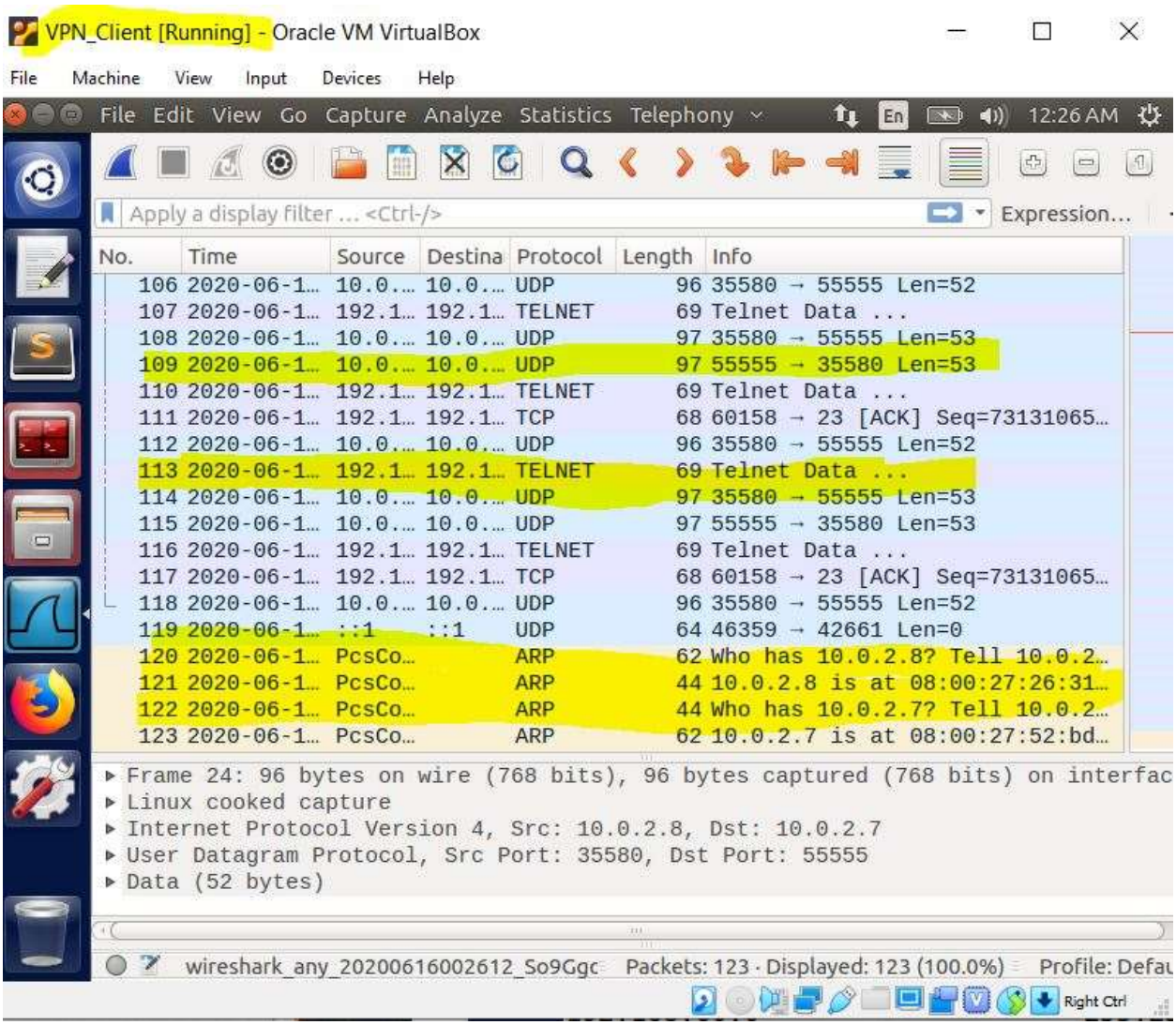
Step 5: Test the VPN Tunnel:

I ping from the VPN Client (i.e. Host U) to Host V:

```
192.168.53.5) seed@HostU$ ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
64 bytes from 192.168.53.1: icmp_seq=1 ttl=64 time=0.453 ms
64 bytes from 192.168.53.1: icmp_seq=2 ttl=64 time=1.30 ms
64 bytes from 192.168.53.1: icmp_seq=3 ttl=64 time=1.27 ms
^C
```

Step 6: Tunnel-Breaking Test:

Wireshark capture.



2.3 Task 3: Encrypting the Tunnel

Running the tlsserver program.

Assigning IP address 192.168.53.1 to the tun0 interface of the server and bringing it up.

I run the client giving the domain and the port number. SSL connection is established between the server and the client. It sends an HTTP Get request to the server and receives a reply.

From the Wireshark capture, we can see that the communication between the client and server happens through the TLS layer and also that the data is encrypted.

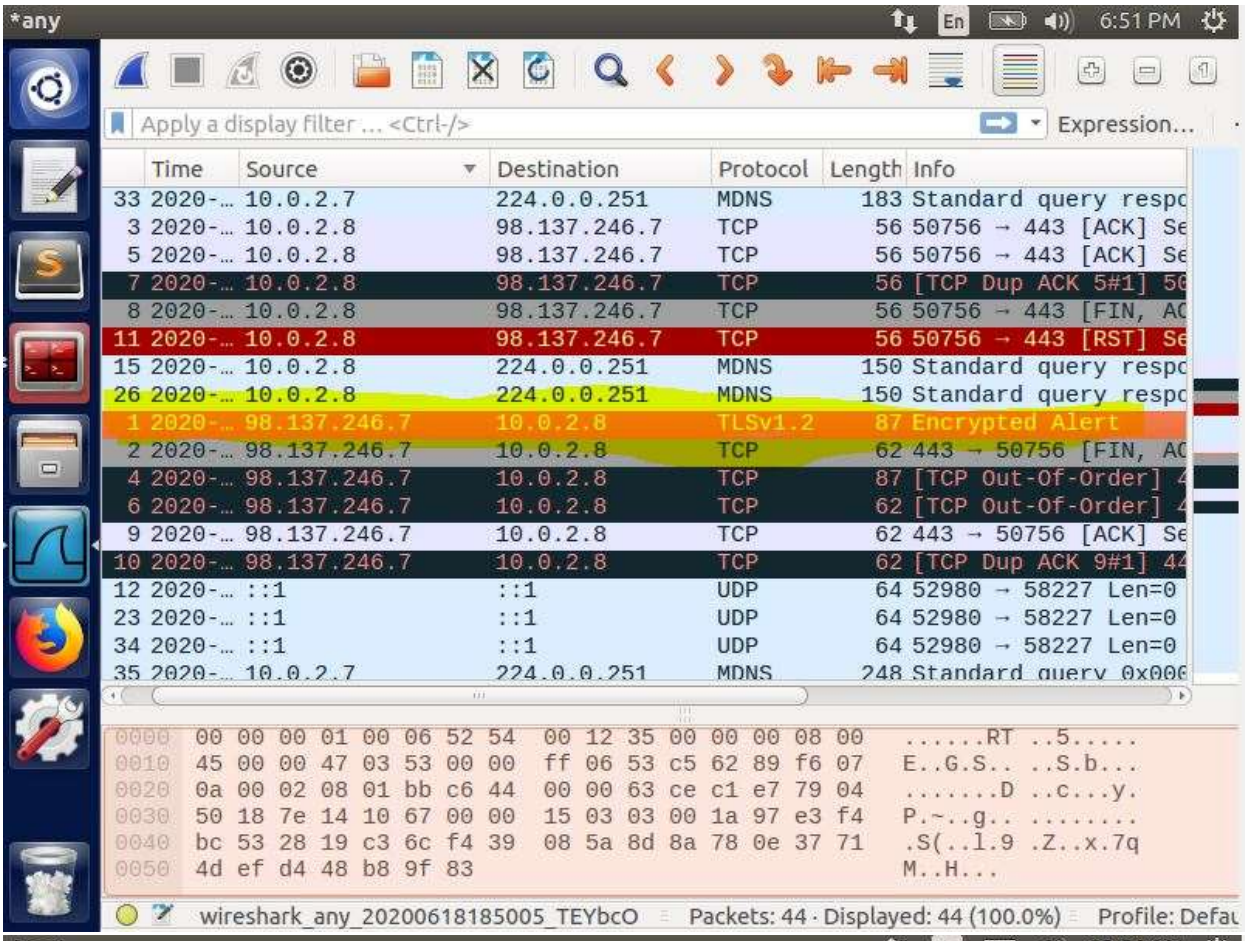
VPN_Server [Running] - Oracle VM VirtualBox

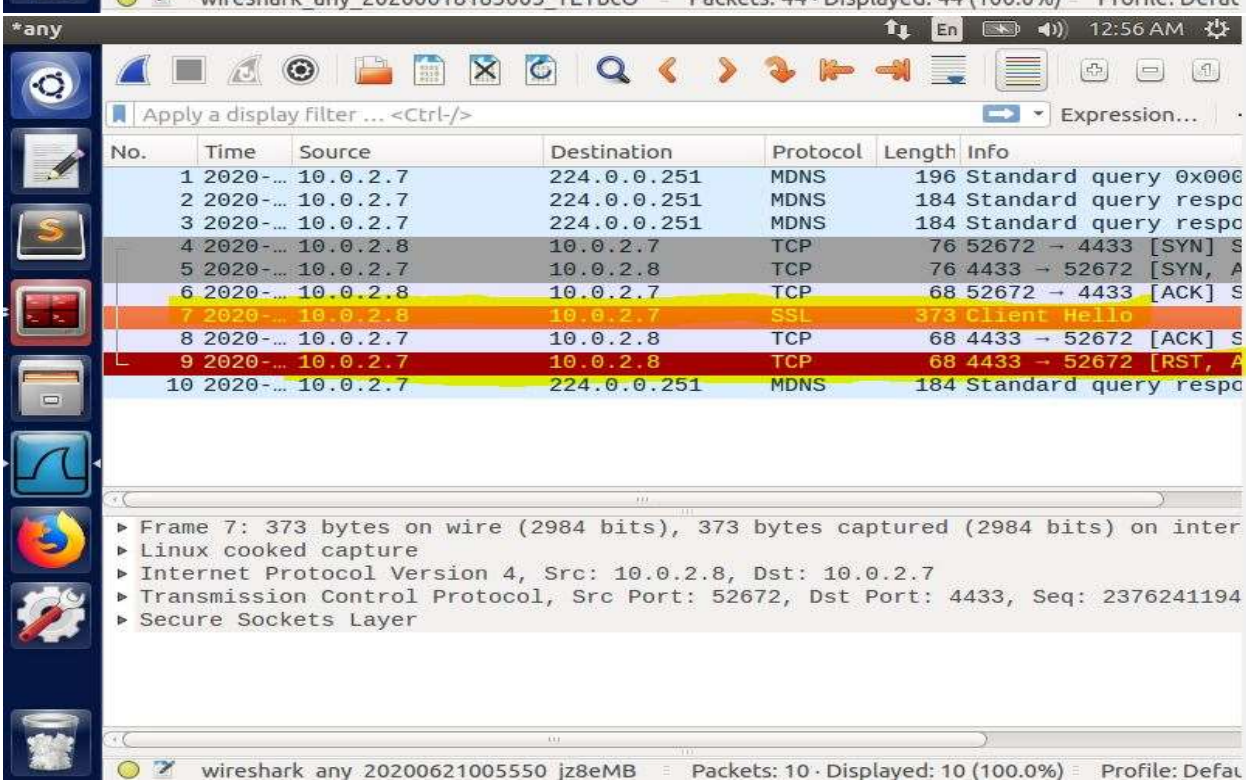
File Machine View Input Devices Help

Terminal 12:49 AM

```
^C
[06/16/20]seed@VM:~/.../tls$ make
gcc -o tlsclient tlsclient.c -lssl -lcrypto
gcc -o tlsserver tlsserver.c -lssl -lcrypto
[06/16/20]seed@VM:~/.../tls$ ./tlsserver
^C
[06/16/20]seed@VM:~/.../tls$ make
gcc -o tlsclient tlsclient.c -lssl -lcrypto
gcc -o tlsserver tlsserver.c -lssl -lcrypto
[06/16/20]seed@VM:~/.../tls$ tlsserver.c
bash: ./tlsserver.c: Permission denied
[06/16/20]seed@VM:~/.../tls$ tlsserver.c -lssl -lcrypto
bash: ./tlsserver.c: Permission denied
[06/16/20]seed@VM:~/.../tls$ gcc -o tlsserver tlsserver
.c -lssl -lcrypto
[06/16/20]seed@VM:~/.../tls$ make
gcc -o tlsclient tlsclient.c -lssl -lcrypto
gcc -o tlsserver tlsserver.c -lssl -lcrypto
[06/16/20]seed@VM:~/.../tls$ gcc -o tlsserver tlsserver
.c -lssl -lcrypto
[06/16/20]seed@VM:~/.../tls$
```

Right Ctrl


*any
6:51 PM
Apply a display filter ... <Ctrl-/>
Expression...
Time Source Destination Protocol Length Info
33 2020-... 10.0.2.7 224.0.0.251 MDNS 183 Standard query respo
3 2020-... 10.0.2.8 98.137.246.7 TCP 56 50756 → 443 [ACK] Se
5 2020-... 10.0.2.8 98.137.246.7 TCP 56 50756 → 443 [ACK] Se
7 2020-... 10.0.2.8 98.137.246.7 TCP 56 [TCP Dup ACK 5#1] 50
8 2020-... 10.0.2.8 98.137.246.7 TCP 56 50756 → 443 [FIN, AC
11 2020-... 10.0.2.8 98.137.246.7 TCP 56 50756 → 443 [RST] Se
15 2020-... 10.0.2.8 224.0.0.251 MDNS 150 Standard query respo
26 2020-... 10.0.2.8 224.0.0.251 MDNS 150 Standard query respo
1 2020-... 98.137.246.7 10.0.2.8 TLSv1.2 87 Encrypted Alert
2 2020-... 98.137.246.7 10.0.2.8 TCP 62 443 → 50756 [FIN, AC
4 2020-... 98.137.246.7 10.0.2.8 TCP 87 [TCP Out-Of-Order] 4
6 2020-... 98.137.246.7 10.0.2.8 TCP 62 [TCP Out-Of-Order] 4
9 2020-... 98.137.246.7 10.0.2.8 TCP 62 443 → 50756 [ACK] Se
10 2020-... 98.137.246.7 10.0.2.8 TCP 62 [TCP Dup ACK 9#1] 44
12 2020-... ::1 ::1 UDP 64 52980 → 58227 Len=0
23 2020-... ::1 ::1 UDP 64 52980 → 58227 Len=0
34 2020-... ::1 ::1 UDP 64 52980 → 58227 Len=0
35 2020-... 10.0.2.7 224.0.0.251 MDNS 248 Standard query 0x000
0000 00 00 00 01 00 06 52 54 00 12 35 00 00 00 08 00 ......RT ..5.....
0010 45 00 00 47 03 53 00 00 ff 06 53 c5 62 89 f6 07 E..G.S.. ..S.b...
0020 0a 00 02 08 01 bb c6 44 00 00 63 ce c1 e7 79 04 .....D ..c...y.
0030 50 18 7e 14 10 67 00 00 15 03 03 00 1a 97 e3 f4 P.~..g.. .....
0040 bc 53 28 19 c3 6c f4 39 08 5a 8d 8a 78 0e 37 71 .S(..l.9 .Z..x.7q
0050 4d ef d4 48 b8 9f 83 M..H...
wireshark_any_20200618185005_TEYbcO
Packets: 44 · Displayed: 44 (100.0%)
Profile: Defau

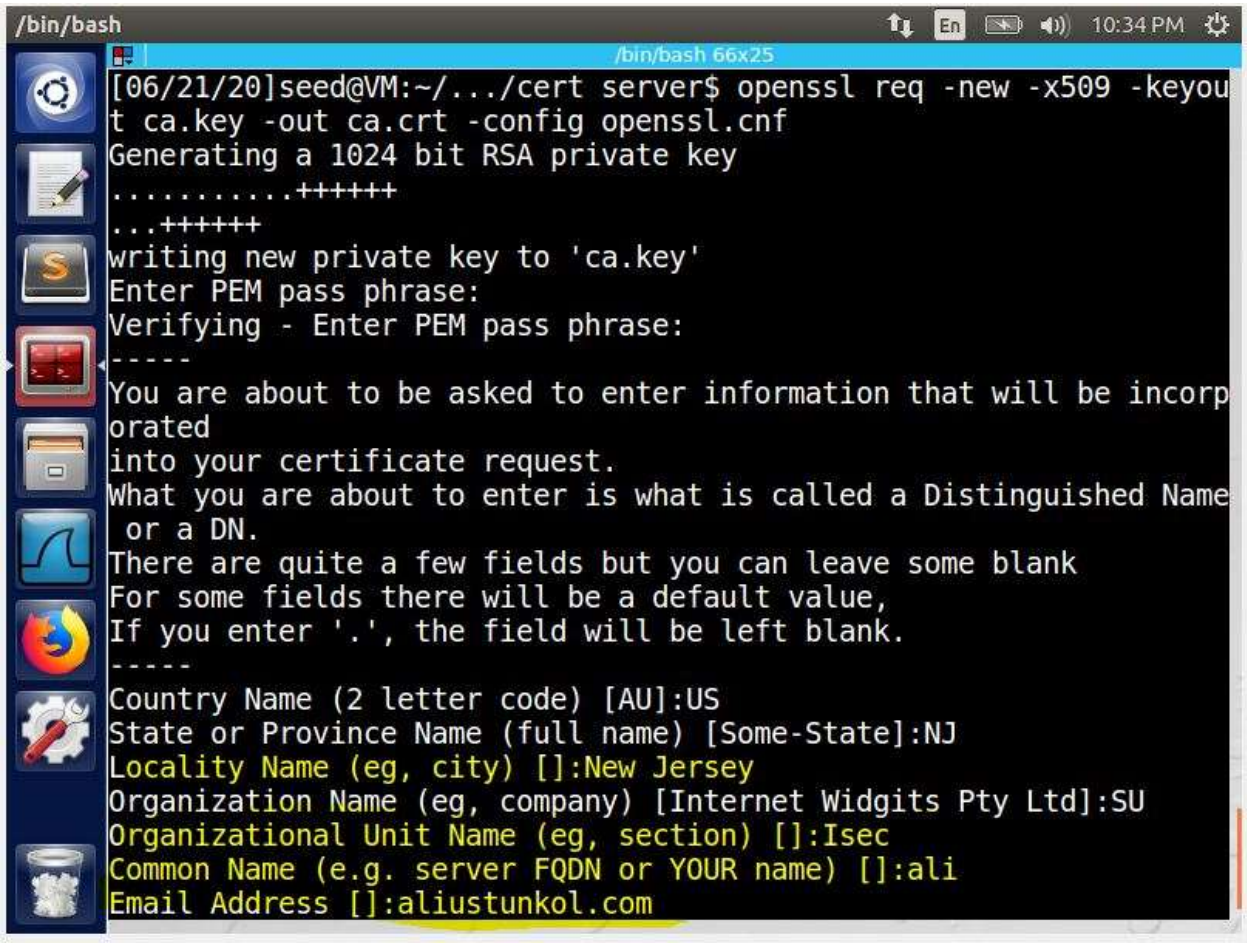

*any
12:56 AM
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
1 2020-... 10.0.2.7 224.0.0.251 MDNS 196 Standard query 0x000
2 2020-... 10.0.2.7 224.0.0.251 MDNS 184 Standard query respo
3 2020-... 10.0.2.7 224.0.0.251 MDNS 184 Standard query respo
4 2020-... 10.0.2.8 10.0.2.7 TCP 76 52672 → 4433 [SYN] S
5 2020-... 10.0.2.7 10.0.2.8 TCP 76 4433 → 52672 [SYN, A
6 2020-... 10.0.2.8 10.0.2.7 TCP 68 52672 → 4433 [ACK] S
7 2020-... 10.0.2.8 10.0.2.7 SSL 373 Client Hello
8 2020-... 10.0.2.7 10.0.2.8 TCP 68 4433 → 52672 [ACK] S
9 2020-... 10.0.2.7 10.0.2.8 TCP 68 4433 → 52672 [RST, A
10 2020-... 10.0.2.7 224.0.0.251 MDNS 184 Standard query respo
▸ Frame 7: 373 bytes on wire (2984 bits), 373 bytes captured (2984 bits) on inter
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.2.8, Dst: 10.0.2.7
▸ Transmission Control Protocol, Src Port: 52672, Dst Port: 4433, Seq: 2376241194
▸ Secure Sockets Layer
wireshark_any_20200621005550_jz8eMB
Packets: 10 · Displayed: 10 (100.0%)
Profile: Defau

2.4 Task 4: Authenticating the VPN Server

Creating a server certificate for aliustunkol.com

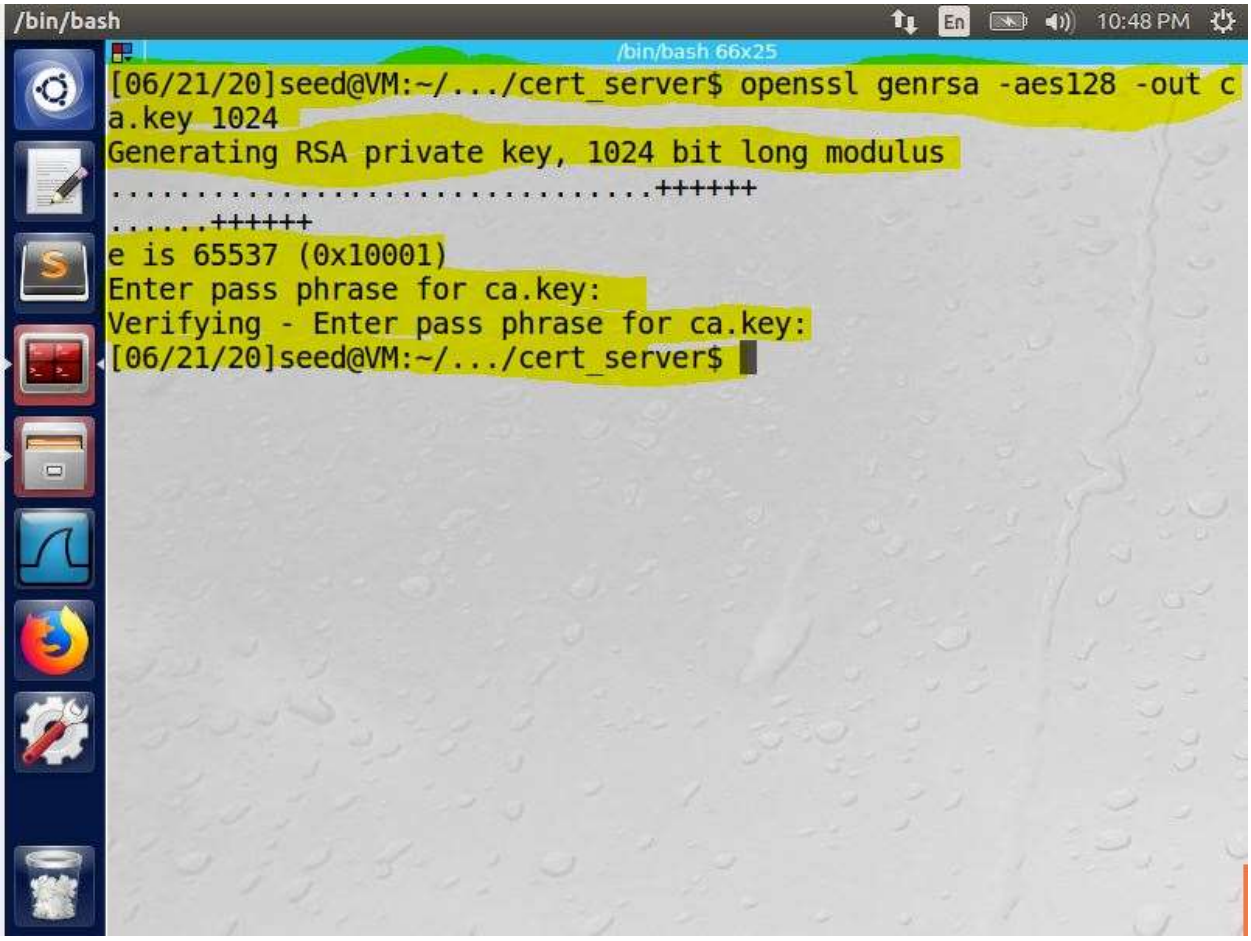
Step 1: Becoming CA

I generate a self-signed certificate for our CA. This means that this CA is totally trusted, and its certificate will serve as the root certificate. The output of the command is: CA's private key and the CA's public-key certificate.



Step 2: Creating a Certificate for *aliustunkol.com*

Generating public/private key pair: We can run the following command to generate an RSA key pair (both private and public keys). We also provide a password to encrypt the private key.



Generate a Certificate Signing Request (CSR)

Now we have the key file, so we generate a Certificate Signing Request (CSR), which basically includes the company's public key. The CSR will be sent to the CA, who will generate a certificate for the key.

```
-out caserver.csr -config openssl.cnf
Enter pass phrase for ca.key:
You are about to be asked to enter information that will be incorp
orated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
 or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:NJ
Locality Name (eg, city) []:NJ
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Ali's C
ompany
Organizational Unit Name (eg, section) []:InternetSecurity
Common Name (e.g. server FQDN or YOUR name) []:aliustunkol.com
Email Address []:aliustunkol.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:ali
string is too short, it needs to be at least 4 bytes long
A challenge password []:aliu
```

Generating Certificates

The CSR file needs to have the CA's signature to form a certificate. The following command turns the certificate signing request (server.csr) into an X509 certificate (server.crt), using the CA's ca.crt and ca.key:

```
/bin/bash                                                    11:41 PM
/bin/bash 66x25
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4098 (0x1002)
        Validity
            Not Before: Jun 22 03:40:55 2020 GMT
            Not After : Jun 22 03:40:55 2021 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = NJ
            organizationName          = ustunkolltd
            organizationalUnitName    = Isec
            commonName                = ali
            emailAddress              = aliustunkol.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                57:DA:80:F7:D8:74:77:9B:E2:27:83:17:51:66:F4:9E:AD
:F1:0E:EE
            X509v3 Authority Key Identifier:
```

```
/bin/bash                                                    11:42 PM
/bin/bash 66x25
        Validity
            Not Before: Jun 22 03:40:55 2020 GMT
            Not After : Jun 22 03:40:55 2021 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = NJ
            organizationName          = ustunkolltd
            organizationalUnitName    = Isec
            commonName                = ali
            emailAddress              = aliustunkol.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                57:DA:80:F7:D8:74:77:9B:E2:27:83:17:51:66:F4:9E:AD
:F1:0E:EE
            X509v3 Authority Key Identifier:
                keyid:58:14:8C:E2:C1:5C:B2:C0:BA:6B:5B:8C:CF:3F:51
:CC:D2:67:9E:52

Certificate is to be certified until Jun 22 03:40:55 2021 GMT (365
 days)
Sign the certificate? [y/n]:
```

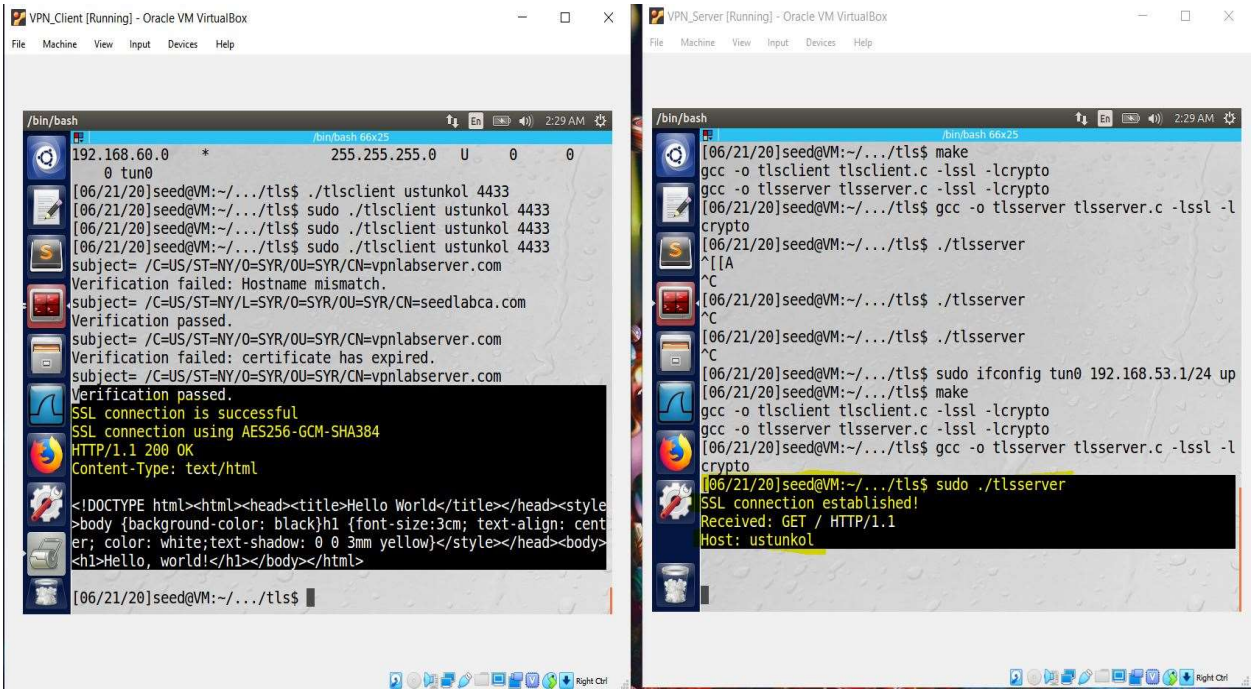
We then copy the root CA's certificate to the client and create a symbolic link to the file using its hash value:

```
Terminator                                                   7:00 PM
/bin/bash
/bin/bash 66x24
[06/18/20]seed@VM:~/.../ca_client$ openssl x509 -in GeoTrustGlobal
CA.pem -noout -subject_hash
2c543cd1
[06/18/20]seed@VM:~/.../ca_client$ ls -l
total 16
-rw-r--r-- 1 seed seed 1236 Mar 16  2018 2c543cd1.0
-rw-r--r-- 1 seed seed 1289 Mar 16  2018 9b58639a.0
-rw-r--r-- 1 seed seed 1289 Mar 16  2018 cacert.pem
-rw-r--r-- 1 seed seed 1236 Mar 16  2018 GeoTrustGlobalCA.pem
[06/18/20]seed@VM:~/.../ca_client$
```

We run the client program giving the correct domain and the port number which the server is using.

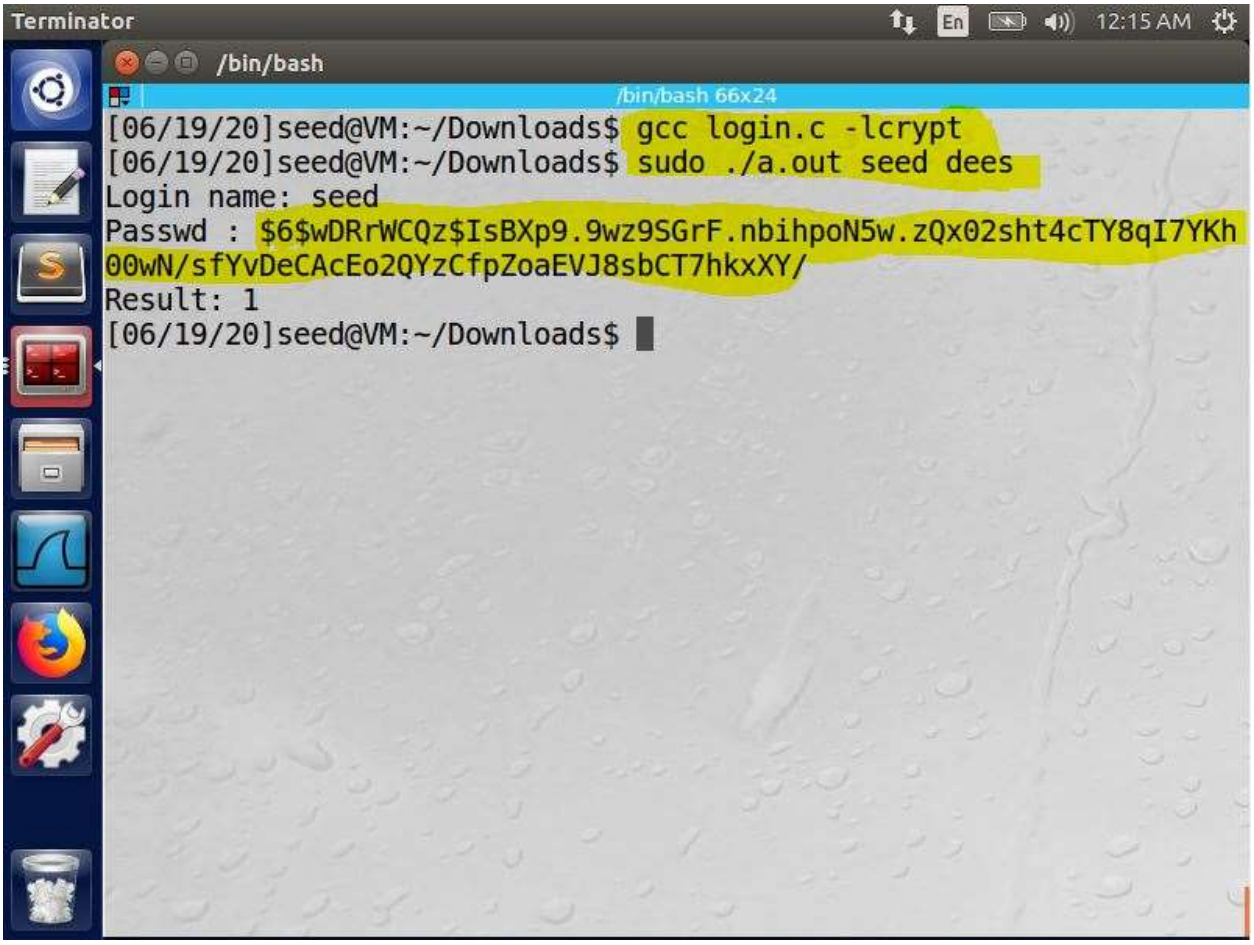
Our server's certificate and our CA's certificates are verified.

Also, the client is asked for authentication. Once he provides correct credentials, the requested data will be given. The username and password are checked in the server's shadow file.

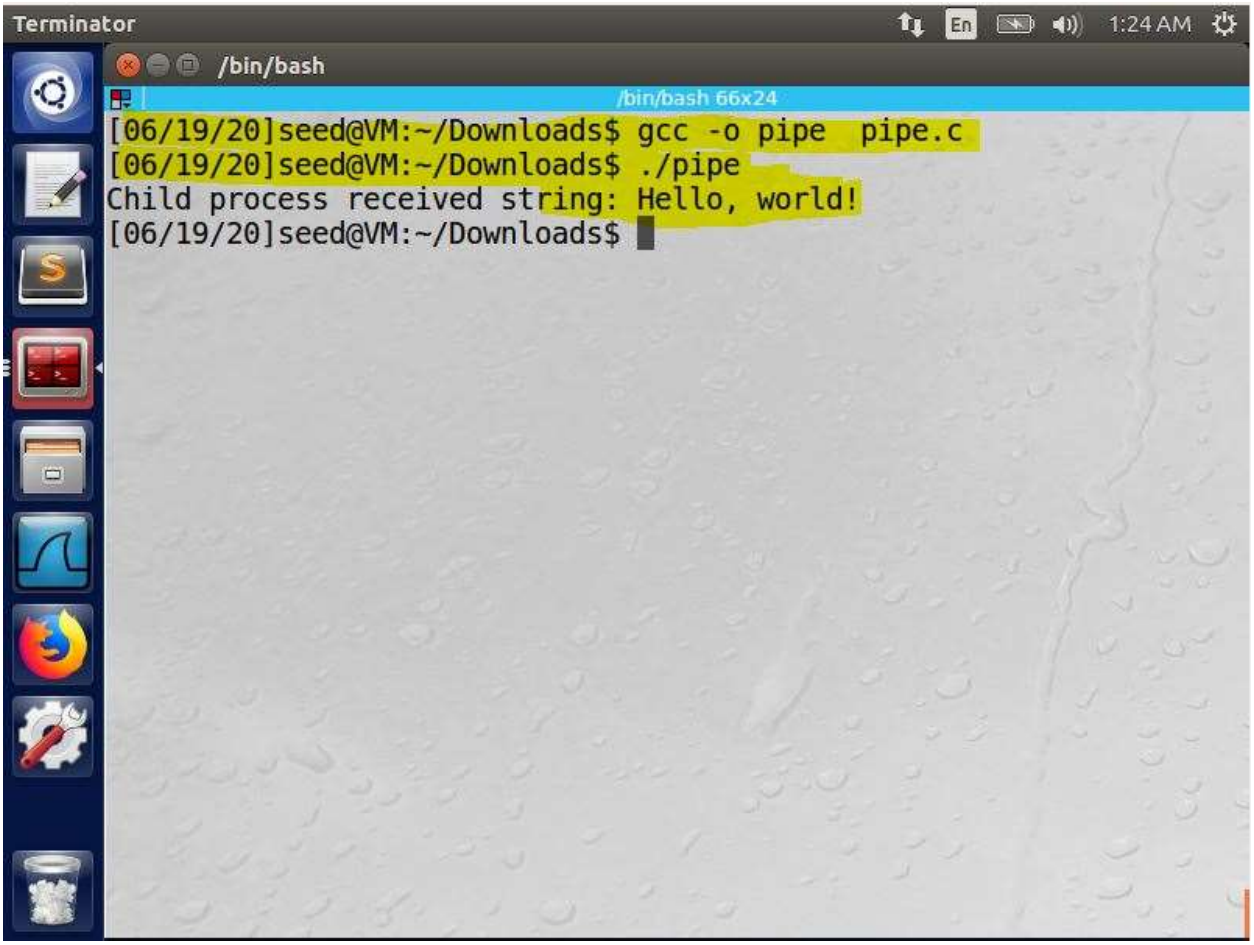


2.5 Task 5: Authenticating the VPN Client

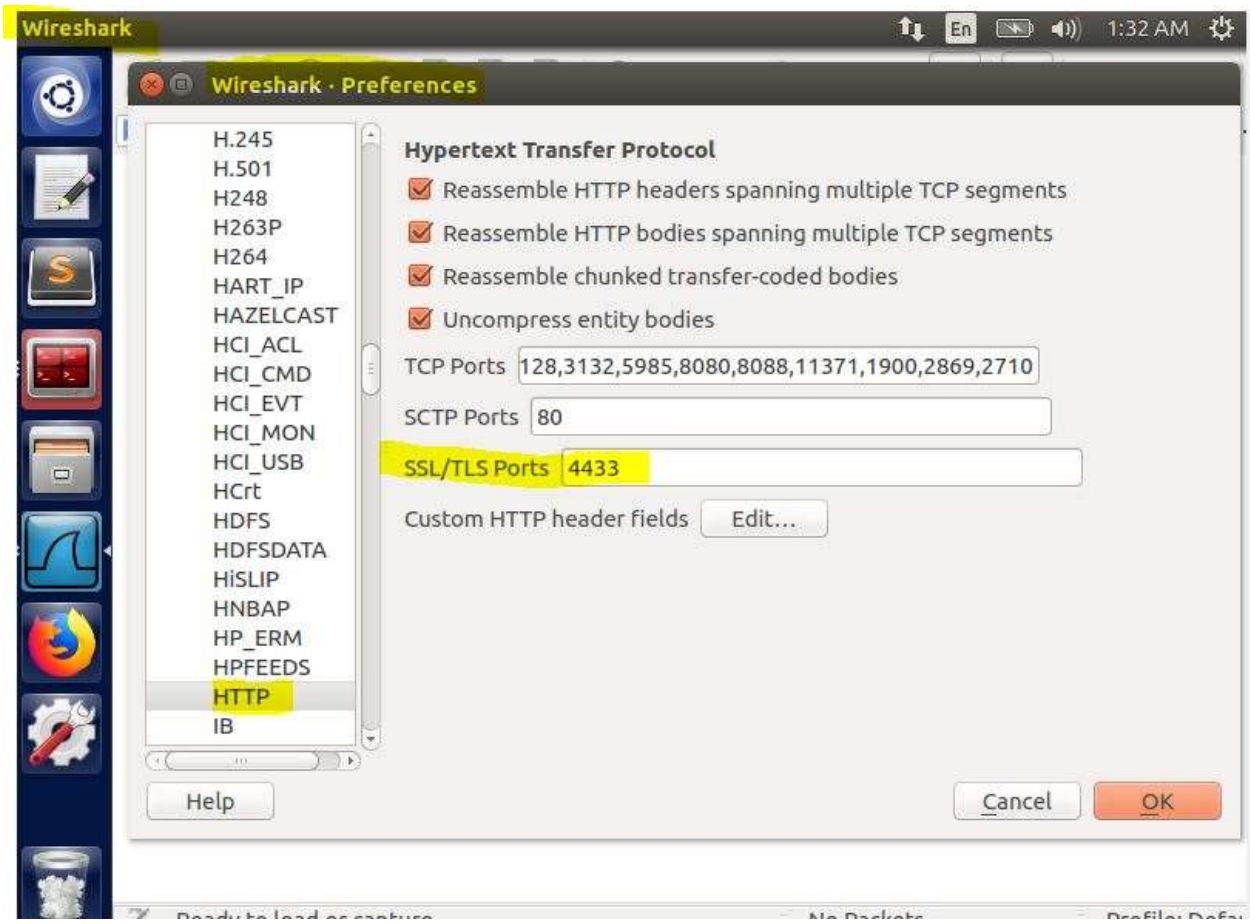
The server will break its connection with the user, and thus no tunnel will be established.



2.6 Task 6: Supporting Multiple Clients



3.1 Displaying TLS Traffic in Wireshark


Wireshark
En
1:32 AM
Wireshark · Preferences
H.245
H.501
H248
H263P
H264
HART_IP
HAZELCAST
HCI_ACL
HCI_CMD
HCI_EVT
HCI_MON
HCI_USB
HCrt
HDFS
HDFSDATA
HiSLIP
HNBAP
HP_ERM
HPFEEDS
HTTP
IB
Hypertext Transfer Protocol
Reassemble HTTP headers spanning multiple TCP segments
Reassemble HTTP bodies spanning multiple TCP segments
Reassemble chunked transfer-coded bodies
Uncompress entity bodies
TCP Ports 128,3132,5985,8080,8088,11371,1900,2869,2710
SCTP Ports 80
SSL/TLS Ports 4433
Custom HTTP header fields Edit...
Help
Cancel
OK

VPN_Client [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

*any

Apply a display filter ... <Ctrl-/>

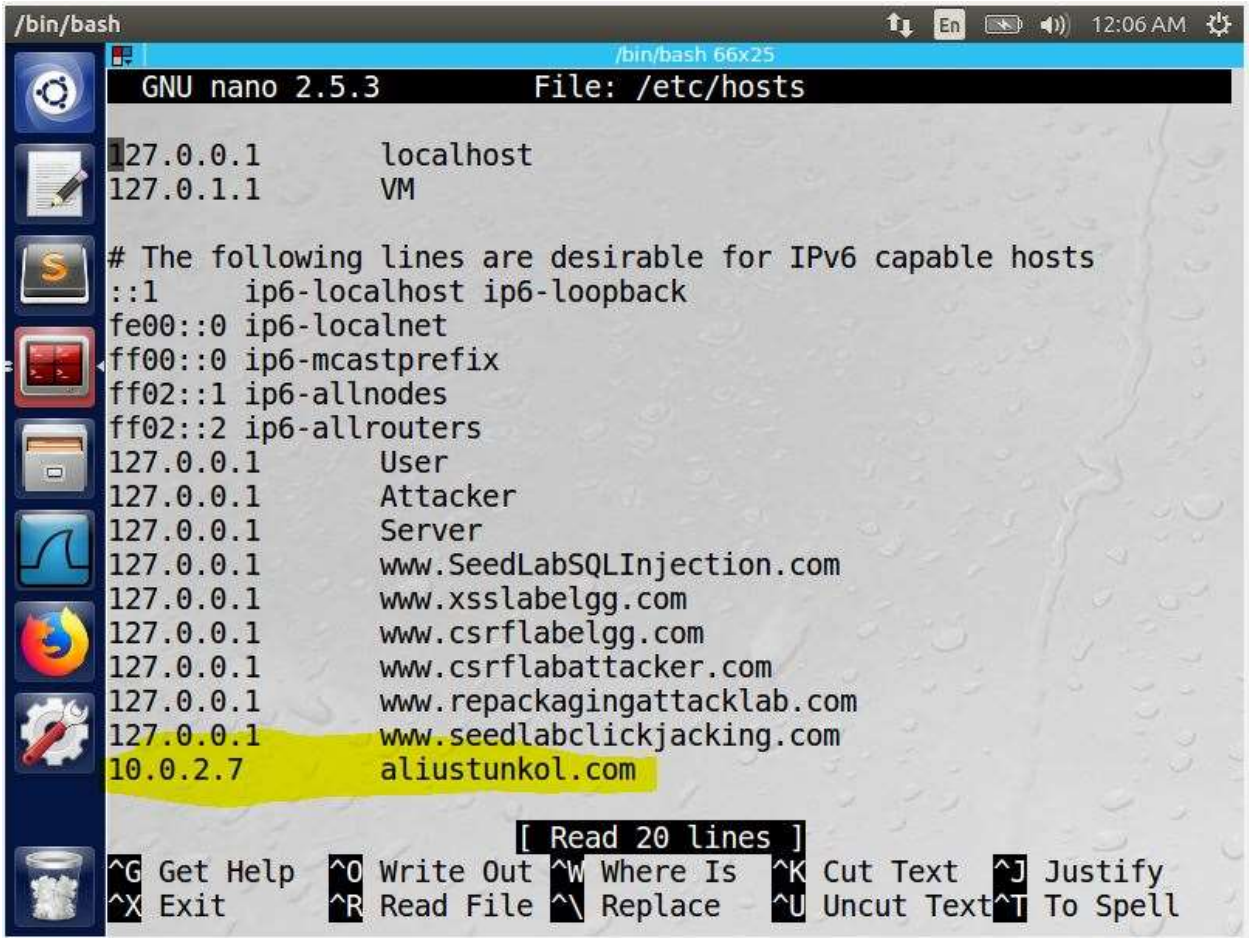
Packet list | Narrow & Wide | Case sensitive | Regular Expression | 443 | Find

No.	Time	Source	Destination	Protocol	Length	Info
1	2020-…	10.0.2.8	104.112.13.64	TCP	56	32814 → 443 [ACK] Se
2	2020-…	104.112.13.64	10.0.2.8	TCP	62	[TCP ACKed unseen se
3	2020-…	104.112.13.64	10.0.2.8	TCP	62	[TCP Dup ACK 2#1] [T
4	2020-…	10.0.2.8	104.18.20.226	TCP	56	49478 → 80 [ACK] Seq
5	2020-…	104.18.20.226	10.0.2.8	TCP	62	[TCP ACKed unseen se
6	2020-…	104.18.20.226	10.0.2.8	TCP	62	[TCP Dup ACK 5#1] [T
7	2020-…	10.0.2.8	66.218.87.15	TCP	679	36618 → 443 [PSH, AC
8	2020-…	10.0.2.8	66.218.87.15	TCP	94	36618 → 443 [PSH, AC
9	2020-…	66.218.87.15	10.0.2.8	TCP	62	443 → 36618 [ACK] Se
10	2020-…	66.218.87.15	10.0.2.8	TCP	62	[TCP Dup ACK 9#1] 44
11	2020-…	66.218.87.15	10.0.2.8	TCP	256	443 → 36618 [PSH, AC
12	2020-…	10.0.2.8	66.218.87.15	TCP	56	36618 → 443 [ACK] Se
13	2020-…	66.218.87.15	10.0.2.8	TCP	256	[TCP Spurious Retran

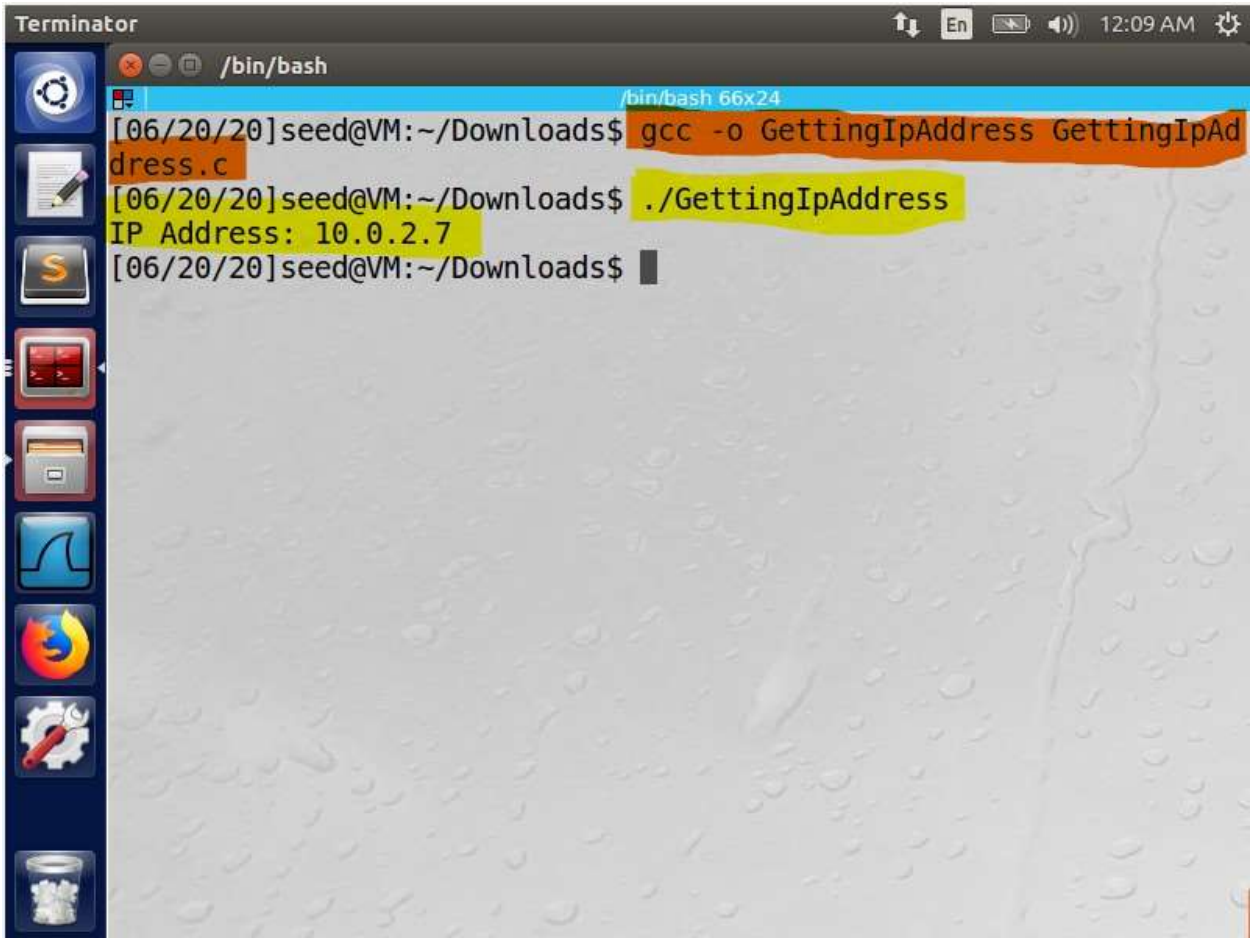
```
▸ Frame 2: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 104.112.13.64, Dst: 10.0.2.8
▾ Transmission Control Protocol, Src Port: 443, Dst Port: 32814, Seq: 5338671, Ac
    Source Port: 443
    Destination Port: 32814
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 5338671
```

wireshark_any_20200620184907_4qunn9 Packets: 5544 · Displayed: 5544 (100.0%)

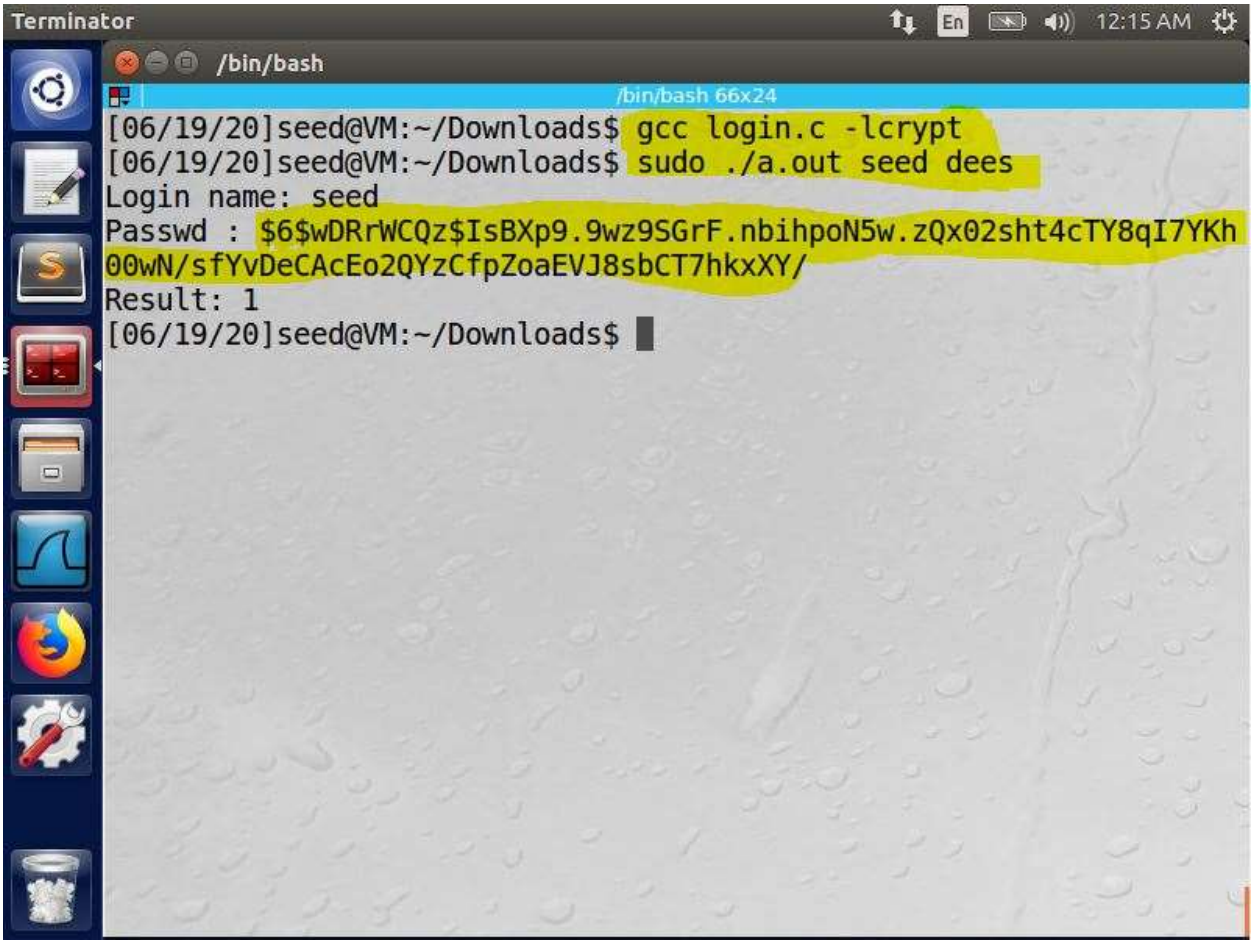
Right Ctrl



3.2 Getting IP Address from Hostname



3.3 Authentication Using the Shadow File



Conclusion

A Virtual Private Network (VPN) is used for creating a private scope of computer communications or providing a secure extension of a private network through an insecure network such as the Internet. VPN is a widely used in network security. VPN can be built upon IPsec or Secure Socket Layer (SSL). These are two fundamentally different approaches for building VPNs. In our work, we focused on the SSL-based VPNs which is often referred to as SSL VPNs. Designing and implementing the SSL VPNs exemplify a number of security principles and technologies, including crypto, integrity, authentication, key management, key exchange, and Public- Key Infrastructure (PKI). To achieve this goal, we implemented a simple SSL VPN for Linux Ubuntu operating system.

Used commands for the project;

[06/29/20]seed@VM:~$ history 250

```
  1  sudo apt-get install xfce4
  2  ping www.google.com
  3  ifconfig
  4  ping www.google.com
  5  ifconfig
  6  ping www.google.com
  7  ping 8.8.8.8
  8  ping www.syr.edu
  9  ping www.google.com
 10  sudo apt-get update
 11  sudo gedit /etc/sudoers
 12  sudo apt-get update
 13  sudo apt-get install xfce4
 14  ls
 15  ping www.google.com
 16  cd /
 17  find . -name 'xfce-teal.jpg'
 18  sudo find . -name 'xfce-teal.jpg'
 19  cd /usr/share/backgrounds/
 20  ls
 21  cp Black_hole_by_Marek_Koteluk_with_logo.jpg  xfce/
 22  ll
 23  sudo cp Black_hole_by_Marek_Koteluk_with_logo.jpg  xfce/
 24  ls
 25  cd xfce/
 26  ls
 27  sudo rm Black_hole_by_Marek_Koteluk_with_logo.jpg
```

```
28  ls
29  cd ..
30  ipconfig
31  clr
32  cls
33  ifconfig
34  ping 192.168.60.1
35  cd /etc
36  ls
37  sudo nano resolv.conf
38  service network.manager restart
39  service network.manager restartresolv.confexit
40  sudo service network.manager restart
41  sudo nano resolv.config
42  sudo service network.manager restart
43  ls
44  ifconfig
45  cd Desktop
46  ls
47  cd Desktop
48  cd Desktop
49  cd Desktop
50  ls
51  ls -al
52  cat article.txt
53  tr [:upper:] [:lower:] < article.txt > lowercase.txt
54  $ tr -cd '[a-z][\n][:space:]' < lowercase.txt > plaintext.txt
55  python
```

56 openssl enc -aes-128-cbc, -e -in plain.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

57 openssl enc -aes-128-cbc -e -in plain.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

58 openssl enc -aes-128-cbc -e -in plain.txt -out aes128.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

59 openssl enc -aes-128-cbc -e -in plain.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

60 openssl enc -aes-128-cbc -e -in plaintext.txt -out alicipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

61 openssl enc -aes-128-cfb8 -e -in plaintext.txt -out alicipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

62 openssl enc -aes-192-cbc -e -in plaintext.txt -out alicipher3.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

63 openssl enc -des-ede3-cbc -e -in plaintext.txt -out alicipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

64 openssl enc -aes-256-ecb -e -in plaintext.txt -out alicipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

65 openssl enc -aes-128-cfb -e -in plaintext.txt -out alicipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

66 ls -1

67 head -c 54 pic_original.bmp > header

68 tail -c +55 pic_original.bmp > body

69 cat header body > new.bmp

70 echo -n "1234567890" > f2.txt

71 echo -n "1234567890123456" > f3.txt

72 openssl enc - -aes-128-cbc -e -in f1.txt -out p1.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708

73 openssl enc - -aes-128-cbc -e -in f1.txt -out p1.txt -K 00112233445566778889aabbccddeeff

74 -iv 0102030405060708

75 openssl enc - -aes-128-cbc -e -in f1.txt -out p1.txt -K 00112233445566778889aabbccddeeff

76 openssl enc - -aes-128-cbc -e -in f1.txt -out p1.txt -K 00112233445566778889aabbccddeeff

77 -iv 0102030405060708

```
78  openssl enc -aes-128-cbc -e -in f1.txt -out p1.txt -K 00112233445566778889aabbccddeeff
79  openssl enc -aes-128-cbc -e -in f1.txt -out p1.txt > -K 00112233445566778889aabbccddeeff
80  -iv 010203040506070
81  openssl enc -aes-128-cbc -e -in f1.txt -out f1cipher.txt -K 00112233445566778889aabbccddeeff
82  -iv 0102030405060708
83  openssl enc -aes-128-cbc -e -in f2.txt -out f2cipher.txt -K 00112233445566778889aabbccddeeff
84  -iv 0102030405060708
85  openssl enc -aes-128-cbc -e -in f3.txt -out f3cipher.txt -K 00112233445566778889aabbccddeeff
86  -iv 0102030405060708
87  hexdump -C f1cipher.txt
88  ls
89  openssl enc -aes-128-ecb -e -in f1.txt -out f1cipher.txt -K 00112233445566778889aabbccddeeff
90  -iv 0102030405060708
91  openssl enc -aes-128-ecb -e -in f2.txt -out f2cipher.txt -K 00112233445566778889aabbccddeeff
92  -iv 0102030405060708
93  openssl enc -aes-128-ecb -e -in f3.txt -out f3cipher.txt -K 00112233445566778889aabbccddeeff
94  -iv 0102030405060708
95  hexdump -C p1.txt
96  hexdump -C f1cipher.txt
97  hexdump -C f2cipher.txt
98  hexdump -C f3cipher.txt
99  echo -n
"1234567890123456789012345678901234567890123456789012345678901234567890123
4567890123456789012345678901234567890123456789012345678901234567890123456
7890123456789012345678901234567890123456789012345678901234567890123456789
0123456789012345678901234567890123456789012345678901234567890123456789012
3456789012345678901234567890123456789012345678901234567890123456789012345
6789012345678901234567890123456789012345678901234567890123456789012345678
9012345678901234567890123456789012345678901234567890123456789012345678901
2345678901234567890123456789012345678901234567890123456789012345678901234
5678901234567890123456789012345678901234567890123456789012345678901234567
8901234567890123456789012345678901234567890123456789012345678901234567890
1234567890123456789012345678901234567890123456789012345678901234567890123
```

4567890123456789012345678901234567890123456789012345678901234567890123456
7890" > p1000.txt

100 openssl enc -aes-128-ecb -e -in p1000.txt -out p1000cipher.txt -K
00112233445566778889aabbccddeeff

101 -iv 0102030405060708

102 cat p1000cipher.txt

103 openssl enc -aes-128-ofb -e -in plaintext6.txt -out plantext6cipher.bin -K
00112233445566778889aabbccddeeff -iv 0102030405060708

104 cat plaintext6cipher.bin

105 cat plantext6cipher.bin

106 openssl enc -aes-128-ofb -d -in plaintext6.txt -out plantext6cipher.bin -K
00112233445566778889aabbccddeeff -iv 0102030405060708

107 openssl enc -aes-128-ofb -e -in plaintext6.txt -out plantext6cipher.bin -K
00112233445566778889aabbccddeeff -iv 0102030405060708

108 cat plaintext6cipher.bin

109 openssl enc -aes-128-ofb -d -in plaintext6.txt -out plantext6cipher.txt -K
00112233445566778889aabbccddeeff -iv 0102030405060708

110 cat plaintext6cipher.bin

111 openssl enc -aes-128-ofb -e -in plaintext6.txt -out plantext6cipher.bin -K
00112233445566778889aabbccddeeff -iv 0102030405060708

112 cat plaintext6cipher.bin

113 cat plantext6cipher.bin

114 openssl enc -aes-128-ofb -d -in plaintext6.txt -out plantext6cipher.txt -K
00112233445566778889aabbccddeeff -iv 0102030405060708

115 openssl enc -aes-128-ofb -d -in plantext6cipher.bin -out plantext6cipher.txt -K
00112233445566778889aabbccddeeff -iv 0102030405060708

116 openssl enc -aes-128-cbc -e -in plaintext6.txt -out plaintext6cbc.txt -K
00112233445566778889aabbccddeeff -iv 0102030405060708

117 exit

118 assdfgdgdghghdf

119 gfghfhyfhf

120 geyerjufrj

```
121  ping 192.168.60.101
122  cd Downloads
123  ls
124  exit
125  ifconfig
126  ping 192.168.1
127  ping 192.168.10.1
128  ping 192.168.60.1
129  sudo route add –net 192.168.53.0/24 gw 192.168.60.1 enp0s9
130  ping 192.168.60.1
131  ping 192.168.101
132  ifconfig
133  cd VPN
134  cd Downloads
135  cd VPN
136  cd vpn
137  exit
138  cd Downloads
139  ls
140  unzip tls.zip
141  ls
142  cd CA
143  ls
144  cd cert_server
145  ls
146  openssl genrsa -aes128 -out server.key 1024
147  openssl rsa -in server.key -text
148  openssl req -new -key server.key -out server.csr -config openssl.cnf
149  $ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
```

```
150  cd ..
151  $ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key $ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key \
152  $ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key config openssl.cnf
153  $ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
154  cd crt_server
155  cd cert_server
156  openssl req -new -key server.key -out server.csr -config openssl.cnf
157  openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key \-config openssl.cnf
158  cd Desktop
159  cd CA
160  cd PKI
161  openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf
162  clear
163  openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf
164  clear
165  openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf
166  openssl genrsa -aes128 -out server.key 1024
167  openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
168  sudo nano -w/etc/hosts
169  sudo nano -w /etc/hosts
170  cp server.key server.pem
171  cat server.crt >> server.pem
172  openssl s_server -cert server.pem -www
173  sudo nano -w /etc/hosts
174  cp server.key server.pem
175  cat server.crt >> server.pem
176  openssl s_server -cert server.pem -www
177  sudo nano -w /etc/hosts
```

```
178  cp server.key server.pem
179  cat server.crt >> server.pem
180  openssl s_server -cert server.pem -www
181  sudo nano ca.crt
182  openssl s_server -cert server.pem -www
183  sudo nano ca.crt
184  openssl s_server -cert server.pem -www
185  cp server.key server.pem
186  cat server.crt >> server.pem
187  openssl s_server -cert server.pem -www
188  sudo nano ca.crt
189  cp server.key server.pem
190  cat server.crt >> server.pem
191  openssl s_server -cert server.pem -www
192  sudo nano 000-default.conf
193  sudo nano default-ssl.conf
194  sudo apachet1 configtest
195  sudo nano default-ssl.conf
196  sudo nano 000-default.conf
197  sudo apachet1 configtest
198  sudo nano default-ssl.conf
199  sudo nano 000-default.conf
200  sudo apachet1 configtest
201  sudo apachetl configtest
202  sudo apachectl configtest
203  sudo nano 000-default.conf
204  sudo nano default-ssl.conf
205  sudo nano ca.crt
206  sudo nano -w /etc/hosts
```

```
207  cp server.key server.pem
208  cat server.crt >> server.pem
209  openssl s_server -cert server.pem -www
210  sudo nano -w /etc/hosts
211  cp server.key server.pem
212  cat server.crt >> server.pem
213  openssl s_server -cert server.pem -www
214  sudo nano 000-default.conf
215  sudo nano default-ssl.conf
216  sudo apachectl configtest
217  sudo
218  sudo nano 000-default.conf
219  sudo nano default-ssl.conf
220  sudo apachectl configtest
221  sudo service apache2 restart
222  sudo apachectl configtest
223  sudo a2enmod ssl
224  sudo service apache2 restart
225  sudo nano 000-default.conf
226  sudo apachectl configtest
227  sudo a2enmod ssl
228  sudo service apache2 restart
229  sudo nano 000-default.conf
230  sudo nano default-ssl.conf
231  sudo nano -w /etc/hosts
232  cd /Desktop/CA/PKI
233  ls
234  cd demoCA
235  ls
```

```
236  cat indext.txt
237  cat index.txt
238  cd ..
239  sudo a2ensite default-ssl
240  service apache2 reload
241  sudo service apache2 restart
242  sudo vi /etc/hosts
243  sudo apt-get update
244  sudo apt-get install apache2
245  sudo mkdir -p /var/www/example.com/public_html
246  sudo mkdir -p /var/www/test.com/public_html
247  sudo chown -R $USER:$USER /var/www/example.com/public_html
248  sudo chown -R $USER:$USER /var/www/test.com/public_html
249  sudo chmod -R 755 /var/www
250  nano /var/www/example.com/public_html/index.html
251  cp /var/www/example.com/public_html/index.html /var/www/test.com/public_html/index.html
252  nano /var/www/test.com/public_html/index.html
253  sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-
available/example.com.conf
254  sudo nano /etc/apache2/sites-available/example.com.conf
255  sudo cp /etc/apache2/sites-available/example.com.conf /etc/apache2/sites-
available/test.com.conf
256  sudo nano /etc/apache2/sites-available/test.com.conf
257  sudo a2ensite example.com.conf
258  sudo a2ensite test.com.conf
259  sudo a2dissite 000-default.conf
260  sudo systemctl restart apache2
261  sudo service apache2 restart
262  sudo nano /etc/hosts
```

```
 263  ifconfig
 264  sudo nano /etc/hosts
 265  nano /var/www/example.com/public_html/index.html
 266  sudo nano /etc/apache2/sites-available/example.com.conf
 267  sudo nano /etc/apache2/sites-available/test.com.conf
 268  sudo nano /etc/hosts
 269  sudo nano /etc/apache2/sites-available/test.com.conf
 270  sudo nano /etc/hosts
 271  sudo nano /etc/apache2/sites-available/test.com.conf
 272  sudo nano /etc/apache2/sites-available/example.com.conf
 273  sudo nano /etc/hosts
 274  sudo nano server.pem
 275  sudo nano /etc/apache2/sites-available/test.com.conf
 276  nano /var/www/example.com/public_html/index.html
 277  cp /var/www/example.com/public_html/index.html /var/www/test.com/public_html/index.html
 278  history 500
[06/29/20]seed@VM:~$
```