



University
of Windsor

Lab 6

Course: Networking and Data Security

COMP8677-1-R-2023F

Professor: Dr. Shaoquan Jiang

Prepared by

Harshil Hitendrabhai Panchal (110096129)

Due date: November 2, 2023

Part I (create server public-key and its certificate)

In TLS, the server must have a RSA public key. To guarantee this public key is owned by this server, it must be certified by an authority. In reality, this is done by some special company such as VeriSign. In our experiment, we ourselves will play the role of an authority. This authority will generate its own public/private key and certificate (which is a self-signed signature), just as what a real root CA has done. It then will generate the certificate for the TLS server. The task can be done by following the following procedure.

1. Copy `/usr/lib/ssl/openssl.cnf` to your current working directory and make the following change to this file:

"policy = policy_match" to "policy = policy_anything"

`/*this allows the CA to generate certificate for more potential users. */`

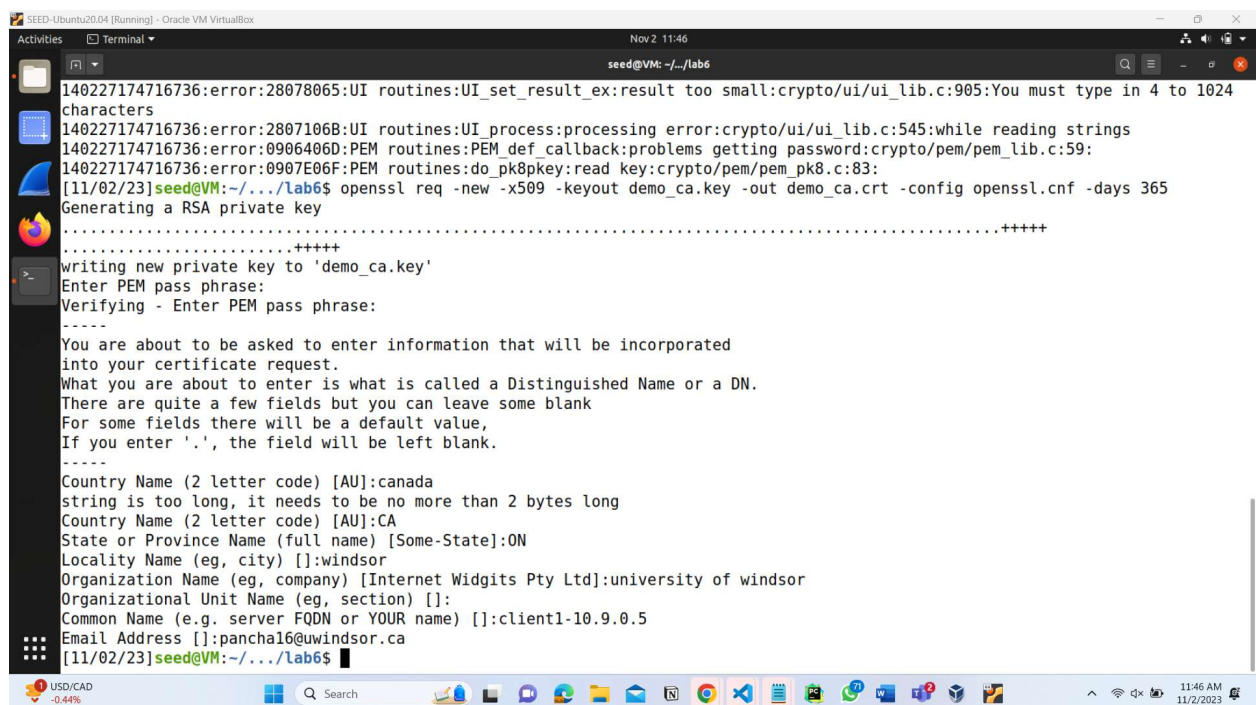
2. Create a new directory `demoCA` in the current directory. Then, do the following.
 - a. Create new directories `certs`, `crl` and `newcerts` in `demoCA` and empty files `index.txt` and `serial`. Put a single serial number (.e.g., 1000) in the file `serial`.

(do the following steps outside demoCA)

- b. Generate a self-signed certificate for our certificate authority (CA).

```
$openssl req -new -x509 -keyout demo_ca.key -out demo_ca.crt -config openssl.cnf -days 365
```

`/* demo_ca.key has private RSA key for CA; demo_ca.crt is its self-signed certificate with 365 days validity. */`

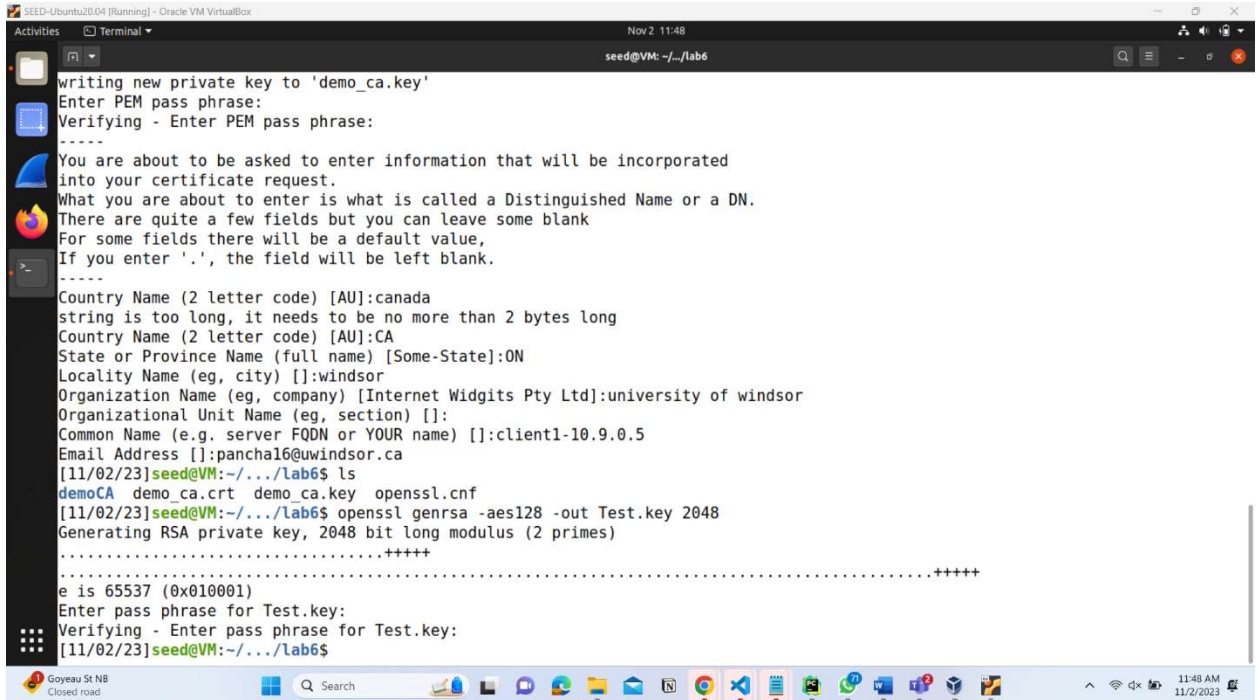


```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
Nov 2 11:46
seed@VM: ~/../Lab6
140227174716736:error:28078065:UI routines:UI_set_result_ex:result too small:crypto/ui/ui_lib.c:905:You must type in 4 to 1024
characters
140227174716736:error:2807106B:UI routines:UI_process:processing error:crypto/ui/ui_lib.c:545:while reading strings
140227174716736:error:0906406D:PEM routines:PEM_def_callback:problems getting password:crypto/pem/pem_lib.c:59:
140227174716736:error:0907E06F:PEM routines:do_pk8pkey:read key:crypto/pem/pem_pk8.c:83:
[11/02/23]seed@VM:~/../Lab6$ openssl req -new -x509 -keyout demo_ca.key -out demo_ca.crt -config openssl.cnf -days 365
Generating a RSA private key
.....+++++
.....+++++
writing new private key to 'demo_ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
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]:canada
string is too long, it needs to be no more than 2 bytes long
Country Name (2 letter code) [AU]:CA
State or Province Name (full name) [Some-State]:ON
Locality Name (eg, city) []:windsor
Organization Name (eg, company) [Internet Widgits Pty Ltd]:university of windsor
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:client1-10.9.0.5
Email Address []:panchal16@uwindsor.ca
[11/02/23]seed@VM:~/../Lab6$
```

- c. Create a certificate for our test TLS server, signed by our authority's key demo_ca.key.
- i. Generate a RSA private key for TLS server.

```
$ openssl genrsa -aes128 -out Test.key 2048
```

```
/*To view the file, $openssl rsa -in Test.key -noout -text */
```



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
Nov 2 11:48
seed@VM: ~/.../lab6

writing new private key to 'demo_ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
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]:canada
string is too long, it needs to be no more than 2 bytes long
Country Name (2 letter code) [AU]:CA
State or Province Name (full name) [Some-State]:ON
Locality Name (eg, city) []:windsor
Organization Name (eg, company) [Internet Widgits Pty Ltd]:university of windsor
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:client1-10.9.0.5
Email Address []:panchal6@uwindsor.ca
[11/02/23]seed@VM:~/.../lab6$ ls
demoCA  demo_ca.crt  demo_ca.key  openssl.cnf
[11/02/23]seed@VM:~/.../lab6$ openssl genrsa -aes128 -out Test.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
Enter pass phrase for Test.key:
Verifying - Enter pass phrase for Test.key:
[11/02/23]seed@VM:~/.../lab6$
```

- ii. Generate a certificate signing request:

```
$ openssl req -new -key Test.key -out Test.csr -config openssl.cnf
```

```
/* this generates a certificate request so that CA can sign a cert for TLS server:
common name uses the container name (e.g., client1-10.9.0.5). */
```

```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
Nov 2 11:52
seed@VM: ~/../lab6

demoCA demo_ca.crt demo_ca.key openssl.cnf
[11/02/23]seed@VM:~/../Lab6$ openssl genrsa -aes128 -out Test.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
Enter pass phrase for Test.key:
Verifying - Enter pass phrase for Test.key:
[11/02/23]seed@VM:~/../Lab6$ openssl req -new -key Test.key -out Test.csr -config openssl.cnf
Enter pass phrase for Test.key:
You are about to be asked to enter information that will be incorporated
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]:CA
State or Province Name (full name) [Some-State]:ON
Locality Name (eg, city) [:]windsor
Organization Name (eg, company) [Internet Widgits Pty Ltd]:university of windsor
Organizational Unit Name (eg, section) [:]
Common Name (e.g. server FQDN or YOUR name) [:]client1-10.9.0.5
Email Address [:]panchal6@uwindsor.ca

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password [:]harshil
An optional company name [:]
[11/02/23]seed@VM:~/../Lab6$
```

iii. Generate the certificate for TLS server:

```
$ openssl ca -in Test.csr -out Test.crt -cert demo_ca.crt -keyfile demo_ca.key -
config openssl.cnf
```

/*Test.crt is the certificate for TLS server */

```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
Nov 2 11:55
seed@VM: ~/../lab6

140379267425600:error:0907B00D:PEM routines:PEM_read_bio_PrivateKey:ASN1 lib:crypto/pem/pem_pkey.c:88:
[11/02/23]seed@VM:~/../Lab6$ openssl ca -in Test.csr -out Test.crt -cert demo_ca.crt -keyfile demo_ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for demo_ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 4096 (0x1000)
    Validity
        Not Before: Nov  2 15:55:15 2023 GMT
        Not After : Nov  1 15:55:15 2024 GMT
    Subject:
        countryName           = CA
        stateOrProvinceName   = ON
        localityName          = windsor
        organizationName      = university of windsor
        commonName            = client1-10.9.0.5
        emailAddress          = panchal6@uwindsor.ca
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        Netscape Comment:
            OpenSSL Generated Certificate
        X509v3 Subject Key Identifier:
            FA:D7:2B:80:A6:1A:33:EB:16:9D:87:90:C9:E1:AC:49:E7:E7:7A:FC
        X509v3 Authority Key Identifier:
            keyid:9C:C0:EB:B1:2F:5C:54:C7:6C:83:5A:81:DC:86:A4:6A:56:D1:1C:01

Certificate is to be certified until Nov  1 15:55:15 2024 GMT (365 days)
Sign the certificate? [y/n]:
```

```

Certificate Details:
Serial Number: 4096 (0x1000)
Validity
  Not Before: Nov  2 15:55:15 2023 GMT
  Not After : Nov  1 15:55:15 2024 GMT
Subject:
  countryName      = CA
  stateOrProvinceName = ON
  localityName     = windsor
  organizationName  = university of windsor
  commonName       = client1-10.9.0.5
  emailAddress      = pancha16@uwindsor.ca
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  Netscape Comment:
    OpenSSL Generated Certificate
  X509v3 Subject Key Identifier:
    FA:D7:2B:80:A6:1A:33:EB:16:9D:87:90:C9:E1:AC:49:E7:E7:7A:FC
  X509v3 Authority Key Identifier:
    keyid:9C:C0:EB:B1:2F:5C:54:C7:6C:83:5A:81:DC:86:A4:6A:56:D1:1C:01

Certificate is to be certified until Nov  1 15:55:15 2024 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]:y
Write out database with 1 new entries
Data Base Updated
[11/02/23]seed@VM:~/lab6$

```

/*the following steps operated in volumes might need sudo; volumes is the folder created by your docker-compose (using the newly provided yml file) */

- iv. Copy your certificate Test.crt and Test.key to a folder certS in the shared folder volumes (your server program such as server.py will send Test.crt to client and use Test.key to decrypt or sign a document).

```

1 out of 1 certificate requests certified, commit? [y/n]:y
Write out database with 1 new entries
Data Base Updated
[11/02/23]seed@VM:~/lab6$ docker cp demo_ca.crt 793de2de45ef:/volumes/certS
[11/02/23]seed@VM:~/lab6$ docker cp demo_ca.key 793de2de45ef:/volumes/certS
[11/02/23]seed@VM:~/lab6$ docker cp Test.crt 793de2de45ef:/volumes/certS
[11/02/23]seed@VM:~/lab6$ docker cp Test.key 793de2de45ef:/volumes/certS
[11/02/23]seed@VM:~/lab6$

```

- v. Copy demo_ca.crt to folder (such as certC) in the shared folder volumes. Later your client program (such as client.py) will use this demo_ca.crt to verify if the server certificate Test.crt is


```
seed@seed:~$ docker cp [OPTIONS] CONTAINER:SRC_PATH|- DEST_PATH|-
Usage: docker cp [OPTIONS] CONTAINER:SRC_PATH|- CONTAINER:DEST_PATH

1 out of 1 certificate requests certified, commit? [y/n]
Write out database with file descriptor 3
Data Base Updated
[11/02/23]seed@VM:~/..$ ls
demoCA  demo_ca.crt  demo_ca.key  client.py
[11/02/23]seed@VM:~/..$ rm Test.crt
root@793de2de45ef:/# rm Test.crt
root@793de2de45ef:/# rm Test.key
root@793de2de45ef:/# rm demo_ca.crt
root@793de2de45ef:/# rm demo_ca.key
[11/02/23]seed@VM:~/..$ ls
bin  dev  etc  lib  lib32  lib64  media  mnt  opt  proc  run  srv  tmp  var  volumes
[11/02/23]seed@VM:~/..$ cd volumes
root@793de2de45ef:/volumes# ls
certS
root@793de2de45ef:/volumes# cd certS/
root@793de2de45ef:/volumes/certS# ls
Test.crt  Test.key
root@793de2de45ef:/volumes/certS# ls
Test.crt  Test.key  demo_ca.crt
root@793de2de45ef:/volumes/certS#
[11/02/23]seed@VM:~/..$ docker cp [OPTIONS] CONTAINER:SRC_PATH|- DEST_PATH|-
Usage: docker cp [OPTIONS] CONTAINER:SRC_PATH|- CONTAINER:DEST_PATH

Copy files/folders between a container and the local filesystem
[11/02/23]seed@VM:~/..$ docker cp Test.crt 793de2de45ef:/volumes/certS
[11/02/23]seed@VM:~/..$ docker cp Test.key 793de2de45ef:/volumes/certS
[11/02/23]seed@VM:~/..$ docker cp demo 793de2de45ef:/volumes/certS
demoCA/  demo_ca.crt  demo_ca.key
[11/02/23]seed@VM:~/..$ docker cp demo_ca.crt 793de2de45ef:/volumes/certS
[11/02/23]seed@VM:~/..$
```

```
8f838d2e /*this is the hash value; your case might be different*/
```

The image shows a dual-monitor setup of a Kali Linux desktop environment. The left monitor displays a terminal window with the following content:

```
seed@VM: ~/ - /lab6
db:94:e0:d4:d0:29:76:59:ea:84:bb:4d:84:8e:8b:01:8f:60:
a4:b0:84:4a:db:61:f0:ea:40:64:83:a5:78:6d:45:7b:df:45:
83:30:3e:af:35:96:22:f8:1a:05:39:c5:5b:c7:4f:7c:05:84:
fa:2e:08:6a:39:97:29:3c:1f:95:98:9d:8f:c8:64:7c:bf:37:
b4:65:40:79:c1:4e:c3:0d:05:d2:c1:37:02:ef:04:61:c2:08:
20:fb:6a:1d:a1:c9:ca:80:97:74:c9:07:65:ca:17:a5:ac:da:
36:38:bb:d2:6e:d5:c0:8b:40:49:5c:78:66:fa:5b:e5:fb:85:
7e:b7:ef:0d:fa:e6:96:b1:02:75:a0:f7:6d:8e:4b:ed:7c:e5:
07:b4:de:fc:44:eb:3a:b7:61:17:56:45:a0:5b:f6:16:f2:db:
04:a8:99:a0:e2:b8:c7:5d:a9:80:4c:16:8a:a7:10:25:0e:72:
9a:b3:ae:38:f6:42:4e:94:d4:13:4a:99:ab:47:ea:06:69:03:
9b:8c:45:3b:9f:8d:e0:52:87:20:d4:48:c2:89:55:f6:33:52:
28:aa:0c:b2:2f:0e:43:dd:47:f5:53:67:4a:25:05:4b:f8:
66:f8:c1:4c
root@b73202694eed:/volumes/certC# openssl x509 -in demo_ca.crt
-noout -subject_hash
95340a82
root@b73202694eed:/volumes/certC# ln -s demo_ca.crt 95340a82.0
root@b73202694eed:/volumes/certC# ls
95340a82.0 demo_ca.crt demo_ca.key
root@b73202694eed:/volumes/certC# cd ../certs
root@b73202694eed:/volumes/certS# ls
Test.crt client.py demo_ca.key
Test.key demo_ca.crt server.py
root@b73202694eed:/volumes/certS# python3 server.py
Enter PEM pass phrase:
Traceback (most recent call last):
  File "server.py", line 15, in <module>
    context.load_cert_chain(SERVER_CERT, SERVER_PRIVATE)
OSError: [Errno 22] Invalid argument
```

The right monitor displays a web browser window showing the output of a Python script. The output is as follows:

```
File "server.py", line 23, in <module>
newsock, fromaddr = sock.accept()
File "/usr/lib/python3.8/socket.py", line 292, in accept
fd, addr = self._accept()
KeyboardInterrupt

root@93de2de45ef:/volumes/certS# python3 server.py
Enter PEM pass phrase:
TCP connect
TLS connection fails
^CTraceback (most recent call last):
  File "server.py", line 23, in <module>
    newsock, fromaddr = sock.accept()
File "/usr/lib/python3.8/socket.py", line 292, in accept
fd, addr = self._accept()
KeyboardInterrupt

root@93de2de45ef:/volumes/certS# ps
PID TTY TIME CMD
24 pts/1 00:00:00 bash
47 pts/1 00:00:00 ps
root@93de2de45ef:/volumes/certS# python3 server.py
Enter PEM pass phrase:
TCP connect
TLS connection fails
TCP connect
TLS connection established
"Request: b'GET / HTTP/1.0\r\nHost: client1-10.9.0.5\r\n\r\n"
\r\n"
```

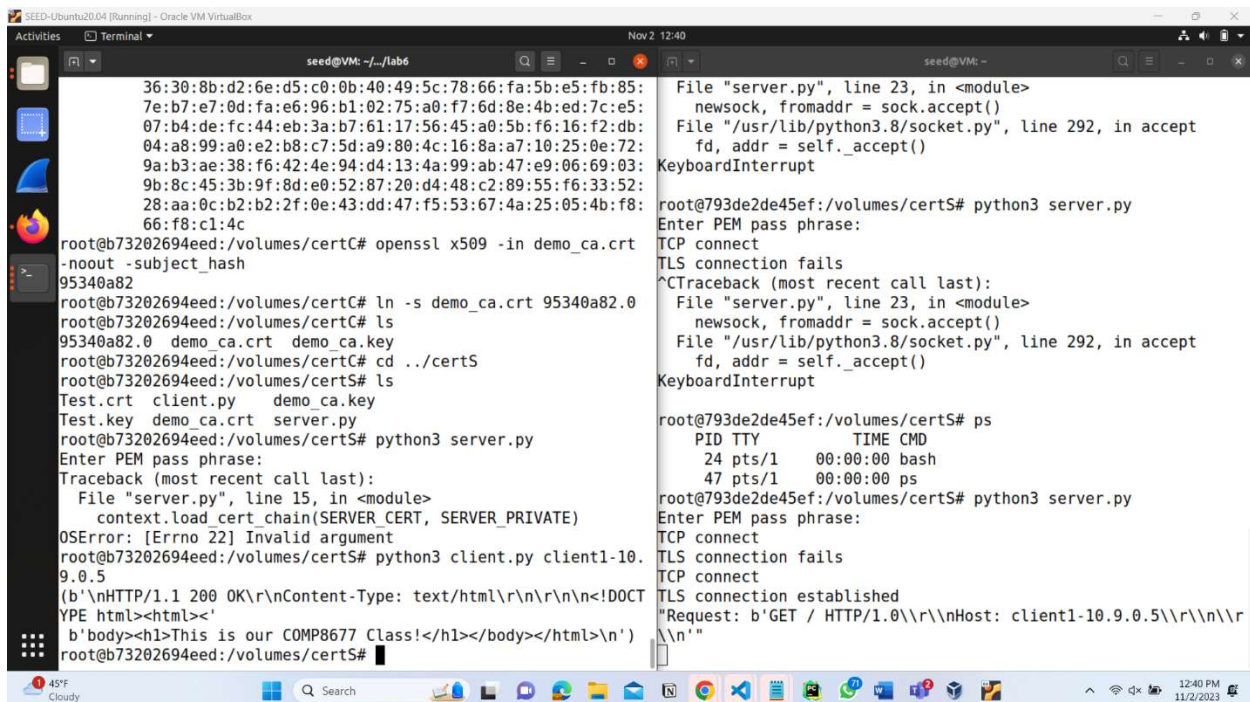
Part II. (TLS Client and Server Communication)

In this part, you need to create a TLS server and TLS client with some functions. Toward this, you are encouraged to run the provided TLS server and client to get familiar with how TLS client and server can be connected.

1. Step 1. Make sure that Part I has been done.
2. Step 2. Use the provided client.py and server.py. Modify the certificate directory cadir in server.py to make sure it is the directory certS for the server certificate and server private key directory (in Part I). Also modify the certificate directory cadir in client.py to make sure that it is the directory certC of the CA's certificate demoCA.crt.
3. Step 3. Run `$sudo python3 server.py` on the server VM (should be the same as in Test.crt)
4. Step 4. Run client.py with server container name (e.g., client1-10.0.2.5):

\$ python3 client.py client1-10.9.0.5

Then, if you receive the response from server, then you are done.



```
seed@VM: ~/lab6
36:30:8b:d2:6e:d5:c0:0b:40:49:5c:78:66:fa:5b:e5:fb:85:
7e:b7:e7:0d:fa:e6:96:b1:02:75:a0:f7:6d:8e:4b:ed:7c:e5:
07:b4:de:fc:44:eb:3a:b7:61:17:56:45:a0:5b:f6:16:f2:db:
04:a8:99:a0:e2:b8:c7:5d:a9:80:4c:16:8a:a7:10:25:0e:72:
9a:b3:ae:38:f6:42:4e:94:d4:13:4a:99:ab:47:e9:06:69:03:
9b:8c:45:3b:9f:8d:e0:52:87:20:d4:48:c2:89:55:f6:33:52:
28:aa:0c:b2:2f:0e:43:dd:47:f5:53:67:4a:25:05:4b:f8:
66:f8:c1:4c
root@b73202694eed:/volumes/certC# openssl x509 -in demo_ca.crt
-noout -subject_hash
95340a82
root@b73202694eed:/volumes/certC# ln -s demo_ca.crt 95340a82.0
root@b73202694eed:/volumes/certC# ls
95340a82.0 demo_ca.crt demo_ca.key
root@b73202694eed:/volumes/certC# cd ../certS
root@b73202694eed:/volumes/certS# ls
Test.crt client.py demo_ca.key
Test.key demo_ca.crt server.py
root@b73202694eed:/volumes/certS# python3 server.py
Enter PEM pass phrase:
TCP connect
TLS connection fails
^CTraceback (most recent call last):
  File "server.py", line 23, in <module>
    newsock, fromaddr = sock.accept()
  File "/usr/lib/python3.8/socket.py", line 292, in accept
    fd, addr = self._accept()
KeyboardInterrupt
root@793de2de45ef:/volumes/certS# ps
PID TTY TIME CMD
24 pts/1 00:00:00 bash
47 pts/1 00:00:00 ps
root@793de2de45ef:/volumes/certS# python3 server.py
Enter PEM pass phrase:
TCP connect
TLS connection fails
TCP connect
TLS connection established
"Request: b'GET / HTTP/1.0\\r\\nHost: client1-10.9.0.5\\r\\n\\r\\n'"
\\n"
root@b73202694eed:/volumes/certS#
```

Task. You are required to modify the client.py and server.py satisfying the following.

- 1) Client can interactively communicate with the server. The client takes the input from user and send to server; when the server receives the client message, it reverses it and sends back (e.g., hello will become olleh). The client then prints to the screen and waits for the next user input.

```
root@9134f2568f8e:/# cd volumes/CertS
ERROR: Encountered errors while bringing up the project.
[11/02/23]seed@VM:~/../lab6$ dockps
fd0337dce0b0 client1-10.9.0.5
9134f2568f8e client3-10.9.0.7
8326c1b03eb2 server-10.9.0.2
00e67d98dd02 client2-10.9.0.6
[11/02/23]seed@VM:~/../lab6$ docksh 83
root@8326c1b03eb2:/# cd volumes/certS
root@8326c1b03eb2:/volumes/certS# [11/02/23]seed@VM:~/../lab6$ d
ocksh 83
root@8326c1b03eb2:/# cd volumes/certS
root@8326c1b03eb2:/volumes/certS# python3 client.py client1-10.9.
0.5
Enter a message (or 'exit' to quit): hello
Received: olleh
Enter a message (or 'exit' to quit): panchal
Received: lahcnap
Enter a message (or 'exit' to quit): hi
Received: ih
Enter a message (or 'exit' to quit): hello
Received: olleh
Enter a message (or 'exit' to quit): how are you
Received: uoy era woh
Enter a message (or 'exit' to quit):

Enter PEM pass phrase:
TCP connect
TLS connection established
Received: hello
Reversed: olleh
TCP connect
TLS connection established
Received: hi
Reversed: ih
TCP connect
TLS connection established
Received: harshil
Reversed: lihsrah
TCP connect
TLS connection established
Received: panchal
Reversed: lahcnap
TCP connect
TLS connection established
Received: hi
Reversed: ih
TCP connect
TLS connection established
Received: hello
Reversed: olleh
TCP connect
TLS connection established
Received: harshil
Reversed: lihsrah
TCP connect
TLS connection established
Received: how are you
Reversed: uoy era woh
```

2) Server should support multi-threads. That is, it can communicate with several clients.

```
root@9134f2568f8e:/# cd volumes/CertS
bash: cd: volumes/CertS: No such file or directory
root@9134f2568f8e:/# cd volumes/certS
root@9134f2568f8e:/volumes/certS# ls
Test.crt Test.key client.py demo.ca.crt demo.ca.key server.py
root@9134f2568f8e:/volumes/certS# python3 client.py client1-10.9.0.5
Enter a message (or 'exit' to quit): hi
Received: ih
Enter a message (or 'exit' to quit): harshil
Received: lihsrah
Enter a message (or 'exit' to quit):
^C (press Ctrl+C again to force)

8326c1b03eb2 server-10.9.0.2
00e67d98dd02 client2-10.9.0.6
[11/02/23]seed@VM:~/../lab6$ docksh 83
root@8326c1b03eb2:/# cd volumes/certS
root@8326c1b03eb2:/volumes/certS# python3 client.py client1-10.9.
0.5
Enter a message (or 'exit' to quit): hello
Received: olleh
Enter a message (or 'exit' to quit): panchal
Received: lahcnap
Enter a message (or 'exit' to quit):

Enter PEM pass phrase:
11/02/23]seed@VM:~$ dockps
0337dce0b0 client1-10.9.0.5
134f2568f8e client3-10.9.0.7
326c1b03eb2 server-10.9.0.2
e67d98dd02 client2-10.9.0.6
11/02/23]seed@VM:~$ docksh fd
pot@fd0337dce0b0:/# cd volumes/certS
pot@fd0337dce0b0:/volumes/certS# ls
Test.crt client.py demo.ca.key
Test.key demo.ca.crt server.py
pot@fd0337dce0b0:/volumes/certS# python3 server.py
Enter PEM pass phrase:
TCP connect
TLS connection established
Received: hello
Reversed: olleh
TCP connect
TLS connection established
Received: hi
Reversed: ih
TCP connect
TLS connection established
Received: harshil
Reversed: lihsrah
TCP connect
TLS connection established
Received: panchal
Reversed: lahcnap
```