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. */

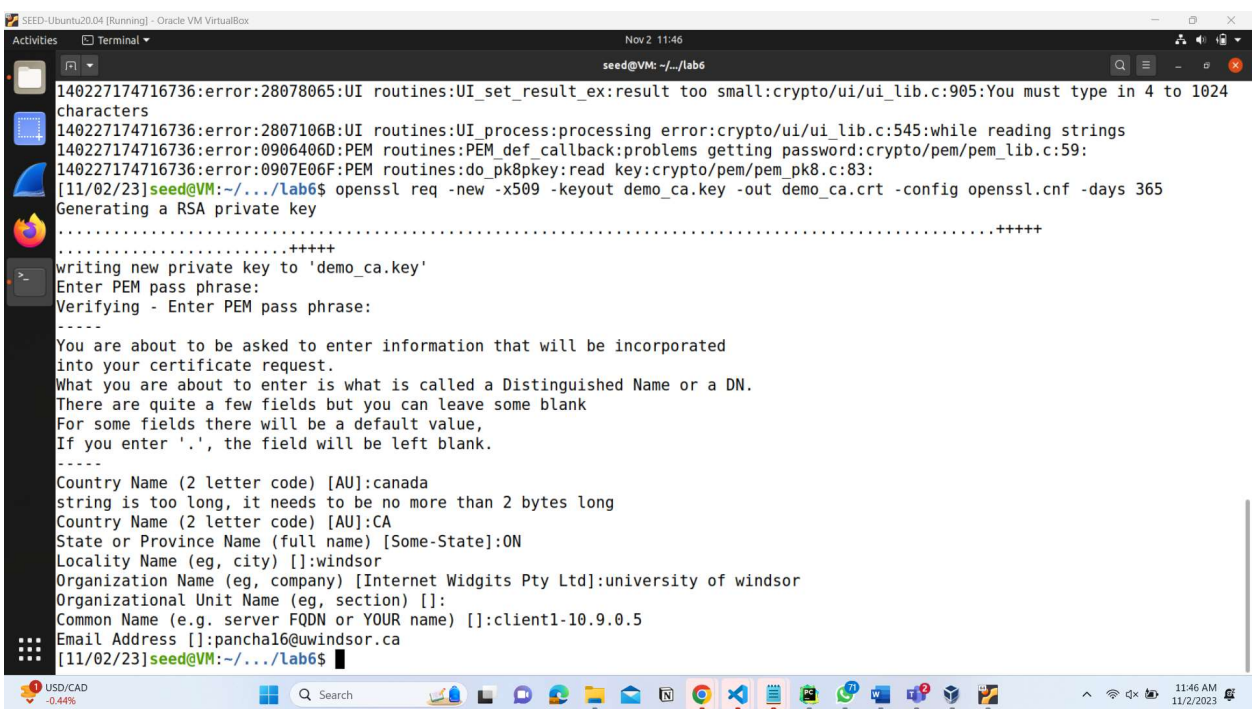
2. Create a new directory demoCA in the current directory. Then, do the following.
   a. Create new directories certs, crl and newcerts in demoCA and empty files index.txt and serial. Put a single serial number (.e.g., 1000) in the file serial.

   (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. */

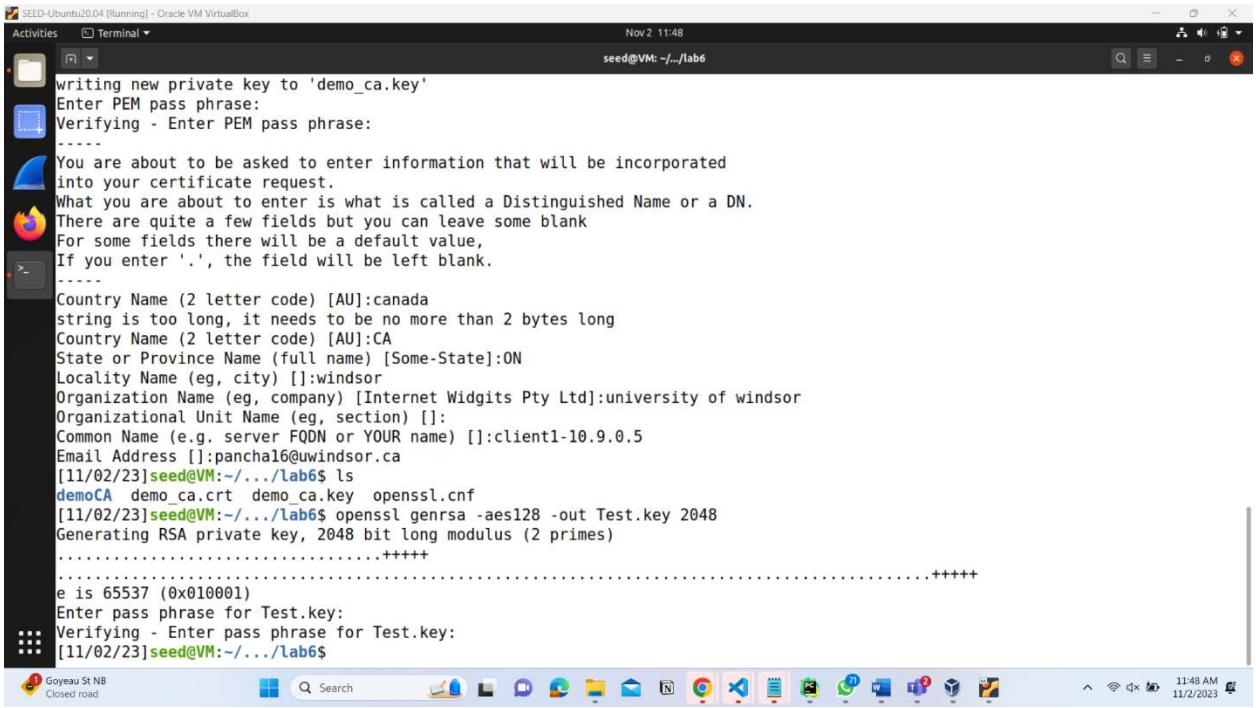


c. Create a certificate for our test TLS server, signed by our authority's key demo_ca.key.

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 */



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). */

```
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 []:pancha16@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$
```

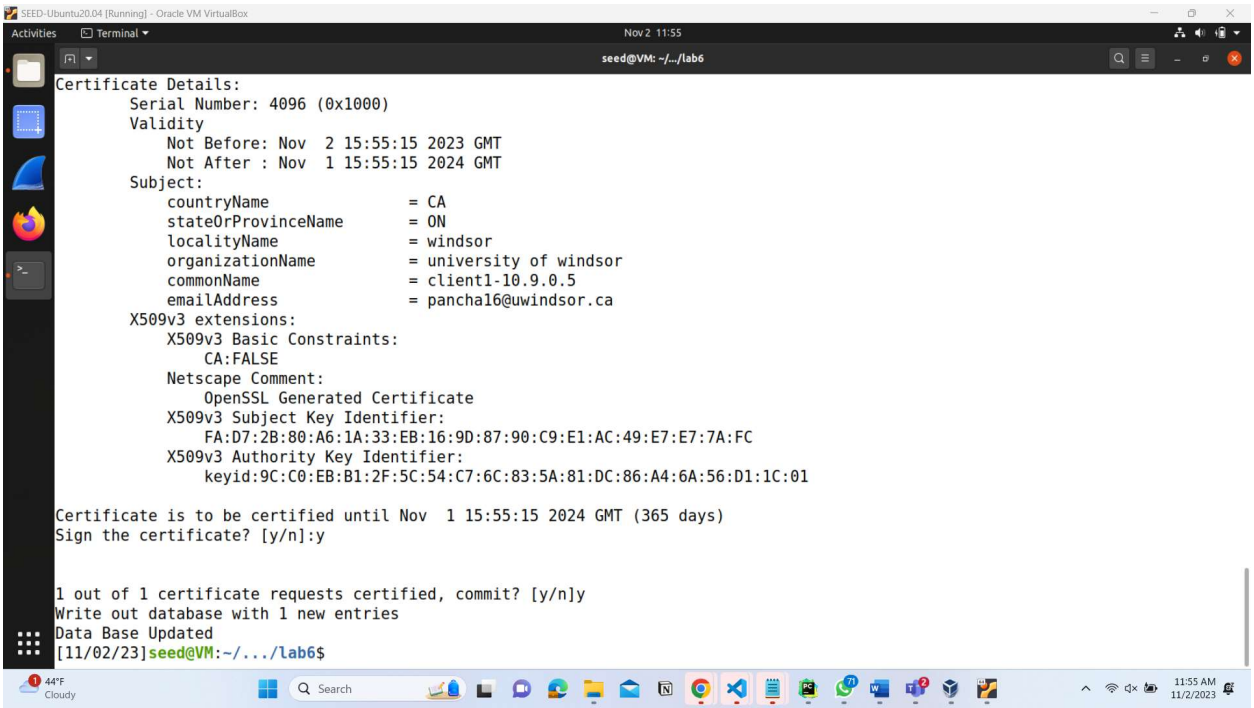
iii. Generate the certificate for TLS server:

$ 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 */

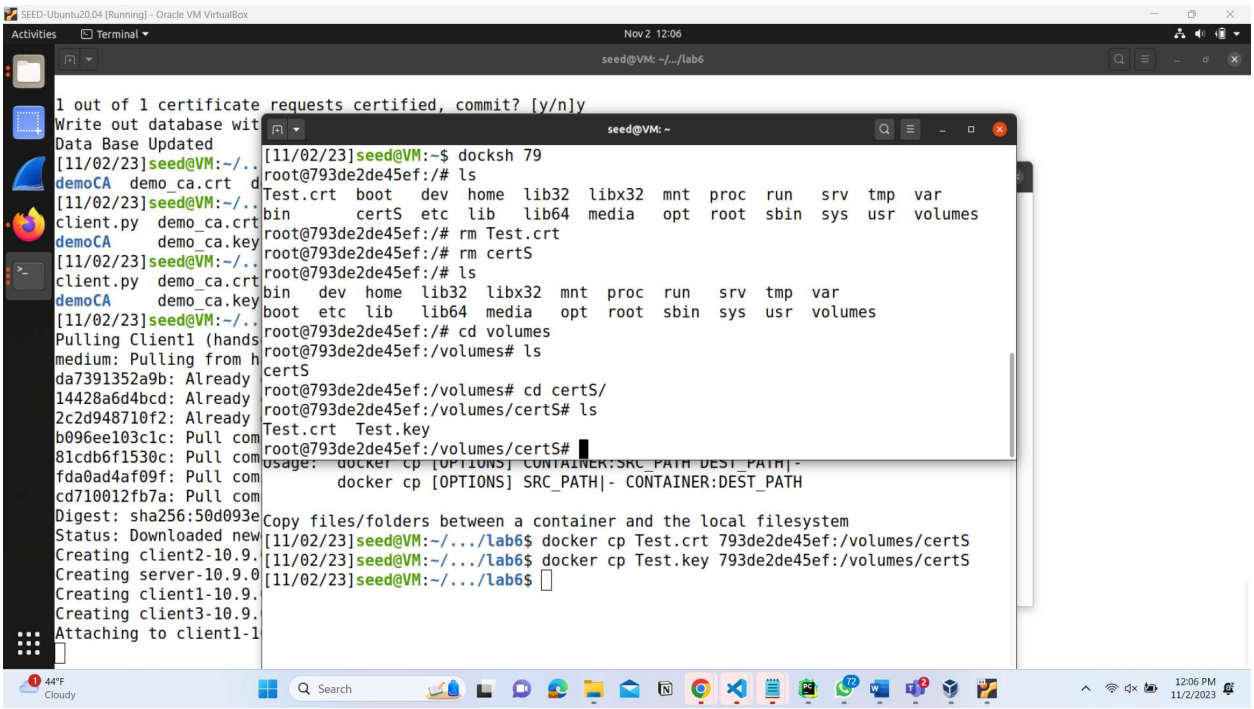
```
140379267425600:error:0907B00D:PEM routines:PEM_read_bio_PrivateKey:ASN1 lib:crypto/pem/pem_pkey.c:88:
[11/02/23]seed@VM:~/.../lab6$ openssl ca -in Test.csr -out Test.crt -cert demo_ca.crt -keyfile demo_ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for demo_ca.key:
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              = 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]:
```



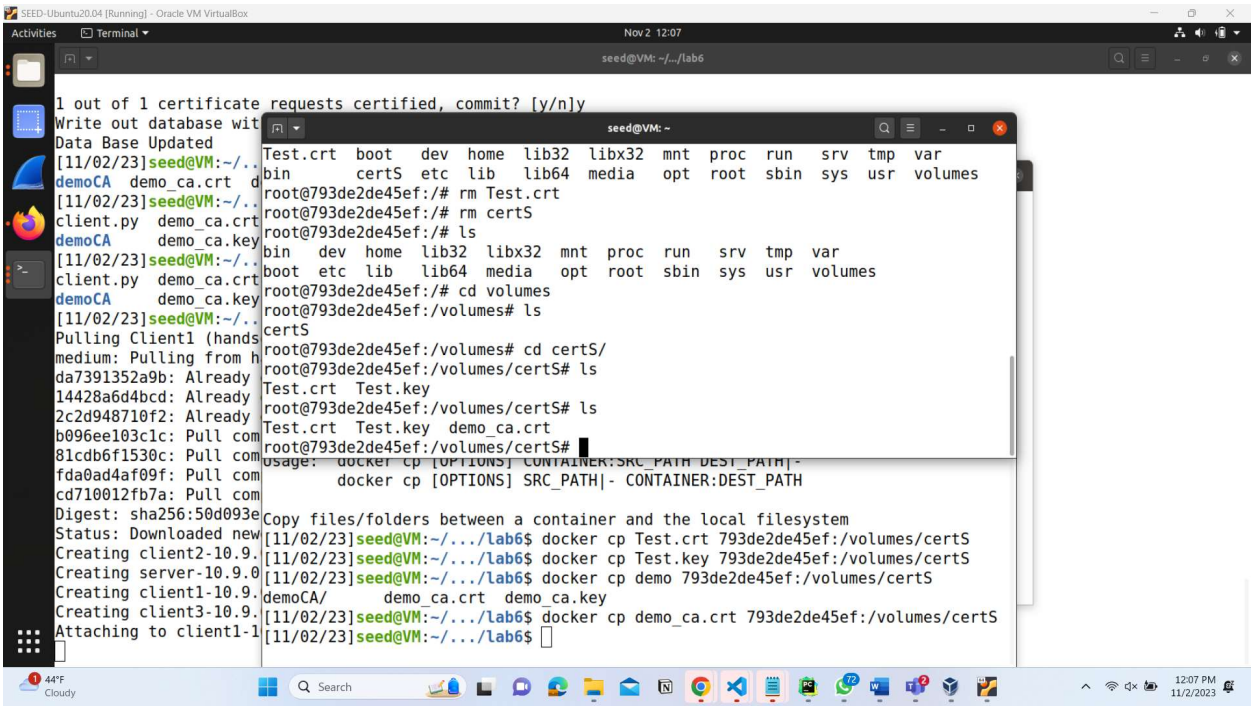
/*the following steps operated in volumes might need sudo; volumes is the folder created by your docker-compose (using the newly provided yml file) */

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).



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

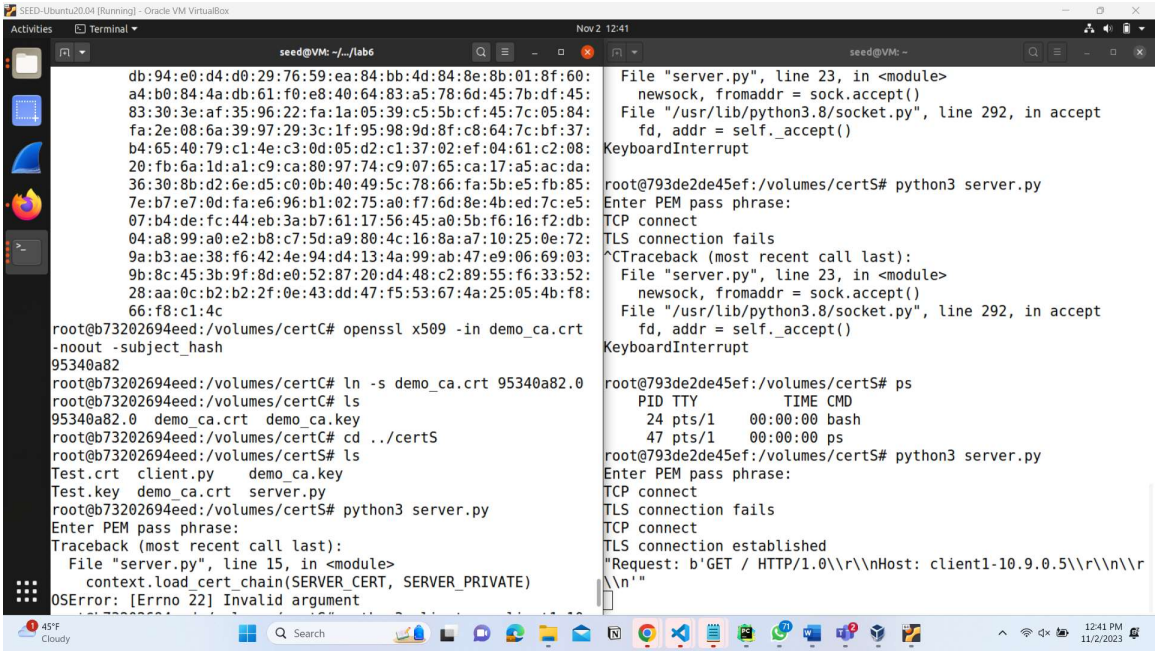
indeed owned by your server VM. However, your client VM can not directly locate demo_ca.crt. You need to create a symbolic link that links to demo_ca.crt. Your client VM can use this symbolic link to find demo_ca.crt in certs.



(a) openssl x509 -in demo_ca.crt -noout -subject_hash

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

(b) ln -s demo_ca.crt 8f838d2e.0 /*warning: do not forget .0 */



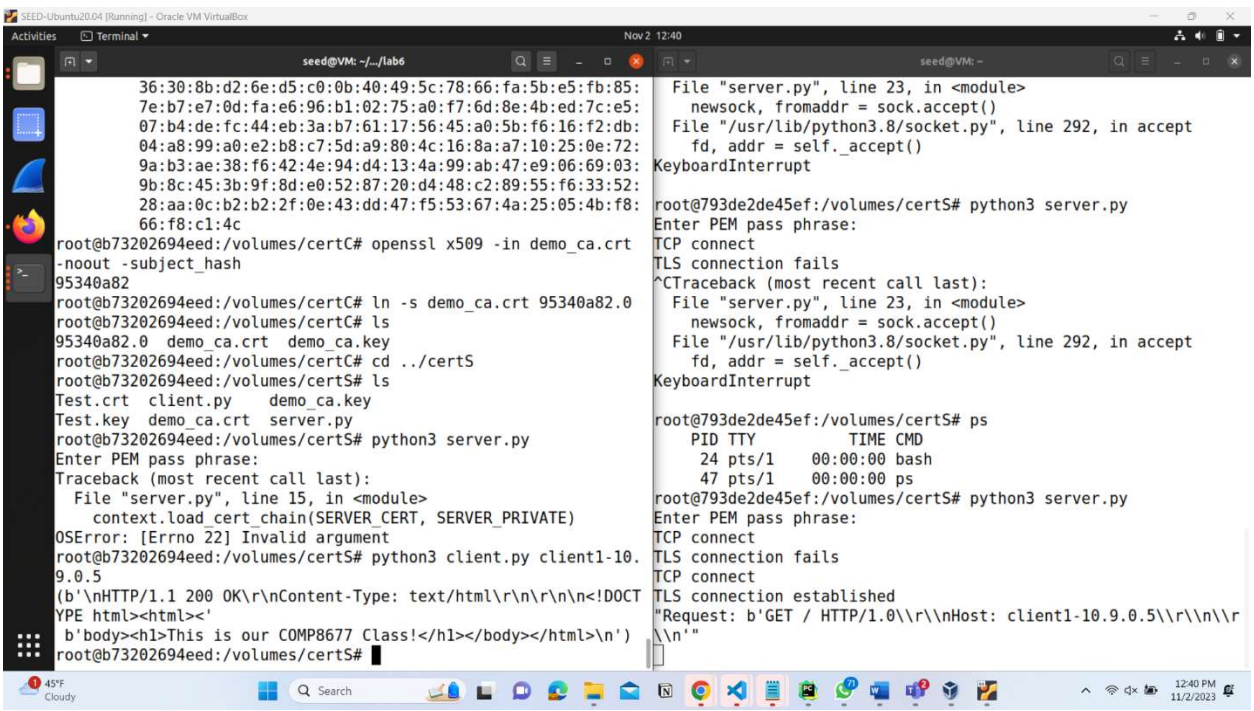
Part II. (TLS Client and Server Communication)

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):

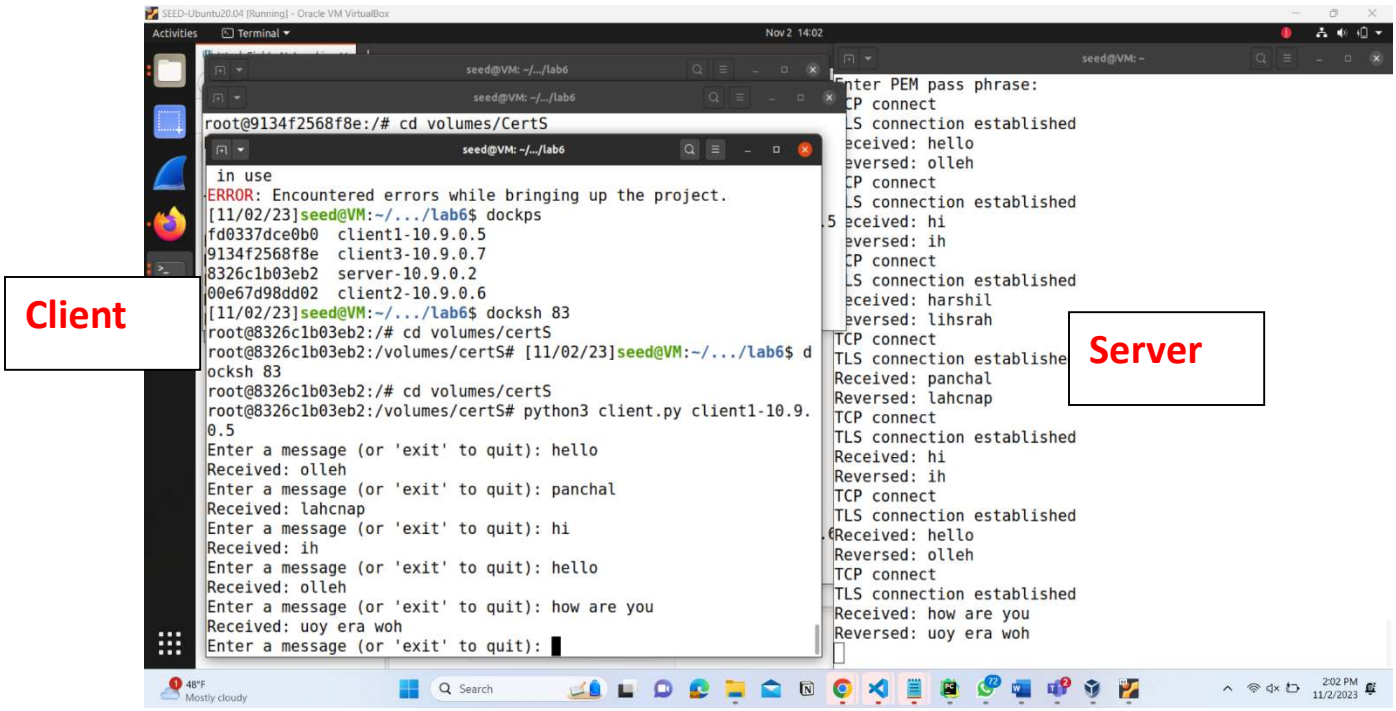
$ python3 client.py client1-10.9.0.5

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



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.



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