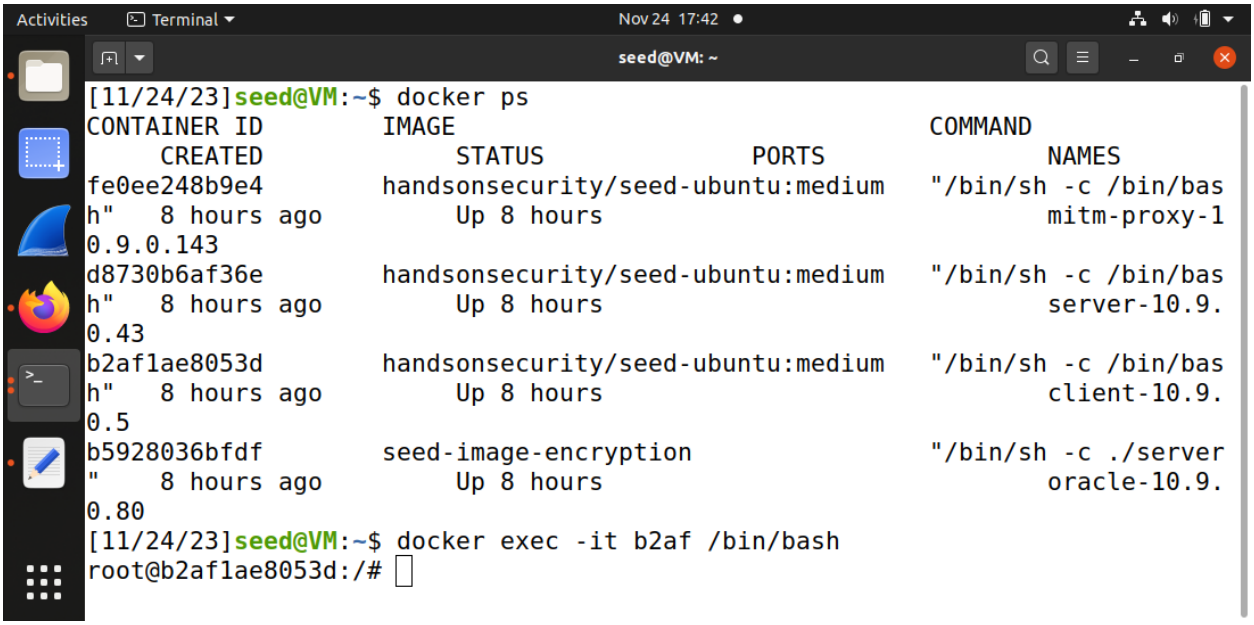
Transport Layer Protocol

Container Setup and Commands. We have followed the lab setup for the environment according to the seed labs TLS pdf.

Container setup shown below:



Task 1: TLS Client

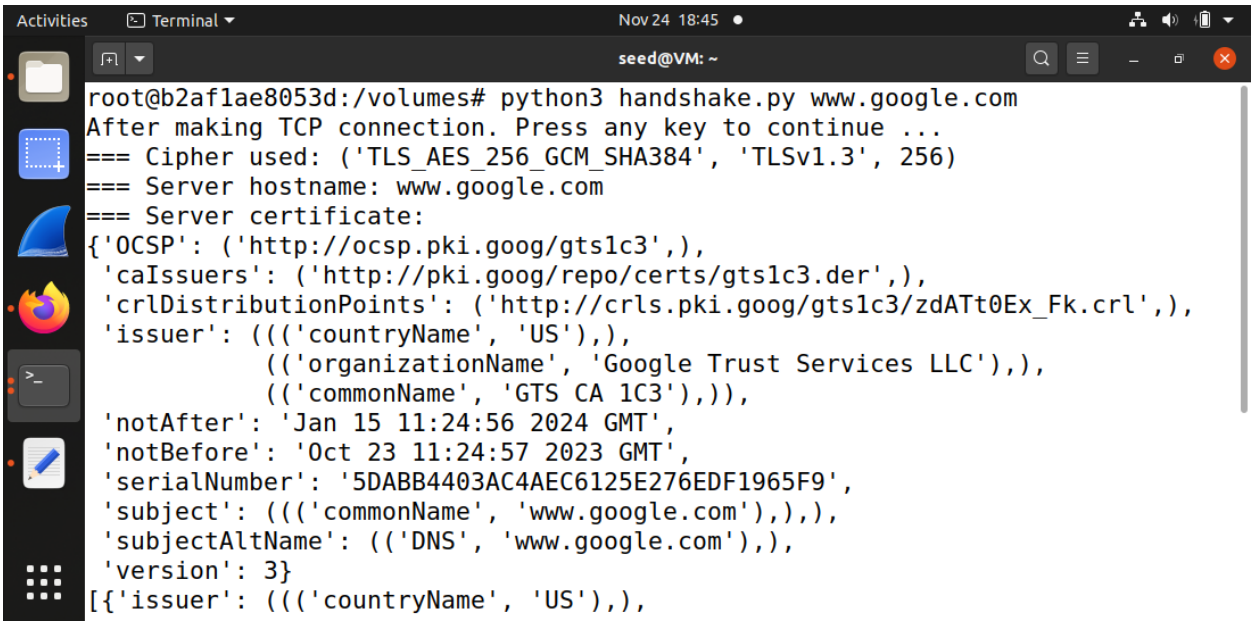
Steps to build a TLS client program.

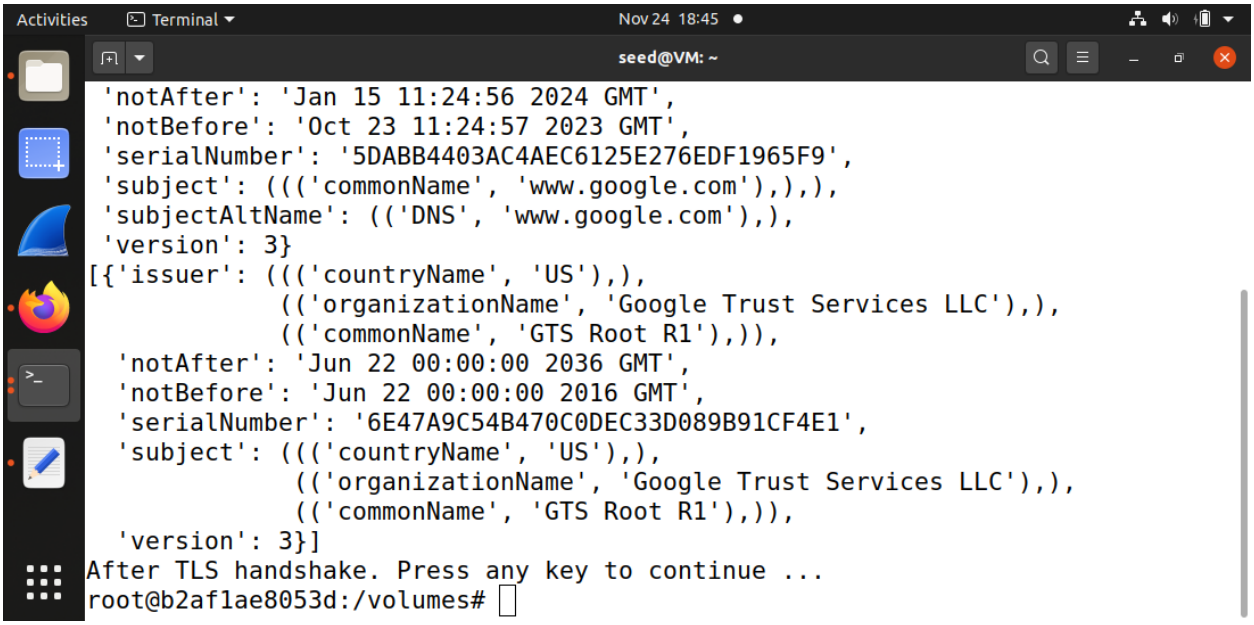
Task 1.a- TLS handshake

TLS handshake protocol is used to set up what encryption algorithm and key will be used, what MAC algorithm will be used, what algorithm should be used for the key exchange, etc. These cryptographic parameters need to be agreed upon by the client and the server. This is the primary purpose of the TLS handshake.
The code is given to us in the seed Lab pdf as “handshake.py”

We have to use this code to communicate with the real https server.

Running the handshake.py file to communicate with the “www.google.com” https server



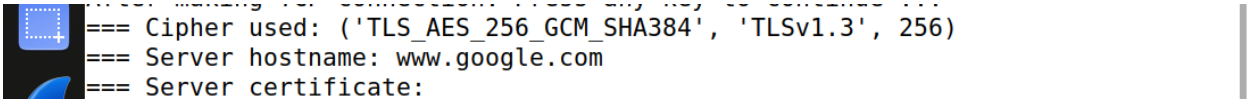


- What is the cipher used between the client and the server?

  We cant see which Cipher is used in the program given we have to add a new line to the handshake.py file to display the Cipher used.

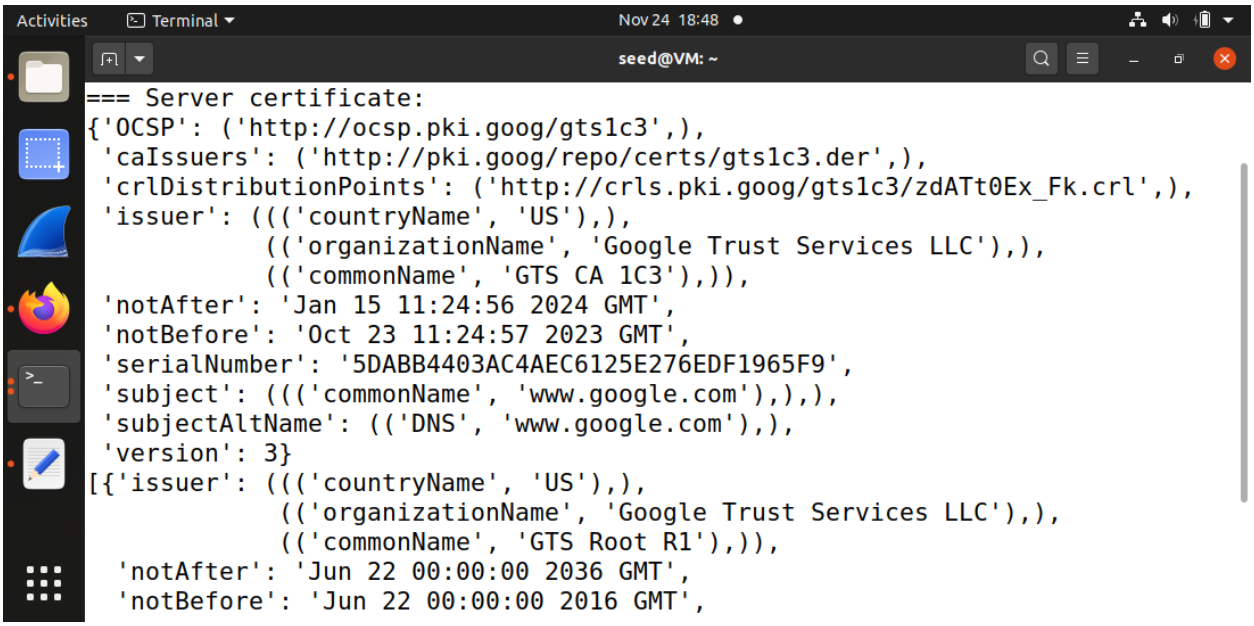
  print(ssock.cipher()[0])

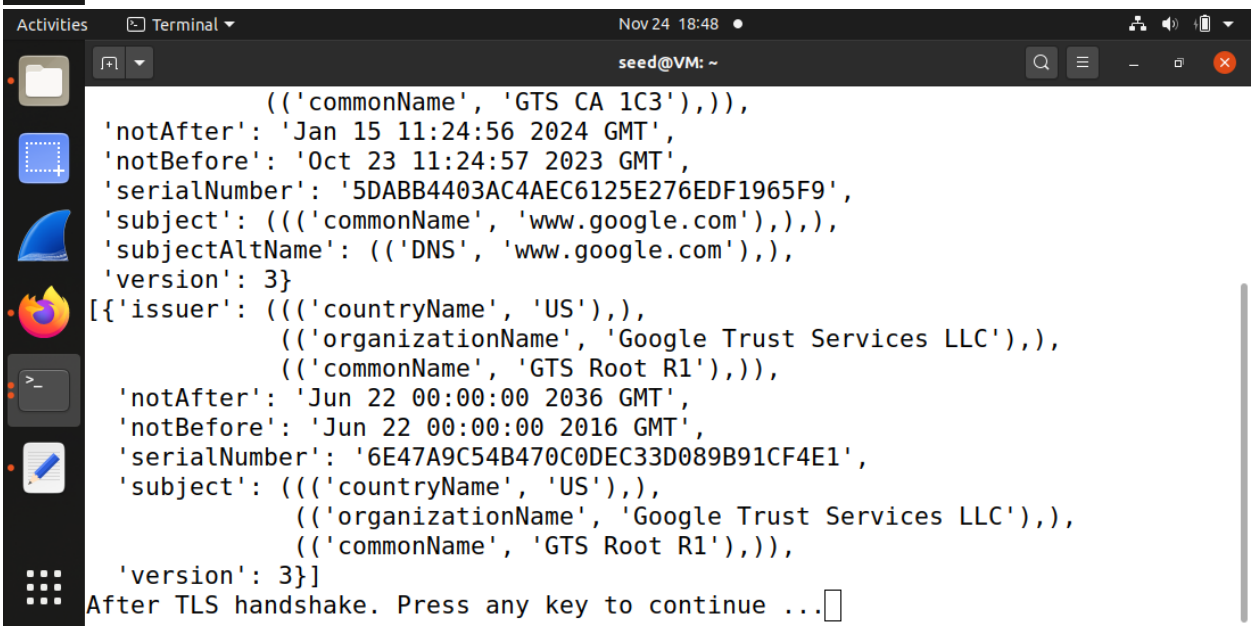
  As we can see here when we ran our program with the google server we can see the output given. In the output under Cipher used we see which cipher is used.



- Please print out the server certificate in the program
  In the handshake.py program the **pprint.pprint(ssock.getpeercert())** gives us the Sever Certificate in the output

  In output when we executed our handshake.py file with the google server the server certificate generated is shown after the Server Certificate place in the output

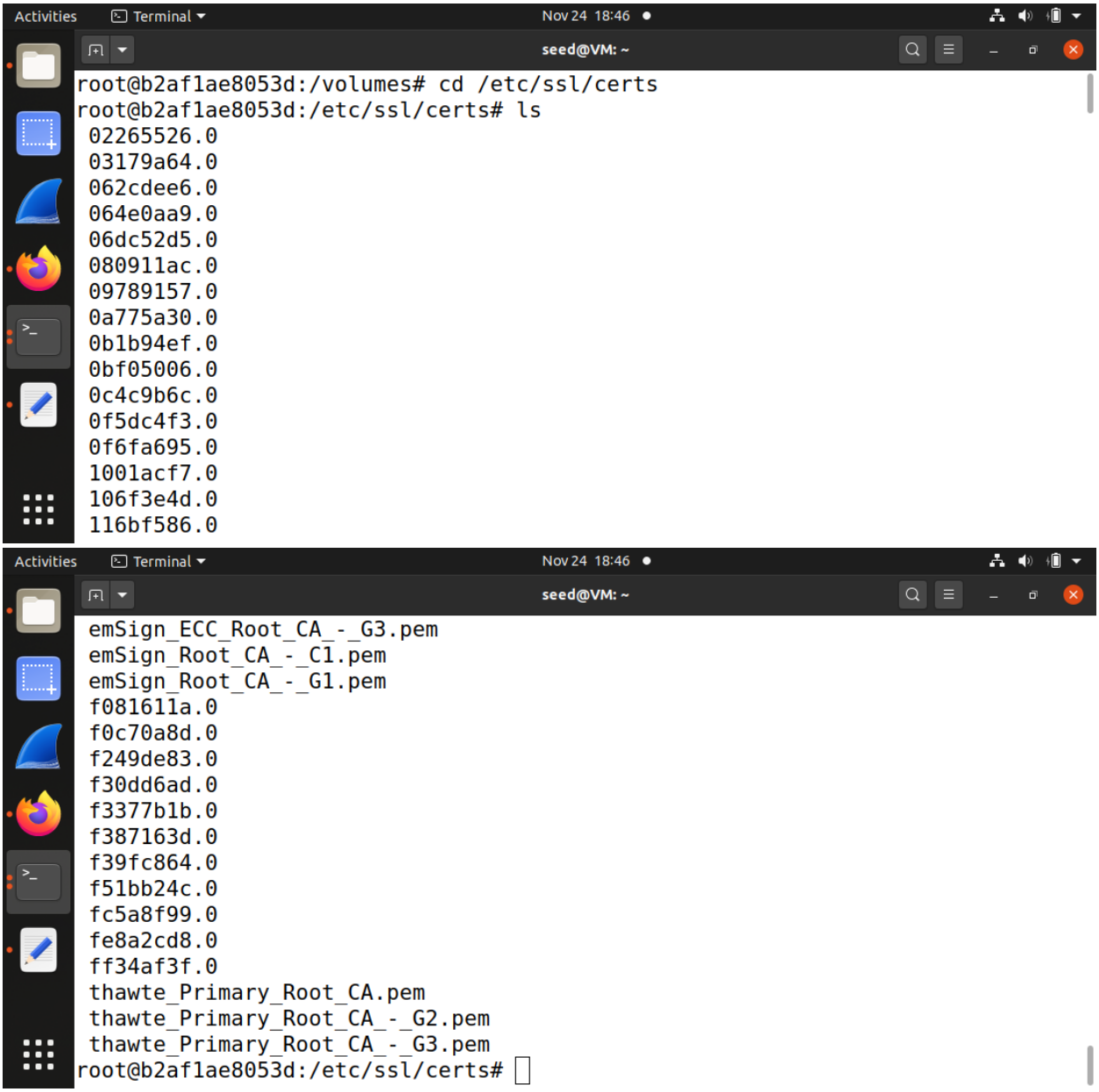




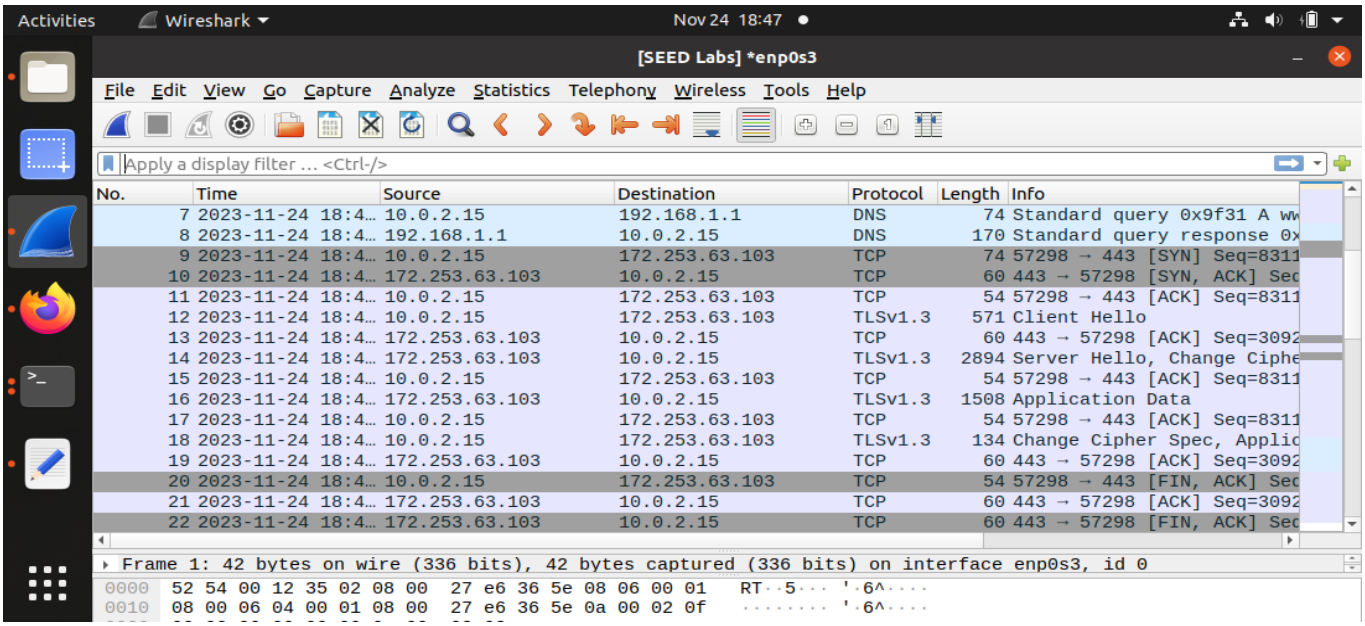
- Explain the purpose of /etc/ssl/certs.?
  So, this is used to check if the server certificate is generated by a valid/ trusted CA not someone else. When a user or client connects to the SSL/TLS server in our case it is google.com server it

checks this server certificate chain with the public key certificate that is on the clients system.The root CA required the verify the certificate is stored here.

The /etc/ssl/certs path is where the trusted CA certificates will be stored.



- Use Wireshark to capture the network traffics during the execution of the program, and explain your observation. In particular, explain which step triggers the TCP handshake, and which step triggers the TLS handshake. Explain the relationship between the TLS handshake and the TCP handshake.

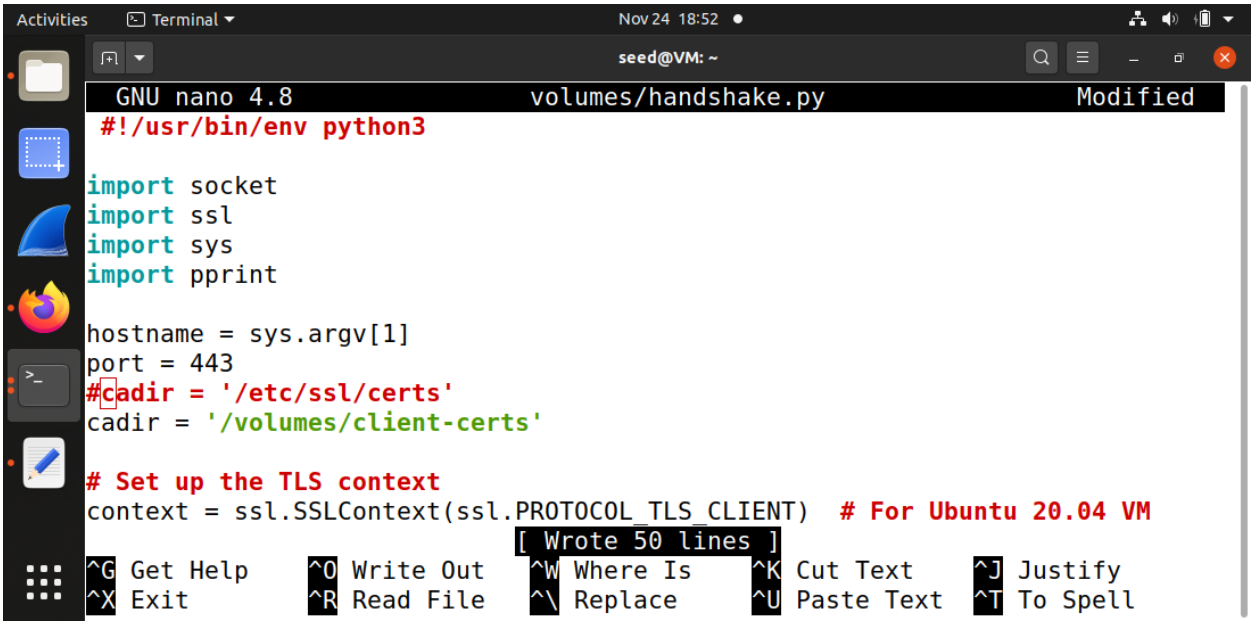


- Lines 9 to 11 are the TCP 3-way handshake.
- Line 9 a SYN packet is set from the client to the server
- Line 10 the server sends a SYN and ACK(Acknowledgment) to the client based on the previous step
- Once we finish TCP handshake it is TLS handshake .This is the process of negotiating encryption parameters and establishing a secure channel between the client and the server.
- Line 12 to 18 is the TLS handshake which includes the following steps
- There are several steps in the TLS handshake, including:
- ClientHello: A message describing supported cryptographic algorithms along with additional parameters is sent by the client. Client sends hello first
- ServerHello: The encryption methods and other settings selected by the server are returned. Then server responds with hello
- Key Exchange: To create a shared secret, the client and server exchange important data.
- Completed: Both individuals affirm that the handshake has ended.
- Line 19- 20 TCP termination using TCP FIN,ACK

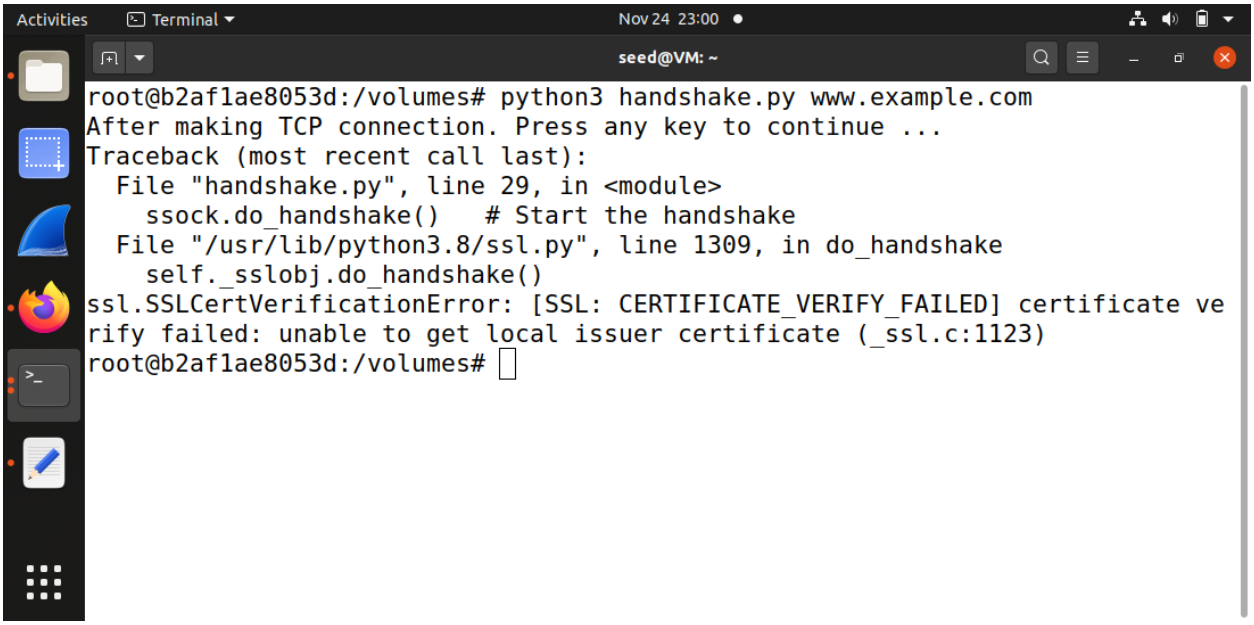
Explain the relationship between the TLS handshake and the TCP handshake.
Any data transfer requires two endpoints that must first create a trustworthy connection-oriented communication channel via the three-way TCP handshake. Before any sensitive data is transmitted, a safe, encrypted connection is first established between two endpoints via the TLS handshake, which takes place on top of the TCP connection.Handshake messages need to be exchanged across a stable and established TCP connection,which is why the TLS handshake depends on the TCP handshake.

Task 1.b: CA's Certificate

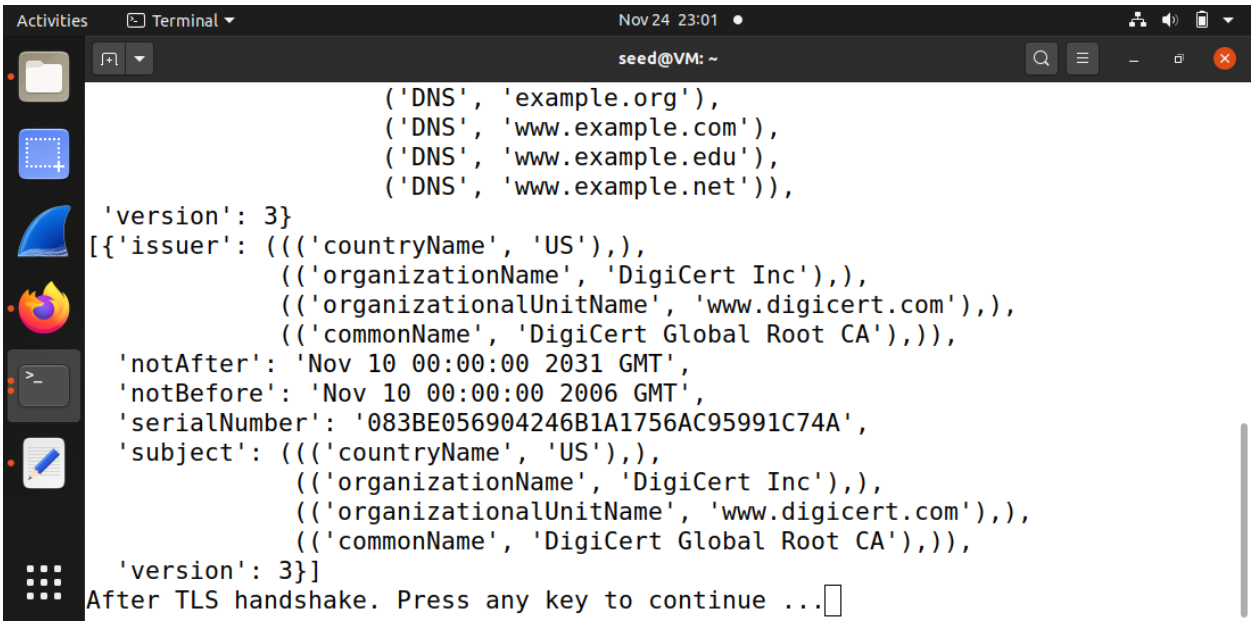
Created a folder called client-certs, and changed the cadir line in the client program to the following:



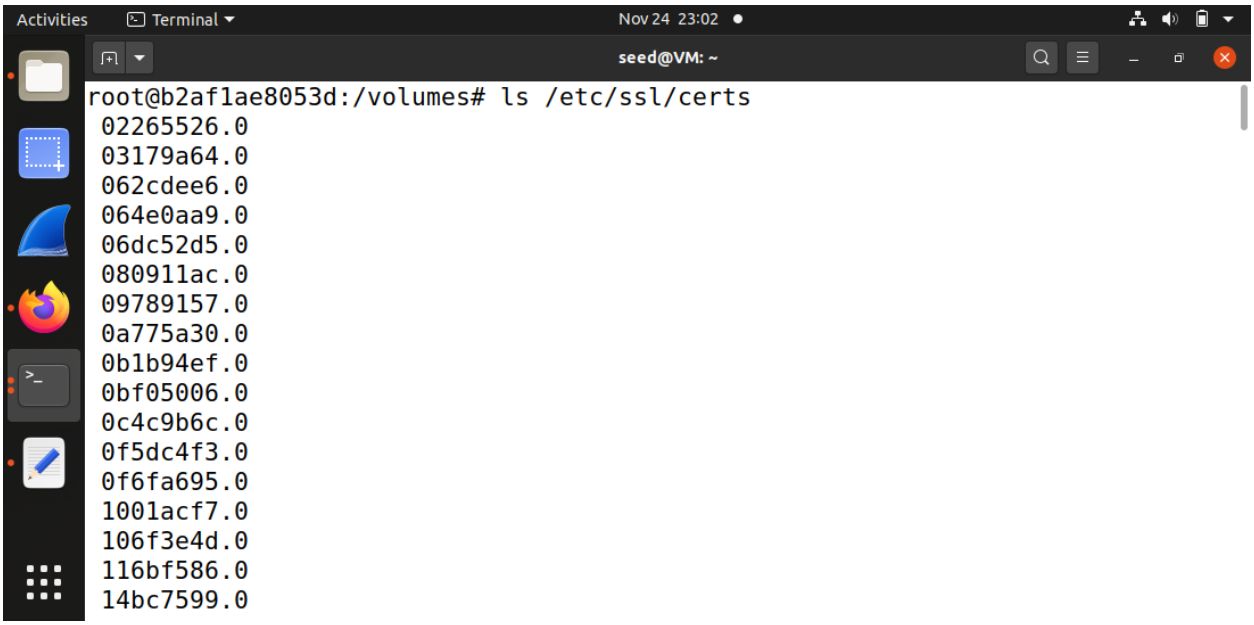
When we run the client program again, we get an error as “unable to get local issuer certificate”

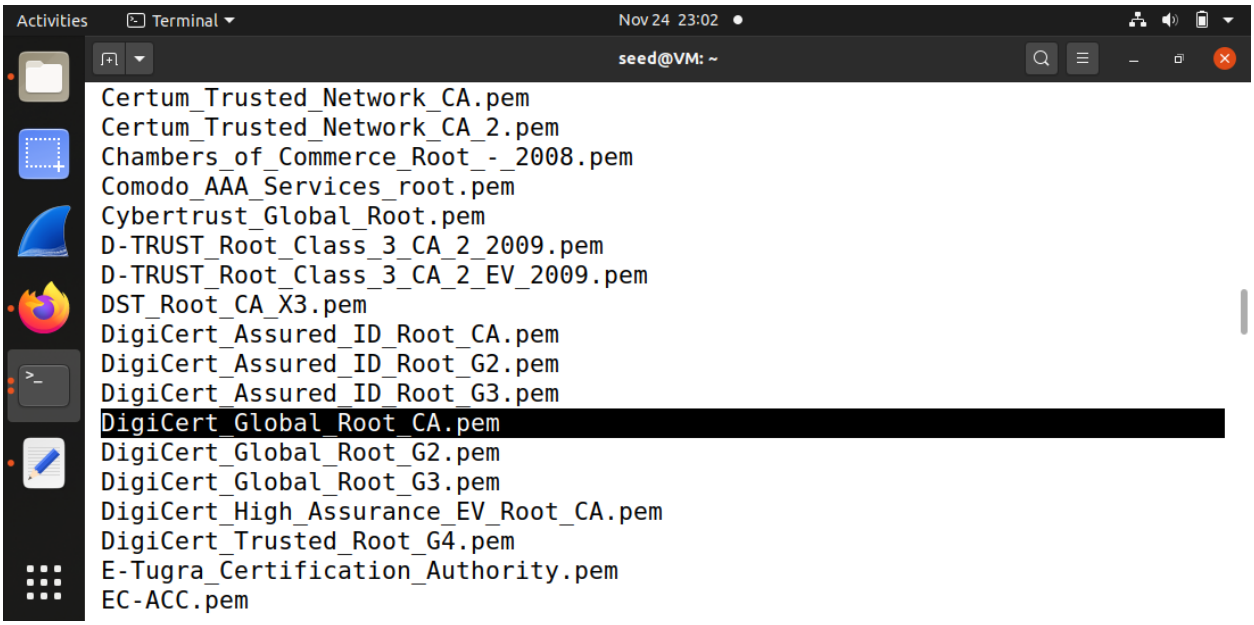


To fix this we have to place the corresponding CA’s certificate into your client-certs folder. Ie.
I used my client program to find out what CA certificate is needed to verify the www.example.com server’s certificate, and then copied the certificate from the /etc/ssl/certs to my folder.

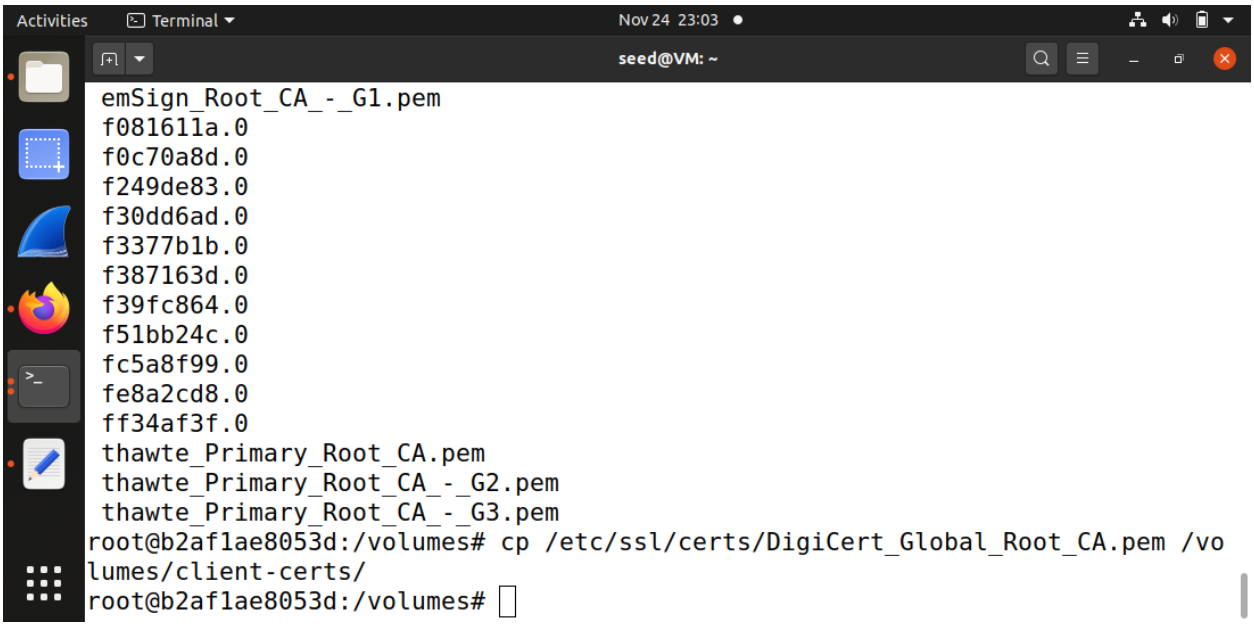


So, checking the /etc/ssl/certs folder





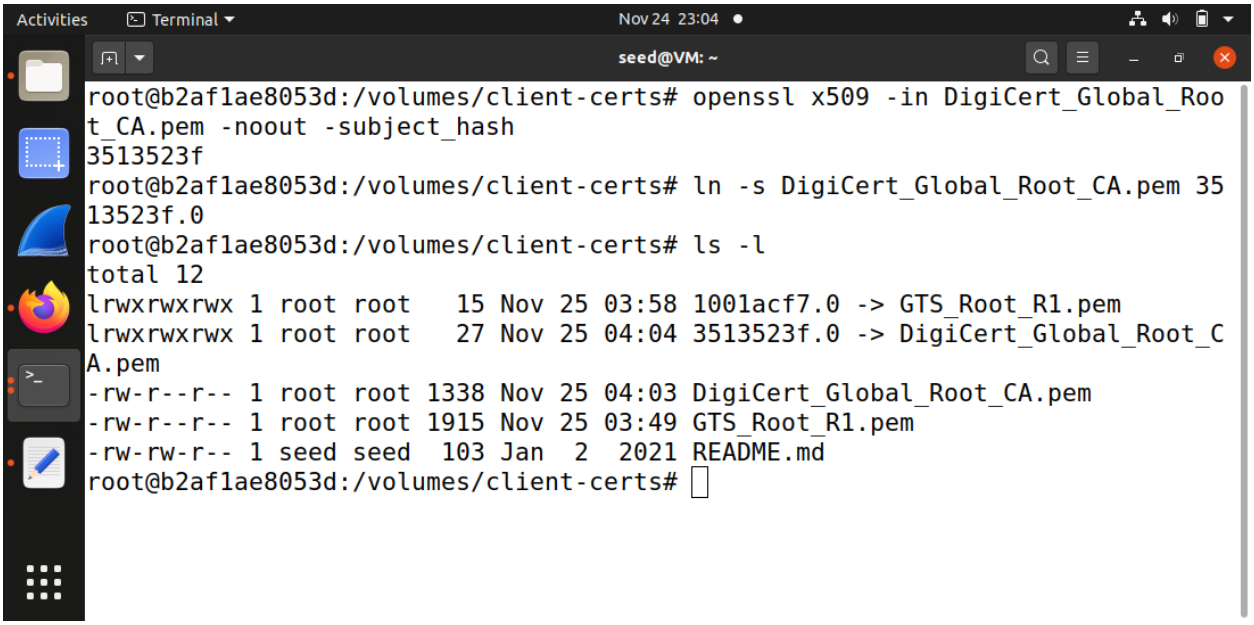
We find the CA certificate needed and we copy it to the new file.



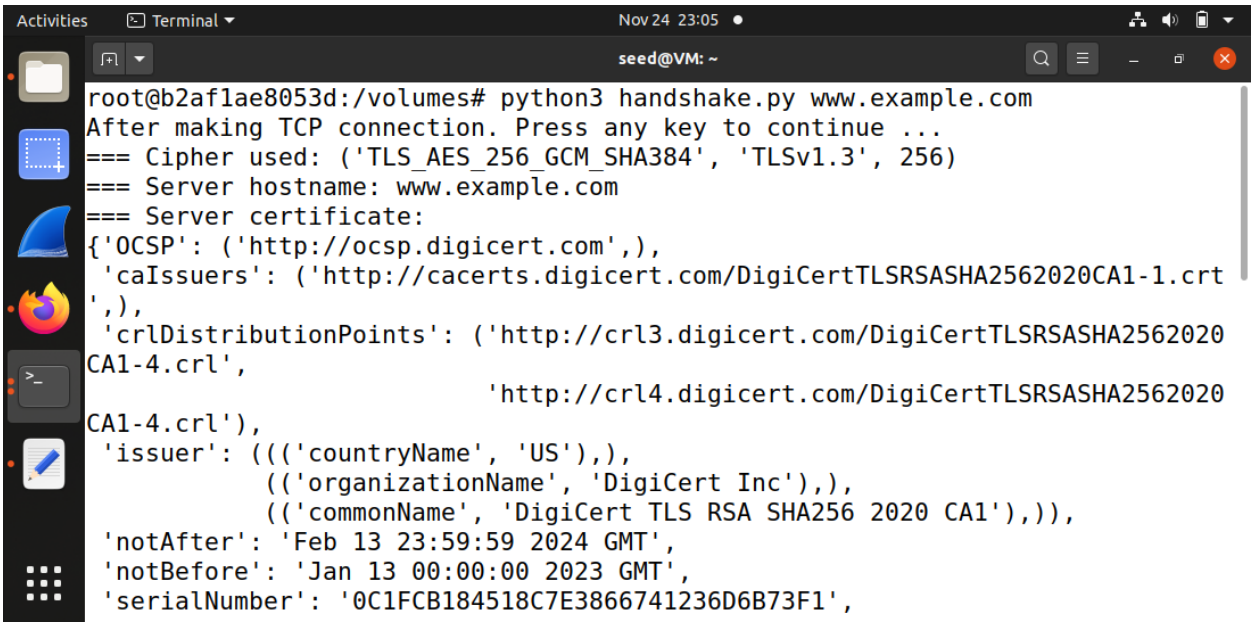
Copying CA's certificate to the "./client-certs" folder is not enough. After finding the CA certificate to copy and copying it the folder we used openssl command to generate a hash value for that certificate. We created a symbolic link for this.

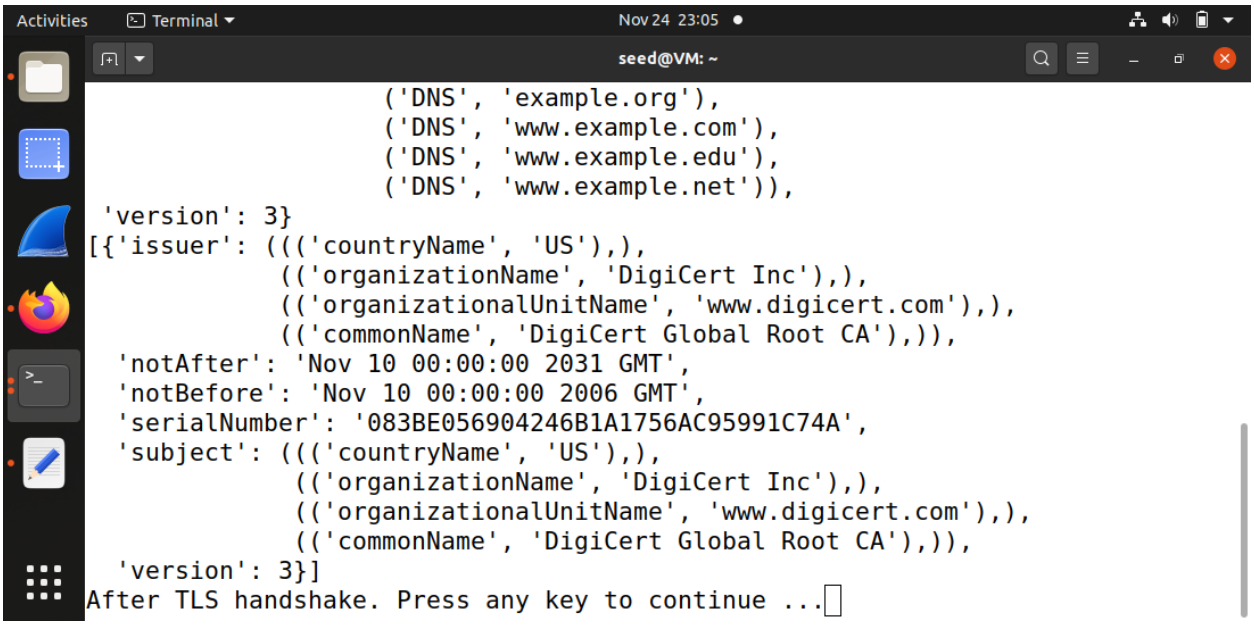
When TLS tries to verify a server certificate, it will generate a hash value from the issuer's identify information, we use this hash value as part of the file name, and then use this name to find the issuer's certificate in the "./client-certs" folder. Therefore, we need to rename each CA's certificate using the hash value generated from its subject field, or we can make a symbolic link out of the hash value. In the

following command, we use openssl to generate a hash value, which is then used to create a symbolic link.



After that we run the “handshake.py” program again and we see that it works now.



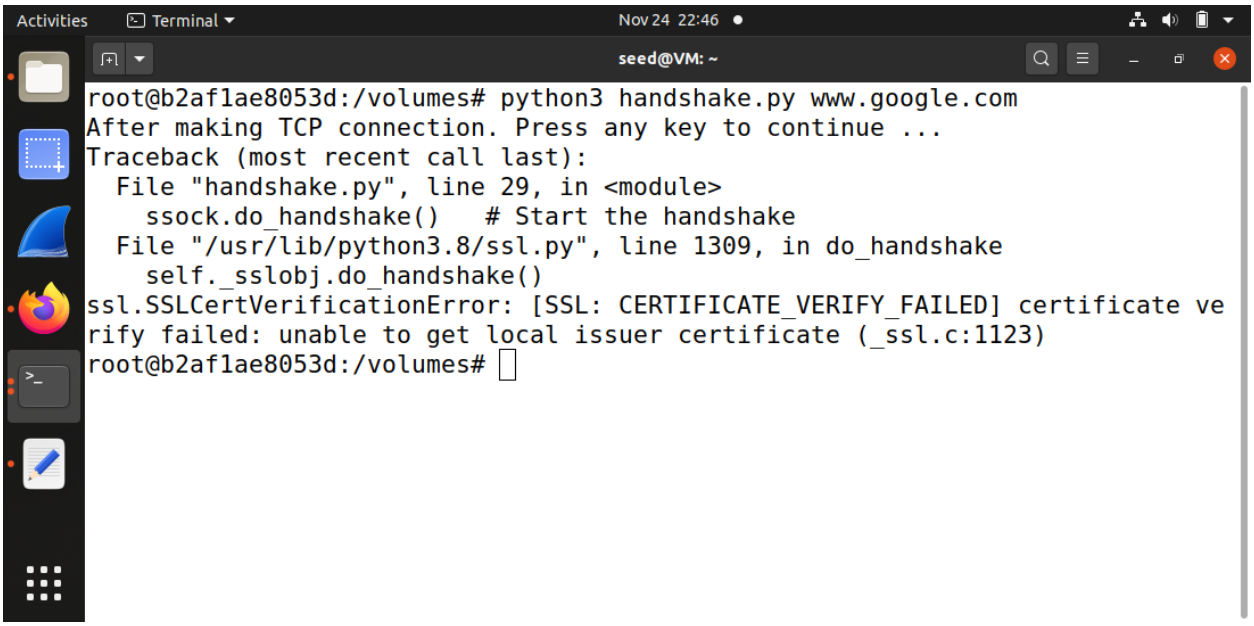


Conducted it for 2 other sites:

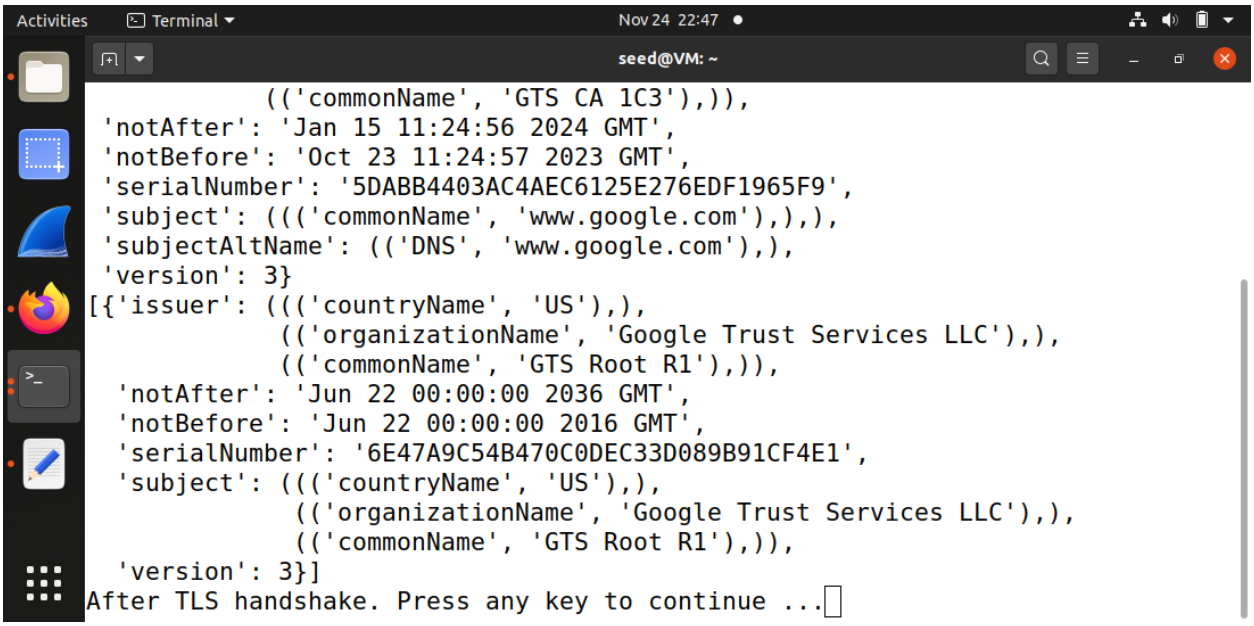
www.google.com

Error:

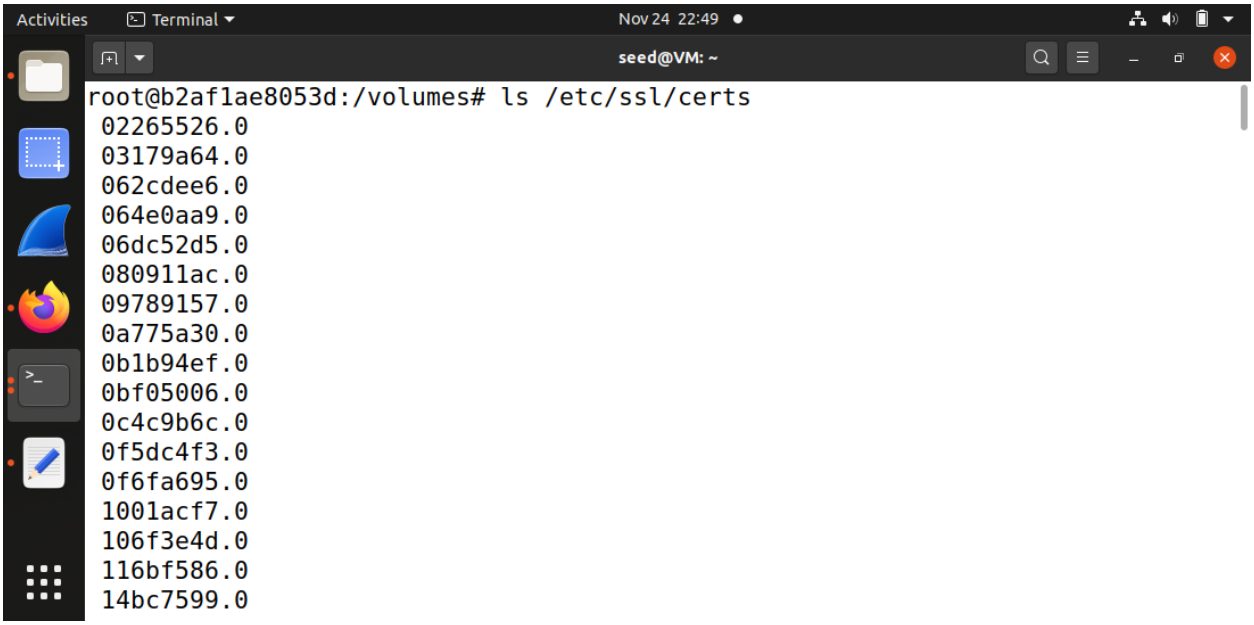
We get before for not having any certificates in new folder.

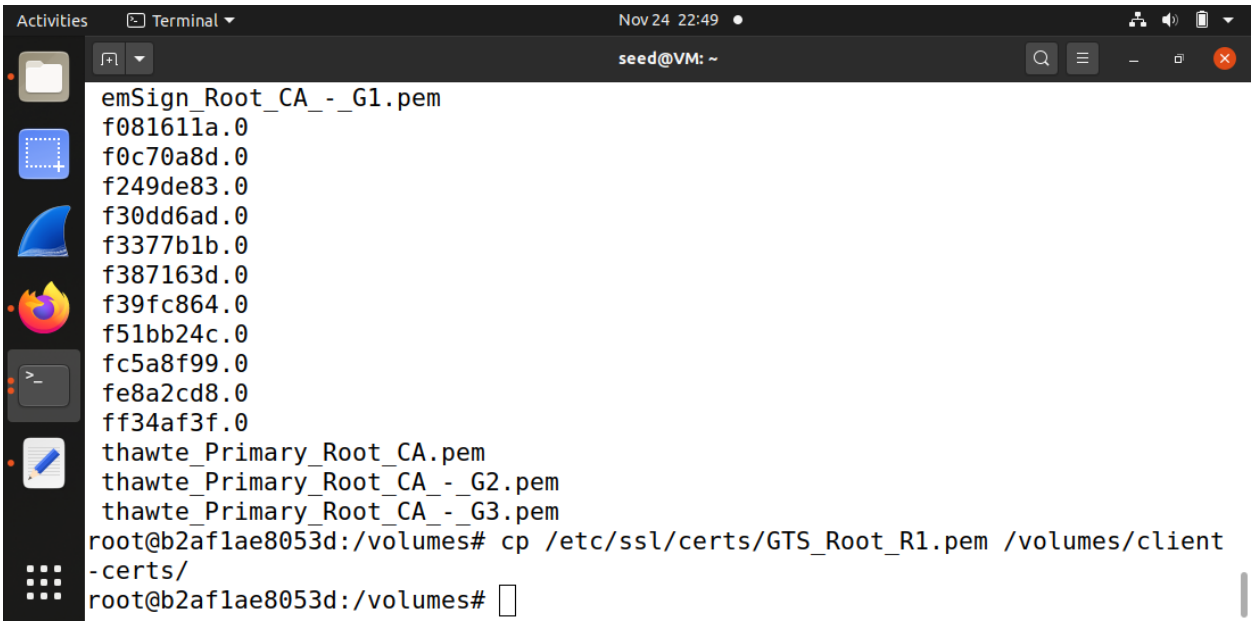


Using client program to figure out the CA certificate needed to be copied to the new folder.

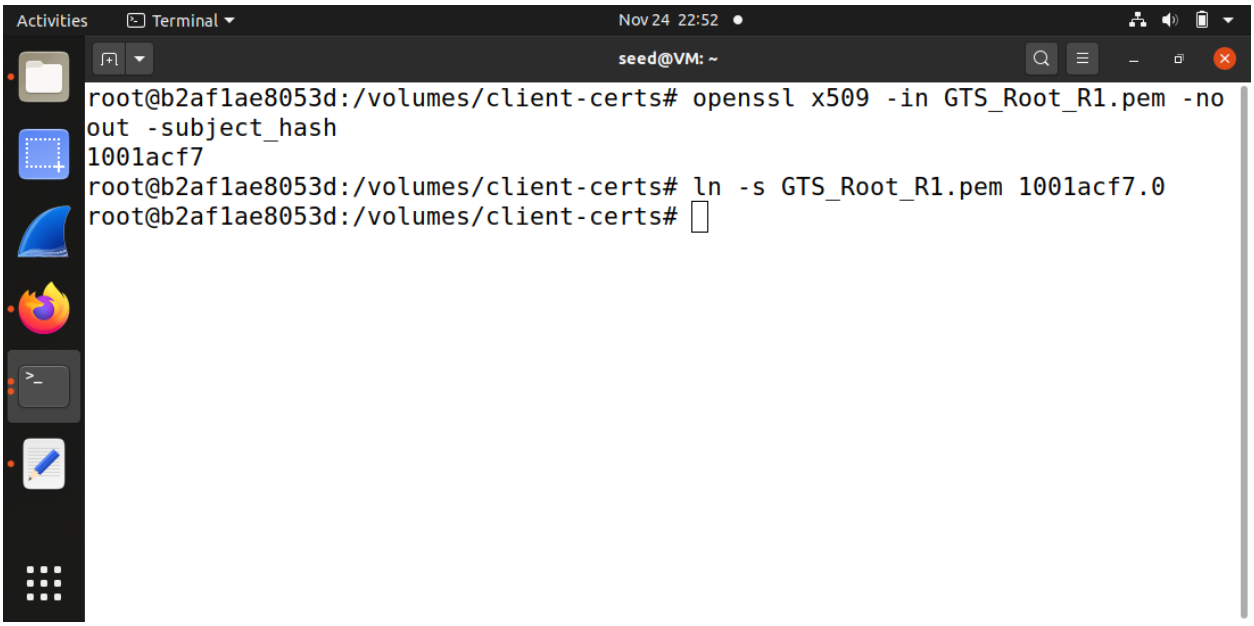


Looking for the certs in the /etc/ssl/certs file and copying the CA certificate to the new folder

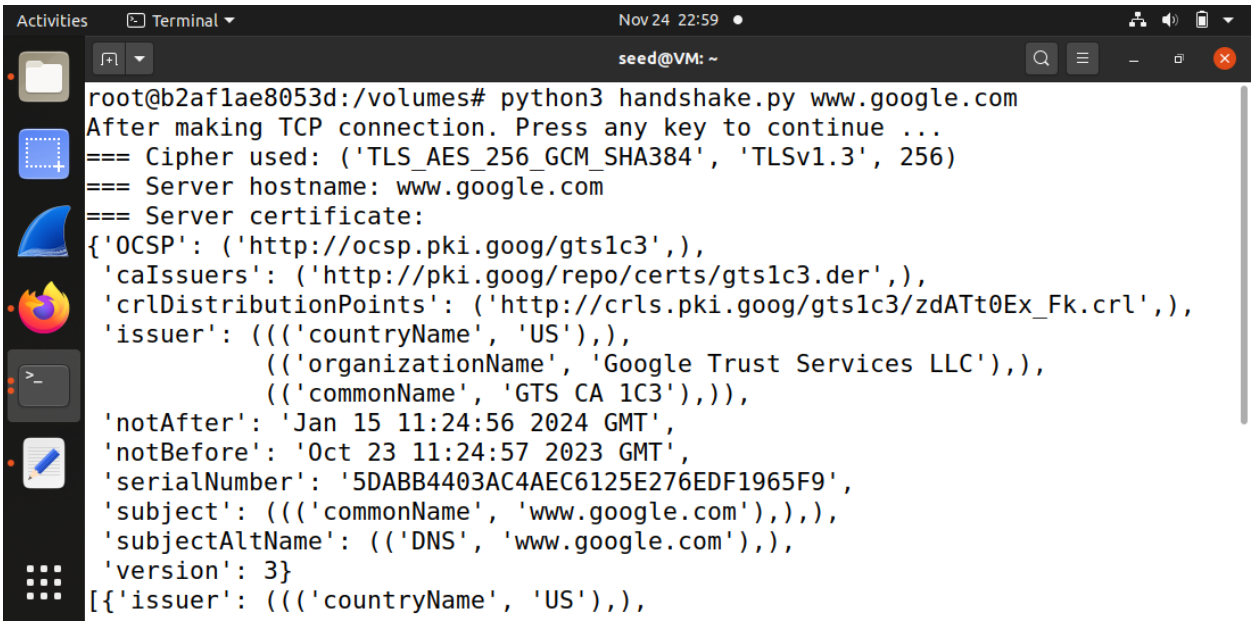


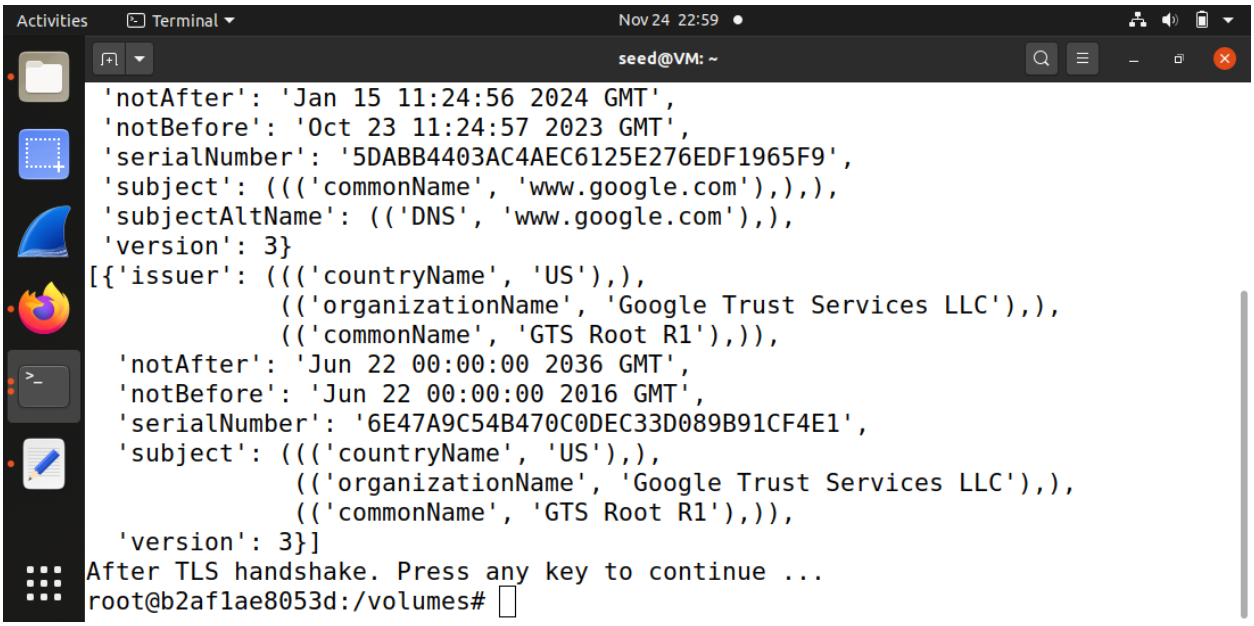


After finding the CA certificate to copy and copying it the folder we used openssl command to generate a hash value for that certificate. We created a symbolic link for this.



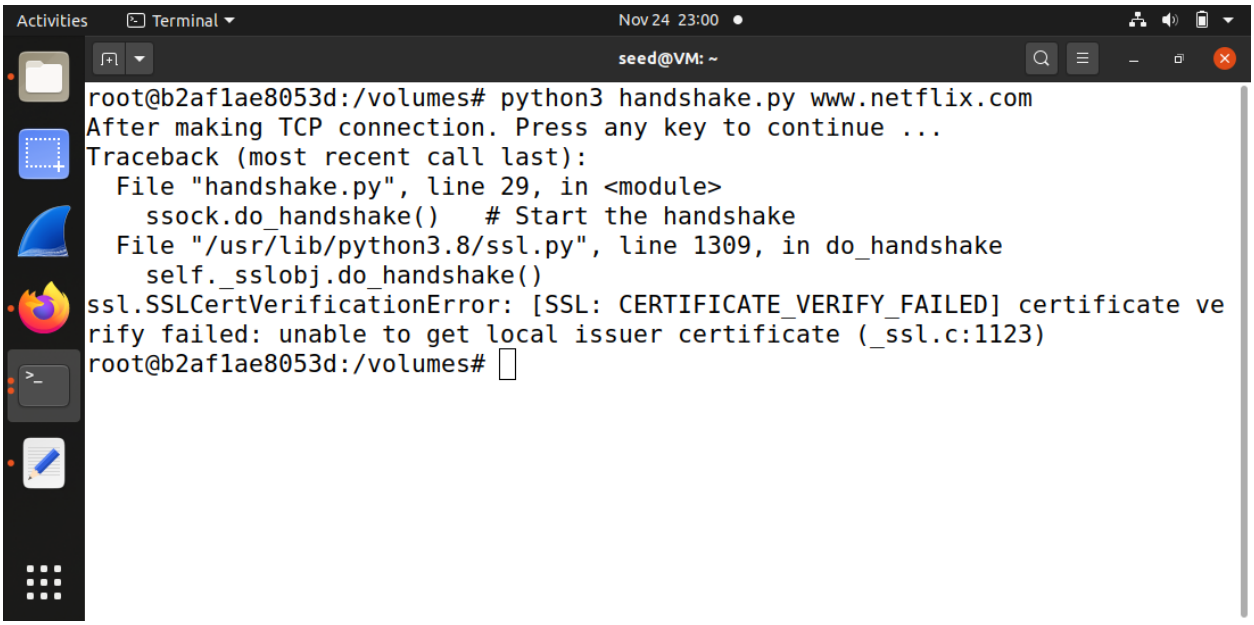
Now we run the “handshake.py” program.



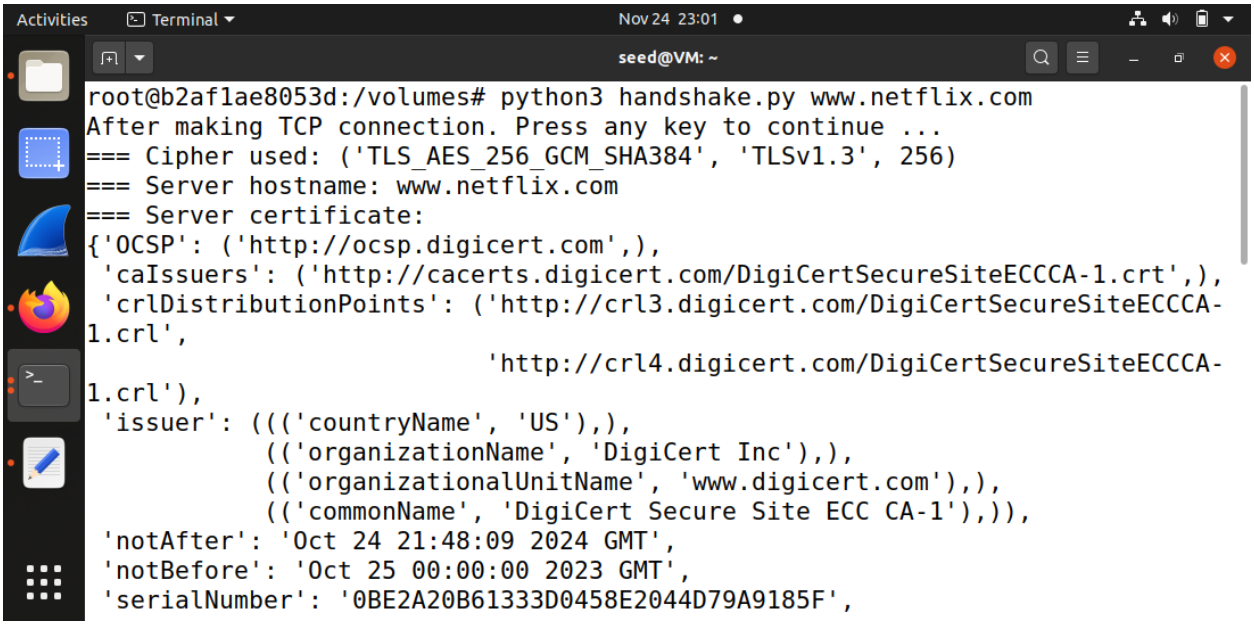


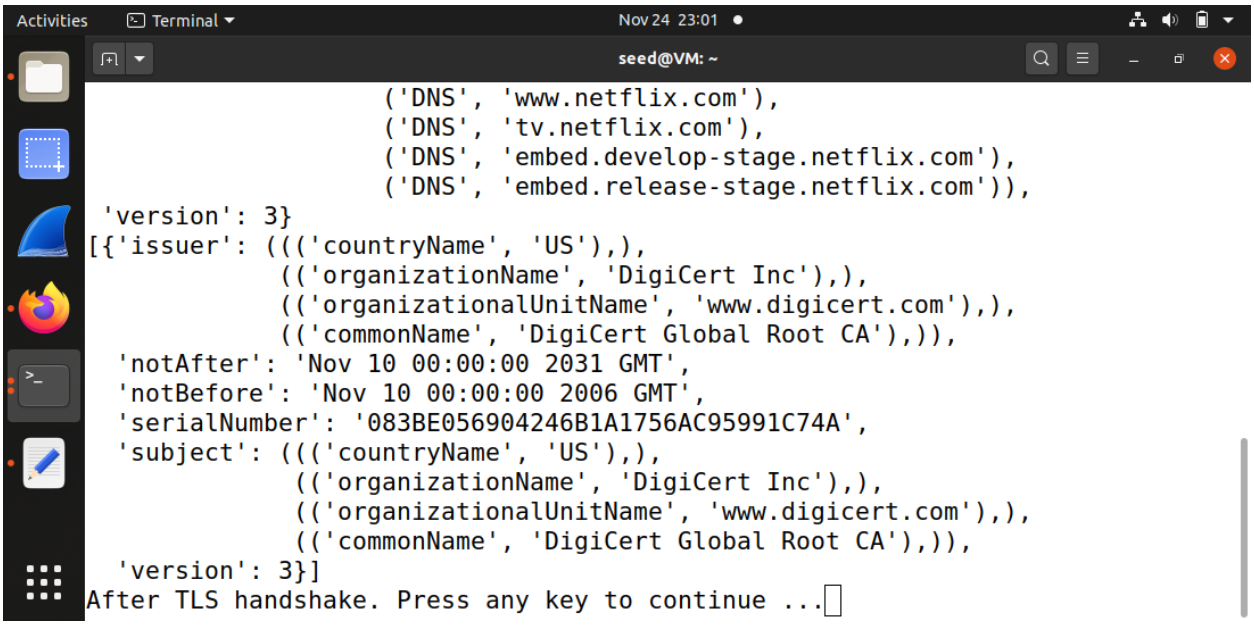
For

www.netflix.com



Using client program to figure out the CA certificate needed to be copied to the new folder.

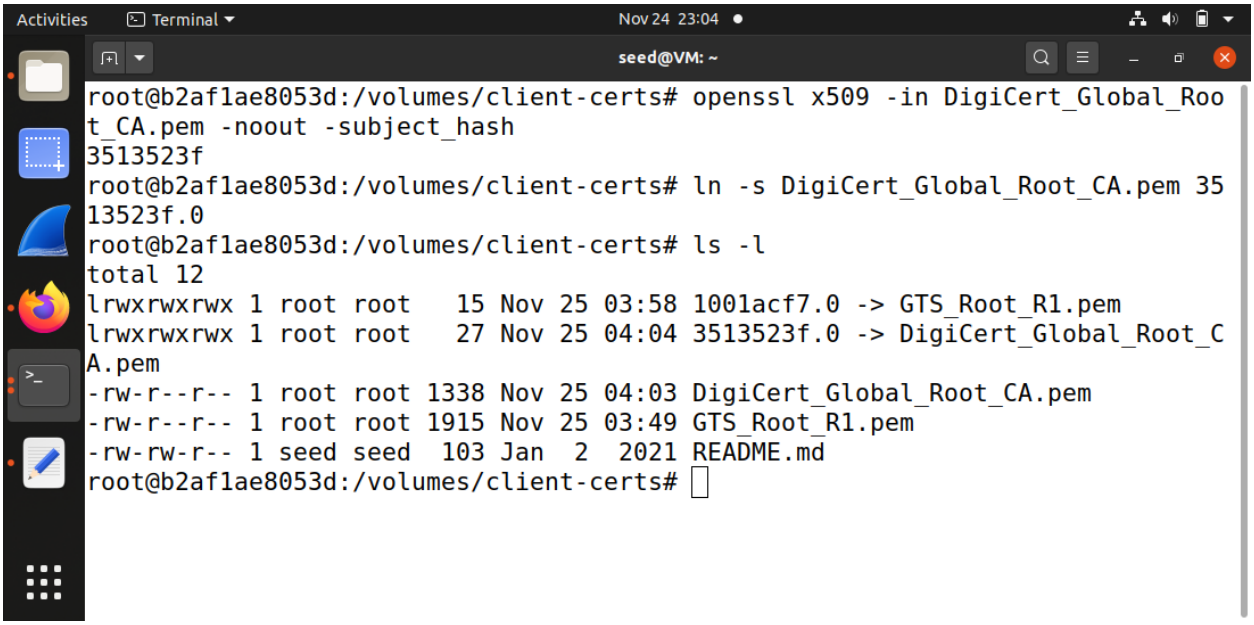




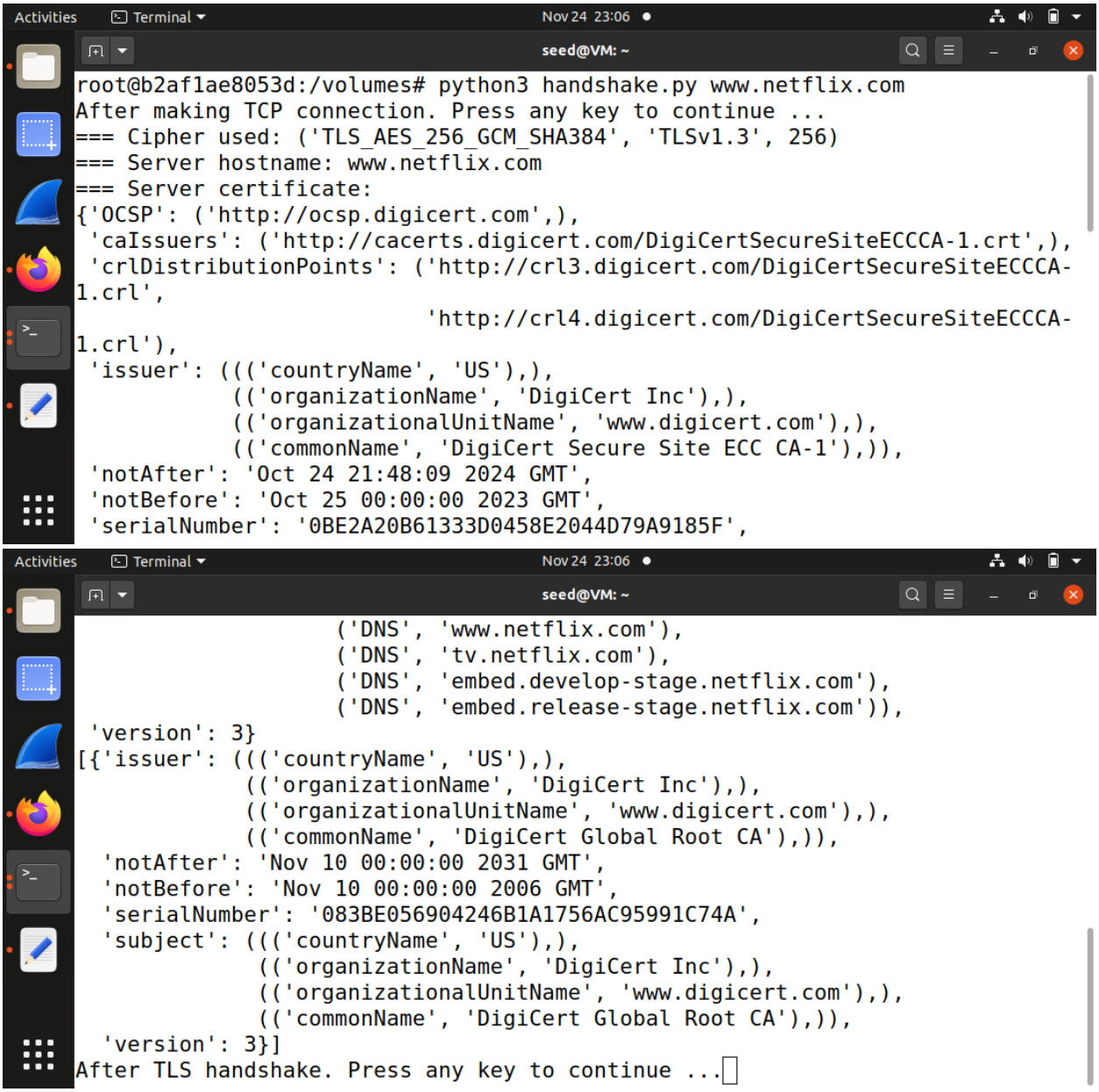
Copying to new folder

```
root@b2af1ae8053d:/volumes# cp /etc/ssl/certs/DigiCert_Global_Root_CA.pem /vo
lumes/client-certs/
root@b2af1ae8053d:/volumes#
```

After finding the CA certificate to copy and copying it the folder we used openssl command to generate a hash value for that certificate. We created a symbolic link for this.



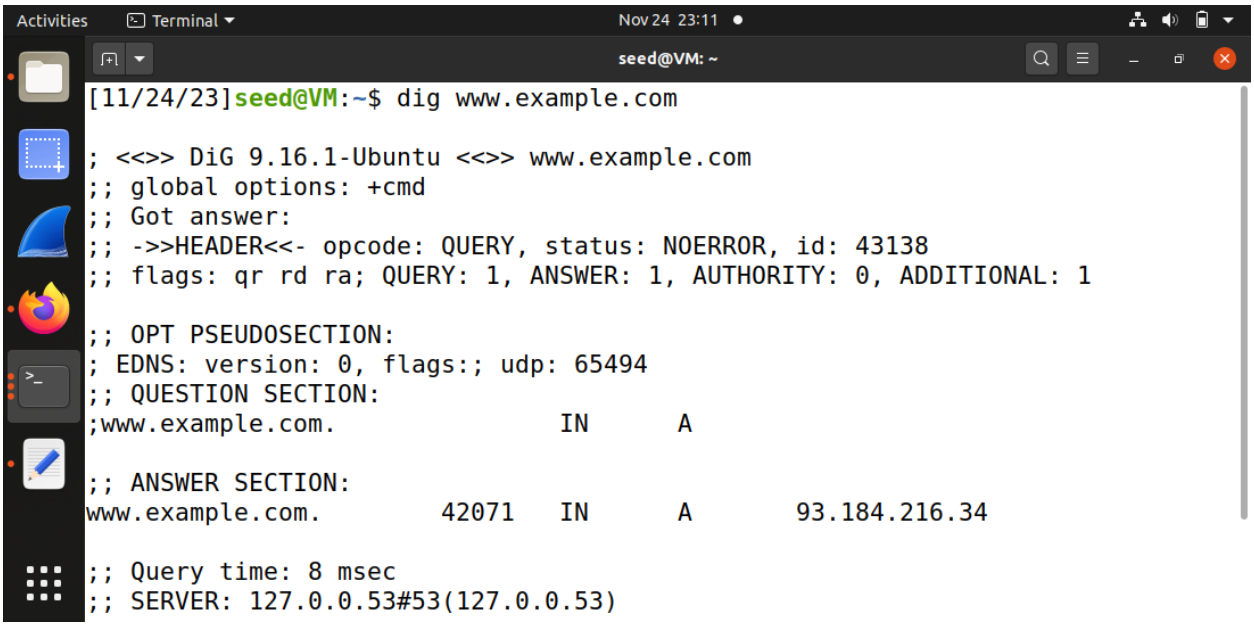
Now we run the “handshake.py” program.



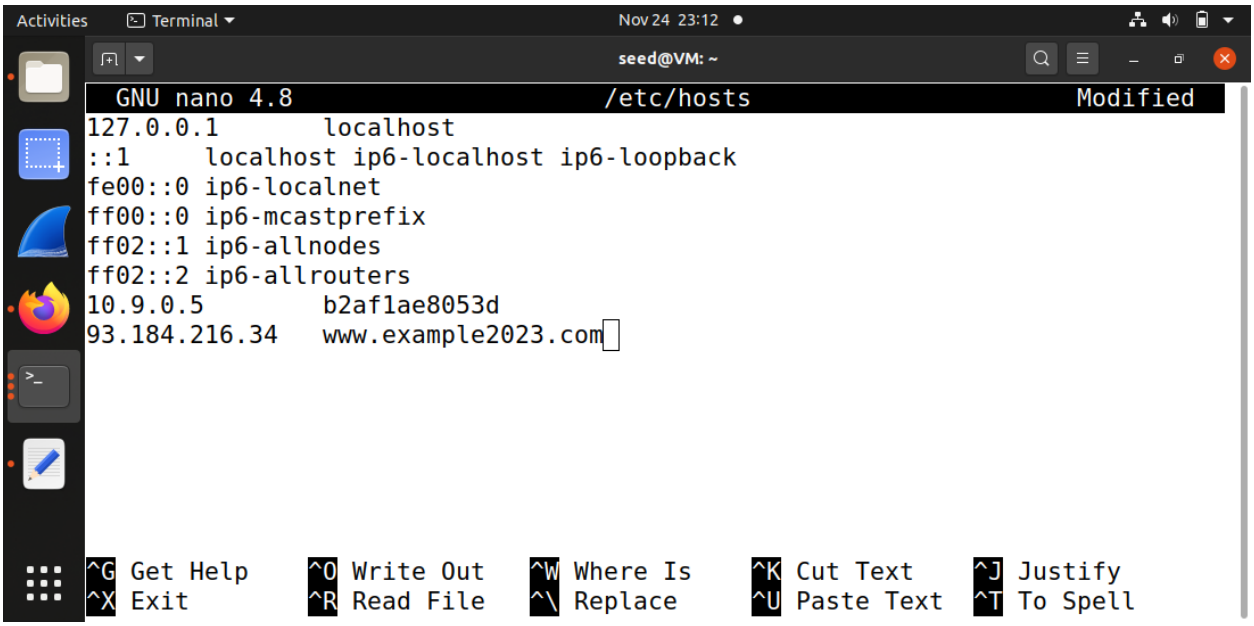
Task 1.c: Experiment with the hostname check

To help us understand the importance of hostname checks at the client side.

• Step 1: Get the IP address of www.example.com using the dig command, such as the following (you may want to run this command in your VM or host computers, because the dig command is not installed inside the container):

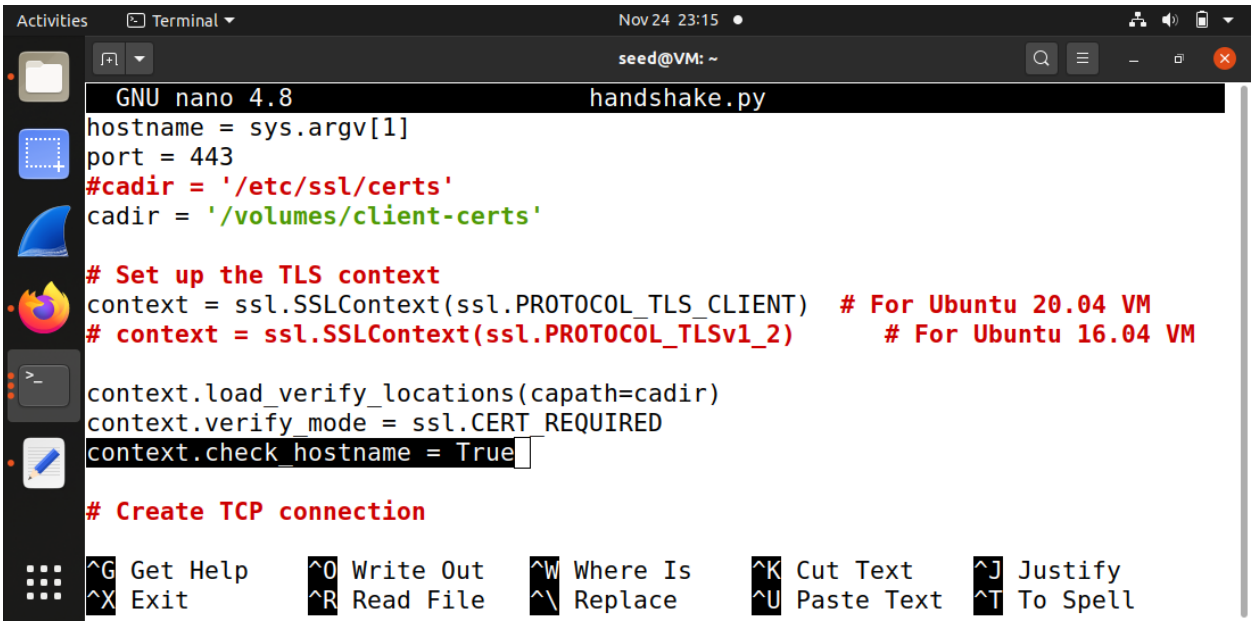


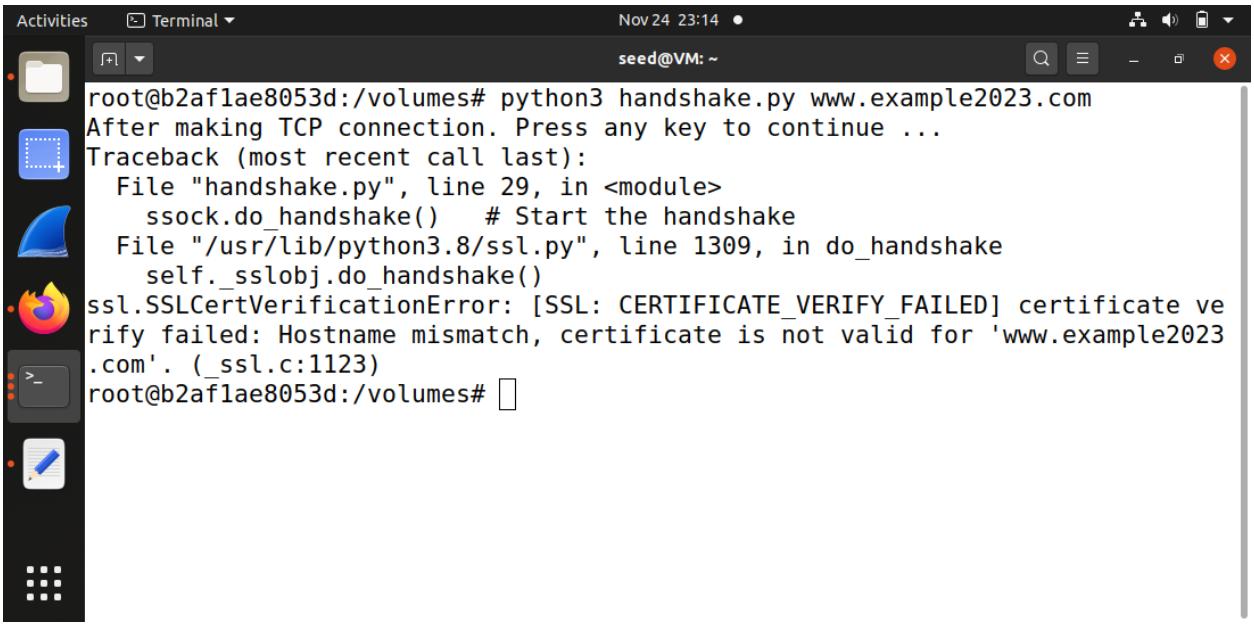
• Step 2: Modify the /etc/hosts file (inside the container), add the following entry at the end of the file (the IP address is what you get from the dig command). 93.184.216.34 www.example2023.com



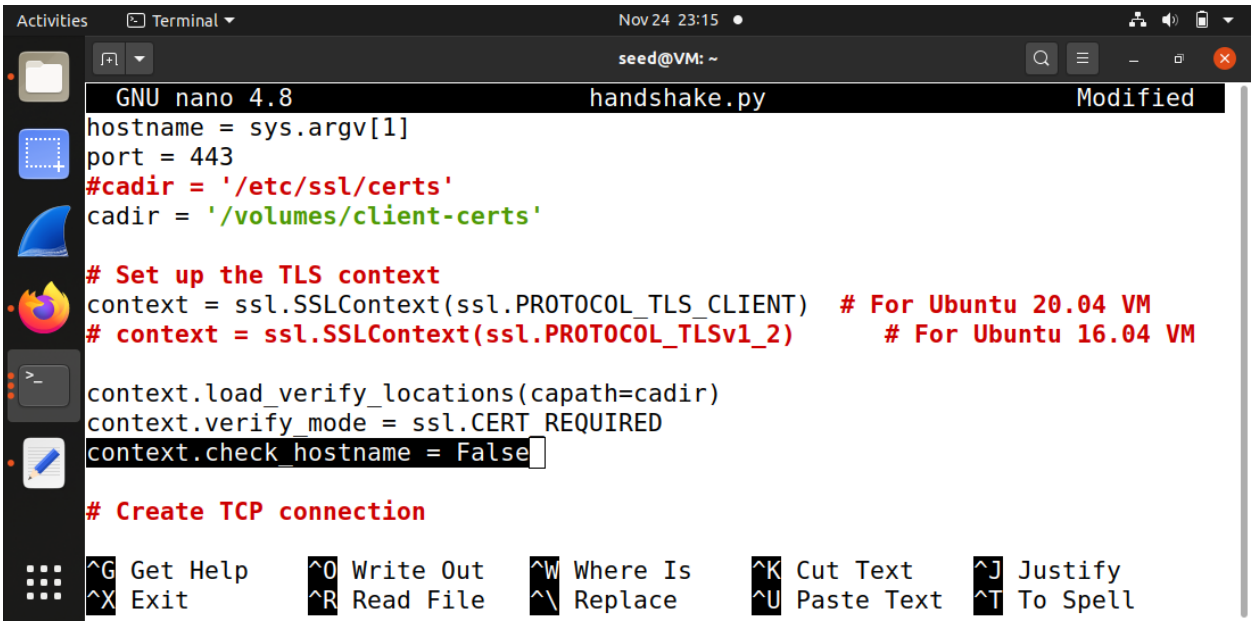
• Step 3: Switch the following line in the client program between True and False, and then connect your client program to www.example2023.com. Describe and explain your observation.

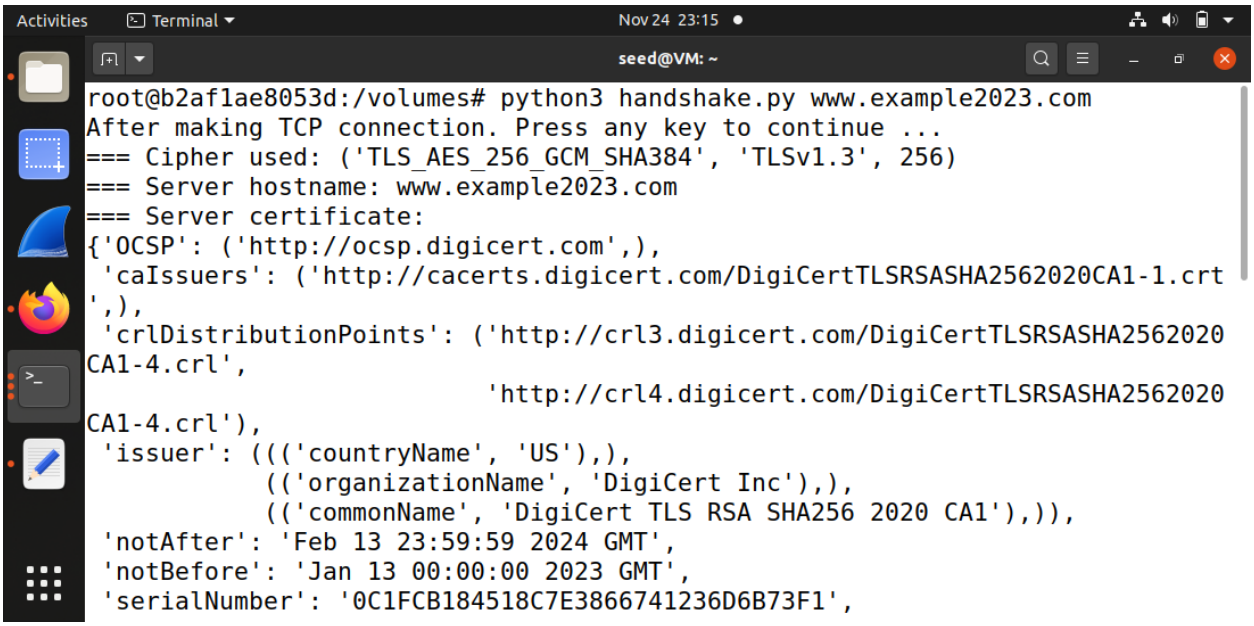
- So, when the check_hostname is “True” in the client program we get:





- So, when the check_hostname is “False” in the client program we get:





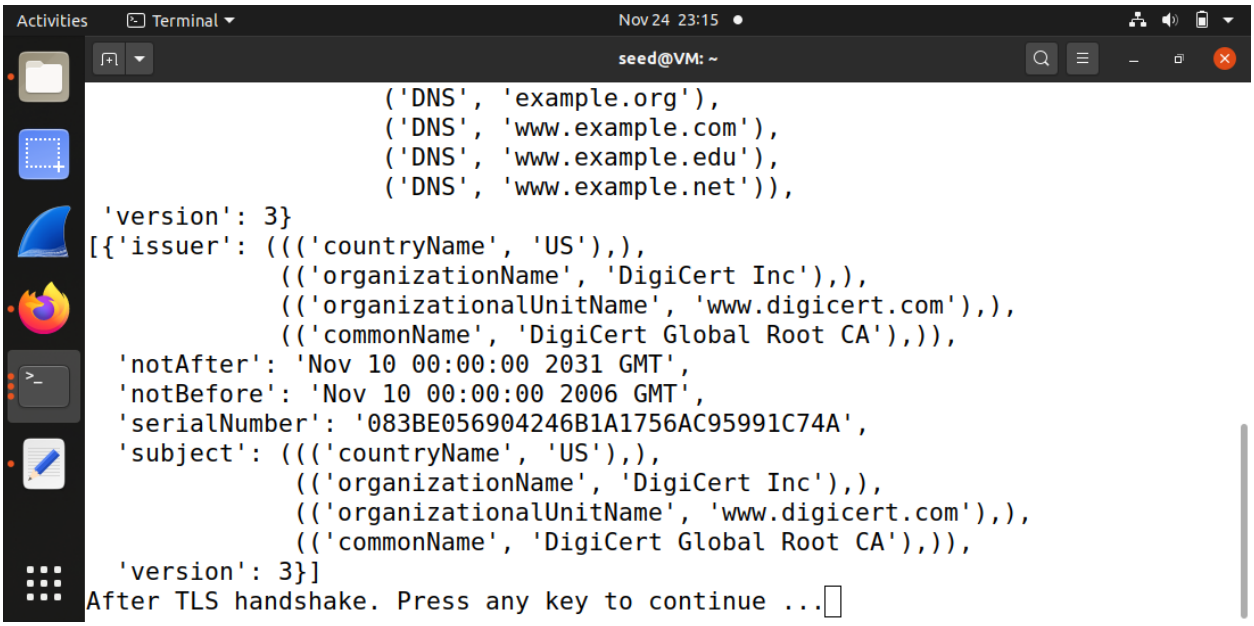
Observations from experiment:

When check_hostname is “True” we get an error and when it is “False” we get an output.

We get an error for True because we have given a different hostname than the one that is specified in the server certificate.

When giving check_hostname as False we get an output as it bypasses the need to check the validity of the hostname i.e. if the hostname given resembles the one in the certificate

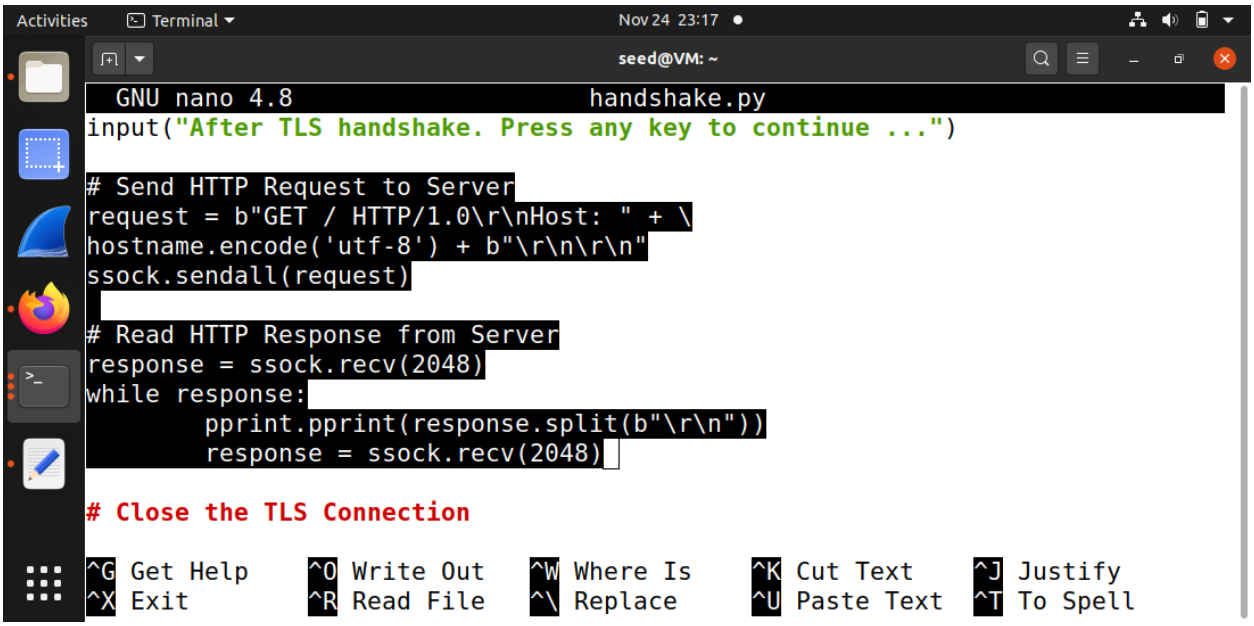
This shows the importance of domain name checking.



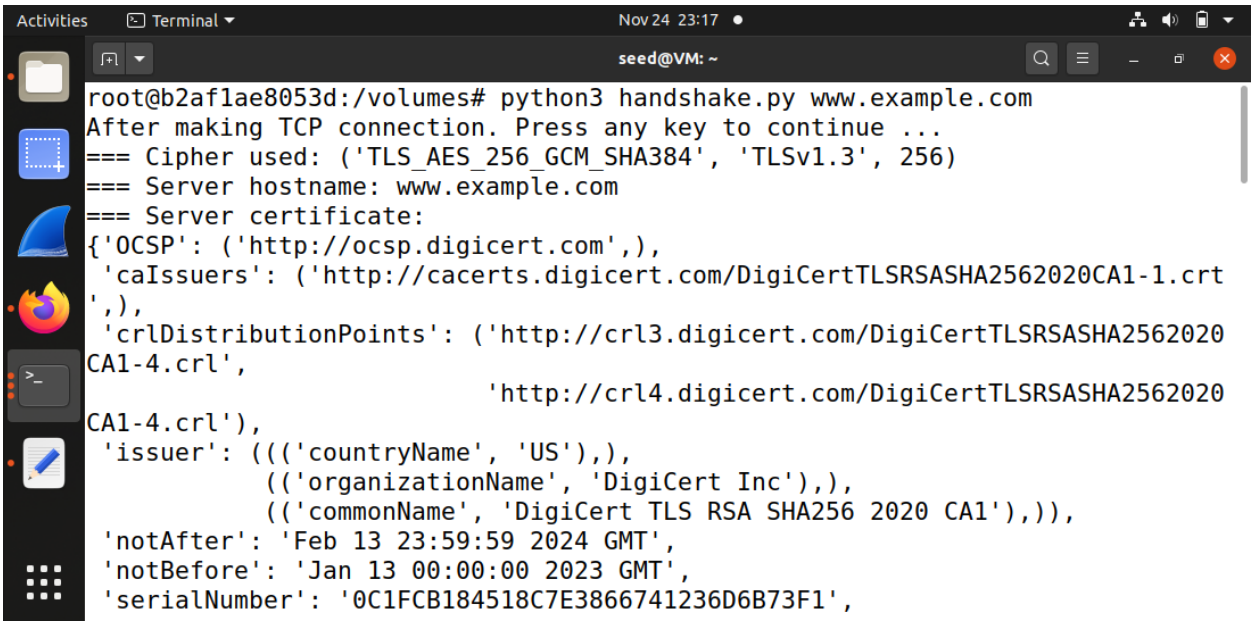
Task 1.d: Sending and getting Data

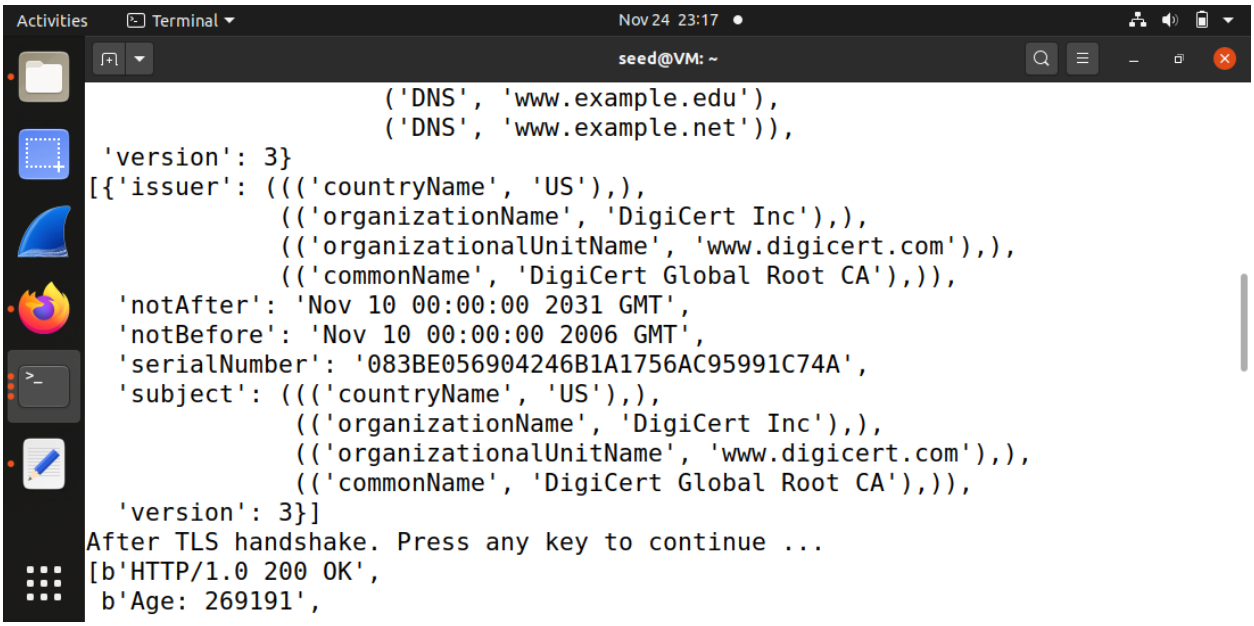
In this task, we will send data to the server and get its response.

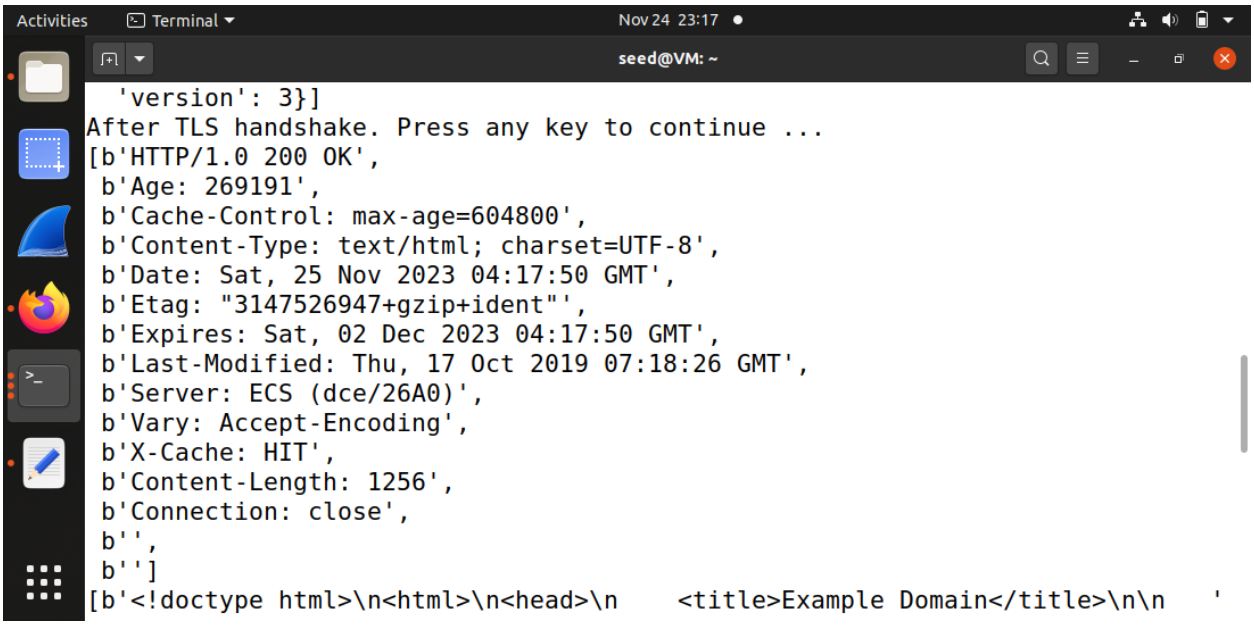
(1) Please add the data sending/receiving code to your client program and report your observation.

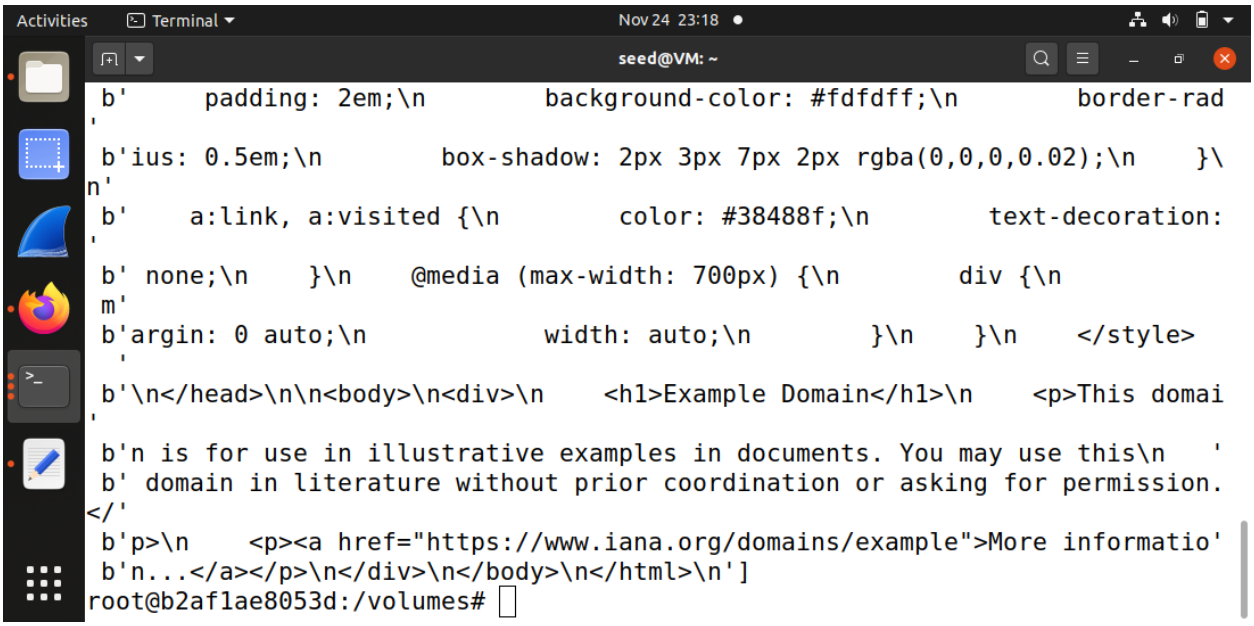


Observations:


Activities
Terminal
Nov 24 23:17
seed@VM: ~
root@b2af1ae8053d:/volumes# python3 handshake.py www.example.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
=== Server hostname: www.example.com
=== Server certificate:
{'OCSP': ('http://ocsp.digicert.com',),
'caIssuers': ('http://cacerts.digicert.com/DigiCertTLSRSASHA2562020CA1-1.crt
',),
'crlDistributionPoints': ('http://crl3.digicert.com/DigiCertTLSRSASHA2562020
CA1-4.crl',
'http://crl4.digicert.com/DigiCertTLSRSASHA2562020
CA1-4.crl'),
'issuer': ((('countryName', 'US'),),
(('organizationName', 'DigiCert Inc'),),
(('commonName', 'DigiCert TLS RSA SHA256 2020 CA1'),)),
'notAfter': 'Feb 13 23:59:59 2024 GMT',
'notBefore': 'Jan 13 00:00:00 2023 GMT',
'serialNumber': '0C1FCB184518C7E3866741236D6B73F1',

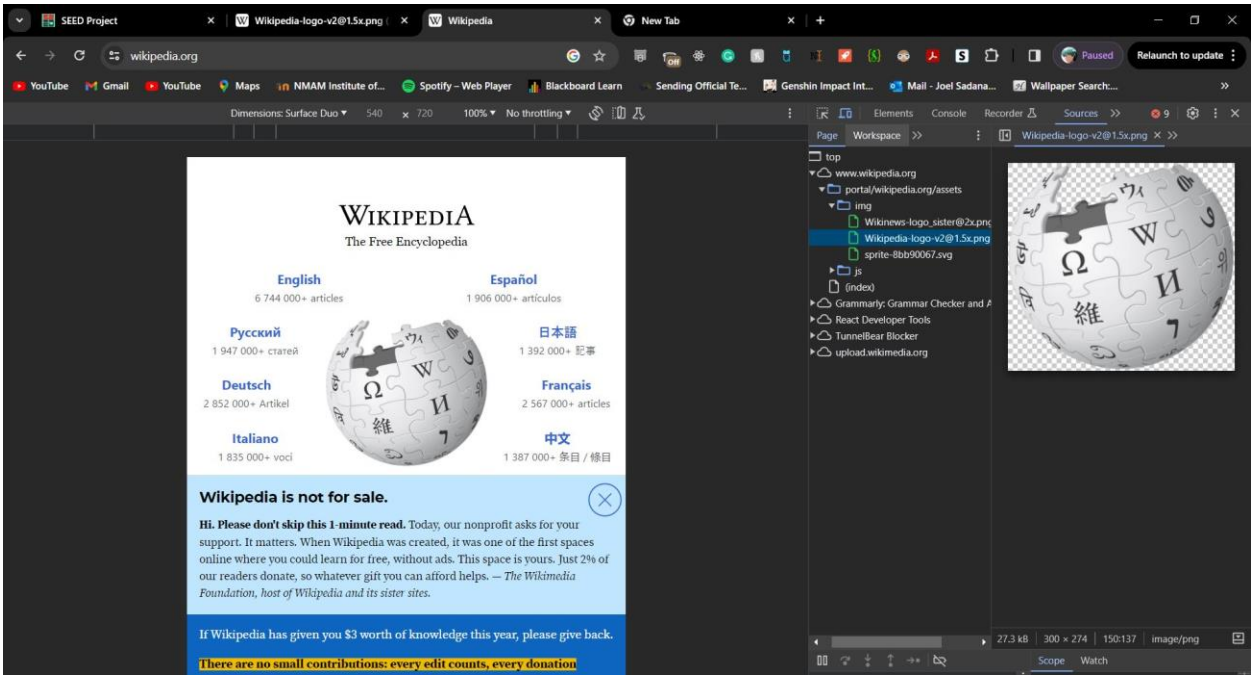

Activities
Terminal
Nov 24 23:17
seed@VM: ~
('DNS', 'www.example.edu'),
('DNS', 'www.example.net')),
'version': 3}
[{'issuer': ((('countryName', 'US'),),
(('organizationName', 'DigiCert Inc'),),
(('organizationalUnitName', 'www.digicert.com'),),
(('commonName', 'DigiCert Global Root CA'),)),
'notAfter': 'Nov 10 00:00:00 2031 GMT',
'notBefore': 'Nov 10 00:00:00 2006 GMT',
'serialNumber': '083BE056904246B1A1756AC95991C74A',
'subject': ((('countryName', 'US'),),
(('organizationName', 'DigiCert Inc'),),
(('organizationalUnitName', 'www.digicert.com'),),
(('commonName', 'DigiCert Global Root CA'),)),
'version': 3}]
After TLS handshake. Press any key to continue ...
[b'HTTP/1.0 200 OK',
b'Age: 269191',

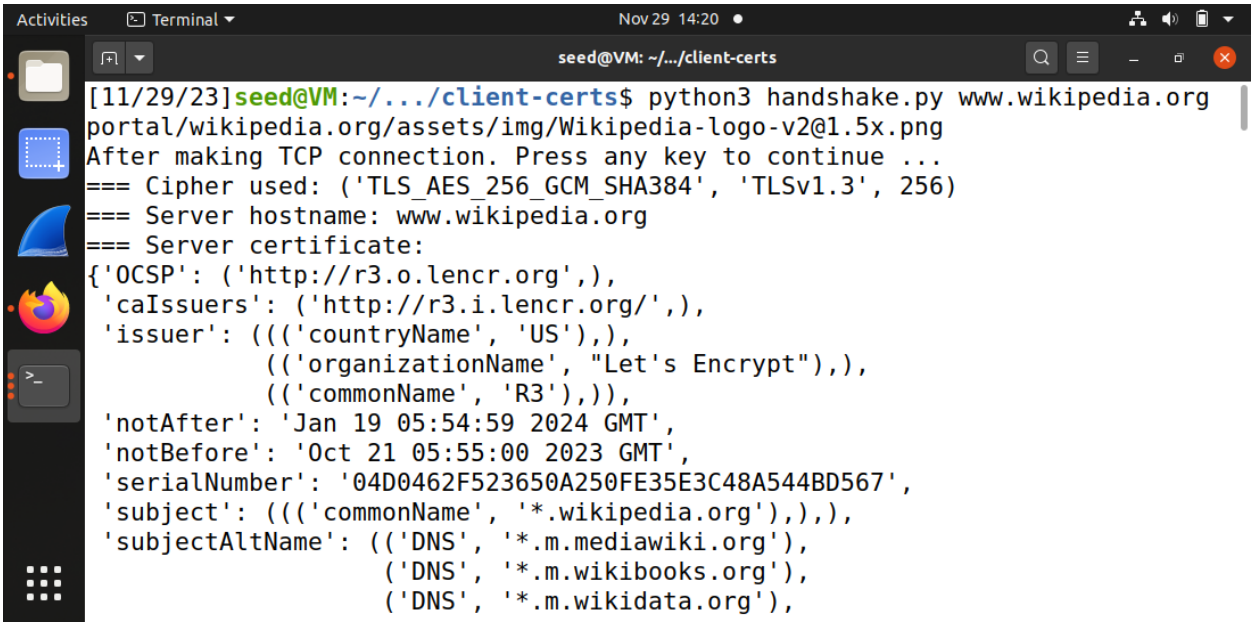


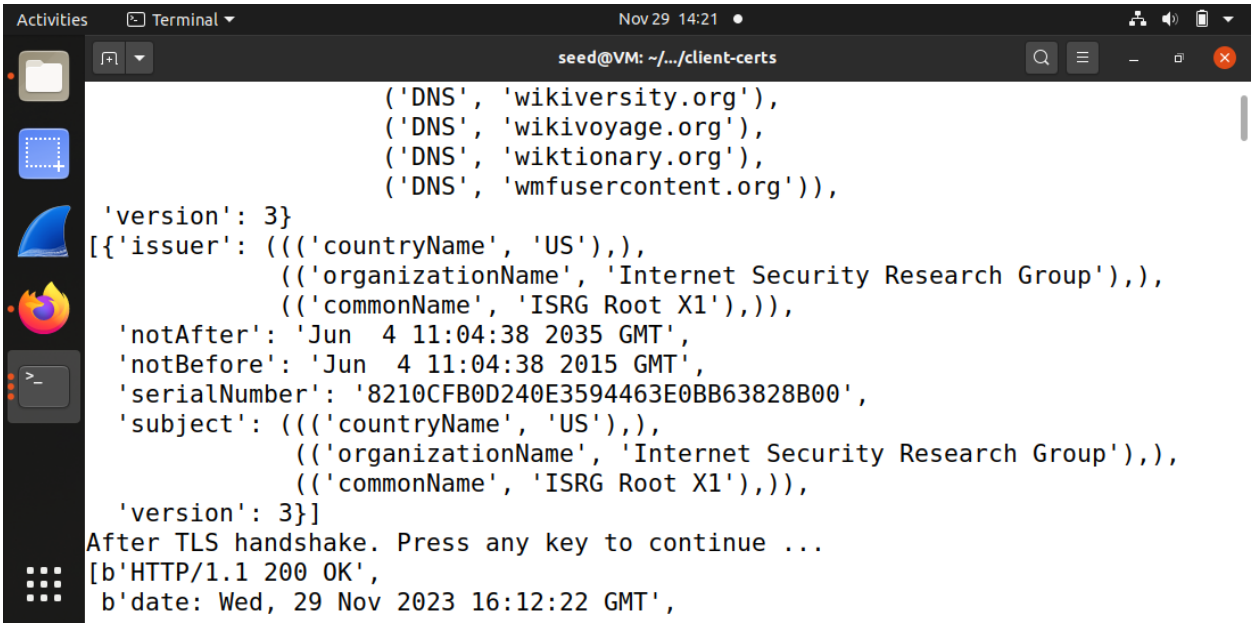


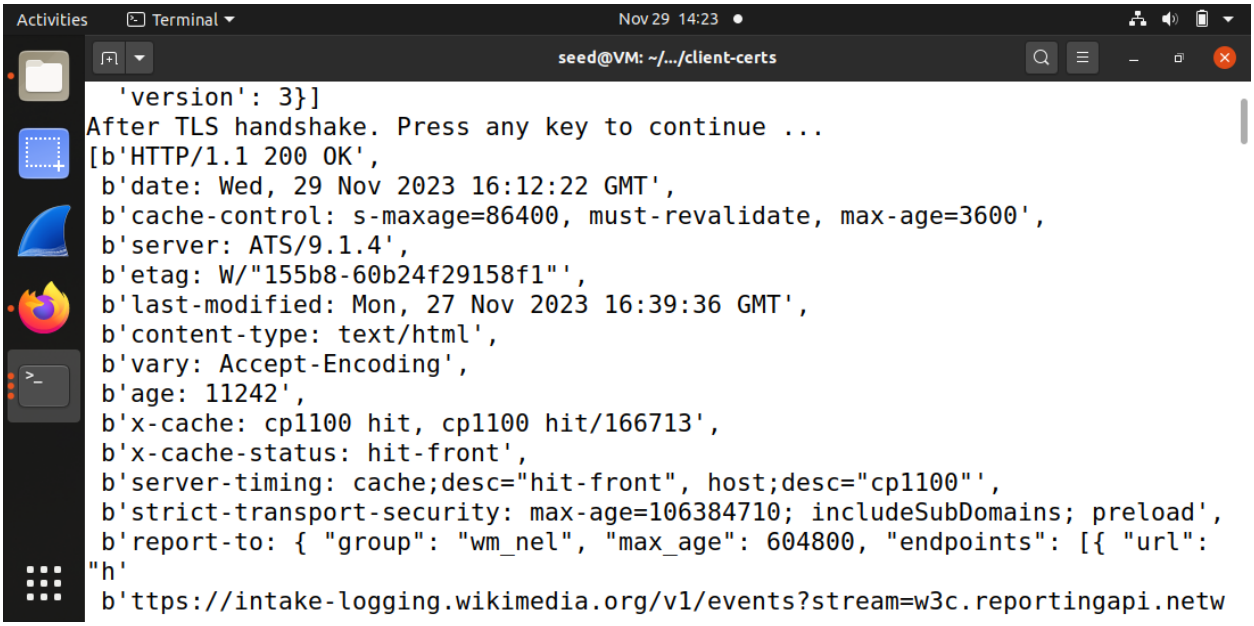
Using socket programming we sent a https get request to the server and we get a response from the server. The response is the page source code for the website "www.example.com" in html.

(2) Please modify the HTTP request, so you can fetch an image file of your choice from an HTTPS server Here I am going to fetch the image file for "wikipedia.com".


SEED Project
Wikipedia-logo-v2@1.5x.png
Wikipedia
New Tab
wikipedia.org
Relaunch to update
WIKIPEDIA
The Free Encyclopedia
English
6 744 000+ articles
Español
1 906 000+ artículos
Русский
1 947 000+ статей
日本語
1 392 000+ 記事
Deutsch
2 852 000+ Artikel
Français
2 567 000+ articles
Italiano
1 835 000+ voci
中文
1 387 000+ 条目 / 條目
Wikipedia is not for sale.
Hi. Please don't skip this 1-minute read. Today, our nonprofit asks for your support. It matters. When Wikipedia was created, it was one of the first spaces online where you could learn for free, without ads. This space is yours. Just 2% of our readers donate, so whatever gift you can afford helps. — The Wikimedia Foundation, host of Wikipedia and its sister sites.
If Wikipedia has given you $3 worth of knowledge this year, please give back.
There are no small contributions: every edit counts, every donation.


Activities
Terminal
Nov 29 14:20
seed@VM: ~/.../client-certs
[11/29/23]seed@VM:~/.../client-certs$ python3 handshake.py www.wikipedia.org portal/wikipedia.org/assets/img/Wikipedia-logo-v2@1.5x.png
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
=== Server hostname: www.wikipedia.org
=== Server certificate:
{'OCSP': ('http://r3.o.lencr.org',),
 'caIssuers': ('http://r3.i.lencr.org/',),
 'issuer': ((('countryName', 'US'),),
            (('organizationName', "Let's Encrypt"),),
            (('commonName', 'R3'),)),
 'notAfter': 'Jan 19 05:54:59 2024 GMT',
 'notBefore': 'Oct 21 05:55:00 2023 GMT',
 'serialNumber': '04D0462F523650A250FE35E3C48A544BD567',
 'subject': ((('commonName', '*.wikipedia.org'),),),
 'subjectAltName': (('DNS', '*.m.mediawiki.org'),
                    ('DNS', '*.m.wikibooks.org'),
                    ('DNS', '*.m.wikidata.org'),





Here we can see we get the html from the Wikipedia page.

Task 2 - TLS Server

Abstract:

This report documents the execution and observations made in Task 2 - TLS Server, focusing on implementing a TLS server, testing server-client interaction using a browser, and configuring certificates with multiple hostnames. The tasks encompassed understanding TLS server implementation, certificate handling, and hostname variations in server certificates.

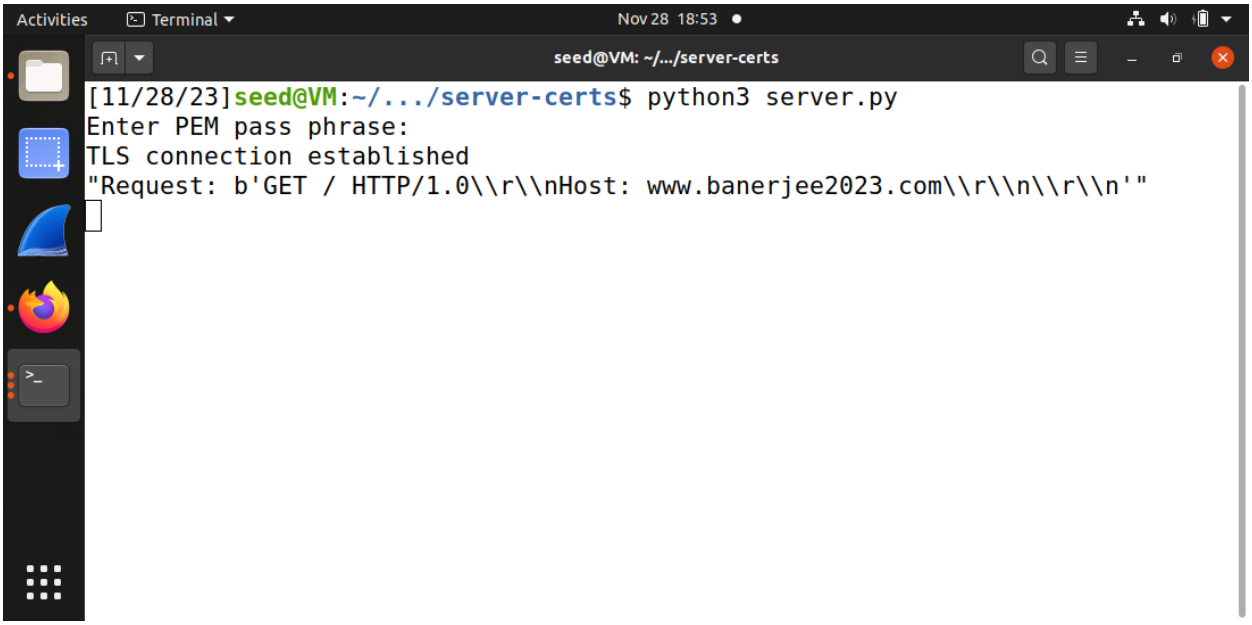
1. Introduction:

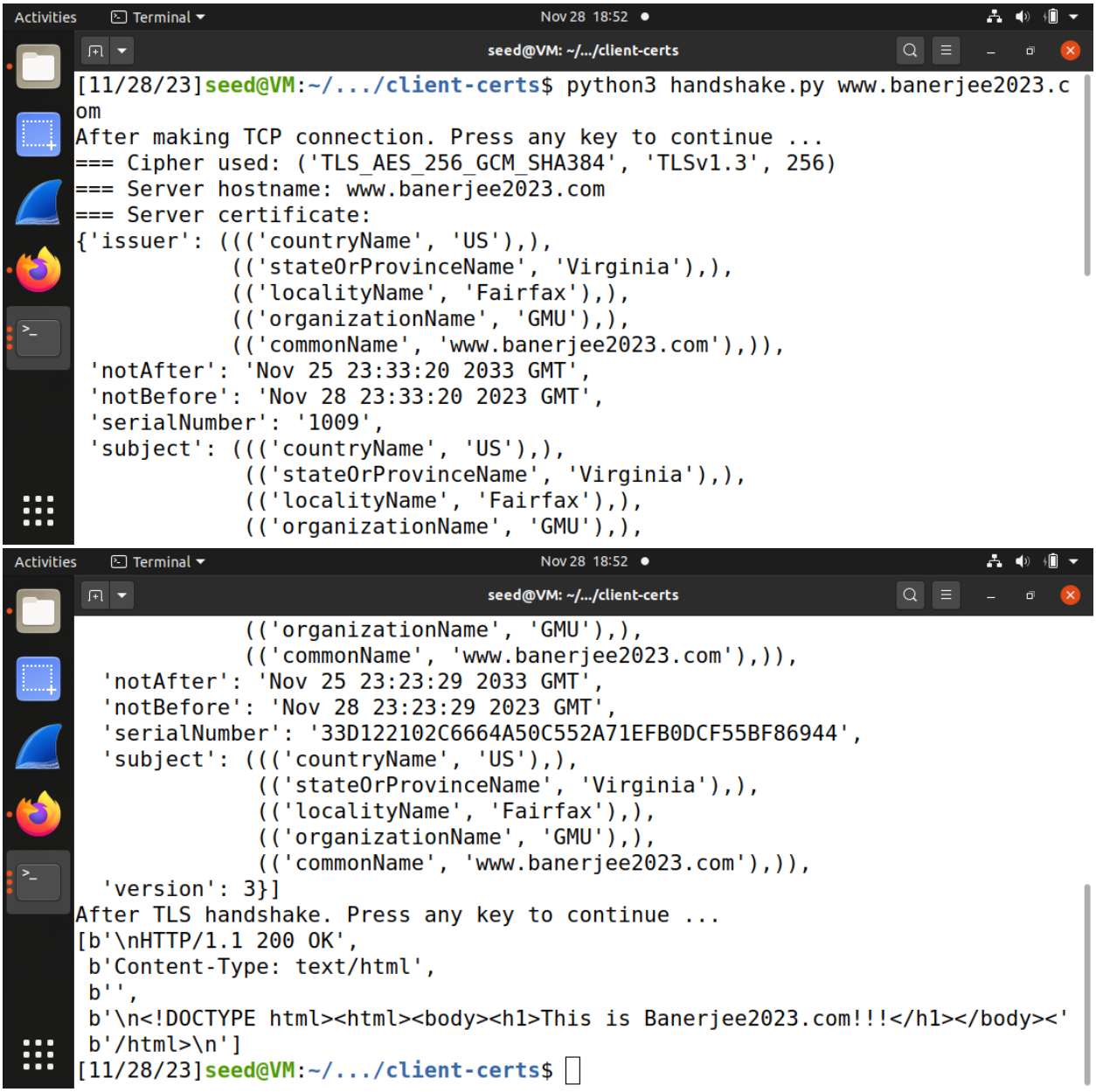
The primary objective of this lab experiment was to establish a simple TLS server, perform testing with clients, particularly browsers, and explore the configuration of server certificates to support multiple hostnames. Task 2 aimed to comprehend TLS server setup, certificate validation, and management of Subject Alternative Names (SAN) in certificates.

2. Methodology:

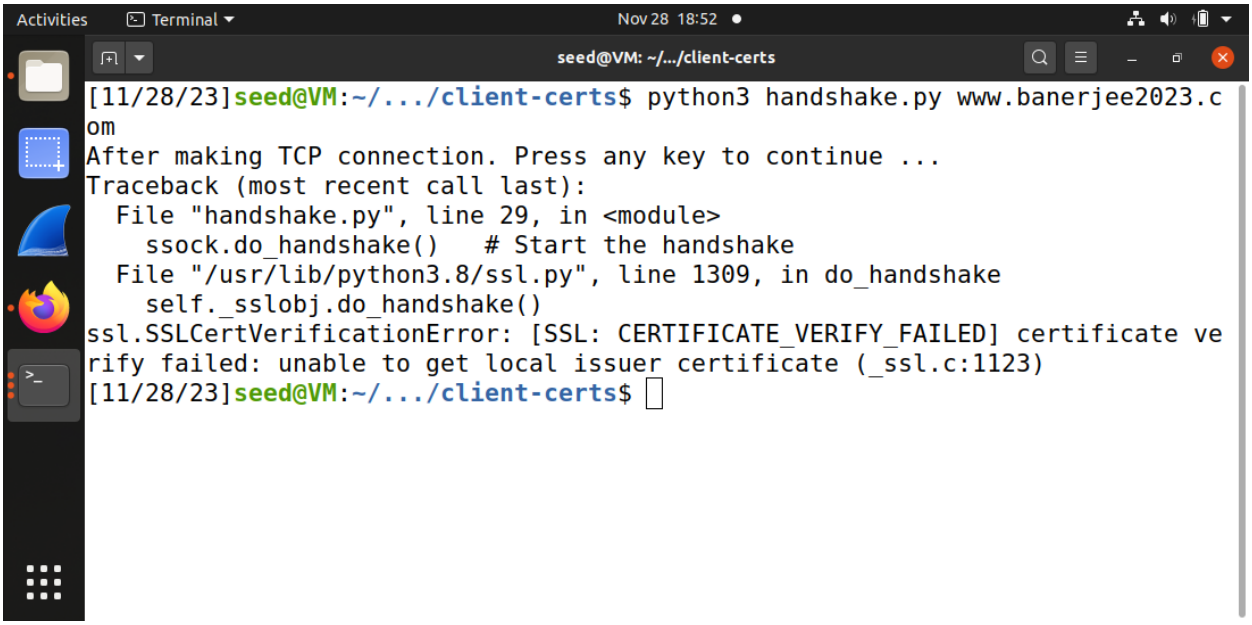
Task 2.a: Simple TLS Server Implementation

To set up a simple TLS server, I first organized the necessary CA.crt file in the **/client-certs** directory, ensuring its availability for server-client verification. I created a server program, **server.py**, configuring it to run on port 4433, utilizing the hostname "www.banerjee2023.com" with its associated IP in the local **/etc/hosts** file. Running the server, I initiated a TLS handshake from the client (handshake.py), specifying the "cadir" path as "/etc/ssl/certs/client-certs". This setup successfully established a connection between the server and client.





Switching the "cadir" path in the client to "/etc/ssl/certs" resulted in an error due to the absence of the necessary SSL Certificate in that directory. Consequently, the program threw an error due to the missing certificate.

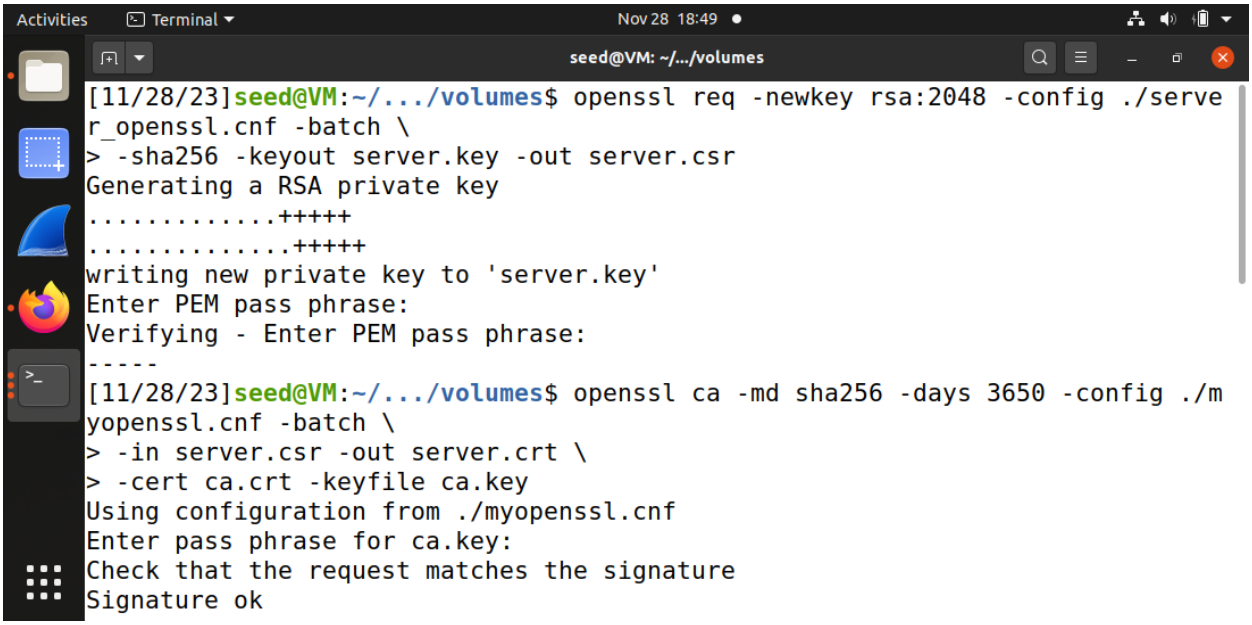


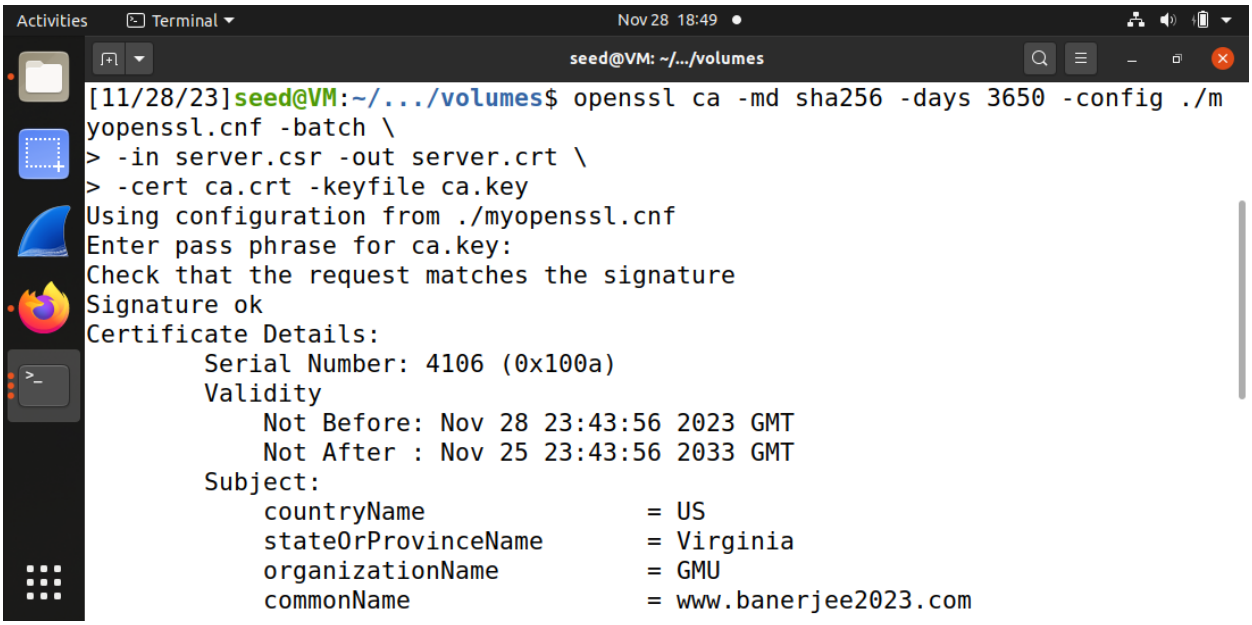
Task 2.b: Browser-based Testing of the Server Program

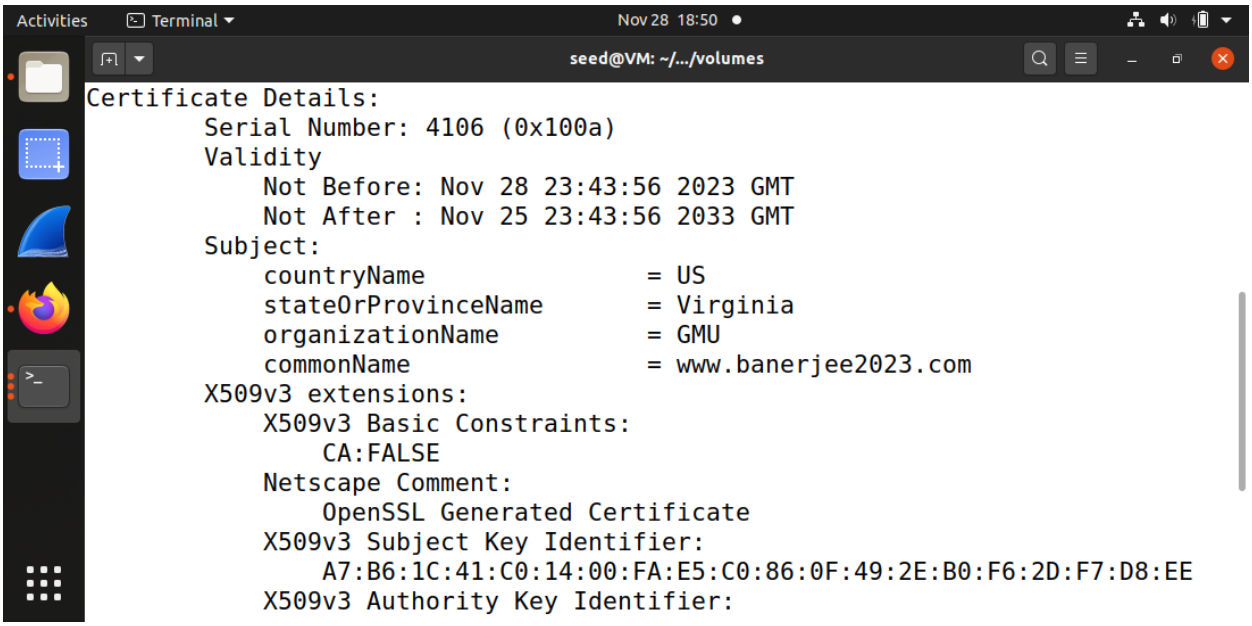
I performed browser testing after configuring the CA certificate in the browser during a prior PKI assignment. With the server running, I accessed the website hosted on port 4433, which is where the server was listening. The browser successfully displayed the content, affirming the communication between the TLS server and the browser.

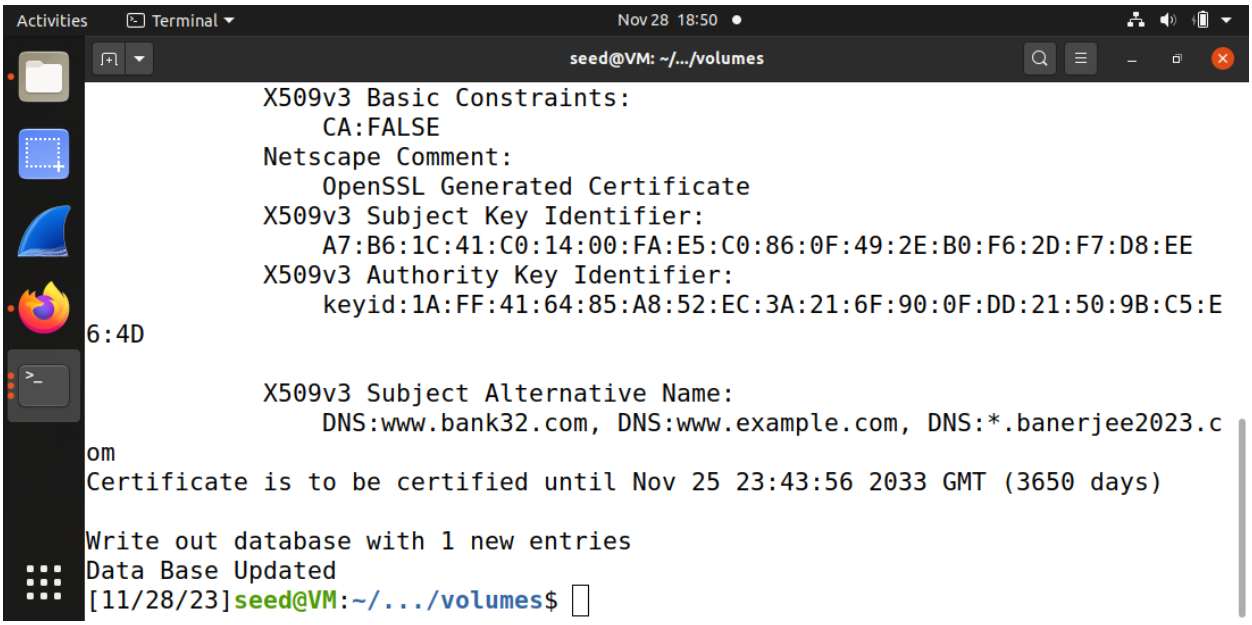
Task 2.c: Certificate Configuration with Multiple Names

Utilizing the Subject Alternative Name (SAN) feature, I configured the server certificate to support multiple hostnames. I included various hostnames such as www.banerjee2023.com, www.example.com, and *.banerjee2023.com in the SAN extension. Testing this setup, I accessed the server through different browsers, each pointing to distinct hostnames within the banerjee2023.com domain. This test illustrated the server's ability to accommodate multiple hostnames within its domain.




Activities
Terminal
Nov 28 18:49
seed@VM: ~/.../volumes
[11/28/23]seed@VM:~/.../volumes$ openssl ca -md sha256 -days 3650 -config ./myopenssl.cnf -batch \
> -in server.csr -out server.crt \
> -cert ca.crt -keyfile ca.key
Using configuration from ./myopenssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
Serial Number: 4106 (0x100a)
Validity
Not Before: Nov 28 23:43:56 2023 GMT
Not After : Nov 25 23:43:56 2033 GMT
Subject:
countryName = US
stateOrProvinceName = Virginia
organizationName = GMU
commonName = www.banerjee2023.com


Activities
Terminal
Nov 28 18:50
seed@VM: ~/.../volumes
Certificate Details:
Serial Number: 4106 (0x100a)
Validity
Not Before: Nov 28 23:43:56 2023 GMT
Not After : Nov 25 23:43:56 2033 GMT
Subject:
countryName = US
stateOrProvinceName = Virginia
organizationName = GMU
commonName = www.banerjee2023.com
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
Netscape Comment:
OpenSSL Generated Certificate
X509v3 Subject Key Identifier:
A7:B6:1C:41:C0:14:00:FA:E5:C0:86:0F:49:2E:B0:F6:2D:F7:D8:EE
X509v3 Authority Key Identifier:



As shown below multiple hostnames with multiple browsers displaying www.banerjee2023.com

Conclusion:

- **Summary of Findings:** Summarized key observations and achievements in accomplishing the objectives outlined in Task 2.
- **Objective Achievement:** Evaluated whether the experiment achieved its intended learning objectives concerning TLS server setup and certificate management.

Team Members:
Task 1:
Abhijeet Amitava Banerjee (G01349260)
Joel Sadanand Samson (G01352483)
Task 2:
Soham Patil (G01390169)
Mandar Suresh Chaudhari (G01393699)