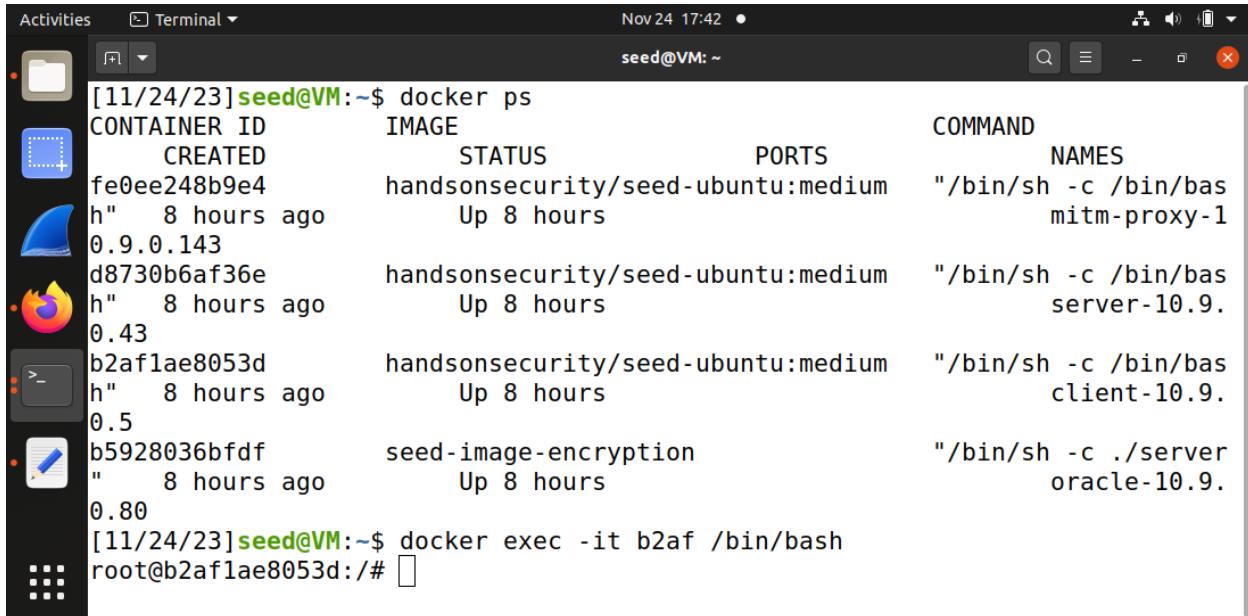


Transport Layer Protocol

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

Container setup shown below:



The screenshot shows a terminal window titled "seed@VM: ~" running on a Linux desktop. The window title bar includes the date and time: "Nov 24 17:42". The terminal displays the output of the "docker ps" command, listing five containers. The containers are based on the "handsonsecurity/seed-ubuntu:medium" image and are named "mitm-proxy-1", "server-10.9.", "client-10.9.", and "oracle-10.9.". One container, "seed-image-encryption", is running a custom command. Below this, a command "docker exec -it b2af /bin/bash" is run, and the user is prompted with "root@b2af1ae8053d: #".

CONTAINER ID	IMAGE	COMMAND
CREATED	STATUS	PORTS
fe0ee248b9e4 h" 8 hours ago 0.9.0.143	handsonsecurity/seed-ubuntu:medium Up 8 hours	"/bin/sh -c /bin/bas mitm-proxy-1
d8730b6af36e h" 8 hours ago 0.43	handsonsecurity/seed-ubuntu:medium Up 8 hours	"/bin/sh -c /bin/bas server-10.9.
b2af1ae8053d h" 8 hours ago 0.5	handsonsecurity/seed-ubuntu:medium Up 8 hours	"/bin/sh -c /bin/bas client-10.9.
b5928036bfdf " 8 hours ago 0.80	seed-image-encryption Up 8 hours	"/bin/sh -c ./server oracle-10.9.

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

```
Activities Terminal Nov 24 18:45 • seed@VM: ~
root@b2af1ae8053d:/volumes# python3 handshake.py www.google.com
After making TCP connection. Press any key to continue ...
==> Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
==> Server hostname: www.google.com
==> Server certificate:
{'OCSP': ('http://ocsp.pki.goog/gts1c3',),
 'caIssuers': ('http://pki.goog/repo/certs/gts1c3.der',),
 'crlDistributionPoints': ('http://crls.pki.goog/gts1c3/zdATt0Ex_Fk.crl',),
 'issuer': (((('countryName', 'US'),),
              (('organizationName', 'Google Trust Services LLC'),),
              (('commonName', 'GTS CA 1C3'))),
             {'notAfter': 'Jan 15 11:24:56 2024 GMT',
              'notBefore': 'Oct 23 11:24:57 2023 GMT',
              'serialNumber': '5DABB4403AC4AEC6125E276EDF1965F9',
              'subject': (((('commonName', 'www.google.com'),),
                           {'version': 3}
                           [{}{'issuer': (((('countryName', 'US'),),
                               (('organizationName', 'Google Trust Services LLC'),),
                               (('commonName', 'GTS Root R1'))),
                             {'notAfter': 'Jun 22 00:00:00 2036 GMT',
                              'notBefore': 'Jun 22 00:00:00 2016 GMT',
                              'serialNumber': '6E47A9C54B470C0DEC33D089B91CF4E1',
                              'subject': (((('countryName', 'US'),),
                                           (('organizationName', 'Google Trust Services LLC'),),
                                           (('commonName', 'GTS Root R1'))),
                                         {'version': 3})
                           After TLS handshake. Press any key to continue ...
root@b2af1ae8053d:/volumes#
```

```
Activities Terminal Nov 24 18:45 • seed@VM: ~
root@b2af1ae8053d:/volumes# python3 handshake.py www.google.com
After making TCP connection. Press any key to continue ...
==> Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
==> Server hostname: www.google.com
==> Server certificate:
{'notAfter': 'Jan 15 11:24:56 2024 GMT',
 'notBefore': 'Oct 23 11:24:57 2023 GMT',
 'serialNumber': '5DABB4403AC4AEC6125E276EDF1965F9',
 'subject': (((('commonName', 'www.google.com'),),
              {'version': 3}
              [{}{'issuer': (((('countryName', 'US'),),
                  (('organizationName', 'Google Trust Services LLC'),),
                  (('commonName', 'GTS Root R1'))),
                {'notAfter': 'Jun 22 00:00:00 2036 GMT',
                 'notBefore': 'Jun 22 00:00:00 2016 GMT',
                 'serialNumber': '6E47A9C54B470C0DEC33D089B91CF4E1',
                 'subject': (((('countryName', 'US'),),
                              (('organizationName', 'Google Trust Services LLC'),),
                              (('commonName', 'GTS Root R1'))),
                           {'version': 3})
               After TLS handshake. Press any key to continue ...
root@b2af1ae8053d:/volumes#
```

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

We can't 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])
```

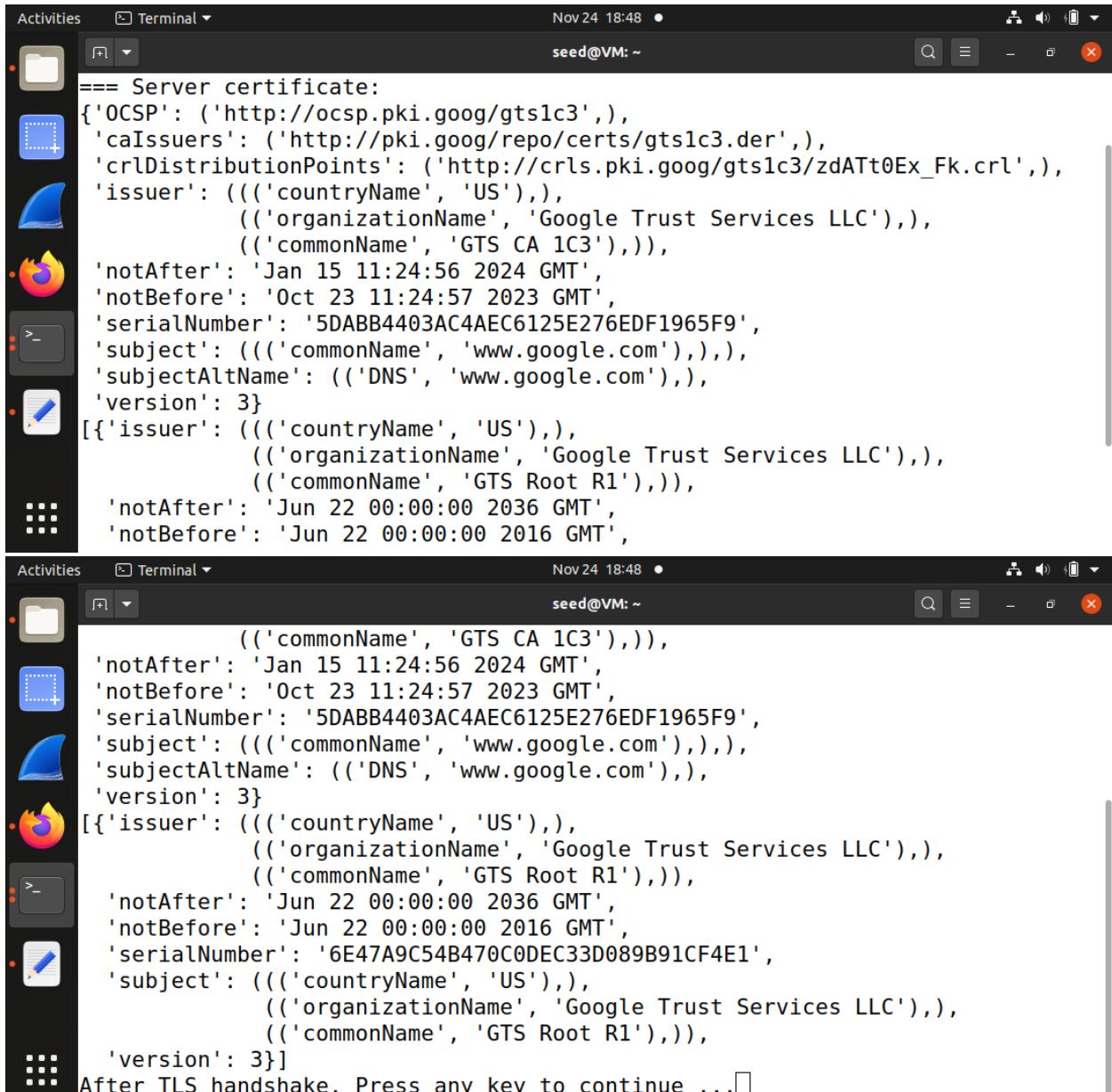
As we can see here when we ran our program with the google server we can see the output given. In the output under Cipher used we see which cipher is used.

```
Activities Terminal Nov 24 18:45 • seed@VM: ~
root@b2af1ae8053d:/volumes# python3 handshake.py www.google.com
After making TCP connection. Press any key to continue ...
==> Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
==> Server hostname: www.google.com
==> Server certificate:
```

- Please print out the server certificate in the program

In the handshake.py program the `pprint.pprint(ssock.getpeercert())` gives us the Server 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



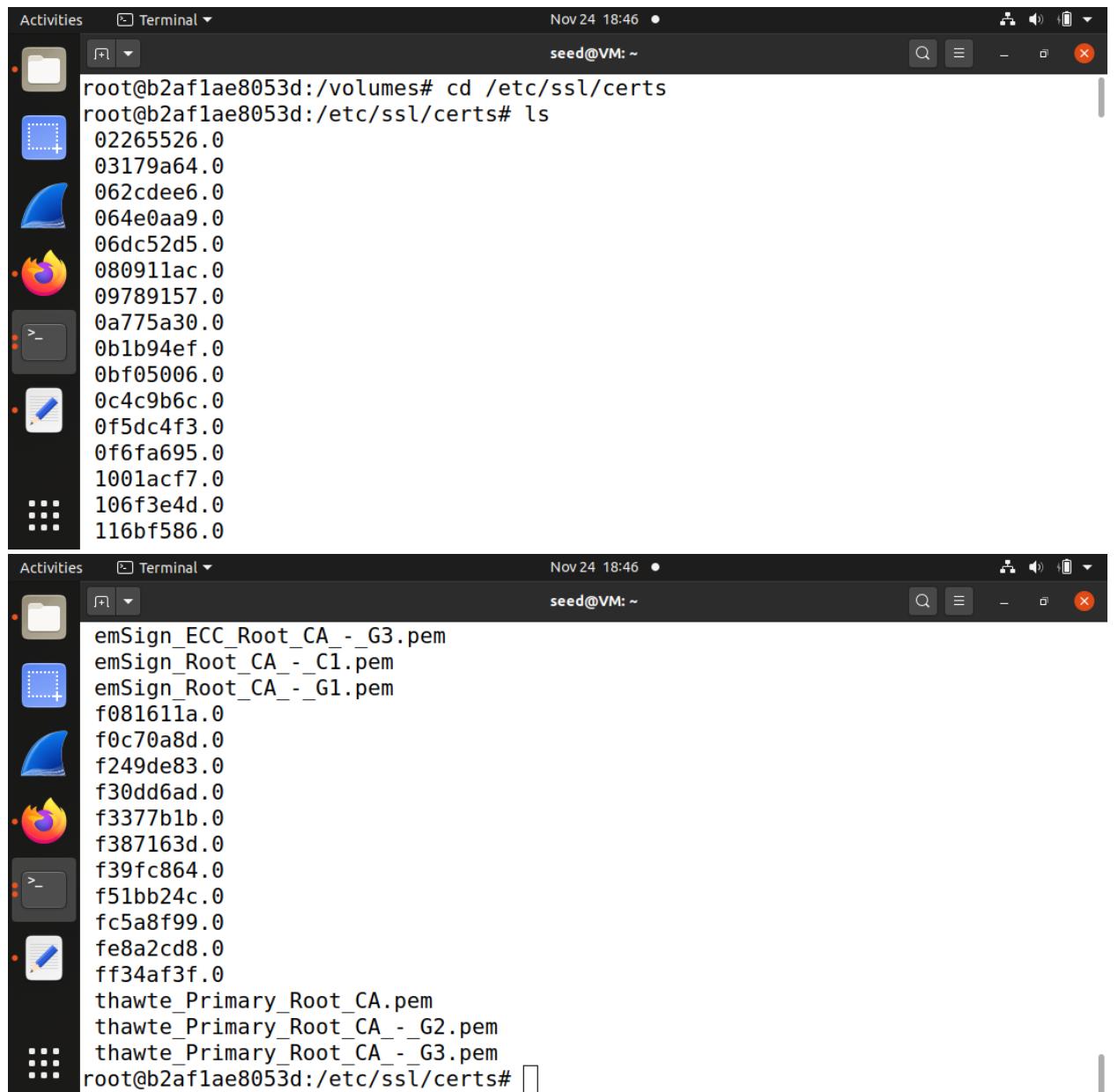
```
Activities Terminal Nov 24 18:48 seed@VM: ~
==== Server certificate:
{'OCSP': ('http://ocsp.pki.goog/gts1c3',),
 'caIssuers': ('http://pki.goog/repo/certs/gts1c3.der',),
 'crlDistributionPoints': ('http://crls.pki.goog/gts1c3/zdATt0Ex_Fk.crl',),
 'issuer': (((('countryName', 'US'),),
              (('organizationName', 'Google Trust Services LLC'),),
              (('commonName', 'GTS CA 1C3'))),
             'notAfter': 'Jan 15 11:24:56 2024 GMT',
             'notBefore': 'Oct 23 11:24:57 2023 GMT',
             'serialNumber': '5DABB4403AC4AEC6125E276EDF1965F9',
             'subject': (((('commonName', 'www.google.com'),),
                          'subjectAltName': (((('DNS', 'www.google.com'),),
                                         'version': 3}
[{'issuer': (((('countryName', 'US'),),
                  (('organizationName', 'Google Trust Services LLC'),),
                  (('commonName', 'GTS Root R1'))),
                 'notAfter': 'Jun 22 00:00:00 2036 GMT',
                 'notBefore': 'Jun 22 00:00:00 2016 GMT',
                 'version': 3}
[{'issuer': (((('countryName', 'US'),),
                  (('organizationName', 'Google Trust Services LLC'),),
                  (('commonName', 'GTS Root R1'))),
                 'notAfter': 'Jan 15 11:24:56 2024 GMT',
                 'notBefore': 'Oct 23 11:24:57 2023 GMT',
                 'serialNumber': '5DABB4403AC4AEC6125E276EDF1965F9',
                 'subject': (((('commonName', 'www.google.com'),),
                               'subjectAltName': (((('DNS', 'www.google.com'),),
                                             'version': 3}
[{'issuer': (((('countryName', 'US'),),
                  (('organizationName', 'Google Trust Services LLC'),),
                  (('commonName', 'GTS Root R1'))),
                 'notAfter': 'Jun 22 00:00:00 2036 GMT',
                 'notBefore': 'Jun 22 00:00:00 2016 GMT',
                 'serialNumber': '6E47A9C54B470C0DEC33D089B91CF4E1',
                 'subject': (((('countryName', 'US'),),
                               (('organizationName', 'Google Trust Services LLC'),),
                               (('commonName', 'GTS Root R1'))),
                               'version': 3}]
After TLS handshake. Press any key to continue ...
```

- 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 to verify the certificate is stored here.

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

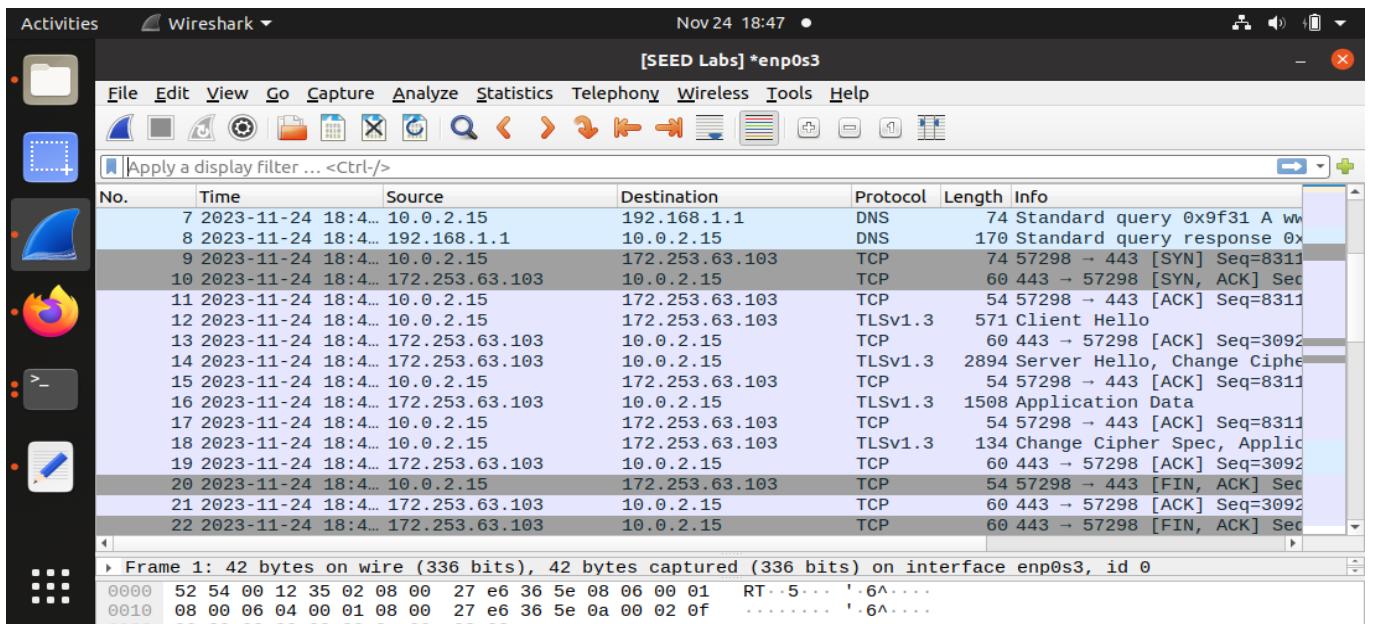


The image shows two terminal windows side-by-side. Both windows have a dark theme with light-colored text. The top window shows the command 'cd /etc/ssl/certs' followed by 'ls' and a list of files. The bottom window shows the same command and list of files, but includes some additional root CA certificates at the top of the list.

```
root@b2af1ae8053d:/Volumes# cd /etc/ssl/certs
root@b2af1ae8053d:/etc/ssl/certs# ls
02265526.0
03179a64.0
062cdee6.0
064e0aa9.0
06dc52d5.0
080911ac.0
09789157.0
0a775a30.0
0b1b94ef.0
0bf05006.0
0c4c9b6c.0
0f5dc4f3.0
0f6fa695.0
1001acf7.0
106f3e4d.0
116bf586.0

Activities Terminal Nov 24 18:46 seed@VM: ~
root@b2af1ae8053d:/etc/ssl/certs# ls
emSign_ECC_Root_CA - G3.pem
emSign_Root_CA - C1.pem
emSign_Root_CA - G1.pem
f081611a.0
f0c70a8d.0
f249de83.0
f30dd6ad.0
f3377b1b.0
f387163d.0
f39fc864.0
f51bb24c.0
fc5a8f99.0
fe8a2cd8.0
ff34af3f.0
thawte_Primary_Root_CA.pem
thawte_Primary_Root_CA - G2.pem
thawte_Primary_Root_CA - G3.pem
root@b2af1ae8053d:/etc/ssl/certs#
```

- 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.



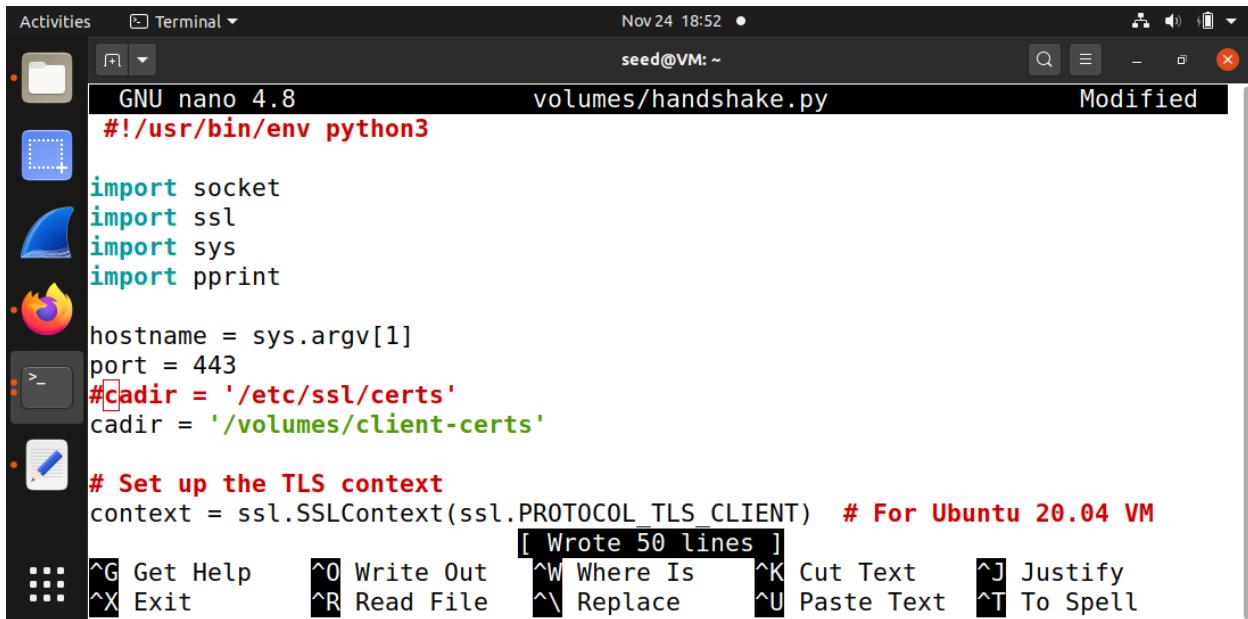
- Lines 9 to 11 are the TCP 3-way handshake.
- Line 9 a SYN packet is set from the client to the server
- Line 10 the server sends a SYN and ACK(Acknowledgment) to the client based on the previous step
- Once we finish TCP handshake it is TLS handshake .This is the process of negotiating encryption parameters 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:



```
GNU nano 4.8          volumes/handshake.py      Modified
#!/usr/bin/env python3

import socket
import ssl
import sys
import pprint

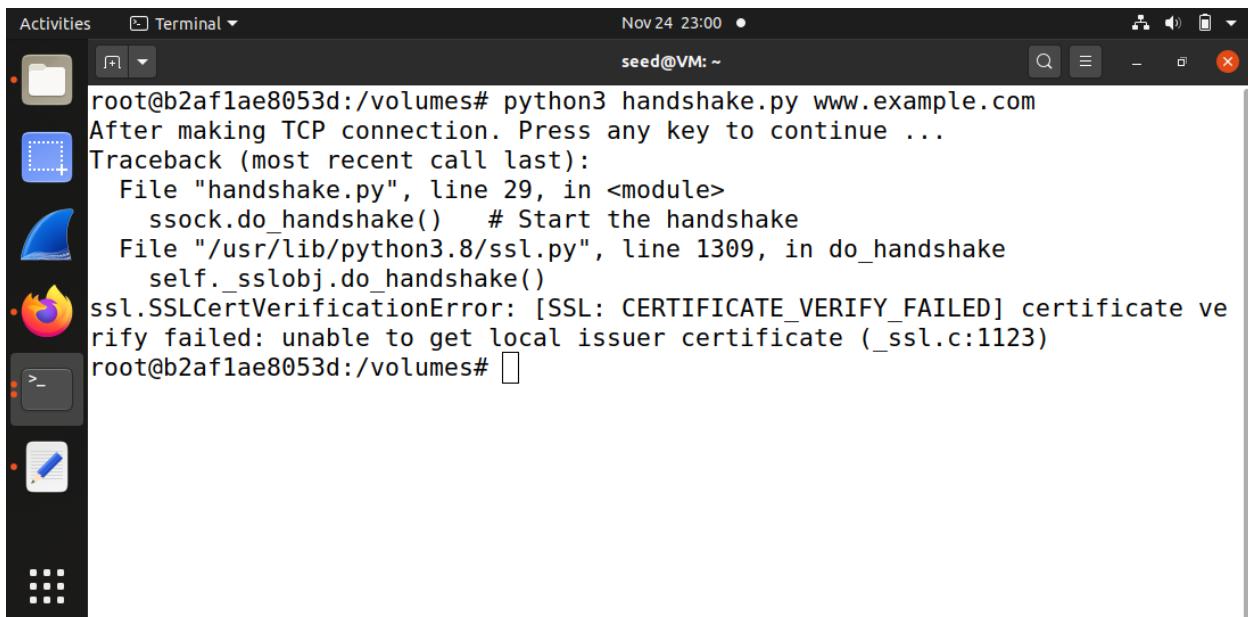
hostname = sys.argv[1]
port = 443
#cadir = '/etc/ssl/certs'
cadir = '/volumes/client-certs'

# Set up the TLS context
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT) # For Ubuntu 20.04 VM
```

[Wrote 50 lines]

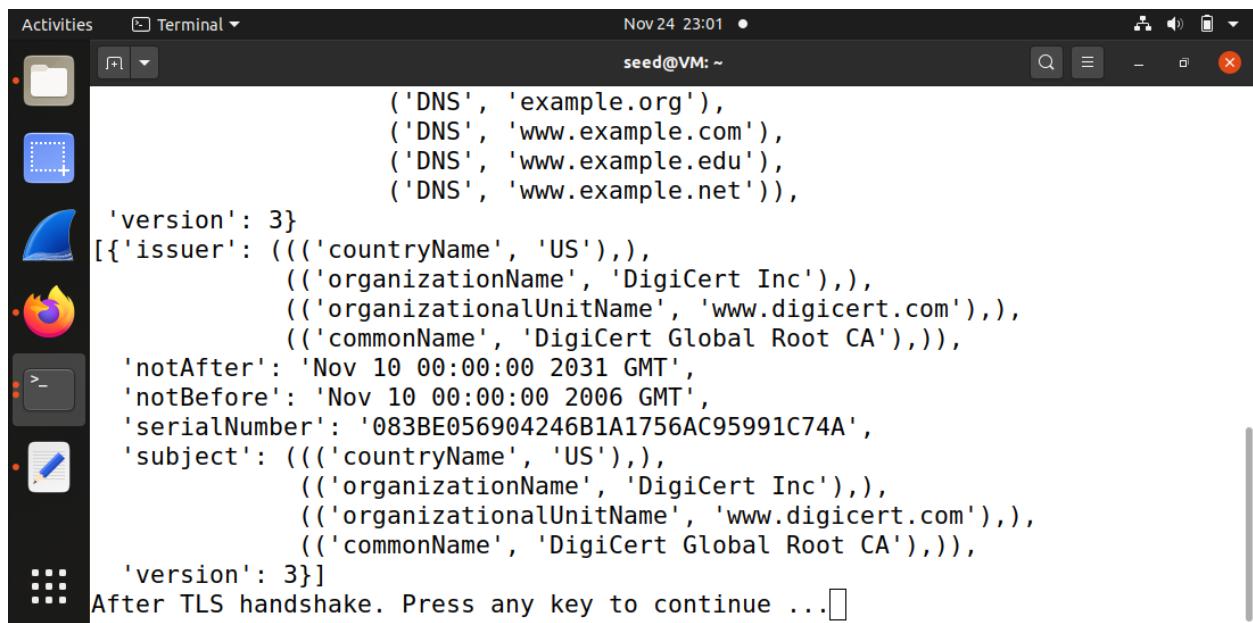
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell

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



```
root@b2af1ae8053d:/volumes# python3 handshake.py www.example.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "handshake.py", line 29, in <module>
    ssock.do_handshake() # Start the handshake
  File "/usr/lib/python3.8/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ssl.c:1123)
root@b2af1ae8053d:/volumes#
```

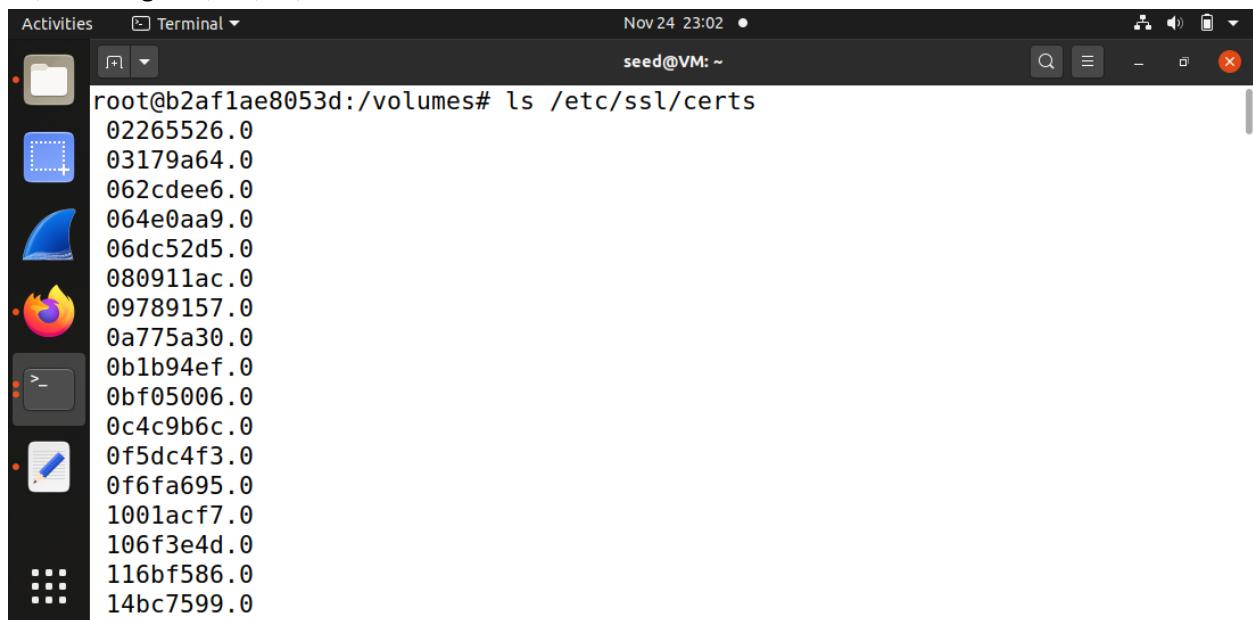
To fix this we have to place the corresponding CA's certificate into your client-certs folder. ie.
I used my client program to find out what CA certificate is needed to verify the www.example.com
server's certificate, and then copied the certificate from the /etc/ssl/certs to my folder.



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark theme with white text. It displays a JSON-like configuration object with various fields such as 'version', 'issuer', 'notAfter', 'notBefore', 'serialNumber', 'subject', and 'version'. At the bottom of the terminal, there is a message: "After TLS handshake. Press any key to continue ...".

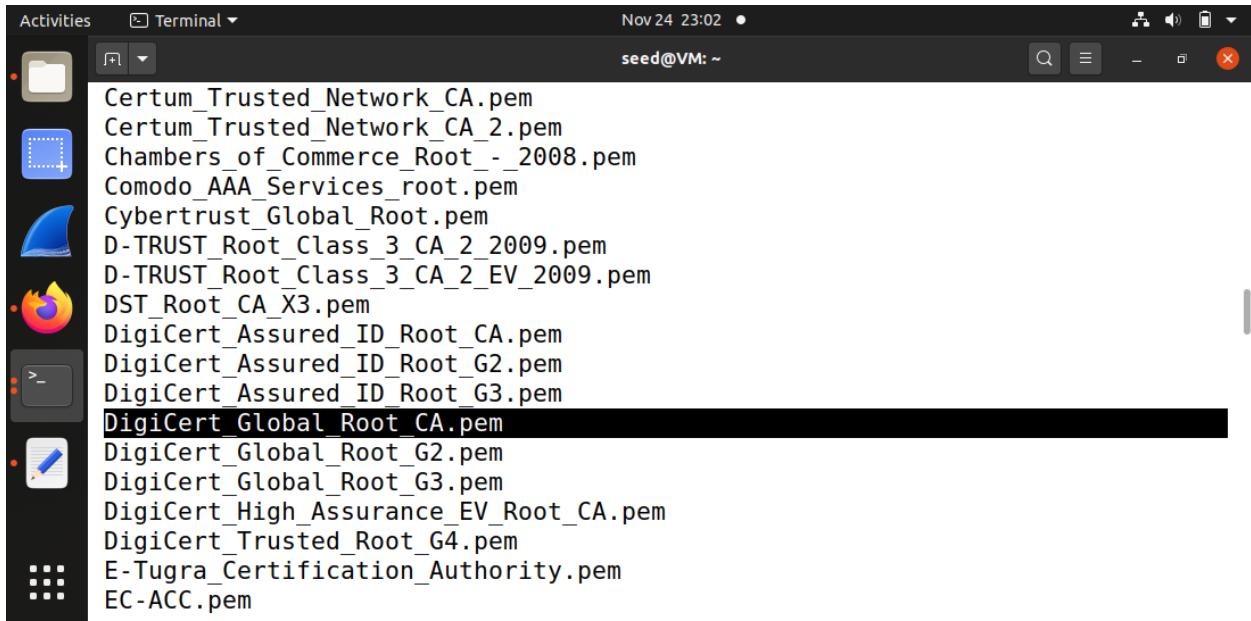
```
( 'DNS', 'example.org'),
( 'DNS', 'www.example.com'),
( 'DNS', 'www.example.edu'),
( 'DNS', 'www.example.net')),
'version': 3}
[{'issuer': (((('countryName', 'US'))),
    (('organizationName', 'DigiCert Inc'))),
    (('organizationalUnitName', 'www.digicert.com'))),
    (('commonName', 'DigiCert 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 ...
```

So, checking the /etc/ssl/certs folder



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark theme with white text. It displays the output of the command 'ls /etc/ssl/certs', which lists numerous files with names starting with '0' followed by a long string of characters and ending with '.0'.

```
root@b2af1ae8053d:/volumes# ls /etc/ssl/certs
02265526.0
03179a64.0
062cdee6.0
064e0aa9.0
06dc52d5.0
080911ac.0
09789157.0
0a775a30.0
0b1b94ef.0
0bf05006.0
0c4c9b6c.0
0f5dc4f3.0
0f6fa695.0
1001acf7.0
106f3e4d.0
116bf586.0
14bc7599.0
```



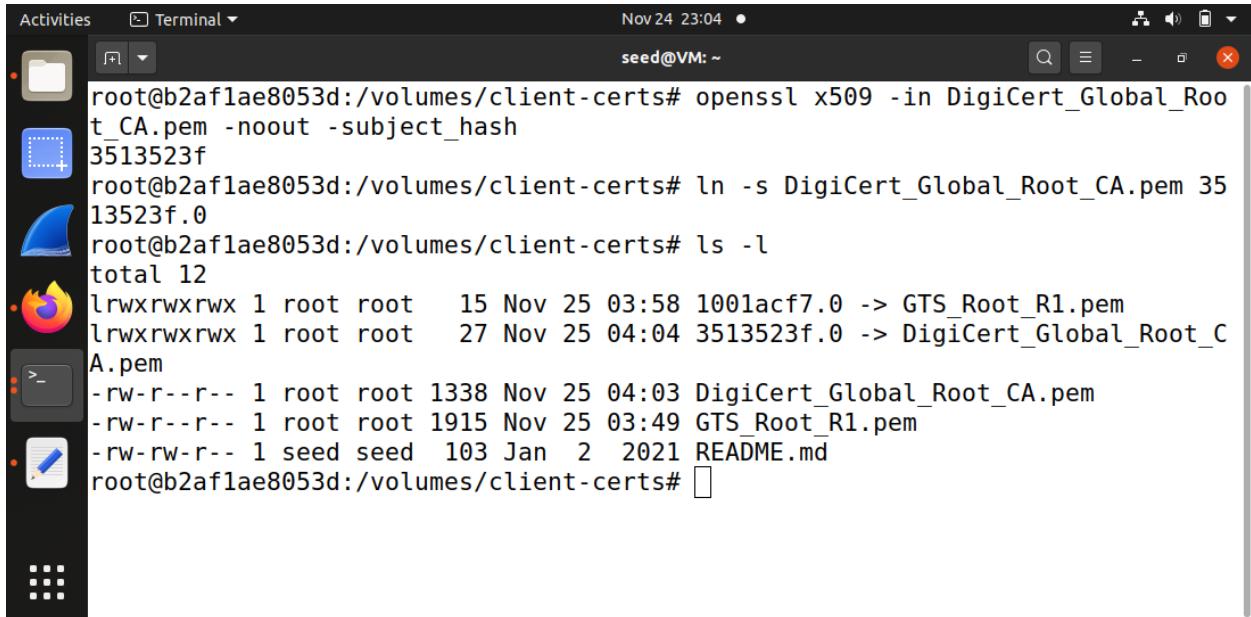
We find the CA certificate needed and we copy it to the new file.

```
emSign_Root_CA_-_G1.pem
f081611a.0
f0c70a8d.0
f249de83.0
f30dd6ad.0
f3377b1b.0
f387163d.0
f39fc864.0
f51bb24c.0
fc5a8f99.0
fe8a2cd8.0
ff34af3f.0
thawte_Primary_Root_CA.pem
thawte_Primary_Root_CA_-_G2.pem
thawte_Primary_Root_CA_-_G3.pem
root@b2af1ae8053d:/volumes# cp /etc/ssl/certs/DigiCert_Global_Root_CA.pem /volumes/client-certs/
root@b2af1ae8053d:/volumes# 
```

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

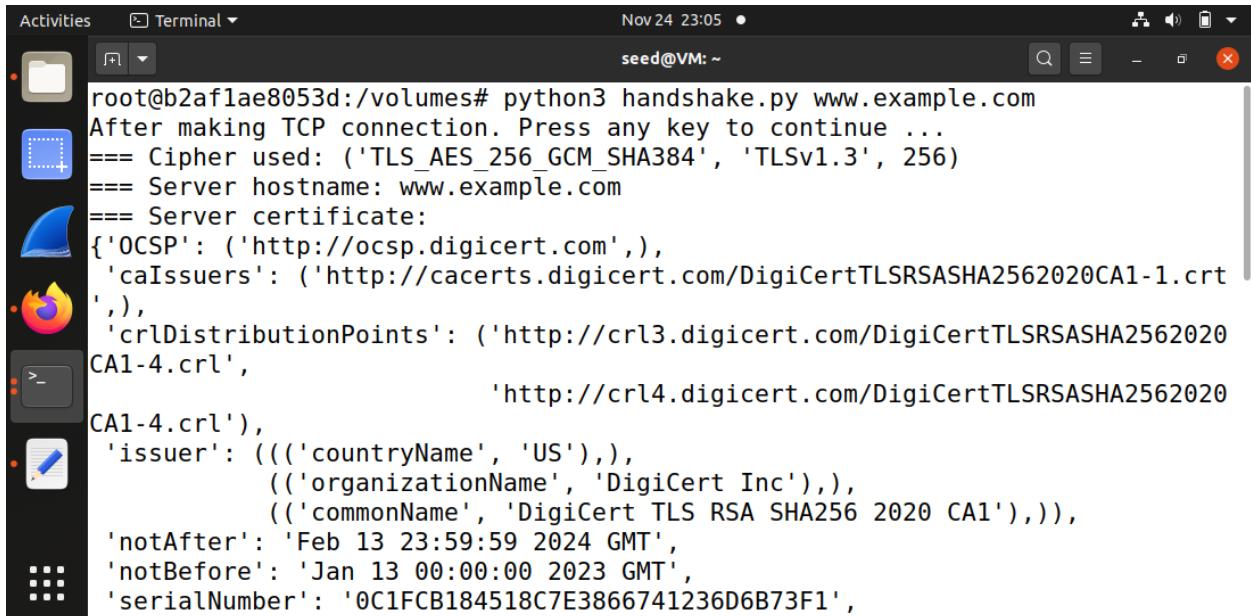
When TLS tries to verify a server certificate, it will generate a hash value from the issuer's identify information, we use this hash value as part of the file name, and then use this name to find the issuer's certificate in the "./client-certs" folder. Therefore, we need to rename each CA's certificate 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.

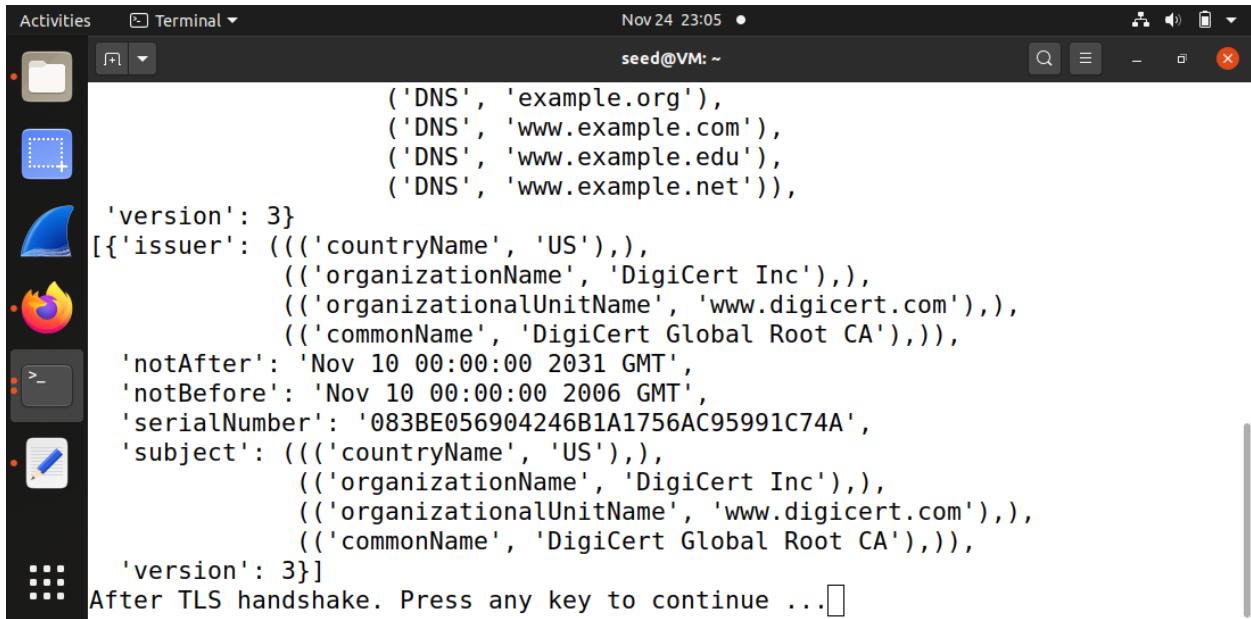


```
Activities Terminal Nov 24 23:04 • seed@VM: ~
root@b2af1ae8053d:/volumes/client-certs# openssl x509 -in DigiCert_Global_Root_CA.pem -noout -subject_hash
3513523f
root@b2af1ae8053d:/volumes/client-certs# ln -s DigiCert_Global_Root_CA.pem 3513523f.0
root@b2af1ae8053d:/volumes/client-certs# ls -l
total 12
lrwxrwxrwx 1 root root 15 Nov 25 03:58 1001acf7.0 -> GTS_Root_R1.pem
lrwxrwxrwx 1 root root 27 Nov 25 04:04 3513523f.0 -> DigiCert_Global_Root_CA.pem
-rw-r--r-- 1 root root 1338 Nov 25 04:03 DigiCert_Global_Root_CA.pem
-rw-r--r-- 1 root root 1915 Nov 25 03:49 GTS_Root_R1.pem
-rw-rw-r-- 1 seed seed 103 Jan 2 2021 README.md
root@b2af1ae8053d:/volumes/client-certs#
```

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



```
Activities Terminal Nov 24 23:05 • 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/DigiCertTLSRSASHA2562020CA1-4.crl',
   ...),
 'issuers': (((('countryName', 'US'),),
   ...),
   ...),
   ...),
 'notAfter': 'Feb 13 23:59:59 2024 GMT',
 'notBefore': 'Jan 13 00:00:00 2023 GMT',
 'serialNumber': '0C1FCB184518C7E3866741236D6B73F1',
```



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark theme with white text. The title bar says "Activities Terminal" and "Nov 24 23:05". The user is at the prompt "seed@VM: ~". The terminal content shows a JSON-like structure of a certificate chain:

```
( 'DNS', 'example.org'),
( 'DNS', 'www.example.com'),
( '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 ...
```

Conducted it for 2 other sites:

www.google.com

Error:

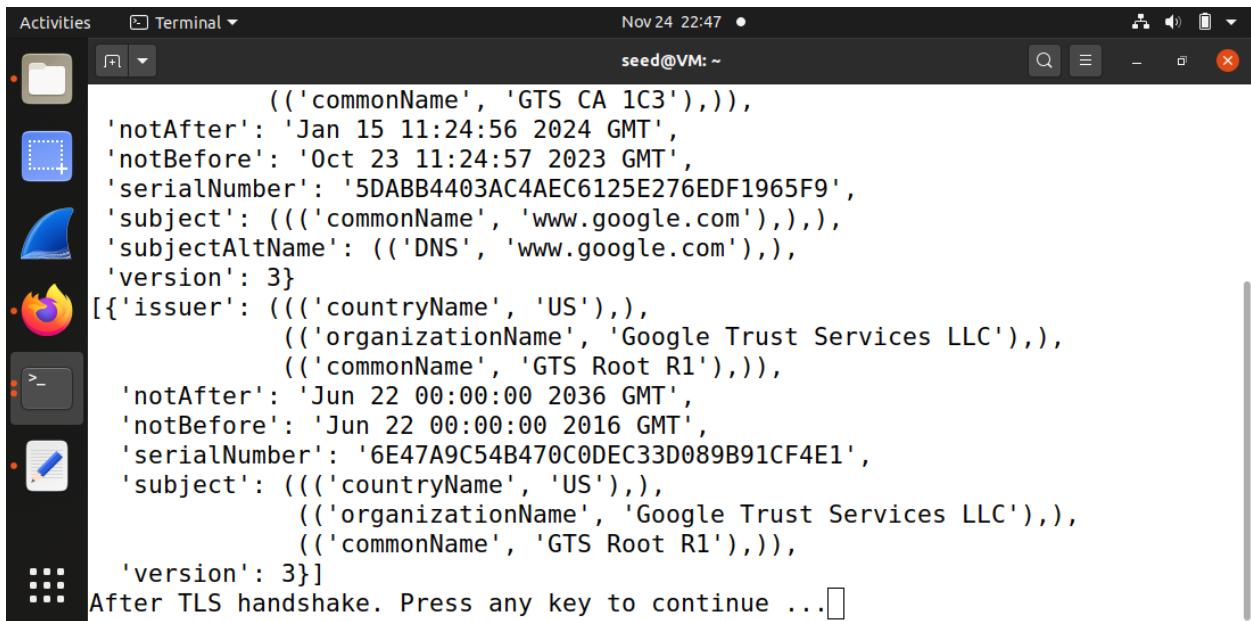
We get before for not having any certificates in new folder.



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window as root. The terminal window has a dark theme with white text. The title bar says "Activities Terminal" and "Nov 24 22:46". The user is at the prompt "seed@VM: ~". The terminal content shows the command being run and the resulting error:

```
root@b2af1ae8053d:/volumes# python3 handshake.py www.google.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "handshake.py", line 29, in <module>
    ssock.do_handshake()  # Start the handshake
  File "/usr/lib/python3.8/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ssl.c:1123)
root@b2af1ae8053d:/volumes#
```

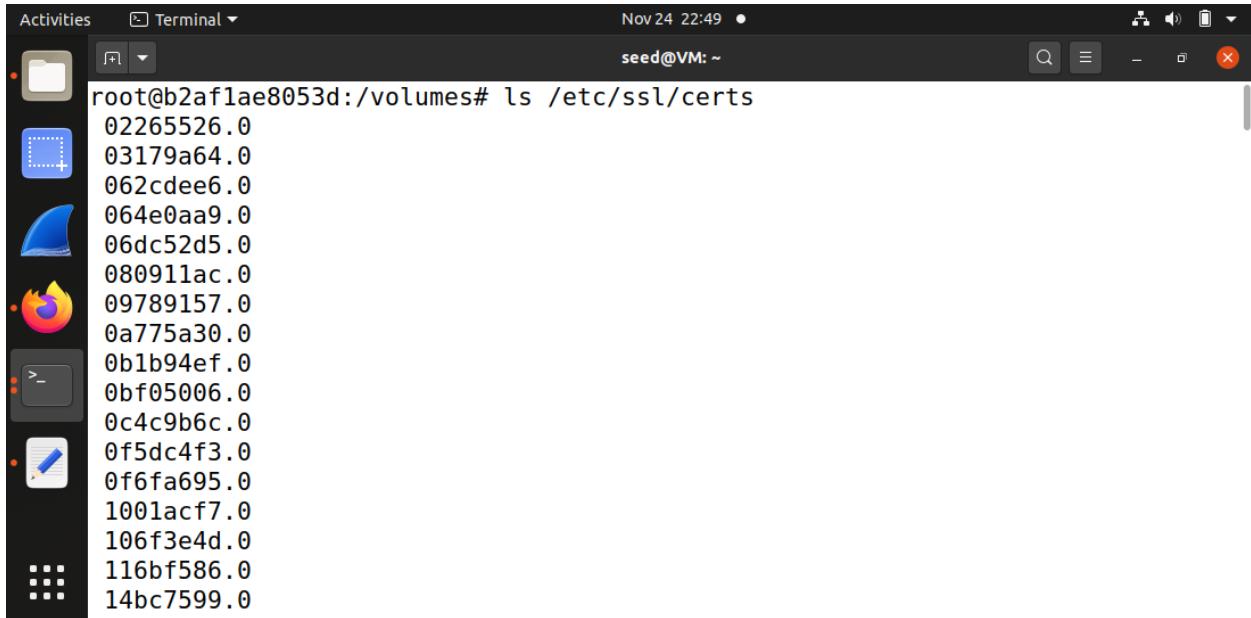
Using client program to figure out the CA certificate needed to be copied to the new folder.



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark theme with white text. It displays a JSON object representing a certificate. The output starts with a large bracketed block of data, followed by a line starting with 'After TLS handshake. Press any key to continue ...'. The terminal window is titled 'Terminal' and shows the date and time 'Nov 24 22:47' and the user 'seed@VM: ~'.

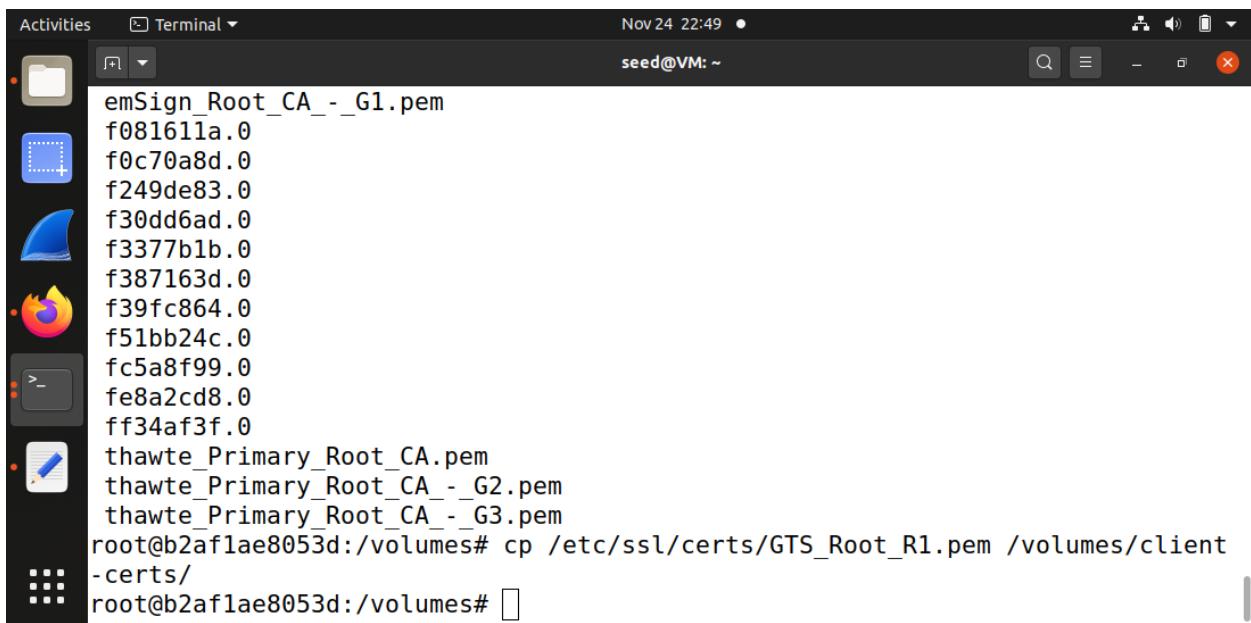
```
((('commonName', 'GTS CA 1C3'))),
'notAfter': 'Jan 15 11:24:56 2024 GMT',
'notBefore': 'Oct 23 11:24:57 2023 GMT',
'serialNumber': '5DABB4403AC4AEC6125E276EDF1965F9',
'subject': (((('commonName', 'www.google.com')))),
'subjectAltName': ((('DNS', 'www.google.com'))),
'version': 3}
[{'issuer': (((('countryName', 'US'))),
    ('organizationName', 'Google Trust Services LLC')),
    ('commonName', 'GTS Root R1'))),
'notAfter': 'Jun 22 00:00:00 2036 GMT',
'notBefore': 'Jun 22 00:00:00 2016 GMT',
'serialNumber': '6E47A9C54B470C0DEC33D089B91CF4E1',
'subject': (((('countryName', 'US'))),
    ('organizationName', 'Google Trust Services LLC')),
    ('commonName', 'GTS Root R1'))),
'version': 3}]
After TLS handshake. Press any key to continue ...
```

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



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark theme with white text. It displays the command 'ls /etc/ssl/certs' followed by a list of numerous files, each ending in '.0'. The terminal window is titled 'Terminal' and shows the date and time 'Nov 24 22:49' and the user 'seed@VM: ~'.

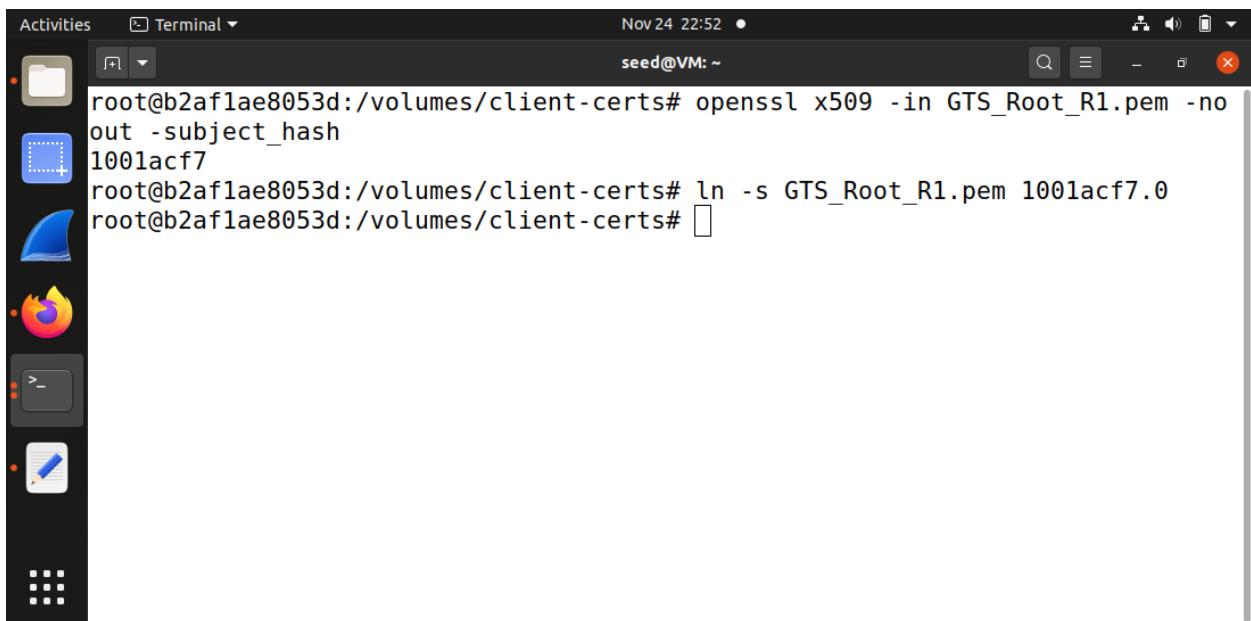
```
root@b2af1ae8053d:/volumes# ls /etc/ssl/certs
02265526.0
03179a64.0
062cdee6.0
064e0aa9.0
06dc52d5.0
080911ac.0
09789157.0
0a775a30.0
0b1b94ef.0
0bf05006.0
0c4c9b6c.0
0f5dc4f3.0
0f6fa695.0
1001acf7.0
106f3e4d.0
116bf586.0
14bc7599.0
```



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark theme and displays the following command-line session:

```
Nov 24 22:49 • seed@VM: ~
cp /etc/ssl/certs/GTS_Root_R1.pem /volumes/client-certs/
root@b2af1ae8053d:/volumes#
```

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



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark theme and displays the following command-line session:

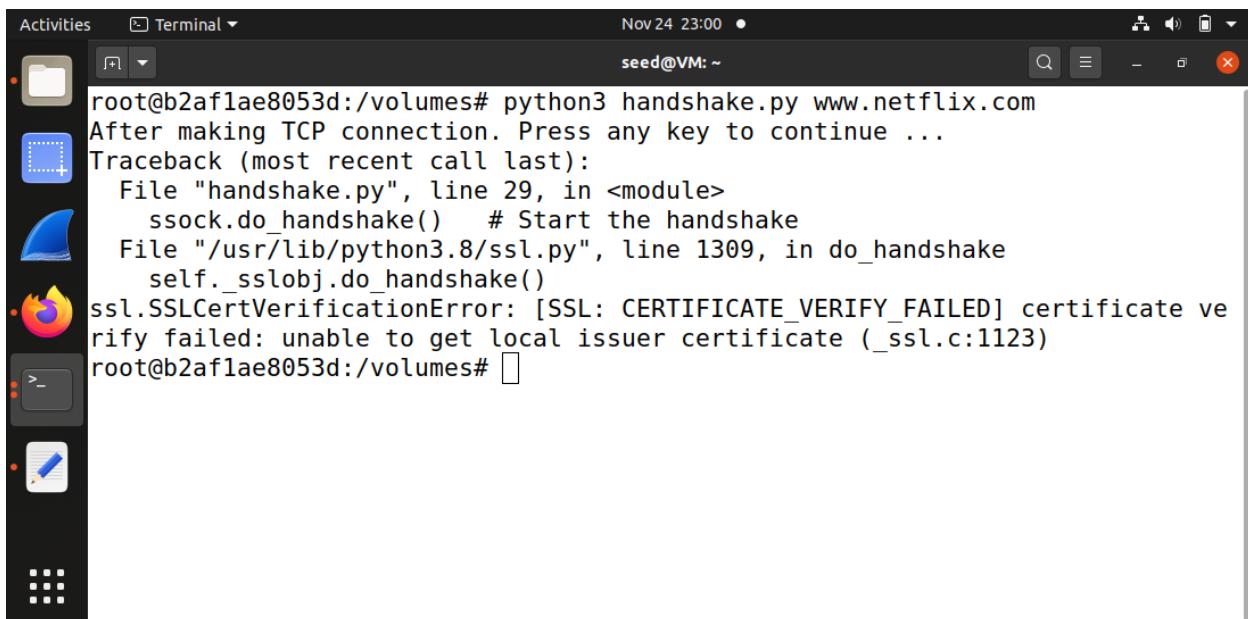
```
Nov 24 22:52 • seed@VM: ~
openssl x509 -in GTS_Root_R1.pem -noout -subject_hash
1001acf7
ln -s GTS_Root_R1.pem 1001acf7.0
root@b2af1ae8053d:/volumes/client-certs#
```

Now we run the “handshake.py” program.

```
Activities Terminal Nov 24 22:59 • seed@VM: ~
root@b2af1ae8053d:/volumes# python3 handshake.py www.google.com
After making TCP connection. Press any key to continue ...
==> Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
==> Server hostname: www.google.com
==> Server certificate:
{'OCSP': ('http://ocsp.pki.goog/gts1c3',),
 'caIssuers': ('http://pki.goog/repo/certs/gts1c3.der',),
 'crlDistributionPoints': ('http://crls.pki.goog/gts1c3/zdATt0Ex_Fk.crl',),
 'issuer': (((('countryName', 'US'),),
              (('organizationName', 'Google Trust Services LLC'),),
              (('commonName', 'GTS CA 1C3'))),
             {'notAfter': 'Jan 15 11:24:56 2024 GMT',
              'notBefore': 'Oct 23 11:24:57 2023 GMT',
              'serialNumber': '5DABB4403AC4AEC6125E276EDF1965F9',
              'subject': (((('commonName', 'www.google.com'),),
                           {'version': 3}
                           [{}{'issuer': (((('countryName', 'US'),),
                               (('organizationName', 'Google Trust Services LLC'),),
                               (('commonName', 'GTS Root R1'))),
                             {'notAfter': 'Jun 22 00:00:00 2036 GMT',
                              'notBefore': 'Jun 22 00:00:00 2016 GMT',
                              'serialNumber': '6E47A9C54B470C0DEC33D089B91CF4E1',
                              'subject': (((('countryName', 'US'),),
                                           (('organizationName', 'Google Trust Services LLC'),),
                                           (('commonName', 'GTS Root R1'))),
                                         {'version': 3}])
                           After TLS handshake. Press any key to continue ...
root@b2af1ae8053d:/volumes#
```

For

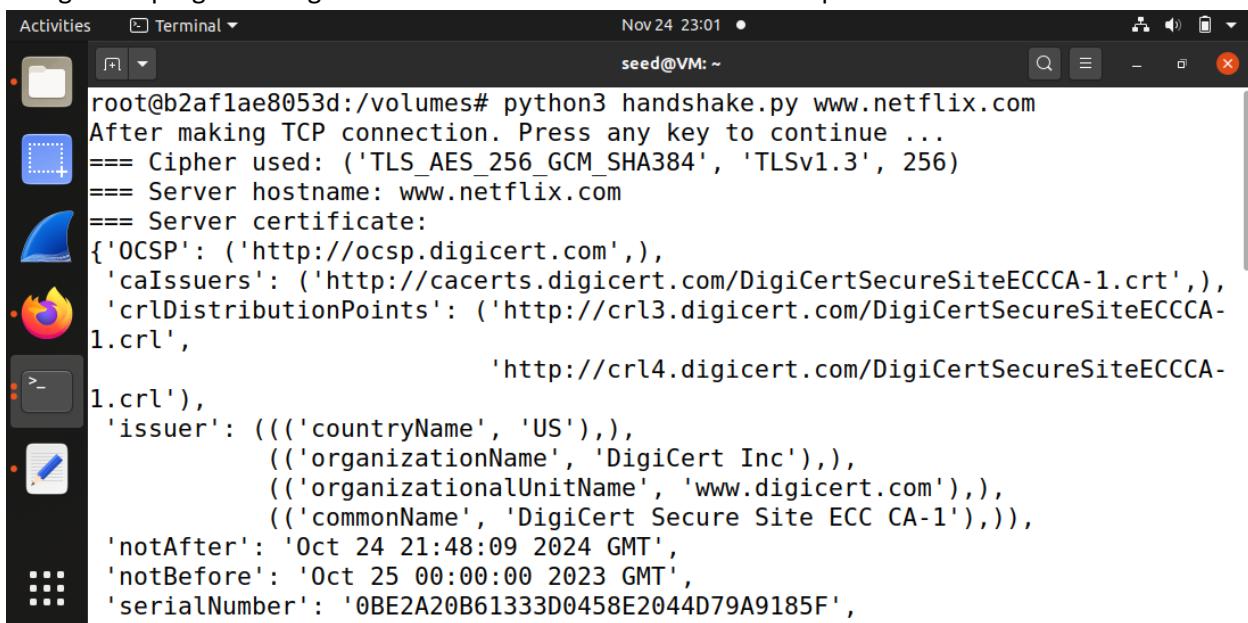
www.netflix.com



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark theme with white text. It displays the command `python3 handshake.py www.netflix.com` being run as root. The output shows a traceback for an SSL certificate verification error, indicating that the certificate verify failed due to an unable to get local issuer certificate issue.

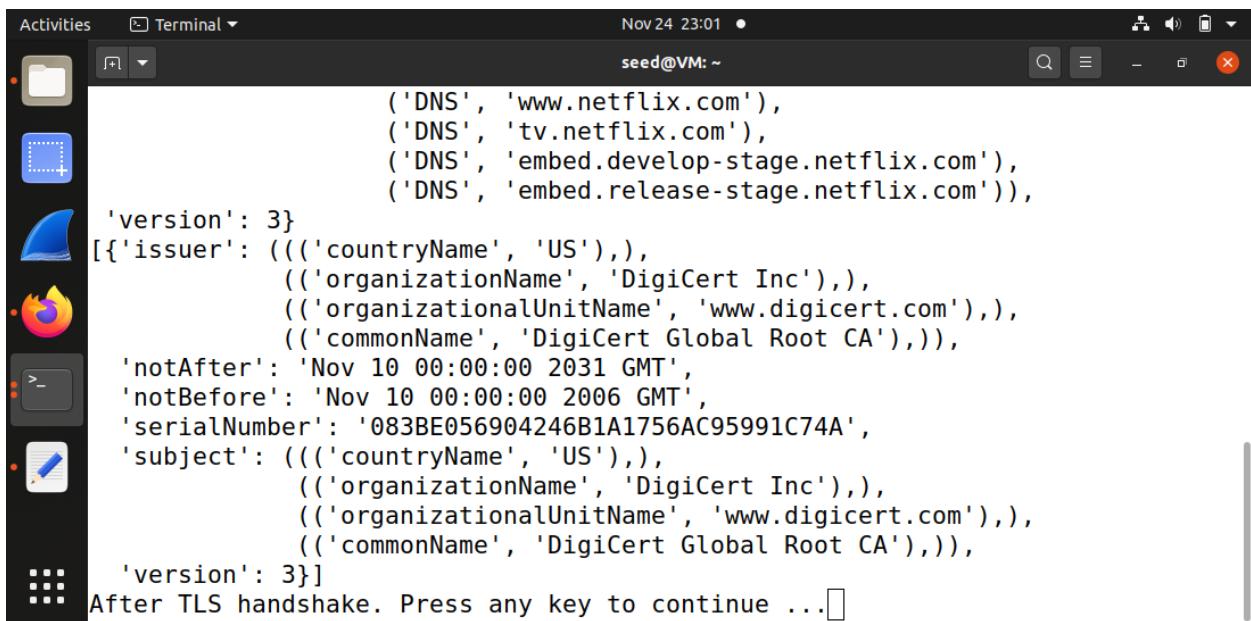
```
root@b2af1ae8053d:/volumes# python3 handshake.py www.netflix.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "handshake.py", line 29, in <module>
    ssock.do_handshake()  # Start the handshake
  File "/usr/lib/python3.8/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ssl.c:1123)
root@b2af1ae8053d:/volumes#
```

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



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark theme with white text. It displays the command `python3 handshake.py www.netflix.com` being run as root. The output provides detailed information about the server's certificate, including the cipher used (TLS_AES_256_GCM_SHA384), the server's hostname (www.netflix.com), and its certificate details. The certificate is issued by DigiCert Inc, with various fields like countryName, organizationName, organizationalUnitName, commonName, notBefore, notAfter, and serialNumber.

```
root@b2af1ae8053d:/volumes# python3 handshake.py www.netflix.com
After making TCP connection. Press any key to continue ...
==> Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
==> Server hostname: www.netflix.com
==> Server certificate:
{'OCSP': ('http://ocsp.digicert.com',),
 'caIssuers': ('http://cacerts.digicert.com/DigiCertSecureSiteECCCA-1.crt',),
 'crlDistributionPoints': ('http://crl3.digicert.com/DigiCertSecureSiteECCCA-1.crl',
                           'http://crl4.digicert.com/DigiCertSecureSiteECCCA-1.crl'),
 'issuer': (((('countryName', 'US'),),
              (('organizationName', 'DigiCert Inc'),),
              (('organizationalUnitName', 'www.digicert.com'),),
              (('commonName', 'DigiCert Secure Site ECC CA-1'))),
             'notAfter': 'Oct 24 21:48:09 2024 GMT',
             'notBefore': 'Oct 25 00:00:00 2023 GMT',
             'serialNumber': '0BE2A20B61333D0458E2044D79A9185F',
```



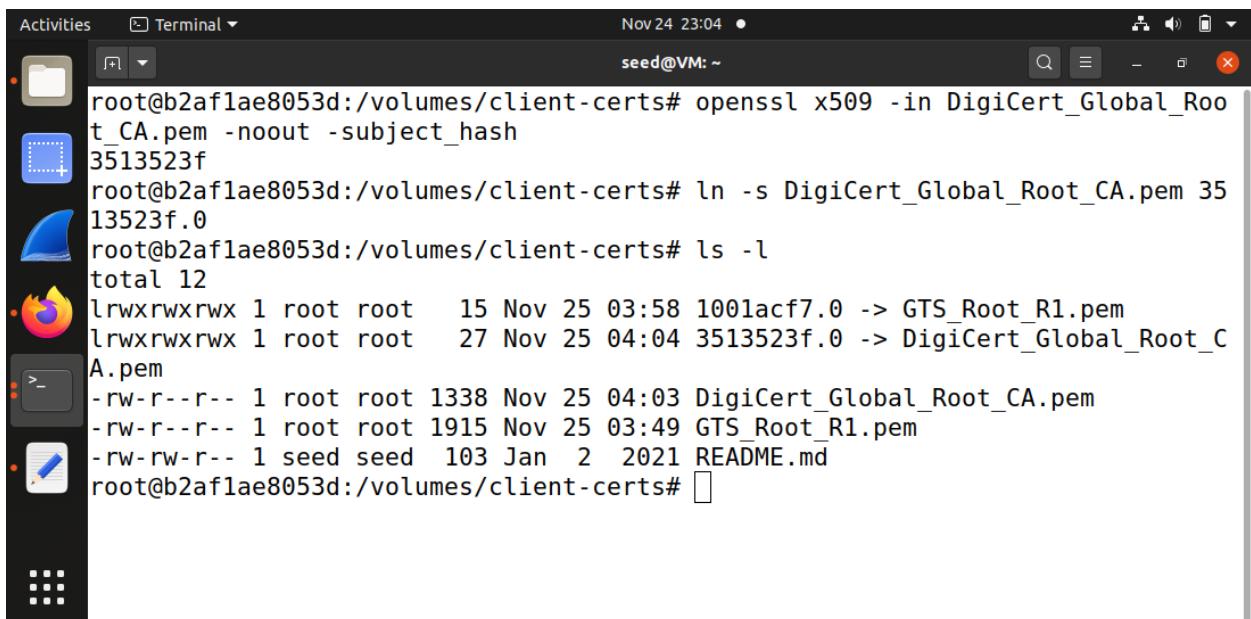
A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark theme with a light-colored background for the code output. The title bar shows "Activities Terminal" and the date "Nov 24 23:01". The user is at the prompt "seed@VM: ~". The code displayed is a JSON object representing a certificate, including fields like DNS names, issuer information, and timestamps. At the bottom of the terminal, it says "After TLS handshake. Press any key to continue ...".

```
( 'DNS', 'www.netflix.com'),
( 'DNS', 'tv.netflix.com'),
( 'DNS', 'embed.develop-stage.netflix.com'),
( 'DNS', 'embed.release-stage.netflix.com'),
'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 ...
```

Copying to new folder

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

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



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark theme with a light-colored background for the code output. The title bar shows "Activities Terminal" and the date "Nov 24 23:04". The user is at the prompt "seed@VM: ~". The command run is "openssl x509 -in DigiCert_Global_Root_CA.pem -noout -subject_hash", which outputs the hash value "3513523f". Then, "ln -s DigiCert_Global_Root_CA.pem 3513523f.0" is run to create a symbolic link. Finally, "ls -l" is run to list the contents of the directory, showing files like GTS_Root_R1.pem, DigiCert_Global_Root_C.A.pem, DigiCert_Global_Root_CA.pem, GTS_Root_R1.pem, and README.md. The user is back at the root prompt "root@b2af1ae8053d:/volumes/client-certs#".

```
root@b2af1ae8053d:/volumes/client-certs# openssl x509 -in DigiCert_Global_Root_CA.pem -noout -subject_hash
3513523f
root@b2af1ae8053d:/volumes/client-certs# ln -s DigiCert_Global_Root_CA.pem 3513523f.0
root@b2af1ae8053d:/volumes/client-certs# ls -l
total 12
lrwxrwxrwx 1 root root 15 Nov 25 03:58 1001acf7.0 -> GTS_Root_R1.pem
lrwxrwxrwx 1 root root 27 Nov 25 04:04 3513523f.0 -> DigiCert_Global_Root_C.A.pem
-rw-r--r-- 1 root root 1338 Nov 25 04:03 DigiCert_Global_Root_CA.pem
-rw-r--r-- 1 root root 1915 Nov 25 03:49 GTS_Root_R1.pem
-rw-rw-r-- 1 seed seed 103 Jan 2 2021 README.md
root@b2af1ae8053d:/volumes/client-certs#
```

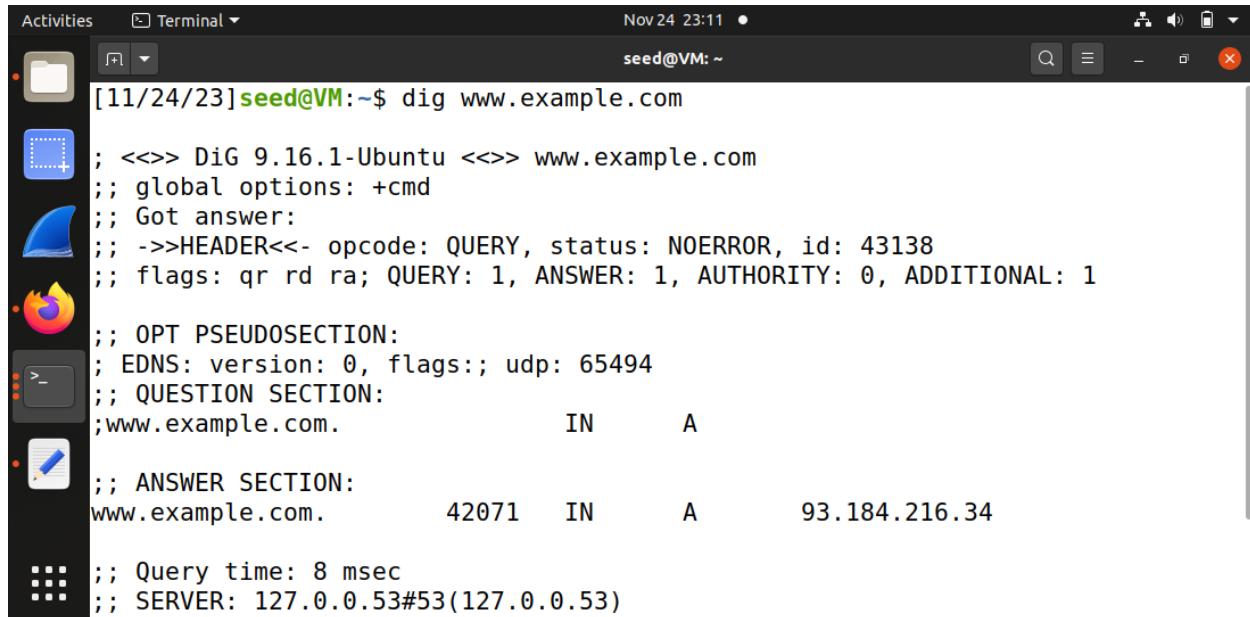
Now we run the “handshake.py” program.

```
Activities Terminal Nov 24 23:06 • seed@VM: ~
root@b2af1ae8053d:/volumes# python3 handshake.py www.netflix.com
After making TCP connection. Press any key to continue ...
==> Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
==> Server hostname: www.netflix.com
==> Server certificate:
{'OCSP': ('http://ocsp.digicert.com',),
 'caIssuers': ('http://cacerts.digicert.com/DigiCertSecureSiteECCCA-1.crt',),
 'crlDistributionPoints': ('http://crl3.digicert.com/DigiCertSecureSiteECCCA-1.crl',
                           'http://crl4.digicert.com/DigiCertSecureSiteECCCA-1.crl'),
 'issuer': (((('countryName', 'US'),),
              (('organizationName', 'DigiCert Inc'),),
              (('organizationalUnitName', 'www.digicert.com'),),
              (('commonName', 'DigiCert Secure Site ECC CA-1'))),
             'notAfter': 'Oct 24 21:48:09 2024 GMT',
             'notBefore': 'Oct 25 00:00:00 2023 GMT',
             'serialNumber': '0BE2A20B61333D0458E2044D79A9185F',
             '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 ...
```

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):



```
[11/24/23]seed@VM:~$ dig www.example.com
; <>> DiG 9.16.1-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43138
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.example.com.           IN      A
;; ANSWER SECTION:
www.example.com.        42071    IN      A      93.184.216.34
;; Query time: 8 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
```

- 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

```
root@b2af1ae8053d:/volumes# nano /etc/hosts
root@b2af1ae8053d:/volumes# 

Activities Terminal Nov 24 23:13 seed@VM: ~
● ○ □ ×

GNU nano 4.8 /etc/hosts Modified
127.0.0.1 localhost
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.9.0.5 b2af1ae8053d
93.184.216.34 www.example2023.com

Activities Terminal Nov 24 23:12 seed@VM: ~
● ○ □ ×

^G Get Help    ^O Write Out    ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File    ^\ Replace     ^U Paste Text   ^T To Spell
```

- 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:

```
GNU nano 4.8                               handshake.py
hostname = sys.argv[1]
port = 443
#cadir = '/etc/ssl/certs'
cadir = '/volumes/client-certs'

# Set up the TLS context
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT) # For Ubuntu 20.04 VM
# context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)      # For Ubuntu 16.04 VM

context.load_verify_locations(capath=cadir)
context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = True

# Create TCP connection
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell

```
root@b2af1ae8053d:/volumes# python3 handshake.py www.example2023.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "handshake.py", line 29, in <module>
    ssock.do_handshake() # Start the handshake
  File "/usr/lib/python3.8/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: Hostname mismatch, certificate is not valid for 'www.example2023.com'. (_ssl.c:1123)
root@b2af1ae8053d:/volumes#
```

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

```

Activities Terminal Nov 24 23:15 • seed@VM: ~
GNU nano 4.8 handshake.py Modified
hostname = sys.argv[1]
port = 443
#cadir = '/etc/ssl/certs'
cadir = '/volumes/client-certs'

# Set up the TLS context
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT) # For Ubuntu 20.04 VM
# context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2) # For Ubuntu 16.04 VM

context.load_verify_locations(capath=cadir)
context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = False

# Create TCP connection

```

Terminal 1 (Nano Editor):

```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell

```

Terminal 2 (Root Shell):

```

root@b2af1ae8053d:/volumes# python3 handshake.py www.example2023.com
After making TCP connection. Press any key to continue ...
==> Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
==> Server hostname: www.example2023.com
==> Server certificate:
{'OCSP': ('http://ocsp.digicert.com',),
 'caIssuers': ('http://cacerts.digicert.com/DigiCertTLSRSASHA2562020CA1-1.crt',
   ),
 'crlDistributionPoints': ('http://crl3.digicert.com/DigiCertTLSRSASHA2562020CA1-4.crl',
   'http://crl4.digicert.com/DigiCertTLSRSASHA2562020CA1-4.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',
}

```

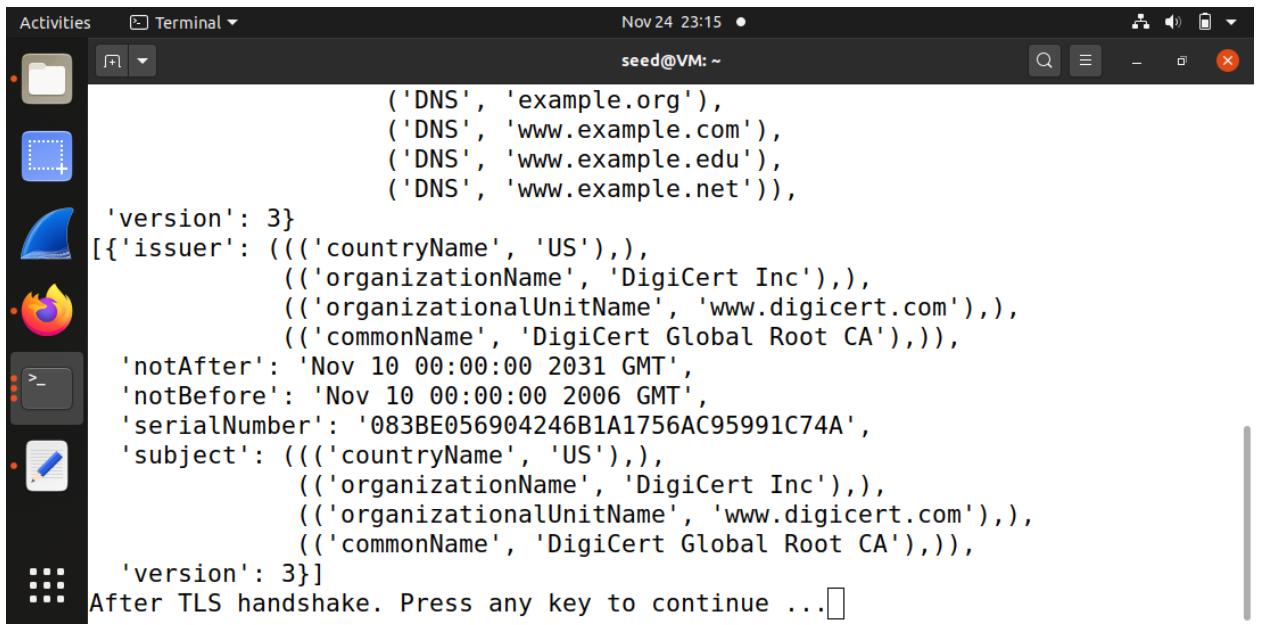
Observations from experiment:

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.

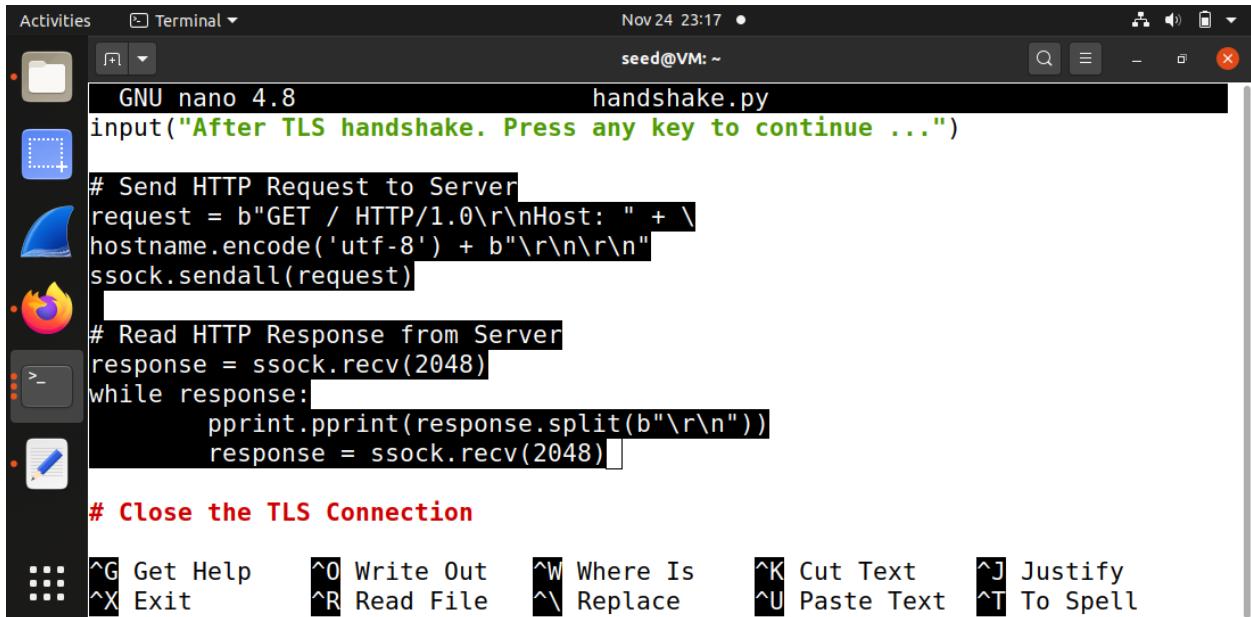
A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark theme and displays a certificate dump in JSON format. The certificate is issued by DigiCert Inc. for the common name 'DigiCert Global Root CA'. It includes fields for DNS names like 'example.org' and 'www.example.com'. The terminal window is titled 'Terminal' and shows the date and time as 'Nov 24 23:15'. The background shows the Unity desktop interface with various icons in the dock.

```
( 'DNS', 'example.org'),
('DNS', 'www.example.com'),
('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 ...
```

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.



```
GNU nano 4.8          handshake.py
input("After TLS handshake. Press any key to continue ...")

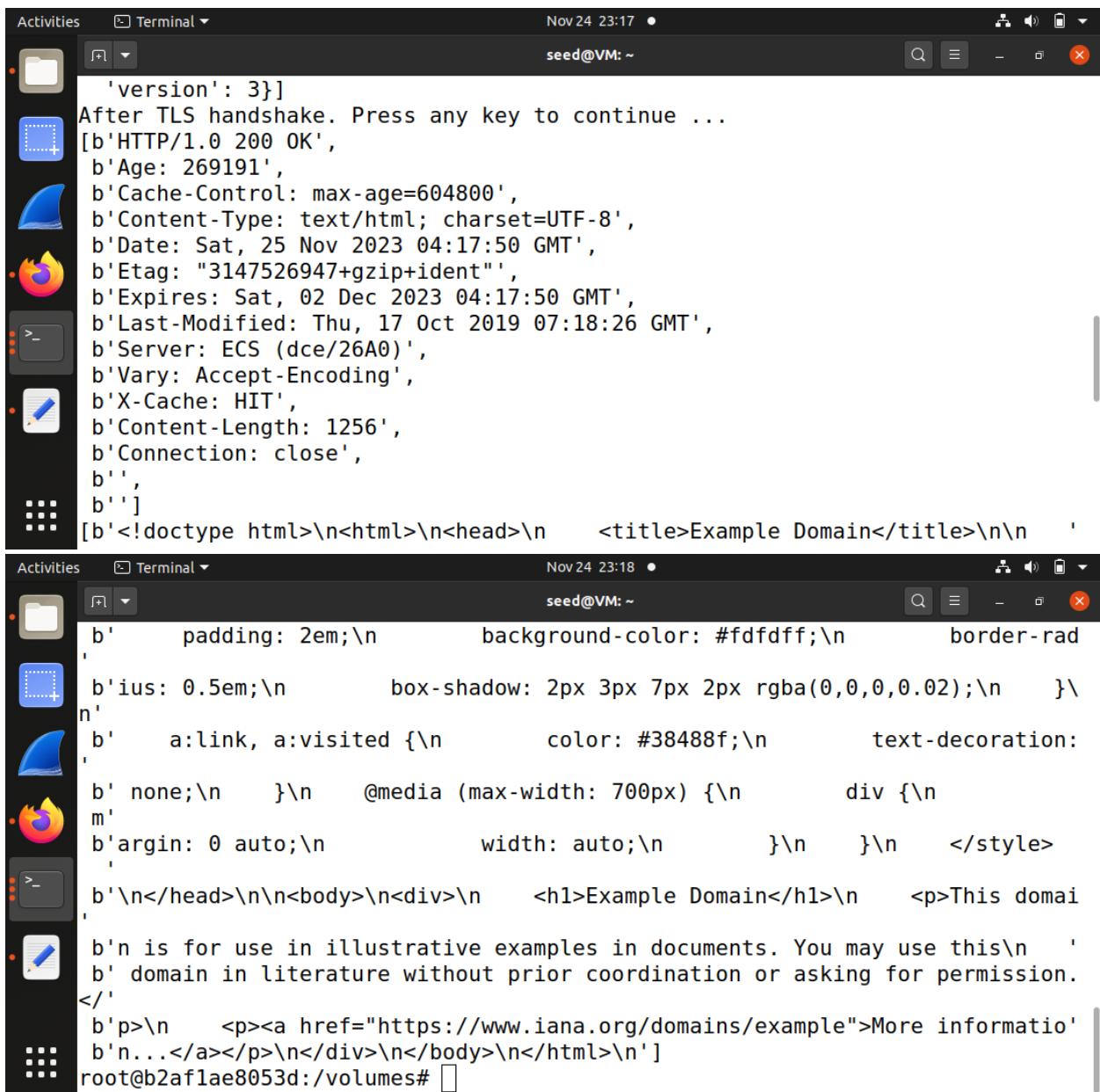
# Send HTTP Request to Server
request = b"GET / HTTP/1.0\r\nHost: " + \
hostname.encode('utf-8') + b"\r\n\r\n"
ssock.sendall(request)

# Read HTTP Response from Server
response = ssock.recv(2048)
while response:
    pprint.pprint(response.split(b"\r\n"))
    response = ssock.recv(2048)

# Close the TLS Connection
```

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/DigiCertTLSRSASHA2562020CA1-4.crl',
   'http://crl4.digicert.com/DigiCertTLSRSASHA2562020CA1-4.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',
 ('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',
```

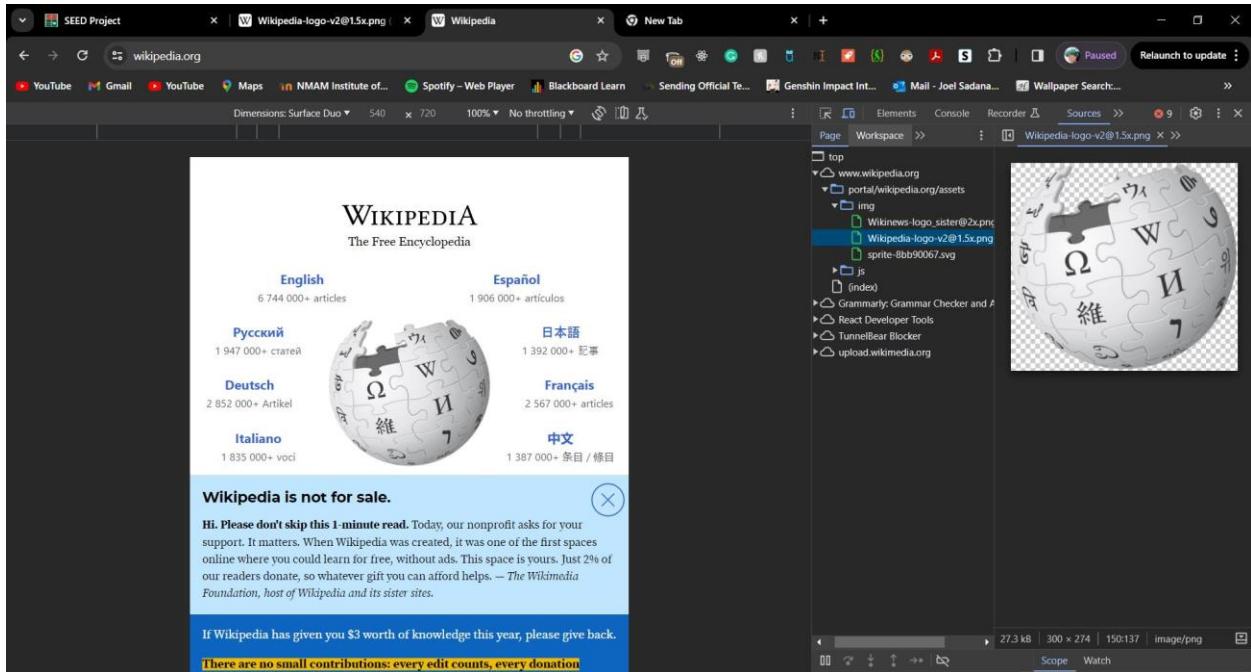


```

Activities Terminal Nov 24 23:17 • seed@VM: ~
'version': 3}]
After TLS handshake. Press any key to continue ...
[b'HTTP/1.0 200 OK',
 b'Age: 269191',
 b'Cache-Control: max-age=604800',
 b'Content-Type: text/html; charset=UTF-8',
 b'Date: Sat, 25 Nov 2023 04:17:50 GMT',
 b'Etag: "3147526947+gzip+ident"',
 b'Expires: Sat, 02 Dec 2023 04:17:50 GMT',
 b'Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT',
 b'Server: ECS (dce/26A0)',
 b'Vary: Accept-Encoding',
 b'X-Cache: HIT',
 b'Content-Length: 1256',
 b'Connection: close',
 b '',
 b'']
[b'<!doctype html>\n<html>\n<head>\n      <title>Example Domain</title>\n\n      <style>\n          padding: 2em;\n          background-color: #fdfdff;\n          border-radius: 0.5em;\n          box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);\n      }\n      a:link, a:visited {\n          color: #38488f;\n          text-decoration: none;\n      }\n      @media (max-width: 700px) {\n          div {\n              margin: 0 auto;\n              width: auto;\n          }\n      }\n  </style>\n</head>\n<body>\n<div>\n      <h1>Example Domain</h1>\n      <p>This domain\n          is for use in illustrative examples in documents. You may use this\n          domain in literature without prior coordination or asking for permission.\n      </p>\n      <p><a href="https://www.iana.org/domains/example">More information...</a></p>\n</div>\n</body>\n</html>\n']
root@b2af1ae8053d:/volumes# 
```

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”.



```
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'),),
```

```
Activities Terminal Nov 29 14:21 • seed@VM: ~/.../client-certs
('DNS', 'wikiversity.org'),
('DNS', 'wikivoyage.org'),
('DNS', 'wiktionary.org'),
('DNS', 'wmfusercontent.org')),
'version': 3}
[{'issuer': (((('countryName', 'US'),),
    ('organizationName', 'Internet Security Research Group'),),
    ('commonName', 'ISRG Root X1'))),
'notAfter': 'Jun 4 11:04:38 2035 GMT',
'notBefore': 'Jun 4 11:04:38 2015 GMT',
'serialNumber': '8210CFB0D240E3594463E0BB63828B00',
'subject': (((('countryName', 'US'),),
    ('organizationName', 'Internet Security Research Group'),),
    ('commonName', 'ISRG Root X1'))),
'version': 3}]
After TLS handshake. Press any key to continue ...
[b'HTTP/1.1 200 OK',
 b'date: Wed, 29 Nov 2023 16:12:22 GMT',
```

```
Activities Terminal Nov 29 14:23 • seed@VM: ~/.../client-certs
'version': 3}]
After TLS handshake. Press any key to continue ...
[b'HTTP/1.1 200 OK',
 b'date: Wed, 29 Nov 2023 16:12:22 GMT',
 b'cache-control: s-maxage=86400, must-revalidate, max-age=3600',
 b'server: ATS/9.1.4',
 b'etag: W/"155b8-60b24f29158f1"',
 b'last-modified: Mon, 27 Nov 2023 16:39:36 GMT',
 b'content-type: text/html',
 b'vary: Accept-Encoding',
 b'age: 11242',
 b'x-cache: cp1100 hit, cp1100 hit/166713',
 b'x-cache-status: hit-front',
 b'server-timing: cache;desc="hit-front", host;desc="cp1100"',
 b'strict-transport-security: max-age=106384710; includeSubDomains; preload',
 b'report-to: { "group": "wm_nel", "max_age": 604800, "endpoints": [{ "url": "h'
 b'ttps://intake-logging.wikimedia.org/v1/events?stream=w3c.reportingapi.netw
```

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.

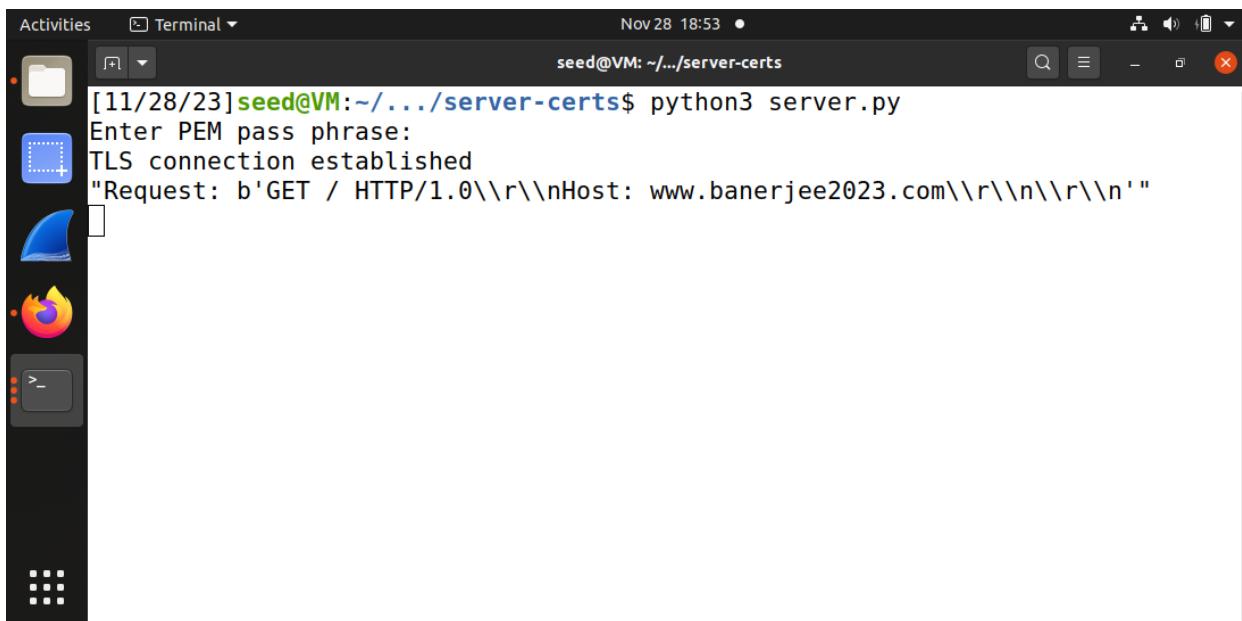
1. Introduction:

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.



```
[11/28/23] seed@VM:~/.../server-certs$ python3 server.py
Enter PEM pass phrase:
TLS connection established
"Request: b'GET / HTTP/1.0\r\nHost: www.banerjee2023.com\r\n\r\n'"
```

```

Activities Terminal Nov 28 18:52
seed@VM: ~/.../client-certs
[11/28/23]seed@VM:~/.../client-certs$ python3 handshake.py www.banerjee2023.com
After making TCP connection. Press any key to continue ...
==> Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
==> Server hostname: www.banerjee2023.com
==> Server certificate:
{'issuer': ((('countryName', 'US'),),
             (('stateOrProvinceName', 'Virginia'),),
             (('localityName', 'Fairfax'),),
             (('organizationName', 'GMU'),),
             (('commonName', 'www.banerjee2023.com'))),
 'notAfter': 'Nov 25 23:33:20 2033 GMT',
 'notBefore': 'Nov 28 23:33:20 2023 GMT',
 'serialNumber': '1009',
 'subject': ((('countryName', 'US'),),
             (('stateOrProvinceName', 'Virginia'),),
             (('localityName', 'Fairfax'),),
             (('organizationName', 'GMU'),),
             (('organizationName', 'GMU'),),
             (('commonName', 'www.banerjee2023.com'))),
 'version': 3}
After TLS handshake. Press any key to continue ...
[b'\nHTTP/1.1 200 OK',
 b'Content-Type: text/html',
 b '',
 b'\n<!DOCTYPE html><html><body><h1>This is Banerjee2023.com!!!</h1></body></html>\n']
[11/28/23]seed@VM:~/.../client-certs$ 

```

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.

The image shows two terminal windows side-by-side on a Linux desktop environment. Both windows have a dark theme with a light background for the terminal area.

Top Terminal:

```
[11/28/23] seed@VM:~/.../server-certs$ python3 server.py
Enter PEM pass phrase:
TLS connection fails
```

Bottom Terminal:

```
[11/28/23] seed@VM:~/.../client-certs$ python3 handshake.py www.banerjee2023.com
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "handshake.py", line 29, in <module>
    ssock.do_handshake() # Start the handshake
  File "/usr/lib/python3.8/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ssl.c:1123)
[11/28/23] seed@VM:~/.../client-certs$
```

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.

A screenshot of a terminal window titled "Terminal". The window shows a command-line session where the user is generating an RSA private key and a server certificate using OpenSSL. The session starts with creating a private key ("Generating a RSA private key") and then moves on to creating a certificate ("writing new private key to 'server.key'"). It then prompts for a PEM pass phrase and verifies it. Finally, it generates a server certificate ("openssl ca -md sha256 -days 3650 -config ./myopenssl.cnf -in server.csr -out server.crt") using the previously generated private key and certificate authority key. The terminal window also shows the system tray with icons for file, clipboard, and other applications.

```
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:
```

```
Activities Terminal Nov 28 18:50
seed@VM: ~/volumes
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:
    keyid:1A:FF:41:64:85:A8:52:EC:3A:21:6F:90:0F:DD:21:50:9B:C5:E
6:4D
X509v3 Subject Alternative Name:
    DNS:www.bank32.com, DNS:www.example.com, DNS:*.banerjee2023.c
om
Certificate is to be certified until Nov 25 23:43:56 2033 GMT (3650 days)
Write out database with 1 new entries
Data Base Updated
[11/28/23]seed@VM:~/volumes$
```

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





This is 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)