

Task 1: TLS Client

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

Task 1.a: TLS handshake

Run the code:

Initiating a TLS handshake with amazon server.

```
[04/07/24]seed@VM:~/.../volumes$ python3 handshake.py www.amazon.com
```

No.	Time	Source	Destination	Protocol	Length	Info
34	2024-04-04 16:4...	10.9.0.43	121.194.14.142	TCP	74	54216 → 443 [SYN] Seq=720410235 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TS...
35	2024-04-04 16:4...	121.194.14.142	10.9.0.43	TCP	58	443 → 54216 [SYN, ACK] Seq=1920001 Ack=720410236 Win=65535 Len=0 MSS=14...
36	2024-04-04 16:4...	10.9.0.43	121.194.14.142	TCP	54	54216 → 443 [ACK] Seq=720410236 Ack=1920002 Win=64240 Len=0
37	2024-04-04 16:4...	02:42:0a:09:00:2b	02:42:0a:09:00:2b	ARP	42	Who has 10.9.0.43? Tell 10.9.0.1
38	2024-04-04 16:4...	02:42:0a:09:00:2b	02:42:0a:09:00:2b	ARP	42	10.9.0.43 is at 02:42:0a:09:00:2b
39	2024-04-04 16:4...	10.9.0.43	121.194.14.142	TLSv1.2	571	Client Hello
40	2024-04-04 16:4...	121.194.14.142	10.9.0.43	TCP	54	443 → 54216 [ACK] Seq=1920002 Ack=720410753 Win=65535 Len=0
41	2024-04-04 16:4...	121.194.14.142	10.9.0.43	TLSv1.2	2974	Server Hello
42	2024-04-04 16:4...	10.9.0.43	121.194.14.142	TCP	54	54216 → 443 [ACK] Seq=720410753 Ack=1922922 Win=62780 Len=0
43	2024-04-04 16:4...	121.194.14.142	10.9.0.43	TLSv1.2	2401	Certificate, Server Key Exchange, Server Hello Done
44	2024-04-04 16:4...	10.9.0.43	121.194.14.142	TCP	54	54216 → 443 [ACK] Seq=720410753 Ack=1925269 Win=60433 Len=0
45	2024-04-04 16:4...	10.9.0.43	121.194.14.142	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
46	2024-04-04 16:4...	121.194.14.142	10.9.0.43	TCP	54	443 → 54216 [ACK] Seq=1925269 Ack=720410879 Win=65535 Len=0
47	2024-04-04 16:4...	121.194.14.142	10.9.0.43	TLSv1.2	296	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
48	2024-04-04 16:4...	10.9.0.43	121.194.14.142	TCP	54	54216 → 443 [ACK] Seq=720410879 Ack=1925511 Win=62780 Len=0
49	2024-04-04 16:4...	10.9.0.43	121.194.14.142	TCP	54	54216 → 443 [FIN, ACK] Seq=720410879 Ack=1925511 Win=62780 Len=0
50	2024-04-04 16:4...	121.194.14.142	10.9.0.43	TCP	54	443 → 54216 [ACK] Seq=1925511 Ack=720410880 Win=65535 Len=0

- Line 34 represents the SYN packet which initiates the TCP three-way handshake.
- Line 39 represents "Client Hello" message which is the first message in TLS handshake.

TLS: The initial step involves the client sending the Client Hello message, followed by the server responding with the Server Hello. Once the client's identity is confirmed, it proceeds to send the key exchange and change cipher spec messages. In response, the server sends the change cipher spec message. This marks the completion of the TLS handshake, allowing subsequent actions to proceed.

TCP: The TCP handshake begins with the client sending a SYN packet to the server, indicating its intention to establish a connection. In response, the server sends a SYN-ACK packet, acknowledging the client's request and indicating its readiness to establish a connection. Finally, the client sends an ACK packet to the server, confirming receipt of the server's acknowledgment.

Cipher used between the client and the server:

TLS_AES_128_GCM_SHA256

The purpose of /etc/ssl/certs:

Stores SSL/TLS certificates issued by Certificate Authorities. It contains trusted root certificates, intermediate certificates, and sometimes client certificates, ensuring the security and integrity of SSL/TLS connections.

The relationship between the TLS handshake and the TCP handshake:

The relationship between the TLS handshake and the TCP handshake is that the TLS handshake occurs within the established TCP connection. TCP sets up the basic connection, and TLS layers security on top of it.

Terminal output:

```
[04/07/24]seed@VM:~/../volumes$ python3 handshake.py www.amazon.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_128_GCM_SHA256', 'TLSv1.3', 128)
=== Server hostname: www.amazon.com
=== Server certificate:
{'OCSP': ('http://ocsp.digicert.com',),
 'caIssuers': ('http://cacerts.digicert.com/DigiCertGlobalCA2.crt',),
 'crlDistributionPoints': ('http://crl3.digicert.com/DigiCertGlobalCA2.crl',
                           'http://crl4.digicert.com/DigiCertGlobalCA2.crl'),
 'issuer': (((('countryName', 'US'),),
               (('organizationName', 'DigiCert Inc'),),
               (('commonName', 'DigiCert Global CA G2'),)),),
 'notAfter': 'Nov 11 23:59:59 2024 GMT',
 'notBefore': 'Nov 28 00:00:00 2023 GMT',
 'serialNumber': '09595080734F983C01FEBD78D23E349C',
 'subject': (((('commonName', 'www.amazon.com'),),),),
 'subjectAltName': (('DNS', 'amazon.com'),
                    ('DNS', 'amzn.com'),
                    ('DNS', 'uedata.amazon.com'),
                    ('DNS', 'us.amazon.com'),
                    ('DNS', 'www.amazon.com'),
                    ('DNS', 'www.amzn.com'),
                    ('DNS', 'corporate.amazon.com'),
                    ('DNS', 'buybox.amazon.com'),
                    ('DNS', 'iphone.amazon.com'),
                    ('DNS', 'yp.amazon.com'),
                    ('DNS', 'home.amazon.com'),
                    ('DNS', 'origin-www.amazon.com'),
```

```
                    ('DNS', 'corporate.amazon.com'),
                    ('DNS', 'buybox.amazon.com'),
                    ('DNS', 'iphone.amazon.com'),
                    ('DNS', 'yp.amazon.com'),
                    ('DNS', 'home.amazon.com'),
                    ('DNS', 'origin-www.amazon.com'),
                    ('DNS', 'buckeye-retail-website.amazon.com'),
                    ('DNS', 'huddles.amazon.com'),
                    ('DNS', 'p-nt-www-amazon-com-kalias.amazon.com'),
                    ('DNS', 'p-yo-www-amazon-com-kalias.amazon.com'),
                    ('DNS', 'p-y3-www-amazon-com-kalias.amazon.com'),
                    ('DNS', 'yellowpages.amazon.com'),
                    ('DNS', 'www.m.amazon.com'),
                    ('DNS', 'www.cdn.amazon.com'),
                    ('DNS', 'test-www.amazon.com'),
                    ('DNS', 'mp3recs.amazon.com'),
                    ('DNS', 'konrad-test.amazon.com')),
 'version': 3}
[{'issuer': (((('countryName', 'US'),),
               (('organizationName', 'DigiCert Inc'),),
               (('organizationalUnitName', 'www.digicert.com'),),
               (('commonName', 'DigiCert Global Root G2'),)),),
 'notAfter': 'Jan 15 12:00:00 2038 GMT',
 'notBefore': 'Aug 1 12:00:00 2013 GMT',
 'serialNumber': '033AF1E6A711A9A0BB2864B11D09FAE5',
 'subject': (((('countryName', 'US'),),
               (('organizationName', 'DigiCert Inc'),),
               (('organizationalUnitName', 'www.digicert.com'),),
               (('commonName', 'DigiCert Global Root G2'),)),),
 'version': 3}]
```

Task 1.b: CA's Certificate

```
Open  handshake.py  Save  -  x
~/Labs/06_tls/Labsetup/volumes

4 import ssl
5 import sys
6 import pprint
7
8 hostname = sys.argv[1]
9 port = 443
10 #cadir = '/etc/ssl/certs'
11 cadir = './client-certs'
12
```

Change the default directory for CA certificates.

Run the code again:

```
[04/07/24]seed@VM:~/../volumes$ python3 handshake.py www.amazon.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)
```

Certificate validation failed as the corresponding CA certificate was not found.

We have seen previously:

```
{'issuer': (((('countryName', 'US'),),
              (('organizationName', 'DigiCert Inc'),),
              (('organizationalUnitName', 'www.digicert.com'),),
              (('commonName', 'DigiCert Global Root G2'),)),
 'notAfter': 'Jan 15 12:00:00 2038 GMT',
 'notBefore': 'Aug 1 12:00:00 2013 GMT',
 'serialNumber': '033AF1E6A711A9A0BB2864B11D09FAE5',
 'subject': (((('countryName', 'US'),),
                (('organizationName', 'DigiCert Inc'),),
                (('organizationalUnitName', 'www.digicert.com'),),
                (('commonName', 'DigiCert Global Root G2'),)),
 'version': 3}]
```

Locate this certificate and place it into the client-certs folder.

```
[04/07/24]seed@VM:~/../volumes$ ll /etc/ssl/certs/ | grep DigiCert_Global_Root_G2
lrwxrwxrwx 1 root root 27 Nov 24 2020 607986c7.0 -> DigiCert_Global_Root_G2.pem
lrwxrwxrwx 1 root root 62 Nov 24 2020 DigiCert_Global_Root_G2.pem -> /usr/share
/ca-certificates/mozilla/DigiCert_Global_Root_G2.crt

[04/07/24]seed@VM:~/../volumes$ cp /usr/share/ca-certificates/mozilla/DigiCert_Global_Root_G2.crt client-certs/DigiCert_Global_Root_G2.crt
[04/07/24]seed@VM:~/../volumes$ cd client-certs/
[04/07/24]seed@VM:~/../client-certs$ ll
total 8
-rw-r--r-- 1 seed seed 1294 Apr 7 06:18 DigiCert_Global_Root_G2.crt
-rw-rw-r-- 1 seed seed 103 Jan 2 2021 README.md
```

Calculate the subject hash of the certificate "DigiCert_Global_Root_G2.crt"

Then, create a symbolic link with the filename "607986c7.0" pointing to the certificate file "DigiCert_Global_Root_G2.crt".


```
root@318fdf3b14ba:/volumes/client-certs# openssl x509 -in DigiCert_Global_Root_G2.crt -no
out -subject_hash
607986c7
root@318fdf3b14ba:/volumes/client-certs# ln -s DigiCert_Global_Root_G2.crt 607986c7.0
root@318fdf3b14ba:/volumes/client-certs# ll
total 8
lrwxrwxrwx 1 root root 27 Apr 7 10:21 607986c7.0 -> DigiCert_Global_Root_G2.crt
-rw-r--r-- 1 seed seed 1294 Apr 7 10:18 DigiCert_Global_Root_G2.crt
-rw-rw-r-- 1 seed seed 103 Jan 2 2021 README.md
```

This step is used to ensure that the certificate can be properly referenced by applications that require it, such as those performing TLS/SSL connections.

Creating symbolic links for CA certificates is necessary because the TLS verification process relies on filenames generated from hash values to locate issuer certificates. Symbolic links ensure that these filenames match the hash values, simplifying certificate management and enabling dynamic updates without manual intervention.

Run the code again:

```
[04/07/24]seed@VM:~/../volumes$ python3 handshake.py www.amazon.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_128_GCM_SHA256', 'TLSv1.3', 128)
=== Server hostname: www.amazon.com
=== Server certificate:
{'OCSP': ('http://ocsp.digicert.com',),
 'caIssuers': ('http://cacerts.digicert.com/DigiCertGlobalCA2.crt',),
 'crlDistributionPoints': ('http://crl3.digicert.com/DigiCertGlobalCA2.crl',
                           'http://crl4.digicert.com/DigiCertGlobalCA2.crl'),
 'issuer': (((('countryName', 'US'),),
               (('organizationName', 'DigiCert Inc'),),
               (('commonName', 'DigiCert Global CA G2'),)),),
 'notAfter': 'Nov 11 23:59:59 2024 GMT',
 'notBefore': 'Nov 28 00:00:00 2023 GMT',
 'serialNumber': '09595080734F983C01FEBD78D23E349C',
 'subject': (((('commonName', 'www.amazon.com'),),),),
 'subjectAltName': (('DNS', 'amazon.com'),
                    ('DNS', 'amzn.com'),
                    ('DNS', 'uedata.amazon.com'),
                    ('DNS', 'us.amazon.com'),
                    ('DNS', 'www.amazon.com'),
                    ('DNS', 'www.amzn.com'),
                    ('DNS', 'corporate.amazon.com'),
                    ('DNS', 'buybox.amazon.com'),
                    ('DNS', 'iphone.amazon.com'),
                    ('DNS', 'yp.amazon.com'),
```

We have successfully connected to 'www.amazon.com'. 

Task 1.c: Experiment with the hostname check

Check the IP of Amazon homepage:

```
[04/07/24]seed@VM:~/../volumes$ dig www.amazon.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.amazon.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 54980
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.amazon.com.                IN      A

;; ANSWER SECTION:
www.amazon.com.                670     IN      CNAME   tp.47cf2c8c9-frontier.amazon.com.
tp.47cf2c8c9-frontier.amazon.com. 14     IN      CNAME   www.amazon.com.edgekey.net.
www.amazon.com.edgekey.net.    7177    IN      CNAME   e15316.dsca.akamaiedge.net.
e15316.dsca.akamaiedge.net.    18      IN      A       184.29.68.98

;; Query time: 19 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sun Apr 07 05:36:22 EDT 2024
;; MSG SIZE rcvd: 172
```

Modify the “/etc/hosts”:

```
31
32 184.29.68.98 www.amazon2024.com
33
```

Switch the following line in the client program between True and False.

```
19 context.check_hostname = True
```

```
[04/07/24]seed@VM:~/../volumes$ python3 handshake.py www.amazon2024.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.amazon2024.com'. (_ssl.c:1123)
```

Certificate validation fails due to inconsistency in domain names.

```
19 context.check_hostname = False
```

```
[04/07/24]seed@VM:~/../volumes$ python3 handshake.py www.amazon2024.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_128_GCM_SHA256', 'TLSv1.3', 128)
=== Server hostname: www.amazon2024.com
=== Server certificate:
{'OCSP': ('http://ocsp.digicert.com',),
 'caIssuers': ('http://cacerts.digicert.com/DigiCertGlobalCA2.crt',),
 'crlDistributionPoints': ('http://crl3.digicert.com/DigiCertGlobalCA2.crl',
 'http://crl4.digicert.com/DigiCertGlobalCA2.crl'),
 'issuer': (((('countryName', 'US'),),
              (('organizationName', 'DigiCert Inc'),),
              (('commonName', 'DigiCert Global CA G2'),)),),
 'notAfter': 'Jan 14 23:59:59 2025 GMT',
 'notBefore': 'Feb 6 00:00:00 2024 GMT',
 'serialNumber': '0A64C321ECABA44EFCE1011050512E36',
 'subject': (((('commonName', 'www.amazon.com'),),),),
 'subjectAltName': (('DNS', 'amazon.com'),
                    ('DNS', 'amzn.com'),
                    ('DNS', 'buybox.amazon.com'),
                    ('DNS', 'corporate.amazon.com'),
                    ('DNS', 'home.amazon.com'),
                    ('DNS', 'iphone.amazon.com'),
                    ('DNS', 'mp3recs.amazon.com'),
                    ('DNS', 'p-nt-www-amazon-com-kalias.amazon.com'),
                    ('DNS', 'p-y3-www-amazon-com-kalias.amazon.com'),
                    ('DNS', 'p-yo-www-amazon-com-kalias.amazon.com'),
                    ('DNS', 'uedata.amazon.com'),
                    ('DNS', 'us.amazon.com'),
                    ('DNS', 'www.amazon.com'),
```

The verification passes directly at this point, this shows the importance of domain name checking.

The importance of hostname check:

Hostname check is crucial for verifying the authenticity of TLS connections. Without it, the client risks connecting to malicious servers, exposing sensitive data to interception and manipulation.

Task 1.d: Sending and getting Data

```
b'.HTTP/1.1 200 OK'
b'Content-Type: text/html',
b'Connection: close',
b'Server: Server',
b'Date: Sun, 07 Apr 2024 09:47:01 GMT',
b'x-amz-rid: XTCB5M9HJC1MM9QZRES',
b'Vary: Content-Type,Accept-Encoding,User-Agent',
b'Strict-Transport-Security: max-age=474747; includeSubDomains; preload',
b'X-Cache: Miss from cloudfront',
b'Via: 1.1 01ff2b265b9f2ac4574d3d644dd9dd26.cloudfront.net (CloudFront)',
b'X-Amz-Cf-Pop: BAH53-P1',
b'Alt-Svc: h3=":443"; ma=86400',
b'X-Amz-Cf-Id: sEB95wDGeQesAeOUU-qu5bkbstvTgt5T1C80PckQ2MvyqHwmi01H0wQ==',
b'',
b'<!DOCTYPE html>\n<!--[if lt IE 7]> <html lang="en-us" class="a-no-js a-lt-  
b'ie9 a-lt-ie8 a-lt-ie7"> <![endif]-->\n<!--[if IE 7]> <html lang="en-u  
b's" class="a-no-js a-lt-ie9 a-lt-ie8"> <![endif]-->\n<!--[if IE 8]> <ht  
b'm! lang="en-us" class="a-no-js a-lt-ie9"> <![endif]-->\n<!--[if gt IE 8]>  
b'<!-->\n<html class="a-no-js" lang="en-us"><!--<![endif]--><head>\n<meta ht  
b't!p-equiv="content-type" content="text/html; charset=UTF-8">\n<meta charse  
b't="utf-8">\n<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"  
b">\n<title dir="ltr">Amazon.com</title>\n<meta name="viewport" content="wid  
b'th=device-width">\n<link rel="stylesheet" href="https://images-na.ssl-ima  
b'ges-amazon.com/images/G/01/AUIClients/AmazonUI-3c913031596ca78a3768f4e934blc  
b'02ce238101.secure_min_v1.css">\n<script>\n\nif (true == true) {\n  var  
b'ue_t0 = (+ new Date()),\nue_csm = window,\nue = { t0: ue_t  
b't0, d: function() { return (+new Date() - ue.t0); } },\nue_furl = '  
b'"fls-na.amazon.com",\nue_mid = "ATVPDKIKX0DER",\nue_sid = '
```

The page source code of the Amazon homepage is successfully obtained.

[illegible]

Image is fetched successfully.

Task 2: TLS Server

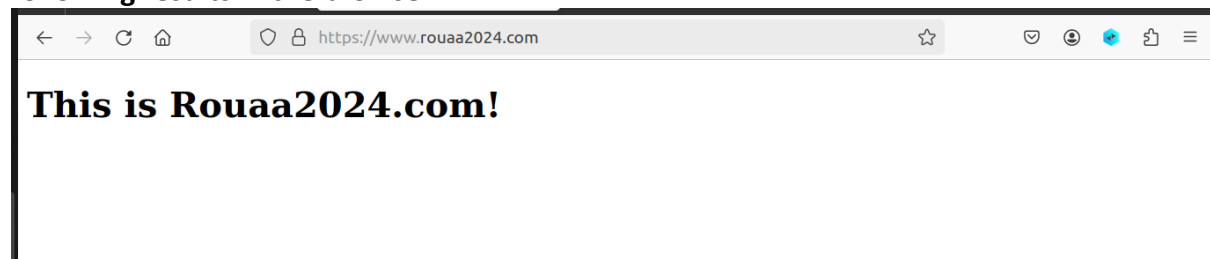
You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

Task 2.a. Implement a simple TLS server

- This part needs to use some files generated in the seedlab Crypto_PKI Lab. In that lab, we set CN to be the server certificate of `www.rouaa2024.com` and related files. In this lab, we will use these files as the certificate of the server program. And send it back to the client program.
 - There are a few things to note:
 - In the `server.py` code, we use `sock.bind(('0.0.0.0', 4433))` to bind the listening port 4433, so, we need to make some modifications based on the client code in task1, that is, modify the port number (Otherwise the client program and server program will not run successfully)
 - We need to configure the static IP in the local `/etc/hosts` file and add the `10.9.0.43 www.rouaa2024.com` entry.
 - Steps to test using the client:
 1. Pre-requisites:
 - Copy the `ca.key` and `ca.crt` into the `client-certs` folder and perform the hashing step, similar to the one done in task 1.
 - Copy the `server.key` and `server.crt` into the `server-certs` folder and perform the hashing step, similar to the one done in task 1.
 2. Open 2 different terminals:
 - Terminal 1:
 - Go into the server's container and run the `server.py`
 - Terminal 2:
 - Go into the client's container and run the `handshake.py`
- ```
version: 0.1
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 Rouaa2024.com!</h1></body></html>'
 b'>\n']
```

## Task 2.b. Testing the server program using browsers

Since in the Crypto PKI experiment, the root CA certificate created by us has been installed in the browser, we can directly test it. After running the `server.py` program, we got the following results in the browser:



## Task 2.c. Certificate with multiple names

- Create an OpenSSL Configuration File:

- must includes the SAN (Subject Alternative Name) details. This file tells OpenSSL what information to include in the certificate. (server\_openssl.cnf)
- **Generate the Certificate Signing Request (CSR) and Key using the server\_openssl.cnf**
- **Sign the CSR using myopenssl.cnf (copied from /usr/lib/ssl)**

```
[04/17/24]seed@VM:~/.../volumes$ openssl req -new -config server_openssl.cnf -keyout ./server-certs/server.key -out ./server-certs/server.csr
Generating a RSA private key
.....+++++
.....+++++
writing new private key to './server-certs/server.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

[04/17/24]seed@VM:~/.../volumes$ openssl ca -md sha256 -days 3650 -config ./myopenssl.cnf -batch \
> -in ./server-certs/server.csr -out ./server-certs/server.crt \
> -cert ./client-certs/ca.crt -keyfile ./client-certs/ca.key
Using configuration from ./myopenssl.cnf
Enter pass phrase for ./client-certs/ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
 Serial Number: 4096 (0x1000)
 Validity
 Not Before: Apr 17 15:51:38 2024 GMT
 Not After : Apr 15 15:51:38 2034 GMT
 Subject:
 countryName = QA
 stateOrProvinceName = DOHA
 organizationName = QU
 commonName = rouaa
 X509v3 extensions:
 X509v3 Basic Constraints:
 CA:FALSE
 Netscape Comment:
 OpenSSL Generated Certificate
 X509v3 Subject Key Identifier:
 23:4F:F2:EC:50:4A:24:37:AE:7F:BB:50:40:EB:FD:C4:3C:A2:0A:CA
 X509v3 Authority Key Identifier:
 keyid:4F:9A:29:EC:36:15:79:4E:8D:0F:5D:80:64:CE:2B:BA:0D:EA:D3:F
0

Certificate is to be certified until Apr 15 15:51:38 2034 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[04/17/24]seed@VM:~/.../volumes$ █
```

- **Run the server.py program in one Terminal, and run the client program in task1 in the other Terminal, and use the host name www.rouaa2025.com in the alt\_name field to access (because it is tested on this machine, it is necessary to configure a static IP in the etc/hosts file , 10.9.0.43 www.rouaa2025.com entry:**



```
root@2d520dal24f8:/volumes# ./handshake.py www.rouaa2025.com
After making TCP connection. Press any key to continue ...
=== Cipher used: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
=== Server hostname: www.rouaa2025.com
=== Server certificate:
{'issuer': (((('countryName', 'QA'),),
 (('stateOrProvinceName', 'DOHA'),),
 (('localityName', 'ALHILAL'),),
 (('organizationName', 'QU'),),
 (('organizationalUnitName', 'ENG'),),
 (('commonName', 'rouaa'),),
 (('emailAddress', 'rn2106763@qu.edu.qa'),)),
 'notAfter': 'Apr 15 17:18:18 2034 GMT',
 'notBefore': 'Apr 17 17:18:18 2024 GMT',
 'serialNumber': '1004',
 'subject': (((('countryName', 'QA'),),
 (('stateOrProvinceName', 'DOHA'),),
 (('organizationName', 'QU'),),
 (('commonName', 'www.rouaa2024.com'),)),
 'subjectAltName': (('DNS', 'www.rouaa2024.com'),
 ('DNS', 'www.rouaa2025.com'),
 ('DNS', '*.rouaa2024.com')),
 'version': 3}
[{'issuer': (((('countryName', 'QA'),),
 (('stateOrProvinceName', 'DOHA'),),
 (('localityName', 'ALHILAL'),),
 (('organizationName', 'QU'),),
 (('organizationalUnitName', 'ENG'),),
 (('commonName', 'rouaa'),),
 (('emailAddress', 'rn2106763@qu.edu.qa'),)),
 'notAfter': 'Mar 16 19:48:27 2034 GMT',
 'notBefore': 'Mar 18 19:48:27 2024 GMT',
 'serialNumber': '422309A3BBC6E95D82C2DED760791D6A831C6AC1',
 'subject': (((('countryName', 'QA'),),
 (('stateOrProvinceName', 'DOHA'),),
 (('localityName', 'ALHILAL'),),
 (('organizationName', 'QU'),),
 (('organizationalUnitName', 'ENG'),),
 (('commonName', 'rouaa'),),
 (('emailAddress', 'rn2106763@qu.edu.qa'),)),
 '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 Rouaa2024.com!</h1></body></html>'
 b'>\n']
```

### Task 3: A Simple HTTPS Proxy

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

#### Step 1:

**First, we need to write the code for proxy file.**

**The code contains:**

The code creates a simple HTTPS proxy server. It sets up a server that listens for incoming connections, accepts HTTPS requests from clients, forwards them to a target server (in our case, "leetcode.com"), and relays the responses back to the clients.

**SSL/TLS Configuration:** Sets up SSL/TLS contexts for both client and server sides to ensure secure communication. It specifies CA certificates, verification modes, and hostname verification.

**Server Setup:** Initializes an SSL context for the server, loads the server's certificate and private key, and creates a TCP socket to listen for incoming connections.

**Main Loop:** Starts an infinite loop to accept incoming connections from clients. Upon accepting a connection, it wraps the client socket with SSL/TLS, spawns a new thread to handle the client request, and continues listening for new connections.

#### Launching MITM Attack Against Our Own Server:

**The client setup:**

Since we will use a browser, we will use the hosting VM as the client/victim, instead of using the client container.

- Add the following entry to **/etc/hosts** in the Host VM.

```
10.9.0.143 www.bawazir2024.com
```

This setup mimics the presence of a proxy, as it routes traffic intended for **www.bawazir2024.com** through the IP address **10.9.0.143**.

- Add **nameserver 8.8.8.8** to the **/etc/resolv.conf** in the **proxy** container.
- Modify the **proxy.py** code.

```
cadir = './client-certs'
```

```
def process_request(sock_for_browser):
 hostname = "www.bawazir2024.com"
```

```
35 # Get response from server, and forward it to browser
36 response = sock_for_server.recv(2048)
37 response = response.replace(b"bawazir2024", b"mHTTPSPproxy")
38 while response:
39 sock_for_browser.sendall(response) # Forward to browser
40 response = sock_for_server.recv(2048)
41 response = response.replace(b"bawazir2024", b"mHTTPSPproxy")
42
```

- Add **10.9.0.43 www.bawazir2024.com** to the `/etc/hosts/` of the proxy container.

```
GNU nano 4.8 /etc/hosts
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
10.9.0.43 www.bawazir2024.com
```

### Launch the MITM attack against our own server...

- Run `server.py` in the `server` container, and `proxy.py` in the `proxy` container.
- Go to the browser and type: [www.bawazir2024.com](https://www.bawazir2024.com)



The HTML response contains the modified HTML content with the substring "Bawazir2024" replaced by "mHTTPSPproxy". This indicates that our MITM attack was successful.

### Launching MITM Attack On a Real HTTPS Website:

We need to copy `openssl.cnf` and rename it to `cf_openssl.cnf` for the new proxy server. Then we need to generate `cf.crt` `cf.key` for the server just like what we did in both tasks 1 & 2.

- Copy `server_openssl.cnf` and rename it to `leetcode_openssl.cnf` for the new proxy server. Modify the content of `leetcode_openssl.cnf` file.

```

proxy.py *leetcode_openssl.cnf
1 [req]
2 prompt = no
3 distinguished_name = req_distinguished_name
4 req_extensions = req_ext
5
6 [req_distinguished_name]
7 C = QA
8 ST = Doha
9 L = Doha
10 O = QU
11 CN = leetcode.com
12
13 [req_ext]
14 subjectAltName = @alt_names
15
16 [alt_names]
17 DNS.1 = leetcode.com

```

- Generate leetcode.crt & leetcode.key.

```

[04/23/24]seed@VM:~/../volumes$ openssl req -new -config leetcode_openssl.cnf -keyout ./server-certs/leetcode.key -out ./server-certs/leetcode.csr
Generating a RSA private key
.....+++++
.....+++++
writing new private key to './server-certs/leetcode.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

```

```

seed@VM: ~/../volumes
[04/23/24]seed@VM:~/../volumes$ openssl ca -md sha256 -days 3650 -config ./openssl.cnf -batch -in ./server-certs/leetcode.csr -out ./server-certs/leetcode.crt -cert ./client-certs/ca.crt -keyfile ./client-certs/ca.key
Using configuration from ./openssl.cnf
Enter pass phrase for ./client-certs/ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
 Serial Number: 4111 (0x100f)
 Validity
 Not Before: Apr 23 13:45:03 2024 GMT
 Not After : Apr 21 13:45:03 2034 GMT
 Subject:
 countryName = QA
 stateOrProvinceName = Doha
 organizationName = QU
 commonName = leetcode.com
 X509v3 extensions:
 X509v3 Basic Constraints:
 CA:FALSE
 Netscape Comment:
 OpenSSL Generated Certificate
 X509v3 Subject Key Identifier:
 66:89:C9:61:7B:1E:C6:57:7E:08:69:C3:E3:86:F8:8B:AE:B3:DC:E5
 X509v3 Authority Key Identifier:
 keyid:74:3A:95:EC:44:F7:A4:99:CA:43:08:3B:A0:DE:21:DD:F7:9B:0D:30

 X509v3 Subject Alternative Name:
 DNS:leetcode.com
Certificate is to be certified until Apr 21 13:45:03 2034 GMT (3650 days)

```

- Add the following entry to **/etc/hosts** in the Host VM.

```
10.9.0.143 leetcode.com
```

- Add **nameserver 8.8.8.8** to the **/etc/resolv.conf** in the **proxy** container.
- Modify the proxy.py code.

```

6 cadir = "/etc/ssl/certs"
7 #cadir = './client-certs'
8
9 def process_request(sock_for_browser):
10 hostname = "leetcode.com"

```

```

51 SERVER_CERT = './server-certs/leetcode.crt'
52 SERVER_PRIVATE = './server-certs/leetcode.key'

```

- Run proxy.py

To ensure our code works, and it is successfully mentoring, we need to visit leetcode.com.



```

root@874f50923023:/volumes# python3 proxy.py
Enter PEM pass phrase:
('10.9.0.1', 35360)
("Request: b'GET / HTTP/1.1\\r\\nHost: leetcode.com\\r\\nUser-Agent: "
'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 '
'Firefox/83.0\\r\\nAccept: '
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\\r\\nAccept-Language: '
'en-US,en;q=0.5\\r\\nAccept-Encoding: gzip, deflate, br\\r\\nConnection: '
'keep-alive\\r\\nCookie: '
'csrftoken=89J01a5Fpdj0wjVw1chbppsVrptjOMPqsJBjilhmjCytVnPTAM5SrPplPD1liIZe; '
'_ga_CDRWKZTDEX=GS1.1.1713882718.3.1.1713883701.60.0.0; '
'_ga=GA1.2.869909478.1713875631; _gid=GA1.2.1777508889.1713875634; '
'_gr_user_id=8ba05013-b0e0-4822-a229-f9e9033d4dbe\\r\\nUpgrade-Insecure-Requests: '
'1\\r\\nCache-Control: max-age=0\\r\\n\\r\\n")
('10.9.0.1', 35368)
('10.9.0.1', 35370)
('10.9.0.1', 35374)
('10.9.0.1', 35376)
('10.9.0.1', 35378)

```