Name: Kwa Li Ying
Student ID: 1003833

50.020 Network Security Lab 4: Public-Key Infrastructure (PKI)

Task 1: Becoming a Certificate Authority (CA)

The OpenSSL configuration file is copied from /usr/lib/ssl/ into the current directory ~/Documents/lab4/

```
[02/28/21]seed@VM:.../ssl$ cp openssl.cnf ~/Documents/ns/lab4/
[02/28/21]seed@VM:.../ssl$ cd -
/home/seed/Documents/ns/lab4
[02/28/21]seed@VM:~/.../lab4$ ls
openssl.cnf
```

From the configuration file, these are the subdirectories that need to be created:

```
[ CA_default ]

dir             = ./demoCA              # Where everything is kept
certs           = $dir/certs            # Where the issued certs are kept
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt        # database index file.
#unique_subject = no                    # Set to 'no' to allow creation of
                                        # several ctificates with same subject.
new_certs_dir   = $dir/newcerts         # default place for new certs.

certificate     = $dir/cacert.pem       # The CA certificate
serial          = $dir/serial           # The current serial number
crlnumber       = $dir/crlnumber        # the current crl number
                                        # must be commented out to leave a V1 CRL
crl             = $dir/crl.pem          # The current CRL
private_key     = $dir/private/cakey.pem# The private key
RANDFILE        = $dir/private/.rand    # private random number file

x509_extensions = usr_cert              # The extentions to add to the cert
```

The following subdirectories are created accordingly:

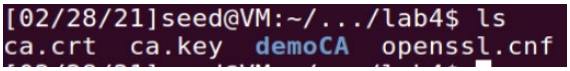
```
[02/28/21]seed@VM:~/.../lab4$ ls -R
.:
demoCA  openssl.cnf

./demoCA:
certs  crl  index.txt  newcerts  serial

./demoCA/certs:

./demoCA/crl:

./demoCA/newcerts:
```

The following command is executed to generate the self-signed certificate. The passphrase used is *liying*. The rest of the fields are left as default:

```
[02/28/21]seed@VM:~/.../lab4$ openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf
Generating a 2048 bit RSA private key
...........................+++
.....................................+++
writing new private key to 'ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
[02/28/21]seed@VM:~/.../lab4$
```

Name: Kwa Li Ying
Student ID: 1003833

The private key and certificate are generated in the current directory as ca.key and ca.crt respectively:



Name: Kwa Li Ying
Student ID: 1003833

Task 2: Creating a Certificate for SEEDPKILab2020.com

Step 1: Generate public/private key pair

The following command is run to create the company's public/private key pair. The password *liyingcompany* is used:

```
[02/28/21]seed@VM:~/.../lab4$ openssl genrsa -aes128 -out server.key 1024
Generating RSA private key, 1024 bit long modulus
.....................................++++++
...++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
```

The keys are store in the file server.key in the current directory:

```
[02/28/21]seed@VM:~/.../lab4$ ls
ca.crt  ca.key  demoCA  openssl.cnf  server.key
```

To view the actual content of the encrypted key-file, the following command is run:

```
Enter pass phrase for server.key:
Private-Key: (1024 bit)
modulus:
    00:bb:46:25:57:ee:47:fc:b9:70:81:0f:9b:91:ae:
    5e:33:18:ef:c2:ff:35:84:27:fb:85:ac:98:be:e1:
    77:8a:7d:80:18:b6:a6:83:99:42:39:40:87:3c:6e:
    16:c4:29:99:0e:ac:36:46:78:55:3a:92:24:b2:04:
    fd:0a:63:7f:4e:c0:e8:be:92:62:c0:ee:45:0c:50:
    22:8b:1b:08:6e:c1:6c:ae:1d:e2:87:8b:be:d9:45:
    45:67:63:df:13:57:c1:4a:0a:9f:3e:96:fc:b2:f5:
    47:fc:cc:b3:a6:82:45:5e:a8:62:3e:79:bd:d0:1c:
    3d:5a:fb:0f:d8:dc:1d:43:e1
publicExponent: 65537 (0x10001)
privateExponent:
    76:9c:42:e2:a2:44:6d:c8:75:7d:08:89:9b:87:38:
    9a:5c:5c:de:c0:0d:01:1a:e1:af:14:55:d9:ef:ab:
    6a:c7:79:ee:2d:20:e1:90:9b:e4:ee:fd:b8:44:71:
    9e:e4:49:d0:85:3a:0b:00:03:db:08:dc:bc:f3:73:
    e0:b4:8b:d8:1b:b8:fa:ba:fc:8b:4f:a3:a6:00:19:
    08:41:20:da:55:7b:78:50:e8:f4:70:7a:3c:52:0e:
    77:3d:23:a1:ea:1c:32:23:a8:8d:b2:e6:9c:4b:b6:
    b0:be:42:d4:c7:0d:60:eb:2f:17:cd:8f:97:ae:08:
    e1:27:fd:f0:92:42:d3:01
prime1:
    00:e3:f6:8b:f2:01:59:39:3d:02:16:73:ec:e5:4a:
    6d:da:2a:53:44:3c:d4:8b:bf:dc:87:7c:de:de:94:
    e5:ea:79:61:8e:fa:ae:f9:c0:41:33:f5:00:0b:75:
    a8:0f:5f:2c:b0:5f:67:20:fa:d8:6b:57:3c:0b:6c:
    5a:45:93:cd:79
```

Name: Kwa Li Ying
Student ID: 1003833

```
prime2:
    00:d2:4e:7f:5f:f8:c4:05:5c:f7:eb:fa:4d:9f:e3:
    e0:0d:a2:6e:e7:8d:a3:40:71:2d:f4:c1:65:4b:5f:
    27:19:c2:97:19:05:3b:ef:e7:36:f3:ce:1c:b8:a2:
    8d:f7:89:43:81:72:80:ec:83:7b:ef:c1:25:45:58:
    fc:df:4a:d7:a9
exponent1:
    4a:61:bf:f8:0f:08:95:ec:9a:29:c9:59:9a:d7:56:
    50:c1:4e:ba:0d:3f:2c:fa:45:72:d0:03:c8:8c:bd:
    18:6c:d2:b0:5c:8b:8b:62:77:e4:04:25:27:98:14:
    66:2f:9f:dd:4c:c3:d1:b7:07:b6:be:98:11:02:21:
    d2:62:0f:c1
exponent2:
    00:98:97:0a:06:59:59:e8:c6:46:c0:3a:31:9e:44:
    59:a9:aa:e4:ab:2f:72:76:ec:67:ba:c1:a3:bc:67:
    42:d8:86:fe:d8:d1:9a:66:7b:ad:bc:82:1d:06:be:
    33:21:9a:bf:97:29:bc:6f:5f:0b:4a:af:2a:5a:c1:
    b1:60:56:4e:59
coefficient:
    70:6c:d1:83:a3:b5:8e:77:fc:8b:23:11:d9:f8:fc:
    ab:0f:43:60:6f:fe:e3:75:8a:07:15:61:e1:8b:4d:
    09:39:06:f7:f3:19:b6:33:96:8b:ce:2f:44:1c:04:
    3d:b7:3e:a4:7b:5c:aa:d8:ec:ea:63:9e:54:32:a4:
    5f:5c:d5:35
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC7RiVX7kf8uXCBD5uRrl4zGO/C/zWEJ/uFrJi+4XeKfYAYtqaD
mUI5QIc8bhbEKZkOrDZGeFU6kiSyBP0KY39OwOi+kmLA7kUMUCKLGwhuwWyuHeKH
i77ZRUVnY98TV8FKCp8+lvyy9Uf8zLOmgkVeqGI+eb3QHD1a+w/Y3B1D4QIDAQAB
AoGADpxC4qJEbch1fQiJm4c4mlxc3sANARrhrxRV2e+rasd57i0g4ZCb5O79uERx
nuRJ0IU6CwAD2wjcvPNz4LSL2Bu4+rr8i0+jpgAZCEEg2lV7eFDo9HB6PFIOdz0j
oeocMiOojbLmnEu2sL5C1McNYOsvF82Pl64I4Sf98JJC0wECQQDj9ovyAVk5PQIW
c+zlSm3aKlNEPNSLv9yHfN7elOXqeWGO+q75wEEz9QALdagPXyywX2cg+thrVzwL
bFpFk815AkEA0k5/X/jEBVz36/pNn+PgDaJu542jQHEt9MFlS18nGcKXGQU77+c2
884cuKKN94lDgXKA7IN778ElRVj830rXqQJASmG/+A8IleyaKclZmtdWUMFOug0/
LPpFctADyIy9GGzSsFyLi2J35AQlJ5gUZi+f3UzD0bcHtr6YEQIh0mIPwQJBAJiX
CgZZWejGRsA6MZ5EWamq5KsvcnbsZ7rBo7xnQtiG/tjRmmZ7rbyCHQa+MyGav5cp
vG9fC0qvKlrBsWBWTlkCQHBs0YOjtY53/IsjEdn4/KsPQ2Bv/uN1igcVYeGLTQk5
BvfzGbYzlovOL0QcBD23PqR7XKrY7OpjnlQypF9c1TU=
-----END RSA PRIVATE KEY-----
```

Step 2: Generate a Certificate Signing Request (CSR)

The following command is run to generate a certificate signing request for the company. The Organization Name and the Common Name are both specified SEEDPKILab2020.com. The rest of the fields are left as default:

```
[02/28/21]seed@VM:~/.../lab4$ openssl req -new -key server.key -out server.csr -config openssl.cnf
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SEEDPKILab2020.com
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:SEEDPKILab2020.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

The CSR is generated as server.csr in the current directory:

```
[02/28/21]seed@VM:~/.../lab4$ ls
ca.crt  ca.key  demoCA  openssl.cnf  server.csr  server.key
```

Name: Kwa Li Ying
Student ID: 1003833

Step 3: Generating Certificates

The following command is run to turn the CSR into an X509 certificate using the CA's certificate and key. As expected, OpenSSL refuses to generate the certificate as the name in the request (SEEDPKILab2020.com) is not the same as that of the CA (Internet Widgits Pty Ltd):

```
[02/28/21]seed@VM:~/.../lab4$ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
The organizationName field needed to be the same in the
CA certificate (Internet Widgits Pty Ltd) and the request (SEEDPKILab2020.com)
```

The matching rule policy is then changed in the configuration file from policy_match to policy_anything:

```
# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy          = policy_anything
```

The command is re-run and the certificate generation succeeds after the policy is changed:

```
[02/28/21]seed@VM:~/.../lab4$ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4097 (0x1001)
        Validity
            Not Before: Mar  1 00:41:48 2021 GMT
            Not After : Mar  1 00:41:48 2022 GMT
        Subject:
            countryName               = AU
            stateOrProvinceName       = Some-State
            organizationName          = SEEDPKILab2020.com
            commonName                = SEEDPKILab2020.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                44:64:66:7F:65:5E:95:FA:AA:56:7C:23:5C:E1:FB:91:B3:6D:65:4C
            X509v3 Authority Key Identifier:
                keyid:BE:13:AD:37:A0:E4:CA:E5:30:25:9F:88:79:24:29:BC:4B:B0:D6:99

Certificate is to be certified until Mar  1 00:41:48 2022 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

The company's certificate is generated as server.crt in the current directory:

```
[02/28/21]seed@VM:~/.../lab4$ ls
ca.crt  ca.key  demoCA  openssl.cnf  server.crt  server.csr  server.key
```

Name: Kwa Li Ying
Student ID: 1003833

Task 3: Deploying Certificate in an HTTPS Web Server

Step 1: Configuring DNS

The /etc/hosts file is edited to map the hostname SEEDPKILab2020.com to our localhost IP:

```
127.0.0.1       SEEDPKILab2020.com
```

Step 2: Configuring the web server

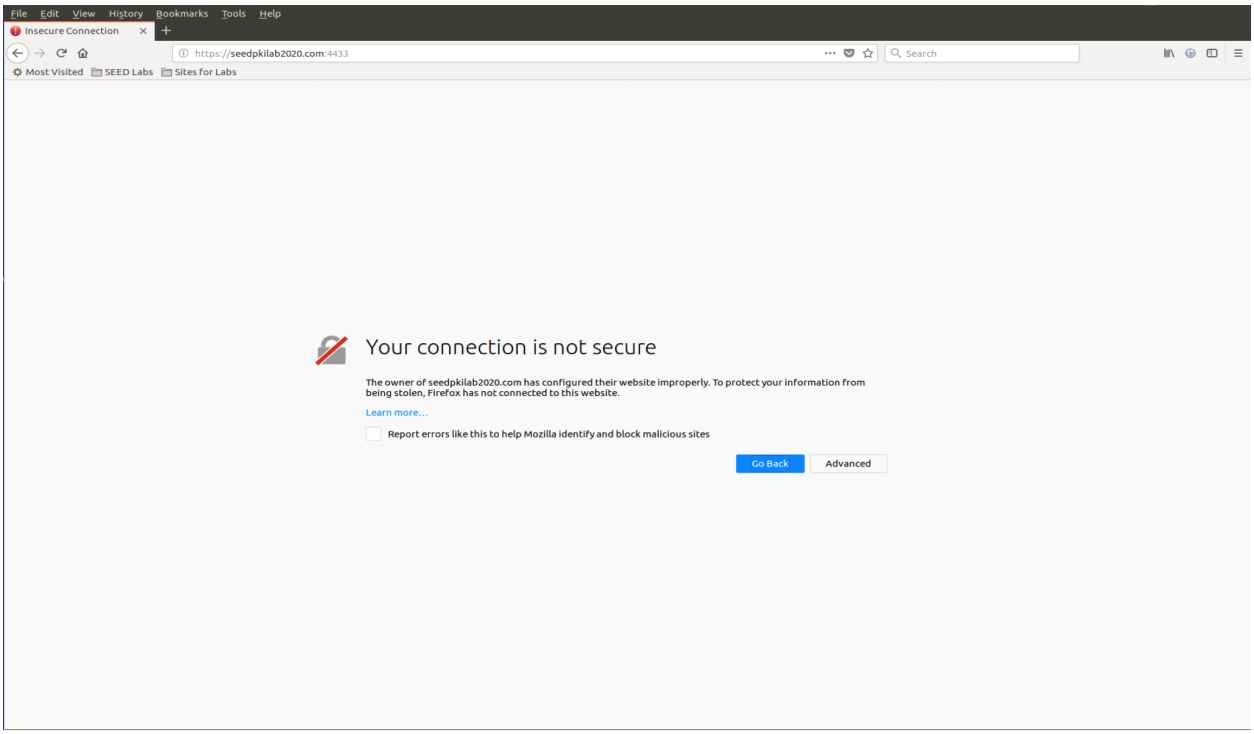
The server's key and certificate is combined into one file, named server.pem:

```
[02/28/21]seed@VM:~/.../lab4$ cat server.crt >> server.pem
[02/28/21]seed@VM:~/.../lab4$ nano server.pem
```

The following command is run to start a simple web server using server.pem:

```
[02/28/21]seed@VM:~/.../lab4$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
Using default temp DH parameters
ACCEPT
```

As expected, we get an error message from Firefox telling us that "The owner of seedpkilab2020.com has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website."

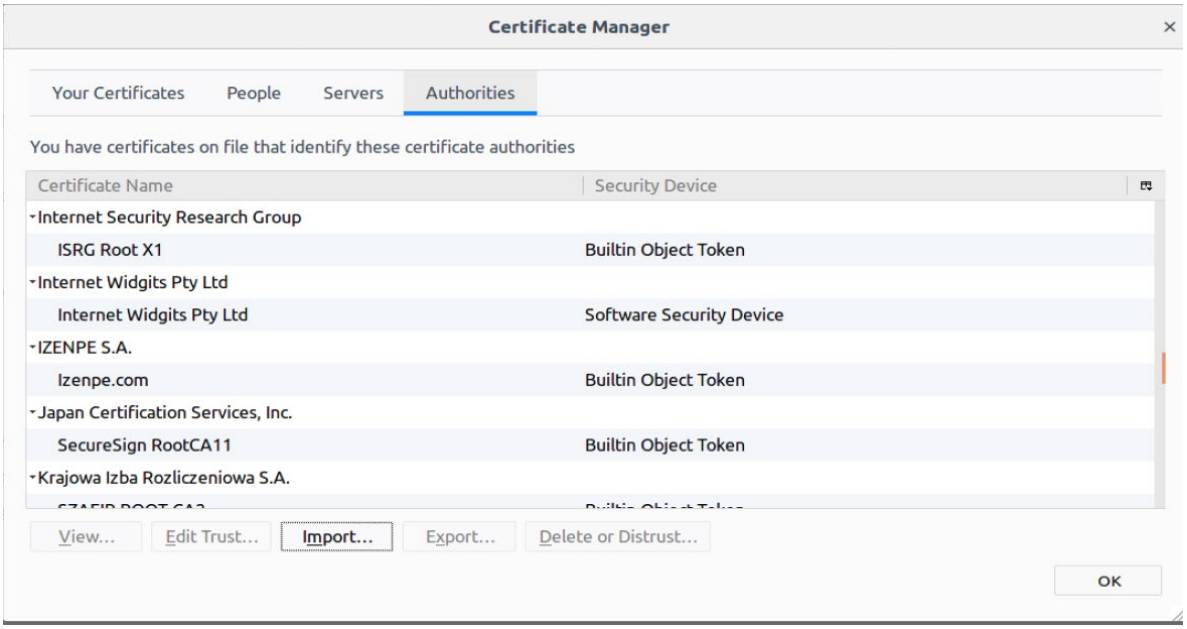


Name: Kwa Li Ying
Student ID: 1003833

Step 3: Getting the browser to accept our CA certificate

To get the Firefox browser to accept our CA certificate, we manually add this certificate to the Firefox browser:

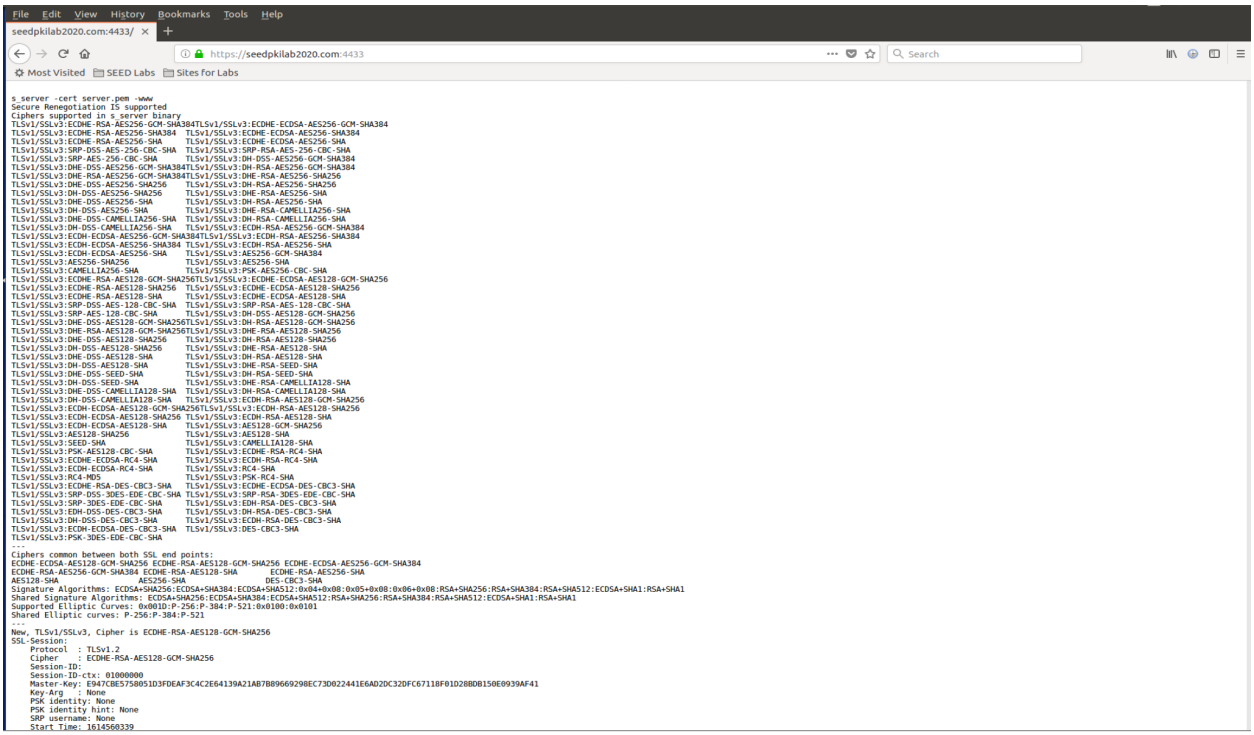
Our CA's certificate is now in Firefox's list of the accepted certificates:



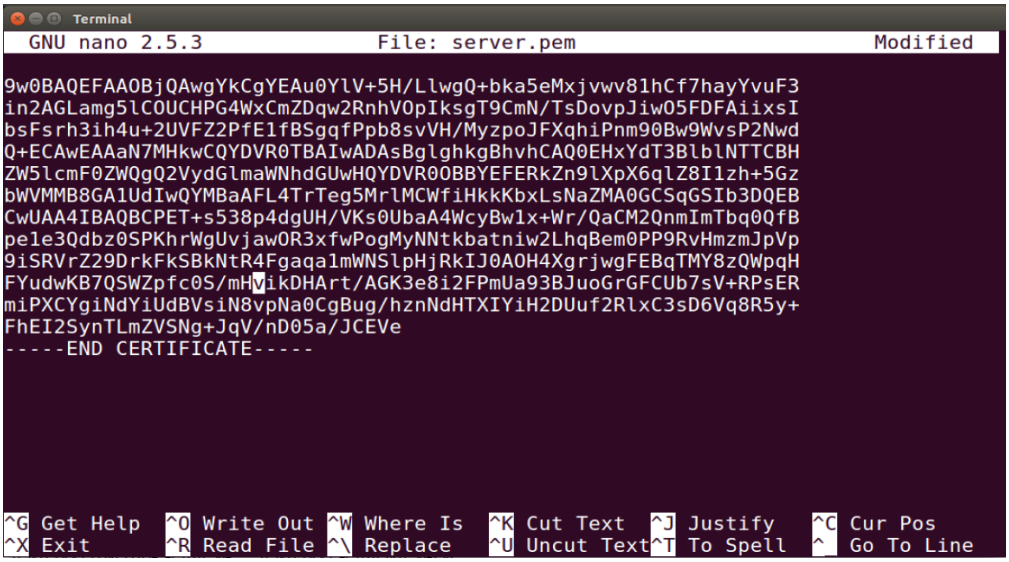
Name: Kwa Li Ying
Student ID: 1003833

Step 4: Testing our HTTPS website

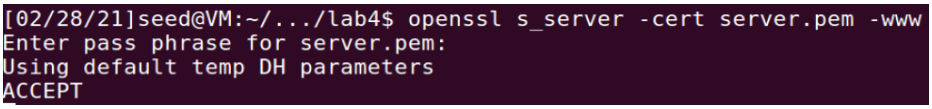
The webpage is refreshed and the page loads successfully:



A single byte of server.pem is modified as shown (G is changed to H):

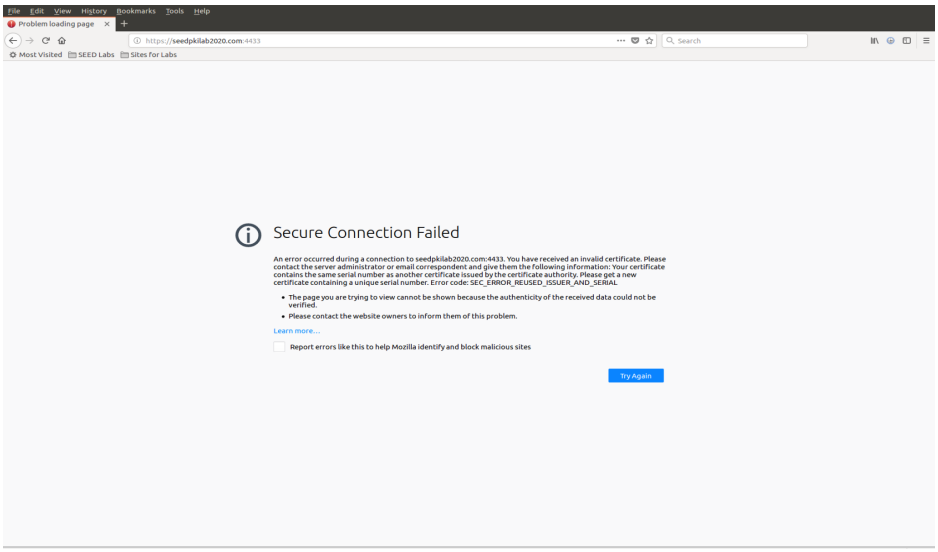


The server is restarted:



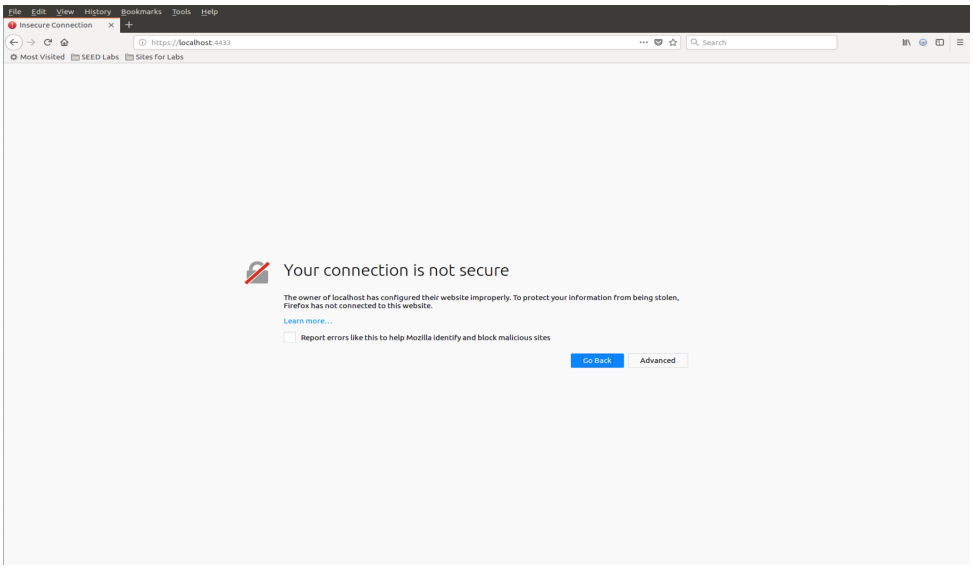
Name: Kwa Li Ying
Student ID: 1003833

The URL of the browser is reloaded and the webpage fails to load. The message given is “An error occurred during a connection to seedpkilab2020.com:4433. You have received an invalid certificate. Please contact the server administrator or email correspondent and give them the following information: Your certificate contains the same serial number as another certificate issued by the certificate authority. Please get a new certificate containing a unique serial number. Error code: SEC_ERROR_REUSED_ISSUER_AND_SERIAL”:



The original server.pem is restored afterwards.

We attempt to connect to SEEDPKILab2020 by using https://localhost:4433 as the URL. The webpage fails to load as we get an error message from Firefox telling us that “The owner of localhost has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.”



Name: Kwa Li Ying
Student ID: 1003833

Task 4: Deploying Certificate in an Apache-Based HTTPSWebsite

The Apache's configuration file default-ssl.conf is edited to include the VirtualHost entry as shown below:

```
<VirtualHost *:443>
        ServerName SEEDPKILab2020.com
        DocumentRoot /var/www/lab4
        DirectoryIndex index.html

        SSLEngine On
        SSLCertificateFile /home/seed/Documents/ns/lab4/server.crt
        SSLCertificateKeyFile /home/seed/Documents/ns/lab4/server.pem
</VirtualHost>
```

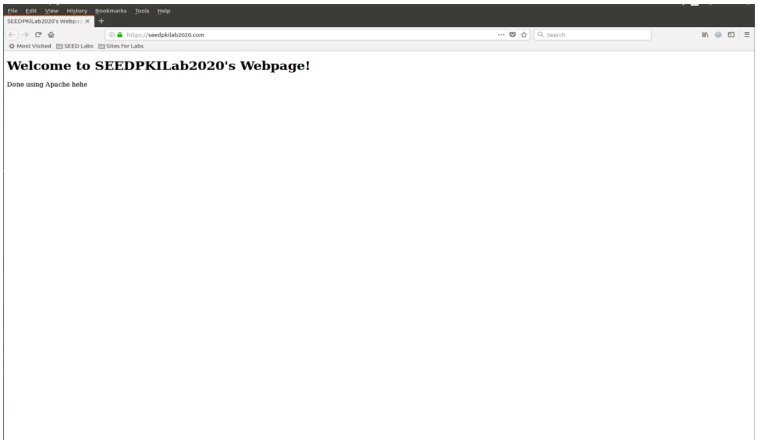
A simple index.html is done in the DocumentRoot folder specified:

```
<!DOCTYPE html>
<html>
  <head>
    <title>SEEDPKILab2020's Webpage</title>
  </head>
  <body>
    <h1>Welcome to SEEDPKILab2020's Webpage!</h1>
    <p>Done using Apache hehe</p>
  </body>
</html>
```

A series of commands is run to enable SSL:

```
[03/01/21]seed@VM:.../sites-available$ sudo apachectl configtest
AH00112: Warning: DocumentRoot [/var/www/seedlabclickjacking] does not exist
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive glob
ally to suppress this message
Syntax OK
[03/01/21]seed@VM:.../sites-available$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  service apache2 restart
[03/01/21]seed@VM:.../sites-available$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  service apache2 reload
[03/01/21]seed@VM:.../sites-available$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for SEEDPKILab2020.com:443 (RSA): ************
[03/01/21]seed@VM:.../sites-available$
```

We proceed to visit https://seedpkilab2020.com on Firefox:



Since the contents of the webpage matches the contents of index.html, we have successfully browsed the HTTPS site.

Name: Kwa Li Ying
Student ID: 1003833

Task 5: Launching a Man-In-The-Middle Attack

Network Addresses

Malicious Server: 10.0.2.128

Victim Machine: 10.0.0.129

Step 1: Setting up the malicious website

The VirtualHost entry set up in Task 4 is edited to have the ServerName changed to example.com:

```
<VirtualHost *:443>
        ServerName example.com
        DocumentRoot /var/www/lab4
        DirectoryIndex index.html

        SSLEngine On
        SSLCertificateFile /home/seed/Documents/ns/lab4/server.crt
        SSLCertificateKeyFile /home/seed/Documents/ns/lab4/server.pem
</VirtualHost>
```

The series of commands are run again:

```
[03/01/21]seed@VM:.../sites-available$ sudo apachectl configtest
AH00112: Warning: DocumentRoot [/var/www/seedlabclickjacking] does not exist
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive glob
ally to suppress this message
Syntax OK
[03/01/21]seed@VM:.../sites-available$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
[03/01/21]seed@VM:.../sites-available$ sudo a2ensite default-ssl
Site default-ssl already enabled
[03/01/21]seed@VM:.../sites-available$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for example.com:443 (RSA): ************
[03/01/21]seed@VM:.../sites-available$
```

Step 2: Becoming the man in the middle

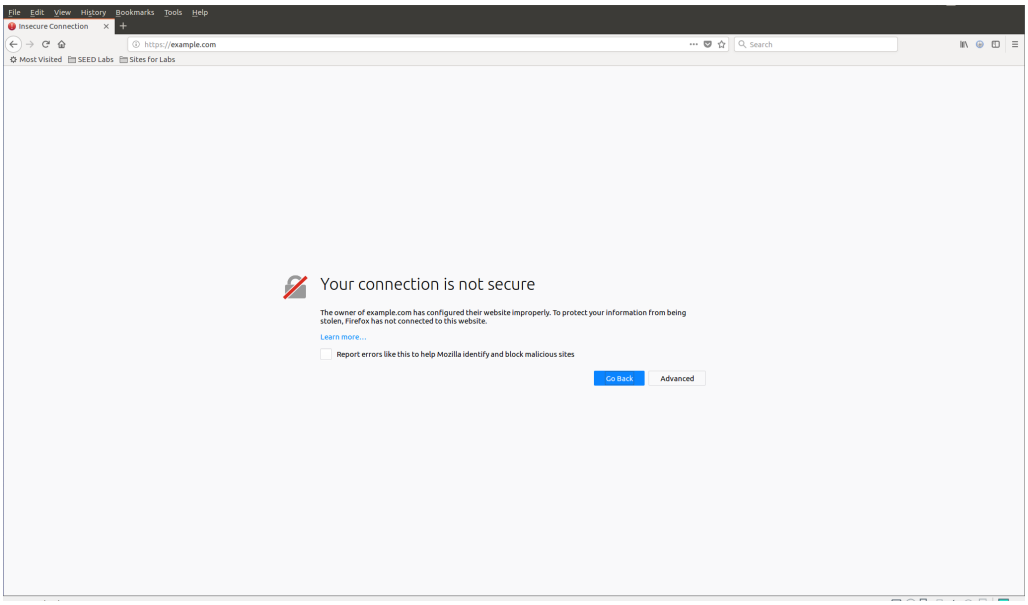
The victim machine (10.0.2.129)'s /etc/hosts file is edited to have the following entry:

```
10.0.2.128      example.com
```

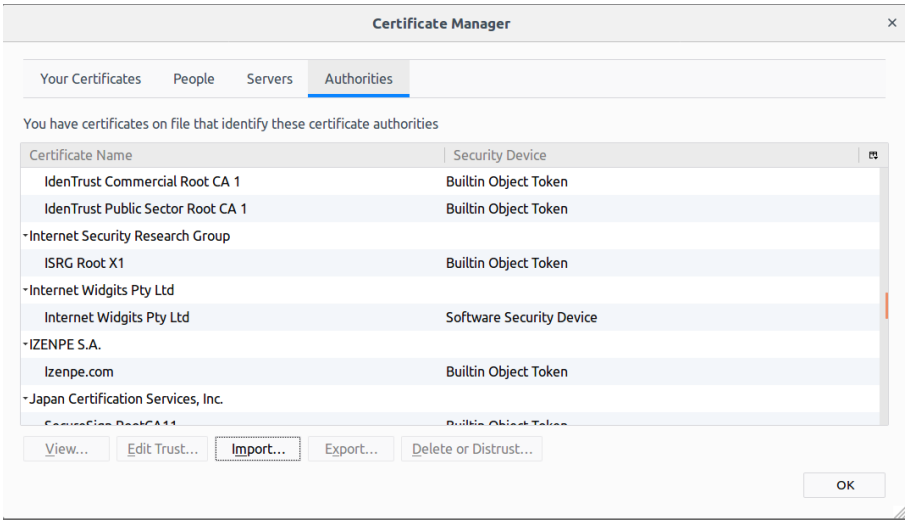
Name: Kwa Li Ying
Student ID: 1003833

Step 3: Browse the target website

On the victim machine, we proceed to visit https://example.com on Firefox:

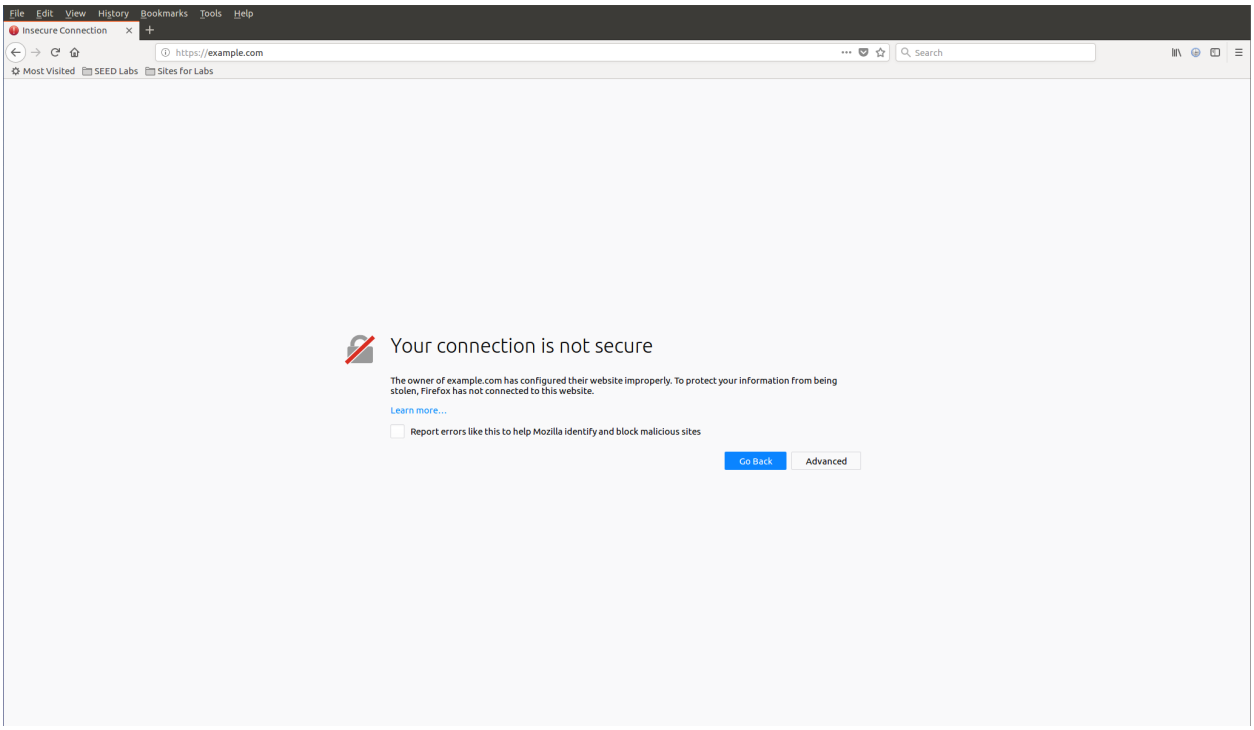


This is due to the fact that we have not added our root CA's certificate to the list of recognised CA's in the victim machine's browser. We copy ca.crt into this victim machine and add it to the list of recognised CAs:



The changes are saved and the webpage is refreshed but the victim machine's browser still shows the same message:

Name: Kwa Li Ying
Student ID: 1003833



This happens because we are visiting the website with the domain name example.com but the server's certificate and key-pair were created using the SEEDPKILab2020.com domain name. This mismatch in the certificate's subject (common name) and the hostname of the server causes this to happen.

Name: Kwa Li Ying
Student ID: 1003833

Task 6: Launching a Man-In-The-Middle Attack with a Compromised CA

In task 5, the attack fails because the same certificate for SEEDPKILab2020.com is used for example.com. For the MITM attack to succeed, the attacker has to generate a valid certificate with the common name specified as the actual domain name he would like to impose as. For this experiment, we shall use youtube.com.

The steps to create the certificate for the fake website using the root CA's private key is repeated, as in Task 2.

First, the following command is run to create the company's public/private key pair in task6server.key. The password *task6company* is used:

```
[03/01/21]seed@VM:~/.../lab4$ openssl genrsa -aes128 -out task6server.key 1024
Generating RSA private key, 1024 bit long modulus
..............++++++
....++++++
e is 65537 (0x10001)
Enter pass phrase for task6server.key:
Verifying - Enter pass phrase for task6server.key:
```

The following command is run to generate a certificate signing request for the company. The CSR is saved as task6server.csr. The Organization Name and the Common Name are both specified as www.youtube.com. The rest of the fields are left as default:

```
[03/01/21]seed@VM:~/.../lab4$ openssl req -new -key task6server.key -out task6se
rver.csr -config openssl.cnf
Enter pass phrase for task6server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:www.youtube.com
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:www.youtube.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

After this, the following command is run to turn the CSR into an X509 certificate task6server.crt using the CA's certificate and key:

```
[03/01/21]seed@VM:~/.../lab4$ openssl ca -in task6server.csr -out task6server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4098 (0x1002)
        Validity
            Not Before: Mar  1 23:27:47 2021 GMT
            Not After : Mar  1 23:27:47 2022 GMT
        Subject:
            countryName               = AU
            stateOrProvinceName       = Some-State
            organizationName          = www.youtube.com
            commonName                = www.youtube.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                D5:D1:05:F5:B1:14:8F:15:BC:32:79:30:BF:7C:50:3E:B5:BA:2B:F4
            X509v3 Authority Key Identifier:
                keyid:BE:13:AD:37:A0:E4:CA:E5:30:25:9F:88:79:24:29:BC:4B:B0:D6:99

Certificate is to be certified until Mar  1 23:27:47 2022 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Name: Kwa Li Ying
Student ID: 1003833

The key and certificate is then combined into one file called task6server.pem:

```
[03/01/21]seed@VM:~/.../lab4$ cp task6server.key task6server.pem
[03/01/21]seed@VM:~/.../lab4$ cat task6server.crt >> task6server.pem
```

Now that we have created the key and the certificate for the fake www.youtube.com domain, we proceed to set up the Apache server to host this fake webpage.

The VirtualHost entry in the /etc/apache2/sites-available/default-ssl.conf configuration file is edited as shown:

```
<VirtualHost *:443>
        ServerName www.youtube.com
        DocumentRoot /var/www/lab4
        DirectoryIndex index.html

        SSLEngine On
        SSLCertificateFile /home/seed/Documents/ns/lab4/task6server.crt
        SSLCertificateKeyFile /home/seed/Documents/ns/lab4/task6server.pem
</VirtualHost>
```

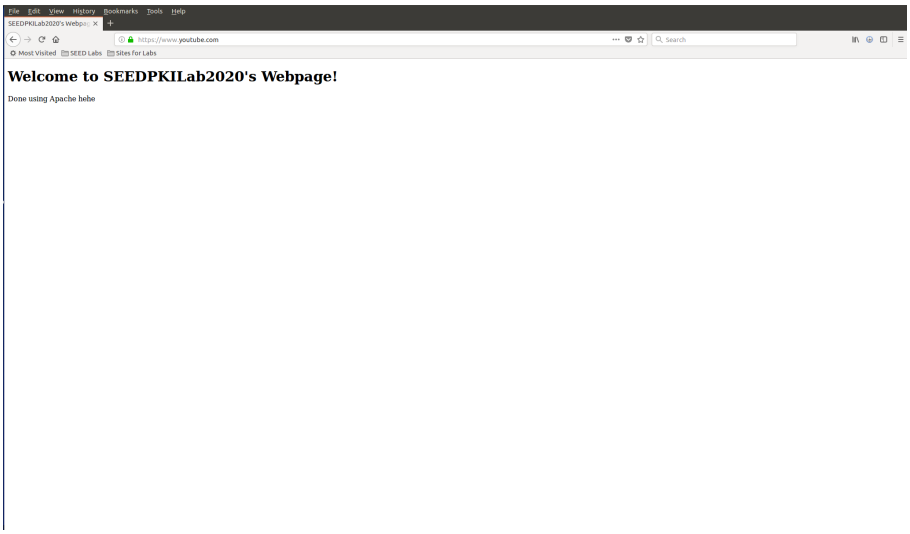
The following commands are run for the Apache server to restart with the changes applied:

```
[03/01/21]seed@VM:~/.../lab4$ sudo apachectl configtest
AH00112: Warning: DocumentRoot [/var/www/seedlabclickjacking] does not exist
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive glob
ally to suppress this message
Syntax OK
[03/01/21]seed@VM:~/.../lab4$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
[03/01/21]seed@VM:~/.../lab4$ sudo a2ensite default-ssl
Site default-ssl already enabled
[03/01/21]seed@VM:~/.../lab4$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for www.youtube.com:443 (RSA): ***********
```

Now that the server is up and running, we edit the /etc/hosts file on the victim machine to simulate a DNS poisoning attack:

```
10.0.2.128      www.youtube.com
```

Finally, we attempt to visit https://www.youtube.com on the victim machine:



The MITM attack is successful! ☺