CSE 644 Internet Security Lab-10 (Public Key Infrastructure)

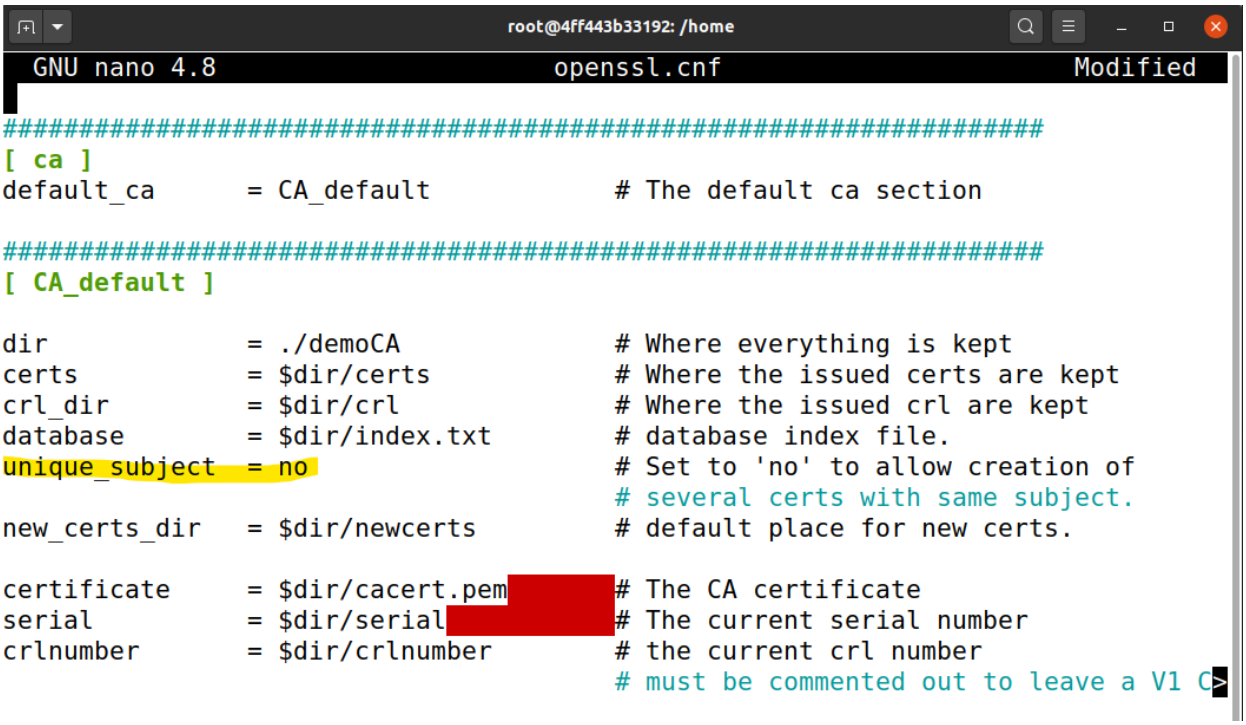
Sneden Rebello

Task 1 : Becoming a Certificate Authority (CA)

We copy the openssl config file actually located in /usr/lib/ssl/openssl.cnf to start the lab.

```
root@4ff443b33192:/# cp /usr/lib/ssl/openssl.cnf /home/
root@4ff443b33192:/# cd /
root@4ff443b33192:/# cd home/
root@4ff443b33192:/home# ls
openssl.cnf
root@4ff443b33192:/home#
```

I uncomment the unique subject line to allow creation of certifications with the same subject.



I create the required files to satisfy the CA_default setting requirement.

```
root@4ff443b33192: /home
root@4ff443b33192:/lib/ssl# cd /home/
root@4ff443b33192:/home# ls
openssl.cnf
root@4ff443b33192:/home# mkdir ./demoCA
root@4ff443b33192:/home# cd ./demoCA
root@4ff443b33192:/home/demoCA# mkdir certs
root@4ff443b33192:/home/demoCA# mkdir crl
root@4ff443b33192:/home/demoCA# mkdir newcerts
root@4ff443b33192:/home/demoCA# touch index.txt
root@4ff443b33192:/home/demoCA# echo "1000" > serial
```

We need to generate a self-signed certificate for our CA. This means that this CA is totally trusted, and its certificate will serve as the root certificate. Running the command below :

```
root@4ff443b33192: /home
root@4ff443b33192:/home# openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 \
> -keyout ca.key -out ca.crt
Generating a 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]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:Syracuse
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Sneden
Organizational Unit Name (eg, section) []:Sneden
Common Name (e.g. server FQDN or YOUR name) []:Sneden
Email Address []:Sneden@gmail.com
root@4ff443b33192:/home#
```

This gives us the CA.crt (public key) and CA.key (private key)

```
root@4ff443b33192: /home
root@4ff443b33192:/home# ls
ca.crt  ca.key  demoCA  openssl.cnf
root@4ff443b33192:/home#
```

We run the following commands to look at the decoded content of the X509 certificate and the RSA key (-text means decoding the content into plain text; -noout means not printing out the encoded version):

openssl x509 -in ca.crt -text -noout

openssl rsa -in ca.key -text -noout

```
root@4ff443b33192:/home# openssl x509 -in ca.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            34:6d:58:a5:8c:ed:93:0c:45:e6:6f:1f:5c:9f:05:0a:fc:a8:96:f1
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = New York, L = Syracuse, O = Sneden, OU = Sneden, CN
 = Sneden, emailAddress = Sneden@gmail.com
        Validity
            Not Before: May  3 02:37:03 2022 GMT
            Not After : Apr 30 02:37:03 2032 GMT
        Subject: C = US, ST = New York, L = Syracuse, O = Sneden, OU = Sneden, C
N = Sneden, emailAddress = Sneden@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:bf:cd:b5:74:c8:66:87:68:c2:a5:f1:eb:30:54:
                    87:15:21:04:3b:de:e2:20:5e:33:f0:a8:bf:c5:6b:
                    2c:a4:c1:c6:05:7a:ae:44:0a:e3:3b:c7:80:a6:98:
                    f3:df:68:b2:e0:1d:4d:e6:fb:fe:27:e4:47:aa:0d:
```

This shows us all the information of the CA.

Below, shows us the RSA keys and related data like primes and exponents.

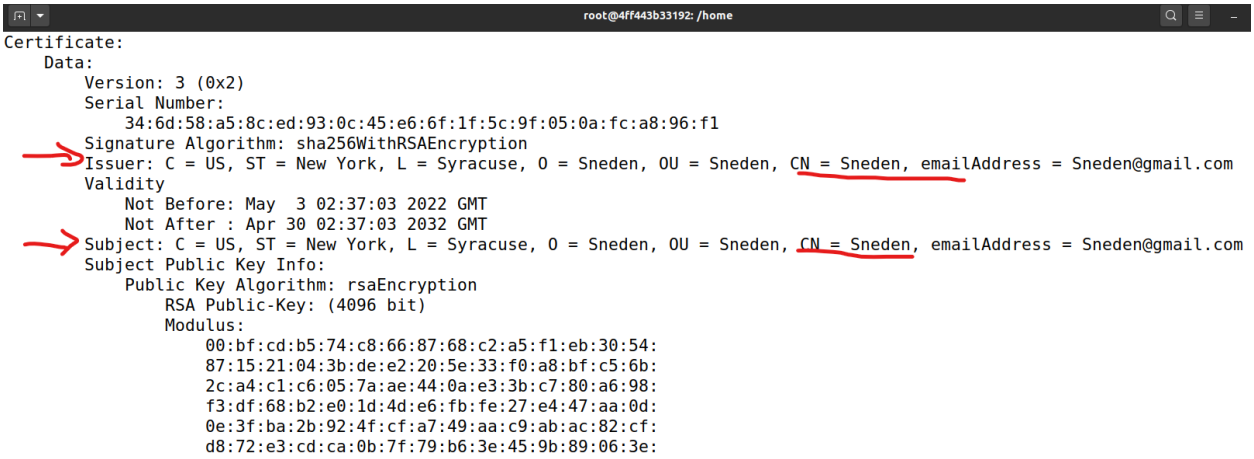
```
root@4ff443b33192:/home# openssl rsa -in ca.key -text -noout
Enter pass phrase for ca.key:
RSA Private-Key: (4096 bit, 2 primes)
modulus:
    00:bf:cd:b5:74:c8:66:87:68:c2:a5:f1:eb:30:54:
    87:15:21:04:3b:de:e2:20:5e:33:f0:a8:bf:c5:6b:
    2c:a4:c1:c6:05:7a:ae:44:0a:e3:3b:c7:80:a6:98:
    f3:df:68:b2:e0:1d:4d:e6:fb:fe:27:e4:47:aa:0d:
    0e:3f:ba:2b:92:4f:cf:a7:49:aa:c9:ab:ac:82:cf:
    d8:72:e3:cd:ca:0b:7f:79:b6:3e:45:9b:89:06:3e:
    b4:f0:d0:81:7a:15:37:be:0f:c1:16:f5:50:b6:ec:
    38:a6:25:78:1a:de:c6:1d:5a:f5:86:98:d4:47:2c:
    b4:17:39:ba:d0:74:1d:ba:9d:eb:4a:8b:50:7e:2f:
    b7:96:1d:5d:59:85:73:86:14:22:74:c2:8f:cb:ad:
    e2:ef:46:6b:ff:63:0d:88:e2:24:eb:0f:53:91:18:
    bf:5c:13:a1:7c:4a:c3:12:bf:30:81:23:94:31:c2:
    08:b6:06:a0:f5:c6:e0:b1:eb:4e:50:c1:25:31:c5:
    e6:20:00:a4:9d:89:4c:ca:26:0b:25:4b:03:43:b4:
    7f:06:40:91:87:bf:a7:e6:f0:50:88:1f:c6:43:a2:
    ec:2a:57:ab:fc:13:6d:47:dc:db:63:2f:5a:12:38:
    9c:ea:d0:ca:50:00:09:44:fa:81:69:64:c8:f4:6b:
    e2:d4:ae:e0:42:f3:03:87:f2:d6:34:3f:16:31:39:
```

• What part of the certificate indicates this is a CA's certificate?

```
            c6:dc:19:06:a2:73:49:a5:a4:dc:57:9a:58:9a:6b:
            b8:83:15
        Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Subject Key Identifier:
        D5:4E:7D:AB:85:3E:70:5D:D8:88:F1:A9:3A:30:95:87:98:02:83:9C
    X509v3 Authority Key Identifier:
        keyid:D5:4E:7D:AB:85:3E:70:5D:D8:88:F1:A9:3A:30:95:87:98:02:83:9C

    X509v3 Basic Constraints: critical
        CA:TRUE
```

We see that 'CA : true' plays an important role in knowing that it is an authentic CA or no.

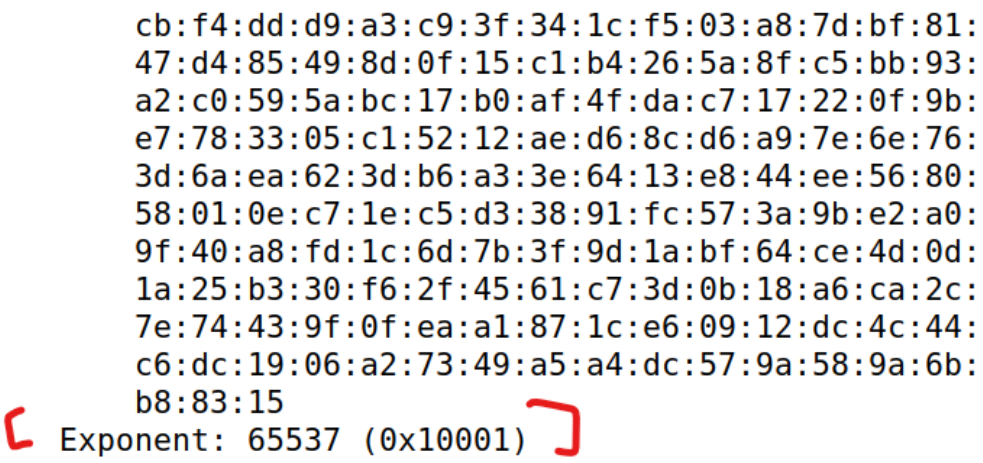
• What part of the certificate indicates this is a self-signed certificate?



As we see that the issuer CN and subject CN are replicas, we know that it is self-signed.

• In the RSA algorithm, we have a public exponent e, a private exponent d, a modulus n, and two secret numbers p and q, such that n = pq. Please identify the values for these elements in your certificate and key files

e :



n :

```
        Not Before: May  3 02:37:03 2022 GMT
        Not After : Apr 30 02:37:03 2032 GMT
    Subject: C = US, ST = New York, L = Syracuse, O = Sneden, OU = Sneden, CN = Sneden, emailAddress = Sneden@gmail.com
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            RSA Public-Key: (4096 bit)
            Modulus:
                00:bf:cd:b5:74:c8:66:87:68:c2:a5:f1:eb:30:54:
                87:15:21:04:3b:de:e2:20:5e:33:f0:a8:bf:c5:6b:
                2c:a4:c1:c6:05:7a:ae:44:0a:e3:3b:c7:80:a6:98:
                f3:df:68:b2:e0:1d:4d:e6:fb:fe:27:e4:47:aa:0d:
                0e:3f:ba:2b:92:4f:cf:a7:49:aa:c9:ab:ac:82:cf:
                d8:72:e3:cd:ca:0b:7f:79:b6:3e:45:9b:89:06:3e:
                b4:f0:d0:81:7a:15:37:be:0f:c1:16:f5:50:b6:ec:
                38:a6:25:78:1a:de:c6:1d:5a:f5:86:98:d4:47:2c:
                b4:17:39:ba:d0:74:1d:ba:9d:eb:4a:8b:50:7e:2f:
                b7:96:1d:5d:59:85:73:86:14:22:74:c2:8f:cb:ad:
                e2:ef:46:6b:ff:63:0d:88:e2:24:eb:0f:53:91:18:
                bf:5c:13:a1:7c:4a:c3:12:bf:30:81:23:94:31:c2:
                08:b6:06:a0:f5:c6:e0:b1:eb:4e:50:c1:25:31:c5:
                e6:20:00:a4:9d:89:4c:ca:26:0b:25:4b:03:43:b4:
                7f:06:40:91:87:bf:a7:e6:f0:50:88:1f:c6:43:a2:
                ec:2a:57:ab:fc:13:6d:47:dc:db:63:2f:5a:12:38:
                9c:ea:d0:ca:50:00:09:44:fa:81:69:64:c8:f4:6b:
                e2:d4:ae:e0:42:f3:03:87:f2:d6:34:3f:16:31:39:
                2e:5e:a3:c2:ca:7f:f5:75:dc:e7:d8:6e:d6:9e:db:
                f4:88:c9:84:74:60:38:0a:65:b9:b9:62:d2:81:a9:
                35:6c:1f:03:0f:58:53:b1:94:bc:e9:34:80:69:1e:
                93:d0:9d:4b:29:ea:8d:83:dd:a0:de:ea:d0:ee:80:
                b1:c8:5d:d3:3d:33:57:b6:41:98:2a:b3:93:81:78:
                b4:15:66:02:36:89:08:96:c3:c5:f0:0b:be:6b:18:
                cb:f4:dd:d9:a3:c9:3f:34:1c:f5:03:a8:7d:bf:81:
                47:d4:85:49:8d:0f:15:c1:b4:26:5a:8f:c5:bb:93:
                a2:c0:59:5a:bc:17:b0:af:4f:da:c7:17:22:0f:9b:
                e7:78:33:05:c1:52:12:ae:d6:8c:d6:a9:7e:6e:76:
```

d :

```
publicExponent: 65537 (0x10001)
privateExponent:
    7d:1a:35:72:b8:8b:77:62:b3:22:fd:c9:c3:3a:3e:
    e9:5f:21:9d:d3:60:76:70:3b:3c:8b:34:9a:15:af:
    86:bf:04:e3:ea:02:e4:4a:9d:b0:0e:0e:31:9b:ad:
    e5:58:2a:e3:d6:f9:4a:e1:ae:02:62:f9:03:47:84:
    b5:b8:3e:57:2e:4b:68:f7:b9:b7:d4:8b:ae:be:d6:
    95:09:54:de:a5:e5:3d:83:ca:d6:27:fe:95:de:2c:
    b2:ad:ac:e5:ee:14:ef:98:e2:fe:90:7f:56:f0:78:
    7a:96:11:e2:a4:cf:5f:b2:46:56:c6:34:1f:40:5f:
    2e:8e:ee:f4:ab:e8:00:22:a4:a7:78:7a:c5:f3:65:
    b1:39:f1:fb:43:f9:f7:c8:06:39:55:f5:e8:89:c0:
    6d:04:e7:c8:29:d9:58:20:f8:a7:80:d5:a6:dc:17:
    64:7d:41:65:0a:76:76:31:d1:f7:49:37:8d:c4:db:
    5d:ae:b1:29:dd:d0:0c:fc:47:96:90:84:2c:14:c2:
    66:b9:44:a5:a4:3a:3b:d7:d8:89:38:86:26:91:02:
    11:17:e8:54:3c:83:cd:33:8d:b3:5b:d6:3b:a3:76:
    9a:9f:98:83:d0:40:05:11:5e:df:30:b1:75:15:0b:
    f4:74:be:a0:b3:85:58:42:74:4a:43:fc:76:c0:2f:
    b0:19:cc:24:94:0f:e1:c4:c6:c7:15:36:57:54:96:
    a7:b6:da:f1:76:48:bc:89:9e:d4:f7:8d:b8:cc:a8:
    68:5d:22:22:bb:33:7f:56:3c:41:c6:d1:14:42:97:
```

p :

```
    fe:01
prime1:
    00:fd:02:c7:96:c1:30:36:5f:ee:e3:40:8b:ed:68:
    57:2f:e9:e5:7a:61:de:fd:74:90:51:29:f4:78:57:
    75:f8:23:f9:a4:14:30:63:bb:80:77:c4:a2:00:57:
    14:28:a8:e6:9c:57:81:da:e7:a1:64:4d:9b:fb:67:
    35:cd:89:01:e0:82:f8:30:12:12:8b:c6:5f:30:49:
    8b:8a:22:42:e2:6f:c2:29:c5:20:91:d4:ee:0f:c0:
    00:ca:c0:08:54:ad:13:43:5c:2e:d3:c7:6a:22:8d:
    cd:b1:b6:7c:4d:21:02:44:ab:12:cf:3a:9c:3d:7c:
    11:62:62:56:db:35:6d:96:5a:fd:2f:34:01:55:96:
    2f:95:f0:68:af:f3:d8:aa:bf:69:2c:a9:96:97:af:
    28:74:70:1c:07:21:d4:ea:11:7b:3e:73:0e:eb:04:
    84:72:be:49:30:57:af:f5:b8:0e:4a:87:fd:be:e0:
    5b:22:34:7a:8b:44:55:27:f5:66:03:08:c8:30:e2:
    90:86:b2:53:b5:0d:fb:55:a1:32:f3:d7:45:31:55:
    ad:05:9c:48:d0:da:8c:e3:f3:d1:7e:26:0c:3f:12:
    6c:e0:57:d2:ef:bc:87:e5:86:aa:e3:7d:d6:eb:1b:
    d9:2c:3f:2d:05:3e:2a:12:b6:be:75:8f:38:2f:a0:
    2f:c1
```

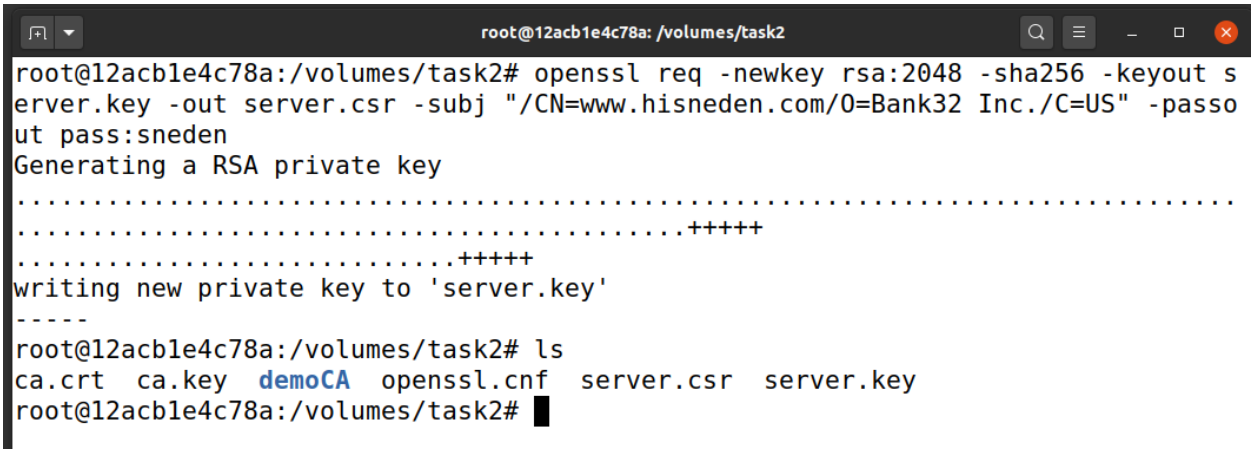
q :

```
prime2:
    00:c2:11:cf:71:5c:74:86:d5:9d:4f:ae:3c:57:70:
    a0:47:80:72:fc:88:3a:c3:11:1e:e5:f3:30:36:c9:
    22:2a:02:52:c4:eb:19:0f:66:08:82:7e:d7:56:4b:
    f5:98:ed:b5:41:08:3c:b2:ae:dd:43:a9:d7:0b:ee:
    9a:55:e4:17:c3:63:80:69:bc:1c:7a:d6:65:f0:9b:
    cf:c7:e9:96:f8:61:68:9c:42:e9:a5:08:6e:90:c6:
    f2:b6:46:cb:82:3d:ce:e3:3a:f3:36:09:35:a2:e9:
    25:e1:de:d6:13:de:04:e5:68:98:54:5c:a8:60:41:
    e2:df:fc:9f:9d:92:77:a0:11:f3:7a:22:3d:02:64:
    0a:30:e9:7b:e7:a8:92:74:82:76:18:af:db:68:7e:
    4f:1b:ce:17:ec:cd:44:63:a1:41:27:e6:90:f3:a6:
    ed:33:8d:9c:37:e8:60:1d:ab:21:94:6a:4a:37:80:
    a1:cc:60:1f:51:7f:c6:9d:3a:88:69:5d:28:69:ac:
    e9:49:70:21:03:91:a7:e0:02:71:bd:50:50:18:b2:
    b5:e2:68:84:c1:6b:0f:5c:fb:e1:d9:91:1e:16:26:
    2e:10:91:3e:14:50:ec:7a:a3:3c:98:42:f6:7d:d2:
    7b:f3:ba:fd:74:c2:e9:42:93:66:ca:b9:62:48:b2:
    a8:55
```

Task 2 : Generating a Certificate Request for Your Web Server

For this task and onwards, I create my own webserver by name : www.hisneden.com

Now I generate a Certificate Signing Request (CSR), which basically includes the company's public key and identity information. I receive the

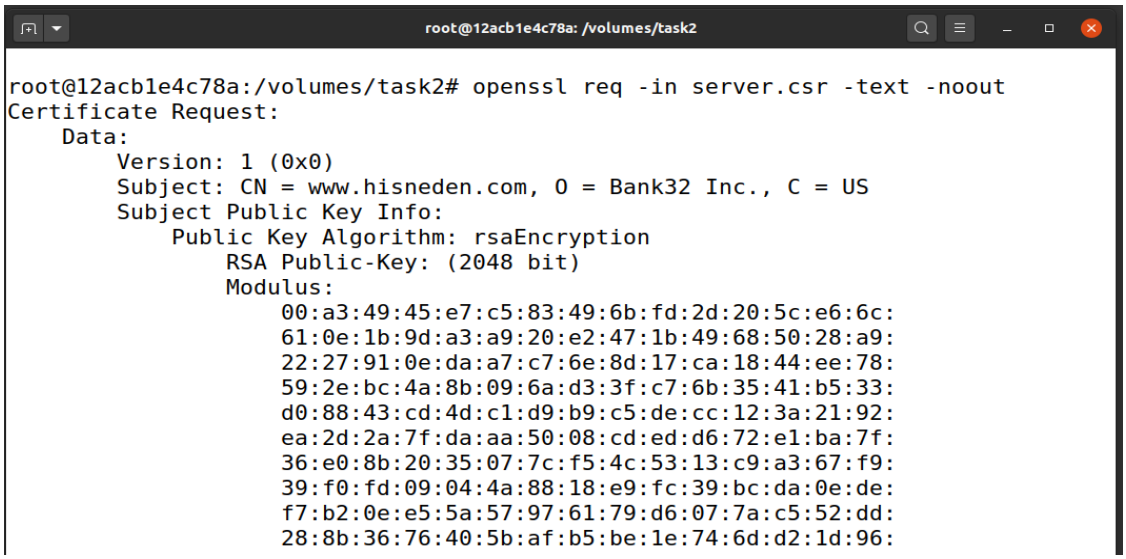


This generates two keys, a public and private which is then used to create the csr.

I use the following command to look at the decoded content of the CSR and private key files:

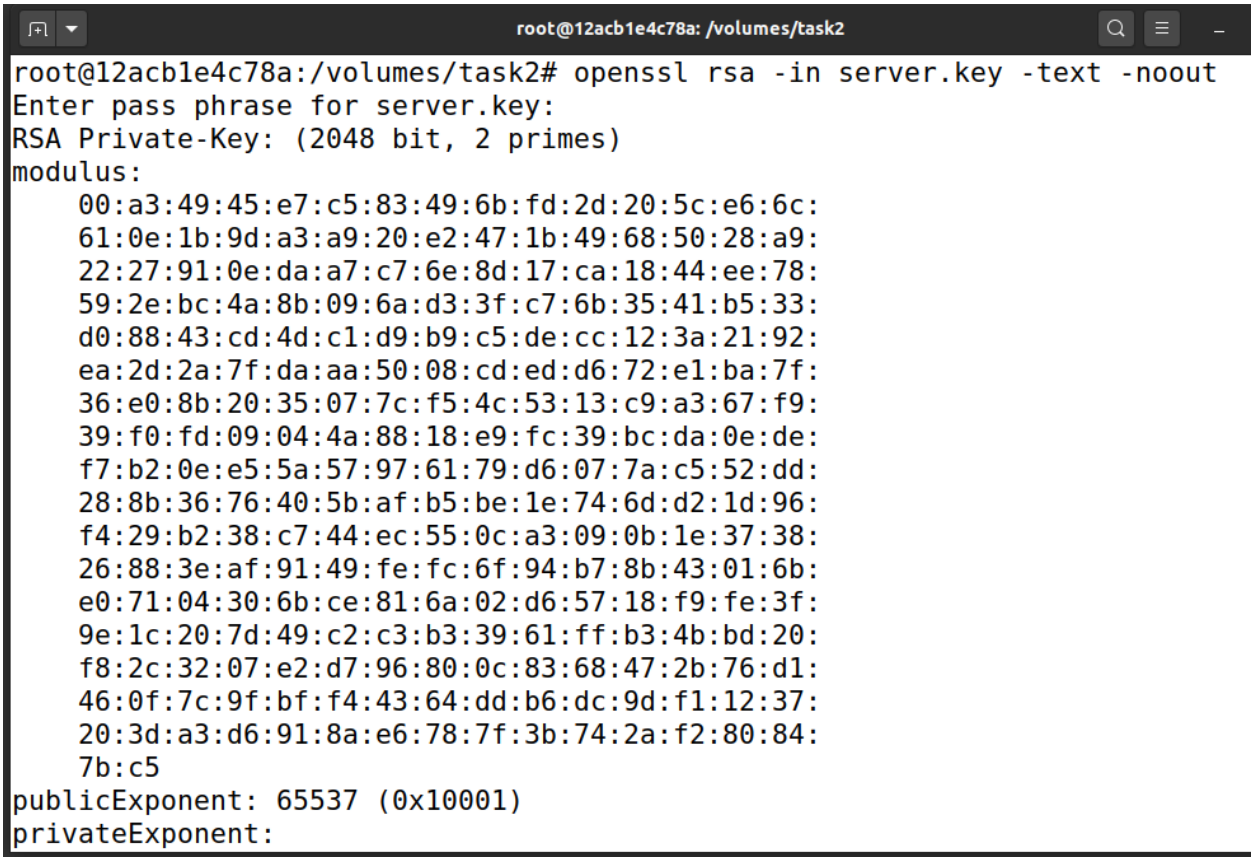
openssl req -in server.csr -text -noout

openssl rsa -in server.key -text -noout



This shows the information of the server.csr.

Below shows the information about server.key.



Many websites have different URLs. For example, www.example.com, example.com, example.net, and example.org are all pointing to the same web server. Due to the hostname matching policy enforced by browsers, the common name in a certificate must match with the server's hostname, or browsers will refuse to communicate with the server.

To allow a certificate to have multiple names, the X.509 specification defines extensions to be attached to a certificate called Subject Alternative Name (SAN).

```
root@12acb1e4c78a: /volumes/task2
root@12acb1e4c78a:/volumes/task2# openssl req -newkey rsa:2048 -sha256 -keyout s
erver.key -out server.csr -subj "/CN=www.hisneden.com/O=sneden Inc./C=US" -passo
ut pass:sneden -addext "subjectAltName = DNS:www.hisneden.com, \
DNS:www.hisnedenA.com, \
DNS:www.hisnedenB.com"
Generating a RSA private key
.....................+++++
.................+++++
writing new private key to 'server.key'
-----
root@12acb1e4c78a:/volumes/task2# ls
ca.crt  ca.key  demoCA  openssl.cnf  server.csr  server.key
root@12acb1e4c78a:/volumes/task2#
```

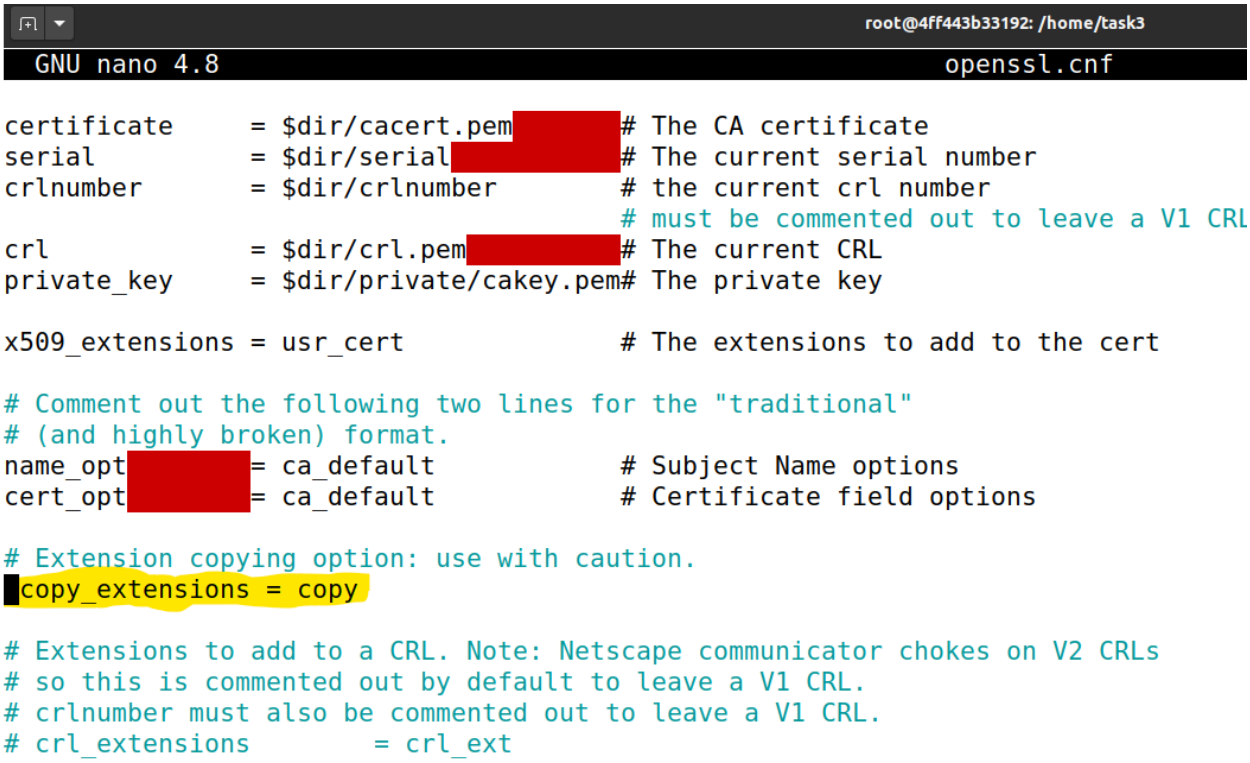
Below shows the SAN details in the server.csr file.

```
root@12acb1e4c78a: /volumes/task2
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: CN = www.hisneden.com, O = sneden Inc., C = US
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:de:27:b0:7a:4f:ee:1c:e6:f2:fe:c0:33:0d:97:
                    d7:2f:06:f7:b7:8a:70:80:75:05:4e:1d:94:0a:ee:
                    77:97:fa:5b:7a:be:17:d3:c4:48:78:8c:25:4b:e1:
                    a4:ad:f6:59:76:dc:f1:1d:76:30:2e:9d:a4:1e:d6:
                    eb:57:ed:70:21:d3:ac:ae:72:59:ad:5c:59:b8:5b:
                    67:2a:ef:f4:87:23:a0:8d:11:0a:03:1d:13:df:20:
                    26:0f:26:46:6f:e7:98:74:66:1c:1b:82:d4:7e:73:
                    3d:5e:6c:b6:3b:f3:6c:ac:04:53:5a:2d:2b:63:e7:
                    28:0c:a9:f0:af:bb:f6:d8:86:a2:df:2d:03:c3:90:
                    28:8a:b9:cb:62:27:ad:e8:95:7e:07:55:1c:7b:7e:
                    18:b9:8c:03:55:51:47:6a:e1:71:e4:a6:2b:17:1e:
                    48:9e:af:66:3f:57:da:a5:b4:ad:b5:16:c1:ac:6c:
                    82:f4:b8:38:35:12:eb:8c:f8:89:a3:6a:08:ed:2b:
                    d1:bc:ce:7b:dc:2f:ef:41:2b:f0:a2:3d:60:ab:27:
                    10:f4:ad:51:82:ff:9b:32:cc:d5:76:6b:8f:53:e8:
                    49:e2:95:7a:d2:e1:b3:57:f1:fd:53:5a:87:b9:03:
                    a7:13:e9:5a:34:43:f6:cf:ef:d7:01:9a:bd:ea:0a:
                    cd:4b
                Exponent: 65537 (0x10001)
        Attributes:
        Requested Extensions:
            X509v3 Subject Alternative Name:
                DNS:www.hisneden.com, DNS:www.hisnedenA.com, DNS:www.hisnedenB.c
om
    Signature Algorithm: sha256WithRSAEncryption
```

Task 3 : Generating a Certificate for your server

The default setting in openssl.cnf does not allow the "openssl ca" command to copy the extension field from the request to the final certificate.

To enable that, we can go to our copy of the configuration file, uncomment the following line: ccopy_extensions = copy



The following command turns the certificate signing request (server.csr) into an X509 certificate (server.crt), using the CA's ca.crt and ca.key:

```
root@12acb1e4c78a:/volumes/task3# openssl ca -config openssl.cnf -policy policy_
anything -md sha256 -days 3650 -in server.csr -out server.crt -batch -cert ca.cr
t -keyfile ca.key
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: May  3 22:21:34 2022 GMT
            Not After : Apr 30 22:21:34 2032 GMT
        Subject:
            countryName               = US
            organizationName          = sneden Inc.
            commonName                = www.hisneden.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                CE:E4:31:B2:E0:72:61:92:D2:ED:57:E5:B7:A6:34:EA:39:3F:B2:BD
            X509v3 Authority Key Identifier:
                keyid:D5:4E:7D:AB:85:3E:70:5D:D8:88:F1:A9:3A:30:95:87:98:02:83:9
C

            X509v3 Subject Alternative Name:
                DNS:www.hisneden.com, DNS:www.hisnedenA.com, DNS:www.hisnedenB.c
om
Certificate is to be certified until Apr 30 22:21:34 2032 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
root@12acb1e4c78a:/volumes/task3# ls
ca.crt  ca.key  demoCA  openssl.cnf  server.crt  server.csr  server.key
root@12acb1e4c78a:/volumes/task3#
```

Above we see the server.crt is now generated successfully.

We can view the server.crt details via the command below.

```
root@12acb1e4c78a:/volumes/task3# openssl x509 -in server.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4097 (0x1001)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = New York, L = Syracuse, O = Sneden, OU = Sneden, CN
 = Sneden, emailAddress = Sneden@gmail.com
        Validity
            Not Before: May  3 22:21:34 2022 GMT
            Not After : Apr 30 22:21:34 2032 GMT
        Subject: C = US, O = sneden Inc., CN = www.hisneden.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:de:27:b0:7a:4f:ee:1c:e6:f2:fe:c0:33:0d:97:
                    d7:2f:06:f7:b7:8a:70:80:75:05:4e:1d:94:0a:ee:
                    77:97:fa:5b:7a:be:17:d3:c4:48:78:8c:25:4b:e1:
                    a4:ad:f6:59:76:dc:f1:1d:76:30:2e:9d:a4:1e:d6:
                    eb:57:ed:70:21:d3:ac:ae:72:59:ad:5c:59:b8:5b:
                    67:2a:ef:f4:87:23:a0:8d:11:0a:03:1d:13:df:20:
                    26:0f:26:46:6f:e7:98:74:66:1c:1b:82:d4:7e:73:
                    3d:5e:6c:b6:3b:f3:6c:ac:04:53:5a:2d:2b:63:e7:
                    28:0c:a9:f0:af:bb:f6:d8:86:a2:df:2d:03:c3:90:
                    28:8a:b9:cb:62:27:ad:e8:95:7e:07:55:1c:7b:7e:
                    18:b9:8c:03:55:51:47:6a:e1:71:e4:a6:2b:17:1e:
                    48:9e:af:66:3f:57:da:a5:b4:ad:b5:16:c1:ac:6c:
                    82:f4:b8:38:35:12:eb:8c:f8:89:a3:6a:08:ed:2b:
                    d1:bc:ce:7b:dc:2f:ef:41:2b:f0:a2:3d:60:ab:27:
                    10:f4:ad:51:82:ff:9b:32:cc:d5:76:6b:8f:53:e8:
                    49:e2:95:7a:d2:e1:b3:57:f1:fd:53:5a:87:b9:03:
                    a7:13:e9:5a:34:43:f6:cf:ef:d7:01:9a:bd:ea:0a:
                    cd:4b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                CE:E4:31:B2:E0:72:61:92:D2:ED:57:E5:B7:A6:34:EA:39:3F:B2:BD
            X509v3 Authority Key Identifier:
                keyid:D5:4E:7D:AB:85:3E:70:5D:D8:88:F1:A9:3A:30:95:87:98:02:83:9
C

            X509v3 Subject Alternative Name:
                DNS:www.hisneden.com, DNS:www.hisnedenA.com, DNS:www.hisnedenB.c
```

The above screenshot shows the details of the certificate generated and we can also see the SAN attached details.

Task 4 : Deploying Certificate in an Apache-Based HTTPS Website

Below is the ssl config file where in I made the required changes necessary to show where the file are located. I also copied the cert files to the cert folder on the VM.

hisneden_apache_ssl.conf
~/Desktop/PKI/image_www

```
<VirtualHost *:443>
    DocumentRoot /var/www/hisneden
    ServerName www.hisneden.com
    ServerAlias www.hisnedenA.com
    ServerAlias www.hisnedenB.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /var/www/hisneden
    ServerName www.hisneden.com
    ServerAlias www.hisnedenA.com
    ServerAlias www.hisnedenB.com
    DirectoryIndex index_red.html
</VirtualHost>

# Set the following gloal entry to suppress an annoying warning
message
ServerName localhost
```

Below is the Docker file which I made the required changes replacing the original ‘bank32’ to ‘hisneden’.

Dockerfile
~/Desktop/PKI/image_www

```
FROM handsonsecurity/seed-server:apache-php

ARG WWWDIR=/var/www/hisneden

COPY ./index.html ./index_red.html $WWWDIR/
COPY ./hisneden_apache_ssl.conf /etc/apache2/sites-available
COPY ./certs/server.crt ./certs/server.key  /certs/

RUN  chmod 400 /certs/server.key \
     && chmod 644 $WWWDIR/index.html \
     && chmod 644 $WWWDIR/index_red.html \
     && a2ensite hisneden_apache_ssl

CMD  tail -f /dev/null
```

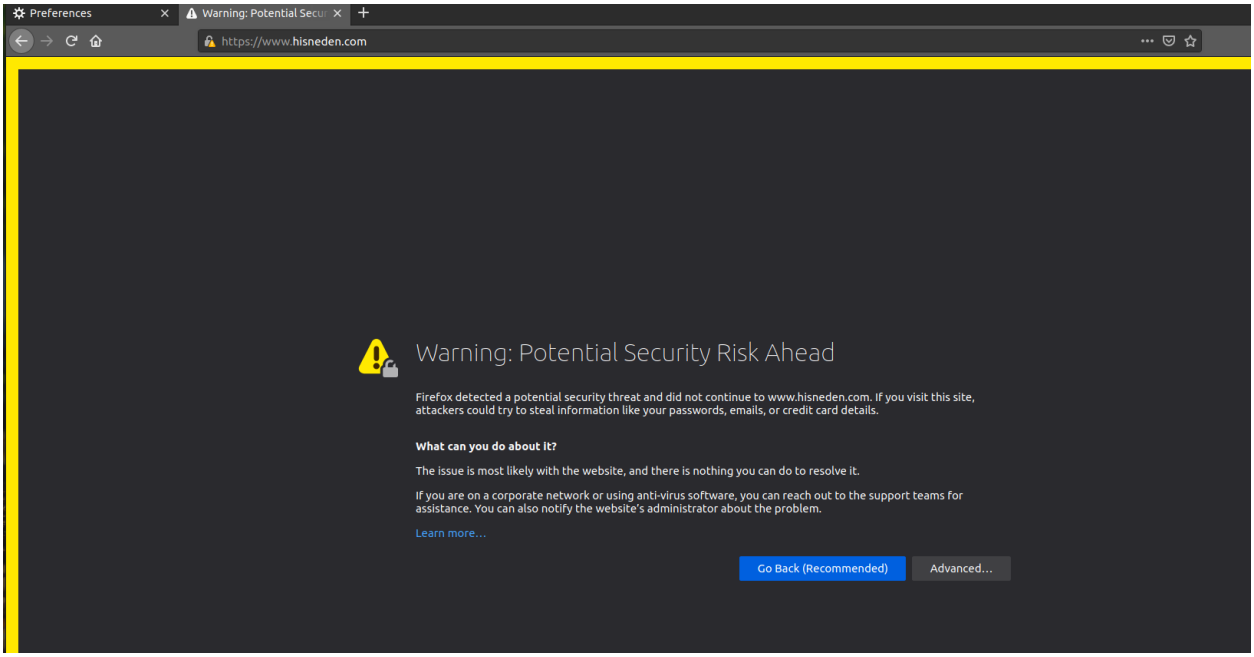
Now I shutdown the containers and build it again. Below I check the apache2 folder to see if the config file I just created is added in.

```
root@f479b29585f0:/# cd /etc/apache2/sites-available/
root@f479b29585f0:/etc/apache2/sites-available# ls
000-default.conf  default-ssl.conf  hisneden_apache_ssl.conf
root@f479b29585f0:/etc/apache2/sites-available#
```

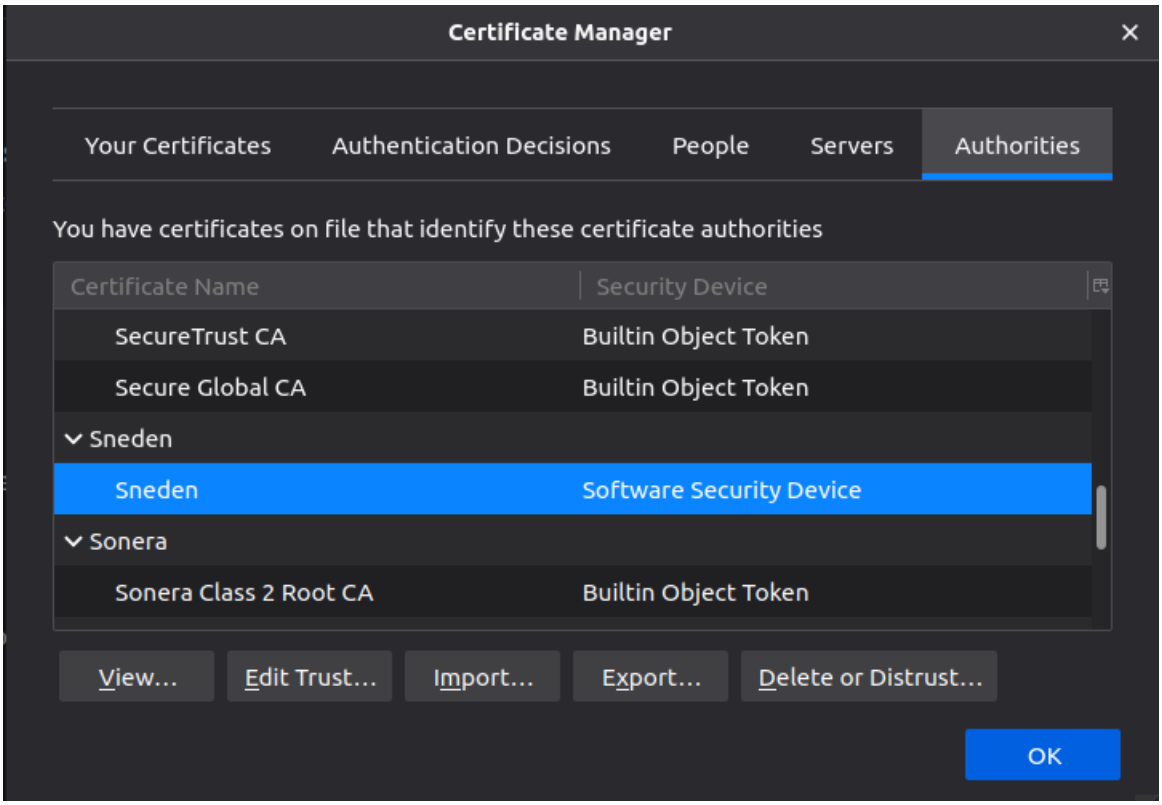
I start the apache2 webserver service. Followed by the passwords for the SSL key.

```
root@f479b29585f0:/# service apache2 status
 * apache2 is not running
root@f479b29585f0:/# service apache2 start
 * Starting Apache httpd web server apache2
Enter passphrase for SSL/TLS keys for www.hisneden.com:443 (RSA):
 *
root@f479b29585f0:/# service apache2 status
 * apache2 is running
```

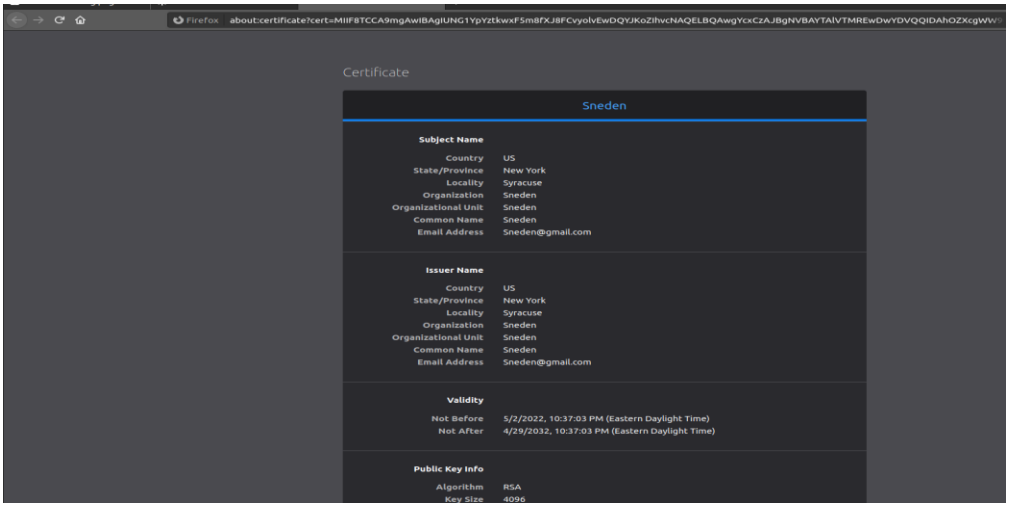
Now I put the website, https://www.hisneden.com:443/ and notice that the error of a security risk shows up. This error is because the the root CA cert has not yet been added in order to sign the csr of www.hisneden.com



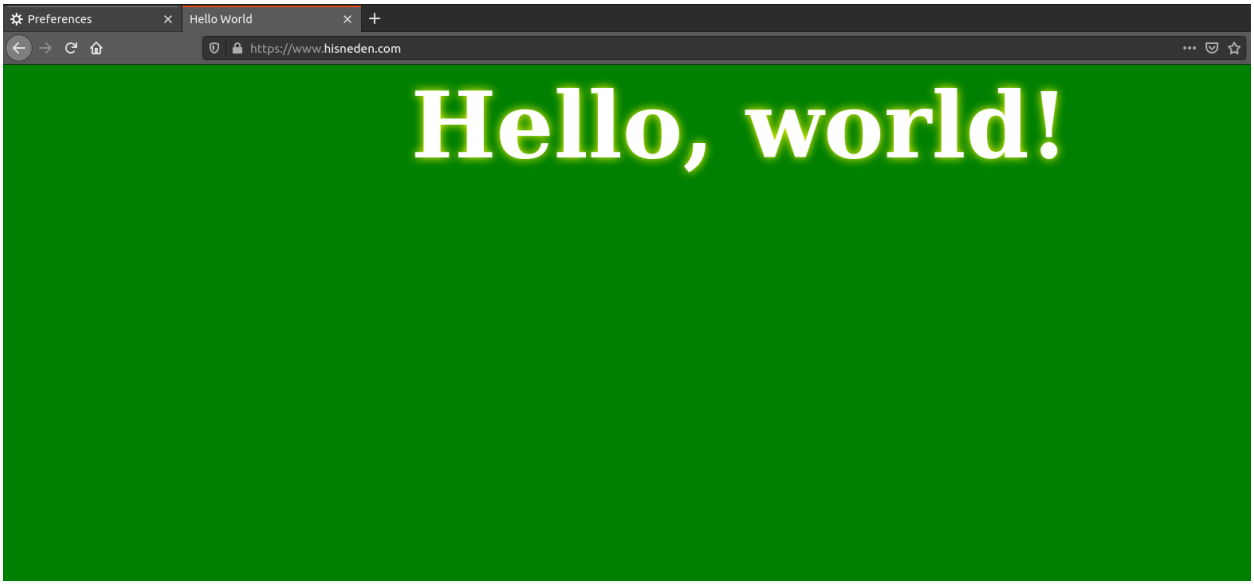
Now I add in the ca certificate to the firefox browser under the authorities section and make sure it is allowed to sign other websites.



Below is the certificate in detail shown in the browser.



Now when I refresh the page, I get access to my webserver. The below runs on domain name, https://www.hisneden.com and shows hello world with a green background.



The below runs on domain name, http://www.hisneden.com and shows hello world with a red background like as a caution of no SSL.

To get it working successfully I added the domain mane to /etc/hosts file on the VM.

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

Step 1: Setting up the malicious website.

Below I add www.example.com to the Servername to impersonate example.com as hisneden.com.

```
<VirtualHost *:443>
    DocumentRoot /var/www/hisneden
    ServerName www.example.com
    ServerAlias www.hisnedenA.com
    ServerAlias www.hisnedenB.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /var/www/hisneden
    ServerName www.hisneden.com
    ServerAlias www.hisnedenA.com
    ServerAlias www.hisnedenB.com
    DirectoryIndex index_red.html
</VirtualHost>

# Set the following gloal entry to suppress an annoying warning
```

Step 2: Becoming the man in the middle

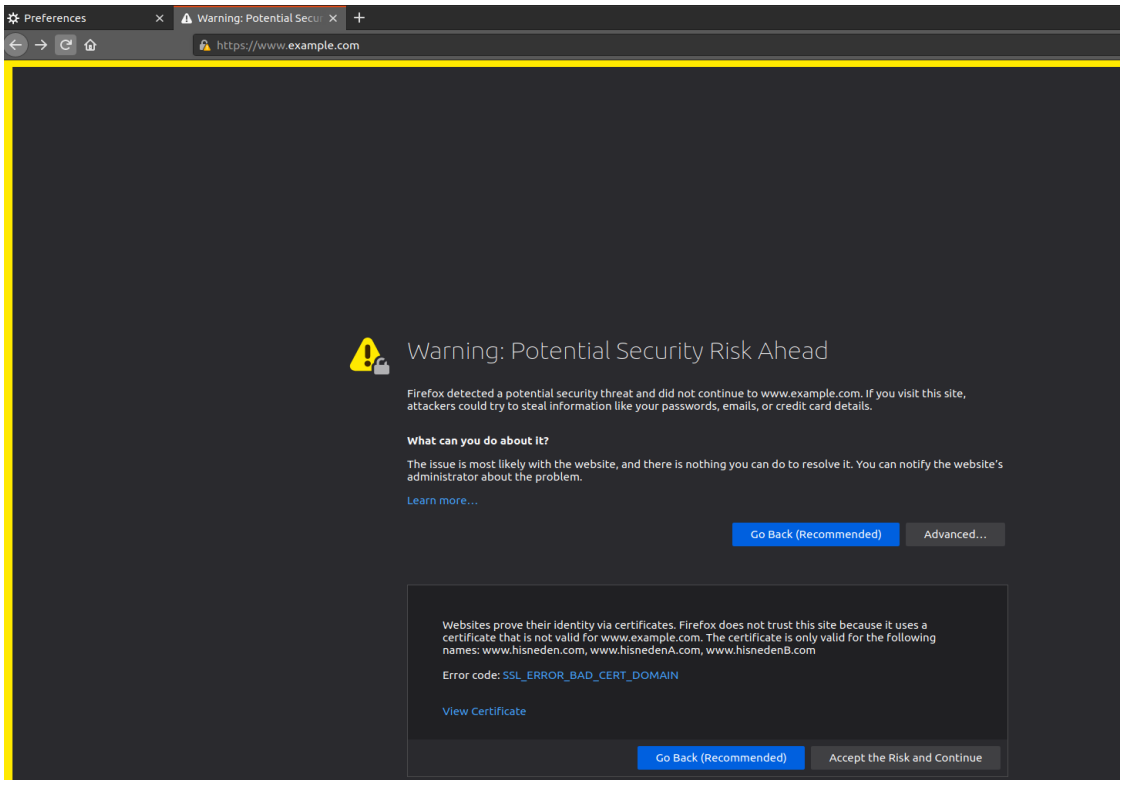
Below I use some method like DNS cache poisoning on the victim thereby poisoning the cache and adding the fake domain name to the cache.

```
root@VM: /etc
 GNU nano 4.8                    hosts

# For XSS Lab
10.9.0.5        www.xsslabelgg.com
10.9.0.5        www.example32a.com
10.9.0.5        www.example32b.com
10.9.0.5        www.example32c.com
10.9.0.5        www.example60.com
10.9.0.5        www.example70.com

# For CSRF Lab
10.9.0.5        www.csrflabelgg.com
10.9.0.5        www.csrflab-defense.com
10.9.0.105      www.csrflab-attacker.com

# For Shellshock Lab
10.9.0.80       www.seedlab-shellshock.com

10.9.0.80       www.hisneden.com
10.9.0.80       www.example.com
```

Step 3: Browse the target website.

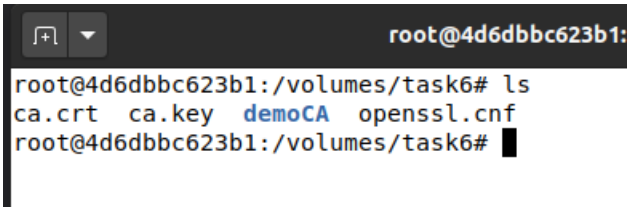
With everything set up, I now visit the target real website – www.example.com

I notice that the browser gives a security error. The error is shown below and mentions that the certificate was for only hisneden and related sub domain names.



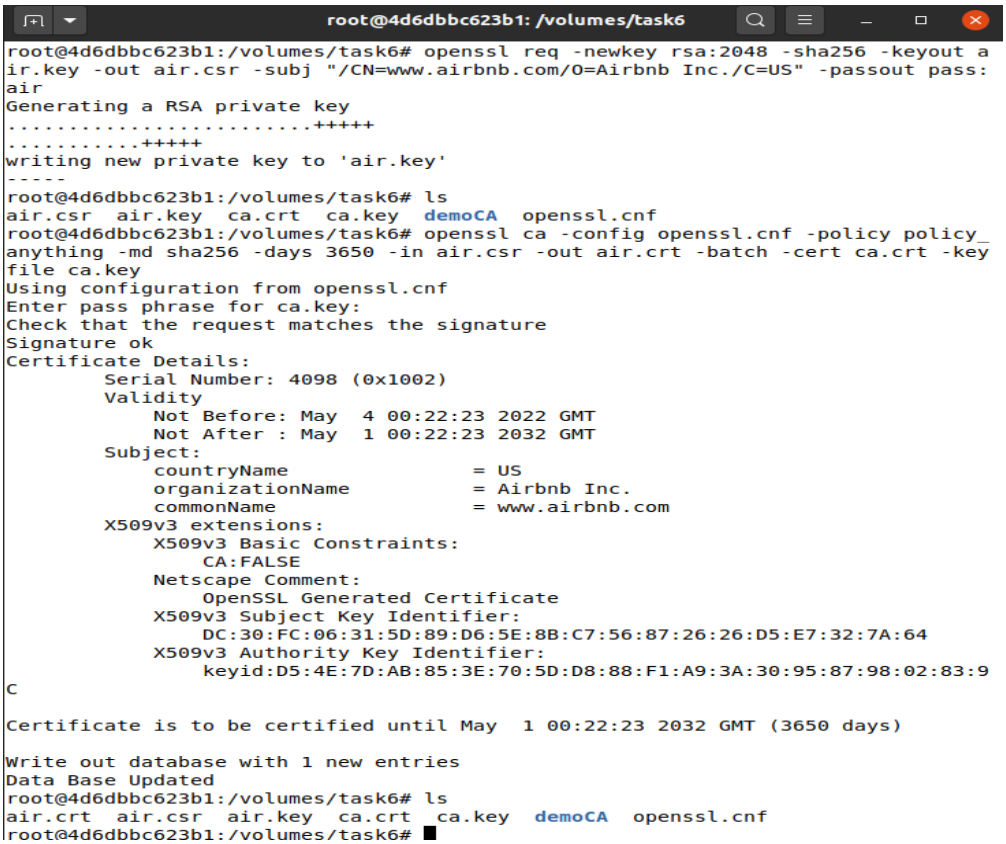
Task 6 : Launching a Man-In-The-Middle Attack with a Compromised CA

Given the attacker has the CA private key. The following shows the information that the attacker would have that is the ca.crt (public) and ca.key (private).

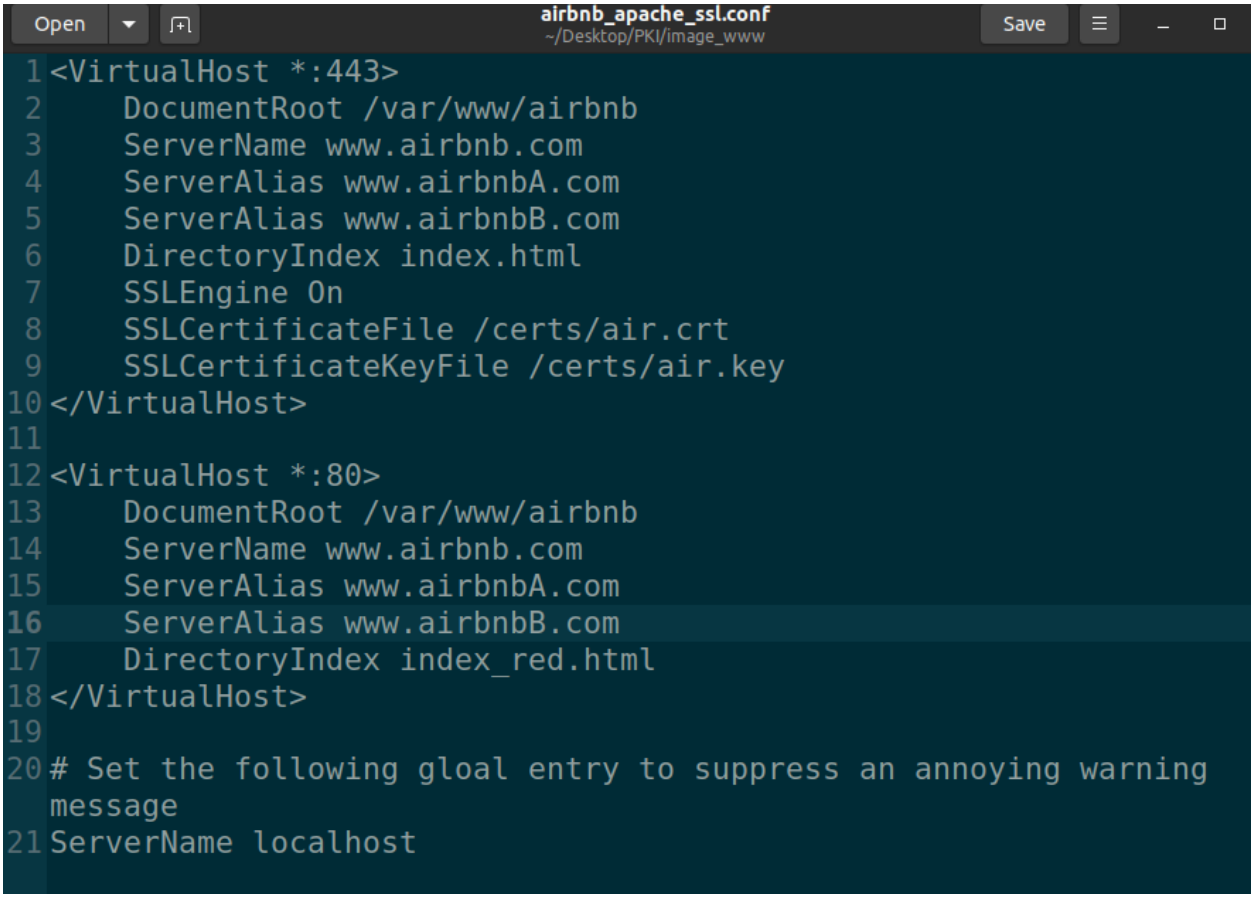


In this attacker I try to impersonate www.airbnb.com. Below I create a csr for www.airbnb.com similar to task 2. I obtain air.crt and air.key.

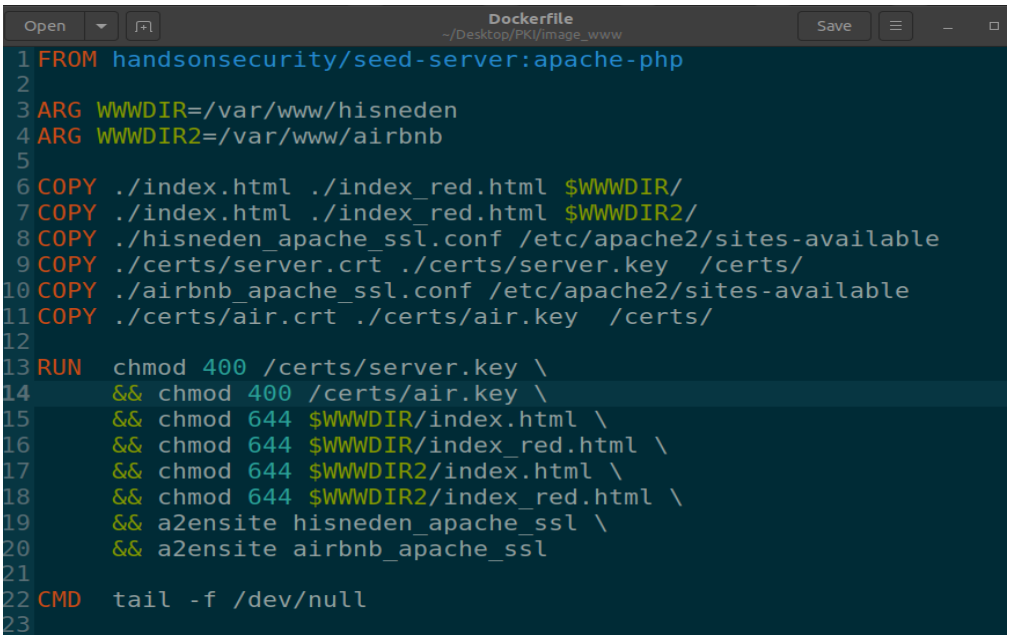
Now similar to task 3 I create the certificate from the Airbnb csr and CA keys to get the Airbnb crt called as air.crt.



Now similar to task 4 I create the Airbnb Apache ssl config file and make the required changes as below.

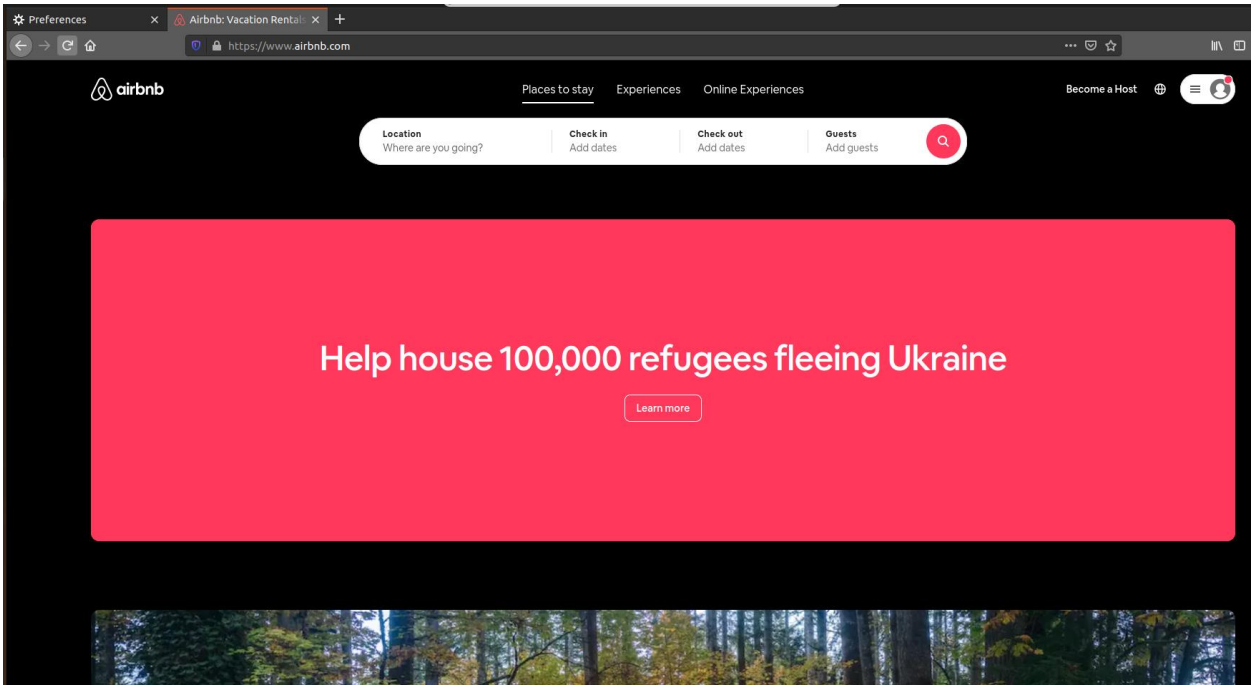


I make the required changes to the Dockerfile as well. As shown below.



I now start the apache2 server and add the passwords for the SSL keys.

I now run www.airbnb.com, this takes me to the authentic website as shown below.



I now include the domain name to the cache via a DNS cache poisoning attack. As shown below is what the cache would look like with the Airbnb domain name included.

```
root@VM: /etc
  GNU nano 4.8                                hosts

# For XSS Lab
10.9.0.5        www.xsslabelgg.com
10.9.0.5        www.example32a.com
10.9.0.5        www.example32b.com
10.9.0.5        www.example32c.com
10.9.0.5        www.example60.com
10.9.0.5        www.example70.com

# For CSRF Lab
10.9.0.5        www.csrflabelgg.com
10.9.0.5        www.csrflab-defense.com
10.9.0.105      www.csrflab-attacker.com

# For Shellshock Lab
10.9.0.80       www.seedlab-shellshock.com

10.9.0.80       www.hisneden.com
10.9.0.80       www.example.com
10.9.0.80       www.airbnb.com
```

I now refresh the browser and notice that the attack took place and was successfully able to impersonate the authentic Airbnb website with the fake one.