ASSIGNMENT # 4

Student Name & ID	**Abdulrazzaq Alsiddiq, 202004464**
Student Name & ID	**Anas Madkoor, 202104114**
Student Name & ID	**Omar Amin, 202003122**
Student Name & ID	**Lance Eric Ruben, 202005801**
Student Name & ID	**Ali Zair, 202109964**

Task 1: Becoming a Certificate Authority (CA)

Copying the /usr/lib/ssl/openssl.conf file into the current directory because we will be modifying it:

```
[03/18/24]seed@VM:~/.../assignment4$ cp /usr/lib/ssl/openssl.cnf ./myCA_openssl.cnf
[03/18/24]seed@VM:~/.../assignment4$ gedit myCA_openssl.cnf
```

Open *myCA_openssl.cnf ~/netsec/assignment4

```
37
38 ####################################################################
39 [ ca ]
40 default_ca  = CA_default        # The default ca section
41
42 ####################################################################
43 [ CA_default ]
44
45 dir     = ./demoCA      # Where everything is kept
46 certs       = $dir/certs        # Where the issued certs are kept
47 crl_dir     = $dir/crl      # Where the issued crl are kept
48 database    = $dir/index.txt    # database index file.
49 unique_subject  = no            # Set to 'no' to allow creation of
50                     # several certs with same subject.
51 new_certs_dir   = $dir/newcerts     # default place for new certs.
52
53 certificate = $dir/cacert.pem   # The CA certificate
54 serial      = $dir/serial       # The current serial number
55 crlnumber   = $dir/crlnumber    # the current crl number
56                     # must be commented out to leave a V1 CRL
57 crl     = $dir/crl.pem      # The current CRL
58 private_key = $dir/private/cakey.pem# The private key
```

We uncomment the unique_subject line to allow creation of several certificates with the same subject.

Generating the self-signed certificate for our CA (Certificate Authority):

```
[03/18/24]seed@VM:~/.../assignment4$ 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]:QA
State or Province Name (full name) [Some-State]:Doha
Locality Name (eg, city) []:Doha
Organization Name (eg, company) [Internet Widgits Pty Ltd]:QACA
Organizational Unit Name (eg, section) []:Qatar Certificate Authority
Common Name (e.g. server FQDN or YOUR name) []:QACA
Email Address []:qaca@qaca.com
```

Inspecting the content of the X509 certificate:

```
[03/18/24]seed@VM:~/.../assignment4$ openssl x509 -in ca.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0c:68:98:5d:a1:48:1b:ab:b1:a7:9c:84:fe:4c:6d:cb:1a:fe:0b:33
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = QA, ST = Doha, L = Doha, O = QACA, OU = Qatar Certificate Authority, CN = QACA, emailAddress = qaca@qaca.com
        Validity
            Not Before: Mar 18 17:27:37 2024 GMT
            Not After : Mar 16 17:27:37 2034 GMT
        Subject: C = QA, ST = Doha, L = Doha, O = QACA, OU = Qatar Certificate Authority, CN = QACA, emailAddress = qaca@qaca.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:af:bf:9f:63:f9:95:5d:5e:a8:4e:95:06:ad:57:
                    12:7c:85:59:19:a3:0d:cb:85:2e:a4:b0:5a:c3:a7:
                    78:27:d4:53:b0:5a:55:5b:82:c0:95:0a:67:c9:25:
                    23:97:57:42:30:6e:47:61:62:49:b9:64:99:33:83:
                    b2:b1:1b:a2:c1:df:60:aa:55:ad:40:26:61:52:2c:
                    86:bd:b6:53:3a:c0:98:f6:82:82:cb:ef:4b:5e:9b:
                    af:2f:83:a8:7e:95:54:14:df:d6:28:74:13:15:97:
                    41:03:8f:55:54:37:3b:02:a6:18:8f:a0:04:30:cc:
                    78:d9:80:8a:01:4c:dd:9e:70:ac:b7:55:9e:0d:d1:
                    fc:73:34:32:89:ff:9e:5a:a7:9f:b0:16:82:ad:a0:
                    63:ad:9a:22:67:6b:02:bc:97:68:ca:83:75:25:66:
                    46:f1:ca:9a:f0:dc:51:29:4c:62:e5:86:38:d5:b0:
                    56:4c:e8:a4:33:a3:cc:5d:11:63:c1:ec:63:42:56:
                    00:8a:99:fc:50:75:52:3b:7b:4b:ad:75:6f:d2:84:
                    29:0b:36:1c:79:e0:2c:6f:90:1d:04:d0:0b:96:48:
                    a8:0c:1a:21:6f:86:a9:a5:8c:3a:96:6e:5b:d1:ff:
                    29:03:c7:89:58:d7:8c:57:42:24:cb:fc:da:e9:03:
                    76:f0:c4:01:a7:63:36:28:ac:66:56:cd:00:16:c1:
```

Inspecting the content of the RSA Key:

```
[03/18/24]seed@VM:~/.../assignment4$ openssl rsa -in ca.key -text -noout
Enter pass phrase for ca.key:
RSA Private-Key: (4096 bit, 2 primes)
modulus:
    00:af:bf:9f:63:f9:95:5d:5e:a8:4e:95:06:ad:57:
    12:7c:85:59:19:a3:0d:cb:85:2e:a4:b0:5a:c3:a7:
    78:27:d4:53:b0:5a:55:5b:82:c0:95:0a:67:c9:25:
    23:97:57:42:30:6e:47:61:62:49:b9:64:99:33:83:
    b2:b1:1b:a2:c1:df:60:aa:55:ad:40:26:61:52:2c:
    86:bd:b6:53:3a:c0:98:f6:82:82:cb:ef:4b:5e:9b:
    af:2f:83:a8:7e:95:54:14:df:d6:28:74:13:15:97:
    41:03:8f:55:54:37:3b:02:a6:18:8f:a0:04:30:cc:
    78:d9:80:8a:01:4c:dd:9e:70:ac:b7:55:9e:0d:d1:
    fc:73:34:32:89:ff:9e:5a:a7:9f:b0:16:82:ad:a0:
    63:ad:9a:22:67:6b:02:bc:97:68:ca:83:75:25:66:
    46:f1:ca:9a:f0:dc:51:29:4c:62:e5:86:38:d5:b0:
    56:4c:e8:a4:33:a3:cc:5d:11:63:c1:ec:63:42:56:
    00:8a:99:fc:50:75:52:3b:7b:4b:ad:75:6f:d2:84:
    29:0b:36:1c:79:e0:2c:6f:90:1d:04:d0:0b:96:48:
    a8:0c:1a:21:6f:86:a9:a5:8c:3a:96:6e:5b:d1:ff:
    29:03:c7:89:58:d7:8c:57:42:24:cb:fc:da:e9:03:
    76:f0:c4:01:a7:63:36:28:ac:66:56:cd:00:16:c1:
    ce:fc:b0:fa:ad:cf:50:ca:94:d1:4e:c4:48:8d:3e:
    bc:bc:3c:5b:24:42:20:cc:e9:5b:5d:50:af:5a:ea:
    41:4a:c9:46:c1:2e:92:73:0b:bc:7d:f7:71:e8:70:
    61:a4:27:4c:06:76:7f:ce:56:33:e2:3b:99:fa:96:
    93:7a:c5:c8:aa:a7:d7:a0:d3:5e:2b:87:f6:d2:38:
    0d:7c:f1:35:34:00:b8:47:68:e4:30:56:26:45:14:
    f6:44:de:f3:b7:49:45:5e:5a:22:9a:5a:77:c1:2b:
    80:18:8c:c9:3e:58:da:f8:bb:98:71:a8:09:b7:2a:
    65:bc:79:33:70:f0:fc:38:52:0e:6c:03:06:0a:31:
    81:df:90:d5:6d:a6:fa:43:0c:32:f4:7e:3c:5d:0e:
    5f:2f:2f:af:34:d3:9b:8f:92:5c:52:e3:e2:03:e5:
    db:39:7e:aa:94:a4:71:5e:f2:f6:8d:54:19:70:4d:
```

- What part of the certificate indicates this is a CA's certificate?
  While inspecting the output of the ca.crt file, we can clearly see a field "CA" which has the value set to TRUE indicating us that the certificate is for a certificate authority:

```
X509v3 Basic Constraints: critical
    CA:TRUE
```

- What part of the certificate indicates this is a self-signed certificate?
  The issuer field and the subject field have the same content showing us that the certificate is a self-signed one:

```
Serial Number:
    0c:68:98:5d:a1:48:1b:ab:b1:a7:9c:84:fe:4c:6d:cb:1a:fe:0b:33
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = QA, ST = Doha, L = Doha, O = QACA, OU = Qatar Certificate Authority, CN = QACA, emailAddress = qaca@qaca.com
Validity
    Not Before: Mar 18 17:27:37 2024 GMT
    Not After : Mar 16 17:27:37 2034 GMT
Subject: C = QA, ST = Doha, L = Doha, O = QACA, OU = Qatar Certificate Authority, CN = QACA, emailAddress = qaca@qaca.com
```

- 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.
  From the certificate file and the private key file, we can see the modulus n and public exponent e:

```
RSA Private-Key: (4096 bit, 2 primes)
modulus:
    00:af:bf:9f:63:f9:95:5d:5e:a8:4e:95:06:ad:57:
    12:7c:85:59:19:a3:0d:cb:85:2e:a4:b0:5a:c3:a7:
    78:27:d4:53:b0:5a:55:5b:82:c0:95:0a:67:c9:25:
    23:97:57:42:30:6e:47:61:62:49:b9:64:99:33:83:
    b2:b1:1b:a2:c1:df:60:aa:55:ad:40:26:61:52:2c:
    86:bd:b6:53:3a:c0:98:f6:82:82:cb:ef:4b:5e:9b:
    af:2f:83:a8:7e:95:54:14:df:d6:28:74:13:15:97:
    41:03:8f:55:54:37:3b:02:a6:18:8f:a0:04:30:cc:
    78:d9:80:8a:01:4c:dd:9e:70:ac:b7:55:9e:0d:d1:
    fc:73:34:32:89:ff:9e:5a:a7:9f:b0:16:82:ad:a0:
    63:ad:9a:22:67:6b:02:bc:97:68:ca:83:75:25:66:
    46:f1:ca:9a:f0:dc:51:29:4c:62:e5:86:38:d5:b0:
    56:4c:e8:a4:33:a3:cc:5d:11:63:c1:ec:63:42:56:
    00:8a:99:fc:50:75:52:3b:7b:4b:ad:75:6f:d2:84:
    29:0b:36:1c:79:e0:2c:6f:90:1d:04:d0:0b:96:48:
    a8:0c:1a:21:6f:86:a9:a5:8c:3a:96:6e:5b:d1:ff:
    29:03:c7:89:58:d7:8c:57:42:24:cb:fc:da:e9:03:
    76:f0:c4:01:a7:63:36:28:ac:66:56:cd:00:16:c1:
    ce:fc:b0:fa:ad:cf:50:ca:94:d1:4e:c4:48:8d:3e:
    bc:bc:3c:5b:24:42:20:cc:e9:5b:5d:50:af:5a:ea:
    41:4a:c9:46:c1:2e:92:73:0b:bc:7d:f7:71:e8:70:
    61:a4:27:4c:06:76:7f:ce:56:33:e2:3b:99:fa:96:
    93:7a:c5:c8:aa:a7:d7:a0:d3:5e:2b:87:f6:d2:38:
    0d:7c:f1:35:34:00:b8:47:68:e4:30:56:26:45:14:
    f6:44:de:f3:b7:49:45:5e:5a:22:9a:5a:77:c1:2b:
    80:18:8c:c9:3e:58:da:f8:bb:98:71:a8:09:b7:2a:
    65:bc:79:33:70:f0:fc:38:52:0e:6c:03:06:0a:31:
    81:df:90:d5:6d:a6:fa:43:0c:32:f4:7e:3c:5d:0e:
    5f:2f:2f:af:34:d3:9b:8f:92:5c:52:e3:e2:03:e5:
    db:39:7e:aa:94:a4:71:5e:f2:f6:8d:54:19:70:4d:
    d4:ee:70:87:a3:44:32:83:49:e6:8a:9e:71:a4:c6:
    2d:df:86:d9:15:fc:e5:29:2a:a1:9e:d9:12:d8:d0:
    f3:39:e6:ab:12:67:8b:c6:ca:6a:60:5b:94:44:e4:
    32:0d:e7:fd:25:fa:9b:0b:ed:e9:65:00:e4:6e:a5:
    bf:27:07
publicExponent: 65537 (0x10001)
```

From the key file, we can see the private exponent d and the two secret numbers p and q:

```
privateExponent:
    00:92:45:89:12:3b:3a:9e:60:56:f2:38:44:3b:66:
    b9:c3:1b:74:e1:ca:7c:83:c3:cf:e3:4f:c7:eb:09:
    6a:0e:b4:40:07:09:d7:fa:f6:e2:f3:e8:9a:22:a6:
    1f:6e:29:38:b4:78:44:3e:80:00:5e:25:a5:00:63:
    ff:08:3a:b9:06:64:b4:de:6f:ba:67:26:ca:5f:0b:
    22:05:a9:46:b2:22:73:ec:cf:08:af:54:f5:44:c3:
    8c:55:9e:5d:51:25:55:ba:9d:e5:6e:fb:d4:9a:66:
    ad:bc:99:c1:e9:fa:a1:c9:7d:95:f9:8b:b3:91:2d:
    cc:f3:31:71:2a:cd:df:b1:ab:93:84:6e:e4:c2:19:
    df:e3:be:30:c5:fc:12:be:de:be:20:5f:4b:d9:d8:
    20:bd:ef:07:ed:4c:fa:15:1e:6b:20:09:b3:29:32:
    6b:9f:78:01:e6:af:67:79:58:d3:02:b6:e8:7c:a9:
    6b:da:4c:d1:c5:0c:f1:dd:0f:c9:18:30:77:18:51:
    dd:d7:7d:7b:ab:37:49:fe:a9:8a:7d:ce:bd:07:d7:
    e9:15:24:af:24:9c:78:3f:4e:11:68:dd:02:97:cc:
    d6:d8:63:b3:2a:50:48:c7:b8:e3:56:2a:79:8f:58:
    af:b9:2e:8c:58:7d:e5:97:06:3c:fb:ed:f2:00:77:
    5a:b8:7b:e8:ae:5b:db:08:23:94:a4:f3:29:59:62:
    2c:39:29:ec:b0:65:d9:91:0f:18:ed:f2:a8:ec:76:
    e4:bf:15:54:af:f4:9f:a4:b5:67:76:ed:24:5b:3e:
    db:a4:25:d8:38:84:37:61:38:ee:d2:82:53:90:83:
    7f:a6:a8:24:bc:07:fa:96:48:55:6a:fa:59:8d:94:
    19:b1:9b:23:65:73:d1:73:ee:27:d7:57:78:74:ca:
    06:d7:61:a7:c5:02:d7:bc:5a:9a:b6:39:c0:61:49:
    1d:24:1e:c0:be:2e:04:ed:af:38:5a:55:36:c4:62:
    c7:d4:86:88:09:77:35:dc:3e:52:11:d3:36:f8:8d:
    dc:f0:7c:dc:50:bb:0e:5f:c2:5f:e9:0b:4e:8b:a8:
    f8:03:a2:dd:40:c5:81:c4:06:53:b3:7e:fe:2b:c6:
    04:f2:38:86:2c:aa:8b:99:4e:fc:06:17:14:0d:e5:
    50:65:06:ca:80:77:3a:f6:0f:b7:bc:37:85:6f:2e:
    33:bd:66:15:6d:61:73:14:bb:e2:99:16:47:47:e5:
    1a:c8:08:b5:44:f0:8d:f1:cb:17:18:c4:f9:35:e4:
    22:b3:e7:cf:56:8d:8b:4c:6c:b5:73:89:64:e2:5e:
    13:b6:05:97:c9:23:30:1b:f5:58:5b:e7:a2:35:c3:
    a2:fb:71
```

```
prime1:
    00:e7:98:6a:87:fb:c9:11:1b:97:d3:63:56:eb:37:
    ee:1e:f9:49:d6:f2:10:9e:37:eb:40:dc:44:ef:8c:
    27:3f:e7:1b:87:36:cc:07:95:9b:9d:5e:4f:9f:98:
    5f:c1:1f:6e:6b:f1:e5:70:67:d5:cb:2f:ab:dd:25:
    d4:82:bc:03:e0:9b:38:ea:3c:e0:67:fb:dc:70:e7:
    1d:ec:1a:73:d4:93:a5:c7:a6:cd:12:42:cc:7a:bf:
    90:e6:5f:d7:f3:55:57:f6:c5:00:65:55:f2:df:ea:
    6b:07:8c:35:48:f0:b9:e4:79:88:4f:11:21:11:8b:
    b1:a3:1f:5d:67:fd:39:23:c5:5e:b3:32:96:82:45:
    6f:60:6f:a0:82:ee:de:35:4a:ec:9c:f3:9c:6f:c5:
    43:e6:7b:37:5c:27:43:35:9e:68:5d:d9:5a:5a:5d:
    ef:c4:ad:d3:c9:2d:5e:2e:88:ae:3d:4c:40:15:5c:
    e5:c7:e6:52:4b:a1:08:9d:ac:99:54:8c:20:cd:ec:
    df:de:cf:83:22:55:0b:07:3d:a9:67:b4:4d:67:ea:
    eb:60:79:6d:ab:1a:91:d3:22:22:22:50:41:36:3c:
    74:f9:e8:48:90:99:e5:69:44:4f:88:8a:39:b7:fa:
    57:19:a2:ac:30:96:f9:d3:02:b2:28:ec:b9:b5:82:
    e8:09
prime2:
    00:c2:44:aa:6b:ad:c6:2d:69:68:f5:e4:cc:a9:ab:
    f0:db:1c:2e:f7:25:2d:3d:20:64:b3:e5:90:35:3d:
    37:76:00:d7:e3:b2:59:0e:5b:d4:93:08:c3:ac:a9:
    a5:42:05:6a:aa:8f:5e:60:3a:12:53:ae:c6:bb:cf:
    cf:a9:9d:8f:1d:a6:86:d4:96:14:01:51:b8:29:6c:
    09:65:f3:8a:b9:14:e8:44:f8:c8:6a:77:dd:c2:51:
    92:52:4c:78:34:78:28:68:5b:fb:37:c5:1f:60:2b:
    f3:d7:dc:37:dd:ab:69:02:09:fd:a0:2e:29:4b:a4:
    75:56:a6:83:77:6d:a8:be:b0:58:db:b0:84:2a:99:
    80:4c:34:87:06:17:ed:52:37:d0:f4:55:b0:8a:82:
    e8:db:9f:ba:32:29:a7:c1:db:09:b7:c8:4d:d4:3d:
    82:6e:b4:e3:e9:7c:85:c1:59:99:de:9b:ef:d2:8c:
    3c:8f:87:0e:6f:4e:a1:3c:77:72:3f:af:12:45:87:
    da:a2:18:61:34:db:66:2d:4e:d7:9b:69:25:64:30:
    d7:61:c5:84:f8:47:a8:95:c7:91:92:28:17:7a:75:
    53:1c:e3:e9:9d:43:95:2f:ef:40:2b:fd:a1:7b:9c:
    40:17:29:a0:dd:c3:b3:4f:97:98:67:e8:79:95:01:
    ba:8f
```

Task 2: Generating a Certificate Request for Your Web Server

Generating a certificate signing request for our web server:

```
[03/18/24]seed@VM:~/.../assignment4$ openssl req -newkey rsa:2048 -sha256 -keyout foss.key -out foss.c
sr -subj "/CN=www.foss.com/O=Foss/C=QA" -passout pass:dees -addext "subjectAltName = DNS:www.foss.com,
 \
> DNS:www.fossA.com, \
> DNS:www.fossB.com"
Generating a RSA private key
........+++++
.....................................................................................................
.....................................................+++++
writing new private key to 'foss.key'
-----
```

The -addext flag, allows us to add alternative names to our server. This command has now generated the public / private key of our web server along with a certificate signing request (csr).

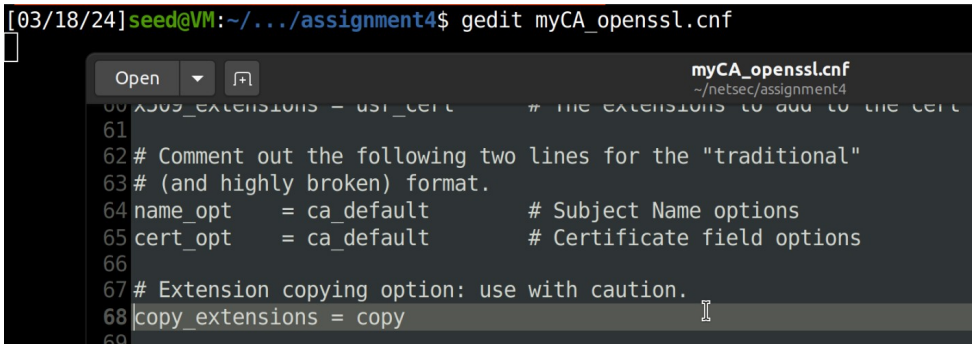
Looking at the decoded content of the csr and the private key file of our web server:

```
[03/18/24]seed@VM:~/.../assignment4$ openssl req -in foss.csr -text -noout
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: CN = www.foss.com, O = Foss, C = QA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b2:d9:72:cc:f8:a0:88:c2:0f:05:9b:f8:07:2b:
                    d4:04:90:17:74:0b:f4:71:3d:6d:d2:f1:5e:ea:b0:
                    c0:b7:c1:c4:1f:4c:96:b2:c7:80:1c:02:ee:8e:7c:
                    9c:63:ca:2c:66:ee:03:3e:3e:eb:21:59:fe:fb:9f:
                    dd:e2:13:44:16:51:30:89:6e:64:02:b9:1e:4e:c6:
                    89:26:5d:27:59:0c:b4:87:58:44:27:6e:c5:c2:c8:
                    91:60:43:ba:86:16:67:60:09:d2:f0:1f:79:1a:7f:
                    07:7e:4e:e9:75:f8:a0:8a:14:c3:71:c2:72:7c:7a:
                    b4:5f:de:4f:54:b0:3e:39:f3:6f:14:73:2a:e0:0b:
                    34:ef:a3:d4:4e:1e:58:4a:e5:52:93:d8:74:5f:20:
                    42:37:13:9e:dc:bc:3c:a0:1c:07:36:ce:43:13:70:
                    aa:81:ca:87:fe:59:15:bb:06:0c:97:04:23:ad:12:
                    cd:97:7c:61:c7:e0:01:39:85:3a:8d:4e:e1:27:9f:
                    4d:19:e0:3d:dc:23:54:0c:b5:6c:9d:4f:f1:f5:57:
                    1f:8a:5f:43:d6:cc:89:53:38:42:0a:01:46:96:c5:
                    15:f2:d1:22:f5:04:f0:95:2a:76:f8:94:b7:bc:e7:
                    fc:ca:65:83:21:e1:5d:c0:62:ac:2b:72:f3:6c:30:
                    44:01
                Exponent: 65537 (0x10001)
```

```
[03/18/24]seed@VM:~/.../assignment4$ openssl rsa -in foss.key -text -noout
Enter pass phrase for foss.key:
RSA Private-Key: (2048 bit, 2 primes)
modulus:
    00:b2:d9:72:cc:f8:a0:88:c2:0f:05:9b:f8:07:2b:
    d4:04:90:17:74:0b:f4:71:3d:6d:d2:f1:5e:ea:b0:
    c0:b7:c1:c4:1f:4c:96:b2:c7:80:1c:02:ee:8e:7c:
    9c:63:ca:2c:66:ee:03:3e:3e:eb:21:59:fe:fb:9f:
    dd:e2:13:44:16:51:30:89:6e:64:02:b9:1e:4e:c6:
    89:26:5d:27:59:0c:b4:87:58:44:27:6e:c5:c2:c8:
    91:60:43:ba:86:16:67:60:09:d2:f0:1f:79:1a:7f:
    07:7e:4e:e9:75:f8:a0:8a:14:c3:71:c2:72:7c:7a:
    b4:5f:de:4f:54:b0:3e:39:f3:6f:14:73:2a:e0:0b:
    34:ef:a3:d4:4e:1e:58:4a:e5:52:93:d8:74:5f:20:
    42:37:13:9e:dc:bc:3c:a0:1c:07:36:ce:43:13:70:
    aa:81:ca:87:fe:59:15:bb:06:0c:97:04:23:ad:12:
    cd:97:7c:61:c7:e0:01:39:85:3a:8d:4e:e1:27:9f:
    4d:19:e0:3d:dc:23:54:0c:b5:6c:9d:4f:f1:f5:57:
    1f:8a:5f:43:d6:cc:89:53:38:42:0a:01:46:96:c5:
    15:f2:d1:22:f5:04:f0:95:2a:76:f8:94:b7:bc:e7:
    fc:ca:65:83:21:e1:5d:c0:62:ac:2b:72:f3:6c:30:
    44:01
publicExponent: 65537 (0x10001)
privateExponent:
    00:a4:48:37:3a:6e:3a:cf:c4:29:96:46:71:2a:ed:
    28:60:54:97:26:82:80:b3:af:f0:7c:6e:38:78:ad:
    89:28:81:b5:0b:e7:07:1a:0b:44:f2:f6:02:79:21:
    9f:69:ce:60:c0:df:6a:5d:37:e0:35:8f:7c:37:57:
```

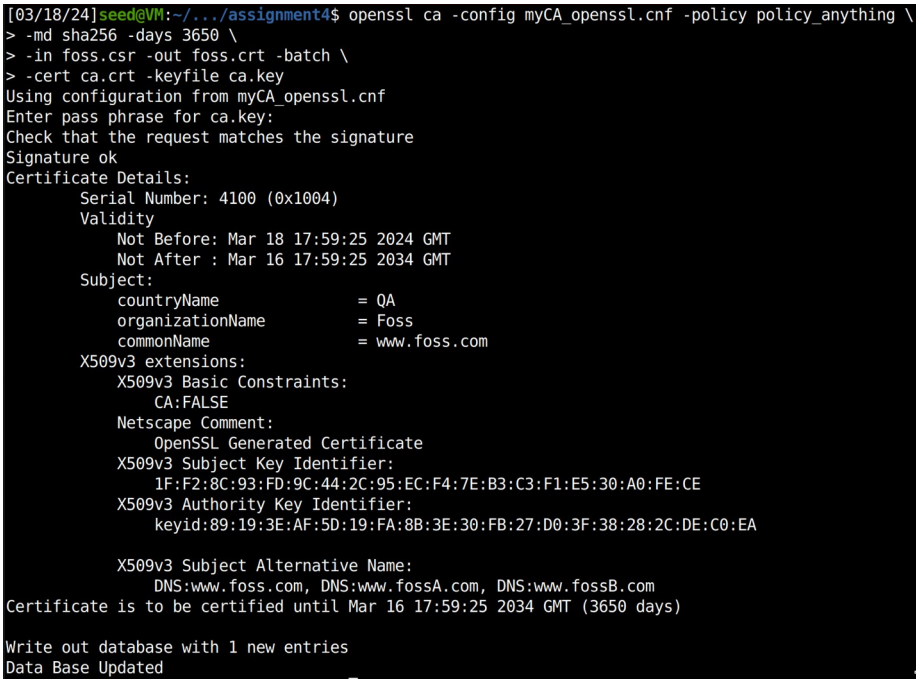
Task 3: Generating a Certificate for your server

Editing the the copy of the openssl.cnf file we took in task 1, and uncommenting the line copy_extension = copy to copy the extension field from the certificate signing request to the final certificate:

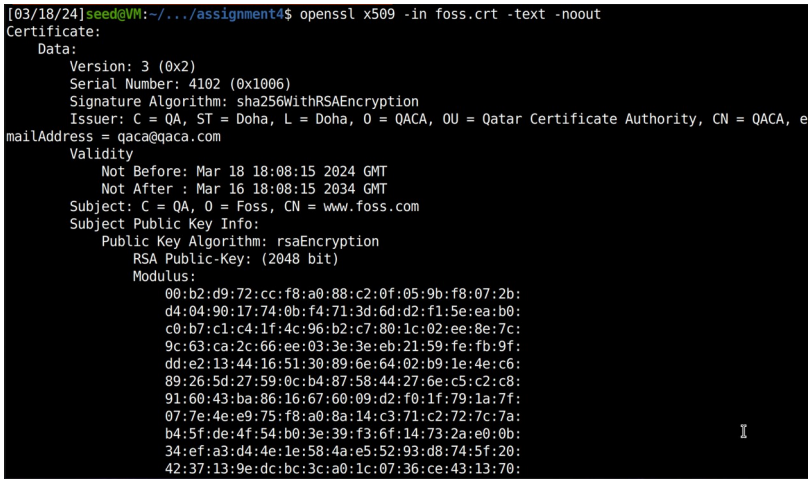


Creating the directories and files that the openssl configuration file requires to create the certificate:

Generating the certificate for our web server using the certificate signing request:



Printing out the decoded content of the certificate to see the alternative names being included in the final certificate:



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

Generating the certificate signing request for our web server:

```
[03/19/24]seed@VM:~/.../assignment4$ openssl req -newkey rsa:2048 -sha256 -keyout foss.key -out fos
s.csr -subj "/CN=www.foss.com/O=Foss LTD../C=QA" -passout pass:dees -addext "subjectAltName = DNS:w
ww.foss.com, DNS:www.fossA.com, DNS:www.fossB.com"
Generating a RSA private key
.....................................................+++++
.....................................................................+++++
writing new private key to 'foss.key'
-----
```

Generating the certificate for our web server using the certificate signing request:

```
[03/19/24]seed@VM:~/.../assignment4$ openssl ca -config myCA_openssl.cnf -policy policy_anything -md sh
a256 -days 3650 -in foss.csr -out foss.crt -batch -cert ca.crt -keyfile ca.key
Using configuration from myCA_openssl.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: Mar 19 17:17:36 2024 GMT
            Not After : Mar 17 17:17:36 2034 GMT
        Subject:
            countryName               = QA
            organizationName          = Foss LTD..
            commonName                = www.foss.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                59:A1:8A:D0:7A:A6:62:49:68:D9:29:8E:81:73:1A:50:AF:33:00:22
            X509v3 Authority Key Identifier:
                keyid:89:19:3E:AF:5D:19:FA:8B:3E:30:FB:27:D0:3F:38:28:2C:DE:C0:EA

            X509v3 Subject Alternative Name:
                DNS:www.foss.com, DNS:www.fossA.com, DNS:www.fossB.com
Certificate is to be certified until Mar 17 17:17:36 2034 GMT (3650 days)
```

Then we navigate to the Labsetup folder, run the dcbuild and dcup commands to start the docker container. After that, we get a shell on the docker container using the docksh command. Then we navigate to the /etc/apache2/sites-available directory of the docker container and copy the bank32_apache_ssl.conf file to make a copy for our own web server. Following that, we edit this conf file for our web server and give it the right values for the fields:

```
 GNU nano 4.8                        foss_apache_ssl.conf                        Modified
<VirtualHost *:443>
    DocumentRoot /var/www/foss
    ServerName www.foss.com
    ServerAlias www.fossA.com
    ServerAlias www.fossB.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/foss.crt
    SSLCertificateKeyFile /certs/foss.key
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /var/www/foss
    ServerName www.foss.com
    DirectoryIndex index_red.html
</VirtualHost>

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

Then we navigate to the /var/www directory and create a new directory where we are going to host the html files for our web server:

```
root@dd8464aebde7:/var/www# mkdir foss
root@dd8464aebde7:/var/www# cp bank32/* foss/
root@dd8464aebde7:/var/www# cd foss/
root@dd8464aebde7:/var/www/foss# ls
index.html  index_red.html
```

Copying the foss.crt and foss.key files in the volumes (shared folder between our virtual machine and the docker container):

```
[03/19/24]seed@VM:~/.../assignment4$ cp foss.crt foss.key Labsetup/volumes/
[03/19/24]seed@VM:~/.../assignment4$
```

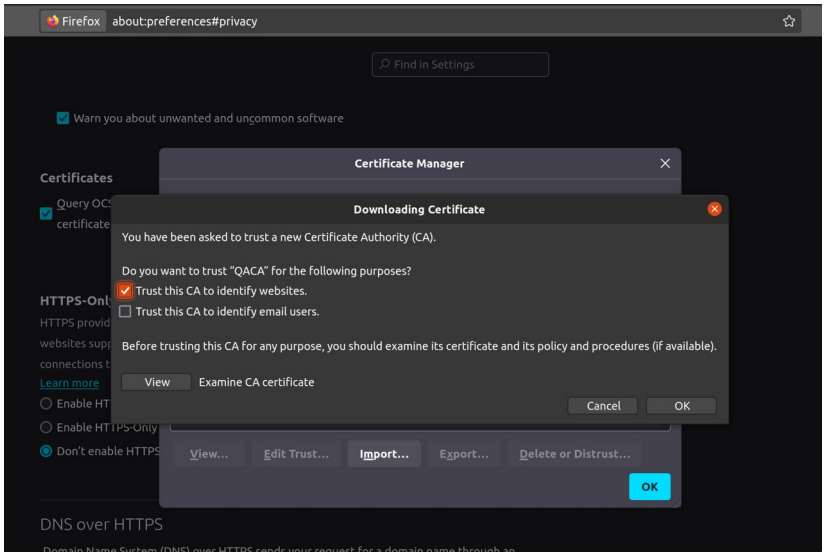
Copying the web server's key and certificate files in the /certs directory because that is what we saved as the location of the files in the config file under the /etc/apache2/sites-available directory:

```
root@dd8464aebde7:/volumes# cp foss.crt foss.key /certs/
root@dd8464aebde7:/volumes#
```

Now, we need to edit the /etc/hosts file on our virtual machine to make the domain www.foss.com be resolved to the IP address of our docker container:

```
 GNU nano 4.8                                    /etc/hosts
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# For DNS Rebinding Lab
192.168.60.80   www.seedIoT32.com

# For SQL Injection Lab
10.9.0.5        www.SeedLabSQLInjection.com

# 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.foss.com
```

Then we need to paste this “about:preferences#privacy” in the url box of firefox and scroll down to find the “View Certificates” button and click on it. We then import the the certificate authority’s certificate in firefox and make it trusted:



After this, we need to change our current directory back to /etc/apache2/sites-available and run the a2ensite command on the foss_apache_ssl.conf file:

```
root@dd8464aebde7:/etc/apache2/sites-available# a2ensite foss_apache_ssl.conf
Enabling site foss_apache_ssl.
To activate the new configuration, you need to run:
  service apache2 reload
root@dd8464aebde7:/etc/apache2/sites-available# service apache2 reload
 * Reloading Apache httpd web server apache2
 *
root@dd8464aebde7:/etc/apache2/sites-available# service apache2 restart
 * Restarting Apache httpd web server apache2
Enter passphrase for SSL/TLS keys for www.foss.com:443 (RSA):
                                                                    [ OK ]
root@dd8464aebde7:/etc/apache2/sites-available#
```

Eventually, when we visit the web server in firefox through our virtual machine, we are able to securely connect to our web server using https:

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

For this task we are going to try to impersonate another legitimate website however, we will be unable to do so proving the point of digital certificates and why they are important for security and integrity:

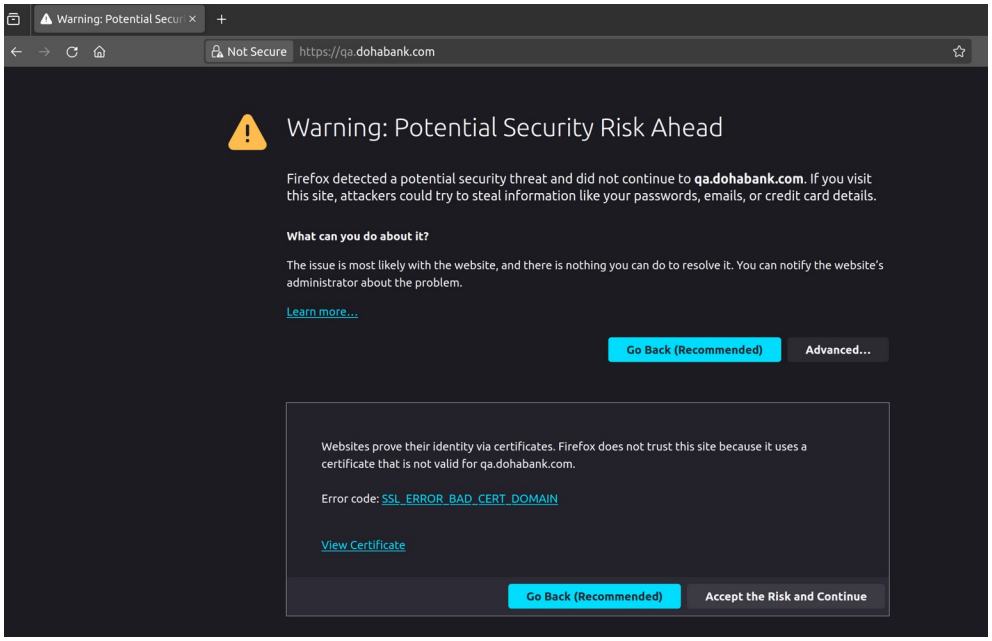
First, we edit the configuration file of our web server to change the server name to the domain of the website we are trying to impersonate (in our case, qa.dohabank.com):

```
 GNU nano 4.8                              foss_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/foss
    ServerName qa.dohabank.com
    ServerAlias www.fossA.com
    ServerAlias www.fossB.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/foss.crt
    SSLCertificateKeyFile /certs/foss.key
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /var/www/foss
    ServerName qa.dohabank.com
    DirectoryIndex index_red.html
</VirtualHost>

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

Then, we are going to mimic a dns cache poisoning attack on our system by modifying our /etc/hosts file to map the qa.dohabank.com website to our web server's IP address:

```
 GNU nano 4.8                                        /etc/hosts
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# For DNS Rebinding Lab
192.168.60.80   www.seedIoT32.com

# For SQL Injection Lab
10.9.0.5        www.SeedLabSQLInjection.com

# 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       qa.dohabank.com
```

Now, when we try to visit the qa.dohabank.com website, we see the following error message:



So here, we can see that the certificate that our web server provided for the qa.dohabank.com website was invalid since the certificate was issued for the website: www.foss.com. Therefore, firefox has detected that the digital certificate the website is providing to prove it's identity is not valid for the qa.dohabank.com domain. This highlights the huge advantage of digital certificates being that even if some device has been attacked using dns cache poisoning, the browser is still going to be able to notice that the website the user is navigating to is not the legitimate website.

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

Copying the certificate authority's private key and certificate file as well as the openssl configuration file to the volumes shared folder (since we are assuming the attacker has compromised them):

```
[03/19/24]seed@VM:~/.../assignment4$ cp ca.key ca.crt myCA_openssl.cnf Labsetup/volumes/
[03/19/24]seed@VM:~/.../assignment4$
```

Generating the certificate signing request to impersonate the dohabank website:

```
root@dd8464aebde7:/volumes# openssl req -newkey rsa:2048 -sha256 -keyout dohabank.key -out dohabank
.csr -subj "/CN=qa.dohabank.com/O=Doha Bank/C=QA" -passout pass:dees -addext "subjectAltName = DNS:
qa.dohabank.com, DNS:qa.dohabankA.com, DNS:qa.dohabankB.com"
Generating a RSA private key
.............................+++++
...........+++++
writing new private key to 'dohabank.key'
-----
```

Creating the directories required by the openssl.cnf file for the creation of the certificate:

```
root@dd8464aebde7:/volumes# mkdir demoCA
root@dd8464aebde7:/volumes# cd demoCA/
root@dd8464aebde7:/volumes/demoCA# mkdir crl newcerts certs
root@dd8464aebde7:/volumes/demoCA# touch index.txt
root@dd8464aebde7:/volumes/demoCA# echo "1000" > serial
root@dd8464aebde7:/volumes/demoCA# cd ..
```

Generating the certificate using the certificate authority's private key and certificate:

```
root@dd8464aebde7:/volumes# openssl ca -config myCA_openssl.cnf -policy policy_anything -md sha256 -
days 3650 -in dohabank.csr -out dohabank.crt -batch -cert ca.crt -keyfile ca.key
Using configuration from myCA_openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4099 (0x1003)
        Validity
            Not Before: Mar 19 18:42:47 2024 GMT
            Not After : Mar 17 18:42:47 2034 GMT
        Subject:
            countryName               = QA
            organizationName          = Doha Bank
            commonName                = qa.dohabank.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                C4:89:43:8F:9C:C8:C2:E1:A9:21:B3:B7:9F:97:38:F9:67:07:3C:08
            X509v3 Authority Key Identifier:
                keyid:89:19:3E:AF:5D:19:FA:8B:3E:30:FB:27:D0:3F:38:28:2C:DE:C0:EA

            X509v3 Subject Alternative Name:
                DNS:qa.dohabank.com, DNS:qa.dohabankA.com, DNS:qa.dohabankB.com
Certificate is to be certified until Mar 17 18:42:47 2034 GMT (3650 days)
```

Copying the foss_apache2_ssl.conf file to create another copy for the dohabank website and editing it's contents:

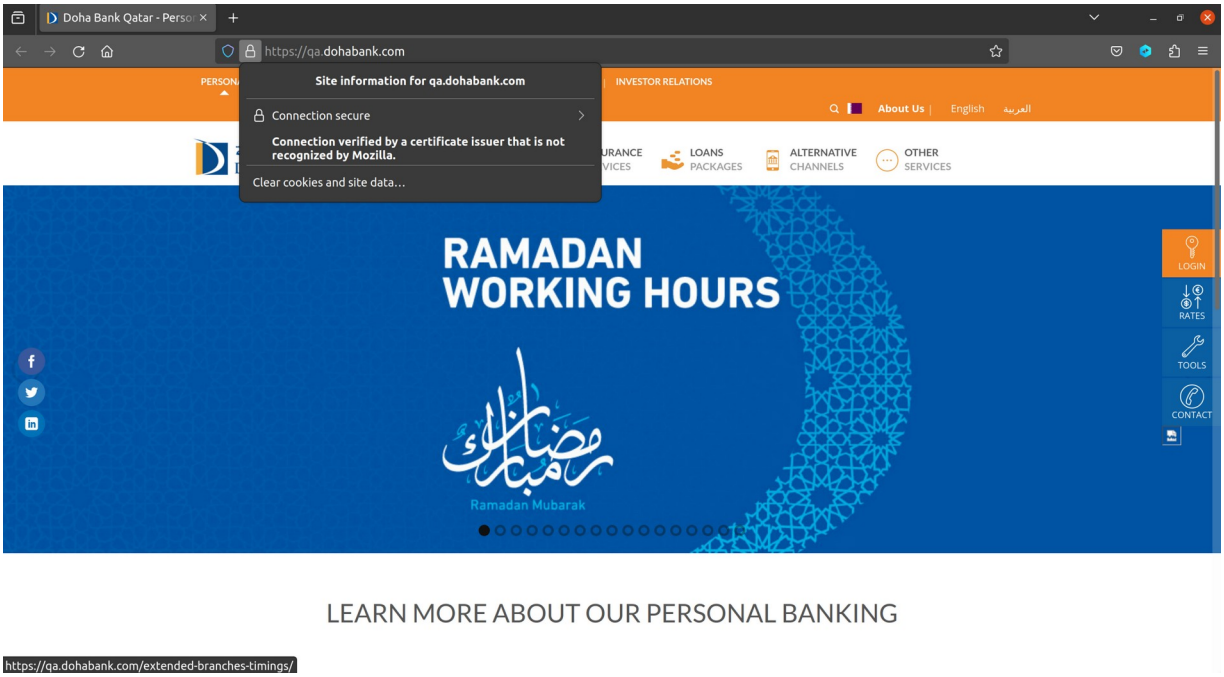
```
  GNU nano 4.8                          dohabank_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/dohabank
    ServerName qa.dohabank.com
    ServerAlias qa.dohabankA.com
    ServerAlias qa.dohabankB.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/dohabank.crt
    SSLCertificateKeyFile /certs/dohabank.key
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot /var/www/dohabank
    ServerName qa.dohabank.com
    DirectoryIndex index_red.html
</VirtualHost>

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

Then we make a dohabank directory under the /var/www and copy the html of the original dohabank website in the index.html file under this directory.

Continuing, we enable the website running from the configuration of the dohabank_apache_ssl.conf file and restart the apache2 service:

```
root@dd8464aebde7:/etc/apache2/sites-available# a2ensite dohabank_apache_ssl.conf
Enabling site dohabank_apache_ssl.
To activate the new configuration, you need to run:
  service apache2 reload
root@dd8464aebde7:/etc/apache2/sites-available# service apache2 reload
 * Reloading Apache httpd web server apache2
 *
root@dd8464aebde7:/etc/apache2/sites-available# service apache2 stop
 * Stopping Apache httpd web server apache2
 *
root@dd8464aebde7:/etc/apache2/sites-available# service apache2 start
 * Starting Apache httpd web server apache2
Enter passphrase for SSL/TLS keys for qa.dohabank.com:443 (RSA):
 *
root@dd8464aebde7:/etc/apache2/sites-available#
```

After that, we edit the /etc/hosts in our virtual machine to map qa.dohabank.com to the IP address of our docker container, and then we navigate to it:



As we can see, we see that we have connected securely to the dohabank's website however, we are connected to a malicious web server. This is possible because the certificate authority's private key and certificate had been compromised by the attacker.