

Public-Key Infrastructure (PKI) Lab

Jatin Kesnani
21K-3204 Student
FAST University

Abstract

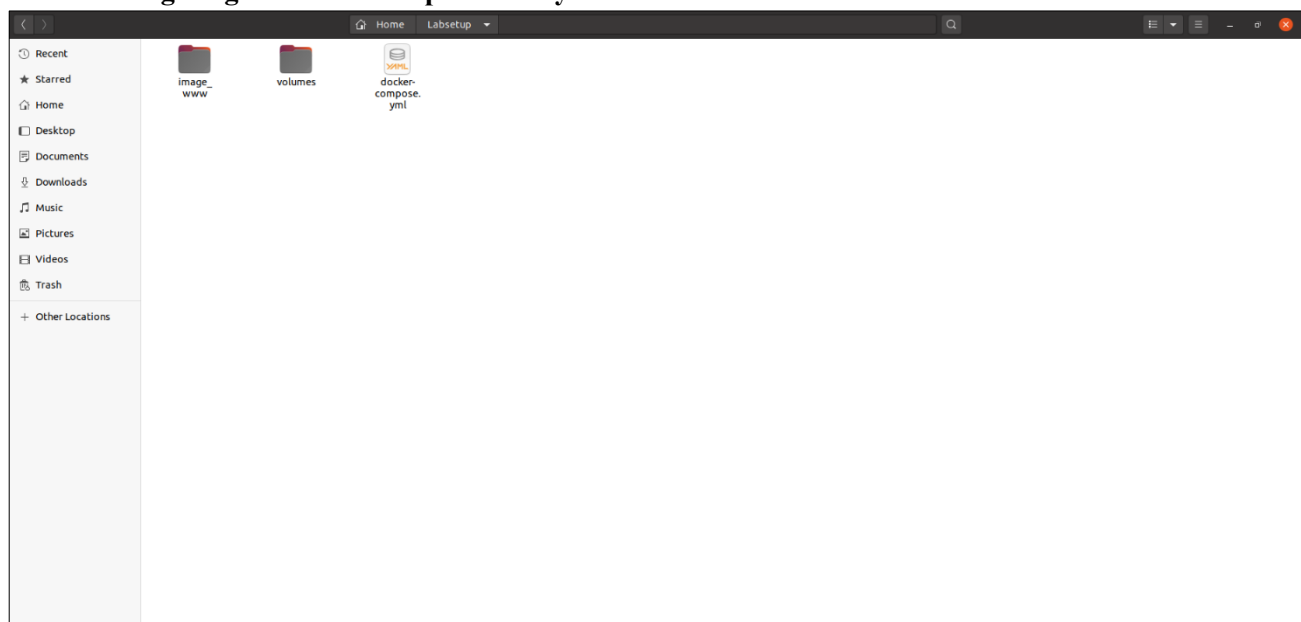
The Public Key Infrastructure (PKI) Lab aims to provide hands-on experience in understanding and implementing secure communications using public key cryptography. PKI serves as a critical solution for verifying the ownership of public keys and preventing man-in-the-middle attacks. This lab covers essential concepts like public-key encryption, X.509 certificates, and the role of Certificate Authorities (CAs) in establishing trust. By exploring these topics, students gain insights into the security mechanisms protecting web communications and the implications of a compromised root of trust.

1. Overview

The PKI Lab focuses on exploring the significance of public key cryptography as a foundation for secure communications. Public key cryptography, while secure, can still be vulnerable to man-in-the-middle attacks when verifying public key ownership. PKI addresses this issue by introducing a hierarchical structure involving CAs and X.509 certificates, providing a method for verifying the legitimacy of public keys. This lab explores these mechanisms in detail, including the use of root CAs, the role of Apache in configuring HTTPS, and the process of mitigating man-in-the-middle attacks. Through this lab, students develop an understanding of the importance of PKI and the risks associated with broken trust in this infrastructure.

2. Lab Environment

2.1 Navigating to the Labsetup Directory



2.2 Building the Docker Container

```
seed@VM: ~/Labsetup
[10/23/24]seed@VM:~/Labsetup$ docker-compose build
Building web-server
Step 1/7 : FROM handsonsecurity/seed-server:apache-php
--> 2365d0ed3ad9
Step 2/7 : ARG WWWDIR=/var/www/bank32
--> Using cache
--> ca67d3182e6f
Step 3/7 : COPY ./index.html ./index_red.html $WWWDIR/
--> Using cache
--> bc0d3035fda3
Step 4/7 : COPY ./bank32_apache_ssl.conf /etc/apache2/sites-available
--> Using cache
--> 293baef3bae0
Step 5/7 : COPY ./certs/bank32.crt ./certs/bank32.key /certs/
--> Using cache
--> f338fb7c3c50
Step 6/7 : RUN chmod 400 /certs/bank32.key && chmod 644 $WWWDIR/index.html && chmod 644 $WWWDIR/index_red.html && a2ensite ba
nk32_apache_ssl
--> Using cache
--> cd21dd0ceacf
Step 7/7 : CMD tail -f /dev/null
--> Using cache
--> ad2402b55894

Successfully built ad2402b55894
Successfully tagged seed-image-www-pki:latest
[10/23/24]seed@VM:~/Labsetup$
```

2.3 Starting the Docker Container

```
seed@VM: ~/Labsetup
[10/23/24]seed@VM:~/Labsetup$ docker-compose up
Creating network "net-10.9.0.0" with the default driver
WARNING: Found orphan containers (oracle-10.9.0.80) for this project. If you removed or renamed this service in your compose file, you can r
un this command with the --remove-orphans flag to clean it up.
Creating www-10.9.0.80 ... done
Attaching to www-10.9.0.80
```

2.4 Verifying the Container is Running

```
seed@VM: ~/Labsetup
[10/23/24]seed@VM:~/Labsetup$ docker ps --format "{{.ID}} {{.Names}}"
04bcf7bd2e98 www-10.9.0.80
[10/23/24]seed@VM:~/Labsetup$
```

2.5 Accessing the Container Shell

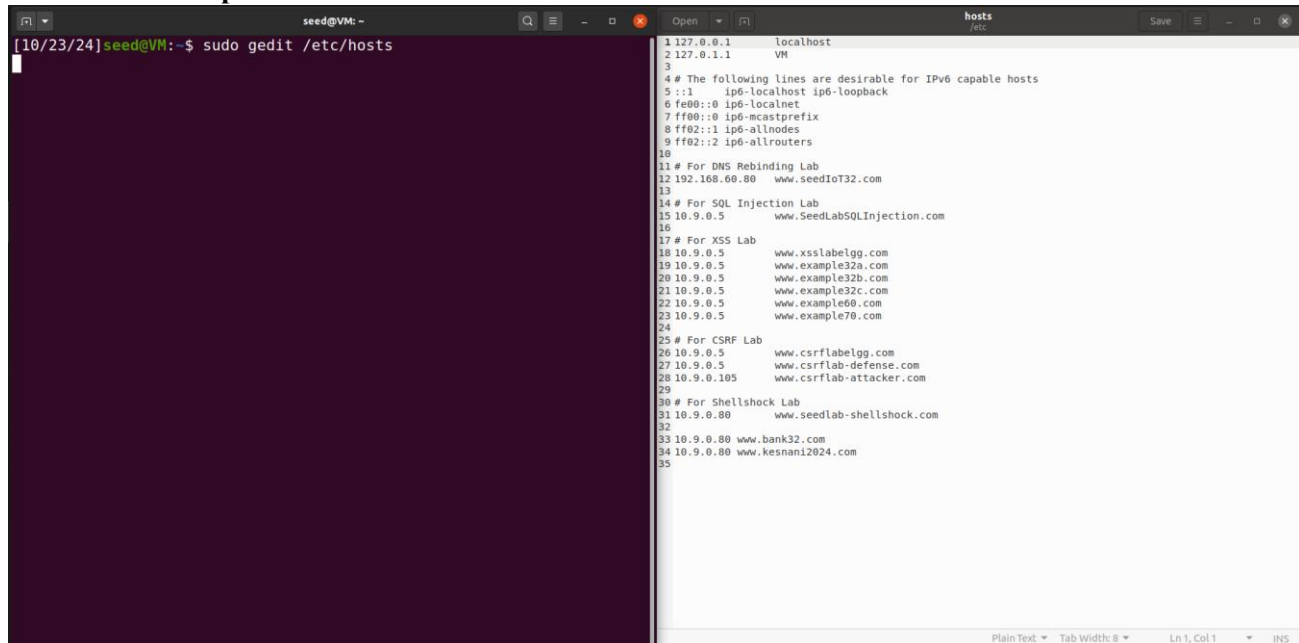
```
seed@VM: ~/Labsetup
[10/23/24]seed@VM:~/Labsetup$ docker ps --format "{{.ID}} {{.Names}}"
04bcf7bd2e98 www-10.9.0.80
[10/23/24]seed@VM:~/Labsetup$ docksh 04
root@04bcf7bd2e98:/# exit
exit
[10/23/24]seed@VM:~/Labsetup$
```

2.6 Shutting Down the Container

```
seed@VM: ~/Labsetup
[10/23/24]seed@VM:~/Labsetup$ docker-compose up
Creating network "net-10.9.0.0" with the default driver
WARNING: Found orphan containers (oracle-10.9.0.80) for this project.
If you removed or renamed this service in your compose file, you can r
un this command with the --remove-orphans flag to clean it up.
Creating www-10.9.0.80 ... done
Attaching to www-10.9.0.80
www-10.9.0.80 exited with code 137
[10/23/24]seed@VM:~/Labsetup$

seed@VM: ~/Labsetup
[10/23/24]seed@VM:~/Labsetup$ docker-compose down
Stopping www-10.9.0.80 ... done
WARNING: Found orphan containers (oracle-10.9.0.80) for this project.
If you removed or renamed this service in your compose file, you can r
un this command with the --remove-orphans flag to clean it up.
Removing www-10.9.0.80 ... done
Removing network net-10.9.0.0
[10/23/24]seed@VM:~/Labsetup$
```

2.7 DNS setup

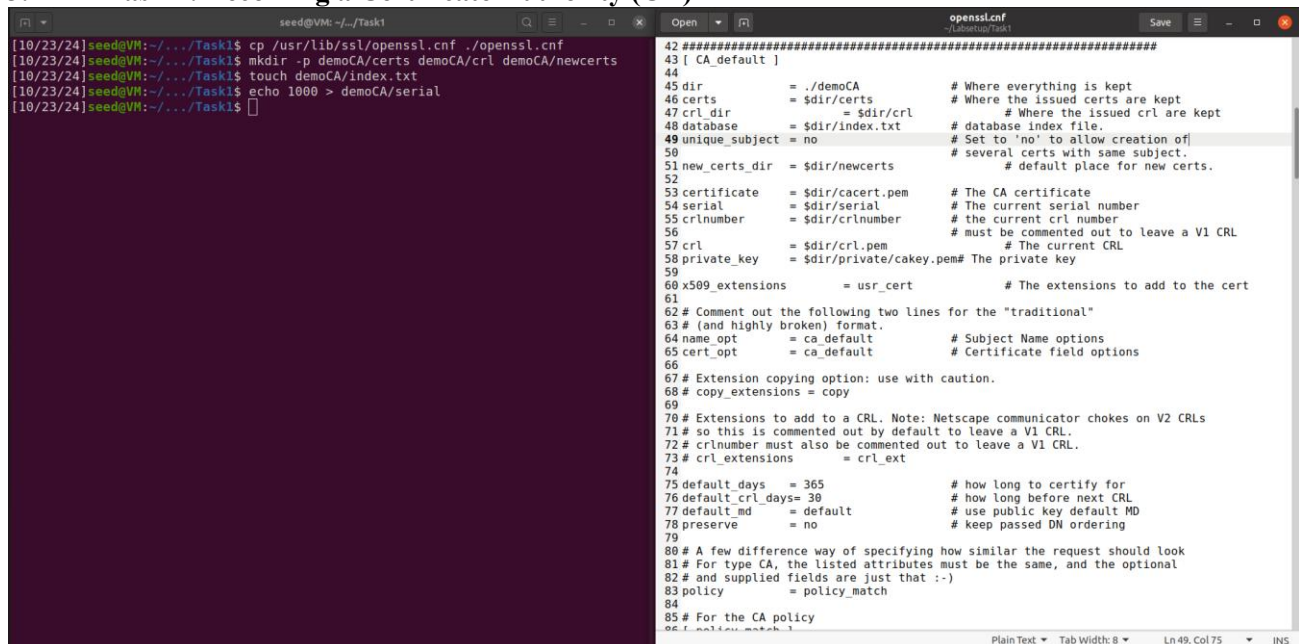


```
[10/23/24]seed@VM:~$ sudo gedit /etc/hosts

127.0.0.1 localhost
2 127.0.0.1 VM
3
4 # The following lines are desirable for IPv6 capable hosts
5 ::1 ip6-localhost ip6-loopback
6 fe80::0 ip6-localnet
7 ff00::0 ip6-mcastprefix
8 ff02::1 ip6-allnodes
9 ff02::2 ip6-allrouters
10
11 # For DNS Rebinding Lab
12 192.168.68.80 www.seedIoT32.com
13
14 # For SQL Injection Lab
15 10.9.0.5 www.SeedLabSQLInjection.com
16
17 # For XSS Lab
18 10.9.0.5 www.xsslabegg.com
19 10.9.0.5 www.example32a.com
20 10.9.0.5 www.example32b.com
21 10.9.0.5 www.example32c.com
22 10.9.0.5 www.example60.com
23 10.9.0.5 www.example70.com
24
25 # For CSRF Lab
26 10.9.0.5 www.csrflabelgg.com
27 10.9.0.5 www.csrfab-defense.com
28 10.9.0.105 www.csrfab-attacker.com
29
30 # For Shellshock Lab
31 10.9.0.80 www.seedlab-shellshock.com
32
33 10.9.0.80 www.bank32.com
34 10.9.0.80 www.kesnani2024.com
35
```

3. Lab Tasks

3.1 Task 1: Becoming a Certificate Authority (CA)



```
[10/23/24]seed@VM:~/Task1$ cp /usr/lib/ssl/openssl.cnf ./openssl.cnf
[10/23/24]seed@VM:~/Task1$ mkdir -p demoCA/certs demoCA/crl demoCA/newcerts
[10/23/24]seed@VM:~/Task1$ touch demoCA/index.txt
[10/23/24]seed@VM:~/Task1$ echo 1000 > demoCA/serial
[10/23/24]seed@VM:~/Task1$

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
59
60 x509_extensions    = usr_cert          # The extensions to add to the cert
61
62 # Comment out the following two lines for the "traditional"
63 # (and highly broken) format.
64 name_opt           = ca_default        # Subject Name options
65 cert_opt           = ca_default        # Certificate field options
66
67 # Extension copying option: use with caution.
68 # copy_extensions = copy
69
70 # Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
71 # so this is commented out by default to leave a V1 CRL.
72 # crlnumber must also be commented out to leave a V1 CRL.
73 # crl_extensions    = crl_ext
74
75 default_days       = 365               # how long to certify for
76 default_crl_days   = 30               # how long before next CRL
77 default_md          = default          # use public key default MD
78 preserve           = no               # keep passed DN ordering
79
80 # A few difference way of specifying how similar the request should look
81 # For type CA, the listed attributes must be the same, and the optional
82 # and supplied fields are just that :-))
83 policy              = policy_match
84
85 # For the CA policy
86 policy              = policy_match
```

In this task, I set up the environment to become a Certificate Authority (CA). I began by copying the OpenSSL configuration file (openssl.cnf) to my working directory and created the necessary directories for managing certificates and CRLs. Additionally, I prepared the index and serial files required by OpenSSL to track issued certificates, setting the initial serial number to 1000. This setup enables me to issue self-signed root CA certificates for subsequent tasks.

```
seed@VM: ~/Task1
[10/23/24]seed@VM:~/Task1$ 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]:PK
State or Province Name (full name) [Some-State]:Sindh
Locality Name (eg, city) []:Karachi
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FAST NUCES
Organizational Unit Name (eg, section) []:BSCS
Common Name (e.g. server FQDN or YOUR name) []:Jatin Kesnani
Email Address []:k213204@nu.edu.pk
[10/23/24]seed@VM:~/Task1$
```

```
seed@VM: ~/Task1
[10/23/24]seed@VM:~/Task1$ openssl x509 -in ca.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0a:e9:ef:62:8d:1d:ca:eb:2f:5d:25:62:c4:ac:fd:0c:64:4a:c7:e3
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = PK, ST = Sindh, L = Karachi, O = FAST NUCES, OU = BSCS, CN = J
        atin Kesnani, emailAddress = k213204@nu.edu.pk
        Validity
            Not Before: Oct 23 03:17:41 2024 GMT
            Not After : Oct 21 03:17:41 2034 GMT
        Subject: C = PK, ST = Sindh, L = Karachi, O = FAST NUCES, OU = BSCS, CN =
        Jatin Kesnani, emailAddress = k213204@nu.edu.pk
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public-Key: (4096 bit)
            Modulus:
                00:cb:89:37:9b:96:99:90:ea:fb:a5:eb:c3:18:7c:
                4a:1f:41:23:6b:0a:bf:cb:d5:6b:a5:71:43:97:46:
                74:a8:84:69:dd:a8:1c:bc:e7:ed:e1:db:c7:cc:cb:
                ef:d1:6c:89:fa:d7:02:fe:b5:73:18:2c:aa:da:df:
                fd:cf:79:95:6a:d9:ef:35:1a:6e:de:c0:b7:8f:5f:
                05:1a:e6:a2:59:2f:fd:82:2f:6d:56:b6:2e:1d:cf:
                62:89:8b:8d:f8:16:95:3b:9b:2a:2b:19:b2:a9:da:
                9a:c6:18:5c:ce:36:84:ed:c2:99:4b:e1:0e:70:15:
                4e:b5:2d:76:28:56:60:a3:0e:91:96:d5:a0:45:3d:
                c4:8b:80:82:ff:cb:83:22:ee:ff:ab:b3:0b:a9:24:
                19:0a:3c:d7:6e:e0:ab:95:0f:99:65:6e:3c:4e:02:
                0a:4d:66:b4:d1:a3:a1:23:53:5d:d6:5d:79:72:02:
                a0:b4:1b:39:7c:3d:df:b1:0a:78:0c:c8:ce:2e:13:
                ce:30:a2:93:f0:6a:26:c1:3f:54:ec:77:25:22:00:
                a7:6f:b8:54:84:c0:eb:49:2c:b3:f6:52:61:6b:85:
                1d:97:48:0c:dd:34:ec:be:6f:5f:e0:58:5a:25:04:
                87:52:52:bb:d2:14:7e:6b:51:43:67:9d:21:91:5f:
                42:b8:3b:92:47:27:1e:41:fe:47:5d:bd:9f:52:ec:
                bf:d4:42:5d:0e:ad:ab:9e:ef:4f:9d:8d:93:f6:c3:
                54:1b:c9:82:2e:da:70:95:4c:a7:5a:f3:b2:0e:9c:
                00:ca:81:d2:c9:70:52:74:e2:26:8c:07:40:07:50:
                af:77:99:70:b6:cc:d0:3b:b9:d1:5d:e4:fe:9d:0b:
                52:a1:41
            publicExponent: 65537 (0x10001)

seed@VM: ~/Task1
[10/23/24]seed@VM:~/Task1$ openssl rsa -in ca.key -text -noout
Enter pass phrase for ca.key:
RSA Private-Key: (4096 bit, 2 primes)
modulus:
    00:cb:89:37:9b:96:99:90:ea:fb:a5:eb:c3:18:7c:
    4a:1f:41:23:6b:0a:bf:cb:d5:6b:a5:71:43:97:46:
    74:a8:84:69:dd:a8:1c:bc:e7:ed:e1:db:c7:cc:cb:
    ef:d1:6c:89:fa:d7:02:fe:b5:73:18:2c:aa:da:df:
    fd:cf:79:95:6a:d9:ef:35:1a:6e:de:c0:b7:8f:5f:
    05:1a:e6:a2:59:2f:fd:82:2f:6d:56:b6:2e:1d:cf:
    62:89:8b:8d:f8:16:95:3b:9b:2a:2b:19:b2:a9:da:
    9a:c6:18:5c:ce:36:84:ed:c2:99:4b:e1:0e:70:15:
    4e:b5:2d:76:28:56:60:a3:0e:91:96:d5:a0:45:3d:
    c4:8b:80:82:ff:cb:83:22:ee:ff:ab:b3:0b:a9:24:
    19:0a:3c:d7:6e:e0:ab:95:0f:99:65:6e:3c:4e:02:
    0a:4d:66:b4:d1:a3:a1:23:53:5d:d6:5d:79:72:02:
    a0:b4:1b:39:7c:3d:df:b1:0a:78:0c:c8:ce:2e:13:
    ce:30:a2:93:f0:6a:26:c1:3f:54:ec:77:25:22:00:
    a7:6f:b8:54:84:c0:eb:49:2c:b3:f6:52:61:6b:85:
    1d:97:48:0c:dd:34:ec:be:6f:5f:e0:58:5a:25:04:
    87:52:52:bb:d2:14:7e:6b:51:43:67:9d:21:91:5f:
    42:b8:3b:92:47:27:1e:41:fe:47:5d:bd:9f:52:ec:
    bf:d4:42:5d:0e:ad:ab:9e:ef:4f:9d:8d:93:f6:c3:
    54:1b:c9:82:2e:da:70:95:4c:a7:5a:f3:b2:0e:9c:
    00:ca:81:d2:c9:70:52:74:e2:26:8c:07:40:07:50:
    af:77:99:70:b6:cc:d0:3b:b9:d1:5d:e4:fe:9d:0b:
    52:a1:41
publicExponent: 65537 (0x10001)
```

In this step, I generated a self-signed certificate for our Certificate Authority (CA). Using OpenSSL, I created a 4096-bit RSA key pair and a certificate valid for 10 years. The private key was stored in ca.key, and the public certificate was saved in ca.crt. To verify the content, I inspected the details of the CA certificate and private key using the openssl x509 and openssl rsa commands respectively, ensuring successful generation of the CA's credentials.

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

```
seed@VM: ~/Task1
X509v3 extensions:
X509v3 Subject Key Identifier:
AB:FC:58:86:07:DE:6A:A0:4B:3C:20:D4:5C:70:24:FB:15:C7:50:8F
X509v3 Authority Key Identifier:
keyid:AB:FC:58:86:07:DE:6A:A0:4B:3C:20:D4:5C:70:24:FB:15:C7:50:8F

X509v3 Basic Constraints: critical
CA:TRUE
Signature Algorithm: sha256WithRSAEncryption
57:8b:a8:74:e8:58:ed:98:e2:c2:d0:a1:07:05:51:35:2a:f3:
31:6b:01:d3:b3:aa:61:dd:56:16:60:00:3e:de:c4:bc:da:45:
a4:f1:b8:c4:cf:2b:fa:53:f3:95:fd:d1:1f:84:6e:1d:77:32:
74:a7:40:aa:28:54:5d:a4:1f:79:f1:7c:16:19:ad:72:a1:6a:
a9:78:a5:6f:06:aa:05:6d:55:3f:a6:e7:be:5f:53:44:f5:aa:
d9:fe:d1:14:c2:c6:37:78:b5:23:2f:3f:bd:af:2c:d3:e3:7a:
78:cd:1e:c4:0d:f4:30:69:53:48:a9:05:aa:47:61:7f:4f:5c:
db:7d:78:36:8e:df:a5:d9:b6:4e:c8:9f:b4:72:f6:00:5b:05:
e1:cc:b5:0a:dc:c5:99:c1:93:01:ab:ba:14:69:bc:71:30:a5:
29:c0:2a:88:20:0e:cb:ce:f9:48:a7:f2:ae:0b:72:18:e8:f9:
bb:2e:93:1a:ec:1b:65:e8:d1:74:fa:23:63:3f:de:43:b3:24:
2d:05:62:5e:42:fb:fc:54:59:60:05:09:f8:d4:c7:1d:a3:1c:
64:19:a1:76:e5:d9:f8:ec:3e:b8:be:9e:9f:cb:62:9e:82:5f:
f9:88:2d:aa:b8:3c:2e:18:1a:5a:66:30:0f:b4:e2:1a:ee:a9:
bd:6b:7c:3a:83:39:5e:02:ca:ef:1f:42:ef:52:b7:26:11:07:
6e:68:a7:e4:a1:58:15:db:2d:46:99:81:3d:8b:4f:3d:11:d9:
4a:21:0c:74:5b:8f:fe:ec:ae:b0:01:e9:31:ae:80:2d:08:cd:
c8:e5:b9:ff:cb:89:16:b2:46:44:6f:78:28:0b:8b:a3:7a:c0:
9d:d2:cc:fd:0a:ed:c9:75:16:6f:44:a8:52:fe:b3:9f:cd:ef:
a5:a8:50:6c:5d:5c:d7:a3:ef:a7:df:13:b6:bd:40:f6:d0:a2:
89:8f:c4:7b:70:74:19:49:2c:4d:16:ad:df:94:a8:5f:c2:b4:
0d:47:e4:3a:a5:0b:ca:a9:fb:87:1d:4a:16:90:23:06:08:
cf:ce:89:9e:e0:30:08:5e:bf:4f:d9:71:0a:41:60:aa:71:
e5:ea:02:7f:80:3a:8c:06:4f:8f:41:52:7c:04:56:3b:8d:db:
ff:be:58:63:68:bc:e2:94:2a:fc:68:88:2e:29:05:d4:f2:1c:
49:12:11:e9:e1:6a:d2:7a:56:bc:4e:c0:1a:0a:55:dc:0e:c7:
ee:2b:cb:f5:3e:74:b2:47:20:dc:8b:47:a7:d8:34:bf:5d:b7:
14:f6:03:a8:10:d1:34:01:8a:48:1b:74:60:53:13:77:74:8c:
9b:d9:2c:6c:19:2f:19:23
[10/23/24]seed@VM:~/Task1$
```

The CA:TRUE value specifies that this certificate belongs to a Certificate Authority (CA). The Basic Constraints extension defines whether a certificate can be used as a CA certificate or not. If this extension has CA:TRUE, the certificate can issue other certificates.

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

```
seed@VM: ~/Task1
[10/23/24]seed@VM:~/Task1$ openssl x509 -in ca.crt -text -noout
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
    0a:e9:ef:62:8d:1d:ca:eb:2f:5d:25:62:c4:ac:fd:0c:64:4a:c7:e3
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C = PK, ST = Sindh, L = Karachi, O = FAST NUCES, OU = BSCS, CN = Jatin Kesnani, emailAddress = k213204@nu.edu.pk
  Validity
    Not Before: Oct 23 03:17:41 2024 GMT
    Not After : Oct 21 03:17:41 2034 GMT
  Subject: C = PK, ST = Sindh, L = Karachi, O = FAST NUCES, OU = BSCS, CN = Jatin Kesnani, emailAddress = k213204@nu.edu.pk
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public-Key: (4096 bit)
    Modulus:
      00:cb:89:37:9b:96:99:90:ea:fb:a5:eb:c3:18:7c:
      4a:1f:41:23:6b:0a:bf:cb:d5:6b:a5:71:43:97:46:
      74:a8:84:69:dd:a8:1c:bc:e7:ed:e1:db:c7:cc:cb:
      ef:d1:6c:89:fa:d7:02:fe:b5:73:18:2c:aa:da:df:
      fd:cf:79:95:0a:d9:ef:35:1a:6e:de:c0:b7:8f:5f:
      05:1a:e6:a2:59:2f:fd:82:2f:6d:56:b6:2e:1d:cf:
      62:89:8b:8d:f8:16:95:3b:9b:2a:2b:19:b2:a9:da:
      9a:c6:18:5c:ce:36:84:ed:c2:99:4b:c1:0e:70:15:
      4e:b5:2d:76:28:56:60:a3:0e:91:96:d5:a0:45:3d:
      c4:8b:80:82:ff:cb:83:22:ee:ff:ab:b3:0b:a9:24:
      19:0a:3c:d7:6e:e0:ab:95:0f:99:65:6e:3c:4e:02:
      0a:4d:66:b4:d1:a3:a1:23:53:5d:d6:5d:79:72:02:
      a0:b4:1b:39:7c:3d:df:b1:0a:78:0c:c8:ce:2e:13:
      ce:30:a2:93:f0:6a:26:c1:3f:54:ce:77:25:22:00:
      a7:6f:b8:54:84:c0:eb:49:2c:b3:f6:52:61:6b:85:
      1d:97:48:0c:dd:34:ec:be:6f:5f:e0:58:5a:25:04:
      87:52:52:bb:d2:14:7e:6b:51:43:67:9d:21:91:5f:
      42:b8:3b:92:47:27:1e:41:fe:47:5d:bd:9f:52:ec:
      bf:d4:42:5d:0e:ad:ab:9e:ef:4f:9d:8d:93:f6:c3:
      54:1b:c9:82:2e:da:70:95:4c:a7:5a:f3:b2:0e:9c:
      00:ca:81:d2:c9:70:52:74:e2:26:8c:07:40:07:50:
      af:77:99:70:b6:cc:d0:3b:b9:d1:5d:e4:fe:9d:0b:
      83:f1:72:35:34:d9:64:81:0e:ce:5a:90:96:5f:13:
      8f:df:46:eb:c3:0a:87:fb:2d:51:8a:86:86:0c:54:
```

Since the Issuer and Subject are exactly the same, it indicates that this certificate was signed by itself, i.e., it is self-signed.

- 3) 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.

```
seed@VM: ~/Task1
Modulus:
00:cb:89:37:9b:96:99:90:ea:fb:a5:eb:c3:18:7c:
4a:1f:41:23:6b:0a:bf:cb:d5:6b:a5:71:43:97:46:
74:a8:84:69:dd:a8:1c:bc:e7:ed:e1:db:c7:cc:cb:
ef:d1:6c:89:fa:d7:02:fe:b5:73:18:2c:aa:da:df:
fd:cf:79:95:6a:d9:ef:35:1a:6e:de:c0:b7:8f:5f:
05:1a:e6:a2:59:2f:fd:82:2f:6d:56:b6:2e:1d:cf:
62:89:8b:8d:f8:16:95:3b:9b:2a:2b:19:b2:a9:da:
9a:c6:18:5c:ce:36:84:ed:c2:99:4b:e1:0e:70:15:
4e:b5:2d:76:28:56:60:a3:0e:91:96:d5:a0:45:3d:
c4:8b:80:82:ff:cb:83:22:ee:ff:ab:b3:0b:a9:24:
19:0a:3c:d7:6e:e0:ab:95:0f:99:65:6e:3c:4e:02:
0a:4d:66:b4:d1:a3:a1:23:53:5d:d6:5d:9f:72:02:
a0:b4:1b:39:7c:3d:df:b1:0a:78:0c:c8:ce:2e:13:
ce:30:a2:93:f0:6a:26:c1:3f:54:ec:77:25:22:00:
a7:6f:b8:54:84:c0:eb:49:2c:b3:f6:52:61:6b:85:
1d:77:48:0c:dd:34:ec:be:6f:5f:e0:58:5a:25:04:
87:52:52:bb:d2:14:7e:6b:51:43:67:9d:21:91:5f:
42:b8:3b:92:47:27:1e:41:fe:47:5d:bd:9f:52:ec:
bf:d4:42:5d:0e:ad:ab:9e:ef:4f:9d:8d:93:f6:c3:
54:1b:c9:82:2e:da:70:95:4c:a7:5a:f3:b2:0e:9c:
00:ca:81:d2:c9:70:52:74:e2:26:8c:07:40:07:50:
af:77:99:70:b6:cc:d0:3b:b9:d1:5d:e4:fe:9d:0b:
83:f1:72:35:34:d9:64:81:0e:ce:5a:90:96:5f:13:
8f:df:46:eb:c3:0a:87:fb:2d:51:8a:86:86:0c:54:
11:cc:83:f3:64:4c:35:2a:d3:68:ac:0a:e0:70:95:
c3:04:ed:5e:08:8b:fd:40:18:1f:20:c3:aa:20:f9:
e3:22:64:55:b5:85:ac:77:5e:fe:e6:6d:be:9b:ad:
c4:61:91:5e:dd:9b:f9:ee:54:43:07:27:ab:79:7d:
99:2e:e4:3e:d2:d0:73:56:48:63:ad:38:22:bd:84:
ca:c3:4b:16:da:54:59:eb:d6:e9:75:3f:af:bf:ff:
76:cc:1f:9b:23:a5:2c:c5:de:c9:b4:dd:1a:5f:64:
c0:09:94:68:bb:b3:19:e7:7a:c8:af:a0:9c:51:cd:
46:7d:1f:0c:15:dc:e7:de:97:73:74:ce:1c:c3:22:
fd:95:48:9c:4e:88:9d:46:08:7e:59:3e:da:c7:30:
52:a1:41
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Key Identifier:
AB:FF:58:86:07:DF:6A:A8:4B:3C:20:D4:5C:70:24:FB:15:C7:50:8F

seed@VM: ~/Task1
privateExponent:
00:b9:79:34:44:43:2b:73:84:be:87:07:5d:c4:8d:
56:5a:3e:d8:90:ad:bd:f3:78:6b:5c:da:e6:f6:0e:
4d:36:57:5b:c5:92:71:85:af:6f:f5:f6:7a:8e:e4:
74:88:89:f8:fe:ad:3a:5c:73:6b:0b:67:80:d6:6d:
71:73:c4:5e:e5:7c:ed:5f:9d:d3:d4:87:17:7e:bd:
df:00:11:95:75:e2:a6:88:20:c1:e9:57:a0:94:a7:
ac:2a:9d:12:65:35:e7:0d:e7:2a:b9:15:f4:da:95:
8b:9e:e3:0a:87:1b:e5:6c:68:7d:9b:48:de:08:43:
52:73:05:97:b7:d0:a2:c5:53:fc:55:d0:1b:f9:9c:
fc:e3:06:db:d9:4f:52:13:de:68:fb:c4:ef:e6:55:
2f:c0:45:8b:27:cd:f2:6d:a6:46:69:c8:d7:89:5a:
90:9a:d1:f2:0a:6b:aa:e6:6f:0c:be:d0:75:5f:f0:
87:13:39:0f:dc:68:e1:d4:0a:2f:35:af:33:3b:87:
ee:c8:21:0e:e0:f1:4a:99:a5:25:b5:4a:29:7b:5f:
cf:34:9e:6b:de:81:92:a7:1e:e6:6b:9b:cc:8c:d7:
44:0c:dc:aa:64:55:db:3a:fa:71:0b:16:58:27:0d:
86:66:fd:5f:8c:64:02:83:b7:0c:ce:79:c0:e2:21:
45:c0:ce:1e:aa:c5:d3:1b:dc:0b:b5:fb:67:6e:12:
98:02:80:b1:19:43:c3:61:69:bf:96:a1:34:af:b0:
24:60:92:c2:67:d2:b0:34:fc:4a:38:a9:c6:b4:48:
d2:89:24:f9:c1:ee:d7:d4:fd:19:f4:b4:99:b4:54:
ac:80:21:d7:a6:f5:4e:90:31:9b:fc:be:46:84:8e:
55:32:fc:cf:26:d1:de:fb:9f:2d:86:13:37:7c:05:
5d:ad:19:6f:07:1d:ae:b2:3c:a9:69:ce:e2:60:17:
40:26:52:72:5c:88:39:b6:fe:a8:b6:51:37:e0:64:
67:7e:98:af:74:b3:65:5d:f8:6a:e9:2b:8b:30:45:
d0:39:e8:22:63:65:44:11:84:a6:a7:0e:ba:20:39:
37:1d:60:5b:fa:a4:22:92:09:26:a9:47:76:7b:cb:
c2:a5:22:ee:4e:09:7b:15:d9:18:30:a7:38:f3:d4:
b4:ef:3a:54:7a:33:44:43:a9:de:08:73:ca:44:2d:
b7:67:32:18:2a:e4:c2:5b:81:fa:b6:23:7c:3e:96:
c8:62:46:60:74:ff:da:e8:24:3f:d3:3c:21:12:83:
74:94:07:36:31:2f:70:76:64:02:5e:3d:8d:5e:89:
ab:3e:ad:c8:0c:65:b1:65:61:90:05:0b:e2:2c:79:
71:fb:e5
primel:
00:f3:67:a1:66:07:83:43:18:b3:ef:e0:e0:1d:1b:
57:6a:0f:6b:a8:a6:40:1a:28:20:c6:f6:b9:43:d2:
e8:81:79:ef:c6:a6:cb:7c:79:c5:a8:b5:3d:89:1e:
12:bd:05:d4:4e:cb:5c:fe:7b:17:85:bf:31:d5:
99:32:b7:8c:36:f4:01:0a:66:2f:f2:a0:dd:c4:56:
7a:9c:94:91:7e:02:ea:c0:7b:40:3e:1f:a2:0a:65:
d4:f0:27:79:23:4b:2e:7a:54:a1:5b:c5:88:6a:3a:
cc:40:a6:7e:06:44:b2:19:bb:84:f3:82:04:ad:18:
69:35:42:39:90:a2:30:3e:4f:08:f6:a8:61:70:bc:
28:5b:04:7f:9f:fa:a7:00:27:1c:2f:cb:53:76:3d:
ce:f0:6f:72:bc:ed:8f:3a:ba:87:55:06:16:67:a9:
29:29:df:63:8e:af:cd:79:75:38:d4:00:09:b7:ac:
72:8e:05:da:7a:cb:18:54:72:09:el:98:b8:47:02:
6f:d4:35:11:f6:a8:62:24:a0:2c:ac:cf:a2:c0:d3:
b7:43:4c:66:30:f6:e0:d0:bc:f1:81:6d:62:4d:57:
c8:90:ea:80:df:f4:b4:70:d3:8f:1a:76:3b:90:f0:
b9:54:df:59:7b:3e:be:74:f1:86:e9:62:ae:fe:cc:
6c:e7
prime2:
00:d6:11:72:6f:f0:93:00:bf:3e:f2:64:81:48:55:
ee:58:c7:55:62:b6:f6:a7:a4:b6:e9:5d:44:79:1c:
f3:6a:d1:6c:d2:cf:d4:5d:5b:37:f0:62:b9:c6:6a:
f1:1b:c0:1b:51:55:37:14:e0:38:d9:8c:7e:18:83:
0b:65:6f:18:42:a6:9e:6f:04:1a:36:e2:e9:3e:96:
9a:c2:ab:ea:f1:6d:06:9e:e8:b5:55:d2:76:49:3e:
f2:cb:e4:ab:ed:1a:35:04:2d:78:bf:e9:7a:67:30:
03:9b:b1:01:2a:30:ed:a0:b9:af:d3:56:5b:ca:cc:
43:2a:74:e4:a7:d7:ba:3e:7b:f1:cb:3b:0b:2d:54:
c7:23:76:4a:a8:a2:5f:e1:f9:6c:fe:72:d8:3a:27:
5d:67:79:e3:32:15:48:58:77:94:5e:9e:45:2b:20:
ee:d3:97:df:42:11:50:37:43:49:9b:f0:97:ce:92:
94:c2:ed:f6:39:14:18:cd:4c:87:d5:ea:57:52:be:
70:08:3d:2e:fc:37:08:4e:31:d8:f4:b5:fa:8c:1a:
44:f4:df:6e:c5:1e:55:b3:60:c0:9e:ff:f5:d1:f9:
28:82:34:75:5e:0f:5b:bc:7a:00:ae:94:c5:11:f7:
b2:b9:53:2c:16:c7:7b:7d:03:82:ba:53:db:bc:a4:
d3:97
exponent1:
5d:58:17:a8:56:27:3b:9b:04:9e:70:a1:e1:e7:b3:
```

3.2 Task 2: Generating a Certificate Request for Your Web Server

```
seed@VM: ~/Task2
[10/24/24]seed@VM:~/Task2$ openssl req -newkey rsa:2048 -sha256 -keyout server.key -out server.csr -subj "/CN=www.kesnani2024.com/O=Kesnani2024 Inc./C=PK" -passout pass:dees
Generating a RSA private key
.....+++++
.+++++
writing new private key to 'server.key'
-----
[10/24/24]seed@VM:~/Task2$
```

```
seed@VM: ~/Task2
[10/24/24]seed@VM:~/Task2$ openssl req -in server.csr -text -noout
Certificate Request:
Data:
  Version: 1 (0x0)
  Subject: CN = www.kesnani2024.com, O = Kesnani2024 Inc., C = PK
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public-Key: (2048 bit)
    Modulus:
      00:d3:e4:31:e2:29:ce:b0:3d:3c:32:df:c4:7c:e3:
      66:15:e7:bd:44:77:96:cb:1d:a4:d9:44:0e:df:81:
      ce:d7:b4:6b:8f:55:64:45:a1:8f:f0:7e:dd:6f:cc:
      fa:b8:aa:3b:da:43:1e:05:6c:00:54:28:a6:12:d2:
      cc:6e:cb:fc:46:07:ee:87:48:95:48:07:14:46:
      19:8c:64:32:78:37:6e:d3:69:c2:0d:fa:8b:25:d6:
      7a:06:09:28:7d:0f:1b:14:8d:d9:2e:a0:61:5b:30:
      7e:c3:a4:1c:f4:3a:e9:fd:02:6c:84:60:09:74:78:
      cc:6e:cb:fc:46:07:ee:87:48:95:48:07:14:46:
      16:0f:25:b5:c9:28:f1:0e:45:79:f4:29:19:e8:aa:
      73:3d:28:ed:45:42:34:40:46:c8:cd:e7:44:6b:99:
      41:4b:4a:f0:7f:fe:2d:e3:32:c1:9e:e6:47:ac:f8:
      cd:fd:48:24:75:15:37:cd:b3:47:21:fc:21:5a:ff:
      88:f9:32:a3:94:44:7e:ea:f3:f3:be:ce:8e:a3:e2:
      5c:a2:2b:56:bc:be:75:ce:df:47:b3:67:eb:a1:e0:
      20:30:cd:90:51:bb:f9:4d:22:e7:32:8d:39:0b:e5:
      1b:33:24:b3:25:21:c7:c6:6c:cf:9f:89:58:3d:d6:
      bd:e1
    Exponent: 65537 (0x10001)
  Attributes:
    a0:00
  Signature Algorithm: sha256WithRSAEncryption
    ba:99:2c:34:50:e1:c3:4c:f2:14:a8:f2:f1:2c:32:ac:5e:0b:
    3c:2d:02:9c:4e:9b:55:04:4f:53:04:69:9c:91:cc:d6:6f:59:
    7f:c0:8b:1c:50:4a:d7:33:d4:bb:ff:64:93:c6:fa:34:66:51:
    7d:b5:fa:c1:75:ba:57:09:9b:31:5f:e7:d8:71:62:90:0b:2a:
    dc:01:2f:e3:78:de:44:8f:d9:11:38:29:cd:ce:93:9d:07:15:
    68:bd:4c:86:bc:b8:67:fd:e5:d8:83:8c:19:4c:af:1d:45:8a:
    58:a5:4a:6a:d8:18:81:ce:c3:82:2c:0c:8f:64:ba:48:b3:2e:
    6d:c9:3e:21:78:48:6a:c0:8c:e3:74:dc:9d:6d:e9:08:bd:8d:
    1e:bc:0f:32:82:3e:b3:ca:c5:1d:22:72:7e:5d:25:48:94:b3:

seed@VM: ~/Task2
[10/24/24]seed@VM:~/Task2$ openssl rsa -in server.key -text -noout
Enter pass phrase for server.key:
RSA Private-Key: (2048 bit, 2 primes)
modulus:
  00:d3:e4:31:e2:29:ce:b0:3d:3c:32:df:c4:7c:e3:
  66:15:e7:bd:44:77:96:cb:1d:a4:d9:44:0e:df:81:
  ce:d7:b4:6b:8f:55:64:45:a1:8f:f0:7e:dd:6f:cc:
  fa:b8:aa:3b:da:43:1e:05:6c:00:54:28:a6:12:d2:
  cc:6e:cb:fc:46:07:ee:87:48:95:48:07:14:46:
  19:8c:64:32:78:37:6e:d3:69:c2:0d:fa:8b:25:d6:
  7a:06:09:28:7d:0f:1b:14:8d:d9:2e:a0:61:5b:30:
  7e:c3:a4:1c:f4:3a:e9:fd:02:6c:84:60:09:74:78:
  cc:6e:cb:fc:46:07:ee:87:48:95:48:07:14:46:
  16:0f:25:b5:c9:28:f1:0e:45:79:f4:29:19:e8:aa:
  73:3d:28:ed:45:42:34:40:46:c8:cd:e7:44:6b:99:
  41:4b:4a:f0:7f:fe:2d:e3:32:c1:9e:e6:47:ac:f8:
  cd:fd:48:24:75:15:37:cd:b3:47:21:fc:21:5a:ff:
  88:f9:32:a3:94:44:7e:ea:f3:f3:be:ce:8e:a3:e2:
  5c:a2:2b:56:bc:be:75:ce:df:47:b3:67:eb:a1:e0:
  20:30:cd:90:51:bb:f9:4d:22:e7:32:8d:39:0b:e5:
  1b:33:24:b3:25:21:c7:c6:6c:cf:9f:89:58:3d:d6:
  bd:e1
publicExponent: 65537 (0x10001)
privateExponent:
  34:34:ae:01:ae:11:49:a4:dc:b7:20:20:d6:30:64:
  cf:92:d8:34:0d:4e:ee:de:6f:e6:43:f9:72:22:16:
  67:59:01:83:40:23:c3:70:29:f9:b4:18:34:da:89:
  0f:84:89:d8:9d:de:c0:b1:3a:67:81:d8:61:1c:e4:
  d7:8b:94:ad:60:78:e9:85:fd:99:e0:7d:36:06:8b:
  76:e6:9c:f4:b7:9b:ab:a5:0b:f5:cb:bc:0a:e7:8c:
  0a:fe:a8:a6:1f:59:a9:b7:c5:41:d2:ac:09:fe:a9:
  cc:e4:1b:6a:25:35:cf:6b:90:77:10:fe:63:5f:06:
  93:46:77:6e:2b:69:2e:36:d1:14:c1:34:0d:2a:45:
  82:10:9a:2a:bf:70:46:7e:c5:4a:6b:be:a3:7e:a2:
  5c:e0:77:46:8d:e2:37:76:35:d3:d9:90:36:84:82:
  68:aa:74:90:51:c5:b0:8c:f7:ee:d8:c8:af:74:84:
  e1:22:e0:52:9f:95:90:4a:14:a1:63:27:b4:02:d1:
  7d:7b:da:c4:4d:e6:c2:b4:3e:dd:15:2f:82:be:3e:
  c4:81:b9:00:cf:1a:77:cf:59:0d:c1:21:e1:f8:03:
  f5:0b:81:5d:c9:12:cd:b5:bf:69:6d:f6:f2:a1:57:
```

To generate a Certificate Signing Request (CSR) for my server, I used the `openssl req` command to create a private key and a CSR file. The command included subject information with `CN=www.kesnani2024.com`, `O=Kesnani2024 Inc.`, and `C=PK`. After generating the CSR, I examined its content using `openssl req -text`, and I also inspected the private key using `openssl rsa -text`.

```
seed@VM: ~/Task2
[10/24/24]seed@VM:~/Task2$ openssl req -newkey rsa:2048 -sha256 -keyout server.key -out server.csr -subj "/CN=www.kesnani2024.com/O=Kesnani2024 Inc./C=PK" -passout pass:dees -addext "subjectAltName = DNS:www.kesnani2024.com, DNS:www.kesnani2024A.com, DNS:www.kesnani2024B.com"
Generating a RSA private key
.....+++++
.+++++
writing new private key to 'server.key'
-----
[10/24/24]seed@VM:~/Task2$
```

To generate a Certificate Signing Request (CSR) with alternative names, I used the `openssl req` command. The CSR included subject information like `CN=www.kesnani2024.com`, `O=Kesnani2024 Inc.`, and `C=PK`, and I specified additional domain names using the `-addext` option for the Subject Alternative Name (SAN) field. This allowed the certificate to include `DNS:www.kesnani2024.com`, `DNS:www.kesnani2024A.com`, and `DNS:www.kesnani2024B.com`. These alternative names are essential for matching various hostnames with the server certificate.

3.3 Task 3: Generating a Certificate for your server

```
seed@VM: ~/Task3
[10/24/24]seed@VM:~/Task3$ openssl ca -config openssl.cnf -policy policy_anything -md sha256 -days 3650 -in server.csr -out server.crt -batch -cert ca.crt -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: 4096 (0x1000)
  Validity
    Not Before: Oct 23 22:09:32 2024 GMT
    Not After : Oct 21 22:09:32 2034 GMT
  Subject:
    countryName      = PK
    organizationName = Kesnani2024 Inc.
    commonName       = www.kesnani2024.com
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    Netscape Comment:
      OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
      03:6E:C1:F9:1D:6F:56:01:5C:A6:56:37:ED:57:52:E2:1A:0A:99:D6
    X509v3 Authority Key Identifier:
      keyid:AB:FC:58:06:07:DE:6A:A0:4B:3C:20:D4:5C:70:24:FB:15:C7:50:8F

    X509v3 Subject Alternative Name:
      DNS:www.kesnani2024.com, DNS:www.kesnani2024A.com, DNS:www.kesnani2024B.com
Certificate is to be certified until Oct 21 22:09:32 2034 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
[10/24/24]seed@VM:~/Task3$
```

```
seed@VM: ~/Task3
[10/24/24]seed@VM:~/Task3$ openssl x509 -in server.crt -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 4096 (0x1000)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = PK, ST = Sindh, L = Karachi, O = FAST NUCES, OU = BSCS, CN = Jatin Kesnani, emailAddress = k213204@nu.edu.pk
    Validity
      Not Before: Oct 23 22:09:32 2024 GMT
      Not After : Oct 21 22:09:32 2034 GMT
    Subject: C = PK, O = Kesnani2024 Inc., CN = www.kesnani2024.com
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:9a:ee:4a:38:9c:4b:bf:71:87:62:24:07:64:a7:
        f8:3a:1f:c0:4c:eb:23:70:bd:8e:65:0e:96:8d:2d:
        d5:e2:a8:28:f6:db:e9:68:0c:bf:be:4b:df:f0:bd:
        6a:b7:77:b5:d3:10:54:66:43:2f:e5:64:06:58:a2:
        8c:f7:61:23:a8:34:b1:16:36:07:7b:08:56:9b:01:
        1b:16:ca:8e:0c:ae:af:3e:36:8a:8d:87:9e:03:10:
        04:d6:ea:73:7e:14:8f:b5:44:f1:04:fa:5b:bd:ae:
        55:e5:6d:24:09:2d:40:f1:35:8a:72:1e:28:cd:8c:
        e0:8a:1a:ef:59:0c:1a:7a:1d:3c:30:1e:ff:07:f9:
        09:3e:ff:d9:05:ef:6a:08:05:3d:66:bb:5d:eb:a8:
        01:68:a3:c1:c9:b1:c4:a7:d0:da:75:63:0b:af:56:
        be:89:59:63:93:ff:aa:6c:12:59:0f:20:d5:91:aa:
        19:d7:55:07:88:56:0b:b3:0b:04:b0:0e:c1:14:0e:
        e6:9b:1d:4c:b7:f7:ea:b0:23:55:a8:95:6a:d1:90:
        ffe4:46:5d:0a:91:0b:43:bf:32:88:be:2f:be:00:
        91:6c:e3:c4:38:91:e1:5d:09:58:93:bf:5b:d9:3d:
        1b:0b:3c:20:2d:0a:8f:df:00:02:1e:d2:03:88:f7:
        10:1b
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:FALSE
      Netscape Comment:
        OpenSSL Generated Certificate
      X509v3 Subject Key Identifier:
        03:6E:C1:F9:1D:6F:56:01:5C:A6:56:37:ED:57:52:E2:1A:0A:99:D6
      X509v3 Authority Key Identifier:
        keyid:AB:FC:58:06:07:DE:6A:A0:4B:3C:20:D4:5C:70:24:FB:15:C7:50:8F

      X509v3 Subject Alternative Name:
        DNS:www.kesnani2024.com, DNS:www.kesnani2024A.com, DNS:www.kesnani2024B.com
    Signature Algorithm: sha256WithRSAEncryption
    4b:68:e1:ef:b0:a7:37:2d:7a:a6:35:1a:d5:bb:41:93:24:c6:
    3a:09:c4:b7:9d:c1:d1:7c:7a:1b:5f:c1:36:87:24:95:4f:16:
```

Using the command `openssl ca -config openssl.cnf -policy policy_anything -in server.csr -out server.crt -batch -cert ca.crt -keyfile ca.key`, we utilized our self-created Certificate Authority (CA) to issue the certificate. This command applied the `policy_anything` to allow flexible subject information and included our CA's credentials from `ca.crt` and `ca.key`. After generating the certificate, we confirmed its details, including the Subject Alternative Names, with the command `openssl x509 -in server.crt -text -noout`.

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

```
seed@VM: ~/Labsetup
[10/27/24]seed@VM:~/Labsetup$ dcup
Starting www-10.9.0.80 ... done
Attaching to www-10.9.0.80

root@0f80e3ece29b: /
[10/27/24]seed@VM:~/Labsetup$ dckps
0f80e3ece29b www-10.9.0.80
[10/27/24]seed@VM:~/Labsetup$ docksh 0f
root@0f80e3ece29b:/# ls volumes/
README.md ca.crt ca.key index.html index_red.html server.crt server.key
root@0f80e3ece29b:/# cp volumes/ca.crt certs
root@0f80e3ece29b:/# cp volumes/ca.key certs
root@0f80e3ece29b:/# cp volumes/server.key certs
root@0f80e3ece29b:/# cp volumes/server.crt certs
root@0f80e3ece29b:/# ls certs/
bank32.crt bank32.key ca.crt ca.key server.crt server.key
root@0f80e3ece29b:/# ls var/www/kesnani2024/
index.html index_red.html
root@0f80e3ece29b:/# ls etc/apache2/sites-available/
000-default.conf bank32_apache_ssl.conf default-ssl.conf kesnani_apache_ssl.conf
root@0f80e3ece29b:/# cat etc/apache2/sites-available/kesnani_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/kesnani2024
    ServerName www.kesnani2024.com
    ServerAlias www.kesnani2024A.com
    ServerAlias www.kesnani2024B.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>
root@0f80e3ece29b:/#
```

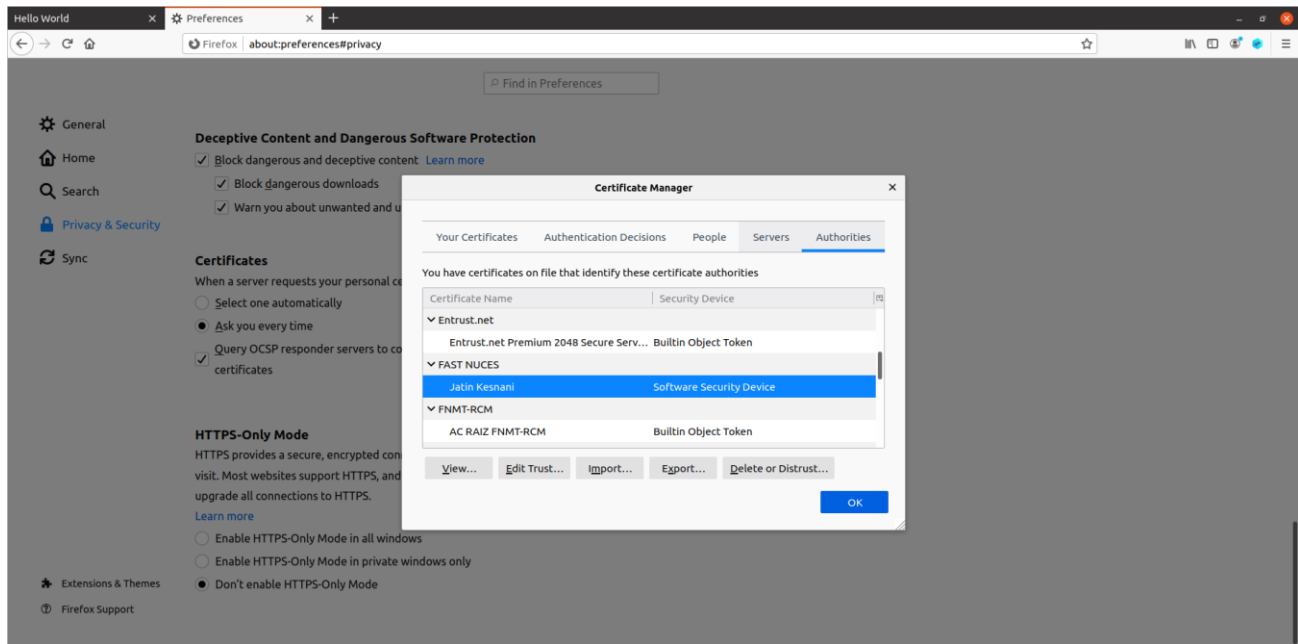
I copied necessary files, including the website's HTML files and SSL certificate files, into the **/volumes** directory within the Apache container. Specifically, I ensured that the SSL certificate and key were located in the **/certs** directory, while the website files were placed in **/var/www/kesnani2024/**.

I configured the Apache VirtualHost settings in the **kesnani_apache_ssl.conf** file to define the document root and SSL configurations. The configuration included specifying the server name, server aliases, enabling the SSL engine, and pointing to the SSL certificate and key files.

```
<VirtualHost *:443>
    DocumentRoot /var/www/kesnani2024
    ServerName www.kesnani2024.com
    ServerAlias www.kesnani2024A.com
    ServerAlias www.kesnani2024B.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>
```

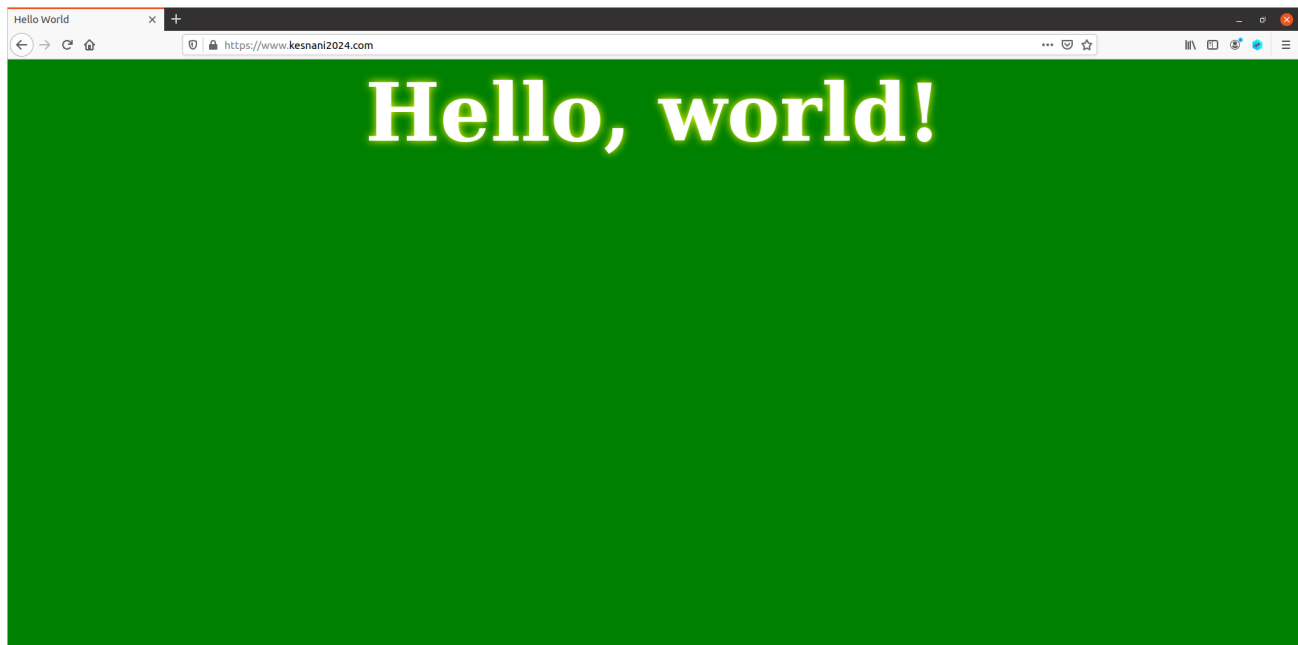
```
root@0f80e3ece29b:/etc/apache2/sites-available#
root@0f80e3ece29b:/etc/apache2/sites-available# cd etc/apache2/sites-available/
root@0f80e3ece29b:/etc/apache2/sites-available# a2ensite kesnani_apache_ssl.conf
Site kesnani_apache_ssl already enabled
root@0f80e3ece29b:/etc/apache2/sites-available# service apache2 start
* Starting Apache httpd web server apache2
*
root@0f80e3ece29b:/etc/apache2/sites-available#
```

After building the Docker image and running the container, I enabled the site configuration using **a2ensite** and started the Apache server. This allowed Apache to serve the HTTPS site on port 443.



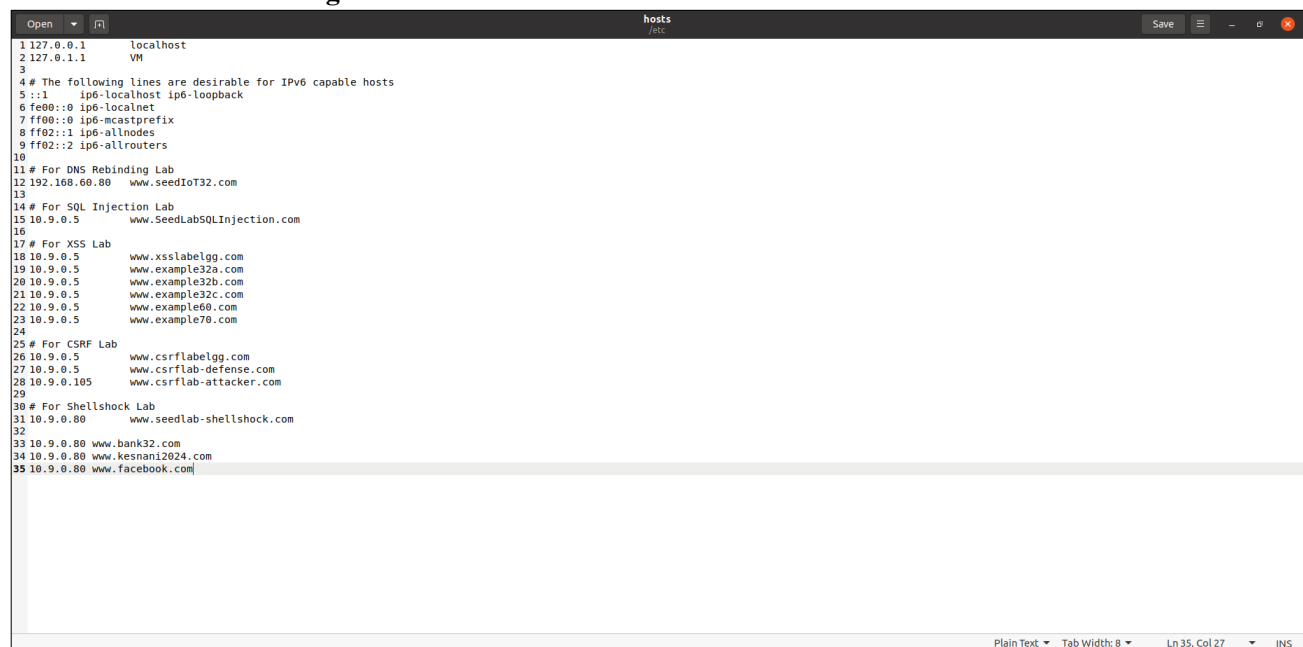
After setting up the HTTPS server, I attempted to browse the website using the URL **"https://www.kesnani2024.com"**. Initially, the browser displayed a security warning and prevented access due to the server's certificate not being recognized by the browser. This occurred because the certificate was not issued by a trusted certificate authority (CA).

To fix this, I imported the **"ca.crt"** certificate into Firefox. I navigated to **"about:preferences#privacy"** in Firefox's address bar and clicked the **"View Certificates"** button. Under the **"Authorities"** tab, I chose to import the **"ca.crt"** certificate and selected the option to **"Trust this CA to identify websites"**.



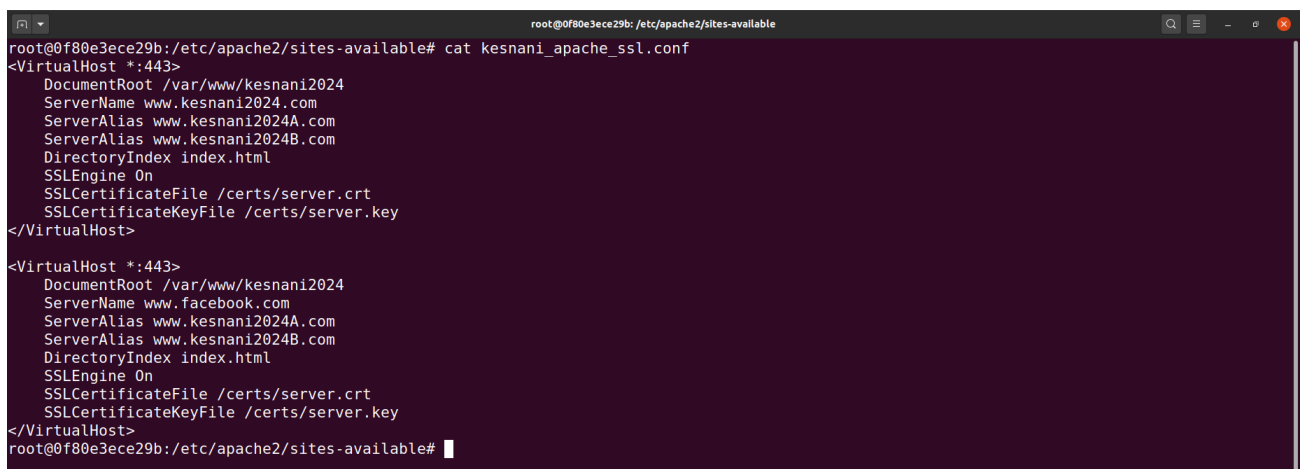
After importing the certificate successfully, I was able to access the website securely without receiving any warnings. This demonstrated the successful configuration of my HTTPS server using the self-signed certificate.

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



```
Open hosts
/etc/
1 127.0.0.1 localhost
2 127.0.1.1 VM
3
4 # The following lines are desirable for IPv6 capable hosts
5 ::1 ip6-localhost ip6-loopback
6 fe00::0 ip6-localnet
7 ff00::0 ip6-mcastprefix
8 ff02::1 ip6-allnodes
9 ff02::2 ip6-allrouters
10
11 # For DNS Rebinding Lab
12 192.168.60.80 www.seedtoT32.com
13
14 # For SQL Injection Lab
15 10.9.0.5 www.SeedLabSQLInjection.com
16
17 # For XSS Lab
18 10.9.0.5 www.xsslabelgg.com
19 10.9.0.5 www.example32a.com
20 10.9.0.5 www.example32b.com
21 10.9.0.5 www.example32c.com
22 10.9.0.5 www.example60.com
23 10.9.0.5 www.example70.com
24
25 # For CSRF Lab
26 10.9.0.5 www.csrflabelgg.com
27 10.9.0.5 www.csrfab-defense.com
28 10.9.0.105 www.csrfab-attacker.com
29
30 # For Shellshock Lab
31 10.9.0.80 www.seedlab-shellshock.com
32
33 10.9.0.80 www.bank32.com
34 10.9.0.80 www.kesnani2024.com
35 10.9.0.80 www.facebook.com
```

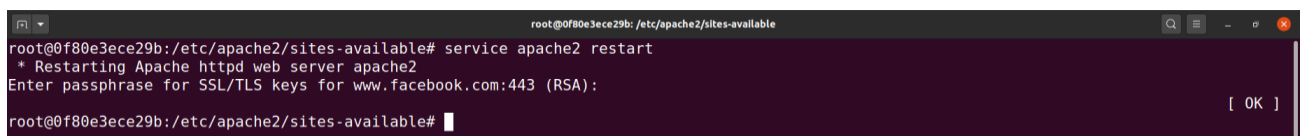
I have targeted the website **www.facebook.com**. I added an entry for the target website in the **/etc/hosts** file. This entry redirected the domain name to the attacker's server.



```
root@0f80e3ece29b: /etc/apache2/sites-available
root@0f80e3ece29b:/etc/apache2/sites-available# cat kesnani_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/kesnani2024
    ServerName www.kesnani2024.com
    ServerAlias www.kesnani2024A.com
    ServerAlias www.kesnani2024B.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>

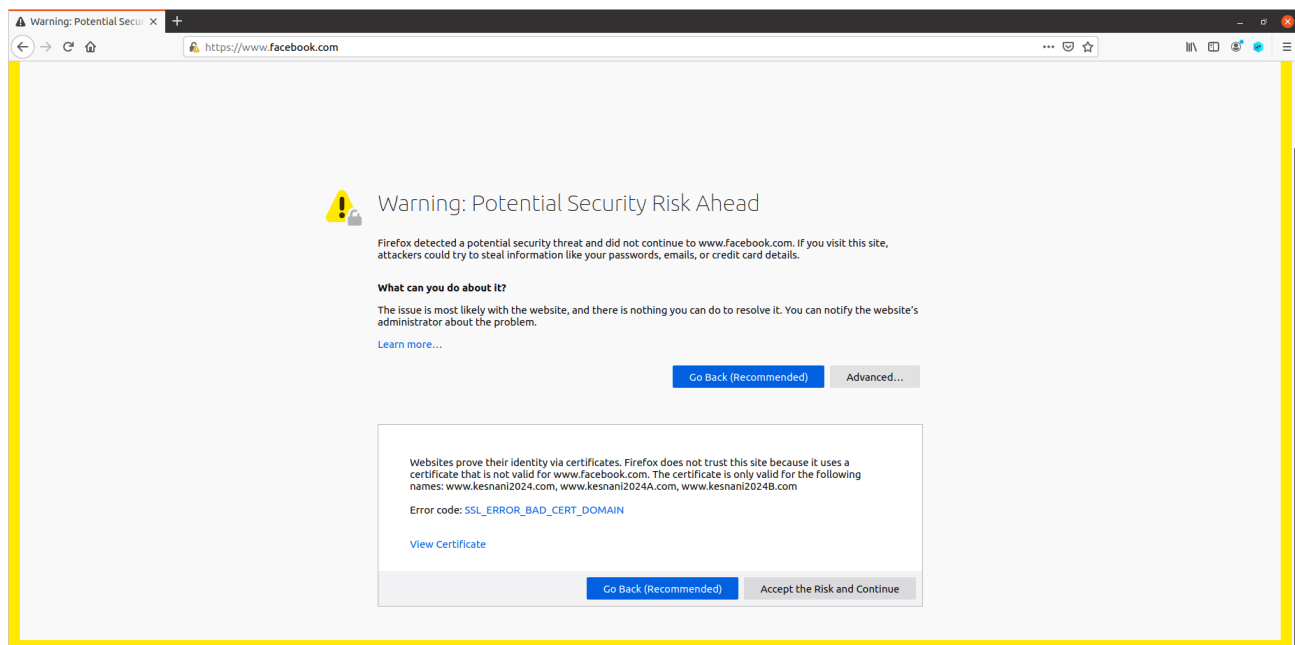
<VirtualHost *:443>
    DocumentRoot /var/www/kesnani2024
    ServerName www.facebook.com
    ServerAlias www.kesnani2024A.com
    ServerAlias www.kesnani2024B.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>
root@0f80e3ece29b:/etc/apache2/sites-available#
```

Next, I configured the Apache server by adding a VirtualHost entry for the malicious website in the **/etc/apache2/sites-available/kesnani_apache_ssl.conf** file. This entry allowed Apache to serve a website for the spoofed domain. Since I can't obtain a legitimate certificate for the target site, I will use the same certificate previously set up for my malicious server.



```
root@0f80e3ece29b: /etc/apache2/sites-available
root@0f80e3ece29b:/etc/apache2/sites-available# service apache2 restart
* Restarting Apache httpd web server apache2
Enter passphrase for SSL/TLS keys for www.facebook.com:443 (RSA):
root@0f80e3ece29b:/etc/apache2/sites-available# [ OK ]
```

I restarted the Apache server with **service apache2 restart** so as to commit the changes made.



When I accessed the website **www.facebook.com**, the request will be directed to my malicious server. The browser received a "**Warning: Potential Security Risk Ahead**" error, the error is caused by an invalid security certificate being used, with the specific error code of **SSL_ERROR_BAD_CERT_DOMAIN**. It indicates that the certificate presented by my server (**www.kesnani2024.com**) does not match the expected certificate of the legitimate website (**www.facebook.com**). This alert is due to the browser verifying the certificate's authenticity, which is a core security feature of PKI.

Observations: When accessing the target website, I observed a "connection not secure" error in the browser. This happened because the browser detected a mismatch between the presented certificate and the expected certificate of **www.facebook.com**. In this emulation, I used a certificate for the site **www.kesnani2024.com** to impersonate **www.facebook.com**. This is why PKI is crucial—it validates the authenticity of certificates and prevents attackers from successfully impersonating legitimate servers.

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

```
seed@VM: ~/Labsetup
[10/27/24]seed@VM:~/Labsetup$ openssl req -new -key server.key -out facebook.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]:PK
State or Province Name (full name) [Some-State]:Sindh
Locality Name (eg, city) []:Karachi
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FAST NUCES
Organizational Unit Name (eg, section) []:BSCS
Common Name (e.g. server FQDN or YOUR name) []:www.facebook.com
Email Address []:k213204@nu.edu.pk

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:dees
An optional company name []:
[10/27/24]seed@VM:~/Labsetup$
```

I started copying the existing private key (server.key) to create a file named **facebook.key**. Using this private key, I generated a Certificate Signing Request (**facebook.csr**)

```
seed@VM: ~/Labsetup
[10/27/24]seed@VM:~/Labsetup$ openssl ca -in facebook.csr -out facebook.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: Oct 27 00:15:24 2024 GMT
        Not After : Oct 27 00:15:24 2025 GMT
    Subject:
        countryName           = PK
        stateOrProvinceName   = Sindh
        organizationName      = FAST NUCES
        organizationalUnitName = BSCS
        commonName            = www.facebook.com
        emailAddress          = k213204@nu.edu.pk
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        Netscape Comment:
            OpenSSL Generated Certificate
        X509v3 Subject Key Identifier:
            26:A0:A1:C0:CF:1B:66:54:7C:FC:EB:DC:D0:CF:27:94:58:EF:D8:E5
        X509v3 Authority Key Identifier:
            keyid:BF:62:F7:00:26:1E:07:F9:1E:1D:CA:E5:D7:70:85:FF:2D:2C:40

Certificate is to be certified until Oct 27 00:15:24 2025 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]
Write out database with 1 new entries
Data Base Updated
[10/27/24]seed@VM:~/Labsetup$
```

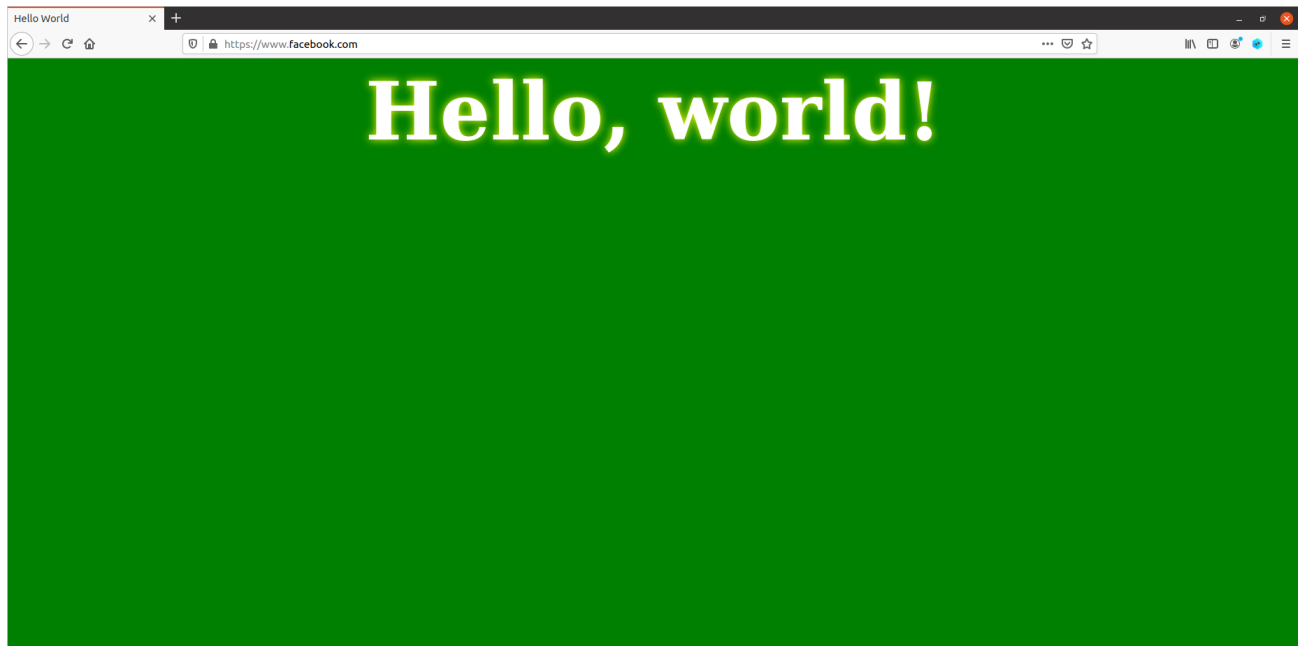
Using the private key of the compromised CA from Task 1, I will generate a fake certificate for **www.facebook.com**. Since the attacker now has access to the CA's private key, the attacker can issue a certificate that appears to be signed by a trusted CA.


```
root@0f80e3ece29b:/# ls volumes/
README.md  ca.key      facebook.key  index_red.html  server.key
ca.crt     facebook.crt  index.html    server.crt
root@0f80e3ece29b:/# cp volumes/facebook.crt certs
root@0f80e3ece29b:/# cp volumes/facebook.key certs
root@0f80e3ece29b:/# ls certs/
bank32.crt  ca.crt  facebook.crt  server.crt
bank32.key  ca.key  facebook.key  server.key
root@0f80e3ece29b:/# cat etc/apache2/sites-available/kesnani_apache_ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/kesnani2024
    ServerName www.kesnani2024.com
    ServerAlias www.kesnani2024A.com
    ServerAlias www.kesnani2024B.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/server.crt
    SSLCertificateKeyFile /certs/server.key
</VirtualHost>

<VirtualHost *:443>
    DocumentRoot /var/www/kesnani2024
    ServerName www.facebook.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /certs/facebook.crt
    SSLCertificateKeyFile /certs/facebook.key
</VirtualHost>
root@0f80e3ece29b:/#
```

Inside the container, I listed the contents of the **volumes/** directory and copied the generated **facebook.key** and **facebook.crt** files to the **certs/** directory. I also configured my malicious Apache server to use this fake certificate. Next, I manually added the compromised CA certificate (**ca.crt**) to the browser's trusted certificates.

```
root@0f80e3ece29b:/etc/apache2/sites-available
root@0f80e3ece29b:/etc/apache2/sites-available# cd etc/apache2/sites-available/
root@0f80e3ece29b:/etc/apache2/sites-available# a2ensite kesnani_apache_ssl.conf
Site kesnani_apache_ssl already enabled
root@0f80e3ece29b:/etc/apache2/sites-available# service apache2 start
* Starting Apache httpd web server apache2
*
root@0f80e3ece29b:/etc/apache2/sites-available#
```



With the setup complete, I attempt to access **www.facebook.com** in the browser. Since the fake certificate is signed by the compromised CA, and the browser has been configured to trust this CA, the browser will not show any security warnings. The victim is unaware that they are on a fake website.

Observations: When visiting www.facebook.com, I observed that the browser did not raise any warnings. The HTTPS connection appeared secure because the certificate was issued by a CA that the browser trusted. This demonstrates the danger of a compromised CA, as it enables an attacker to impersonate any website and perform a successful MITM attack without alerting the victim.