

PKI Lab Seed Lab

Task 1: Becoming a CA Authority

First we have **cd** to the **Labsetup** folder. Then we have to make a directory called demoCA and its subdirectory. We also copy the **openssl.cnf** file from **/usr/lib/ssl** to the **LabSetup** file. This is the open ssl configuration file which is needed for the following tasks.

In this picture below we do all the necessary setups needed to do task 1 according to the question material below.

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



```
seed@seed-virtual-machine:~/src-cloud/Labsetup$ ls
docker-compose.yml  image_www  volumes
seed@seed-virtual-machine:~/src-cloud/Labsetup$ cp /usr/lib/ssl/openssl.cnf .
seed@seed-virtual-machine:~/src-cloud/Labsetup$ ls
docker-compose.yml  image_www  openssl.cnf  volumes
seed@seed-virtual-machine:~/src-cloud/Labsetup$ mkdir demoCA
seed@seed-virtual-machine:~/src-cloud/Labsetup$ cd demoCA/
seed@seed-virtual-machine:~/src-cloud/Labsetup/demoCA$ mkdir certs crl newcerts
seed@seed-virtual-machine:~/src-cloud/Labsetup/demoCA$ ls
certs  crl  newcerts
seed@seed-virtual-machine:~/src-cloud/Labsetup/demoCA$ touch index.txt
seed@seed-virtual-machine:~/src-cloud/Labsetup/demoCA$ touch serial
seed@seed-virtual-machine:~/src-cloud/Labsetup/demoCA$ ls
certs  crl  index.txt  newcerts  serial
seed@seed-virtual-machine:~/src-cloud/Labsetup/demoCA$ cd ..
seed@seed-virtual-machine:~/src-cloud/Labsetup$ ls
demoCA  docker-compose.yml  image_www  openssl.cnf  volumes
seed@seed-virtual-machine:~/src-cloud/Labsetup$ █
```

In **index.txt** we leave it empty and for the file string we store any value
I ran the command **echo 01 > serial** to store the value 01 into serial.

Now we need to generate a self signed certificate for our CA. we do so by running the following command

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 \ -keyout ca.key -out ca.crt
```

We will be asked to enter some details for which I have given the information below

Password: joel

Country: US

State: Virginia

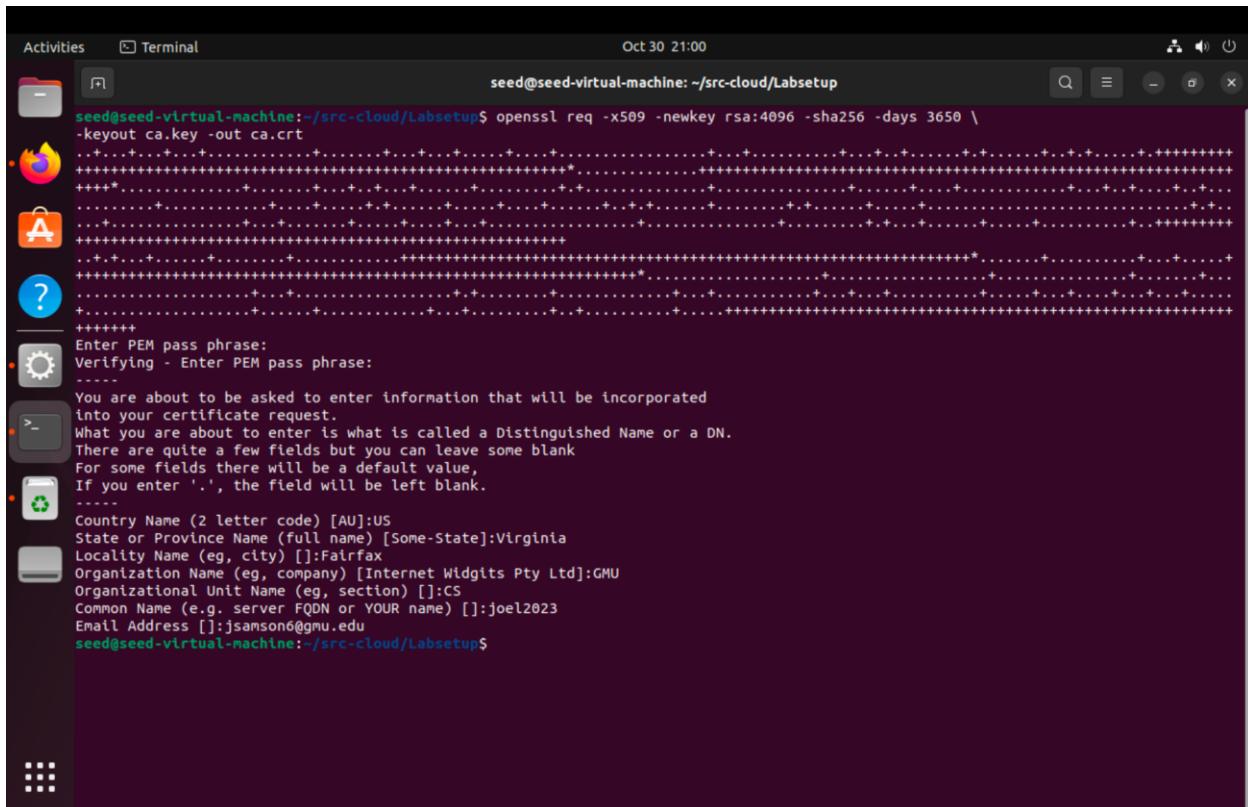
City: Fairfax

Organization Name: GMU

Organization Unit Name: CS

Company Name: joel2023

Email: jsamson6@gmu.edu



The screenshot shows a terminal window titled "seed@seed-virtual-machine: ~/src-cloud/Labsetup". The command entered was:

```
seed@seed-virtual-machine:~/src-cloud/Labsetup$ openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 \
-keyout ca.key -out ca.crt
```

The terminal then prompts for a PEM pass phrase, which is left blank. It continues with the following message:

```
-----  
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.  
-----
```

It then lists the certificate fields and their values:

```
Country Name (2 letter code) [AU]:US  
State or Province Name (full name) [Some-State]:Virginia  
Locality Name (eg, city) []:Fairfax  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:GMU  
Organizational Unit Name (eg, section) []:CS  
Common Name (e.g. server FQDN or YOUR name) []:joel2023  
Email Address []:jsamson6@gmu.edu
```

The command concludes with:

```
seed@seed-virtual-machine:~/src-cloud/Labsetup$
```

The output of the command are stored in two files: **ca.key** and **ca.crt**. The file **ca.key** contains the CA's private key, while **ca.crt** contains the public-key certificate

Files after command:-----

```
seed@seed-virtual-machine:~/src-cloud/Labsetup$ ls
ca.crt  ca.key  demoCA  docker-compose.yml  image_www  openssl.cnf  serial  volumes
seed@seed-virtual-machine:~/src-cloud/Labsetup$
```

ca.crt file details:----

Activities View file Oct 30 21:01 ca.crt

joel2023
Identity: joel2023
Verified by: joel2023
Expires: 10/28/2033

Details

Subject Name
C (Country): US
ST (State): Virginia
L (Locality): Fairfax
O (Organization): GMU
OU (Organizational Unit): CS
CN (Common Name): joel2023
EMAIL (Email Address): jsamson6@gmu.edu

Issuer Name
C (Country): US
ST (State): Virginia
L (Locality): Fairfax
O (Organization): GMU
OU (Organizational Unit): CS
CN (Common Name): joel2023
EMAIL (Email Address): jsamson6@gmu.edu

Issued Certificate
Version: 3
Serial Number: 25 7E DA 69 03 90 C0 9B 39 18 89 F1 96 CE 01 F0 33 07 AD B1
Not Valid Before: 2023-10-31
Not Valid After: 2033-10-28

Certificate Fingerprints
SHA1: 84 EF 69 88 D2 95 11 F5 D0 83 F0 EC 85 69 E5 A2 37 BD 21 F9
MD5: BD C3 E8 74 EB CC 6E E4 A9 1A 69 57 35 81 FD 77

Public Key Info

Close Import

Activities View file Oct 30 21:01

Public Key Info
Key Algorithm: RSA
Key Parameters: 05 00
Key Size: 4096
Key SHA1 Fingerprint: CE 30 CA 1B 22 82 CC 56 6A 38 DF A7 58 00 A3 23 E5 0E 85 AE
Public Key:
30 82 02 0A 02 82 02 01 00 9D A8 68 A8 3E 4A 80 EF 46 86 1D AF FD A1 88 34 37 FC 56 32 72 6D AA 33 45 A9 E1 0B
EF 03 3C D8 7E 22 99 9C 13 77 8A 00 0C 6A 86 ED C0 0A 4C C1 84 95 48 C4 E5 8B 75 B0 8C EB 8B 40 A0 B0 3F 68 62
8B 2C 6D D2 52 95 24 8A 98 2A 64 D2 66 B0 3E F0 2F 0B F2 5E 6D 22 82 19 28 0B 74 65 00 60 DC DF 23 59 32 A1 63
59 6A 04 8F C2 F1 C7 FF 47 1F FE F8 E5 CC D7 E9 1B A8 AC 67 FE 47 99 FB 44 B7 8C BC 49 8D 4A CA 05 DC 2B C0 2E
89 55 53 FA 07 3E 5C A8 FA 32 08 D8 81 7A D6 76 E5 7C A4 BD 25 BB FE 8D 6A 3B 1A 69 B9 B8 D4 D3 24 36 21 B4 2F
5B 4E C7 7C E9 A8 B3 6B 35 93 DA 93 CA 08 4B 94 0D 42 C9 70 8C 4E 61 58 59 5C C7 11 A1 18 2D 7C C1 6B 69 FF
84 91 58 00 7F C9 C8 E6 6D 47 D3 44 A4 AE 36 26 A1 EE F2 A6 15 D8 69 D4 C3 47 F9 AF B2 2E 3D 83 BA 1B 47 6E 9C
16 F1 68 D0 95 A5 BB E3 11 DD 41 03 B7 EE D9 A8 36 E8 C9 42 77 F5 DD 5C 1D 7B E8 97 27 F0 5C E0 99 B7 20 63 6D
50 EC 17 DB F7 9E 30 DE E1 2E 1F C7 AA 61 EB 8B 60 07 8A A0 75 6C 43 40 1D DE BB 41 0E 68 E4 E1 17 7A 75 0F
D0 9B A4 0F 3D B3 17 26 38 9E 99 F1 94 A6 0A 17 DC 7A 3F 87 31 D3 F8 11 23 F0 91 14 EF A5 AB 42 AC EA 49 9D C0
89 D4 5F 77 FB B7 D1 2B E2 88 B7 3E BA 1E 23 6E CD C4 95 09 15 DD 95 31 87 F1 15 52 66 52 10 E5 0F 7B CF E7 37
01 E5 1C 18 43 D2 1D CA EA C0 7A 79 42 35 0C 60 AB D6 D7 DC E8 92 75 69 5F 0D E0 47 16 7E DC BB 28 FD B9 29 66
A0 36 82 6A 25 BF CD E8 EA 77 58 C4 C4 8F 99 25 EF 2D F5 9F 8E 89 AC 74 72 DD 10 EC F8 36 45 24 27 D5 5F EE 0A
C5 93 CE F4 5A 60 42 2C 28 DB 96 AE A6 D1 17 2F 30 F1 FA 63 04 09 71 29 D5 04 79 9A 30 D5 FF CC A6 B3 D9 A4 3A
9C 5C FF 02 03 01 00 01

Subject Key Identifier
Key Identifier: 94 CE B9 9F 34 35 81 CF 0A 1A FC 60 85 79 C8 6A D4 3B 8C D4
Critical: No

Extension
Identifier: 2.5.29.35
Value: 30 16 80 14 94 CE B9 9F 34 35 81 CF 0A 1A FC 60 85 79 C8 6A D4 3B 8C D4
Critical: No

Basic Constraints
Certificate Authority: Yes
Max Path Length: Unlimited
Critical: Yes

Signature

Close

```

Activities View file Oct 30 21:01
ca.crt

01 E5 1C 18 43 D2 1D CA EA C0 7A 79 42 35 0C 60 AB D6 D7 DC E8 92 75 69 5F 0D E0 47 16 7E DC BB 28 FD B9 29 66
A0 36 82 6A 25 BF CD E8 EA 77 58 C4 C4 8F 99 25 EF 2D F5 9F 8E 89 AC 74 72 DD 10 EC FB 36 45 24 27 D5 5F EE 0A
C5 93 CE F4 5A 60 42 2C 28 DB 96 AE A6 D1 17 2F 30 F1 FA 63 04 09 71 29 D5 04 79 9A 30 D5 FF CC A6 B3 D9 A4 3A
9C C5 FF 02 03 01 00 01

Subject Key Identifier
KeyIdentifier: 94 CE B9 9F 34 35 81 CF 0A 1A FC 60 85 79 C8 6A D4 3B 8C D4
Critical: No

Extension
Identifier: 2.5.29.35
Value: 30 16 80 14 94 CE B9 9F 34 35 81 CF 0A 1A FC 60 85 79 C8 6A D4 3B 8C D4
Critical: No

Basic Constraints
Certificate Authority: Yes
Max Path Length: Unlimited
Critical: Yes

Signature
Signature Algorithm: 1.2.840.113549.1.1.11
Signature Parameters: 05 00
Signature: 3E 20 6B 1C 55 1A AA 9B 58 E0 FF A9 0F 55 42 4D 61 3D A4 76 83 0C 39 FA 4F A8 F9 A1 BF 41 C5 3E 7E B4 04 75 DA
14 2E 6E 60 C4 1F 72 52 06 C4 4D E1 F2 B2 F1 26 97 3B 91 15 2D 9A BD 84 E8 50 02 49 52 C6 65 28 A3 EB D2 FA A5
9B 73 9F 79 28 B2 50 07 63 A7 B2 0F 2B 5B BA 17 57 7E 1A 33 E7 AF 9F 82 58 E0 3F 22 1F 9C 0E 38 FC ED 05 26 BF
5E CB 8A FC 04 39 03 90 59 50 E9 B1 FA 90 8A 29 E6 B6 73 DC 06 0E 10 17 03 48 18 1F 8D 63 E2 1A EC 2D 0D 57 12
33 72 06 55 6B 33 33 2F 1A 50 9B 81 92 36 02 DF EA 23 41 89 DC 5E A3 D8 A2 FA 52 CC F2 81 80 F4 D5 A0 1A 7E 68
71 B8 71 00 A9 D0 B6 F0 C3 10 3B 3D 62 C7 62 7B 94 9B 87 8E DA BD 79 EB 7F 25 44 68 BB E1 77 30 E9 9D 97 B2 F0
50 1C AC 7B C8 0D DF 49 20 82 B6 56 A6 95 FD 77 7A 67 64 55 27 FA 2C 3B C2 B5 C2 FE 5D 03 D9 48 F0 27 41 EF 5D
D4 24 79 F3 97 A2 F0 8E A1 8D E5 E6 55 8E 84 8B 52 5A 33 03 A8 AD F7 80 01 59 F1 BC 5B 25 30 E3 2B 60 EC D0
FD 51 6C D6 67 B6 CB F0 F6 E2 F5 30 99 24 96 42 76 7D 49 22 A7 90 0F 96 F8 0E E1 5C 6B 85 E3 B8 D2 BF 4E 55
6F 4C 81 BA 9F FF DD 15 63 08 05 CA 43 9B C2 88 7E A0 66 8C 2E DD 9B 1F A6 AB 50 07 36 E4 C9 8A 32 A5 A7 57 4D
15 81 FC F5 72 09 03 F6 8D C5 9A 54 88 C3 D2 C7 A2 41 14 1C F1 91 EB 23 08 A1 30 00 9A C5 CF 26 71 B8 DA E7 00
BB D0 6C 29 DB 4F 08 6A EF A1 1C 75 0C FA 83 F8 47 74 A6 7B 8C 5D E5 17 CB DB 2D D8 4C 69 D5 42 7D 5B C7 A5 93
FA BD 20 A8 2A 71 CB D0 B1 F3 FB DE BD 31 7E 9F 68 70 43 20 03 FA AA 87 FC 69 5A 35 4F 19 E7

```

We can use the following command to look at the decoded content of the X509 certificate

openssl x509 -in ca.crt -text -noout

Output as follows

```

Activities Terminal Oct 30 21:03
seed@seed-virtual-machine:~/src-cloud/Labsetup$ openssl x509 -in ca.crt -text -noout
seed@seed-virtual-machine:~/src-cloud/Labsetup$ Certificate:
Data:
Version: 3 (0x2)
Serial Number:
25:7e:da:69:03:90:c0:9b:39:18:89:f1:96:ce:01:f0:33:07:ad:b1
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = US, ST = Virginia, L = Fairfax, O = GMU, OU = CS, CN = joel2023, emailAddress = jsamson6@gmu.edu
Validity
Not Before: Oct 31 01:00:16 2023 GMT
Not After : Oct 28 01:00:16 2033 GMT
Subject: C = US, ST = Virginia, L = Fairfax, O = GMU, OU = CS, CN = joel2023, emailAddress = jsamson6@gmu.edu
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (4096 bit)
Modulus:
00:9d:a8:68:a8:3e:4a:80:ef:46:86:1d:af:fd:a1:
88:34:37:fc:56:32:72:6d:aa:33:45:a9:e1:0b:ef:
03:3c:d8:7e:22:99:9c:13:77:8a:00:c6:a8:ed:
c0:0a:4c:c1:84:95:48:c4:e5:8b:75:b0:c1:eb:8b:
40:a0:b0:3f:08:02:8b:2c:6d:d2:52:95:24:8a:98:
23:64:d2:66:b3:3e:f0:2f:0b:f2:5e:0d:22:82:19:
28:0b:74:65:00:0d:cd:df:23:59:32:a1:63:59:6a:
04:8f:c2:f1:c7:ff:47:1f:fe:f8:e5:cc:d7:e9:1b:
a8:ac:67:fe:47:99:fb:44:b7:8c:bc:49:8d:4a:ca:
05:dc:2b:c0:2e:89:55:53:fa:07:3e:5c:a8:f1:32:
08:d8:81:7a:0d:76:e5:7c:a4:bd:25:bb:f8:6a:
3b:1a:69:b9:bb:d4:d3:24:36:21:b4:f5:b4:ec:c7:
7c:e9:a8:b3:6b:35:93:da:93:ca:08:4b:94:0d:42:
c9:70:8c:4e:61:58:59:5c:c7:11:a1:18:2d:7c:7c:
c1:6b:69:ff:84:91:58:0d:7f:c9:c8:6e:6d:47:d3:
4c:a4:ae:36:26:a1:ee:f2:a6:15:d8:69:d4:c3:47:
f9:af:b2:2e:3d:83:ba:1b:47:6e:9c:i6:f1:68:d0:
95:as:bb:e3:11:d4:41:03:b7:ee:d9:a8:36:e8:c9:
42:77:f5:dd:sc:1d:7b:e8:97:27:f0:5c:e0:99:b7:
20:63:6d:50:a1:7:db:f7:9e:30:de:e1:e1:fc:7:
aa:61:eb:8b:60:07:8a:a0:75:6c:43:40:1d:de:bb:
41:0e:68:e4:41:b1:17:7a:75:0f:d0:9b:a4:0f:3d:
b3:17:26:38:9e:99:f1:94:a6:0a:17:dc:7a:3f:87:
31:d3:f8:11:23:f0:91:14:ef:a5:ab:42:ac:ea:49:
```

Activities Terminal Oct 30 21:03

```
seed@seed-virtual-machine: ~/src-cloud/Labsetup
```

31:d3:f8:11:23:f0:91:14:ef:a5:ab:42:ac:ea:49:
9d:c0:89:d4:5f:77:fb:b7:d1:2b:e2:88:b7:3e:ba:
1e:23:6e:cd:c4:95:09:15:dd:95:31:87:f1:15:52:
60:52:10:e5:0f:7:bc:f1:e7:37:01:e5:ic:8:43:d2:
1d:ca:ea:c0:7a:79:42:35:0c:60:ab:d6:d7:dc:e8:
92:75:69:5f:0d:e0:47:16:7e:dc:bb:28:fd:b9:29:
66:a0:36:82:6a:25:bf:cd:e8:ea:77:58:c4:c4:8f:
99:25:ef:zd:f5:9f:8e:89:ac:74:72:dd:0:ec:f8:
36:45:24:27:d5:f1:ee:0:a:c5:93:ce:f4:5a:60:42:
2c:28:db:96:ae:a6:d1:17:2f:30:f1:fa:63:04:09:
71:29:d5:04:79:9a:30:d5:ff:cc:a6:b3:d9:a4:3a:
9c:c5:ff
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Key Identifier:
94:CE:B9:9F:34:35:81:CF:0A:1A:FC:60:85:79:C8:6A:D4:3B:8C:D4
X509v3 Authority Key Identifier:
94:CE:B9:9F:34:35:81:CF:0A:1A:FC:60:85:79:C8:6A:D4:3B:8C:D4
X509v3 Basic Constraints: critical
CA:TRUE
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
3e:20:6b:1c:55:1a:aa:9b:58:e0:ff:a9:0f:55:42:4d:61:3d:
a4:76:83:0c:39:fa:4f:a8:f9:a1:bf:41:c5:3e:7e:b4:04:75:
da:14:2e:6e:60:c4:1f:72:52:06:c4:4d:e1:f2:b2:f1:26:97:
3b:91:15:2d:9a:bd:84:e8:50:02:49:52:c0:65:28:a3:eb:d2:
fa:a5:9b:73:9f:79:28:b2:50:d7:63:a7:b2:0f:2b:5b:ba:17:
57:e1:a3:33:e7:af:9f:82:58:e0:3f:22:1f:9c:0e:38:fc:ed:
05:26:bf:5e:cb:8a:fc:04:39:03:90:59:50:e9:b1:fa:9d:8a:
29:e6:b6:73:dc:06:0e:10:17:03:48:18:1f:8d:63:e2:i:ace:
2d:d0:57:12:33:72:06:55:6b:33:33:2f:1a:50:9b:81:92:36:
d2:df:ea:23:41:89:dc:5e:a3:d1:a2:fa:52:cc:f2:81:b0:f4:
d5:a0:1a:7e:68:71:08:71:00:a9:d0:b0:f0:c3:10:3b:3d:62:
c7:62:7b:94:9b:87:8e:da:bd:79:eb:7f:25:44:68:bb:e1:77:
30:e9:9d:97:b2:f0:50:1c:ac:7b:c8:0d:df:49:20:82:b6:56:
a6:95:fd:77:7a:67:64:55:27:fa:2c:3b:c2:b5:c2:fe:5d:d3:
c9:48:f0:27:41:ef:5d:d4:24:79:f3:97:a2:f0:8e:a1:8d:e5:
eb:6e:55:8e:84:8b:52:5a:33:03:a8:ad:f7:80:01:59:f1:bc:
5b:25:30:e3:2b:66:ec:d0:fd:51:6c:d6:67:51:a2:2b:e3:39:
7e:01:90:4e:8b:49:19:d1:32:93:b2:3c:5b:85:b7:44:40:28:
4f:63:e5:1a:36:00:6c:ba:de:6a:53:88:67:0f:b6:cb:0:f6:
2e:f5:30:99:24:96:42:76:7d:49:22:a7:9d:0f:96:f8:e8:e:
e1:5c:6b:85:e3:b8:d2:bf:4e:55:6f:4c:81:ba:9f:ff:dd:15:
63:08:05:ca:43:9b:c2:88:7e:a0:66:8c:2e:dd:9b:1f:a6:ab:
50:07:36:4e:c9:8a:32:a5:a7:57:4d:15:81:fc:f5:72:09:03:
f6:8d:c5:9a:54:88:c3:d2:c7:a2:41:14:1c:f1:91:eb:23:08:
a1:30:00:9a:c5:cf:26:71:b8:da:e7:00:bb:d0:6c:29:ob:4f:
08:6a:ef:a1:1c:75:0c:fa:83:f8:47:74:a6:7b:8c:5d:e5:17:
cb:db:2d:d8:4c:69:d5:42:7d:5b:ic:7:a5:93:fa:bd:20:a8:2a:
71:c:b:d0:b1:f3:fb:de:bd:31:7e:9f:68:70:43:20:03:fa:aa:
87:fc:69:5a:35:4f:19:e7

Activities Terminal Oct 30 21:03

```
seed@seed-virtual-machine: ~/src-cloud/Labsetup
```

Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Key Identifier:
94:CE:B9:9F:34:35:81:CF:0A:1A:FC:60:85:79:C8:6A:D4:3B:8C:D4
X509v3 Authority Key Identifier:
94:CE:B9:9F:34:35:81:CF:0A:1A:FC:60:85:79:C8:6A:D4:3B:8C:D4
X509v3 Basic Constraints: critical
CA:TRUE
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
3e:20:6b:1c:55:1a:aa:9b:58:e0:ff:a9:0f:55:42:4d:61:3d:
a4:76:83:0c:39:fa:4f:a8:f9:a1:bf:41:c5:3e:7e:b4:04:75:
da:14:2e:6e:60:c4:1f:72:52:06:c4:4d:e1:f2:b2:f1:26:97:
3b:91:15:2d:9a:bd:84:e8:50:02:49:52:c0:65:28:a3:eb:d2:
fa:a5:9b:73:9f:79:28:b2:50:d7:63:a7:b2:0f:2b:5b:ba:17:
57:e1:a3:33:e7:af:9f:82:58:e0:3f:22:1f:9c:0e:38:fc:ed:
05:26:bf:5e:cb:8a:fc:04:39:03:90:59:50:e9:b1:fa:9d:8a:
29:e6:b6:73:dc:06:0e:10:17:03:48:18:1f:8d:63:e2:i:ace:
2d:d0:57:12:33:72:06:55:6b:33:33:2f:1a:50:9b:81:92:36:
d2:df:ea:23:41:89:dc:5e:a3:d1:a2:fa:52:cc:f2:81:b0:f4:
d5:a0:1a:7e:68:71:08:71:00:a9:d0:b0:f0:c3:10:3b:3d:62:
c7:62:7b:94:9b:87:8e:da:bd:79:eb:7f:25:44:68:bb:e1:77:
30:e9:9d:97:b2:f0:50:1c:ac:7b:c8:0d:df:49:20:82:b6:56:
a6:95:fd:77:7a:67:64:55:27:fa:2c:3b:c2:b5:c2:fe:5d:d3:
c9:48:f0:27:41:ef:5d:d4:24:79:f3:97:a2:f0:8e:a1:8d:e5:
eb:6e:55:8e:84:8b:52:5a:33:03:a8:ad:f7:80:01:59:f1:bc:
5b:25:30:e3:2b:66:ec:d0:fd:51:6c:d6:67:51:a2:2b:e3:39:
7e:01:90:4e:8b:49:19:d1:32:93:b2:3c:5b:85:b7:44:40:28:
4f:63:e5:1a:36:00:6c:ba:de:6a:53:88:67:0f:b6:cb:0:f6:
2e:f5:30:99:24:96:42:76:7d:49:22:a7:9d:0f:96:f8:e8:e:
e1:5c:6b:85:e3:b8:d2:bf:4e:55:6f:4c:81:ba:9f:ff:dd:15:
63:08:05:ca:43:9b:c2:88:7e:a0:66:8c:2e:dd:9b:1f:a6:ab:
50:07:36:4e:c9:8a:32:a5:a7:57:4d:15:81:fc:f5:72:09:03:
f6:8d:c5:9a:54:88:c3:d2:c7:a2:41:14:1c:f1:91:eb:23:08:
a1:30:00:9a:c5:cf:26:71:b8:da:e7:00:bb:d0:6c:29:ob:4f:
08:6a:ef:a1:1c:75:0c:fa:83:f8:47:74:a6:7b:8c:5d:e5:17:
cb:db:2d:d8:4c:69:d5:42:7d:5b:ic:7:a5:93:fa:bd:20:a8:2a:
71:c:b:d0:b1:f3:fb:de:bd:31:7e:9f:68:70:43:20:03:fa:aa:
87:fc:69:5a:35:4f:19:e7

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

After running the command to look at the decoded content of the X509 certificate
This part of the decoded X509 certificate indicates the CA certificate.

The part where it is written **CA: True**

```
X509v3 extensions:  
    X509v3 Subject Key Identifier:  
        94:CE:B9:9F:34:35:81:CF:0A:1A:FC:60:85:79:C8:6A:D4:3B:8C:D4  
    X509v3 Authority Key Identifier:  
        94:CE:B9:9F:34:35:81:CF:0A:1A:FC:60:85:79:C8:6A:D4:3B:8C:D4  
    X509v3 Basic Constraints: critical  
        CA:TRUE  
    Signature Algorithm: sha256WithRSAEncryption
```

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

After running the command to look at the decoded content of the X509 certificate
If the issuer and subject fields are identical as shown in the following figure then we know it is a self signed certificate.

```
seed@seed-virtual-machine:~/src-cloud/LabSetup$ openssl x509 -in ca.crt -text -noout  
Certificate:  
Data:  
    Version: 3 (0x2)  
    Serial Number:  
        25:7e:da:69:03:90:c0:9b:39:18:89:f1:96:ce:01:f0:33:07:ad:b1  
    Signature Algorithm: sha256WithRSAEncryption  
    Issuer: C = US, ST = Virginia, L = Fairfax, O = GMU, OU = CS, CN = joel2023, emailAddress = jsamson6@gmu.edu  
    Validity  
        Not Before: Oct 31 01:00:16 2023 GMT  
        Not After : Oct 28 01:00:16 2033 GMT  
    Subject: C = US, ST = Virginia, L = Fairfax, O = GMU, OU = CS, CN = joel2023, emailAddress = jsamson6@gmu.edu  
    Subject Public Key Info:  
        Public Key Algorithm: rsaEncryption  
        Public-Key: (4096 bit)
```

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.

After running the the following commands to look at the decoded content the RSA key

openssl rsa -in ca.key -text -noout

We get the following output

This shows us the values of the public exponent e, a private exponent d, a modulus n, and two secret numbers p and q, such that n = pq.

Activities Terminal Oct 30 21:06

```
seed@seed-virtual-machine:~/src-cloud/Labsetup$ openssl rsa -in ca.key -text -noout
Enter pass phrase for ca.key:
Private-Key: (4096 bit, 2 primes)
modulus:
00:9d:a8:68:a8:3e:4a:80:ef:46:86:1d:af:fd:a1:
88:34:37:fc:56:32:72:6d:aa:33:45:a9:1e:0b:ef:
03:3c:d8:7e:22:99:9c:13:77:8a:00:0c:6a:86:ed:
c0:0a:4c:c1:84:95:48:c4:e5:b8:75:b0:8c:eb:b8:
40:a0:b0:3f:68:62:8b:2c:6d:d2:52:95:24:8a:98:
2a:64:d2:66:b6:3e:f0:2f:0b:f2:5e:0d:22:82:19:
28:0b:74:65:00:60:dc:df:23:59:32:a1:63:59:6a:
04:8f:c2:f1:c7:ff:47:1f:fe:f8:e5:cc:d7:e9:1b:
a8:ac:67:fe:47:99:fb:44:b7:8c:bc:49:8d:4a:ca:
05:dc:2b:c0:2e:89:55:53:fa:07:3e:5c:a8:f1:32:
08:d8:81:7a:d6:76:e5:7c:a1:bd:25:bb:fe:8d:6a:
3b:1a:69:b9:b8:d4:d3:24:36:21:b4:2f:sb:4e:c7:
7c:e9:a8:b3:b6:35:93:da:93:ca:08:4b:94:0d:42:
c9:70:8c:4e:61:58:59:5c:c7:11:a1:18:2d:7c:7c:
c1:6b:69:ff:84:91:58:0d:7f:c9:c8:e6:6d:47:d3:
4c:a4:ae:36:26:a1:ee:f2:a0:15:d8:09:4c:3:47:
f9:af:b2:2e:3d:83:ba:1b:47:0e:9c:16:f1:68:d0:
95:a5:bb:e3:11:d4:41:03:b7:ee:d9:a8:36:e8:c9:
42:77:f5:dd:5c:1d:7b:e8:97:27:f0:5c:e0:99:b7:
20:63:6d:50:ec:17:db:f7:9e:30:de:e1:2e:1f:c7:
aa:61:eb:8b:60:07:8a:a0:75:6c:43:40:1d:de:bb:
41:0e:68:e4:41:b1:b7:7a:75:0f:d0:9b:a4:0f:3d:
b3:17:26:38:9e:99:f1:94:a6:0a:17:dc:7a:3f:87:
31:d3:f8:11:23:f0:91:14:ef:a5:ab:42:ac:e4:49:
9d:c0:89:4d:5f:77:fb:b7:d1:2b:e2:88:b7:3e:ba:
1e:23:6e:cd:c4:95:09:15:dd:95:31:7f:1:15:52:
66:52:10:e5:0f:7b:c7:e7:37:01:e5:1c:18:43:d2:
1d:ca:ea:c0:7a:79:42:35:0c:60:ab:d6:d7:dc:e8:
92:75:69:5f:de:0:47:16:7e:dc:bb:28:fd:b9:29:
66:a0:36:82:6a:25:bf:cd:e8:ea:77:58:c4:c4:8f:
99:25:ef:2d:f5:9f:8e:89:ac:74:72:dd:10:ec:f8:
36:45:24:27:d5:5f:ee:0a:c5:93:ce:f4:5a:60:42:
2c:28:db:96:ae:a6:d1:17:2f:30:f1:fa:63:04:09:
71:29:d5:04:79:9a:30:d5:ff:cc:a6:b3:d9:a4:3a:
9c:c5:ff
publicExponent: 65537 (0x10001)
```

Activities Terminal Oct 30 21:06

```
seed@seed-virtual-machine:~/src-cloud/Labsetup$ privateExponent:
15:f2:f0:3f:1c:0c:fb:4e:54:9f:2f:5a:e5:10:ac:
09:b9:11:a4:5e:79:97:d2:ee:38:70:a2:28:2b:2a:
64:3a:2f:bb:59:75:65:30:6b:41:fe:78:86:38:6b:
58:03:bf:9d:59:bb:3b:ce:49:50:25:38:39:42:b5:
c8:a9:40:ea:a3:6d:84:cd:f5:4e:11:fd:78:5e:1e:
e8:d2:72:02:45:58:e2:f3:e2:d6:4a:93:57:51:71:
45:c6:fa:98:c6:e3:79:1d:4e:b9:c1:c6:37:18:8f:
72:04:4d:ec:ee:19:54:d7:06:87:d7:b0:7:7:36:
48:6a:61:f8:4d:f3:b1:69:6c:6a:7a:42:f5:7a:92:
be:48:49:05:31:91:73:93:87:96:a3:07:03:6d:4b:
d5:f8:90:c5:72:e0:18:2e:47:37:0d:a9:2e:be:ef:
63:89:fc:95:42:eb:f2:4a:a0:fe:92:e9:3f:5a:7e:
85:79:3b:21:5d:e0:2b:ba:0f:92:80:13:28:95:77:
bd:f7:8a:1f:fa:ce:d6:1f:d0:d0:80:b1:34:77:99:
e5:fe:67:d8:68:f3:bd:4d:61:26:5b:0e:33:c5:b3:
11:ea:3f:17:b9:cba:ae:74:78:ee:1c:2d:c6:d2:2a:
6c:cf:7d:a8:8a:c2:4a:31:f3:b6:3f:71:74:8a:35:
8b:19:5b:d1:70:ce:13:76:7d:29:51:ab:6b:44:b4:
ac:eb:eb:bc:40:07:cf:f7:71:86:19:e5:43:03:ce:
c4:45:c6:bb:5c:de:c7:aa:2f:45:e8:26:2c:3b:44:
63:46:f6:fd:b2:43:08:41:9f:12:5d:f2:0c:7c:41:
0b:e7:6c:49:4d:7f:2e:ae:62:89:19:13:c5:b7:69:
ee:ac:d6:73:87:95:4d:53:76:7c:d7:3d:18:d4:d7:
a9:73:7d:f2:f1:e3:3a:00:50:01:a8:69:90:e9:37:
32:b4:0a:b5:8d:e2:47:06:05:a2:ba:df:39:6:44:
f5:93:73:4e:09:e6:b4:98:2a:a4:70:17:fe:1:c7:
dd:7c:1d:01:05:33:1c:0c:9e:8d:ea:9a:54:c0:ea:
7b:a0:db:ab:55:c:e:ac:a4:bc:4d:6c:f1:88:fc:81:
5c:d9:59:67:e5:3c:7c:27:07:c2:8f:2c:4c:51:99:
87:2d:57:98:66:18:c1:54:35:66:3c:12:27:38:7e:
10:af:f9:49:b7:cc:6e:d9:47:a7:81:0e:a5:6f:01:
e4:33:c2:49:28:5c:4b:6d:9c:7c:4a:c5:db:dd:
d3:34:98:f1:fe:60:4e:fe:3e:9b:bf:52:2d:99:e9:
ca:fa:b8:38:bc:de:fb:b7:37:0b:db:51:b8:39:52:
1f:21
prime1:
00:ce:5d:88:8c:17:f9:d7:60:da:b9:2a:0c:be:4b:
7b:38:67:64:fd:df:41:0e:13:92:f3:00:f5:5e:0c:
f5:42:7e:b3:3c:1e:dc:fa:ec:6f:39:98:2a:17:d2:
```

Activities Terminal Oct 30 21:06

```
seed@seed-virtual-machine: ~/src-cloud/Labsetup
```

prime1:
00:ce:5d:88:8c:17:f9:d7:60:da:b9:2a:0c:be:4b:
7b:38:67:64:fd:df:41:0e:13:92:f3:00:f5:5e:0c:
f5:42:7e:b3:3c:1e:dc:fa:ec:6f:39:98:a1:7:d2:
60:96:32:b3:86:b3:f7:74:33:84:f8:8a:45:b0:29:
a8:75:f3:7e:ea:4f:f9:93:d4:07:15:ac:f8:c9:44:
33:02:6b:92:ff:e8:08:b7:8c:33:51:25:b8:cd:77:
37:16:8f:e1:59:54:4d:8e:cb:65:0c:d1:73:49:07:
f4:34:66:bf:9e:a9:c6:ed:53:44:9c:66:14:01:f8:
9a:b9:24:6b:08:8d:da:c3:c8:2a:2a:70:f9:e1:a:
ef:ac:47:33:bd:f1:20:c9:a1:d5:69:1b:13:10:8d:
7a:c7:da:4f:f4:88:f0:10:e1:05:24:f4:02:06:a1:
2b:91:cc:63:ce:9f:a7:bd:f4:9c:39:a3:18:f2:ed:
2f:f1:7f:ae:23:07:e4:23:6b:71:5f:36:02:c7:b2:
87:bc:73:17:a1:a0:59:1a:b3:87:2f:88:4f:d5:b9:
67:9d:30:a7:9b:67:ba:ec:9b:14:a5:44:c4:e9:ec:
2f:2c:43:35:54:ee:21:7e:ec:86:49:9c:87:9b:8e:
7a:32:eb:0c:e9:a5:19:be:d4:fb:13:c7:87:43:a4:
43:d3
prime2:
00:c3:93:d2:cf:a8:a3:64:db:aa:58:14:04:1b:20:
b1:14:4f:3d:9c:86:18:92:cd:47:0e:b7:a9:7:b0:8:
34:77:0d:e5:b4:bb:f7:7b:74:77:f9:d9:66:70:6a:
65:59:8c:2c:0c:19:90:4b:a6:64:cd:75:00:ae:3b:
a5:02:89:ac:4b:96:ca:3c:38:19:1b:36:23:e3:6a:
b3:77:45:02:ad:a7:8b:03:25:c8:53:62:70:e5:ef:
2f:5d:df:cd:89:b5:6e:47:8b:51:fb:a1:84:2a:e4:
b2:fd:9f:a2:86:9c:36:77:13:4a:f8:45:5f:fc:c4:
1f:16:9a:d3:b5:47:15:df:20:10:27:9f:43:51:c3:
27:b2:52:be:63:2e:fc:4e:53:88:50:c7:8f:fd:59:
14:72:9a:67:87:ec:0a:a3:7b:8c:d8:bf:f5:f1:7a:
0f:84:92:11:78:e5:75:2a:11:f0:46:d0:7b:af:2f:
01:dd:91:54:ef:45:id:di:e0:98:9e:c0:62:15:dd:
9b:39:12:5c:5f:ca:a6:7a:00:a5:63:f6:2e:fd:3e:
ef:8c:d8:08:a0:1e:f3:e0:6c:98:de:c0:d6:fa:52:
2e:d0:8a:d7:f3:35:f8:fb:dd:c9:03:52:c9:9b:ea:
da:9a:7d:70:23:5d:d0:f4:75:d8:3a:1e:ce:e0:e8:
55:a5
exponent1:
13:d8:dd:0c:14:36:3c:27:9d:81:97:70:b8:50:8c:

Activities Terminal Oct 30 21:06

```
seed@seed-virtual-machine: ~/src-cloud/Labsetup
```

6f
exponent2:
5f:c6:81:d7:b5:42:b9:e0:6a:17:d1:44:e9:5b:2a:
17:3d:6c:76:b3:98:58:57:34:7f:da:4f:99:a2:bc:
bd:05:0a:cf:5f:af:9c:bb:ba:f9:92:1c:7b:31:3c:
09:89:7d:0f:37:c3:ec:ds:31:59:f5:49:58:5e:
43:4d:d0:12:f9:f2:6d:65:31:b5:86:de:f3:b5:50:
3e:36:85:47:2b:dd:fd:5e:84:23:7e:f0:b3:42:55:
38:6a:98:20:b1:b9:1a:41:f5:93:72:d2:58:5b:
b4:3c:71:bf:1:b7:30:b1:3b:65:2d:1:f1:fe:c8:ae:
47:cc:e9:95:b9:05:d6:45:cg:62:ff:b0:5f:fd:62:
18:06:36:bc:2c:86:ce:7a:4b:24:f1:6f:54:8c:2c:
5c:a7:87:c4:9e:25:df:40:60:06:c6:84:cd:02:eb:
92:06:49:93:9d:13:ad:d3:36:86:15:10:59:06:74:
ad:5b:b7:ef:a5:31:22:4e:09:aa:bf:99:c6:06:c7:
36:3b:ed:09:e1:17:7c:67:d0:91:8e:38:17:38:b6:
e3:87:e9:41:2b:a6:4e:dc:ic:44:02:b9:bd:b8:7a:
bf:13:1b:03:32:0b:82:94:65:13:c1:7f:cc:be:40:
e4:9e:41:46:af:69:dc:25:ce:05:36:8c:26:c4:d9:
19
coefficient:
18:47:0e:dd:f6:f0:cc:87:e9:79:f5:ce:64:dd:c3:
6f:67:58:b8:4e:b4:1d:a9:61:1f:87:a0:c3:23:8b:
a2:a4:c9:02:31:f3:bd:cc:49:b7:7a:3d:21:90:e6:
e3:8d:da:20:67:87:33:fd:f5:6f:1f:5d:1b:5d:d6:
a9:14:43:c6:0f:e1:d1:5b:e7:74:e4:e9:b1:84:e7:
d7:4f:a1:4c:c6:48:fd:ba:9e:0c:3e:56:7f:f3:71:
31:ba:27:ee:a7:21:58:38:dd:43:bc:95:56:cc:a2:
06:0b:f0:e0:5a:01:41:b2:21:13:20:56:76:c8:3b:
b3:54:f7:1b:75:a5:00:b3:1a:c6:10:e9:34:69:
81:fd:14:48:f6:c0:4b:31:0b:3d:5c:1c:5a:92:a7:
f5:82:31:0f:7f:ff:99:8d:c2:f6:13:7b:ec:df:9:c8:
26:cc:72:14:8a:d9:30:39:cc:0f:66:af:ca:54:aa:
ab:37:7e:7d:64:79:46:de:5d:b6:f9:54:de:99:8c:
a3:46:c2:3f:5a:d0:68:cd:7c:19:ff:7a:c8:de:81:
93:9a:f3:14:86:51:ec:50:12:46:d9:8a:80:b0:29:
6a:1a:b3:53:28:79:90:b3:fb:63:91:cf:9f:e0:12:
99:aa:eb:e7:59:33:a3:8b:4f:97:d2:52:25:78:33:
ac
seed@seed-virtual-machine: ~/src-cloud/Labsetup\$

Task 2: Generating a Certificate Request for Your Web Server

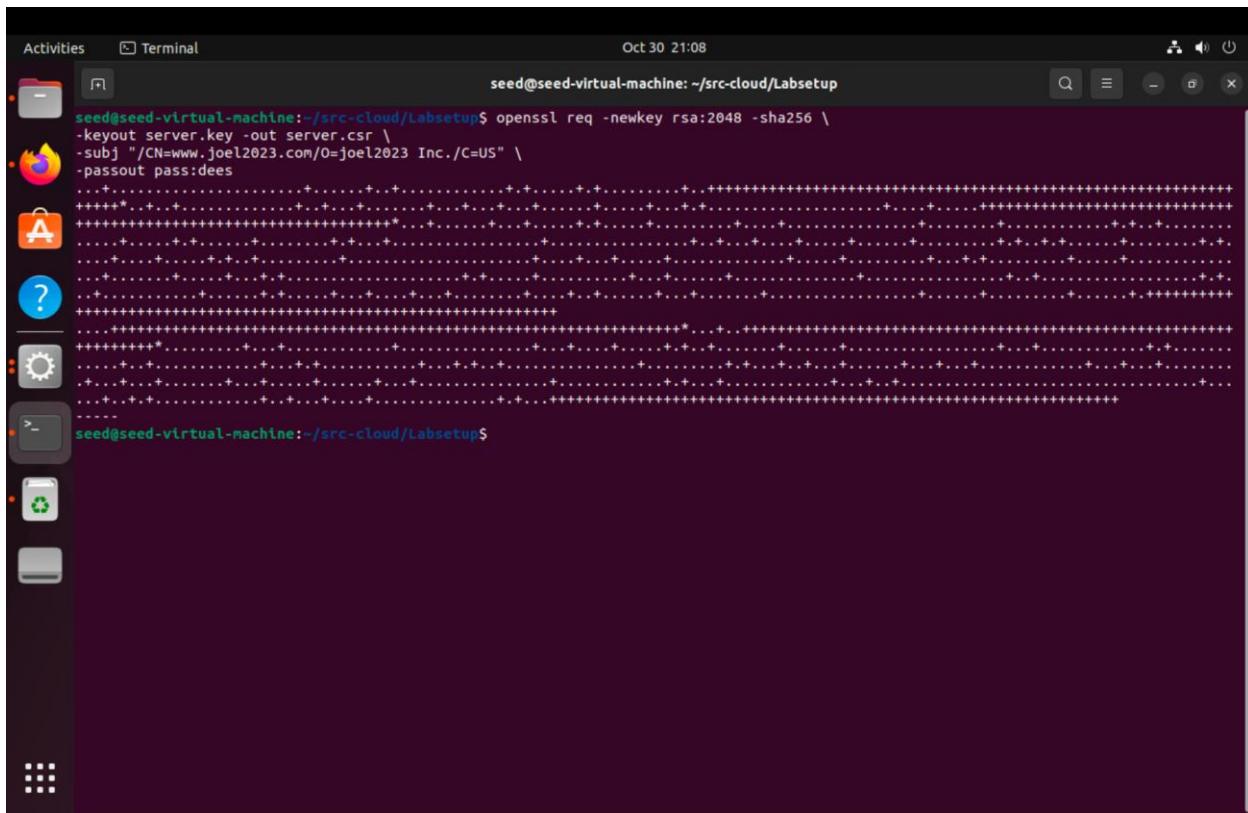
First it needs to generate a Certificate Signing Request (CSR), which basically includes the company's public key and identity information. The CSR will be sent to the CA, who will verify the identity information in the request, and then generate a certificate.

The command to generate a CSR is quite similar to the one we used in creating the self-signed certificate for the CA. The only difference is the **-x509 option**

The command to generate a CSR for my own server name **joel2023** is

```
openssl req -newkey rsa:2048 -sha256 \ -keyout server.key -out server.csr \ -subj "/CN=www.joel2023.com/O=joel2023 Inc./C=US" \ -passout pass:dees
```

Output as follows:



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark background and light-colored text. It displays the command being run and its output. The command is:

```
seed@seed-virtual-machine:~/src-cloud/Labsetup$ openssl req -newkey rsa:2048 -sha256 \
-keyout server.key -out server.csr \
-subj "/CN=www.joel2023.com/O=joel2023 Inc./C=US" \
-passout pass:dees
```

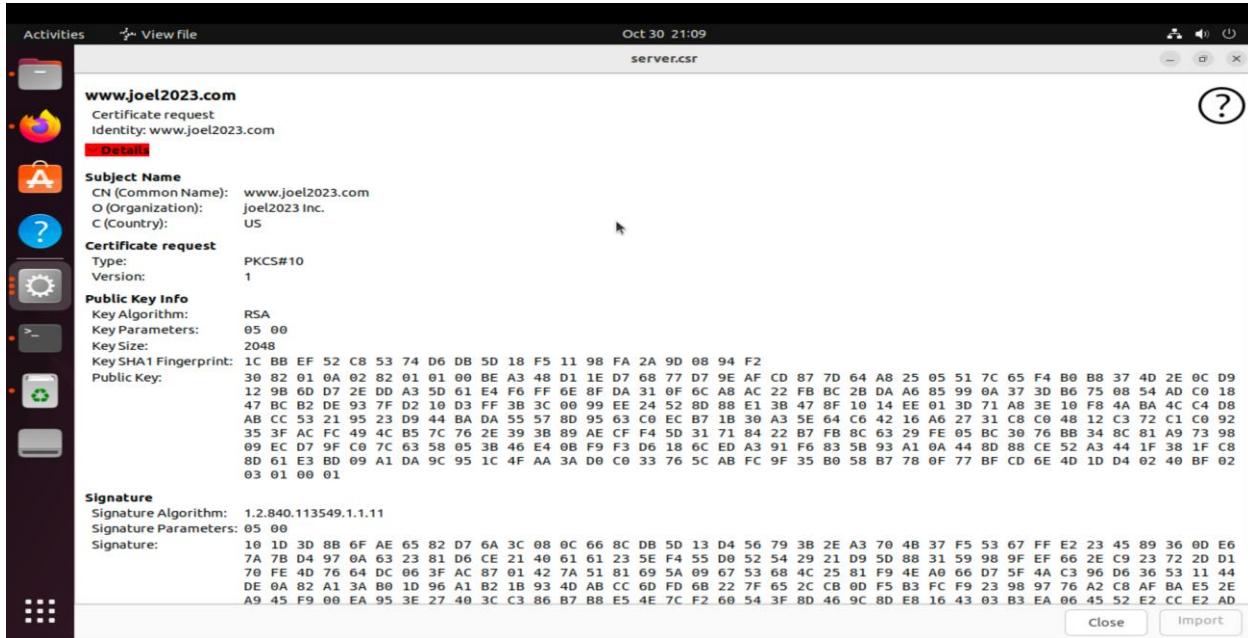
The output shows a large amount of asterisk (*) characters, indicating the progress or completion of the certificate signing request generation process. The terminal window is titled "seed@seed-virtual-machine: ~/src-cloud/Labsetup".

Files after running the command:

The command will generate a pair of **public/private key**, and then create a certificate signing request from the public key.
ie the **server.csr** and the **server.key**

```
seed@seed-virtual-machine:~/src-cloud/Labsetup$ ls
ca.crt ca.key demoCA docker-compose.yml image www openssl.cnf serial server.csr server.key volumes
seed@seed-virtual-machine:~/src-cloud/Labsetup$
```

The content of the **server.csr**(public key)



Note: This method is without adding the alternative names so I removed the **server.csr** and **server.key** generated and added the alternative name through the **-addext** option below.

Adding Alternative names:

To add alternative names we have to add another option to the command done before that is

```
-addext "subjectAltName = DNS:www.joel2023.com, \ DNS:www.joel2023A.com, \
DNS:www.joel2023B.com"
```

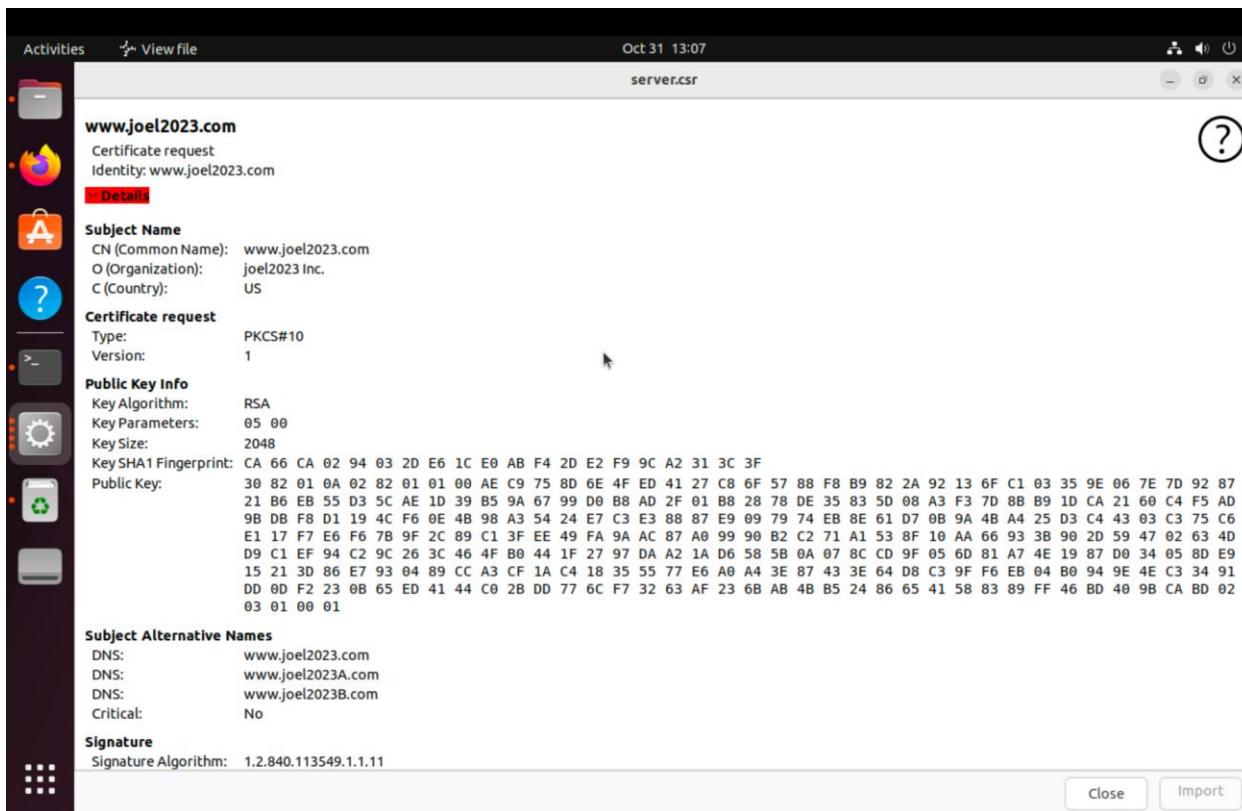
So the command becomes:

```
openssl req -newkey rsa:2048 -sha256 \ -keyout server.key -out server.csr \ -subj
"/CN=www.joel2023..com/O=joel2023 Inc./C=US" \
-addext "subjectAltName = DNS:www.joel2023.com, \ DNS:www.joel2023A.com, \
DNS:www.joel2023B.com" \
-passout pass:dees
```

Here we added 2 alternative names to the certificate

A look at the public key ie the **server.csr** generated:

We can see the alternative names of the website



```
seed@seed-virtual-machine:~/src/cloud/LabSetup$ ls
Backup ca.crt ca.key demo.ca docker-compose.yml image_www openssl.cnf serial server.csr server.key volumes
```

3) Task 3: Generating a Certificate for your server

The **CSR file** needs to have the CA's signature to form a certificate.

Here in the lab we will be using our own trusted CA to generate the certificate for the server. In real world scenarios the CSR files are usually sent to a trusted CA for their signature.

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:

```
openssl ca -config myCA_openssl.cnf -policy policyAnything \ -md sha256 -days 3650 \ -in server.csr -out server.crt -batch \ -cert ca.crt -keyfile ca.key
```

We use the **policyAnything** policy defined in the configuration file.

If we do this we won't be able to see the alternative names to enable this we have to go to the config file or copy the same config file and change some values in it ie uncomment the line:

```
# Extension copying option: use with caution.  
copy_extensions = copy
```

Command execution as follows:

```
seed@seed-virtual-machine:~/src-cloud/LabSetup$ openssl ca -config myCA_openssl.cnf -policy policyAnything -md sha256 -days 3650 -in server.csr -out server.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: 1 (0x1)  
    Validity  
        Not Before: Oct 31 17:14:00 2023 GMT  
        Not After : Oct 28 17:14:00 2033 GMT  
    Subject:  
        countryName      = US  
        organizationName = joel2023 Inc.  
        commonName       = www.joel2023.com  
X509v3 extensions:  
    X509v3 Basic Constraints: I  
        CA:FALSE  
    X509v3 Subject Key Identifier:  
        44:6F:04:71:CF:A0:32:71:76:FA:E2:F2:A5:2A:81:43:9A:1A:63:6F  
    X509v3 Authority Key Identifier:  
        94:CE:B9:9F:34:35:81:CF:0A:1A:FC:60:85:79:C8:6A:D4:3B:8C:D4  
    X509v3 Subject Alternative Name:  
        DNS:www.joel2023.com, DNS:www.joel2023A.com, DNS:www.joel2023B.com  
Certificate is to be certified until Oct 28 17:14:00 2033 GMT (3650 days)  
Write out database with 1 new entries  
Data Base Updated  
seed@seed-virtual-machine:~/src-cloud/LabSetup$
```

After signing the certificate, please use the following command to print out the decoded content of the certificate, and check whether the alternative names are included.

```
openssl x509 -in server.crt -text -noout
```

```

Activities Terminal Oct 31 14:37
seed@seed-virtual-machine:~/src-cloud/Labsetup$ openssl x509 -in server.crt -text -noout
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 1 (0x1)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = US, ST = Virginia, L = Fairfax, O = GMU, OU = CS, CN = joel2023, emailAddress = jsamson6@gmu.edu
Validity
    Not Before: Oct 31 17:14:00 2023 GMT
    Not After : Oct 28 17:14:00 2033 GMT
Subject: C = US, O = joel2023 Inc., CN = www.joel2023.com
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
            Modulus:
                00:ae:c9:75:8d:6e:4f:ed:41:27:c8:6f:57:88:f8:
                b9:82:2a:92:13:6f:c1:03:35:9e:06:7e:d9:2:87:
                21:b6:eb:55:d3:5c:ae:1d:39:b5:9a:67:99:d0:b8:
                ad:2f:01:b8:28:78:de:35:83:5d:08:a3:f3:7d:8b:
                b9:id:ca:21:60:c4:f5:ad:9b:db:f8:d1:19:4c:f6:
                0e:4b:98:a3:54:24:e7:c3:e3:88:87:e9:09:79:74:
                eb:8e:61:d7:0b:9a:4b:a4:25:d3:c4:43:03:c3:75:
                c0:e1:17:f7:6:7b:9f:2c:89:c1:fe:ee:49:fa:
                9a:ac:87:a0:99:90:b2:c2:71:a1:53:8f:10:a:66:
                93:3b:9b:2d:59:47:02:63:4d:d9:c1:ef:94:c2:9c:
                26:3c:46:4f:b0:44:1f:27:97:da:a2:1a:6d:58:5b:
                0a:07:8c:cd:f9:05:6d:81:a7:4e:19:87:d0:34:05:
                8d:e9:15:21:3d:86:e7:93:04:89:cc:a3:c1:a:c4:
                18:35:55:77:6:a:0:a4:3e:87:43:3e:64:d8:c3:9f:
                f6:eb:04:b0:94:9e:4e:c3:34:91:dd:0d:f2:23:0b:
                65:ed:41:44:c0:2:b:dd:77:6c:f7:32:63:af:23:6b:
                ab:4b:b5:24:86:65:41:58:83:89:ff:46:bd:40:9b:
                ca:bd
            Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Basic Constraints:
    CA:FALSE
X509v3 Subject Key Identifier:
    44:6F:04:71:CF:A0:32:71:76:FA:E2:F2:A5:2A:81:43:9A:1A:63:6F
X509v3 Authority Key Identifier:

```

```

Activities Terminal Oct 31 14:37
seed@seed-virtual-machine:~/src-cloud/Labsetup$ curl -s https://www.joel2023.com | grep -i "x509"
X509v3 Basic Constraints:
    CA:FALSE
X509v3 Subject Key Identifier:
    44:6F:04:71:CF:A0:32:71:76:FA:E2:F2:A5:2A:81:43:9A:1A:63:6F
X509v3 Authority Key Identifier:
    94:C:E:B9:9F:34:35:81:CF:0:A1:FC:60:85:79:C8:6A:D4:3B:8C:D4
X509v3 Subject Alternative Name:
    DNS:www.joel2023.com, DNS:www.joel2023A.com, DNS:www.joel2023B.com
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
84:1f:c8:da:8a:7e:26:c9:d1:3f:cb:39:03:54:77:c2:48:a7:
85:6f:c4:45:0:c0:bf:da:72:b7:f2:a3:c8:b1:72:f7:bc:81:a3:
09:67:84:1a:90:f2:82:04:81:af:aa:12:6e:f9:06:82:b7:63:
e4:a1:d2:7b:40:2a:e0:89:c:f:f5:f8:e8:3b:c2:12:40:51:a:
9f:0e:25:06:05:12:64:5b:bf:78:e6:9e:c0:0c:f8:9d:fb:18:
d3:29:ac:0b:f0:7f:27:9a:1a:83:38:7c:f6:ba:6a:cd:8:d8:
5f:50:56:a0:fd:5d:bd:c1:24:e7:12:2a:63:2b:18:6c:b7:87:
bc:7b:0b:af:e0:9d:55:63:83:c:f41:ef:66:10:4a:74:9d:8a:
b9:d:c8:c:df:7a:d8:af:47:ff:b:ca:47:d9:34:a0:3:f6:96:
94:30:d9:82:2b:4b:8b:c5:b8:e0:89:d3:a8:c4:95:8d:08:ba:
b8:f3:f0:ed:d9:40:58:f4:69:57:25:23:b9:c2:84:ba:53:1c:
43:98:2e:5a:d1:68:b3:8c:83:1:e:f0:25:28:80:15:c8:b5:c7:
99:bc:69:9e:80:a1:11:6f:48:52:72:e0:9c:fe:aa:e3:1:f:c:
b7:e9:56:b6:dd:fb:8c:60:09:33:31:ce:33:ff:12:19:7:bdc:
00:11:de:05:be:7b:e1:ed:86:4:b:ed:90:31:c8:f6:7c:01:c:
a9:0:b:00:75:43:4:a:50:9e:89:69:26:72:7d:b4:34:de:29:7b:
15:fd:82:10:7e:10:e9:90:5:15:44:84:15:e8:ad:8f:ac:b9:
36:10:cc:d7:bd:6b:4b:bf:f6:ef:7b:23:cd:e5:e0:16:94:b4:
ac:6d:d1:fd:39:38:ae:43:d6:15:eb:33:c7:59:b9:dc:5b:08:
2d:b9:d6:d0:d4:d3:52:01:ba:f0:e9:52:4:f:e1:be:6:f8:
b8:7:a:36:61:a1:b1:f8:b1:f3:4:b:a4:46:b6:3:f:ab:c:f:c:f9:
97:ec:c:b0:32:c2:ed:12:62:c4:b3:34:8:b:d3:b7:a5:f5:d5:c:
61:ab:d3:40:91:8f:49:46:4:f:d7:9:a:cb:e1:7:f:a4:9:a:e8:2:e:
d4:9:a:b5:32:74:df:a7:99:15:8:b6:b5:f3:61:b:bb:e7:6:c:
09:4:a:1:42:80:a0:67:a5:c:f9:24:f8:43:df:8c:69:d5:d5:
05:c:0:d9:51:52:f2:95:38:50:27:e4:bc:72:bf:72:4:e:9:b:18:
5:a:47:7:a:26:4:a:19:d0:12:0:f:35:85:25:9d:b9:e8:19:12:53:
f8:f0:32:8:b:f6:dc:8:f:e8:51:f:c:a0:4:f:bd:2e:d9:98:56:6:c:
5:c:ae:ce:ab:85:b:f8

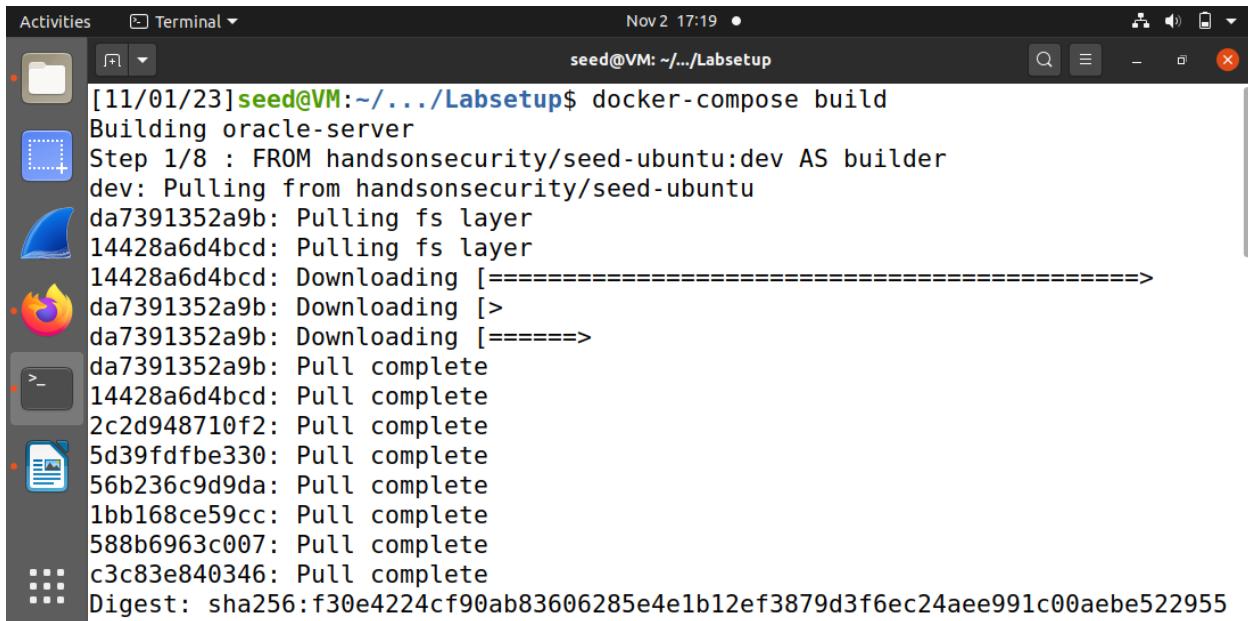
```

Here we can see that it's successful and the alternative names are also present so the generation of the certificate for the server is complete.

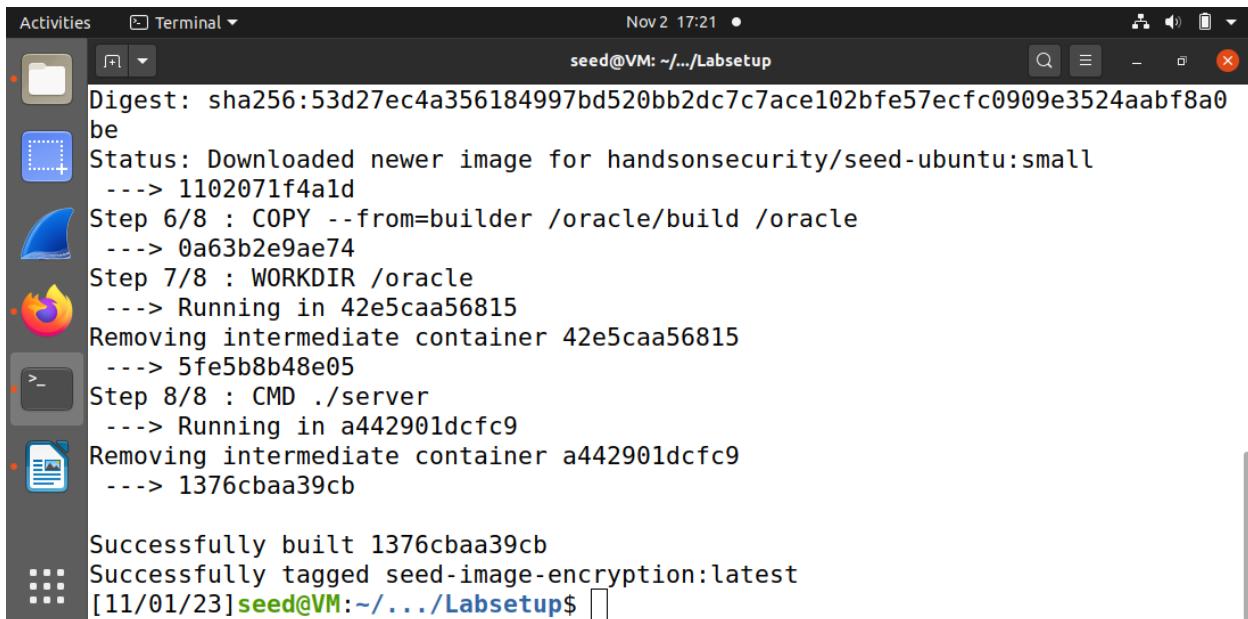
Container Setup:

The following commands used to get the docker container running, these commands were executed in the same folder where the docker-compose.yml file is located.

```
$ docker-compose build      # Command to build the container image  
$ docker-compose up        # Command to start the container
```



```
[11/01/23] seed@VM:~/.../Labsetup$ docker-compose build  
Building oracle-server  
Step 1/8 : FROM handsonsecurity/seed-ubuntu:dev AS builder  
dev: Pulling from handsonsecurity/seed-ubuntu  
da7391352a9b: Pulling fs layer  
14428a6d4bcd: Pulling fs layer  
14428a6d4bcd: Downloading [=====>]  
da7391352a9b: Downloading [>]  
da7391352a9b: Downloading [=====>]  
da7391352a9b: Pull complete  
14428a6d4bcd: Pull complete  
2c2d948710f2: Pull complete  
5d39fdfbe330: Pull complete  
56b236c9d9da: Pull complete  
1bb168ce59cc: Pull complete  
588b6963c007: Pull complete  
c3c83e840346: Pull complete  
Digest: sha256:f30e4224cf90ab83606285e4e1b12ef3879d3f6ec24aee991c00aebe522955
```



```
Digest: sha256:53d27ec4a356184997bd520bb2dc7c7ace102bfe57ecfc0909e3524aabf8a0  
be  
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:small  
---> 1102071f4ald  
Step 6/8 : COPY --from=builder /oracle/build /oracle  
---> 0a63b2e9ae74  
Step 7/8 : WORKDIR /oracle  
---> Running in 42e5caa56815  
Removing intermediate container 42e5caa56815  
---> 5fe5b8b48e05  
Step 8/8 : CMD ./server  
---> Running in a442901dcfc9  
Removing intermediate container a442901dcfc9  
---> 1376cbaa39cb  
  
Successfully built 1376cbaa39cb  
Successfully tagged seed-image-encryption:latest  
[11/01/23] seed@VM:~/.../Labsetup$
```

```
Activities Terminal Nov 2 17:30
seed@VM: ~/.../Labsetup

---> 0a63b2e9ae74
Step 7/8 : WORKDIR /oracle
--> Running in 42e5caa56815
Removing intermediate container 42e5caa56815
--> 5fe5b8b48e05
Step 8/8 : CMD ./server
--> Running in a442901dcfc9
Removing intermediate container a442901dcfc9
--> 1376cbaa39cb

Successfully built 1376cbaa39cb
Successfully tagged seed-image-encryption:latest
[11/01/23]seed@VM:~/.../Labsetup$ docker-compose up
Creating network "net-10.9.0.0" with the default driver
Creating oracle-10.9.0.80 ... done
Attaching to oracle-10.9.0.80
oracle-10.9.0.80 | Server listening on 3000 for known_iv
```

To run shell on the container we use the following commands:

```
$ docker ps
```

```
$ docker exec -it e5086a9e4ec9 /bin/bash
```

```
Activities Terminal Nov 3 19:27
seed@VM: ~

[11/03/23]seed@VM:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            NAMES
e5086a9e4ec9        seed-image-encryption   "/bin/sh -c ./server"   26 hours ago      oracle-10.9.0.80
[11/03/23]seed@VM:~$ docker exec -it e5086a9e4ec9 /bin/bash
root@e5086a9e4ec9:/oracle# 
```

DNS setup:

The following command was used to access and add DNS entries to the /etc/hosts file:

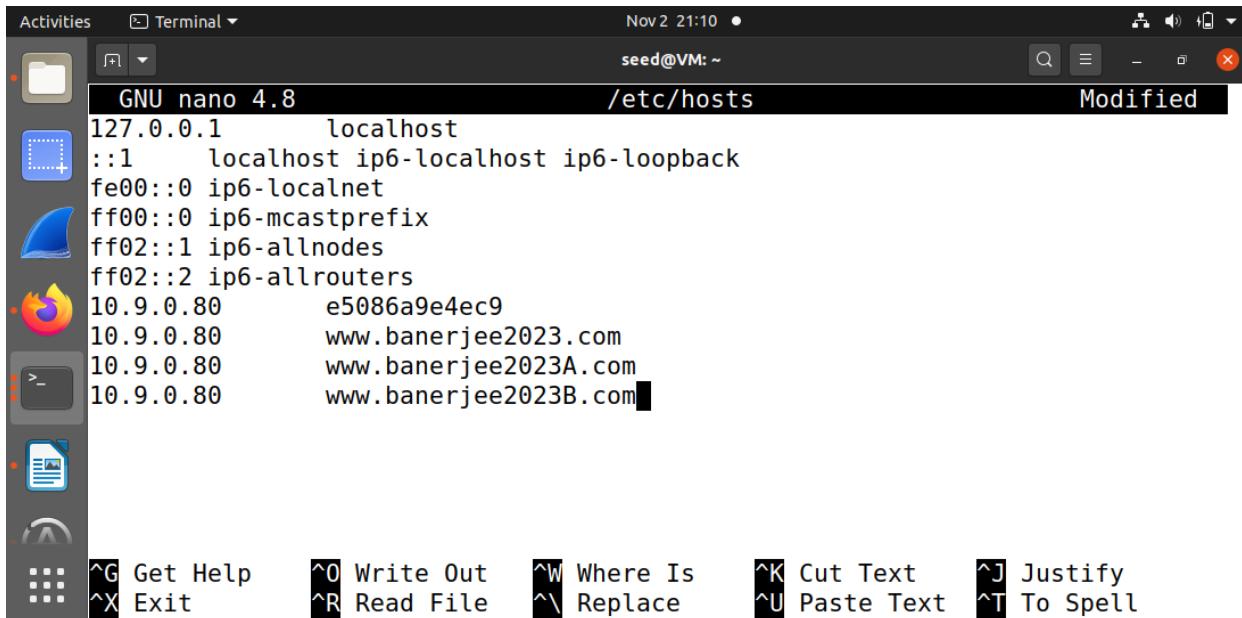
```
$ nano /etc/hosts
```



```
Activities Terminal Nov 2 21:11 • seed@VM: ~
root@e5086a9e4ec9:/oracle# nano /etc/hosts
root@e5086a9e4ec9:/oracle# 
```

The following entries were added:

```
10.9.0.80      www.banerjee2023.com
10.9.0.80      www.banerjee2023A.com
10.9.0.80      www.banerjee2023B.com
```



```
Activities Terminal Nov 2 21:10 • seed@VM: ~
GNU nano 4.8          /etc/hosts          Modified
127.0.0.1      localhost
::1      localhost ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
10.9.0.80      e5086a9e4ec9
10.9.0.80      www.banerjee2023.com
10.9.0.80      www.banerjee2023A.com
10.9.0.80      www.banerjee2023B.com

^G Get Help      ^O Write Out      ^W Where Is      ^K Cut Text      ^J Justify
^X Exit          ^R Read File       ^\ Replace       ^U Paste Text    ^T To Spell
```

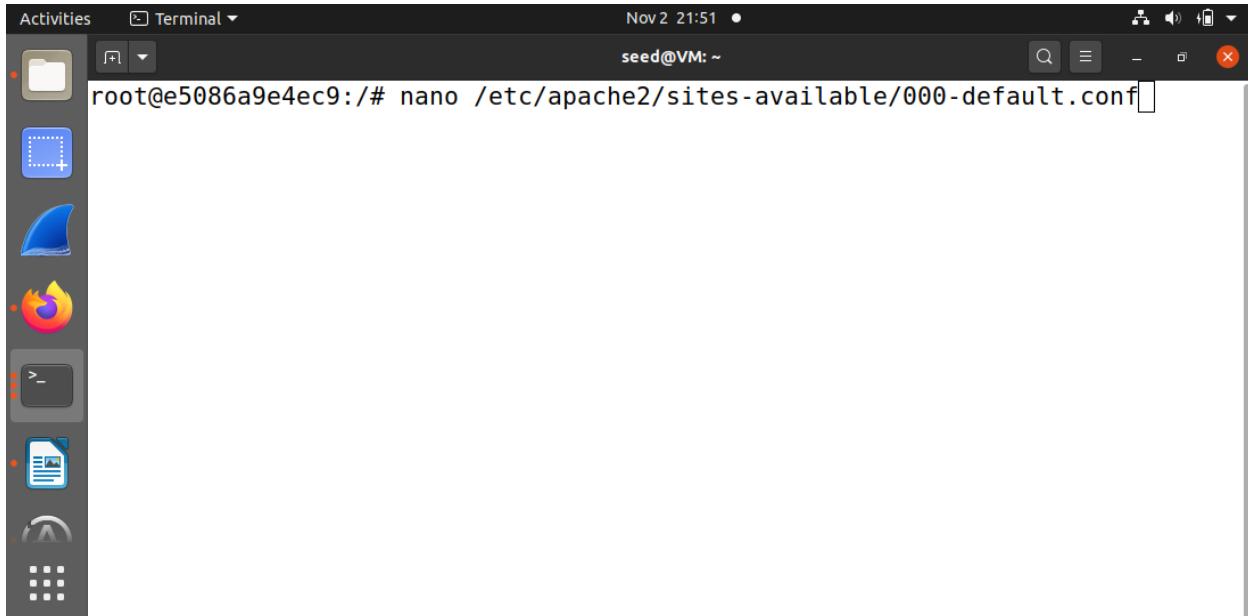
Task 4: Deploying Certificate in an Apache-Based HTTPS Website

In this task, we will see how public-key certificates are used by websites to secure web browsing. We will set up an HTTPS website-based Apache. To create an HTTPS website, we just need to configure the Apache server, so it knows where to get the private key and certificates.

An Apache server can simultaneously host multiple websites. It needs to know the directory where a website's files are stored. This is done via its VirtualHost file, located in the /etc/apache2/sites-available directory. In the sites-available folder, a file named 000-default.conf which contains the information about VirtualHost.

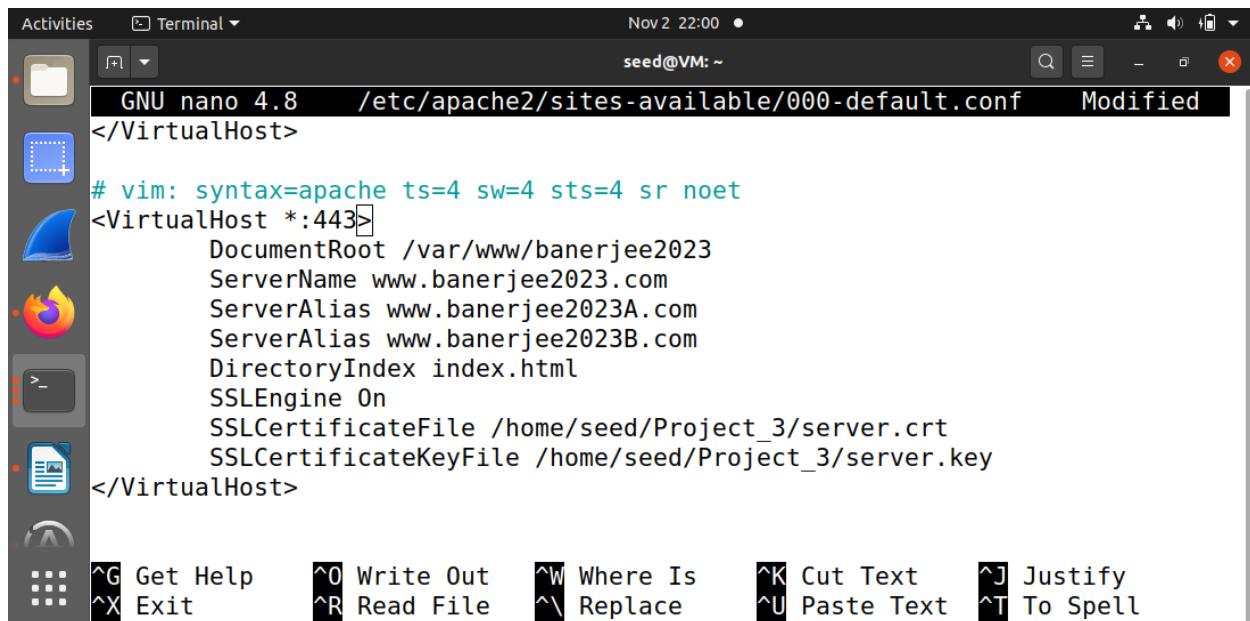
The following command was used to access the .conf file:

```
$ nano /etc/apache2/sites-available/000-default.conf
```

A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark theme with white text. The title bar says "Terminal". The status bar shows "Nov 2 21:51" and "seed@VM: ~". The main area of the terminal shows the command "\$ nano /etc/apache2/sites-available/000-default.conf" being typed. To the left of the terminal is a vertical dock containing icons for various applications like a file manager, terminal, browser, and file editor.

At the end of the file the following VirtualHost entry was added.

```
<VirtualHost *:443>
    DocumentRoot /var/www/banerjee2023
    ServerName www.banerjee2023.com
    ServerAlias www.banerjee2023A.com
    ServerAlias www.banerjee2023B.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /home/seed/Project_3/server.crt
    SSLCertificateKeyFile /home/seed/Project_3/server.key
</VirtualHost>
```



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal title is "seed@VM: ~". The command being run is "nano /etc/apache2/sites-available/000-default.conf". The content of the file shows an Apache VirtualHost configuration for port 443, specifying the DocumentRoot as "/var/www/banerjee2023", ServerName as "www.banerjee2023.com", and ServerAliases for "www.banerjee2023A.com" and "www.banerjee2023B.com". It also sets DirectoryIndex to "index.html", enables SSL Engine, and specifies SSLCertificateFile and SSLCertificateKeyFile paths. The terminal window includes a standard nano editor keymap at the bottom.

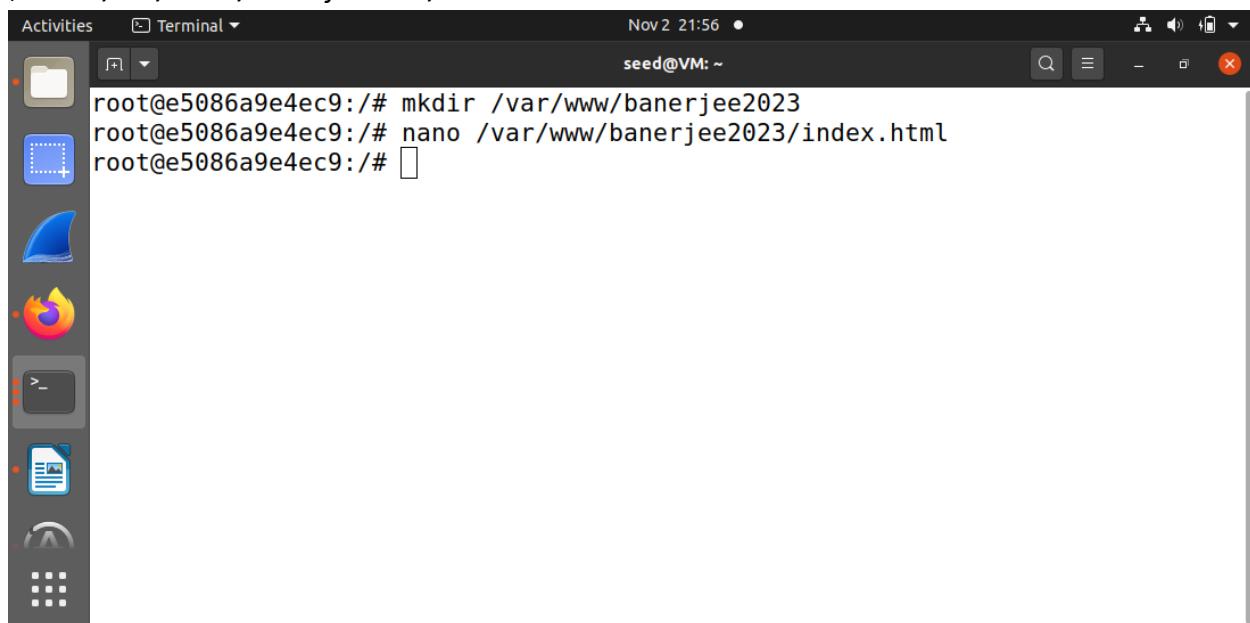
```
GNU nano 4.8      /etc/apache2/sites-available/000-default.conf  Modified
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
<VirtualHost *:443>
    DocumentRoot /var/www/banerjee2023
    ServerName www.banerjee2023.com
    ServerAlias www.banerjee2023A.com
    ServerAlias www.banerjee2023B.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /home/seed/Project_3/server.crt
    SSLCertificateKeyFile /home/seed/Project_3/server.key
</VirtualHost>

^G Get Help      ^O Write Out      ^W Where Is      ^K Cut Text      ^J Justify
^X Exit          ^R Read File       ^\ Replace        ^U Paste Text     ^T To Spell
```

According to the VirtualHost file the index.html file will be shown which is stored in the directory /var/www/banerjee2023. Now we need to create such a file in the mentioned directory. The commands are given below:

```
$ mkdir /var/www/banerjee2023          # to create the directory.
$ nano /var/www/banerjee2023/index.html    # to create a html file and edit it.
```

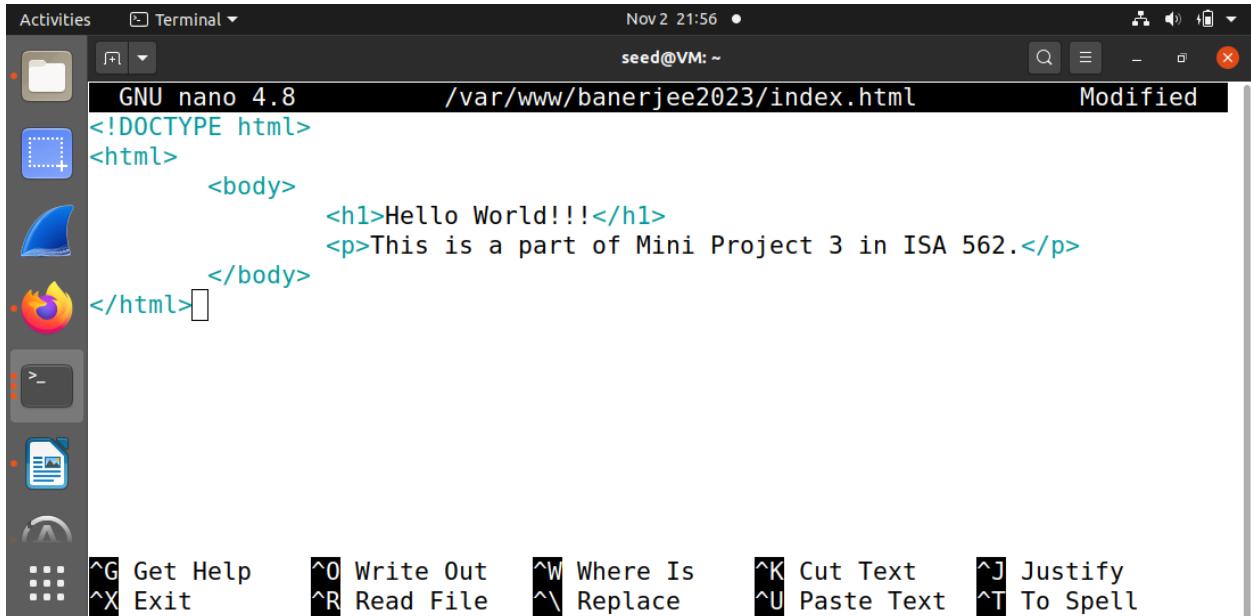


A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal title is "seed@VM: ~". The user runs two commands: "mkdir /var/www/banerjee2023" and "nano /var/www/banerjee2023/index.html". The terminal window includes a standard nano editor keymap at the bottom.

```
root@e5086a9e4ec9:/# mkdir /var/www/banerjee2023
root@e5086a9e4ec9:/# nano /var/www/banerjee2023/index.html
root@e5086a9e4ec9:/#
```

The following code was placed inside the html file, and this will be the content of our page:

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Hello World!!!</h1>
    <p>This is a part of Mini Project 3 in ISA 562</p>
  </body>
</html>
```



A screenshot of a terminal window titled "Terminal". The title bar shows "Nov 2 21:56" and "seed@VM: ~". The main area of the terminal displays the content of an HTML file being edited with the nano 4.8 text editor. The file contains the following code:

```
GNU nano 4.8      /var/www/banerjee2023/index.html  Modified
<!DOCTYPE html>
<html>
  <body>
    <h1>Hello World!!!</h1>
    <p>This is a part of Mini Project 3 in ISA 562.</p>
  </body>
</html>
```

The terminal window has a dark theme and includes a dock on the left with various icons for applications like a file manager, terminal, browser, and file viewer. A status bar at the bottom provides keyboard shortcuts for various functions.

So, to make this website work, we need to. Enable Apache's ssl module and then enable this site. The commands used are as follows:

```
$ a2enmod ssl          # Enable the SSL module
```

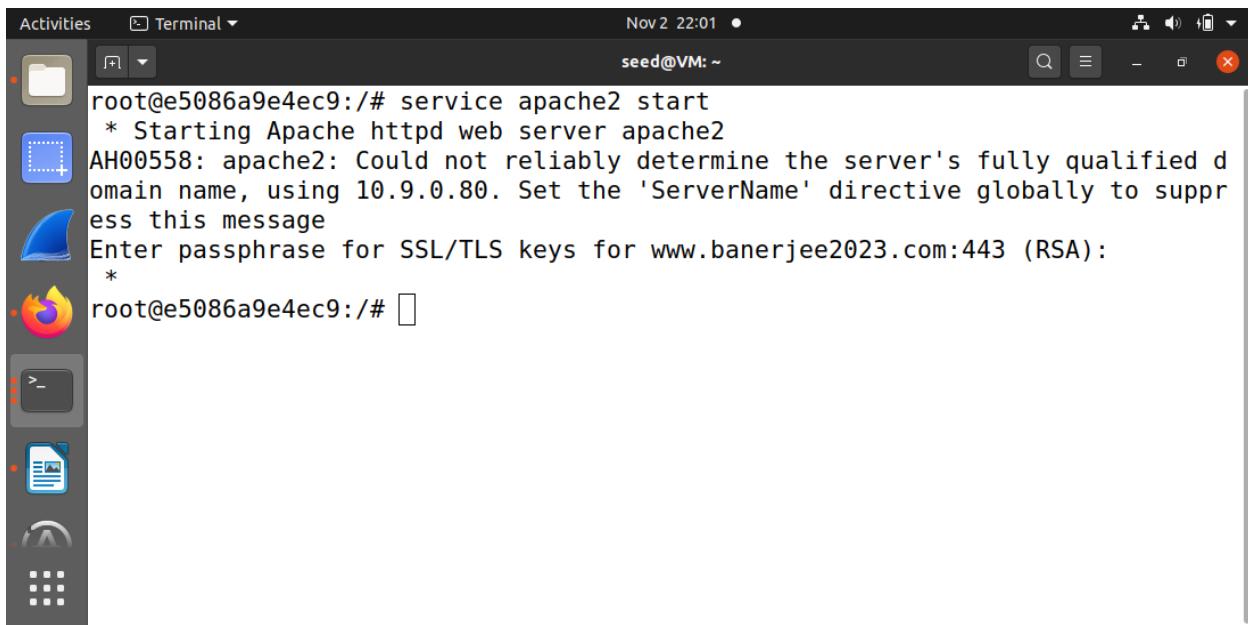
```
Activities Terminal Nov 2 21:56 • seed@VM: ~
root@e5086a9e4ec9:/# a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  service apache2 restart
root@e5086a9e4ec9:/#
```

```
$ apachectl configtest      # Test the Apache configuration for errors
$ a2ensite default-ssl      # Enable the site
```

```
Activities Terminal Nov 2 22:00 • seed@VM: ~
root@e5086a9e4ec9:/# apachectl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 10.9.0.80. Set the 'ServerName' directive globally to suppress this message
Syntax OK
root@e5086a9e4ec9:/# a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  service apache2 reload
root@e5086a9e4ec9:/#
```

Starting the Apache server: The Apache server is not automatically started in the container, because of the need to type the password to unlock the private key.

```
$ service apache2 start      # Start the server
```

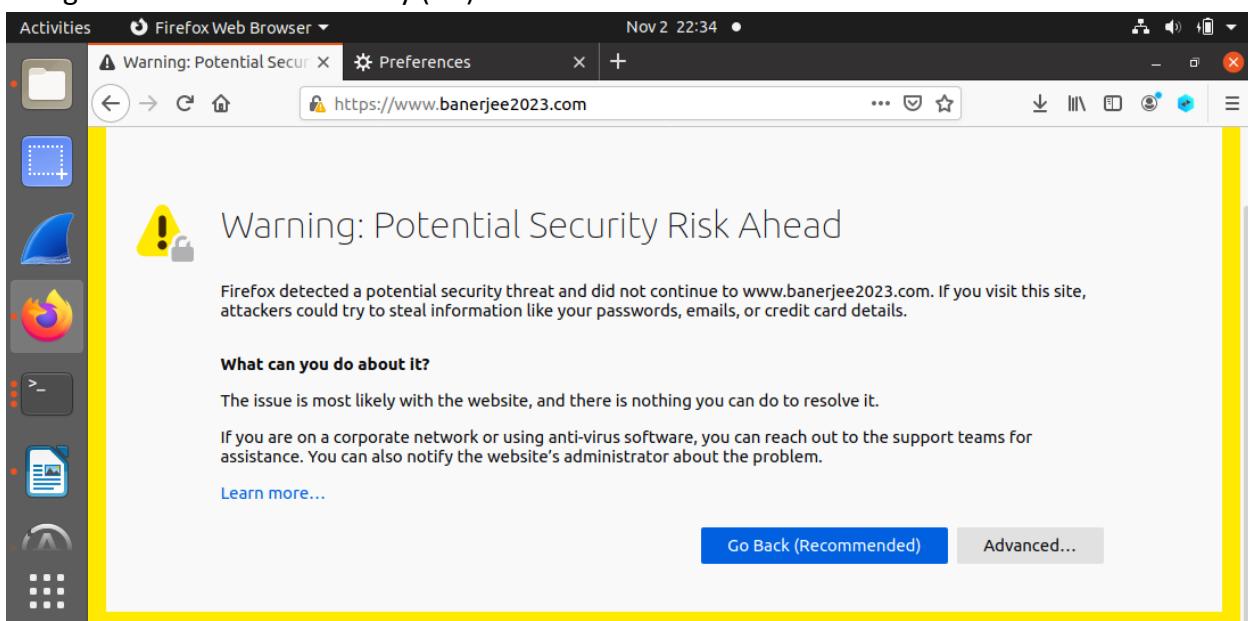


A screenshot of a Linux desktop environment showing a terminal window. The terminal window title is "seed@VM: ~". The terminal content shows the command "service apache2 start" being run, followed by an error message about determining the server's fully qualified domain name, and a prompt for entering a passphrase for SSL/TLS keys.

```
root@e5086a9e4ec9:/# service apache2 start
 * Starting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 10.9.0.80. Set the 'ServerName' directive globally to suppress this message
Enter passphrase for SSL/TLS keys for www.banerjee2023.com:443 (RSA):
 *
root@e5086a9e4ec9:/#
```

When Apache starts, it needs to load the private key for each HTTPS site. Our private key is encrypted, so Apache will ask us to type the password for decryption. Inside the container, the password used for banerjee2023 is dees. Once everything is set up properly, we can browse the web site.

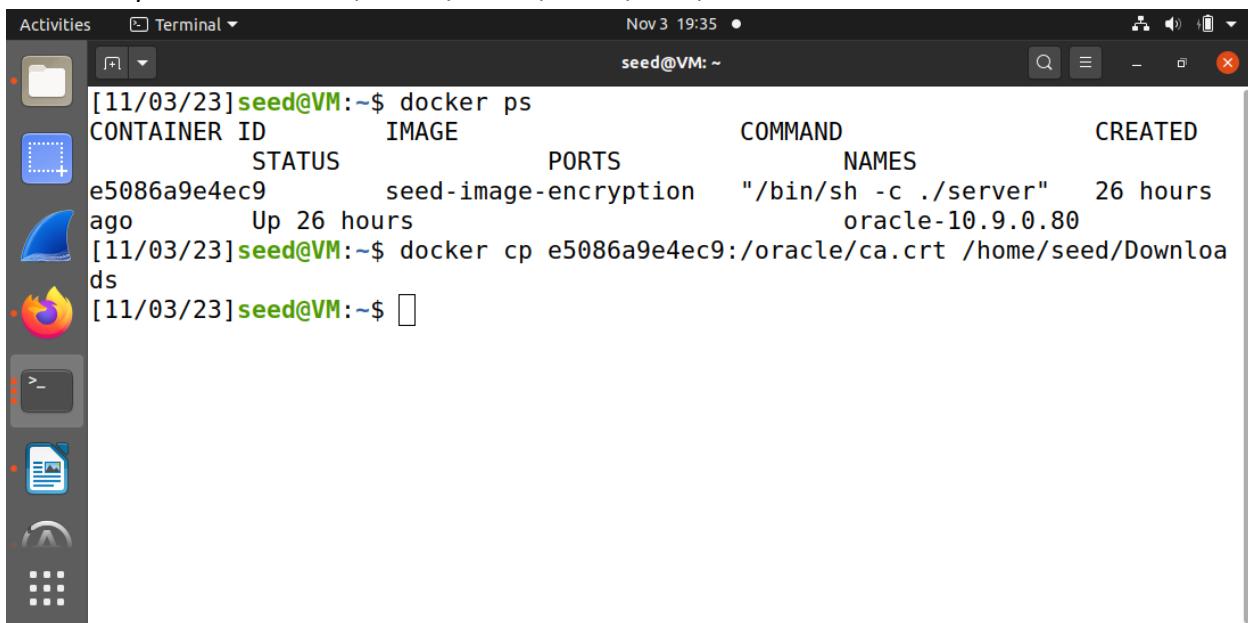
Browsing the website: When we point the browser to your web server, (<https://www.banerjee2023.com>) we are not able to access the contents of the site. A “Warning” appears saying “Potential Security Risk Ahead”. The reason being, the site is using a self-signed certificate, which is not automatically trusted by browser because it isn't signed by a recognized certificate authority (CA).



Now to make the warning go away we need to load the certificate to the browser and make the browser trust the new certificate for this website. For this we first need to get the certificate from inside the container into the host directory. We use the following command to do so:

Command executed in Host CLI.

```
$ docker cp e5086a9e4ec9:/oracle/ca.crt /home/seed/Downloads
```

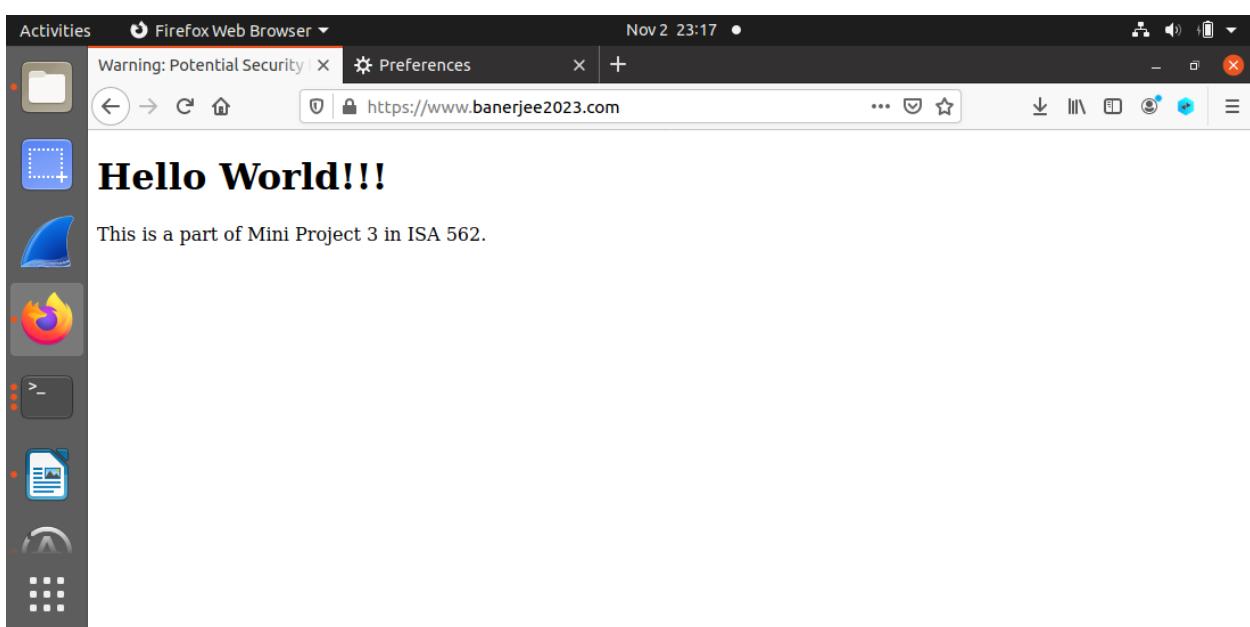
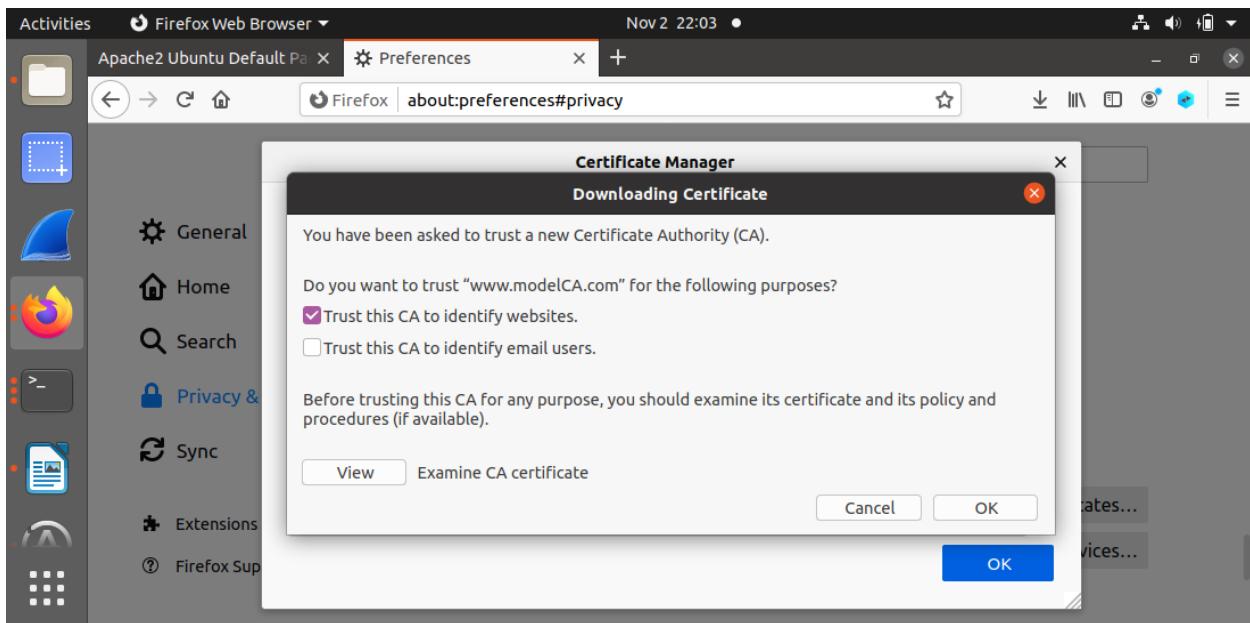


```
[11/03/23]seed@VM:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
e5086a9e4ec9        seed-image-encryption   "/bin/sh -c ./server"   26 hours ago      Up 26 hours          oracle-10.9.0.80
[11/03/23]seed@VM:~$ docker cp e5086a9e4ec9:/oracle/ca.crt /home/seed/Downloads
[11/03/23]seed@VM:~$
```

This command copies the ca.crt file from this folder “oracle” which is inside the container to “Downloads” folder which is in Host.

Now that we have the certificate in Host we do this because the browser can only access the files in Host and not inside the container.

To load the certificate, we enter the following command in the search bar, “about:preference#privacy”, this takes us to the Privacy and Security section of the settings menu, which contains the “Certificates” section, in this section we click on “View Certificates...” Which then gives us the option to scroll through all the certificates present and gives us an option to import new certificate. So, we click “Import” then navigate to the directory where our certificate, in this case “ca.crt”, is located and select it and click on the green import button. This will open a “Downloading Certificate” tab where we need to select “Trust this CA to identify websites.” And click OK. We will see that our certificate is now in Firefox’s list of accepted certificates. Then we can reload the website and see that the content of <https://www.banerjee2023.com> can be accessed. And we can also see that the connection is secure and verified by Model CA LTD.



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

In this task, we will show how PKI can defeat Man-In-The-Middle (MITM) attacks. The goal of this task is to help students understand how PKI can defeat such MITM attacks. In the task, we will emulate an MITM attack, and see how exactly PKI can defeat it. We will select a target website first. In this task, we use www.instagram.com as the target website.

Step 1 - Setting up the malicious website:

In Task 4, we have already set up an HTTPS website. We will use the same Apache server to impersonate www.instagram.com. To achieve that, we have followed the instruction in Task 4 to add a VirtualHost entry to Apache's SSL configuration file: the ServerName should be www.instagram.com, but the rest of the configuration will be the same as that used in Task 4.

In this step, we configured the DNS on the attacker machine, which in our case is the docker container using the following command:

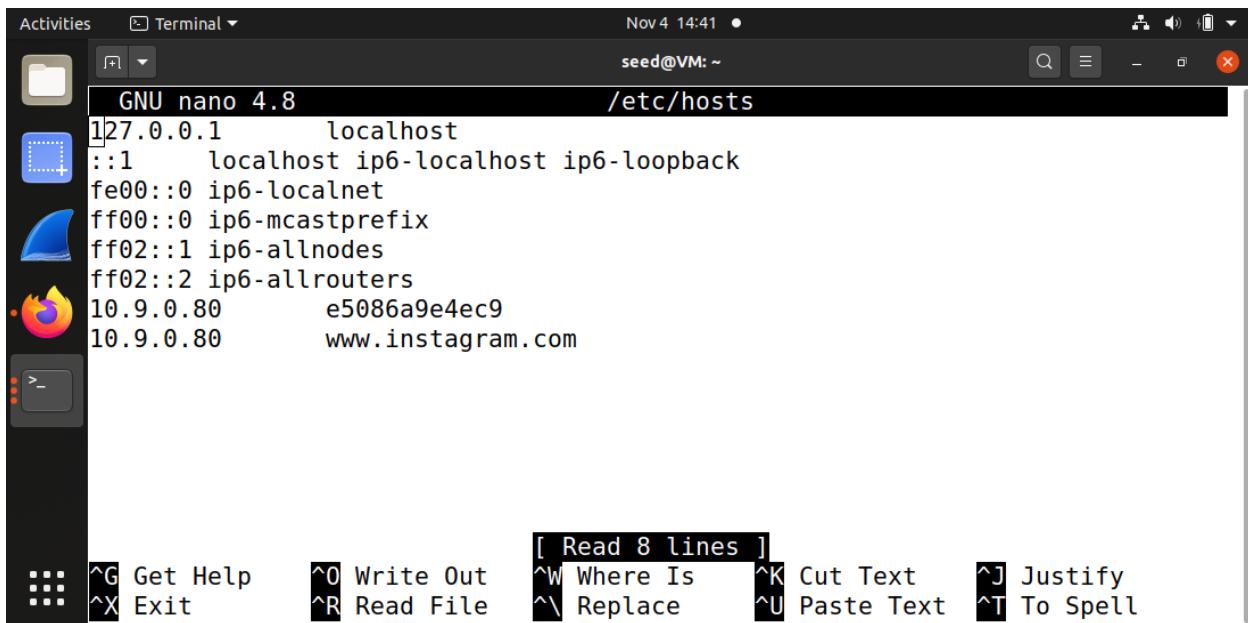
```
$ nano /etc/hosts
```



A screenshot of a Linux desktop environment showing a terminal window. The terminal window title is "Terminal" and the user is "seed@VM: ~". The command "nano /etc/hosts" has been entered and is displayed on the screen. The terminal window is part of a desktop interface with a dock containing icons for various applications like a file manager, terminal, browser, and file browser.

And add our target website name as:

10.9.0.80 www.instagram.com



```
GNU nano 4.8          /etc/hosts
127.0.0.1      localhost
::1      localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.9.0.80      e5086a9e4ec9
10.9.0.80      www.instagram.com
```

[Read 8 lines]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell

Then we add the website as a VirtualHost entry in 000-default.conf using the following command:

```
$ nano /etc/apache2/sites-available/000-default.conf
```



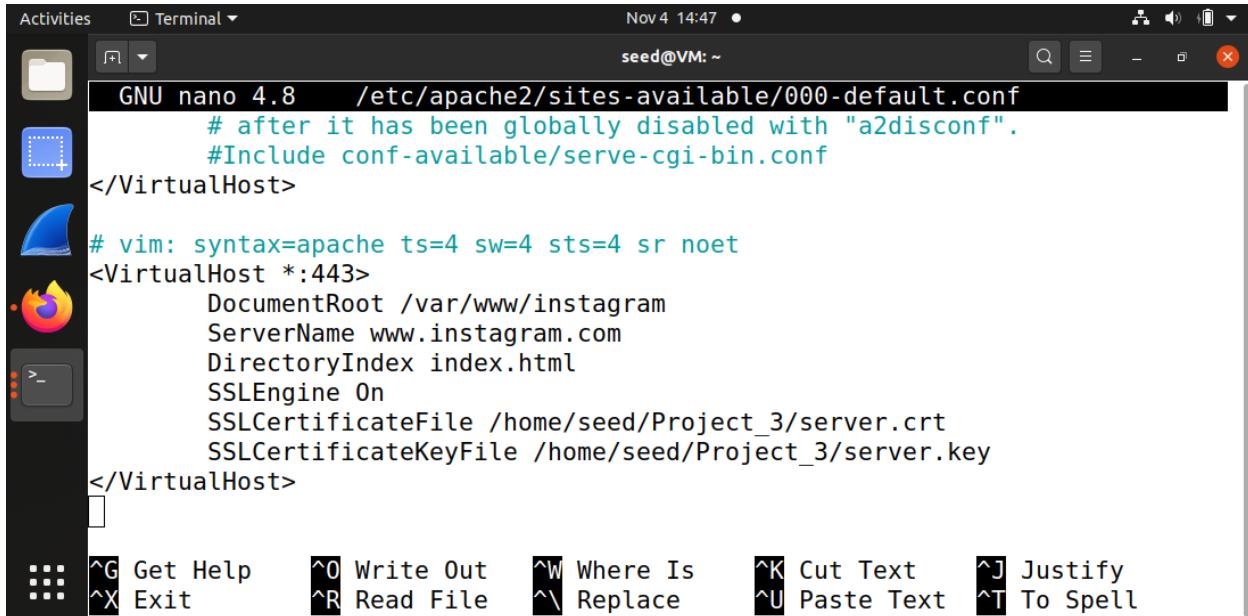
```
root@e5086a9e4ec9:/oracle# nano /etc/apache2/sites-available/000-default.conf
root@e5086a9e4ec9:/oracle#
```

The following VirtualHost entry was add in the end on the file:

```
<VirtualHost *:443>
```

```
DocumentRoot /var/www/Instagram
ServerName www.instagram.com
DirectoryIndex index.html
SSLEngine On
```

```
SSLCertificateFile /home/seed/Project_3/server.crt  
SSLCertificateKeyFile /home/seed/Project_3/server.key  
</VirtualHost>
```



```
GNU nano 4.8      /etc/apache2/sites-available/000-default.conf
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

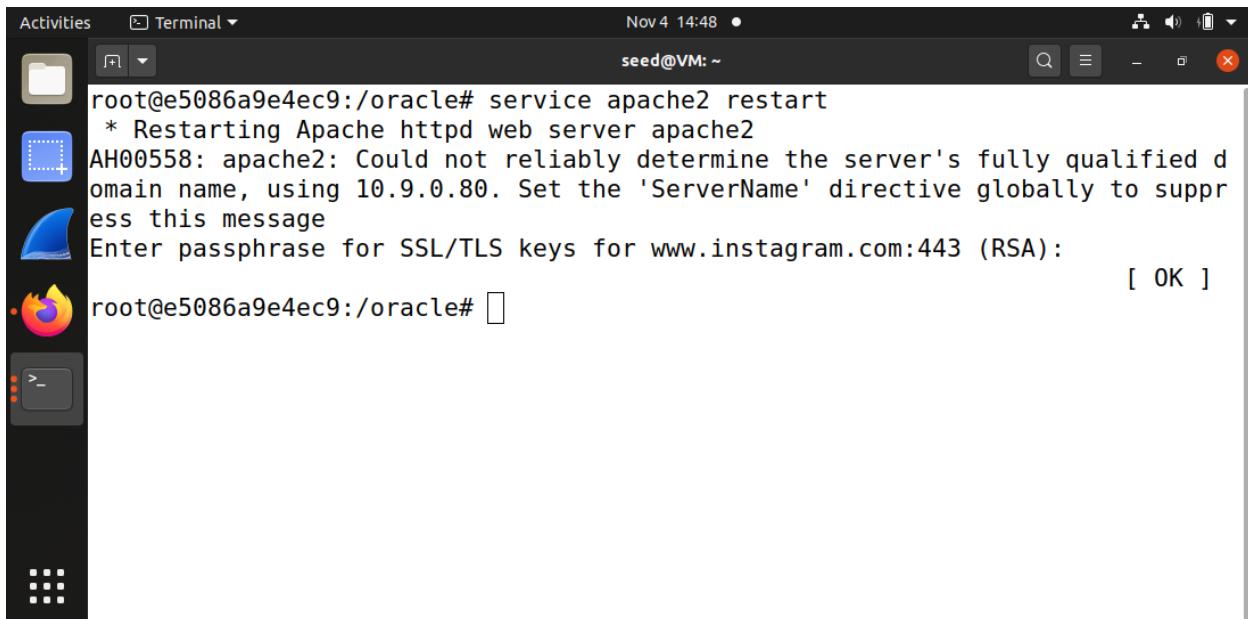
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
<VirtualHost *:443>
    DocumentRoot /var/www/instagram
    ServerName www.instagram.com
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /home/seed/Project_3/server.crt
    SSLCertificateKeyFile /home/seed/Project_3/server.key
</VirtualHost>

```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell

Then we use restart the Apache server to save the changes:

```
$ service apache2 restart
```



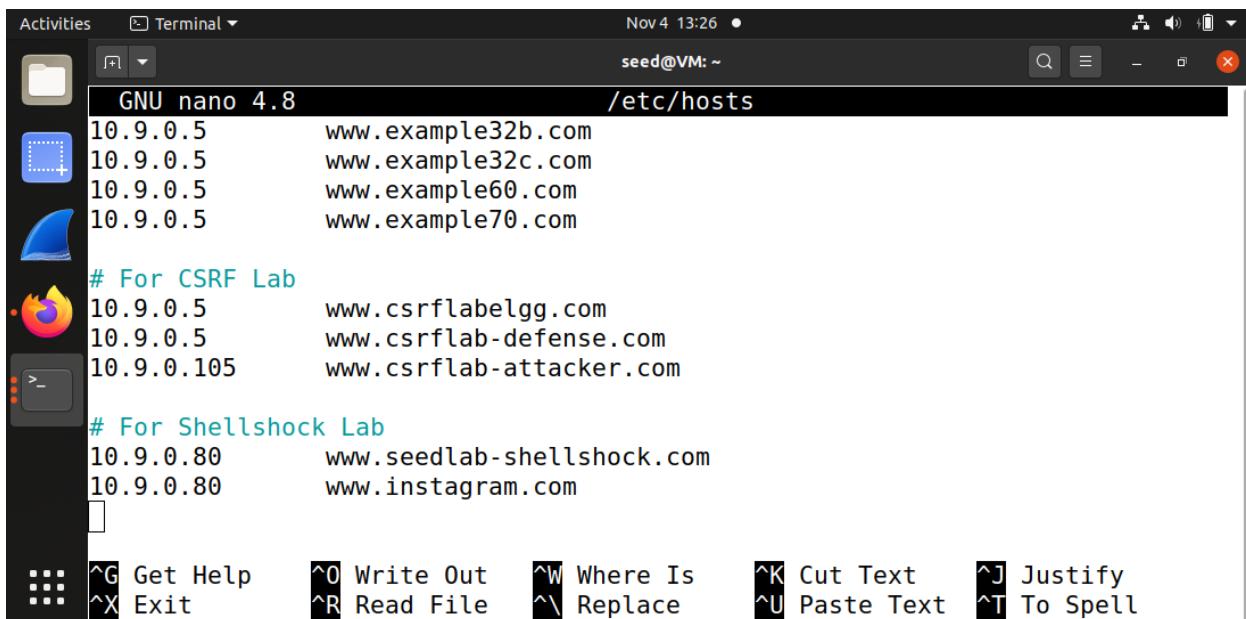
```
root@e5086a9e4ec9:/oracle# service apache2 restart
 * Restarting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 10.9.0.80. Set the 'ServerName' directive globally to suppress this message
Enter passphrase for SSL/TLS keys for www.instagram.com:443 (RSA):
[ OK ]
root@e5086a9e4ec9:/oracle# 
```

Step 2 - Becoming the man in the middle:

In this task, we simulate the attack-DNS approach. Instead of launching an actual DNS cache poisoning attack, we simply modify the victim's machine's /etc/hosts file to emulate the result

of a DNS cache poisoning attack by mapping the hostname www.instagram.com to our malicious web server.

10.9.0.80 www.instagram.com



```
Activities Terminal Nov 4 13:26 seed@VM: ~
GNU nano 4.8 /etc/hosts
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.instagram.com

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell
```

After this step, we tried accessing our target website through the browser, but our secure connection failed. So, we copied the ca certificate from the docker container i.e., the attacker machine and placed it in the Host i.e., victims' machine and loaded the certificate into the browser and added it as an authority, by following the steps described in the previous task.

Following command was used to copy ca.crt for container to host machine:

```
$ docker cp e5086a9e4ec9:/oracle/ca.crt /home/seed/Downloads
```



```
Activities Terminal Nov 4 17:11 seed@VM: ~
[11/04/23] seed@VM:~$ docker cp e5086a9e4ec9:/oracle/ca.crt /home/seed/Downloads
```

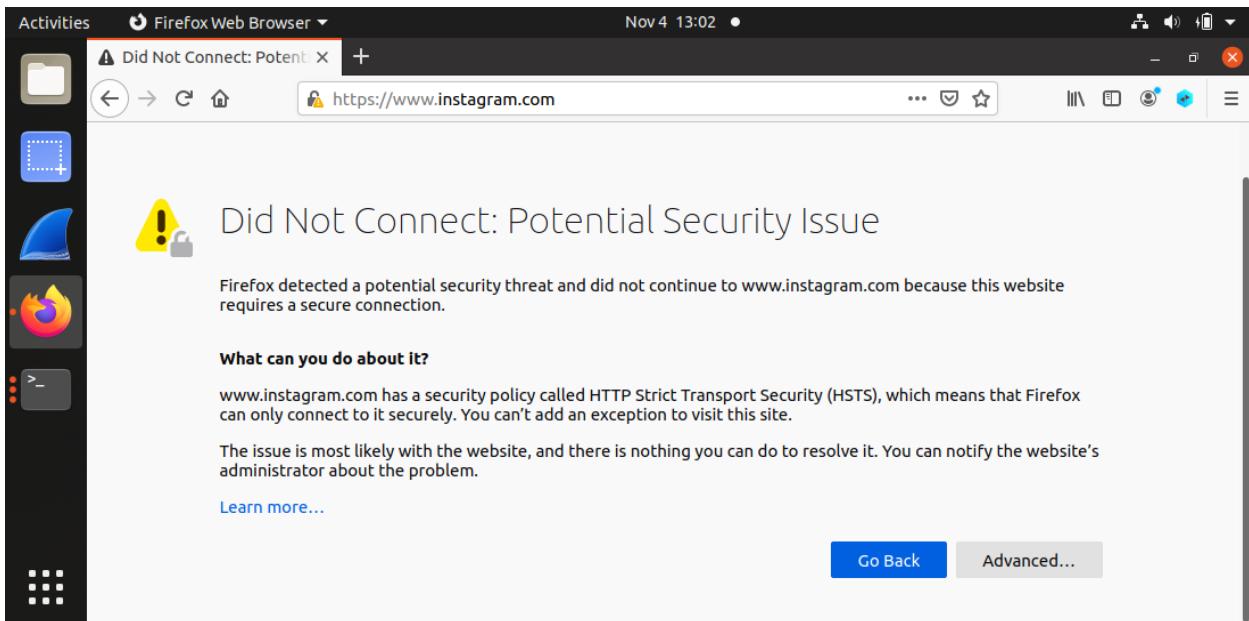
Step 3 - Browse the target website:

With everything set up, now visit the target real website.

We observe that, “Did Not Connect: Potential Security Risk” which is a warning that our browser gives us because the SSL certificate does not match the domain name www.instagram.com.

The browser alerts the user that there is some issue with the website’s security certificate.

This is the first line of defense in PKI.



Team Member:

Joel Sadanand Samson: Task 1,2 and 3
G01352483

Abhijeet Amitava Banerjee: Task 4 and 5
G01349260