



Applied Information Security

Assignment-3

Submitted by:

Muhammad Osama Khalid

20I-1955 (MS-CNS)

Section: 1

Table of Contents

Part-1 Lab Setup.....	1
Part-2 Public-Key Infrastructure Lab.....	4
Task-1 Becoming the Certificate Authority (CA)	4
Step-1 Configuring the OpenSSL	4
Step-2 Certificate Authority (CA)	5
Task-2 Creating a Certificate for osamakhalid.com	7
Step-1 Generate public/private key pair	7
Step-2 Generate a Certificate Signing Request (CSR)	9
Step-3 Generating Certificates.....	10
Task-3 Deploying Certificate in an HTTPS Web Server	12
Step-1 Configuring DNS.....	12
Step-2 Configuring the web server	13
Step-3 Getting the browser to accept our CA certificate	14
Step-4 Testing our HTTPS website	16
Task-4 Deploying Certificate in an Apache-Based HTTPS Website	20
Task-5 Launching a Man-In-The-Middle (MITM) Attack	22
Step-1 Setting up the malicious website	22
Step-2 Becoming the man in the middle	24
Step-3 Browse the target website	24
Task-6 Launching a Man-In-The-Middle Attack with a Compromised CA	26

List of Figures

Figure 1: Creating the Virtual Machine	1
Figure 2: Creating the Virtual Machine	1
Figure 3: Creating new Nat Adapter from Virtual Box Setting.....	2
Figure 4: Adding Nat adapter in Virtual Machine setting	2
Figure 5: Go to Virtual Machine Shared Folder Setting	3
Figure 6: Mount the shared folder to the home directory folder	3
Figure 7: Changing the hostname of the virtual machine	3
Figure 8: Copy the OpenSSL configuration file in the current directory.....	4
Figure 9: Directories to create specified in OpenSSL config file	4
Figure 10: Creating Directories specified in OpenSSL config file	5
Figure 11: Generating certificate for CA	5
Figure 12: Certificate generated successfully	5
Figure 13: Contents of ca.crt.....	6
Figure 14: Contents of ca.crt.....	6
Figure 15: Contents of ca.key	7
Figure 16: Generating the Public/Private key	7
Figure 17: Public/Private key generated and stored in server.key.....	8
Figure 18: Contents of server.key	8
Figure 19: Contents of server.key	8
Figure 20: Generating the CSR	9
Figure 21: CSR generated successfully.....	9
Figure 22: Contents of server.csr	9
Figure 23: Changing the policy in openssl.cnf.....	10
Figure 24: Changing the policy in openssl.cnf.....	10
Figure 25: Generating the certificate for the company	11
Figure 26: Generating the certificate for the company	11
Figure 27: Adding the website details in /etc/hosts file	12
Figure 28: Confirming the changes	12
Figure 29: Starting the webserver.....	13
Figure 30: Visiting the website.....	13
Figure 31: Adding CA's certificate into the browser	14
Figure 32: Adding CA's certificate into the browser	14
Figure 33: Adding CA's certificate into the browser	15
Figure 34: CA's certificate added to the browser successfully	15
Figure 35: Webpage loads successfully	16
Figure 36: Opening the server.pem file	16
Figure 37: Changing a single bit from 7A to 7B.....	17
Figure 38: Changing a single bit from 7A to 7B.....	17
Figure 39: Run the server again	17
Figure 40: Webpage shows error.....	18
Figure 41: Browser throwing the "Connection not secure" error	18
Figure 42: Viewing the Details of the certificate used	19

Figure 43: Certificate issued to osamakhalid.com not localhost	19
Figure 44: Combine the certificate and key into one file.....	20
Figure 45:Opening the 000-default.conf file.....	20
Figure 46: Adding the entry in 000-default.conf file.....	20
Figure 47: Opening the default-ssl.conf file.....	21
Figure 48: Adding the entry in default-ssl.conf file.....	21
Figure 49: Creating the website files	21
Figure 50: Enabling the SSL for website.....	22
Figure 51: Website loads successfully.....	22
Figure 52: Opening the 000-default.conf file.....	23
Figure 53: Adding the entry in 000-default.conf file.....	23
Figure 54: Opening the default-ssl.conf file.....	23
Figure 55: Adding the entry in default-ssl.conf file.....	23
Figure 56: opening the /etc/hosts file	24
Figure 57: Attacker Adding the malicious entry into /etc/hosts file	24
Figure 58: Enabling the SSL for website.....	24
Figure 59:Visiting the website.....	25
Figure 60: Certificate issued to osamakhalid.com not redhat.com.....	25
Figure 61: Attacker Generating public private key pairs from CA's private key.....	26
Figure 62: Attacker generating CSR request	26
Figure 63: Attacker generating certificate from CSR using CA's key and certificate	27
Figure 64: Attacker generating certificate from CSR using CA's key and certificate	27
Figure 65: Attacker Combine the certificate and key into one file	28
Figure 66: Attacker Restart the Apache Server.....	28
Figure 67:Target website redhat.com loads successfully and display the contents of osamakhalid.com .	28

Part-1 Lab Setup

In this part, I am going to set up the Attack Lab in which I will perform the SQL injection and Cross-Site Scripting Attacks in the next parts.

1. First, I download the Seed Virtual Machine from the SEED Lab Website and then open the virtual box and click on the add button. Then I add details of my VM and click Next. After that, I set the memory size (RAM) to 2 GB and click next. After that from the options, I select “Use an existing virtual hard disk file” and go to the path where I download the Seed VM and select that. After that, I click create to create my VM.

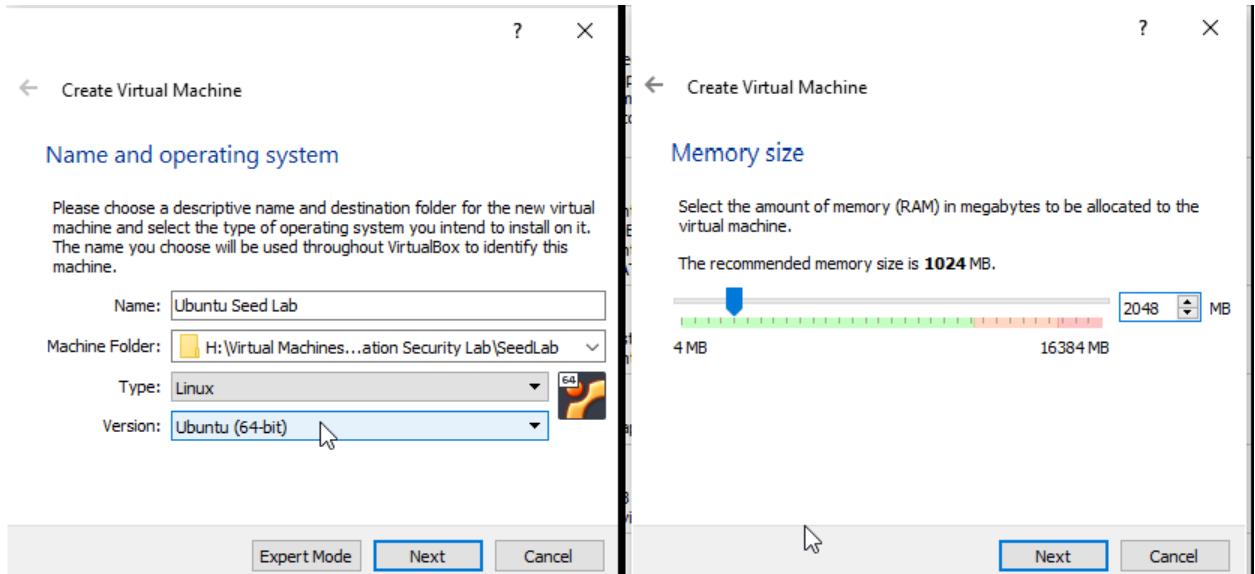


Figure 1: Creating the Virtual Machine

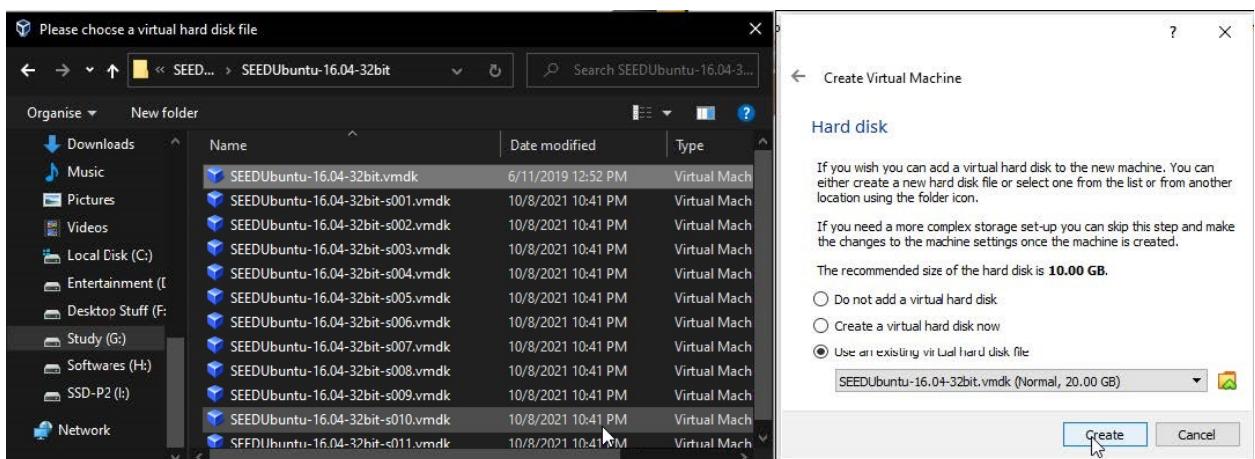


Figure 2: Creating the Virtual Machine

2. After that go to the virtual box setting and select Network from the tab on the left panel. Then I click on the “+” button to create a new NAT Networks adapter. After that, I open the virtual machine setting, select Network from the tab on the left panel, and staying on Adapter 1 and under enable network adapter I click on the “Attah to” drop-down menu and select NAT Network adapter that I just created and click ok.

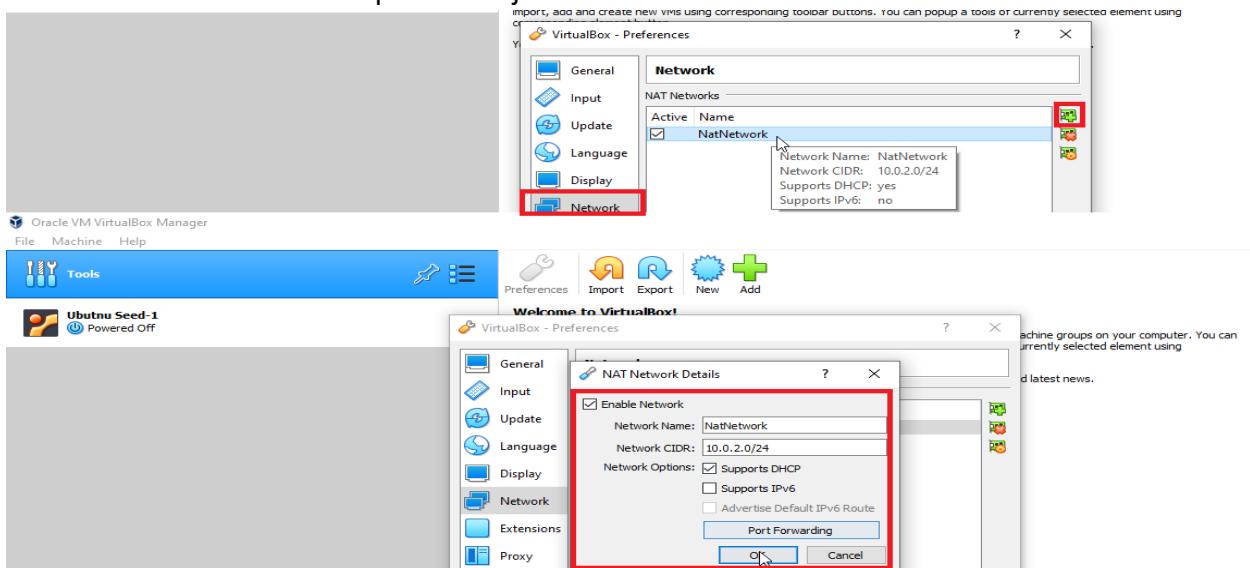


Figure 3: Creating new Nat Adapter from Virtual Box Setting

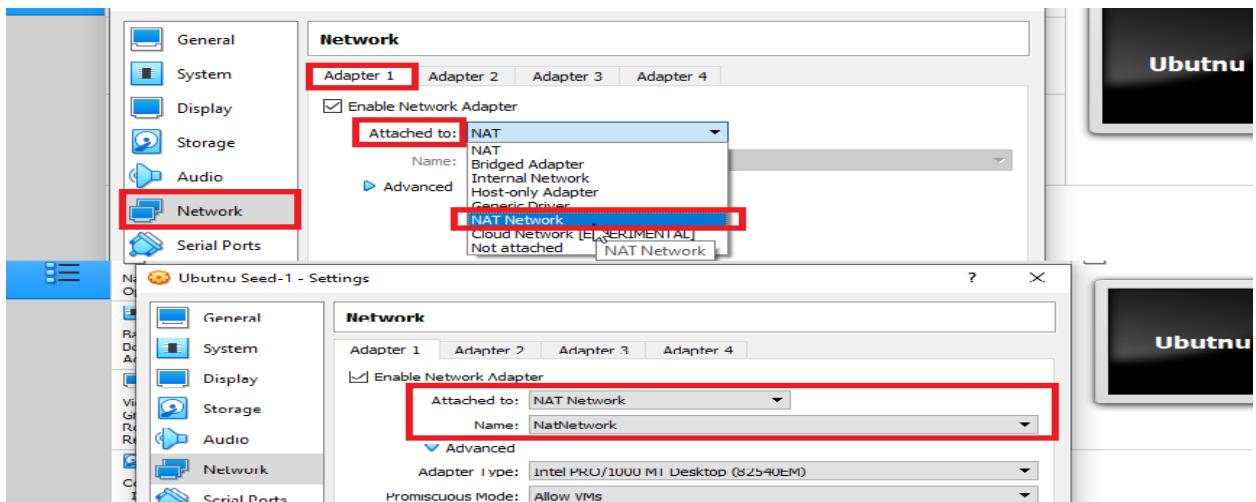


Figure 4: Adding Nat adapter in Virtual Machine setting

3. After that, I go to the VM settings and click on the shared folder to create the shared folder between the host and the virtual machine so that every time I put something in that folder, it can be accessed by both VM and host machine.

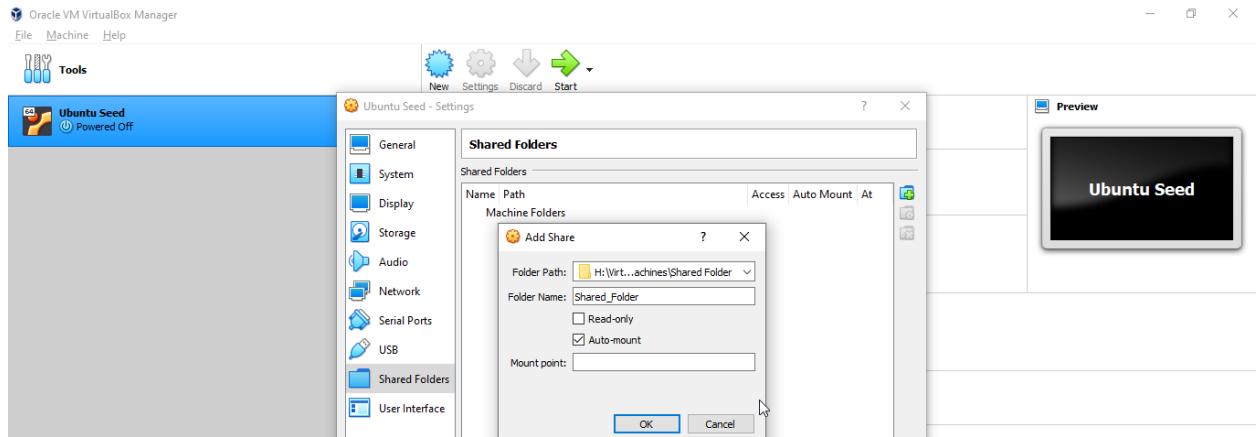


Figure 5: Go to Virtual Machine Shared Folder Setting

4. After that, I start the virtual machine, and then to mount the shared folder to the home directory of the Virtual machine I create a folder called "Share" in the home directory and then mount the shared folder to this "Share" folder.

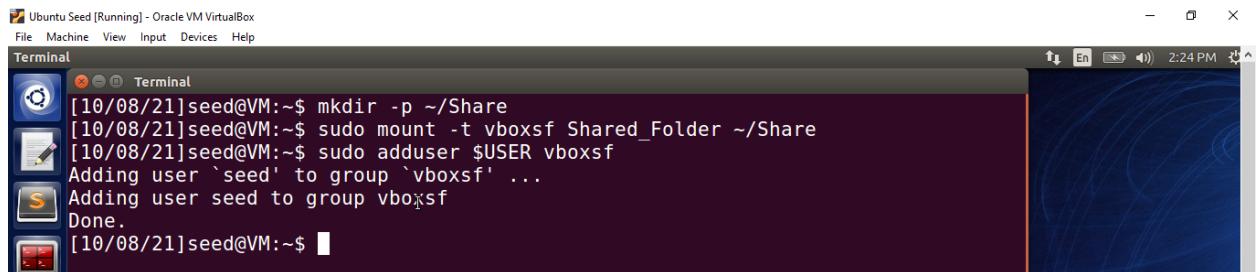


Figure 6: Mount the shared folder to the home directory folder

5. After that, I change the hostname of the virtual machine to my registration number to maintain the authenticity of the work.

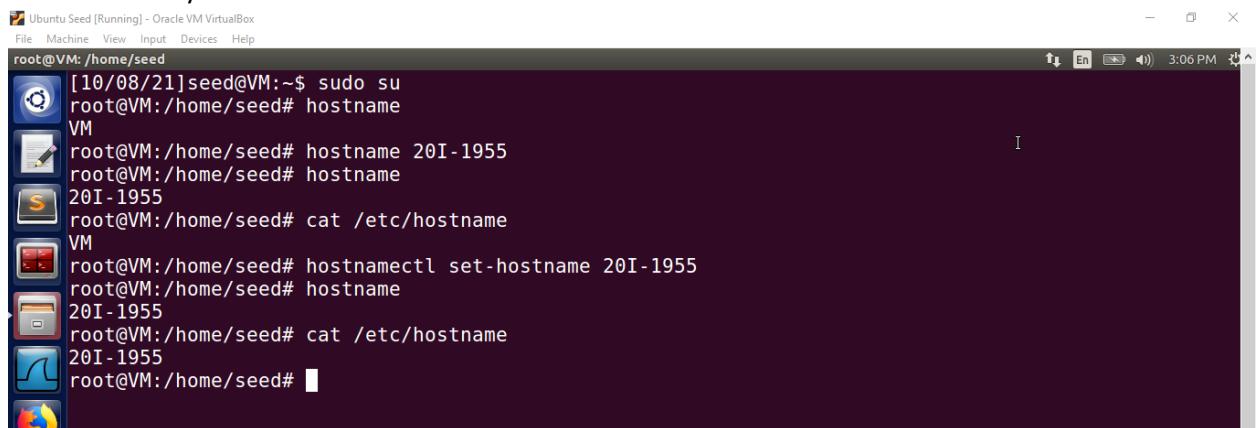


Figure 7: Changing the hostname of the virtual machine

Part-2 Public-Key Infrastructure Lab

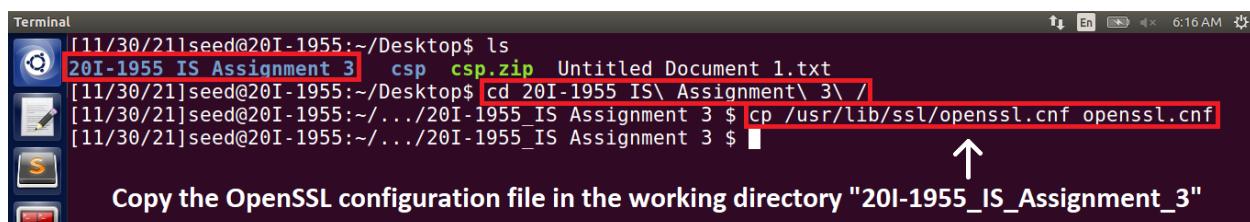
In this lab, I have to create a digital certificate, deploy it on HTTPS Webserver and Apache-based HTTPS Website and then launch Man-In-The-Middle Attack on website and Certificate Authority.

Task-1 Becoming the Certificate Authority (CA)

In this task, I have to become the root certificate authority and then use this to generate the certificates for certificate authority. Now to start the task first I have to configure the OpenSSL.

Step-1 Configuring the OpenSSL

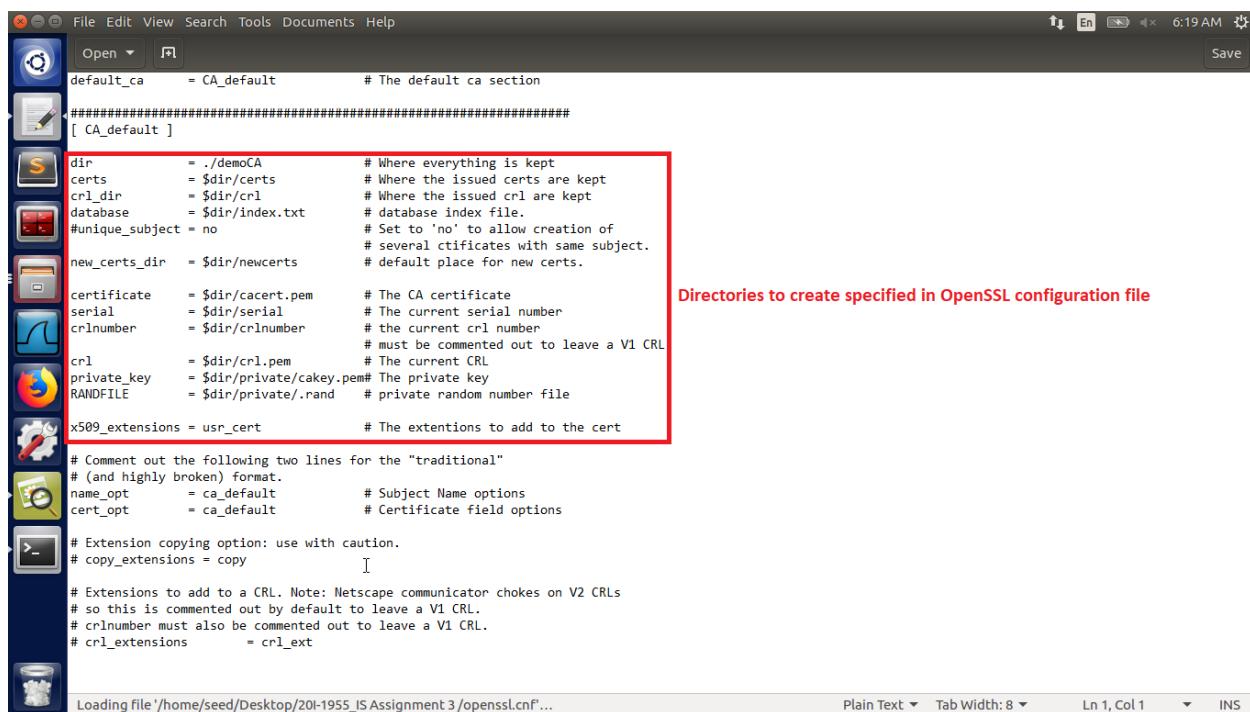
To configure the OpenSSL to create a certificate, I have to copy the OpenSSL configuration file from the "/usr/lib/ssl/openssl.cnf" location into the current working directory and then create the sub-directories specify in the configuration file.



```
[11/30/21]seed@20I-1955:~/Desktop$ ls
20I-1955_IS_Assignment_3  csp  csp.zip  Untitled Document 1.txt
[11/30/21]seed@20I-1955:~/Desktop$ cd 20I-1955_IS\ Assignment\ 3\ /
[11/30/21]seed@20I-1955:~/.../20I-1955_IS\ Assignment\ 3 $ cp /usr/lib/ssl/openssl.cnf openssl.cnf
[11/30/21]seed@20I-1955:~/.../20I-1955_IS\ Assignment\ 3 $
```

Copy the OpenSSL configuration file in the working directory "20I-1955_IS_Assignment_3"

Figure 8: Copy the OpenSSL configuration file in the current directory



```
File Edit View Search Tools Documents Help
Open Save
default_ca      = CA_default      # The default ca section
#####
[ 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 certificates with same subject.
new_certs_dir  = $dir/newcerts    # default place for new certs.

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

x509_extensions = usr_cert       # The extensions to add to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt       = ca_default       # Subject Name options
cert_opt        = ca_default       # Certificate field options

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

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crlnumber must also be commented out to leave a V1 CRL.
# crl_extensions = crl_ext
```

Directories to create specified in OpenSSL configuration file

Figure 9: Directories to create specified in OpenSSL config file

```
[11/30/21]seed@20I-1955:~/Desktop$ ls
20I-1955 IS Assignment 3 csp csp.zip Untitled Document 1.txt
[11/30/21]seed@20I-1955:~/Desktop$ cd 20I-1955_IS\ Assignment\ 3\ /
[11/30/21]seed@20I-1955:~.../20I-1955_IS Assignment 3 $ cp /usr/lib/ssl/openssl.cnf openssl.cnf
[11/30/21]seed@20I-1955:~.../20I-1955_IS Assignment 3 $ mkdir demoCA ← Location to store certificates etc...
[11/30/21]seed@20I-1955:~.../20I-1955_IS Assignment 3 $ cd demoCA/
[11/30/21]seed@20I-1955:~.../demoCA$ mkdir certs crt newcerts ← Directories to store issued certificates, crt and new certificates
[11/30/21]seed@20I-1955:~.../demoCA$ touch index.txt serial ← Directories to store issued certificates, crt and new certificates
[11/30/21]seed@20I-1955:~.../demoCA$ echo 1000 > serial
[11/30/21]seed@20I-1955:~.../demoCA$ ls
certs crt index.txt newcerts serial ← Index file for database and directory for current serial number
[11/30/21]seed@20I-1955:~.../demoCA$
```

Figure 10: Creating Directories specified in OpenSSL config file

Step-2 Certificate Authority (CA)

In this task I have to generate a self-signed certificate for our Certificate Authority and this certificate will serve as a root certificate. For the creation of the certificate, I used the following command “**openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf**”. In this command “ca.crt” contains the CA’s public key and “ca.key” contains CA’s private key. After running the command certificate for CA created successfully. In Figure 13 we can see the same information in subject and issuer, indicating the self-signed certificate. Also, we can see in Figure 14 under “Basic Constraints” Certificate Authority is marked “Yes”. This indicates that this certificate is for CA and can be used to sign other certificates.

```
[11/30/21]seed@20I-1955:~.../20I-1955_IS Assignment 3 $ openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf
Generating a 2048 bit RSA private key
.+++
writing new private key to 'ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PK
State or Province Name (full name) [Some-State]:Islamabad
Locality Name (eg, city) []:Islamabad
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FAST
Organizational Unit Name (eg, section) []:MS Computer Network & Security
Common Name (e.g. server FQDN or YOUR name) []:20I-1955_MuhammadOsamaKhalid
Email Address []:i201955@nu.edu.pk
[11/30/21]seed@20I-1955:~.../20I-1955_IS Assignment 3 $
```

Figure 11: Generating certificate for CA



Figure 12: Certificate generated successfully

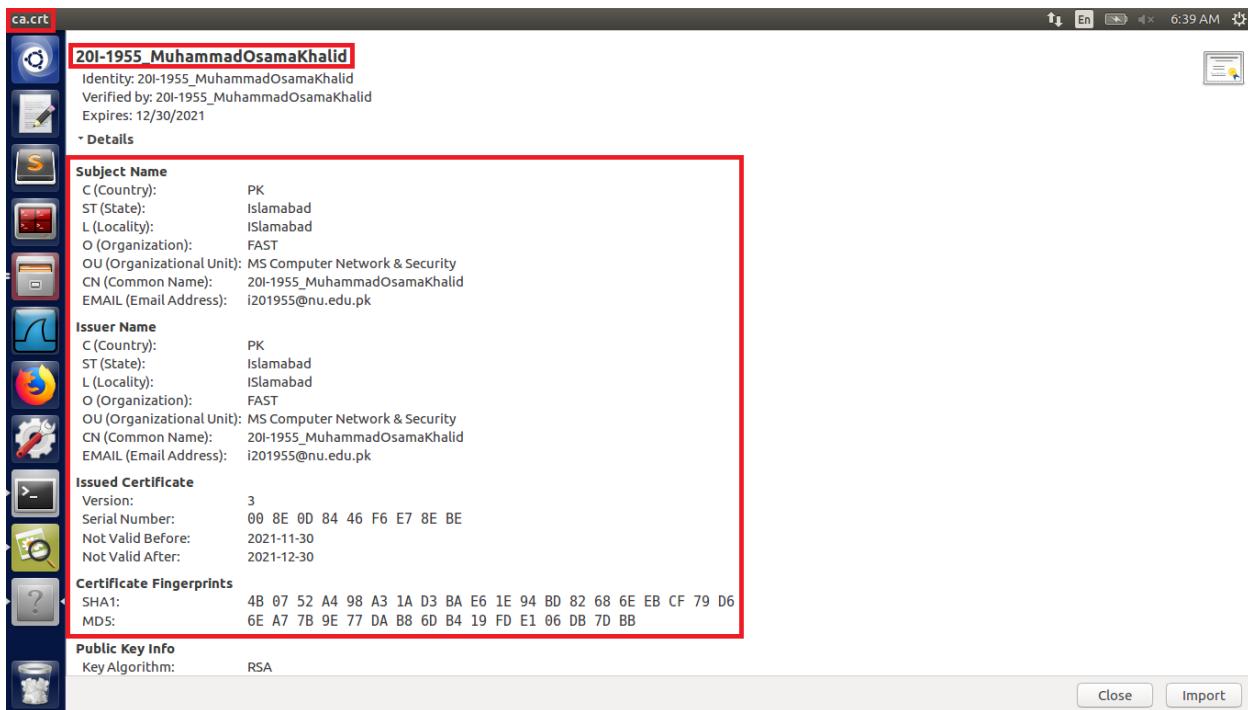


Figure 13: Contents of ca.crt

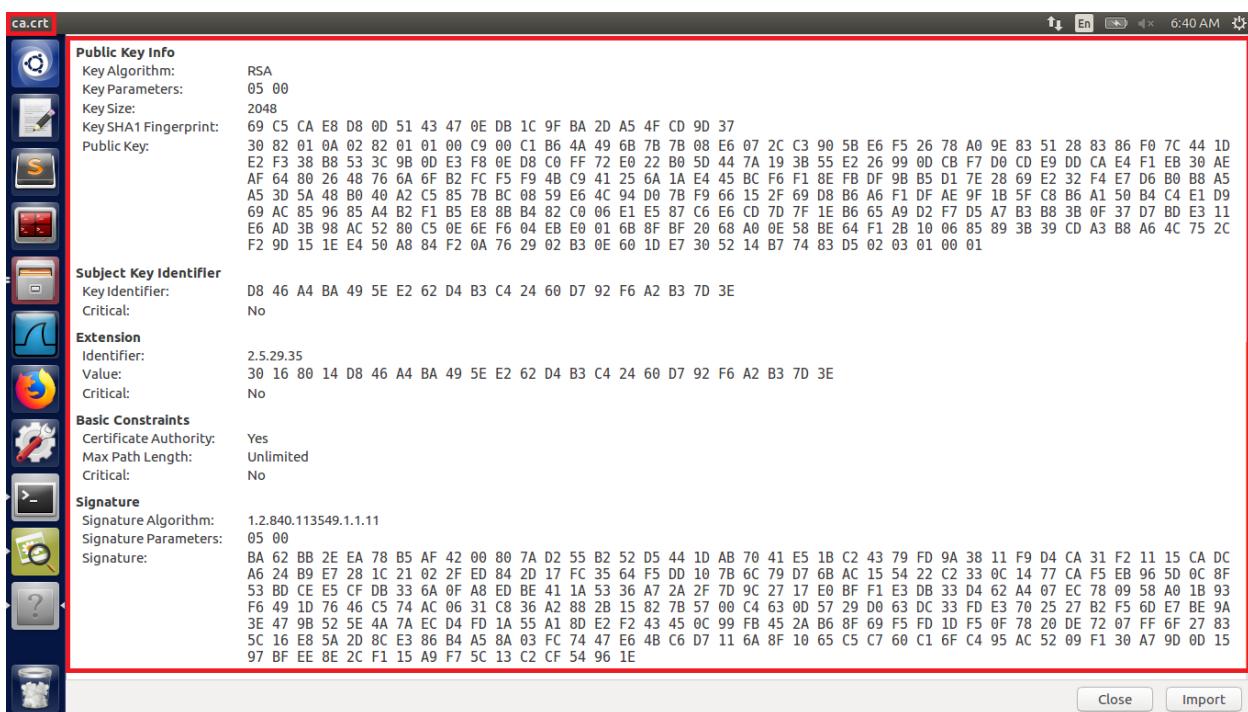


Figure 14: Contents of ca.crt

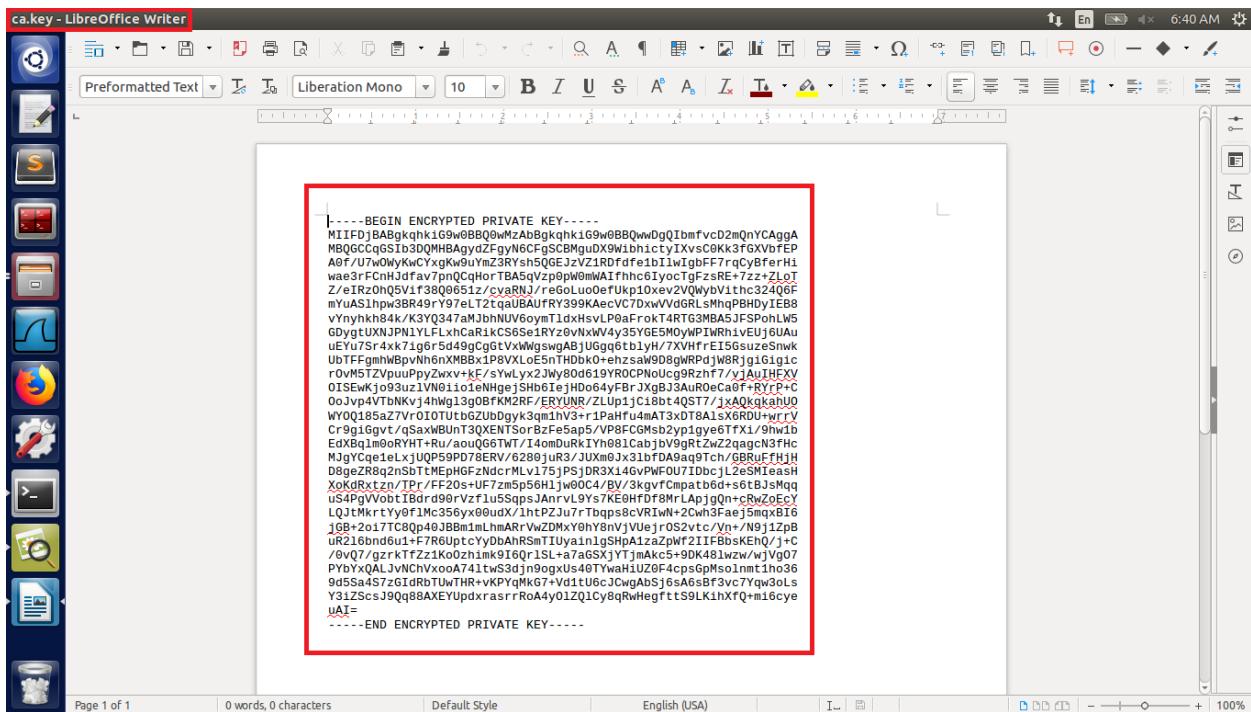


Figure 15: Contents of ca.key

Task-2 Creating a Certificate for osamakhalid.com

Now as we are CA, we can sign digital certificates for the companies' websites and the first website I am going to issue the certificate is "osamakhalid.com". For issuing the certificate to the website I have to go through the following steps.

Step-1 Generate public/private key pair

For getting the certificate from CA, the company first needs to create its public/private key pair. To create the RSA key pair I use the command "**openssl genrsa -aes128 -out server.key 1024**" and also provide the password to encrypt the private key. The private key was encrypted by an AES-128 encryption algorithm. In the command "**server.key**" store the keys. As the "**server.key**" is an encoded text file so to see the contents of the file I use the command "**openssl rsa -in server.key -text**".

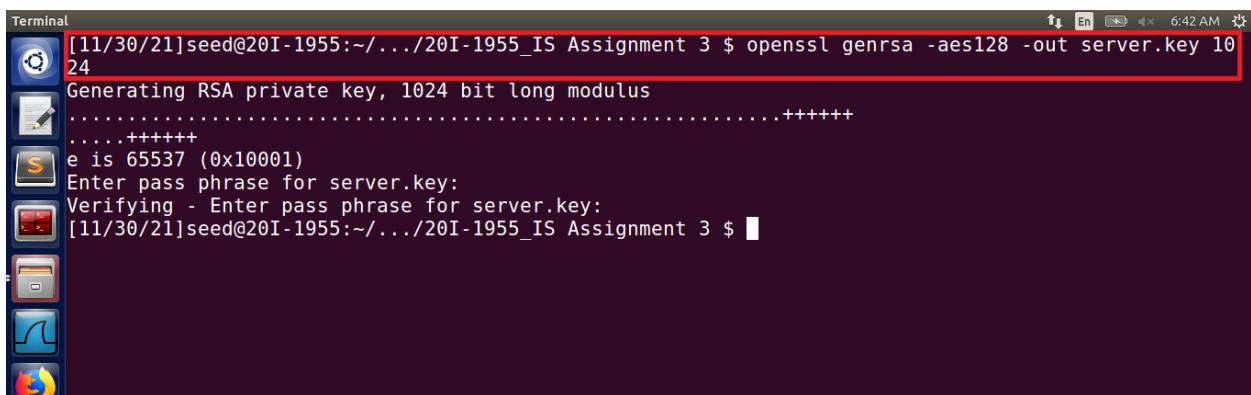


Figure 16: Generating the Public/Private key



Figure 17: Public/Private key generated and stored in the server.key

```
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ openssl rsa -in server.key -text
Enter pass phrase for server.key:
Private-Key: (1024 bit)
modulus:
    00:d1:37:41:d1:0a:96:03:4a:55:33:d5:ec:0c:21:
    37:a4:e7:03:44:07:45:fe:97:23:d6:b9:84:b0:ab:
    92:b5:c0:ce:45:1d:15:dc:11:b2:9b:89:56:03:cb:
    fd:58:4f:d9:47:8b:5b:9a:73:86:61:0e:37:e8:66:
    08:fc:a0:2c:19:7a:f9:90:a0:7d:25:0b:cf:fd:9b:
    3f:b3:ec:59:f8:12:bf:2d:26:e9:15:21:b8:97:88:
    51:8b:9a:f8:1c:c0:e5:84:84:f4:47:cd:78:09:f1:
    9a:4d:ef:8a:31:ef:db:cd:c8:93:bd:96:e6:6c:7f:
    76:e0:15:13:0f:76:f4:0c:99
publicExponent: 65537 (0x10001)
privateExponent:
    00:aa:bf:8d:50:bd:72:07:47:70:58:38:dc:48:5e:
    07:da:e6:cb:6c:86:ff:22:34:5a:04:92:2e:70:3d:
    e7:23:bd:32:db:87:4c:62:ea:72:b9:ac:11:6e:51:
    da:f7:6b:5b:aa:9f:5a:e8:a8:26:61:29:c8:89:3c:
    55:53:50:aa:0b:bb:52:5b:8c:c0:a3:9c:6c:80:84:
    e5:f9:63:c7:f2:d5:19:fb:e2:6e:ab:e9:83:fe:54:
    1c:10:23:e6:f5:7e:bb:5e:b6:8d:1c:6b:ea:30:51:
    58:53:98:37:2c:b8:b3:44:ac:f7:05:3a:d6:6a:69:
    5b:70:e4:73:0c:00:96:d1:8d
prime1:
    00:f4:97:a3:4b:5f:e0:95:f3:0d:a1:58:fa:6c:49:
```

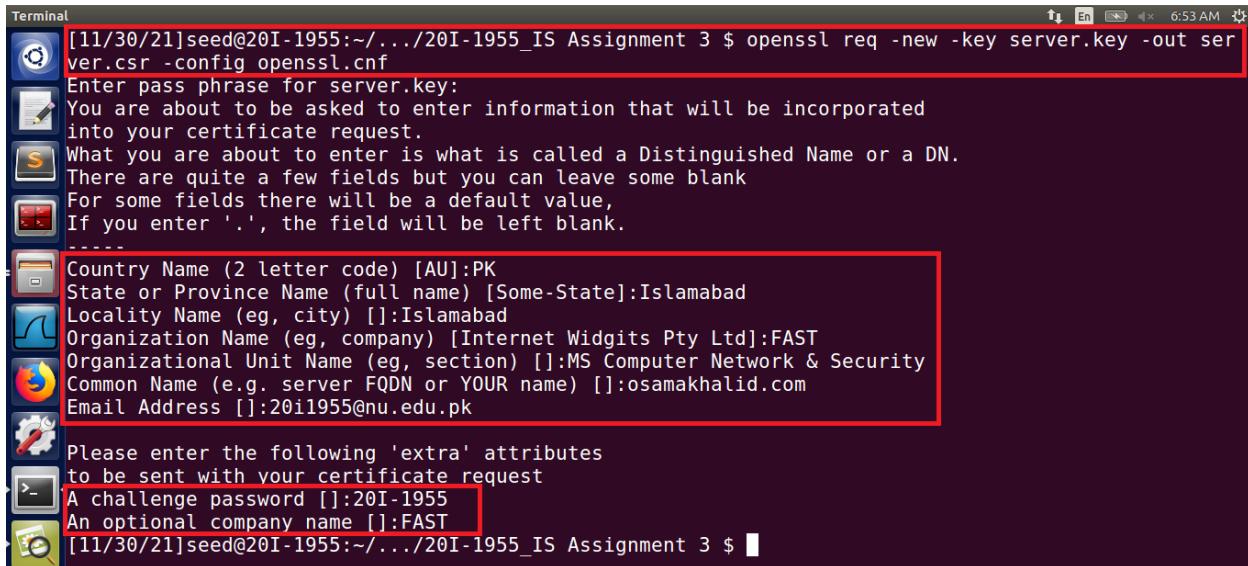
Figure 18: Contents of the server.key

```
exponent2:
    00:a4:27:77:61:fd:37:8c:43:5d:d2:fa:b1:3c:2e:
    ec:85:c7:b3:d2:b0:04:16:88:4b:46:18:9c:76:52:
    43:85:69:92:3a:c6:20:29:a3:19:c6:03:04:5e:56:
    38:2c:dd:6a:11:a6:17:10:c5:e4:89:9e:80:4f:e8:
    50:85:92:36:bd
coefficient:
    27:e6:2a:de:06:e4:d8:2d:bb:f7:45:f9:b0:5f:80:
    8a:86:5f:9f:e8:d1:90:dd:4d:e0:70:99:5d:0f:d5:
    42:33:05:b2:67:70:9c:25:89:42:e8:eb:e7:c4:67:
    2b:96:1a:2c:79:f0:0a:1f:0a:f4:4f:53:d3:87:49:
    3f:a2:27:1b
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDRN0HRCpYDSLuz1ewMITek5wNEB0X+lyPWuYSwq5K1wM5FHRXc
EbKbiVYDy/1YT9lHi1uac4ZhdfoZqj8oCwZevmQoH0lC8/9mz+z7Fn4Er8tJukV
IbiXfGLmvgcw0WEhPRHzXgJ8ZpN74ox79vNyJ09luZsf3bgFRMPdvQMmQIDAQAB
AoGBAKq/jVC9cgdHcFg43EheB9rmy2yG/yIWgSSLnA95y09MtuHTGLqrmsEW5R
2vdrlW6qfwuioJmEpyIk8VVNQggu7UluMwK0cbICE5fljx/LVGfvibqvpg/5UHBAj
5vV+u162jRxr6jBRWF0YNyy4s0Ss9wU61mpw3DkccwAltGNAkEA9JejS1/glfMN
oVj6bEnmvItlIjE1yluijqCHD91Uj7Cgy/a5unp3SCyBkLiNbEnSVgREEdFwmZ3
1bB2mz6UPWJBANr50/DPDxu0ZMlxwldvkGQmcTV96iDkRztZUE/HcU833ey0PuGO
Miyo88S8qPd5JHWYWmjX9SjGdp031vh6yScCQE4Cde4WgdBxvml4yhq0DaoKLbK
VMZ/hqjX/r8RMlIdxu0iKA14h95psG6Q+Qz+FU6uR+aCSqyabAuzwPyGRDsCQ0CK
J3dh/TemQ13S+rE8LuyFx7PSsAQWiEtGGJx2UK0FaZI6xiApoxnGAwReVjgs3WoR
phcQxeSJnoBP6FCFkj9AkAn5ireBuTYLbv3RfmwX4CKhl+f6NGQ3U3gcJldD9VC
MwWyZ3CcJYLC60vnxCgrlhosefAKHwr0T1PTh0k/oicb
-----END RSA PRIVATE KEY-----
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $
```

Figure 19: Contents of the server.key

Step-2 Generate a Certificate Signing Request (CSR)

Once the company generates its public and private key file their next task is to generate a Certificate Signing Request (CSR) that contains companies generated public key and then send this request to CA. On receiving the request then CA will generate the certificate for the key. Now to generate the CSR I use the command “`openssl req -new -key server.key -out server.csr -config openssl.cnf`”.



```
[11/30/21]seed@20I-1955:~/.20I-1955_IS Assignment 3 $ openssl req -new -key server.key -out server.csr -config openssl.cnf
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PK
State or Province Name (full name) [Some-State]:Islamabad
Locality Name (eg, city) []:Islamabad
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FAST
Organizational Unit Name (eg, section) []:MS Computer Network & Security
Common Name (e.g. server FQDN or YOUR name) []:osamakhalid.com
Email Address []:20i1955@nu.edu.pk
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:20I-1955
An optional company name []:FAST
[11/30/21]seed@20I-1955:~/.20I-1955_IS Assignment 3 $
```

Figure 20: Generating the CSR



Figure 21: CSR generated successfully

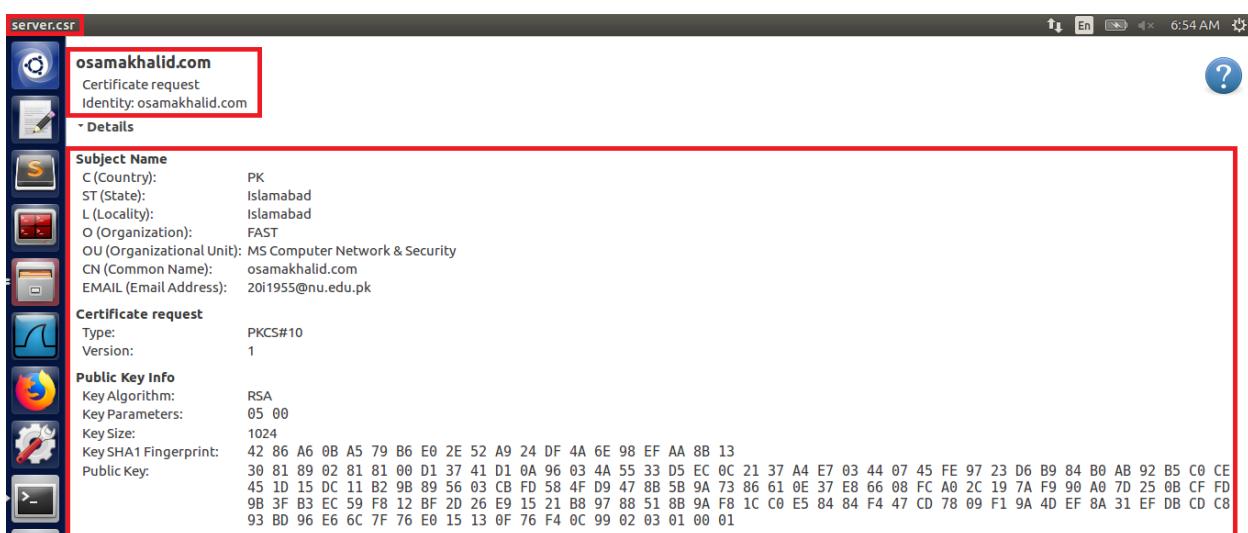
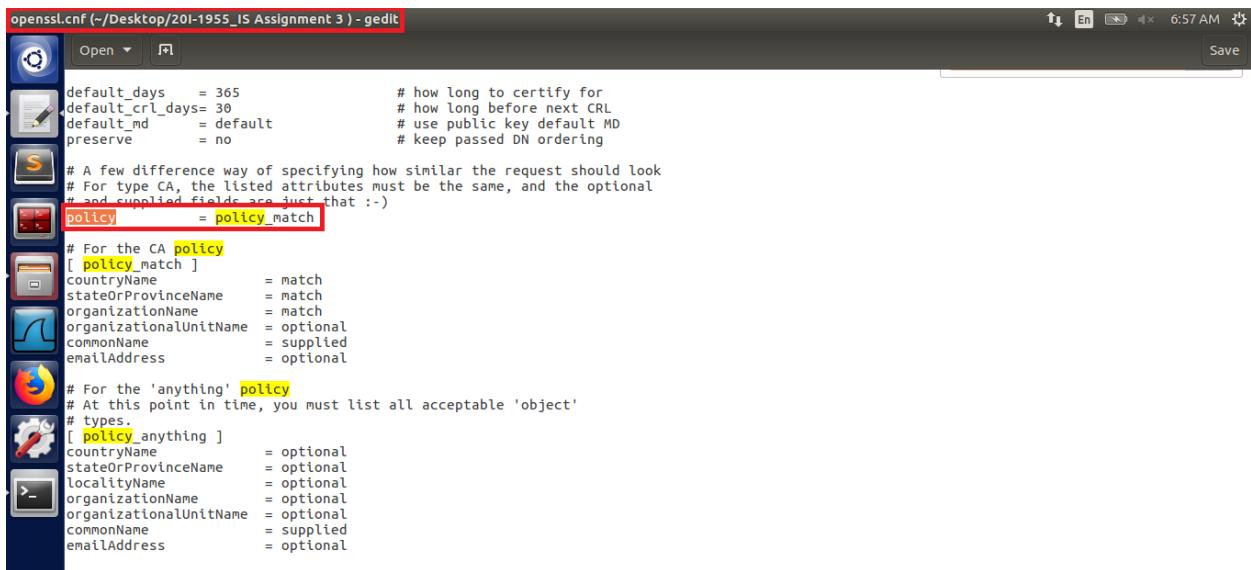


Figure 22: Contents of the server.csr

Step-3 Generating Certificates

On receiving the CSR request from the company, the next task is for the CA to generate the certificate for the company, and for that CSR file needs to have the CA's signature. Now to generate the certificate for the company I use the following command "**openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf**". This command will turn the CSR (server.csr) into an X509 certificate (server.crt) using the CA's "**ca.crt**" and "**ca. the key**". Now before running the command, I change the policy in the "**openssl.cnf**" file to **policyAnything** from **policyMatch** to avoid any errors.



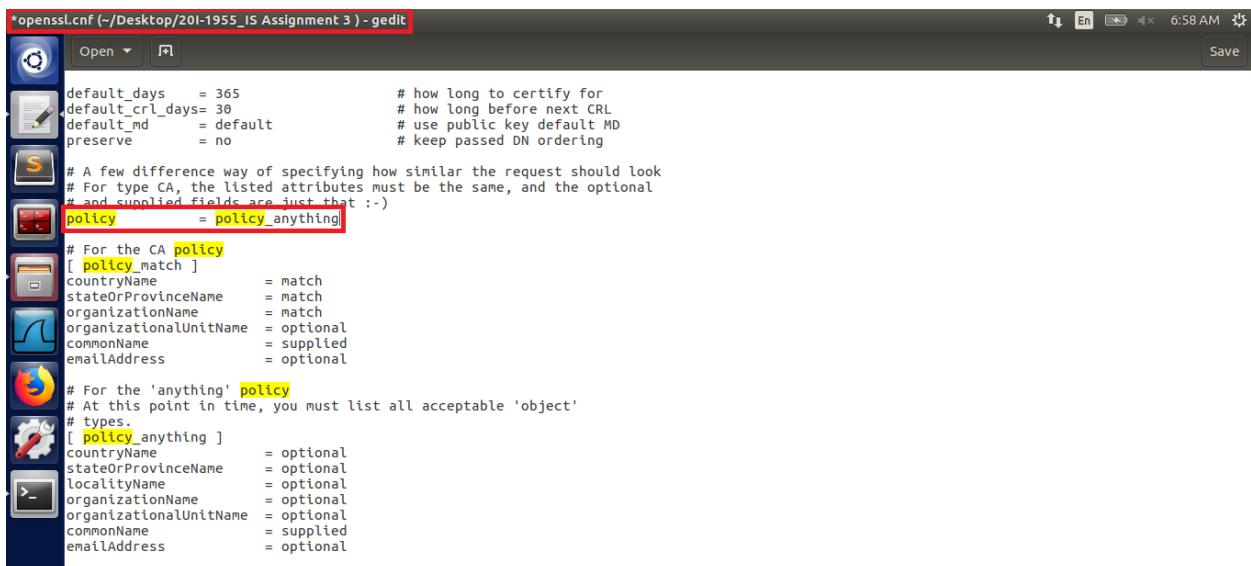
```
openssl.cnf (~/Desktop/20I-1955_IS Assignment 3) - gedit
Save
default_days      = 365          # how long to certify for
default_crl_days= 30          # how long before next CRL
default_md        = default     # use public key default MD
preserve         = no           # keep passed DN ordering

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

# For the CA policy
[ policy_match ]
countryName      = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName       = supplied
emailAddress     = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName      = optional
stateOrProvinceName = optional
localityName     = optional
organizationName = optional
organizationalUnitName = optional
commonName       = supplied
emailAddress     = optional
```

Figure 23: Changing the policy in openssl.cnf



```
openssl.cnf (~/Desktop/20I-1955_IS Assignment 3) - gedit
Save
default_days      = 365          # how long to certify for
default_crl_days= 30          # how long before next CRL
default_md        = default     # use public key default MD
preserve         = no           # keep passed DN ordering

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

# For the CA policy
[ policy_match ]
countryName      = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName       = supplied
emailAddress     = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName      = optional
stateOrProvinceName = optional
localityName     = optional
organizationName = optional
organizationalUnitName = optional
commonName       = supplied
emailAddress     = optional
```

Figure 24: Changing the policy in openssl.cnf

The terminal window shows the command `openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf` being run. The output displays the certificate details, including the subject information (countryName=PK, stateOrProvinceName=Islamabad, localityName=Islamabad, organizationName=FAST, organizationalUnitName=MS Computer Network & Security, commonName=osamakhalid.com, emailAddress=2011955@nu.edu.pk) and the X509v3 extensions.

```
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 4096 (0x1000)
    Validity
        Not Before: Nov 30 12:00:10 2021 GMT
        Not After : Nov 30 12:00:10 2022 GMT
    Subject:
        countryName          = PK
        stateOrProvinceName = Islamabad
        localityName        = Islamabad
        organizationName    = FAST
        organizationalUnitName = MS Computer Network & Security
        commonName           = osamakhalid.com
        emailAddress         = 2011955@nu.edu.pk
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        Netscape Comment:
            OpenSSL Generated Certificate
        X509v3 Subject Key Identifier:
            12:57:8E:D5:23:4F:BB:E5:5D:A2:7E:71:67:86:33:24:FE:69:10:15
        X509v3 Authority Key Identifier:
            keyid:D8:46:A4:BA:49:5E:E2:62:D4:B3:C4:24:60:D7:92:F6:A2:B3:7D:3E
```

Figure 25: Generating the certificate for the company

The terminal window shows the final steps of generating the certificate, including the confirmation of the certificate's validity and the option to sign it. It also shows the database update message.

```
Validity
    Not Before: Nov 30 12:00:10 2021 GMT
    Not After : Nov 30 12:00:10 2022 GMT
Subject:
    countryName          = PK
    stateOrProvinceName = Islamabad
    localityName        = Islamabad
    organizationName    = FAST
    organizationalUnitName = MS Computer Network & Security
    commonName           = osamakhalid.com
    emailAddress         = 2011955@nu.edu.pk
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
        12:57:8E:D5:23:4F:BB:E5:5D:A2:7E:71:67:86:33:24:FE:69:10:15
    X509v3 Authority Key Identifier:
        keyid:D8:46:A4:BA:49:5E:E2:62:D4:B3:C4:24:60:D7:92:F6:A2:B3:7D:3E
Certificate is to be certified until Nov 30 12:00:10 2022 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
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $
```

Figure 26: Generating the certificate for the company

Task-3 Deploying Certificate in an HTTPS Web Server

On receiving the certificate from the CA, the company's next task is to use this certificate with the website to secure web browsing. As I am using the lab setup for this task, I first have to set up the HTTPS website using OpenSSL's built-in web server.

Step-1 Configuring DNS

As the website, I am using is "osamakhalid.com", in this step I need to add this into the "/etc/hosts" file for the machines to recognize the website on the web browser. To add the website in "/etc/hosts" I use the command "**sudo gedit /etc/hosts**" and then to confirm the changes I use the command "**cat /etc/hosts**".



Figure 27: Adding the website details in the /etc/hosts file

The screenshot shows a terminal window with the title 'Terminal'. The status bar at the top right shows '2:53 PM'. The main area contains the contents of the /etc/hosts file. A red box highlights the line '127.0.0.1 osamakhalid.com'.

```
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      20I-1955
# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1      User
127.0.0.1      Attacker
127.0.0.1      Server
127.0.0.1      www.SeedLabSQLInjection.com
127.0.0.1      www.xsslabelgg.com
127.0.0.1      www.csrflabelgg.com
127.0.0.1      www.csrflabattacker.com
127.0.0.1      www.repackagingattacklab.com
127.0.0.1      www.seedlabclickjacking.com
127.0.0.1      osamakhalid.com
127.0.0.1      www.example32.com
127.0.0.1      www.example68.com
127.0.0.1      www.example79.com
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $
```

Figure 28: Confirming the changes

Step-2 Configuring the webserver

In this step, I need to start the simple webserver using the `s_server` command but first I need to combine the key (`server.key`) and certificate (`server.crt`) into one file that is generated in the previous task. To combine the key and certificate into one file I use the following commands “`cp server.key server.pem`” and “`cat server.crt >> server.pem`”. Then to launch the webserver using “`server.pem`”, I use the command “`openssl s_server -cert server.pem -www`”.

```
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ cp server.key server.pem  
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ cat server.c  
server.crt server.csr  
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ cat server.crt >> server.pem  
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ openssl s_server -cert server.pem -www  
Enter pass phrase for server.pem:  
Using default temp DH parameters  
ACCEPT
```

Figure 29: Starting the webserver

By default, the server listens on port 4433 and I can access the server using the URL <https://osamakhalid.com:4433>. On visiting the URL, I see the web browser is displaying the error that the connection is not secure. On clicking on the advance button, I see the error that the issuer certificate is unknown. This is because, even I create the certificate for CA, but this certificate is not known to the browser and due to this web browser cannot validate the website certificate.

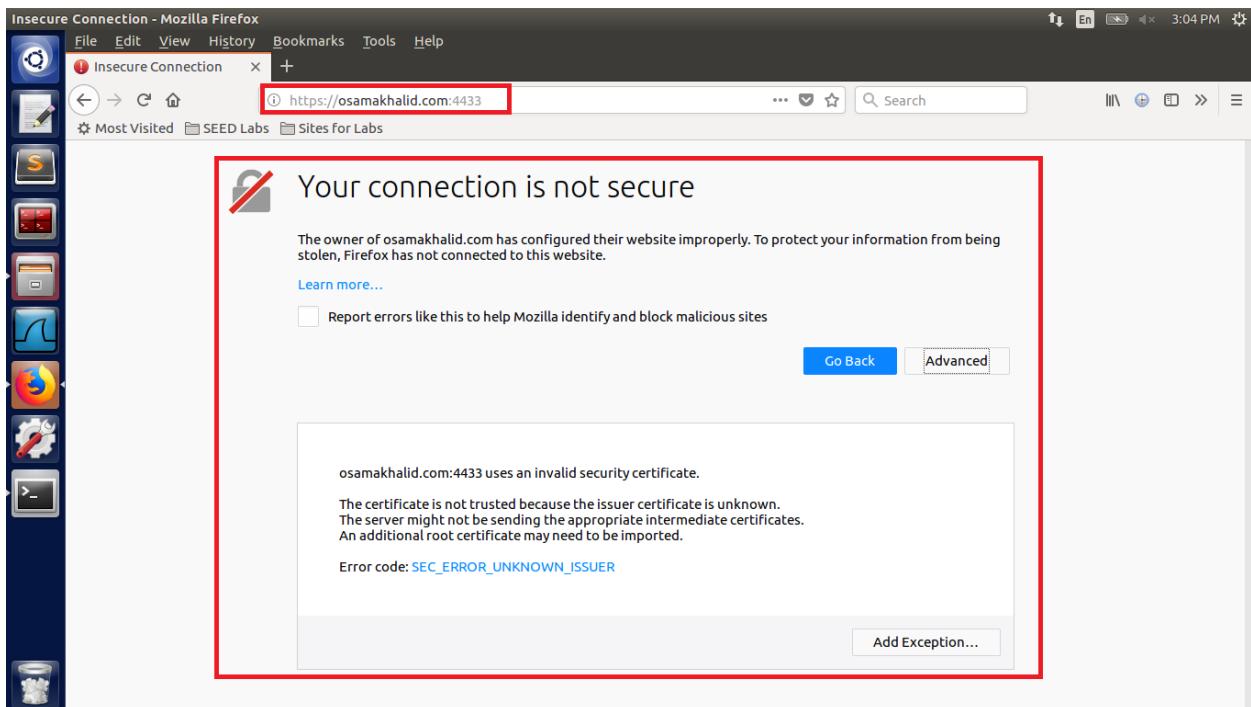


Figure 30: Visiting the website

Step-3 Getting the browser to accept our CA certificate

As we see in the previous step the browser didn't know about our CA certificate so, in this step, I have to manually add the CA's certificate in the browser. To add the CA's certificate in the browser that in this case is Firefox, I go to Menu > preferences > Privacy & Security > Certificates > View Certificates. After that from the popup, I click on "Authorities" and then click on the Import button to import the CA's certificate to the browser. Then I select the certificate and click open. After that in the "Downloading Certificate" popup, I mark the checkboxes and click ok. After that, we can see our CA's certificate in the Certificate Manager, Authorities section.

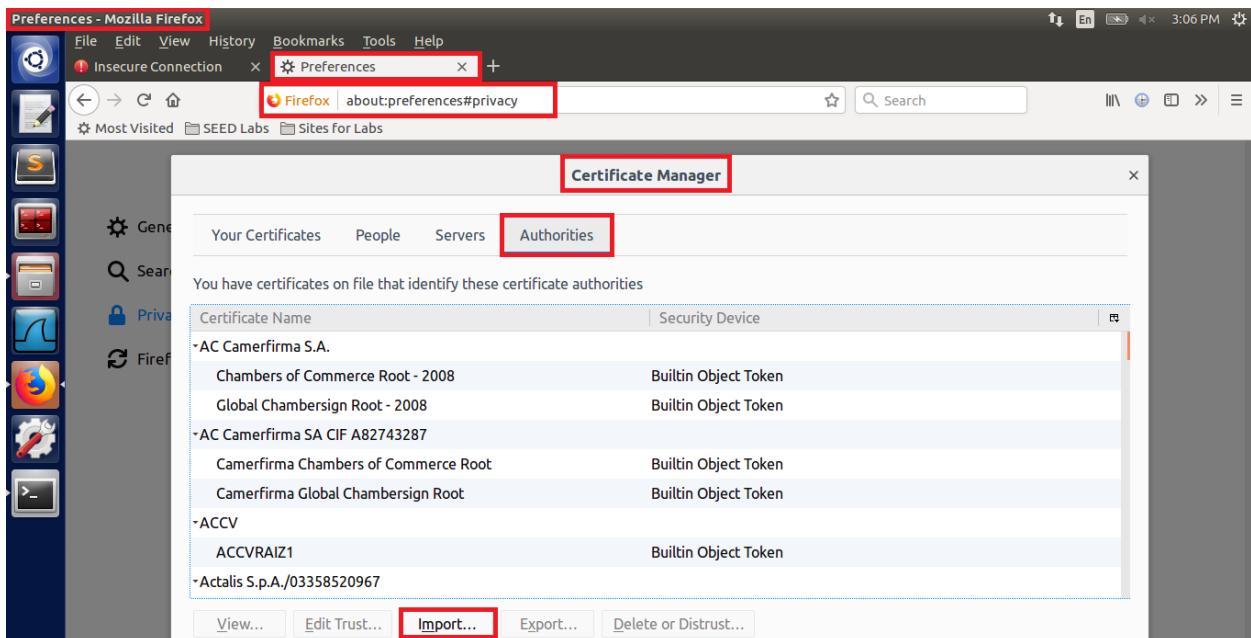


Figure 31: Adding CA's certificate into the browser

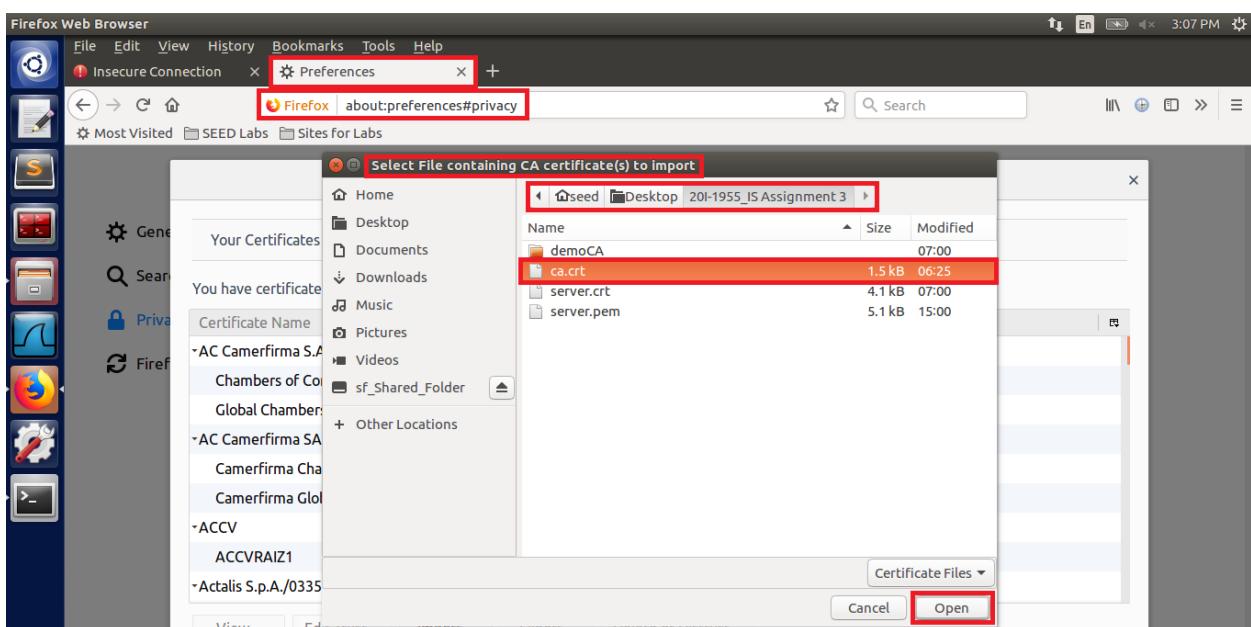


Figure 32: Adding CA's certificate into the browser

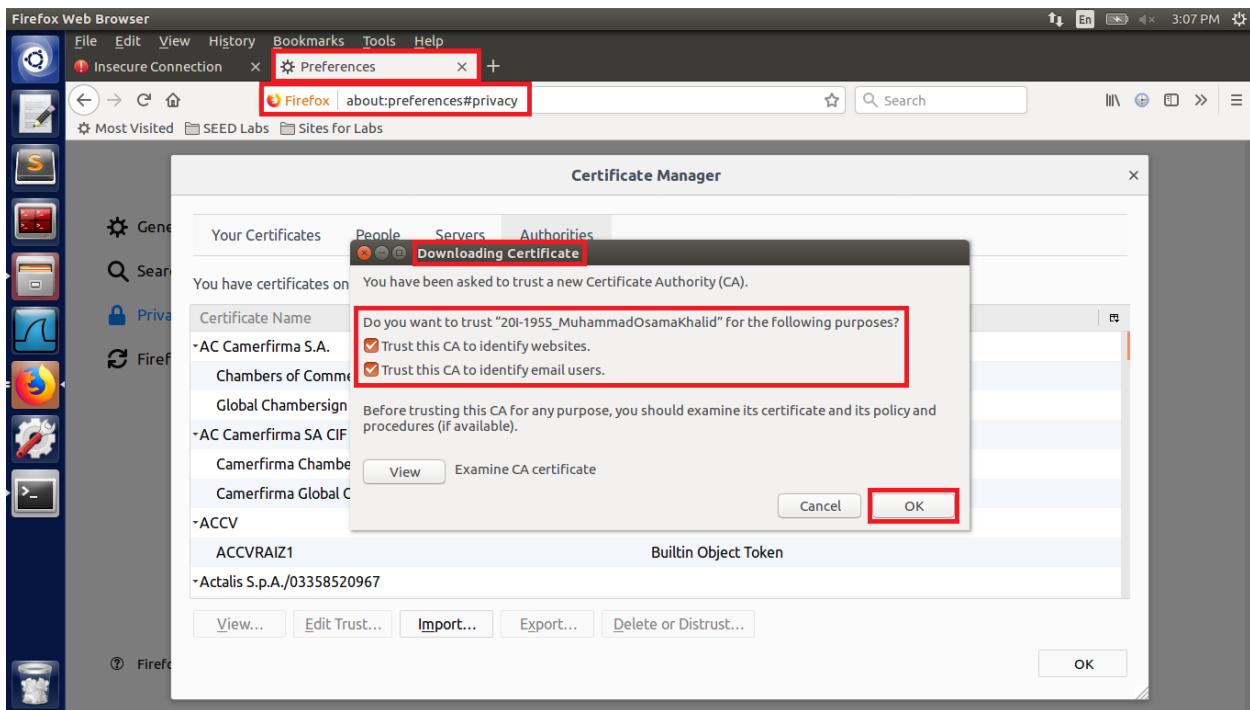


Figure 33: Adding CA's certificate into the browser

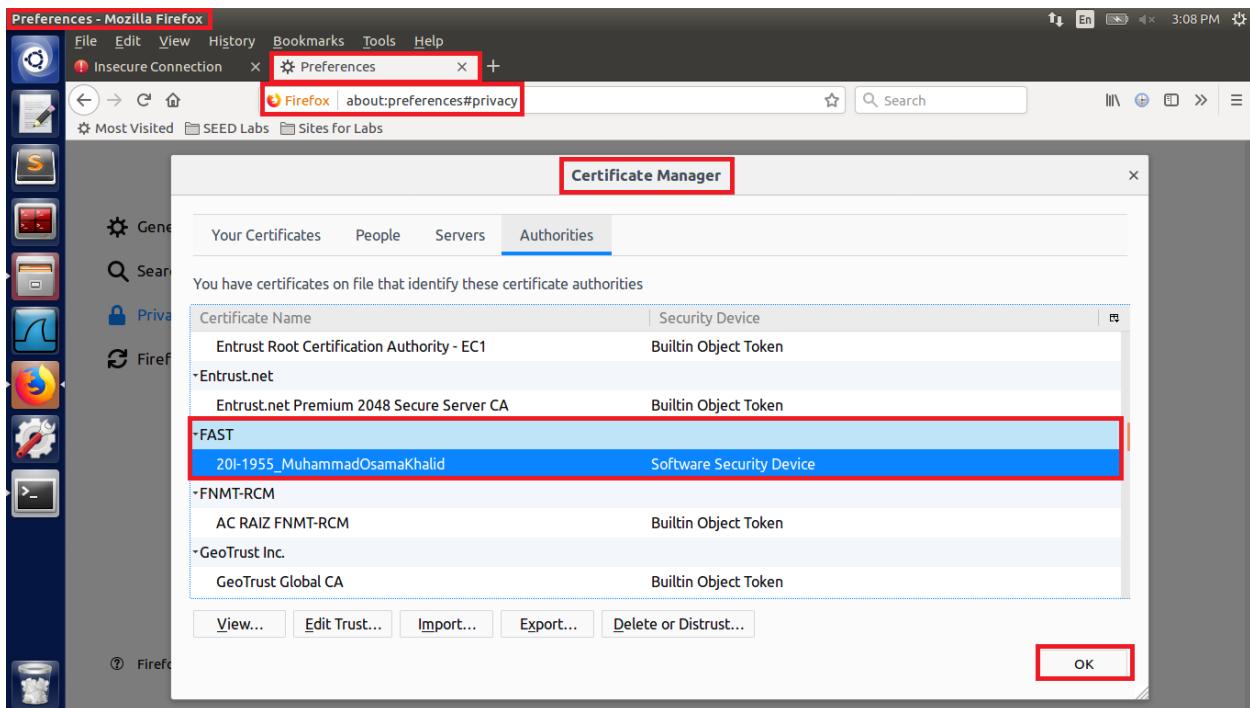


Figure 34: CA's certificate added to the browser successfully

Step-4 Testing our HTTPS website

Now as I add our CA's certificate into the browser now, I have to again visit the URL <https://osamakhalid.com:4433>. On visiting the URL, I see that the webpage loads successfully and we can see the reply from the server. Moreover, we can see that the webpage is also secure.

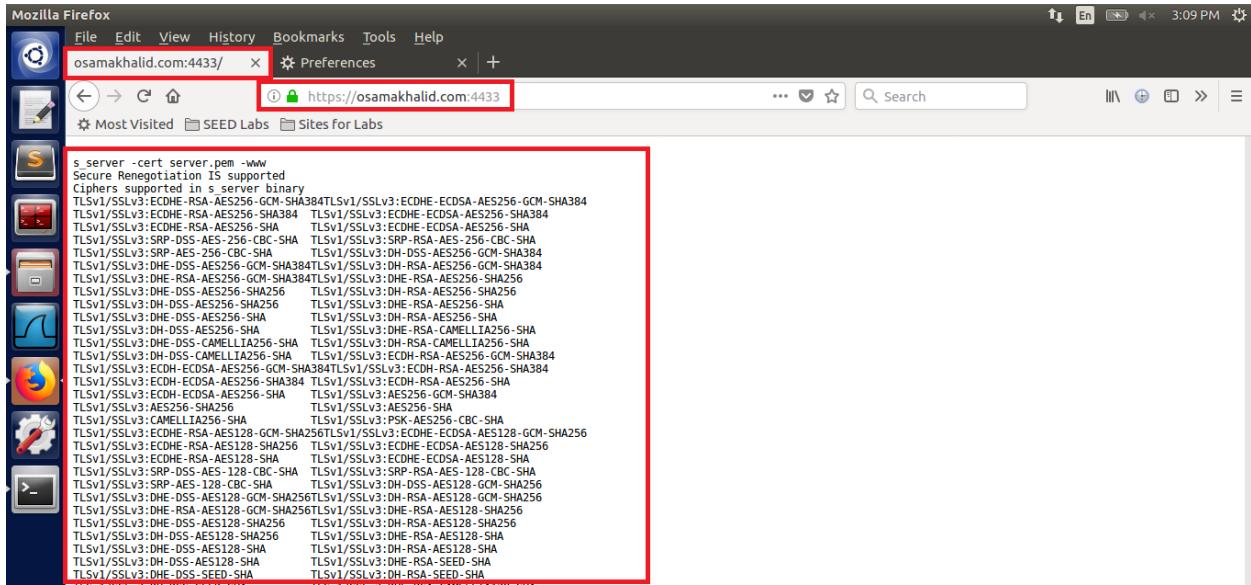


Figure 35: Webpage loads successfully

Q#1 Modify a single byte of the server.pem, and restart the server, and reload the URL. What do you observe? Make sure you restore the original server.pem afterward.

In this question, I have been asked to modify a single byte in the “server.pem” file and then again visit the website. To change modify the “server.pem” file I open the file in the “**Bless Hex Editor**” and then change the single byte of 7A to 7B and then save the file. After that I again run the `s_server` using the command **“openssl s_server -cert server.pem -www”** and then again visit the URL <https://osamakhalid.com:4433>. On visiting the URL, I see the “Secure Connection Failed” error which moreover indicates that the certificate has an invalid signature. This indicates that a single change in the certificate's signature will make it invalid and causes the browser to throw a “Secure Connection Failed” error.

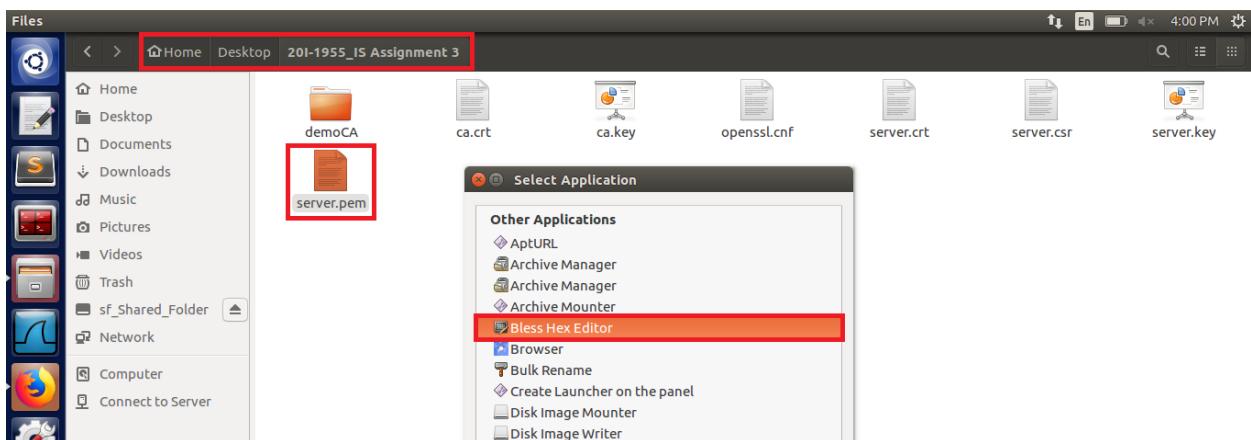


Figure 36: Opening the server.pem file

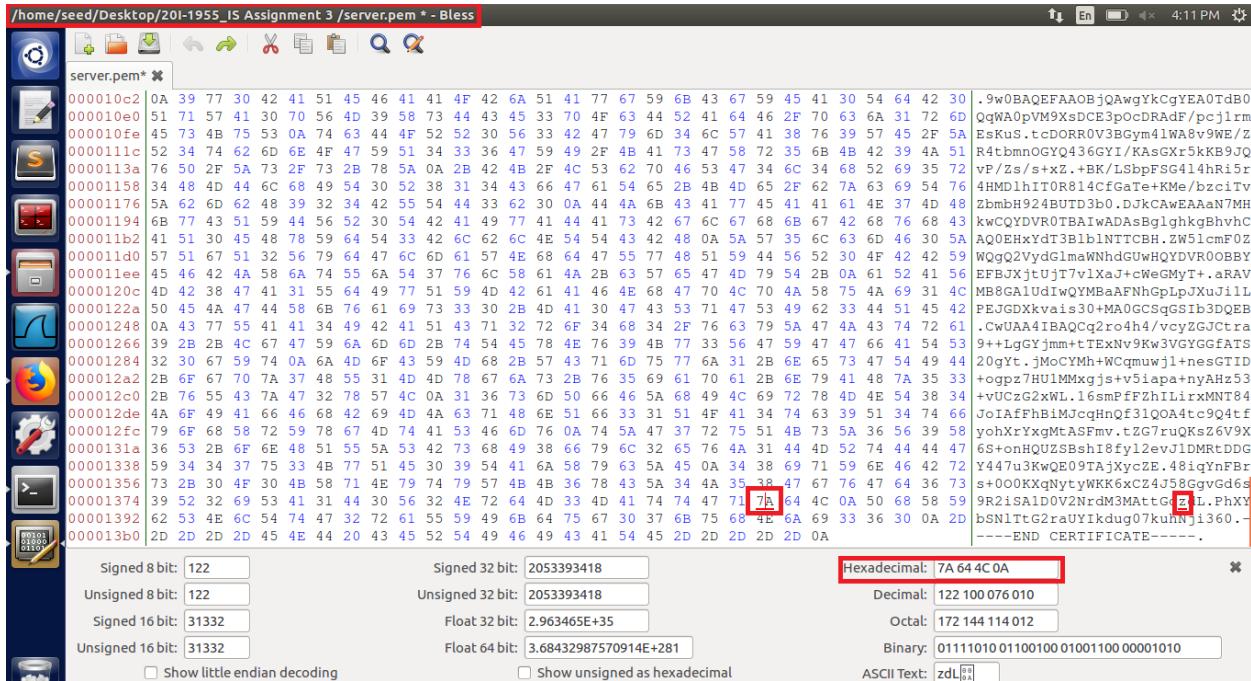


Figure 37: Changing a single bit from 7A to 7B

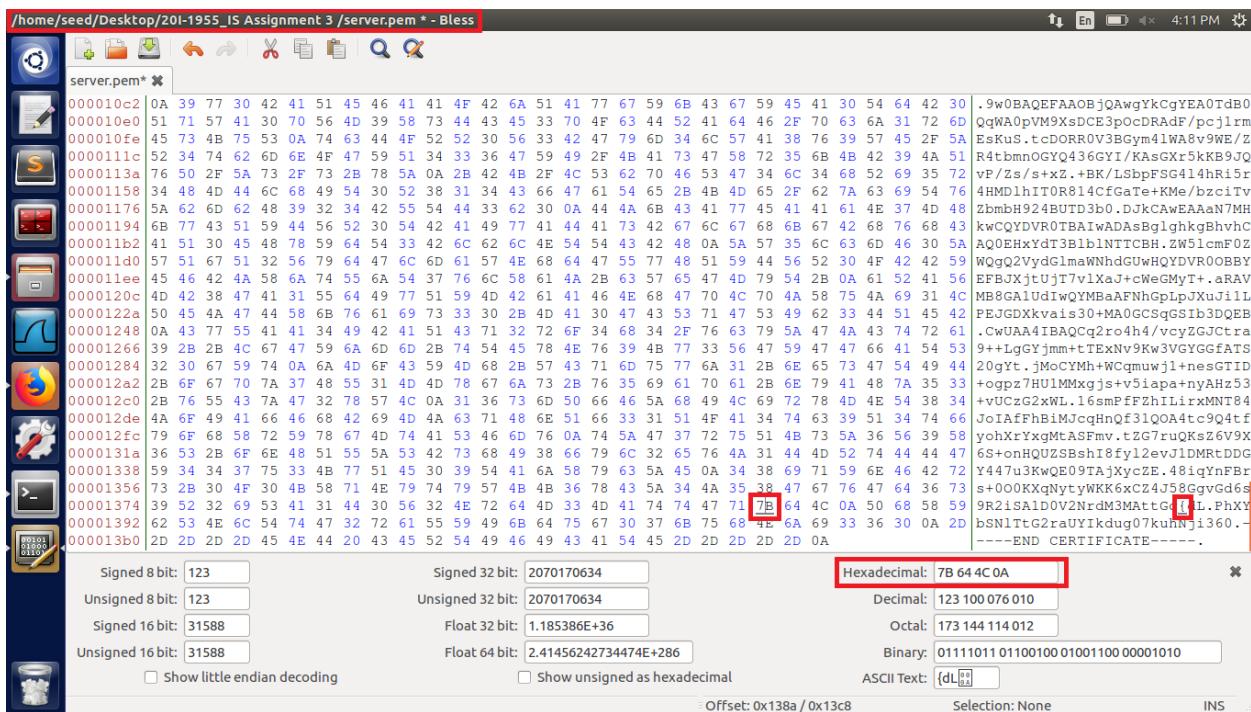


Figure 38: Changing a single bit from 7A to 7B

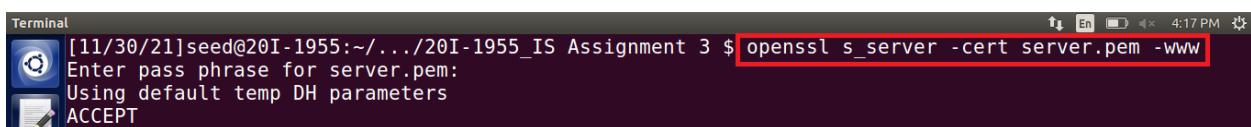


Figure 39: Run the server again

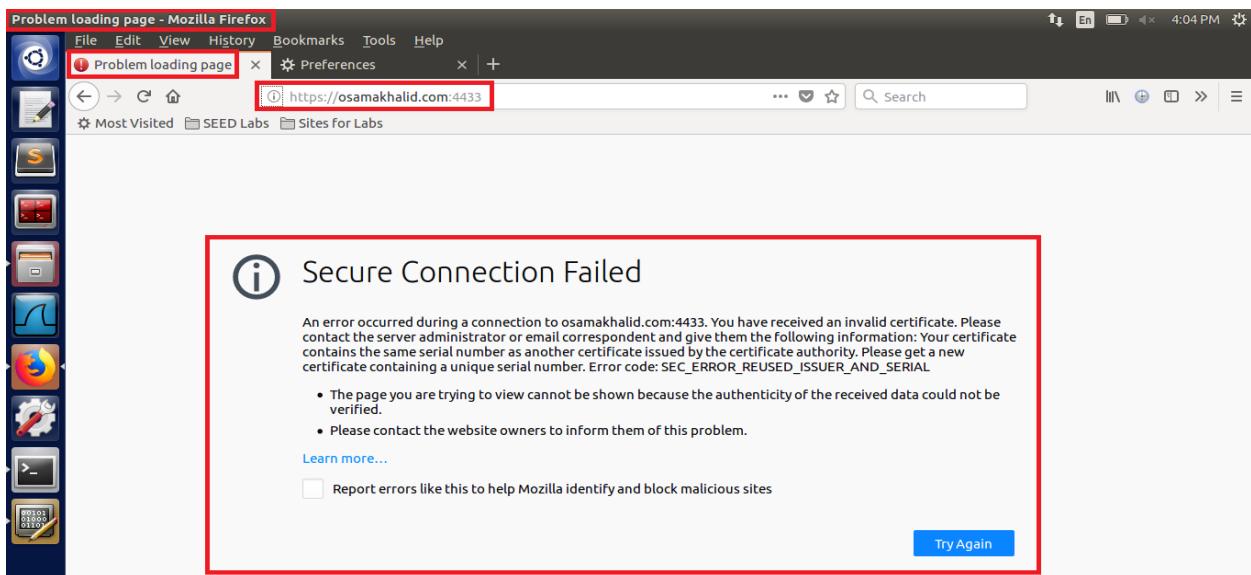


Figure 40: Webpage shows error

Q#2 Since osamakhalid.com points to the localhost, if we use https://localhost:4433 instead, we will be connecting to the same web server. Please do so, describe and explain your observations.

In this question I have been asked to visit localhost instead of osamakhalid.com, for that, I open the browser and visit the URL <https://localhost:4433>. On visiting the URL, I found that the browser is throwing the error “Your Connection is Not Secure”. Moreover, on clicking on the advance button I found the error that indicates that localhost is using the invalid certificate and the certificate used is not valid for localhost. After that, on clicking on “Add Exception” and then clicking on the view button I found that the certificate used is only issued to “osamakhalid.com”. This proves that the browser also checks the correctness of the URL requested and the certificate.

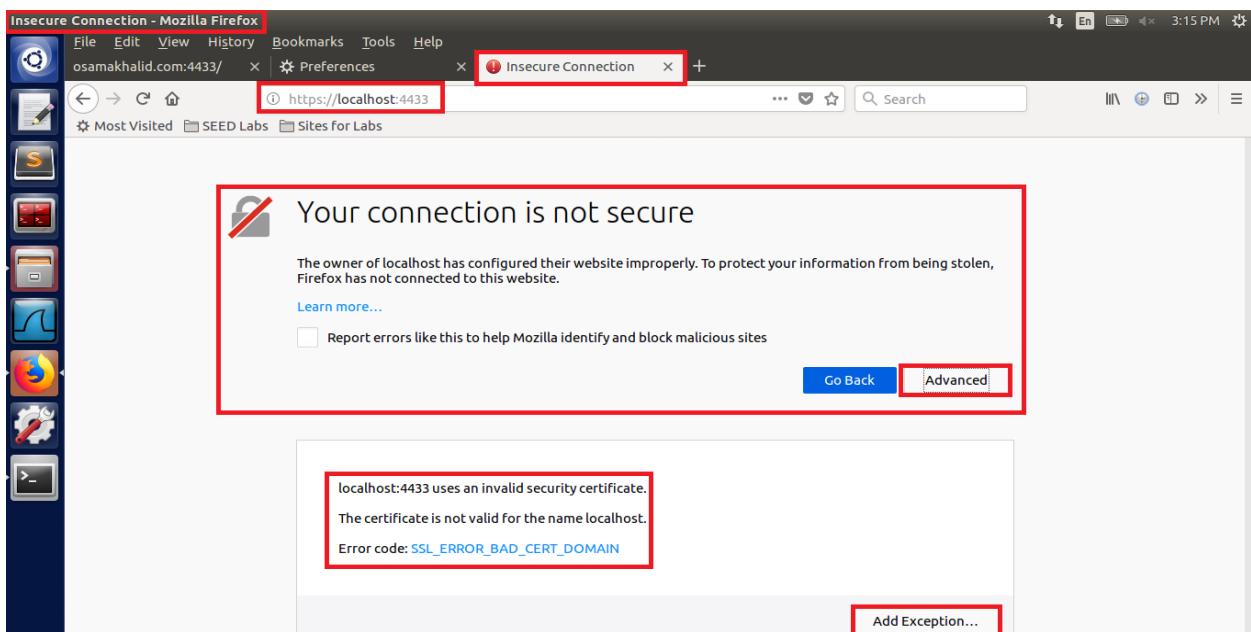


Figure 41: Browser throwing the "Connection not secure" error

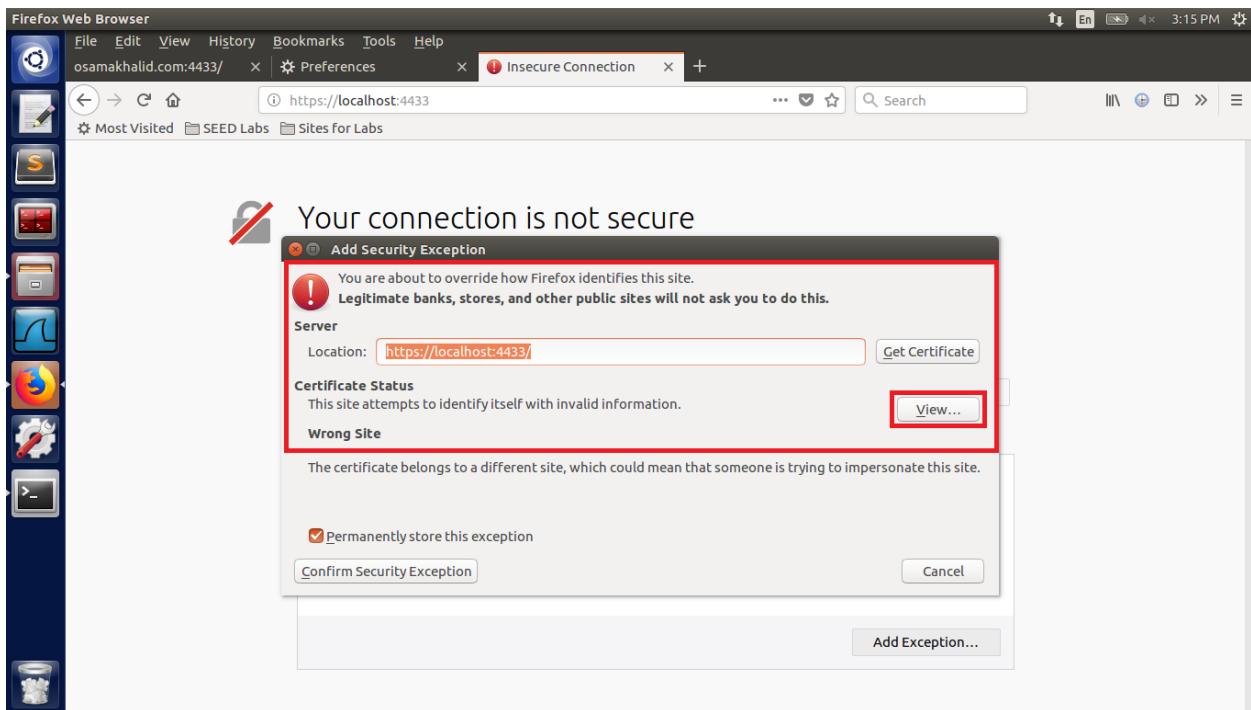


Figure 42: Viewing the Details of the certificate used

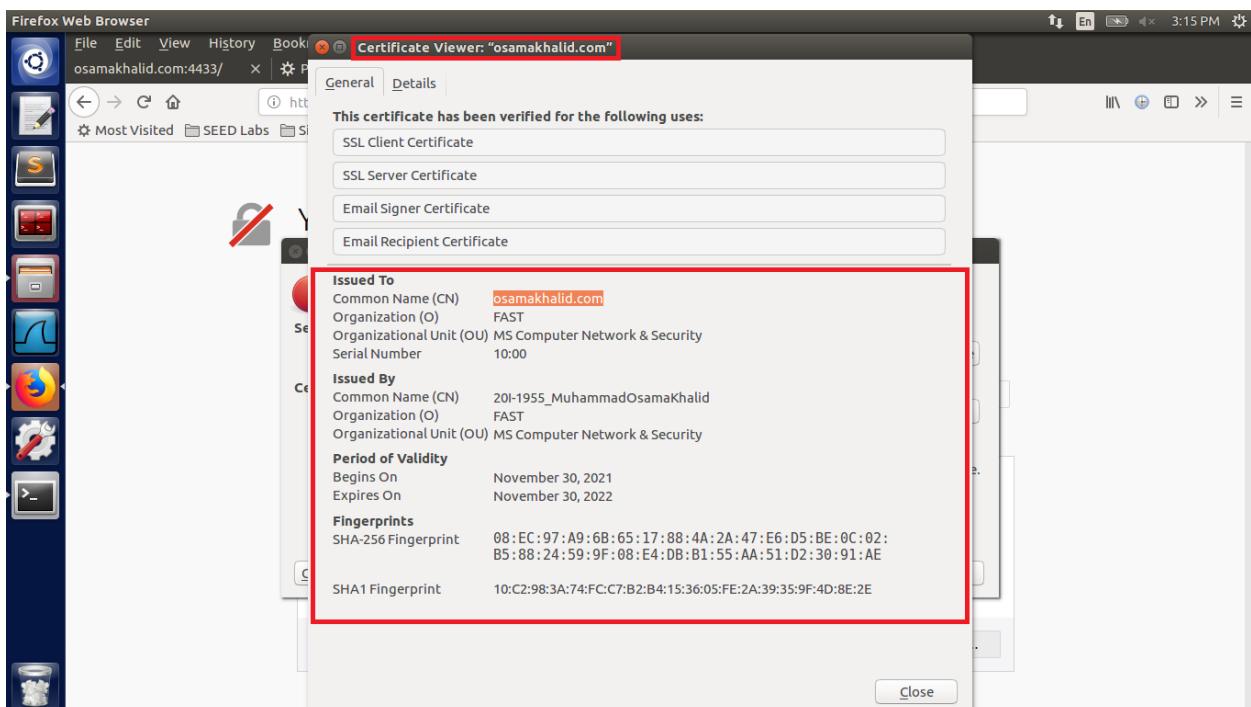
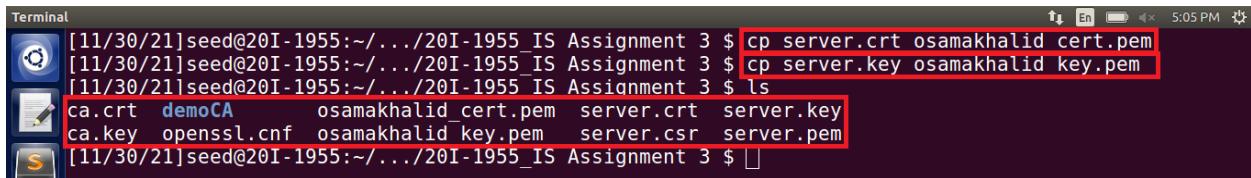


Figure 43: Certificate issued to osamakhalid.com, not localhost

Task-4 Deploying Certificate in an Apache-based HTTPS Website

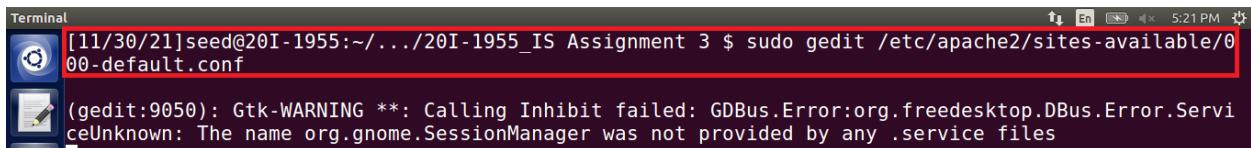
As in the previous task, we are using OpenSSL's s_server, now in this task, I have to set up the real HTTPS web server based on Apache. First, I need to combine the key (server.key) and certificate (server.crt) into one file and then use this file in the website configuration. To combine the key and certificate into one file I use the following commands “`cp server.crt osamakhalid_cert.pem`” and “`cp server.key osamakhalid_key.pem`”.



```
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ cp server.crt osamakhalid cert.pem
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ cp server.key osamakhalid key.pem
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ ls
ca.crt demoCA osamakhalid_cert.pem server.crt server.key
ca.key openssl.cnf osamakhalid_key.pem server.csr server.pem
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $
```

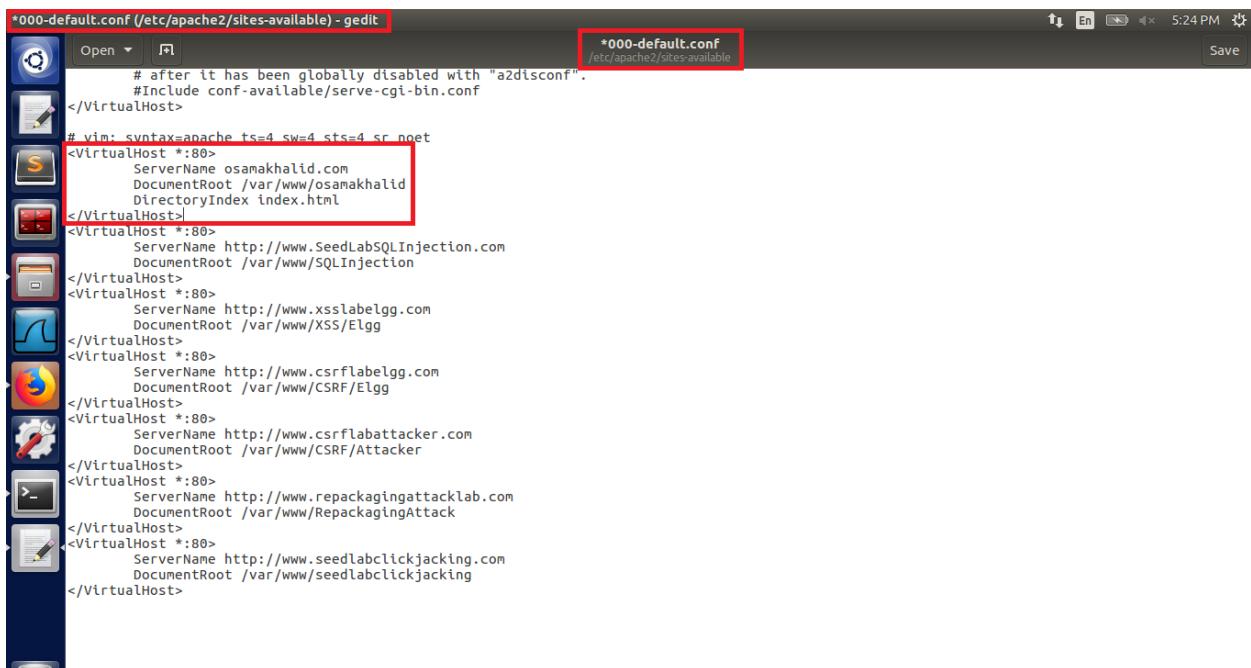
Figure 44: Combine the certificate and key into one file

As Apache server can simultaneously host multiple websites therefore it needs to know that where the website files are being stored and this is done by the VirtualHost file stored in the “/etc/apache2/sites-available” directory. To add the website, I need to add the VirtualHost entry to the two files “000-default.conf” and “default-ssl.conf”. The entry “`ServerName`” specifies the name of the website, “`DocumentRoot`” specifies where the website files are stored, and “`SSLCertificateFile`”, “`SSLCertificateKeyFile`” specifies where the certificate and private key are stored.



```
[11/30/21]seed@20I-1955:~/.../20I-1955_IS Assignment 3 $ sudo gedit /etc/apache2/sites-available/000-default.conf
(gedit:9050): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files
```

Figure 45:Opening the 000-default.conf file



```
*000-default.conf (/etc/apache2/sites-available) - gedit
Open F
*000-default.conf
/etc/apache2/sites-available
Save
# 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 *:80>
    ServerName osamakhalid.com
    DocumentRoot /var/www/osamakhalid
    DirectoryIndex index.html
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.SeedLabsSQLInjection.com
    DocumentRoot /var/www/SQLInjection
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.xsslabelgg.com
    DocumentRoot /var/www/XSS/Elgg
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.csrflabelgg.com
    DocumentRoot /var/www/CSRF/Elgg
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.csrfflabattacker.com
    DocumentRoot /var/www/CSRF/Attacker
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.repackagingattacklab.com
    DocumentRoot /var/www/RepackagingAttack
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.seedlabclickjacking.com
    DocumentRoot /var/www/seedlabclickjacking
</VirtualHost>
```

Figure 46: Adding the entry in 000-default.conf file

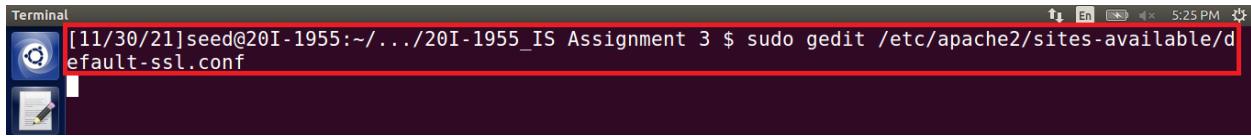


Figure 47: Opening the default-ssl.conf file

The screenshot shows the gedit editor with the file `*default-ssl.conf (/etc/apache2/sites-available) - gedit`. The configuration includes a virtual host section for port 443:

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

    SSLEngine On
    SSLCertificateFile /home/seed/Desktop/20I-1955_IS_Assignment_3/osamakhalid_cert.pem
    SSLCertificateKeyFile /home/seed/Desktop/20I-1955_IS_Assignment_3/osamakhalid_key.pem
</VirtualHost>
```

Figure 48: Adding the entry in default-ssl.conf file

After that, I go to the “`/var/www/`” directory using the command “`cd /var/www`”. Then I create the directory in which I will store the website file and then create the simple webpage “`index.html`” into the website directory “`osamakhalid`”.

The screenshot shows a terminal window with the following commands:

```
[11/30/21]seed@20I-1955:~$ cd /var/www/
[11/30/21]seed@20I-1955:~/www$ ls
CSRF html RepackagingAttack SQLInjection XSS
[11/30/21]seed@20I-1955:~/www$ sudo mkdir osamakhalid
[11/30/21]seed@20I-1955:~/www$ ls
CSRF html osamakhalid RepackagingAttack SQLInjection XSS
[11/30/21]seed@20I-1955:~/www$ cd osamakhalid/
[11/30/21]seed@20I-1955:~/osamakhalid$ ls
index.html
[11/30/21]seed@20I-1955:~/osamakhalid$ cat index.html
<!DOCTYPE html>
<html>
    <head>
        <title>20I-1955_Muhammad Osama Khalid</title>
    </head>
    <body>
        <p>This is the test page by 20I-1955 Muhammad Osama Khalid :)</p>
    </body>
</html>
[11/30/21]seed@20I-1955:~/osamakhalid$
```

Figure 49: Creating the website files

After that, I run several commands to enable the SSL and then start the Apache server. For that, I first check the Apache configuration file for errors using the command “**sudo apachectl configtest**”. Then to enable the SSL module I run the command “**sudo a2enmod ssl**” and then to enable the website I run the command “**sudo a2ensite default-ssl**”. In last to restart the Apache server I run the command “**sudo service apache2 restart**”.

```
[11/30/21]seed@20I-1955:~$ cd /etc/apache2/sites-available/
[11/30/21]seed@20I-1955:.../sites-available$ sudo apachectl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127
.0.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
[11/30/21]seed@20I-1955:.../sites-available$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
    service apache2 restart
[11/30/21]seed@20I-1955:.../sites-available$ service apache2 restart
Failed to restart apache2.service: Connection timed out
See system logs and 'systemctl status apache2.service' for details.
[11/30/21]seed@20I-1955:.../sites-available$ service apache2 restart
[11/30/21]seed@20I-1955:.../sites-available$ sudo service apache2 restart
[11/30/21]seed@20I-1955:.../sites-available$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
    service apache2 reload
[11/30/21]seed@20I-1955:.../sites-available$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for osamakhalid.com:443 (RSA): *****
[11/30/21]seed@20I-1955:.../sites-available$
```

Figure 50: Enabling the SSL for the website

After that when I visit the URL <https://osamakhalid.com> I found that website loads successfully and displays the webpage that I have created for the website “index.html”.



Figure 51: Website loads successfully

Task-5 Launching a Man-In-The-Middle (MITM) Attack

In this task, I have to launch the MITM attack by setting up the malicious website redhat.com that tries to intercept the traffic between the user and osamakhalid.com.

Step-1 Setting up the malicious website

To set up the malicious website I need to add the VirtualHost entry to the two files “000-default.conf” and “default-ssl.conf” located at “/etc/apache2/sites-available”.



Figure 52: Opening the 000-default.conf file

```
*000-default.conf (/etc/apache2/sites-available) - gedit
[12/01/21]seed@20I-1955:....sites-available$ sudo gedit /etc/apache2/sites-available/000-default.conf
onf
  # Following line enables the CGI configuration for this host only
  # after it has been globally disabled with "a2disconf".
  #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
<VirtualHost *:80>
    ServerName redhat.com
    DocumentRoot /var/www/osamakhalid
    DirectoryIndex index.html
</VirtualHost>
<VirtualHost *:80>
    ServerName osamakhalid.com
    DocumentRoot /var/www/osamakhalid
    DirectoryIndex index.html
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.SeedLabsSQLInjection.com
    DocumentRoot /var/www/SQLInjection
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.xsslabelgg.com
    DocumentRoot /var/www/XSS/Elgg
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.csrflabelgg.com
    DocumentRoot /var/www/CSRF/Elgg
</VirtualHost>
```

Figure 53: Adding the entry in 000-default.conf file

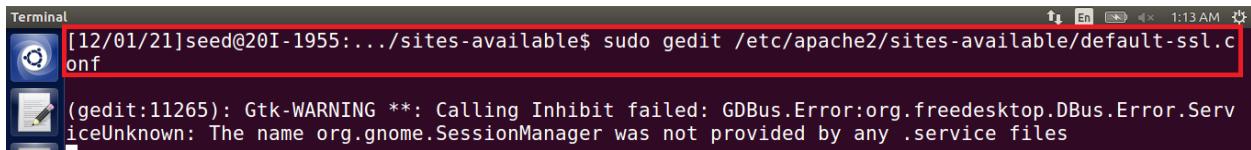


Figure 54: Opening the default-ssl.conf file

```
*default-ssl.conf (/etc/apache2/sites-available) - gedit
[12/01/21]seed@20I-1955:....sites-available$ sudo gedit /etc/apache2/sites-available/default-ssl.conf
onf
(gedit:11265): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files
```

```
# SSL close notify alert is send or allowed to received. This violates
# the SSL/TLS standard but is needed for some brain-dead browsers. Use
# this when you receive I/O errors because of the standard approach where
# mod_ssl sends the close notify alert.
# o ssl-accurate-shutdown:
# This forces an accurate shutdown when the connection is closed, i.e. a
# SSL close notify alert is send and mod_ssl waits for the close notify
# alert of the client. This is 100% SSL/TLS standard compliant, but in
# practice often causes hanging connections with brain-dead browsers. Use
# this only for browsers where you know that their SSL implementation
# works correctly.
# Notice: Most problems of broken clients are also related to the HTTP
# keep-alive facility, so you usually additionally want to disable
# keep-alive for those clients, too. Use variable "nokeepalive" for this.
# Similarly, one has to force some clients to use HTTP/1.0 to workaround
# their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
# "force-response-1.0" for this.
# BrowserMatch "MSIE [2-6]" \
#     nokeepalive ssl-unclean-shutdown \
#     downgrade-1.0 force-response-1.0

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

    SSLEngine On
    SSLCertificateFile /home/seed/Desktop/20I-1955_IS_Assignment_3/osamakhalid_cert.pem
    SSLCertificateKeyFile /home/seed/Desktop/20I-1955_IS_Assignment_3/osamakhalid_key.pem
</VirtualHost>
<VirtualHost *:443>
    ServerName redhat.com
    DocumentRoot /var/www/osamakhalid
    DirectoryIndex index.html

    SSLEngine On
    SSLCertificateFile /home/seed/Desktop/20I-1955_IS_Assignment_3/osamakhalid_cert.pem
    SSLCertificateKeyFile /home/seed/Desktop/20I-1955_IS_Assignment_3/osamakhalid_key.pem
</VirtualHost>
```

Figure 55: Adding the entry in default-ssl.conf file

Step-2 Becoming the man in the middle

Now to become MITM, I have to update the DNS of the victim machine by adding the entry into the “/etc/hosts” file.



Figure 56: opening the /etc/hosts file



Figure 57: Attacker Adding the malicious entry into /etc/hosts file

Step-3 Browse the target website

Now as we all set up, in this task I have to visit the malicious website <https://redhat.com>. Before visiting the website, first I have to run several commands to enable the SSL and then restart the Apache server. For that, I first check the Apache configuration file for errors using the command "**sudo apachectl configtest**". Then to enable the SSL module I run the command "**sudo a2enmod ssl**" and then to enable the website I run the command "**sudo a2ensite default-ssl**". In last to restart the Apache server I run the command "**sudo service apache2 restart**". On visiting the website, I found that the browser is throwing the error "Your Connection is Not Secure". Moreover, on clicking on the advance button I found the error that indicates that redhat.com is using the invalid certificate and the certificate used is not valid for redhat.com. After that, on clicking on "Add Exception" and then clicking on the view button I found that the certificate used is only issued to "osamakhalid.com". This proves that with the use of public key infrastructure MITM attack can be mitigated because the browser checks the correctness of the URL requested and the certificate. The website will only load if it has a valid certificate from CA.

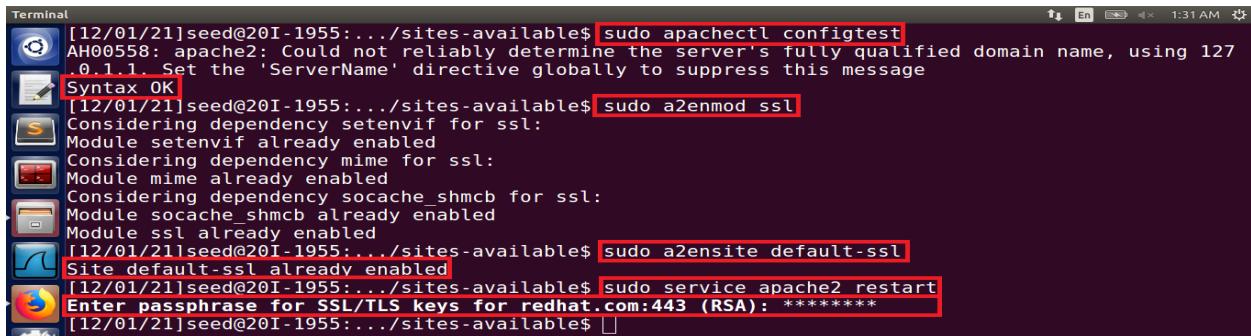


Figure 58: Enabling the SSL for the website

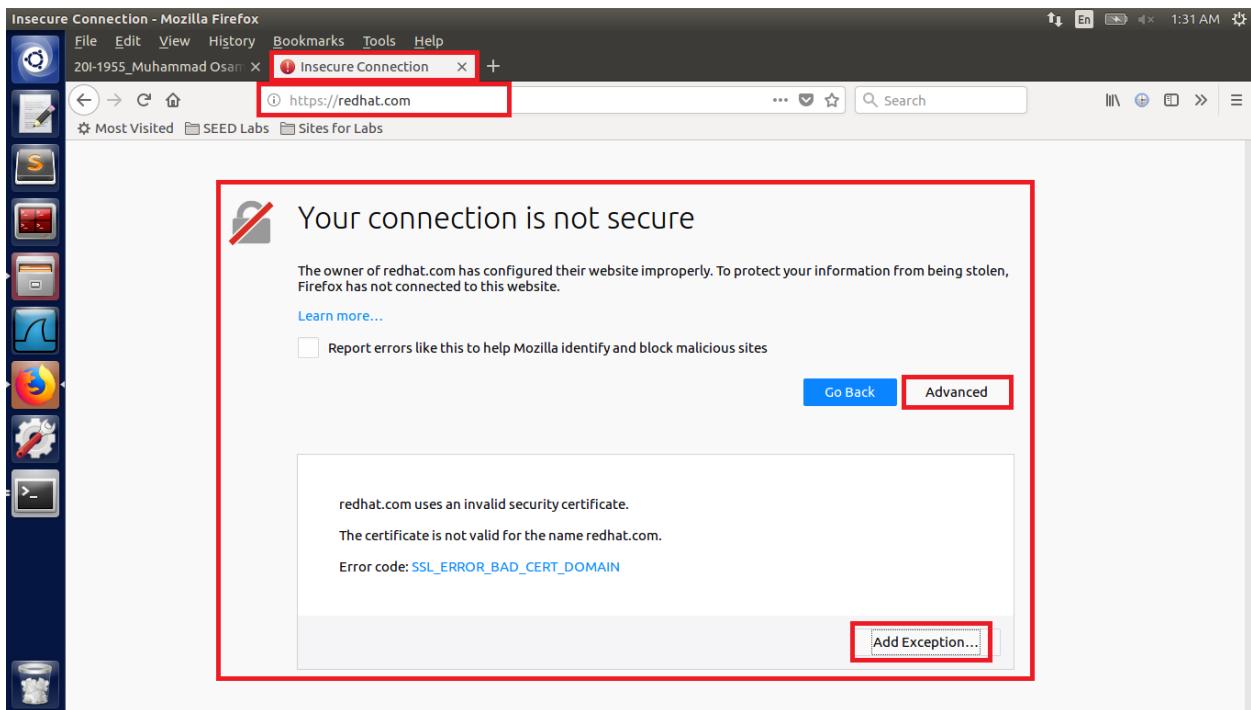


Figure 59:Visiting the website

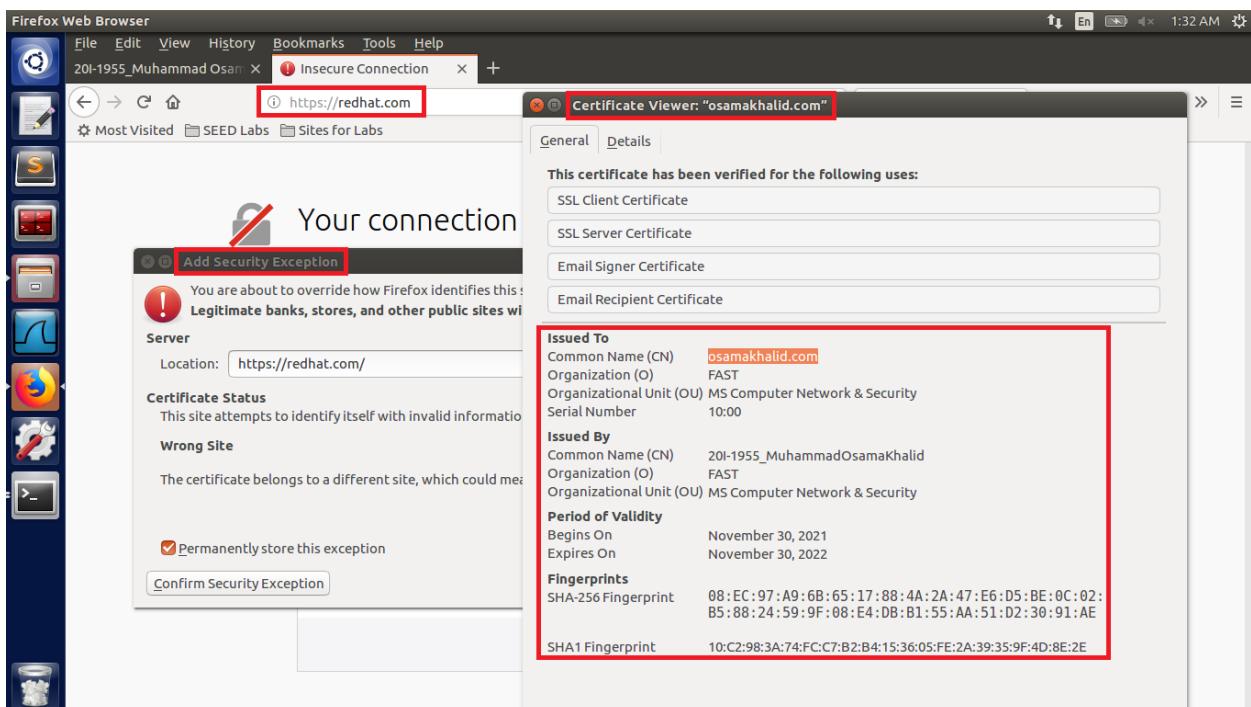
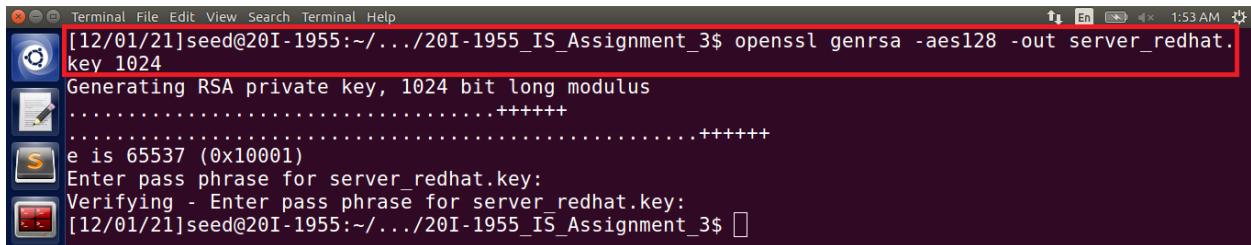


Figure 60: Certificate issued to osamakhalid.com, not redhat.com

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

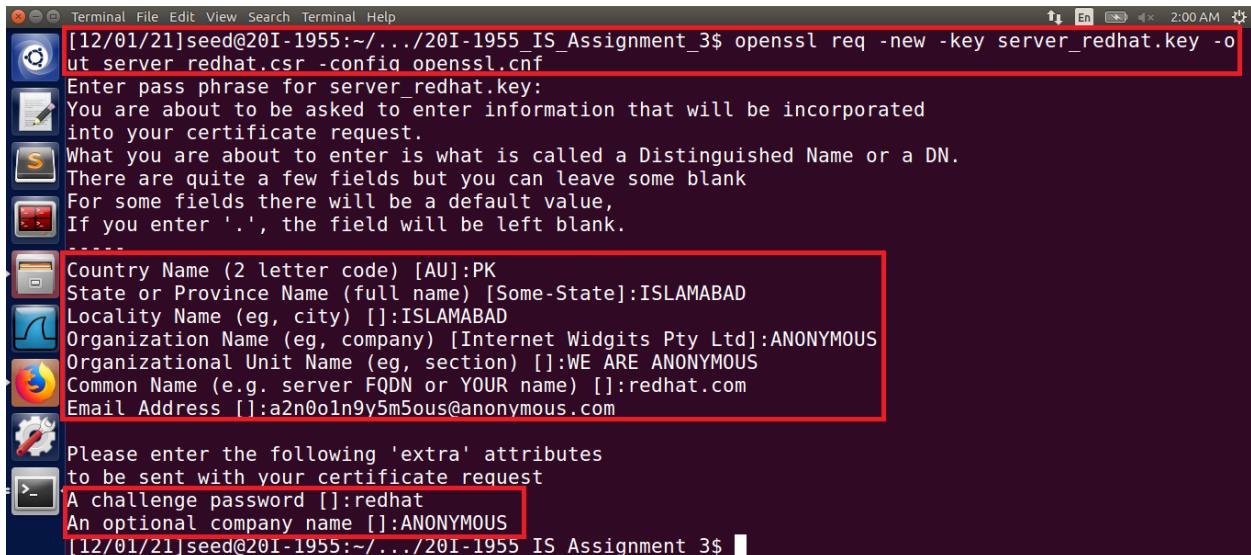
In this task, I have to launch a MITM attack with a compromised CA. As the attacker steals the CA's private key he/she can generate any certificate using this CA's private key. As the attacker gets hands-on CA's private key, he first creates the public and private keys for redhat.com. To create the keys, I use the command "**openssl genrsa -aes128 -out server_redhat.key 1024**".



```
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$ openssl genrsa -aes128 -out server_redhat.key 1024
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
Enter pass phrase for server_redhat.key:
Verifying - Enter pass phrase for server_redhat.key:
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$
```

Figure 61: Attacker Generating public-private key pairs from CA's private key

After generating the public and private key pairs, the attacker's next task is to generate Certificate Signing Request (CSR) that contains a public key generated by the attacker for redhat.com and forward this request to CA. On receiving the request CA will generate the certificate for the key. Now to generate the CSR I use the command "**openssl req -new -key server_redhat.key -out server_redhat.csr -config openssl.cnf**".



```
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$ openssl req -new -key server_redhat.key -out server_redhat.csr -config openssl.cnf
Enter pass phrase for server_redhat.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]:ISLAMABAD
Locality Name (eg, city) [:ISLAMABAD]
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ANONYMOUS
Organizational Unit Name (eg, section) [:WE ARE ANONYMOUS]
Common Name (e.g. server FQDN or YOUR name) [:redhat.com]
Email Address [:a2n0o1n9y5m5ous@anonymous.com]

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password [:redhat]
An optional company name [:ANONYMOUS]
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$
```

Figure 62: Attacker generating CSR request

Now as the attacker has CA's private key and the certificate so the attacker can generate the certificate for himself from the CSR without needing CA and the attacker can do this by running the command "**openssl ca -in server_redhat.csr -out server_redhat.crt -cert ca.crt -keyfile ca.key -config openssl.cnf**". Now as the certificate is signed by CA that is trusted by the browser, therefore browser will also confirm the validity of this certificate.

```
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$ openssl ca -in server_redhat.csr -out server_redhat.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: Dec 1 07:09:43 2021 GMT
        Not After : Dec 1 07:09:43 2022 GMT
    Subject:
        countryName          = PK
        stateOrProvinceName = ISLAMABAD
        localityName        = ISLAMABAD
        organizationName    = ANONYMOUS
        organizationalUnitName = WE ARE ANONYMOUS
        commonName           = redhat.com
        emailAddress         = a2n0o1n9y5m5ous@anonymous.com
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
        CB:0B:3E:2E:69:74:73:6F:88:CE:D1:FD:19:D2:CC:91:FA:7F:39:BE
    X509v3 Authority Key Identifier:
        keyid:D8:46:A4:BA:49:5E:E2:62:D4:B3:C4:24:60:D7:92:F6:A2:B3:7D:3E
```

Figure 63: Attacker generating certificate from CSR using CA's key and certificate

```
Validity
    Not Before: Dec 1 07:09:43 2021 GMT
    Not After : Dec 1 07:09:43 2022 GMT
Subject:
    countryName          = PK
    stateOrProvinceName = ISLAMABAD
    localityName        = ISLAMABAD
    organizationName    = ANONYMOUS
    organizationalUnitName = WE ARE ANONYMOUS
    commonName           = redhat.com
    emailAddress         = a2n0o1n9y5m5ous@anonymous.com
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
        CB:0B:3E:2E:69:74:73:6F:88:CE:D1:FD:19:D2:CC:91:FA:7F:39:BE
    X509v3 Authority Key Identifier:
        keyid:D8:46:A4:BA:49:5E:E2:62:D4:B3:C4:24:60:D7:92:F6:A2:B3:7D:3E
Certificate is to be certified until Dec 1 07:09:43 2022 GMT (365 days)
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$
```

Figure 64: Attacker generating certificate from CSR using CA's key and certificate

After that attacker needs to combine the key (server_redhat.key) and certificate (server_redhat.crt) into one file used by the Apache server and then use this file in the website configuration. To combine the key and certificate into one file I use the following commands “**cp server_redhat.crt osamakhalid_cert.pem**” and “**cp server_redhat.key osamakhalid_key.pem**”.

```
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assianment_3$ ls
ca.crt openssl.cnf      server.crt  server.pem      server_redhat.key
ca.key  osamakhalid_cert.pem  server.csr  server_redhat.crt
demoCA  osamakhalid_key.pem  server.key  server_redhat.csr
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$ cp server_redhat.crt osamakhalid_cert.pem
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$ cp server_redhat.key osamakhalid_key.pem
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$ ls
ca.crt openssl.cnf      server.crt  server.pem      server_redhat.key
ca.key  osamakhalid_cert.pem  server.csr  server_redhat.crt
demoCA  osamakhalid_key.pem  server.key  server_redhat.csr
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$
```

Figure 65: Attacker Combine the certificate and key into one file

After that to save the changes, the attacker restarts the Apache server using the command “**sudo service apache2 restart**”.

```
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for redhat.com:443 (RSA): *****
[12/01/21]seed@20I-1955:~/.../20I-1955_IS_Assignment_3$
```

Figure 66: Attacker Restart the Apache Server

Now on visiting the target website of the attacker browser loads the website successfully without throwing any error because the certificate used is valid and signed by the CA. From here we can conclude that if CA gets compromised then anyone can generate the certificate for himself/herself and mimic any other website they want.



Figure 67: Target website redhat.com loads successfully and display the contents of osamakhalid.com