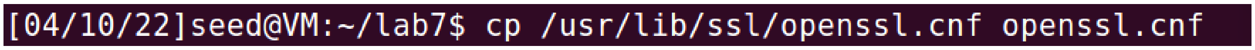
Public Key Infrastructure (PKI Lab)

Lab 7

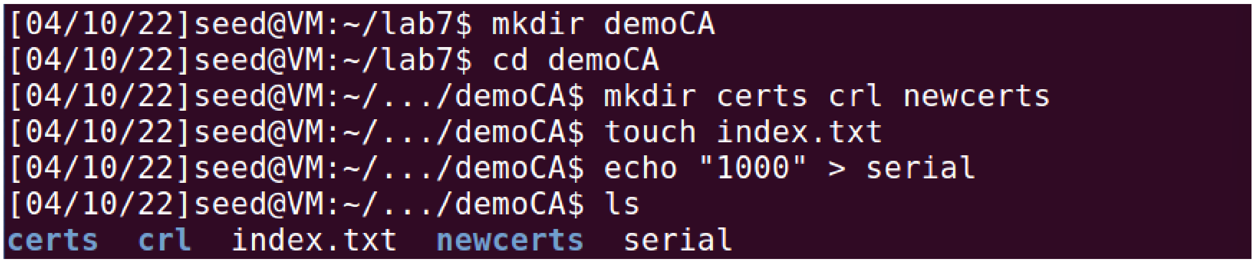
Himanshu Sharma – 202117006

Task 1: Becoming a Certificate Authority (CA)

Creating a directory to hold configuration files and certificates. Copy the configuration file into current directory.

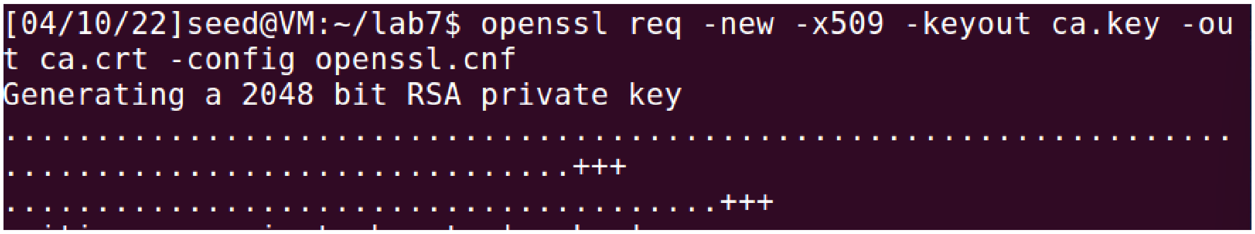


```
[04/10/22]seed@VM:~/lab7$ cp /usr/lib/ssl/openssl.cnf openssl.cnf
```



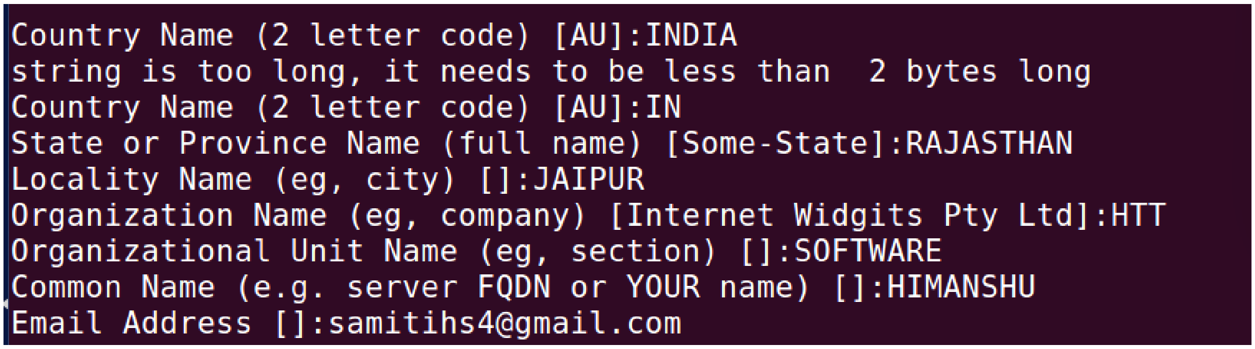
```
[04/10/22]seed@VM:~/lab7$ mkdir demoCA
[04/10/22]seed@VM:~/lab7$ cd demoCA
[04/10/22]seed@VM:~/.../demoCA$ mkdir certs crl newcerts
[04/10/22]seed@VM:~/.../demoCA$ touch index.txt
[04/10/22]seed@VM:~/.../demoCA$ echo "1000" > serial
[04/10/22]seed@VM:~/.../demoCA$ ls
certs  crl  index.txt  newcerts  serial
```

Start to generate self-signed certificate for the CA

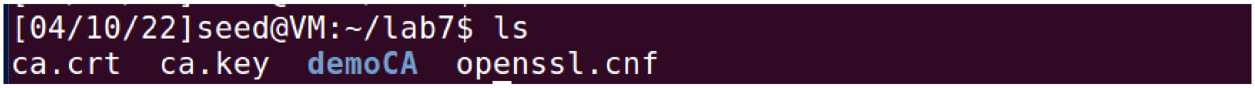


```
[04/10/22]seed@VM:~/lab7$ openssl req -new -x509 -keyout ca.key -ou
t ca.crt -config openssl.cnf
Generating a 2048 bit RSA private key
.....................................................................
..............................+++
......................................+++
```

Fill the appropriate details for certificate. Make country name, state name, organization name same in this policy_match configuration.



```
Country Name (2 letter code) [AU]:INDIA
string is too long, it needs to be less than  2 bytes long
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:RAJASTHAN
Locality Name (eg, city) []:JAIPUR
Organization Name (eg, company) [Internet Widgits Pty Ltd]:HTT
Organizational Unit Name (eg, section) []:SOFTWARE
Common Name (e.g. server FQDN or YOUR name) []:HIMANSHU
Email Address []:samitihs4@gmail.com
```

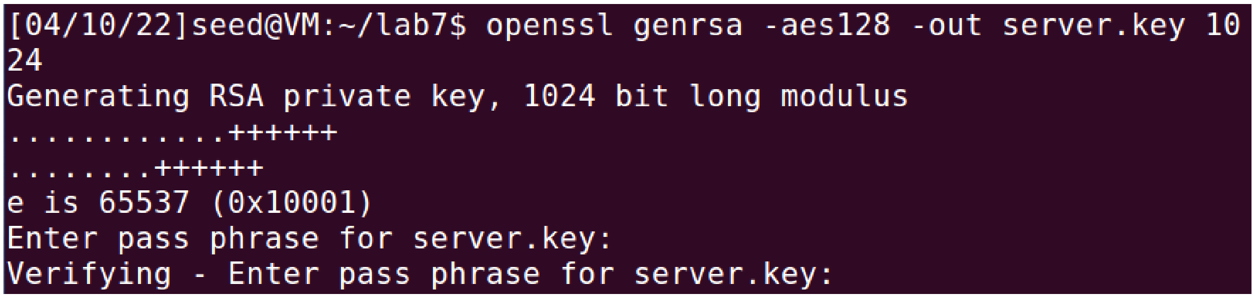


```
[04/10/22]seed@VM:~/lab7$ ls
ca.crt  ca.key  demoCA  openssl.cnf
```

Task 2: Creating a certificate for SEEDPKILab2020.com

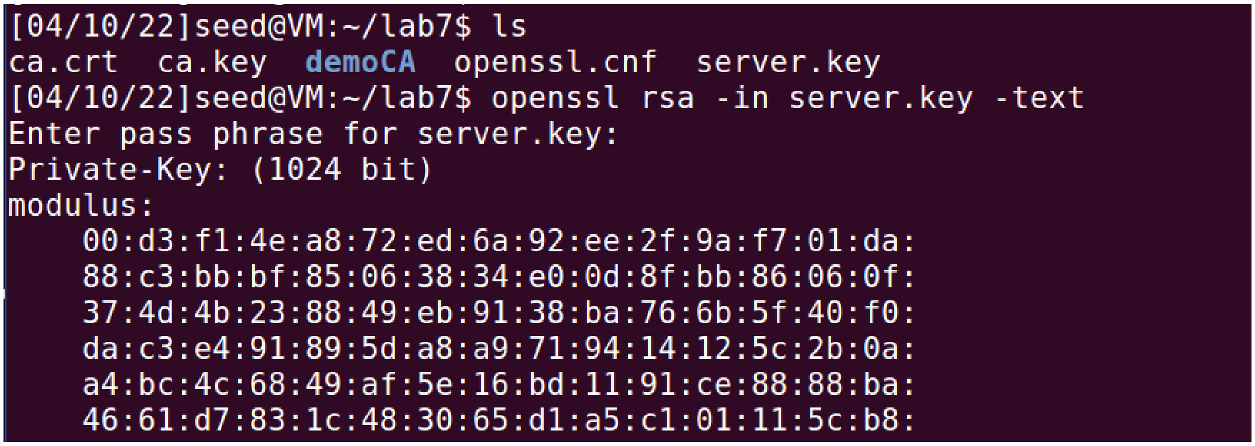
Step1: Generate public/private key pair

We executed the following command to generate key pair and use pass phase to encrypt private key in server.key



```
[04/10/22]seed@VM:~/lab7$ openssl genrsa -aes128 -out server.key 10
24
Generating RSA private key, 1024 bit long modulus
............++++++
........++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
```

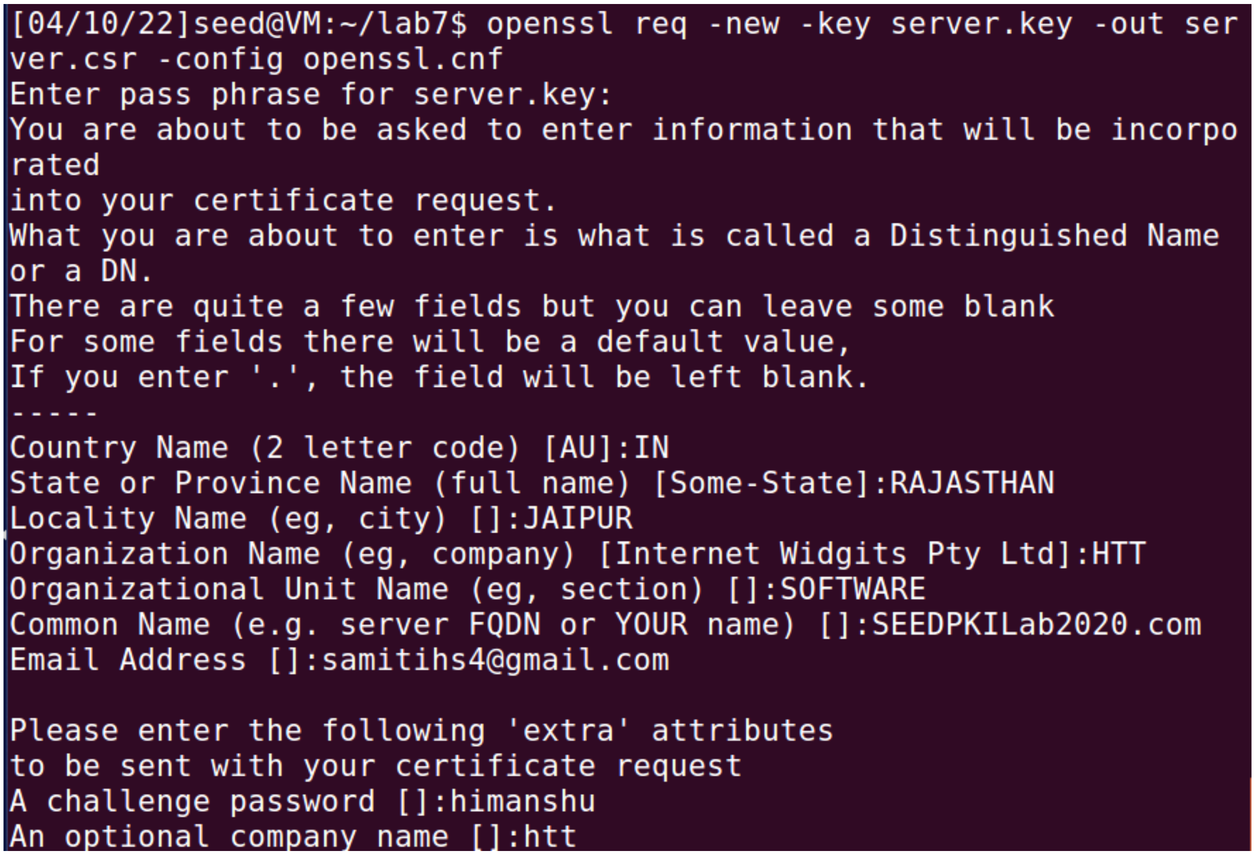
Following command is used to see the content of the file present in server.key



```
[04/10/22]seed@VM:~/lab7$ ls
ca.crt  ca.key  demoCA  openssl.cnf  server.key
[04/10/22]seed@VM:~/lab7$ openssl rsa -in server.key -text
Enter pass phrase for server.key:
Private-Key: (1024 bit)
modulus:
    00:d3:f1:4e:a8:72:ed:6a:92:ee:2f:9a:f7:01:da:
    88:c3:bb:bf:85:06:38:34:e0:0d:8f:bb:86:06:0f:
    37:4d:4b:23:88:49:eb:91:38:ba:76:6b:5f:40:f0:
    da:c3:e4:91:89:5d:a8:a9:71:94:14:12:5c:2b:0a:
    a4:bc:4c:68:49:af:5e:16:bd:11:91:ce:88:88:ba:
    46:61:d7:83:1c:48:30:65:d1:a5:c1:01:11:5c:b8:
```

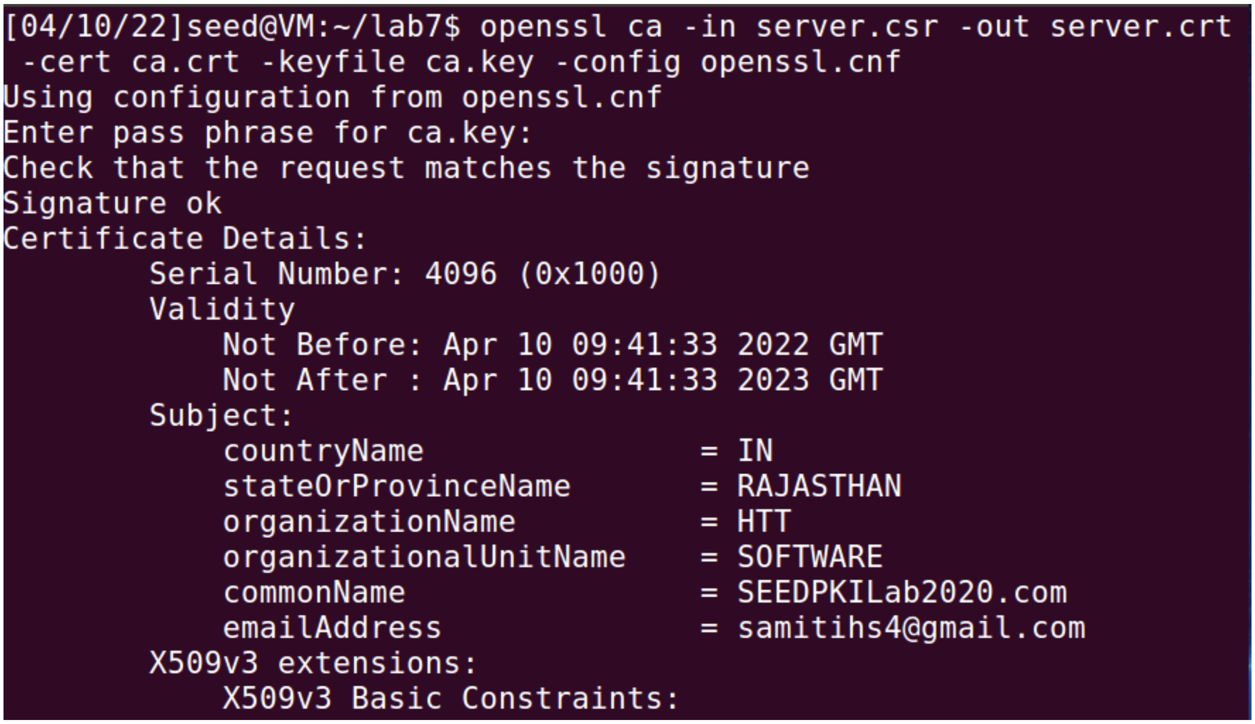
Step 2: Generate a certificate signing request (CSR)

To generate a certificate a signing request is to be generated to sign the keys. Use SEEDPKILab2020.com as common name for the certificate request. While keeping most information same as our root CA.

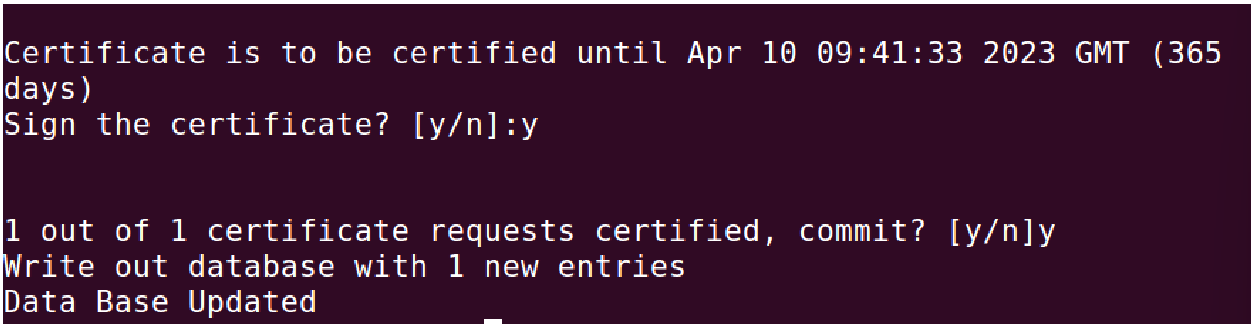


```
[04/10/22]seed@VM:~/lab7$ openssl req -new -key server.key -out ser
ver.csr -config openssl.cnf
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorpo
rated
into your certificate request.
What you are about to enter is what is called a Distinguished 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]:IN
State or Province Name (full name) [Some-State]:RAJASTHAN
Locality Name (eg, city) []:JAIPUR
Organization Name (eg, company) [Internet Widgits Pty Ltd]:HTT
Organizational Unit Name (eg, section) []:SOFTWARE
Common Name (e.g. server FQDN or YOUR name) []:SEEDPKILab2020.com
Email Address []:samitihs4@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:himanshu
An optional company name []:htt
```

Here we are generating the certificate as per the certificate signing request, Our trusted CA will generate certificate and use ca.crt and ca.key to convert server.csr to server.crt



```
[04/10/22]seed@VM:~/lab7$ 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: Apr 10 09:41:33 2022 GMT
            Not After : Apr 10 09:41:33 2023 GMT
        Subject:
            countryName               = IN
            stateOrProvinceName       = RAJASTHAN
            organizationName          = HTT
            organizationalUnitName    = SOFTWARE
            commonName                = SEEDPKILab2020.com
            emailAddress              = samitihs4@gmail.com
        X509v3 extensions:
            X509v3 Basic Constraints:
```



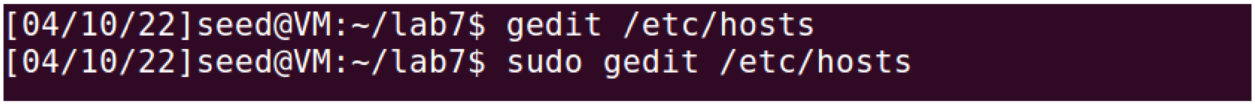
```
Certificate is to be certified until Apr 10 09:41:33 2023 GMT (365
days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Presented all the details of the certificate and accept all condition to generate certificate.

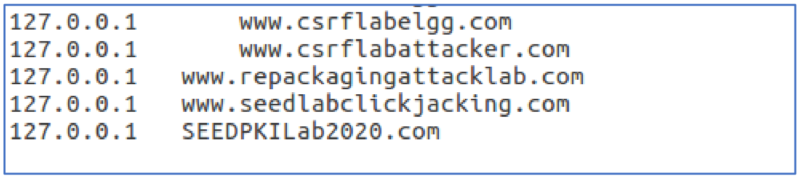
Task 3: Deploying Certificate in an HTTPS Web Server

Step 1: Configuring DNS

To let our computer recognize seedpkilab2020.com as internal domain and direct its traffic to local server we edit /etc/hosts file.

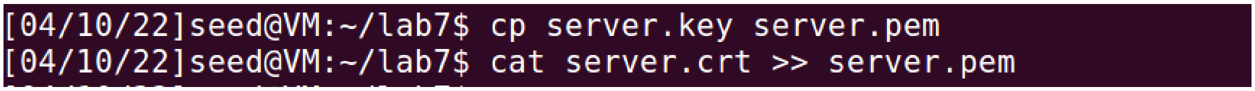


```
[04/10/22]seed@VM:~/lab7$ gedit /etc/hosts
[04/10/22]seed@VM:~/lab7$ sudo gedit /etc/hosts
```



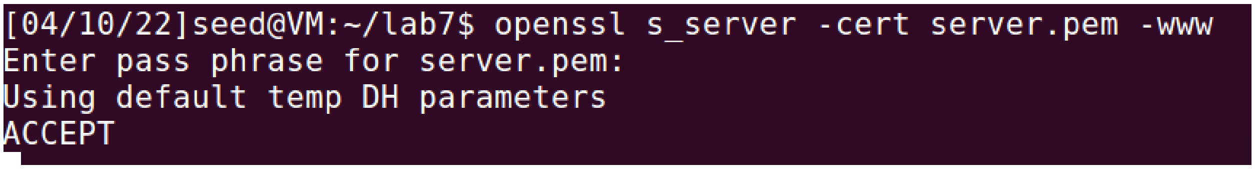
```
127.0.0.1       www.csrflabelgg.com
127.0.0.1       www.csrflabattacker.com
127.0.0.1   www.repackagingattacklab.com
127.0.0.1   www.seedlabclickjacking.com
127.0.0.1   SEEDPKILab2020.com
```

Combine the secret key and certificate into one single file server.pem



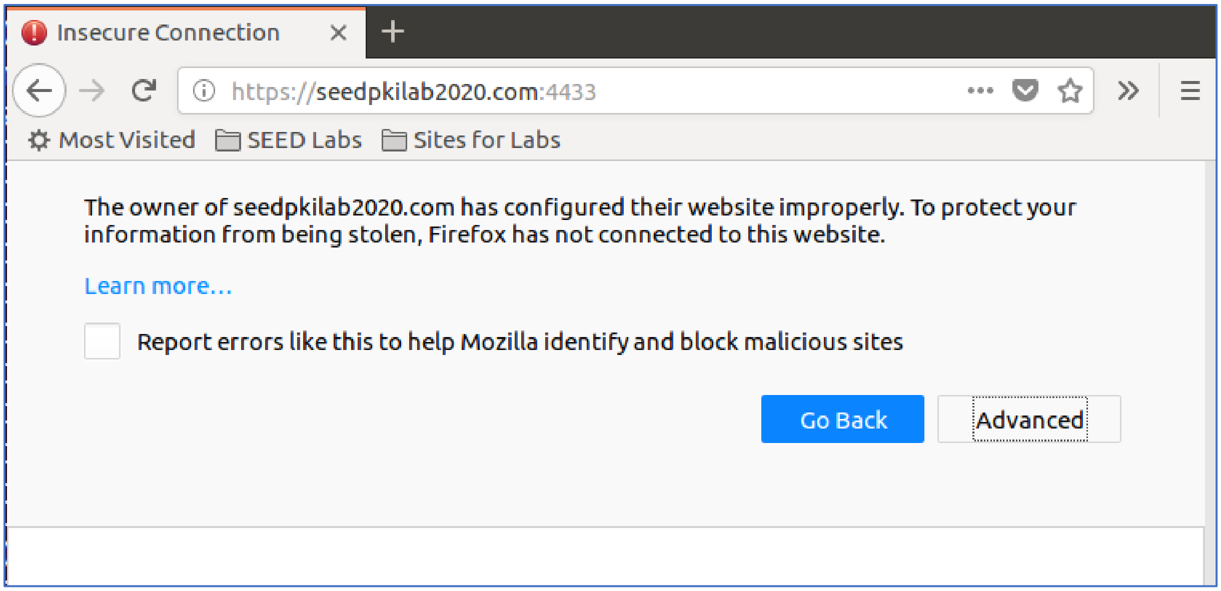
```
[04/10/22]seed@VM:~/lab7$ cp server.key server.pem
[04/10/22]seed@VM:~/lab7$ cat server.crt >> server.pem
```

Launch the web server using server.pem.



```
[04/10/22]seed@VM:~/lab7$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
Using default temp DH parameters
ACCEPT
```

The server is listening on port 4433. Browse https://seedpkilab2018.com:4433



https://seedpkilab2020.com:4433

The owner of seedpkilab2020.com has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

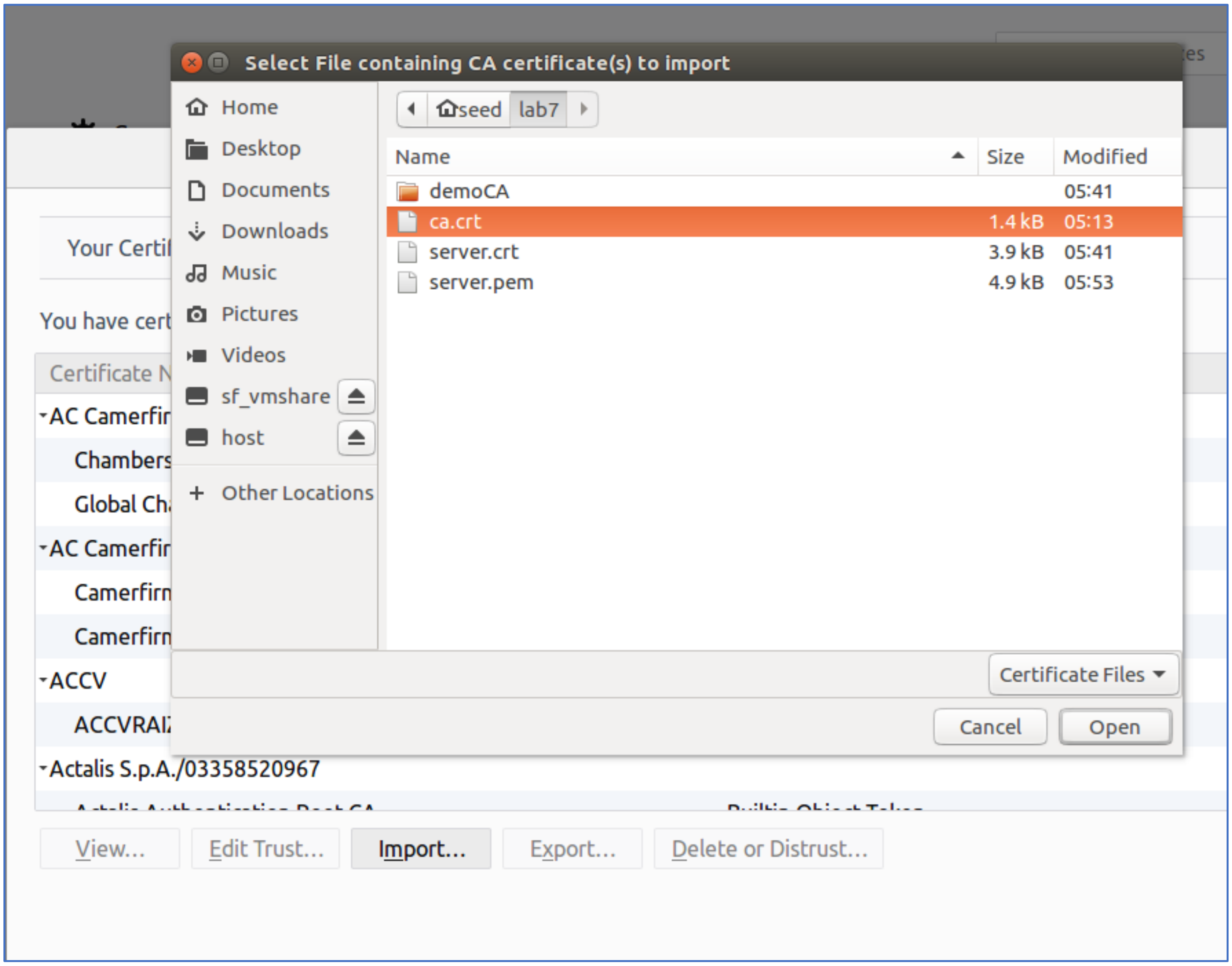
Learn more…

□ Report errors like this to help Mozilla identify and block malicious sites

Go Back Advanced

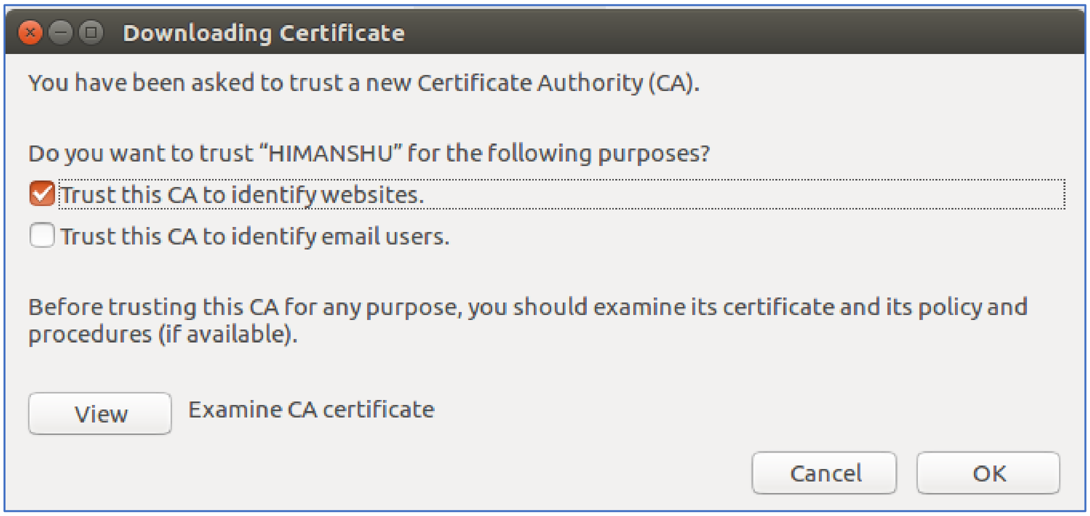
Step 3: Getting the browser to accept our CA certificate

In the previous screenshot we can see the error mentioning the website owner is not configured.



Name	Size	Modified
demoCA		05:41
ca.crt	1.4 kB	05:13
server.crt	3.9 kB	05:41
server.pem	4.9 kB	05:53

It was because our certificate was not included in Firefox certificate repository. We added our CA certificate to the browser.



Downloading Certificate

You have been asked to trust a new Certificate Authority (CA).

Do you want to trust “HIMANSHU” for the following purposes?

☑ Trust this CA to identify websites.

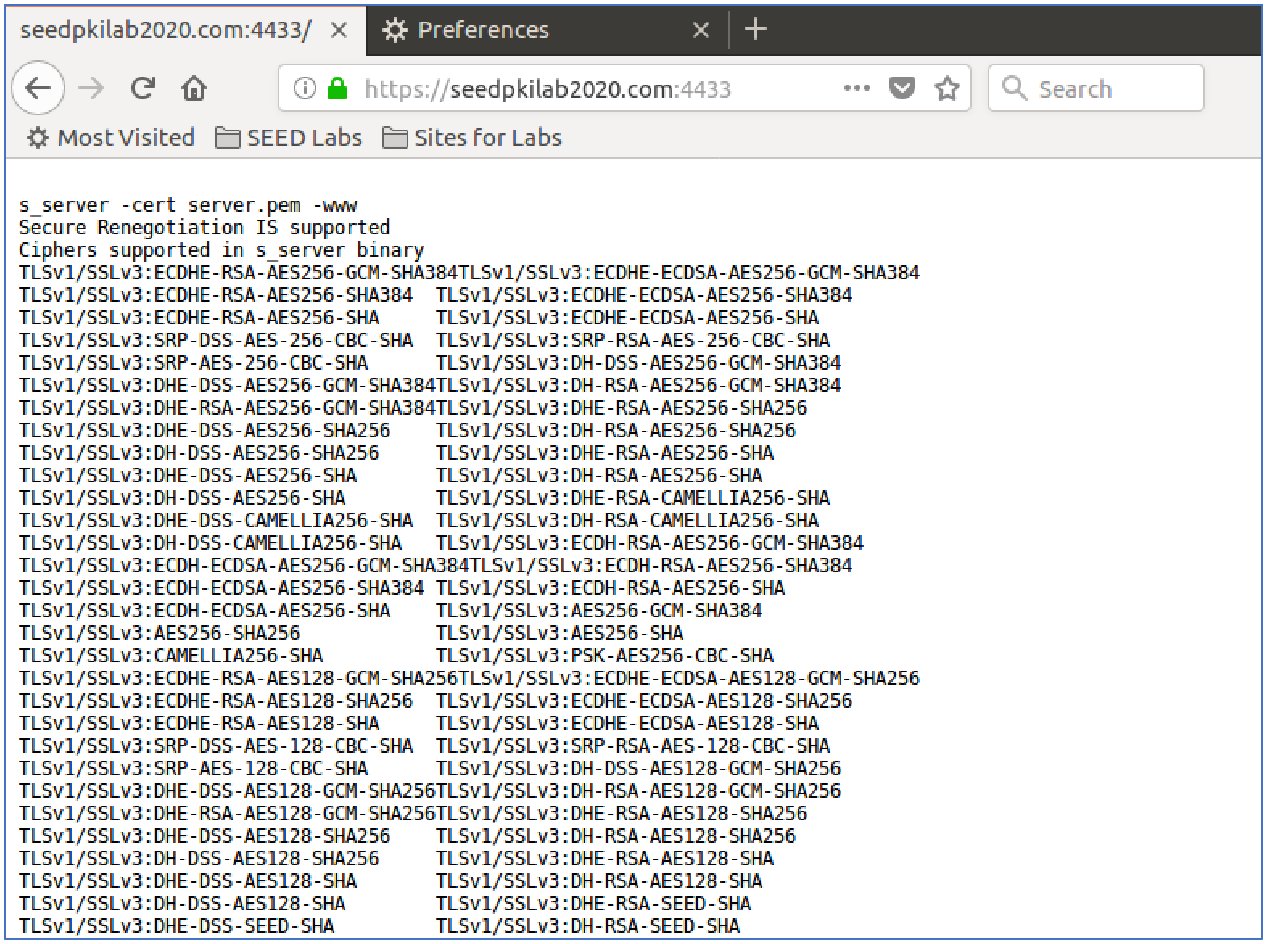
☐ Trust this CA to identify email users.

Before trusting this CA for any purpose, you should examine its certificate and its policy and procedures (if available).

View   Examine CA certificate

Cancel   OK

After adding our certificate to the repo restart the web server. Browser the website.

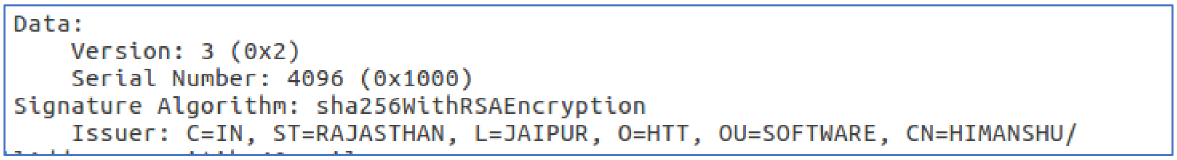


```
s_server -cert server.pem -www
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1/SSLv3:ECDHE-RSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDHE-ECDSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDHE-RSA-AES256-SHA384  TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA384
TLSv1/SSLv3:ECDHE-RSA-AES256-SHA     TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA
TLSv1/SSLv3:SRP-DSS-AES-256-CBC-SHA  TLSv1/SSLv3:SRP-RSA-AES-256-CBC-SHA
TLSv1/SSLv3:SRP-AES-256-CBC-SHA      TLSv1/SSLv3:DH-DSS-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-DSS-AES256-GCM-SHA384TLSv1/SSLv3:DH-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-RSA-AES256-GCM-SHA384TLSv1/SSLv3:DHE-RSA-AES256-SHA256
TLSv1/SSLv3:DHE-DSS-AES256-SHA256    TLSv1/SSLv3:DH-RSA-AES256-SHA256
TLSv1/SSLv3:DH-DSS-AES256-SHA256     TLSv1/SSLv3:DHE-RSA-AES256-SHA
TLSv1/SSLv3:DHE-DSS-AES256-SHA       TLSv1/SSLv3:DH-RSA-AES256-SHA
TLSv1/SSLv3:DH-DSS-AES256-SHA        TLSv1/SSLv3:DHE-RSA-CAMELLIA256-SHA
TLSv1/SSLv3:DHE-DSS-CAMELLIA256-SHA  TLSv1/SSLv3:DH-RSA-CAMELLIA256-SHA
TLSv1/SSLv3:DH-DSS-CAMELLIA256-SHA   TLSv1/SSLv3:ECDH-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDH-ECDSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDH-RSA-AES256-SHA384
TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA384 TLSv1/SSLv3:ECDH-RSA-AES256-SHA
TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA    TLSv1/SSLv3:AES256-GCM-SHA384
TLSv1/SSLv3:AES256-SHA256            TLSv1/SSLv3:AES256-SHA
TLSv1/SSLv3:CAMELLIA256-SHA          TLSv1/SSLv3:PSK-AES256-CBC-SHA
TLSv1/SSLv3:ECDHE-RSA-AES128-GCM-SHA256TLSv1/SSLv3:ECDHE-ECDSA-AES128-GCM-SHA256
TLSv1/SSLv3:ECDHE-RSA-AES128-SHA256  TLSv1/SSLv3:ECDHE-ECDSA-AES128-SHA256
TLSv1/SSLv3:ECDHE-RSA-AES128-SHA     TLSv1/SSLv3:ECDHE-ECDSA-AES128-SHA
TLSv1/SSLv3:SRP-DSS-AES-128-CBC-SHA  TLSv1/SSLv3:SRP-RSA-AES-128-CBC-SHA
TLSv1/SSLv3:SRP-AES-128-CBC-SHA      TLSv1/SSLv3:DH-DSS-AES128-GCM-SHA256
TLSv1/SSLv3:DHE-DSS-AES128-GCM-SHA256TLSv1/SSLv3:DH-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:DHE-RSA-AES128-GCM-SHA256TLSv1/SSLv3:DHE-RSA-AES128-SHA256
TLSv1/SSLv3:DHE-DSS-AES128-SHA256    TLSv1/SSLv3:DH-RSA-AES128-SHA256
TLSv1/SSLv3:DH-DSS-AES128-SHA256     TLSv1/SSLv3:DHE-RSA-AES128-SHA
TLSv1/SSLv3:DHE-DSS-AES128-SHA       TLSv1/SSLv3:DH-RSA-AES128-SHA
TLSv1/SSLv3:DH-DSS-AES128-SHA        TLSv1/SSLv3:DHE-RSA-SEED-SHA
TLSv1/SSLv3:DHE-DSS-SEED-SHA         TLSv1/SSLv3:DH-RSA-SEED-SHA
```

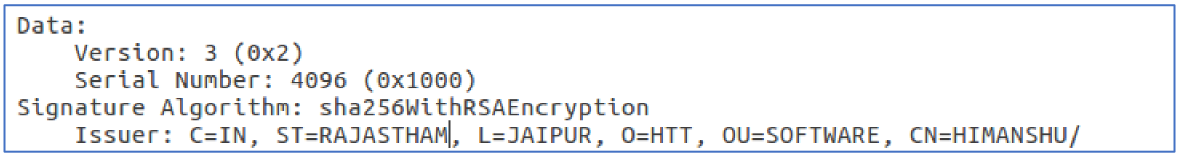
Step 4: Testing our HTTPS website

Now that our website is up and running. We will do following task

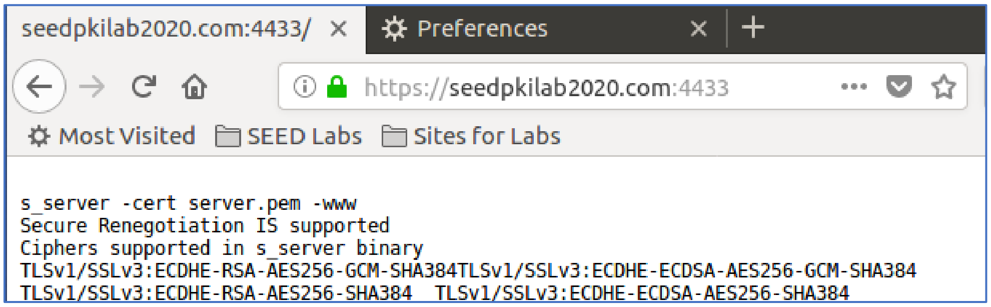
1. Change the byte in the server.pem file and restart the server. In some case the website run smoothly without any changes. But at some places the change in single byte do not restart the server.



```
Data:
    Version: 3 (0x2)
    Serial Number: 4096 (0x1000)
Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=IN, ST=RAJASTHAN, L=JAIPUR, O=HTT, OU=SOFTWARE, CN=HIMANSHU/
```

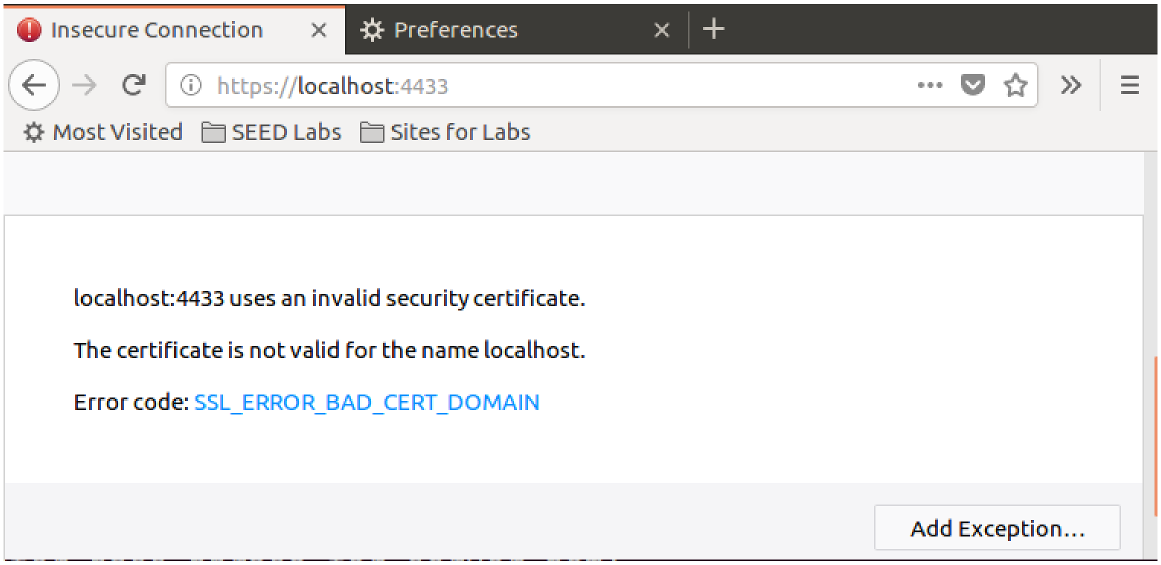


```
Data:
    Version: 3 (0x2)
    Serial Number: 4096 (0x1000)
Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=IN, ST=RAJASTHAM, L=JAIPUR, O=HTT, OU=SOFTWARE, CN=HIMANSHU/
```



```
s_server -cert server.pem -www
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1/SSLv3:ECDHE-RSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDHE-ECDSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDHE-RSA-AES256-SHA384   TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA384
```

2. Use localhost – when browsing https://localhost:4433, it is reported unsafe. Because the localhost has no certificate, the website is using a certificate identified for seedpkilab2020.com



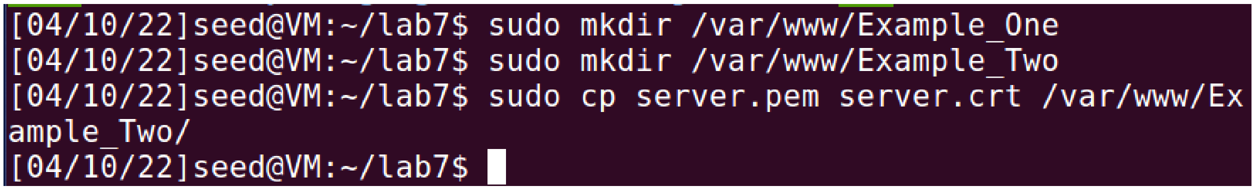
localhost:4433 uses an invalid security certificate.

The certificate is not valid for the name localhost.

Error code: SSL_ERROR_BAD_CERT_DOMAIN

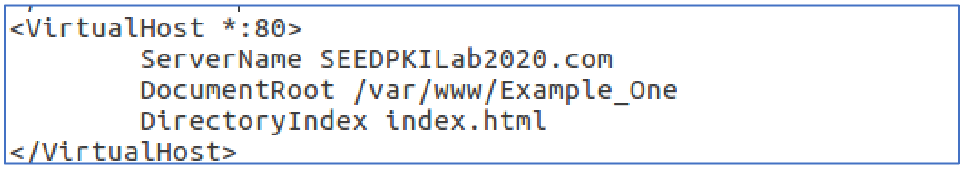
Task 4: Deploying certificate in an Apache-based HTTPS website

We create two directories for https and http protocol website. In https directory we added our server certificate and private key file to the folder.



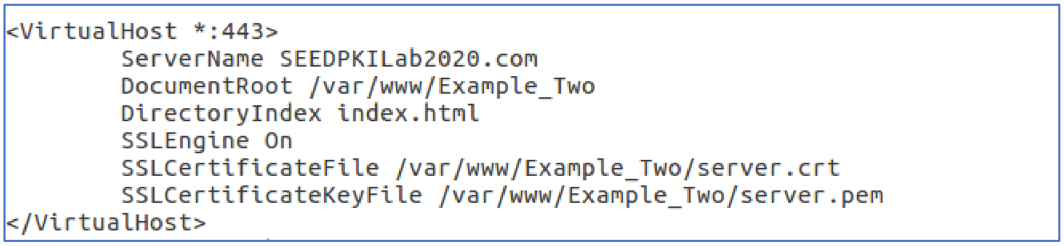
```
[04/10/22]seed@VM:~/lab7$ sudo mkdir /var/www/Example_One
[04/10/22]seed@VM:~/lab7$ sudo mkdir /var/www/Example_Two
[04/10/22]seed@VM:~/lab7$ sudo cp server.pem server.crt /var/www/Ex
ample_Two/
[04/10/22]seed@VM:~/lab7$ 
```

The changes are made in the /etc/apache2/sites-available/000-default.conf to host http website.



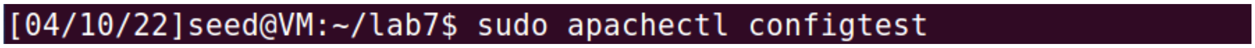
```
<VirtualHost *:80>
        ServerName SEEDPKILab2020.com
        DocumentRoot /var/www/Example_One
        DirectoryIndex index.html
</VirtualHost>
```

The changes are made in the /etc/apache2/sites-available/default-ssl.conf to host http website. These changes are made to configure apache server for our website.



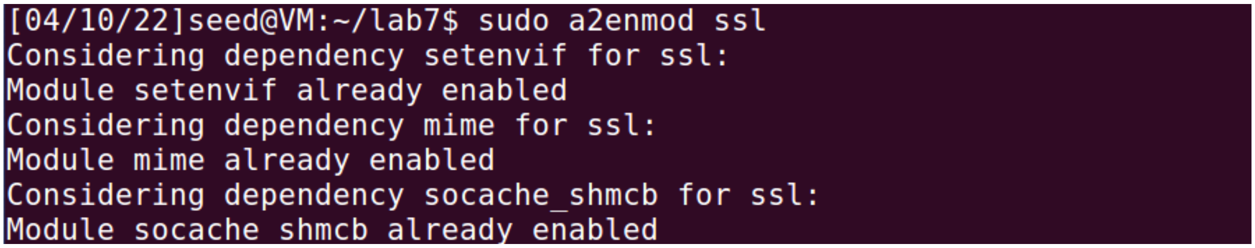
```
<VirtualHost *:443>
        ServerName SEEDPKILab2020.com
        DocumentRoot /var/www/Example_Two
        DirectoryIndex index.html
        SSLEngine On
        SSLCertificateFile /var/www/Example_Two/server.crt
        SSLCertificateKeyFile /var/www/Example_Two/server.pem
</VirtualHost>
```

Now we need to test our configuration and restart our server by following commands



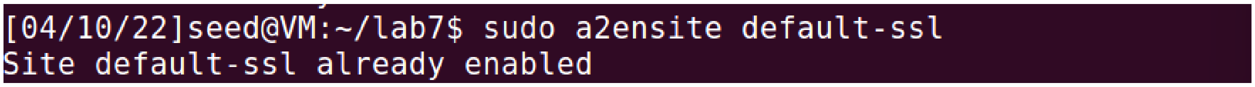
```
[04/10/22]seed@VM:~/lab7$ sudo apachectl configtest
```

Enable SSL module



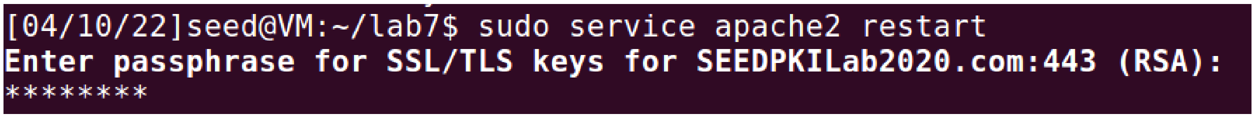
```
[04/10/22]seed@VM:~/lab7$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache shmcb already enabled
```

Enable the site we just edited.



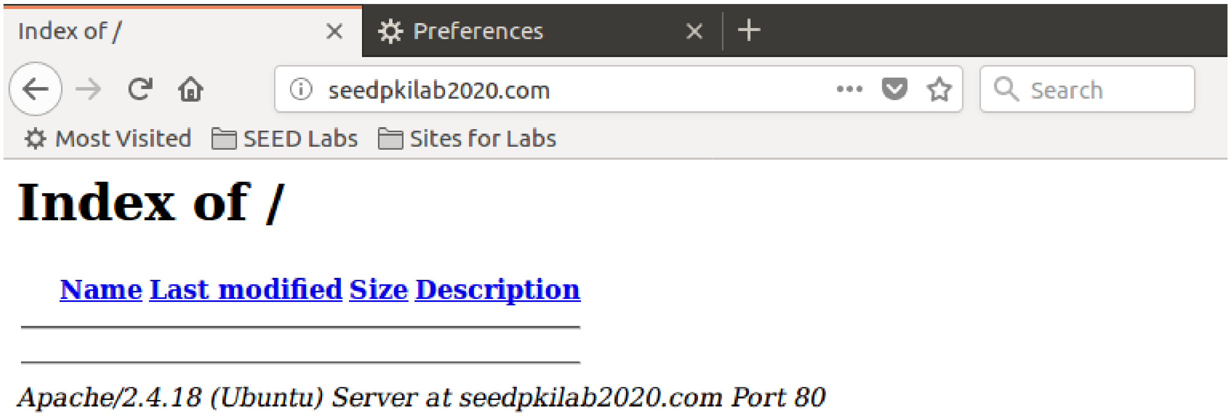
```
[04/10/22]seed@VM:~/lab7$ sudo a2ensite default-ssl
Site default-ssl already enabled
```

Restart Apache server.



```
[04/10/22]seed@VM:~/lab7$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for SEEDPKILab2020.com:443 (RSA):
********
```

Browse the website http://seedpkilab2020.com. It works fine.

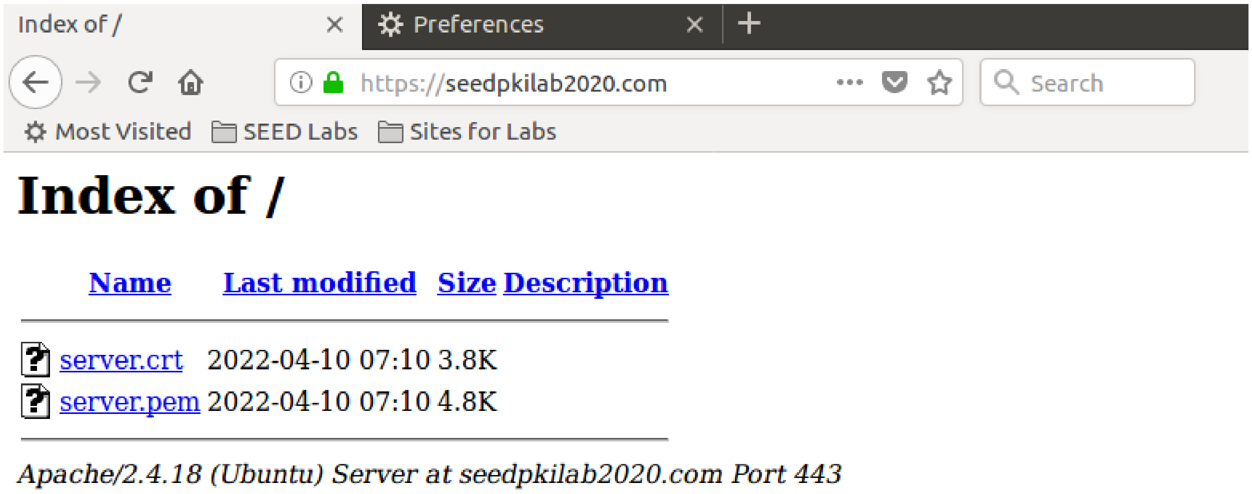


Index of /

Name Last modified Size Description

Apache/2.4.18 (Ubuntu) Server at seedpkilab2020.com Port 80

Browse the website https://seedpkilab2020.com. Our website is up and running.



Index of /

Name	Last modified	Size	Description
server.crt	2022-04-10 07:10	3.8K	
server.pem	2022-04-10 07:10	4.8K	

Apache/2.4.18 (Ubuntu) Server at seedpkilab2020.com Port 443