

Complete Tasks 1-4

Task 1) Copy the configuration file into current directory. Then, make the directories listed in the manual. Create an empty text file in index.txt and put a simple integer number in a file named “serial.”

```
[11/18/21]seed@VM:~/.../PKI$ cp /usr/lib/ssl/openssl.cnf ./openssl.cnf
[11/18/21]seed@VM:~/.../PKI$ ls
openssl.cnf
[11/18/21]seed@VM:~/.../PKI$ mkdir ./demoCA
[11/18/21]seed@VM:~/.../PKI$ cd ./demoCA
[11/18/21]seed@VM:~/.../demoCA$ ls
[11/18/21]seed@VM:~/.../demoCA$ ls ../
demoCA  openssl.cnf
[11/18/21]seed@VM:~/.../demoCA$ mkdir certs
[11/18/21]seed@VM:~/.../demoCA$ mkdir crl
[11/18/21]seed@VM:~/.../demoCA$ mkdir newcerts
[11/18/21]seed@VM:~/.../demoCA$ touch index.txt
[11/18/21]seed@VM:~/.../demoCA$ echo "1000" > serial
[11/18/21]seed@VM:~/.../demoCA$
```

Now we can generate a self-signed certificate by returning to the parent directory and issuing the command below. 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. The passphrase I used was 1358.

```
[11/18/21]seed@VM:~/.../PKI$ 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]:US
State or Province Name (full name) [Some-State]:Maryland
Locality Name (eg, city) []:Baltimore
Organization Name (eg, company) [Internet Widgits Pty Ltd]:JHU
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:Cam
Email Address []:
[11/18/21]seed@VM:~/.../PKI$
```

Task 2)

Step 1: We must generate a public/private key pair. Again, I use the passphrase 1358.

```
[11/18/21]seed@VM:~/.../PKI$ openssl genrsa -aes128 -out server.key 1024
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:
[11/18/21]seed@VM:~/.../PKI$
```

To see the actual content, we must pass the following command and passphrase.

```
[11/18/21]seed@VM:~/.../PKI$ openssl rsa -in server.key -text
Enter pass phrase for server.key:
Private-Key: (1024 bit)
modulus:
 00:a5:3d:c3:e8:83:b1:d5:ff:42:e6:13:f4:bb:f8:
 96:d8:c0:e6:e0:f0:c1:49:a2:80:1a:86:2f:14:91:
 0b:ee:b6:48:f4:7b:59:fa:e5:76:3f:48:09:2d:0b:
 66:41:5b:34:d7:c8:0d:fc:a1:81:ab:19:58:88:5d:
 d0:c2:09:1e:d1:4f:71:4f:0e:f6:81:5d:b5:98:b5:
 01:33:c3:15:75:fa:d9:45:cc:39:09:27:1f:d0:d1:
 04:46:14:ed:06:49:6e:55:2c:ba:5c:9b:6f:e1:3c:
 84:95:a7:38:00:e3:da:e5:1e:c1:2f:d9:c4:02:b7:
 d6:fb:60:92:bf:bb:ae:ec:a1
publicExponent: 65537 (0x10001)
privateExponent:
 65:64:af:0f:bf:af:a1:d6:4e:20:8b:e2:42:6d:59:
 72:8b:7b:a0:4f:5a:eb:6a:11:1d:35:75:32:33:86:
 94:e5:b9:82:7b:26:4d:73:fb:c8:fc:d4:d4:95:08:
 1b:d6:71:68:6e:d3:56:4f:6c:17:a7:27:d2:15:6f:
 01:81:43:2c:c3:b8:74:07:cd:9f:9b:c7:a9:7e:bb:
 f2:6d:83:82:e8:5c:60:1a:35:9c:11:0a:ad:b0:49:
 3d:bf:af:af:83:76:39:d9:07:7e:68:75:a7:5c:a7:
 60:7b:e4:0c:c9:fa:56:3f:9d:7d:bb:95:7e:97:2b:
 a9:4b:a3:f5:43:2d:71:d9
prime1:
 00:d9:a0:99:b5:a8:52:e9:99:0d:7e:32:95:28:2b:
 d3:c5:43:7a:32:be:f8:c4:96:a2:74:ba:0e:2e:4f:
 89:25:8c:28:c2:01:54:69:e7:30:9f:c3:7a:ba:cd:
 af:11:b9:20:db:76:d8:b7:17:39:d5:bb:a0:6e:f5:
 03:76:90:fd:4f
prime2:
 00:c2:60:87:83:2e:c9:d3:15:51:13:e9:9d:d1:b6:
 07:53:a6:51:7f:3c:fc:c6:0e:67:23:79:6d:8f:50:
 f6:6a:b3:2b:f9:db:db:77:78:44:53:60:32:0f:de:
 c6:19:d7:eb:97:89:f7:3b:67:b9:d1:32:82:f6:ee:
 1b:27:4b:5b:0f
exponent1:
 25:d7:ab:62:cf:36:3b:b0:85:8c:20:37:97:98:a7:
 66:71:e9:59:d1:a3:68:6b:d9:2b:fa:fe:64:47:28:
 4e:03:25:43:36:ff:fa:ba:5c:d6:2b:23:b8:3f:f3:
 e5:5d:0f:bd:99:d8:ed:5d:78:8d:15:e7:3b:e4:17:
 0a:13:04:39
exponent2:
 00:88:9c:e9:81:44:a8:2a:c8:27:a4:d4:23:cc:99:
 a0:0a:a5:b1:4f:b9:20:7b:5e:f2:14:57:aa:ad:f8:
 9c:48:1f:c7:7f:a3:8b:5b:2b:56:f2:36:80:d8:f0:
 9c:d0:db:15:c4:fa:fd:9d:6d:e8:86:64:76:f1:70:
 25:fb:77:15:f7
coefficient:
 6d:ee:f5:d8:2d:9a:c2:62:fb:5d:b5:e1:fc:0e:7d:
 7e:64:21:0f:fd:e8:5c:3f:48:47:7b:ef:b6:16:a6:
 c4:1f:2d:0f:f4:cc:ab:06:e8:ed:f2:ff:c0:67:b2:
 d8:59:15:4c:cd:eb:9e:f0:12:bc:c4:b3:10:3b:e8:
 12:4b:be:e1
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICXABAAKAgQCLPcPog7HV/0LmE/S7+JbYw0bg8MFJooAahi8Uk0vutkj0e1n6
5XY/SakTc2ZBwzTXyA38oYGrGviIXdDCCR7RT3FPDvaBXbwYt0EzwxV1+tlfZDkJ
Jx/Q0QRGF00G5W5VLLpcm2/hPISVpzgA49rLHsEv2cQCt9b7YJK/u67soQIDAQAB
AoGAZWsvD7+vodZOIiviQm1Zcot7oE9a62oRHTV1Mj0GL0W5gnsMTXP7yPzU1JUI
```

Step 2: Now we must generate a certificate signing request

```
[11/18/21]seed@VM:~/.../PKI$ 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]:US
State or Province Name (full name) [Some-State]:Maryland
Locality Name (eg, city) []:Baltimore
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:SEEDPKILab2020.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[11/18/21]seed@VM:~/.../PKI$
```

I used SEEDPKILab2020.com as the common name. The new CSR is saved in server.csr, which basically includes the company's public key. Again, the passphrase for server.key is 1358. The CSR will be sent to the CA, who will generate a certificate for the key ensuring that identity info in the CSR matches the server's true identity.

Step 3: Now we must generate the certificate using the previous information

We use our own trusted CA to generate certificates. Ca.crt and ca.key will convert server.csr to server.crt using the below command. For future reference, all information in the request and the CA certificate per the policy "policy_match" in openssl.cnf

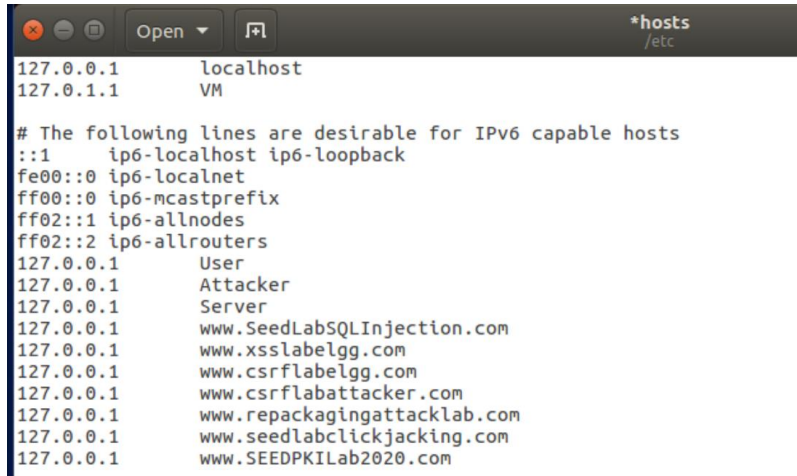
```
[11/18/21]seed@VM:~/.../PKI$ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 4096 (0x1000)
  Validity
    Not Before: Nov 18 14:34:28 2021 GMT
    Not After : Nov 18 14:34:28 2022 GMT
  Subject:
    countryName           = US
    stateOrProvinceName   = Maryland
    organizationName       = JHU
    commonName             = SEEDPKILab2020.com
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    Netscape Comment:
      OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
      D8:63:69:B0:EE:E8:1A:5B:67:CB:1C:A7:69:45:C3:A4:8D:68:5F:10
    X509v3 Authority Key Identifier:
      keyid:27:05:9B:2C:39:A3:DF:2E:48:03:63:19:1F:19:D0:53:C6:E9:D0:9C

Certificate is to be certified until Nov 18 14:34:28 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
[11/18/21]seed@VM:~/.../PKI$
```


Task 3) Deploying certificate in HTTPS web server**Step 1: Configuring DNS**

We'll open and edit /etc/hosts to include SEEDPKILab2020.com



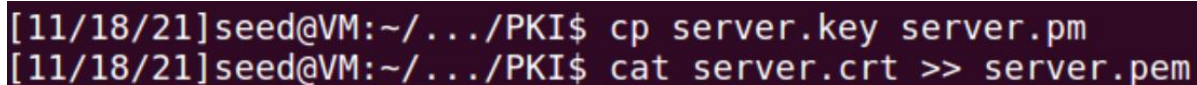
```
*hosts
/etc

127.0.0.1    localhost
127.0.1.1    VM

# 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.csrfattacklab.com
127.0.0.1    www.repackagingattacklab.com
127.0.0.1    www.seedlabclickjacking.com
127.0.0.1    www.SEEDPKILab2020.com
```

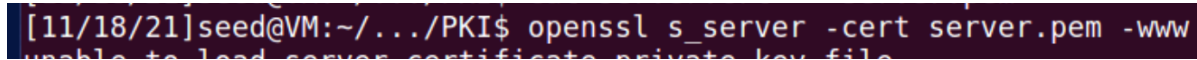
Step 2: Configuring the web server

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



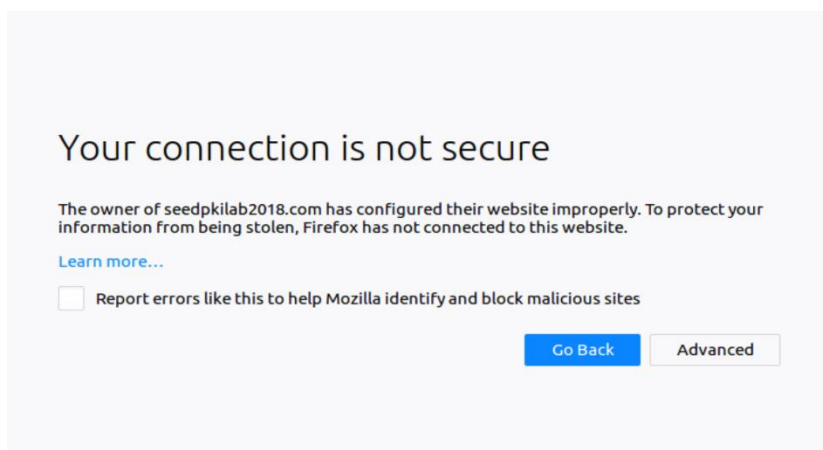
```
[11/18/21]seed@VM:~/.../PKI$ cp server.key server.pm
[11/18/21]seed@VM:~/.../PKI$ cat server.crt >> server.pem
```

Then we can launch the webserver using server.pem



```
[11/18/21]seed@VM:~/.../PKI$ openssl s_server -cert server.pem -www
```

We can go to our browser and go to seedpkilab2020.com:4433 . Firefox does not want to connect because we must manually accept our CA certificate.



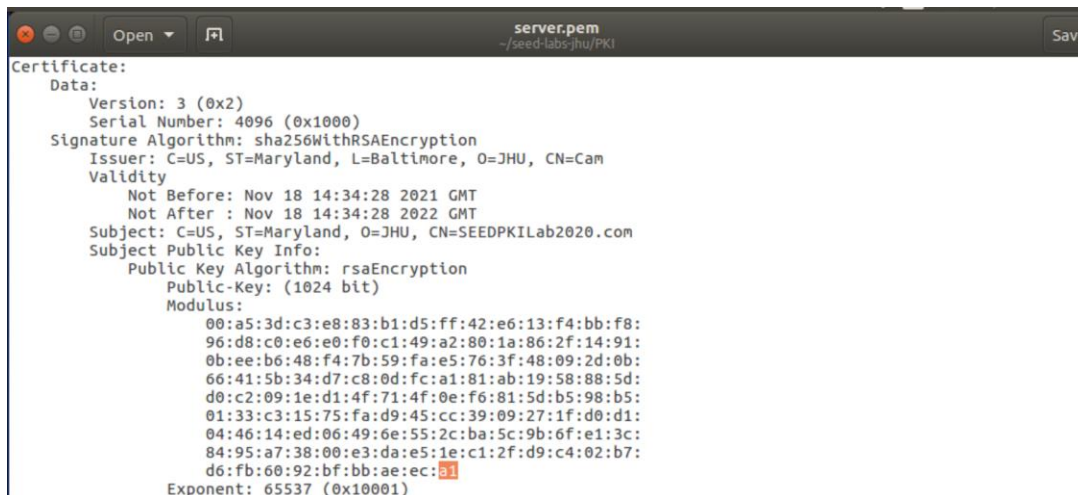
Step 3: Getting our browser to accept the CA certificate

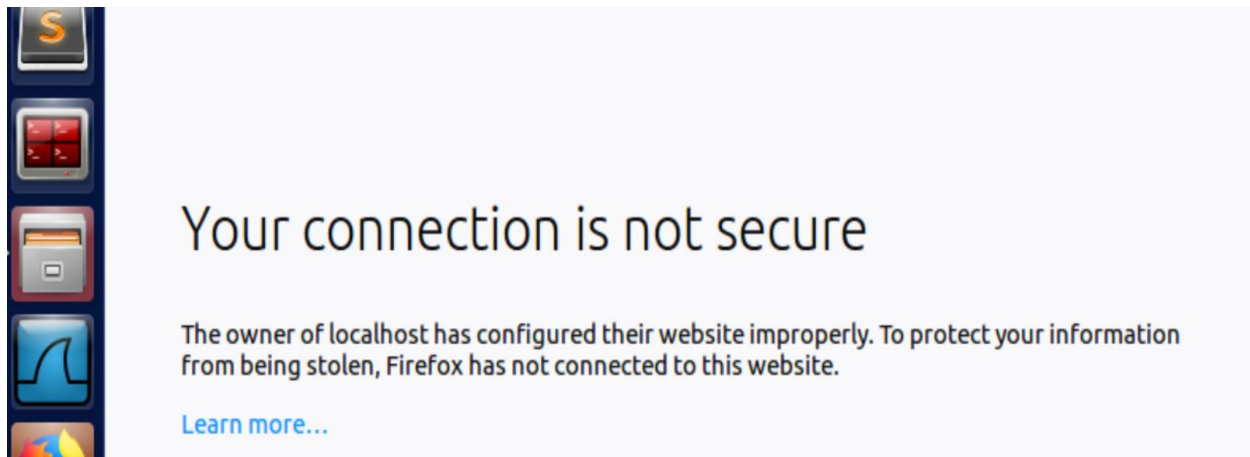
Search for "certificate" in Firefox's Preferences page, click on "View Certificates" and enter "certificate manager", click on "Authorities tab" and import CA.crt. Check "Trust this CA to identify web sites". This is what shows when you reload seedpkilab2020.com:4433

```
s_server -cert server.pem -www
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1/SSLv3:ECDHE-RSA-AES256-GCM-SHA384 TLSv1/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-SHA384 TLSv1/SSLv3:DH-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-RSA-AES256-GCM-SHA384 TLSv1/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:ECDSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDH-ECDSA-AES256-GCM-SHA384 TLSv1/SSLv3:ECDSA-AES256-SHA384
TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA384 TLSv1/SSLv3:ECDSA-AES256-SHA
TLSv1/SSLv3: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:ECDSA-AES128-GCM-SHA256 TLSv1/SSLv3:ECDSA-AES128-GCM-SHA256
TLSv1/SSLv3:ECDSA-AES128-SHA256 TLSv1/SSLv3:ECDSA-AES128-SHA256
TLSv1/SSLv3:ECDSA-AES128-SHA TLSv1/SSLv3: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-SHA256 TLSv1/SSLv3:DH-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:DHE-RSA-AES128-GCM-SHA256 TLSv1/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
```

Step 4: Testing our HTTPS site

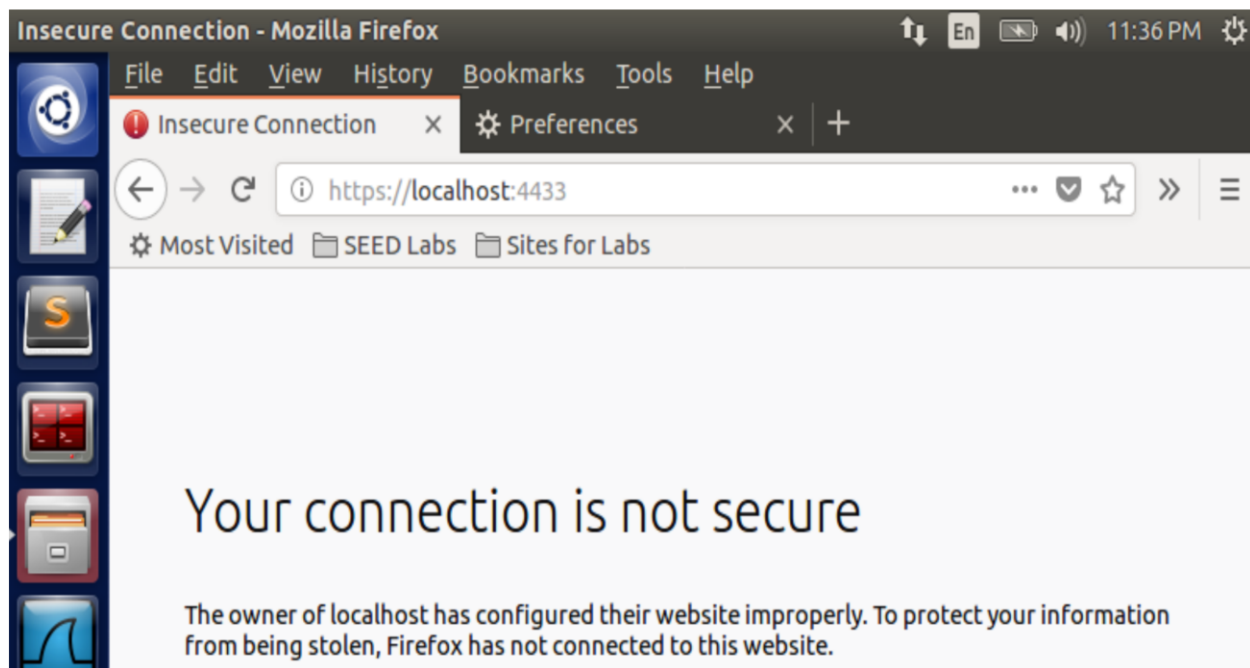
We'll modify one byte in server.pem by `sudo gedit server.pem`. The highlighted byte was originally `a1`, we made it `a2`. The certificate is rendered useless by firefox. It is unsafe to go to.





By modifying a byte in the certificate, the certificate becomes invalid. Even though most bytes make no differences after corruption, this one happened to render the certificate invalid.

Now, I'll use localhost to browse to our site, hosted on localhost:4433. Again, it is reported as unsafe https because localhost has no certificate, the website is using a certificate identified for seedpkilab2020.com.



Task 4) Deploying certificate in an apache-based https website.

I open the config file of the apache https server and I add the entry in the manual.

```
[11/18/21]seed@VM:~/.../PKI$ sudo gedit /etc/apache2/sites-available/default-ssl.conf
```

```
<VirtualHost *:443>
    ServerName SEEDPKILab2018.com
    DocumentRoot /var/www/pki
    DirectoryIndex index.html

    SSLEngine On
    SSLCertificateFile /var/www/pki/server.crt
    SSLCertificateKeyFile /var/www/pki/server.pem
</VirtualHost>
```

Then I copy the server cert and private key to the folder where the apache2 server will look. I test the apache configuration and it is successful. It returned a warning about a different lab clickjacking, which is not relevant here.

```
[11/18/21]seed@VM:~/.../PKI$ sudo mkdir /var/www/pki
[11/18/21]seed@VM:~/.../PKI$ sudo cp server.pem server.crt /var/www/pki
[11/18/21]seed@VM:~/.../PKI$ sudo apachectl configtest
AH00112: Warning: DocumentRoot [/var/www/seedlabclickjacking] does not exist
```

I enable SSL, I enable the site we have just edited and added to the apache2 server, and then I restart the apache2 server.

```
[11/18/21]seed@VM:~/.../PKI$ 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
[11/18/21]seed@VM:~/.../PKI$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
    service apache2 reload
[11/18/21]seed@VM:~/.../PKI$ sudo service apache2 restart
[11/18/21]seed@VM:~/.../PKI$
```

Afterward, I test out our site on Firefox and it runs properly!

