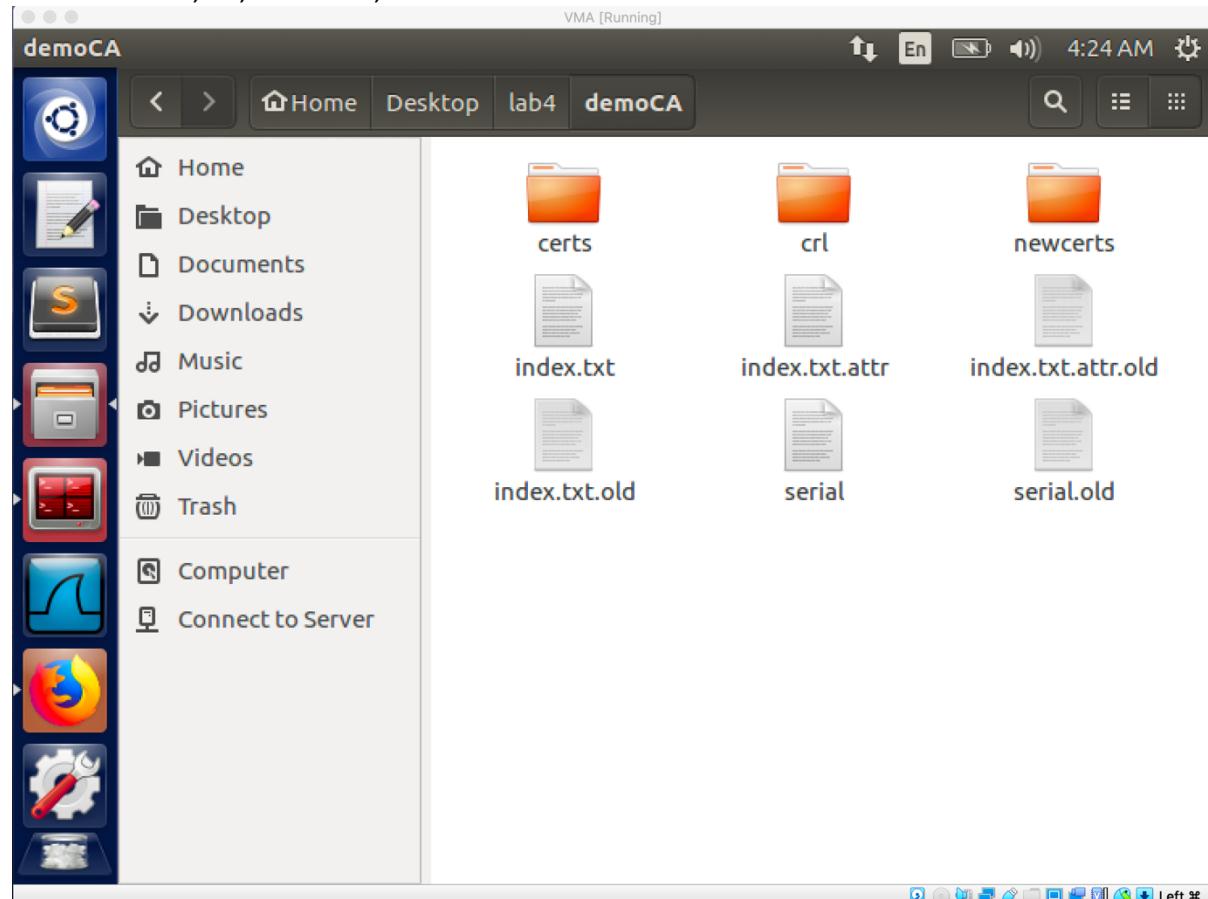


### Task1: Becoming a Certificate Authority (CA)

In order to use OpenSSL to create certificates we need a config file which is openssl.conf, I make a copy of openssl.conf to my current directory which is '/home/seed/Desktop/lab4' and I named it by openssl1.com. I also create a sub-directory which is called demoCA, it contains certs, crt, newcerts, index.txt and serial.



After creating the directory, I use the command 'openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl1.cnf' to generate a self-signed certificate for the CA. Two files will be generated, ca.key contains the private key of the CA, ca.crt contains the public key of the CA.

```
-----  
Country Name (2 letter code) [AU]:Singapore  
string is too long, it needs to be less than 2 bytes long  
Country Name (2 letter code) [AU]:SG  
State or Province Name (full name) [Some-State]:Singapore  
Locality Name (eg, city) []:singapore  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:aaron  
Organizational Unit Name (eg, section) []:aaron  
Common Name (e.g. server FQDN or YOUR name) []:zewen.com  
Email Address []:www.zewen.com  
[02/26/21]seed@VM:~/.../lab4$ ls  
ca.crt certs index.txt openssl1.cnf  
ca.key crt newcerts serial
```

### Task2: Creating a Certificate for SEEDPKILab2020.com

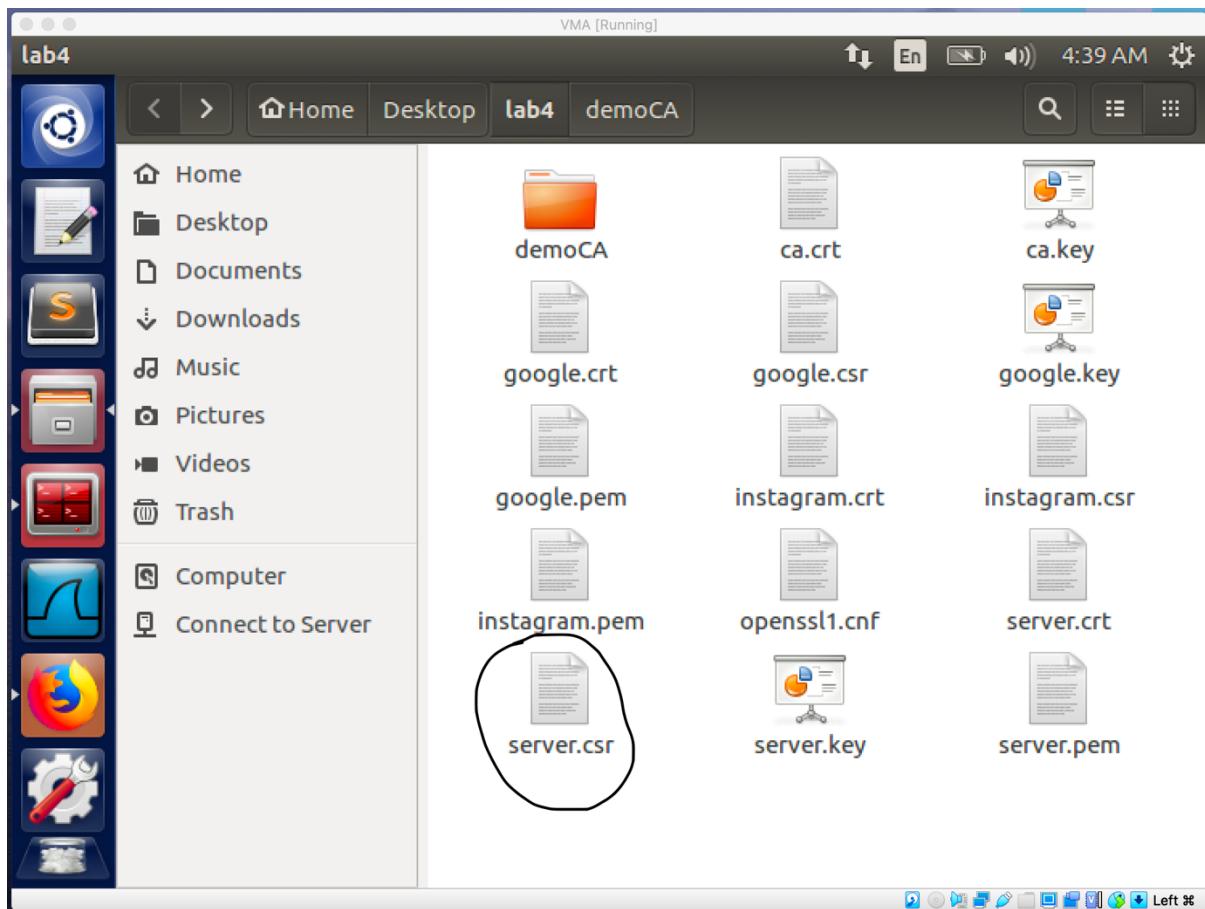
Step1:

I use command ‘openssl genrsa -aes128 -out server.key 1024’ to generate a public/private key pair which encrypted by aes128 algorithm. A server.key file will be outputted. We can check the modulo, exponent by using command ‘openssl rsa -in server.key -text’.

```
[02/28/21]seed@VM:~/.../lab4$ openssl rsa -in server.key -text
Enter pass phrase for server.key:
Private-Key: (1024 bit)
modulus:
    00:d6:c2:a9:4b:ae:f3:e2:a0:40:9d:14:02:88:db:
    8e:93:6a:64:63:9d:6f:c5:1f:4e:7c:dc:b3:b5:0c:
    51:b3:33:5c:28:cc:9d:3e:fc:fc:96:90:23:7e:8c:
    e1:d6:33:f3:6b:11:cf:8b:48:0f:78:35:b8:e3:6b:
    4f:4c:df:43:29:c5:18:6c:d5:a2:ef:e9:55:19:0a:
    dd:e9:76:d4:d2:58:cc:79:36:08:9b:f2:a2:3f:06:
    61:92:82:17:86:e4:9b:5b:c7:95:b0:74:a3:89:d4:
    a8:b4:c6:d3:ac:ac:2b:e1:c4:c0:98:71:90:91:9c:
    af:25:68:c8:a4:13:28:cd:69
publicExponent: 65537 (0x10001)
privateExponent:
    00:c7:ef:80:17:12:b0:c0:fa:7a:4d:02:74:fd:9e:
    91:d4:d2:06:d5:ea:9b:44:c4:49:98:db:0b:7e:f1:
```

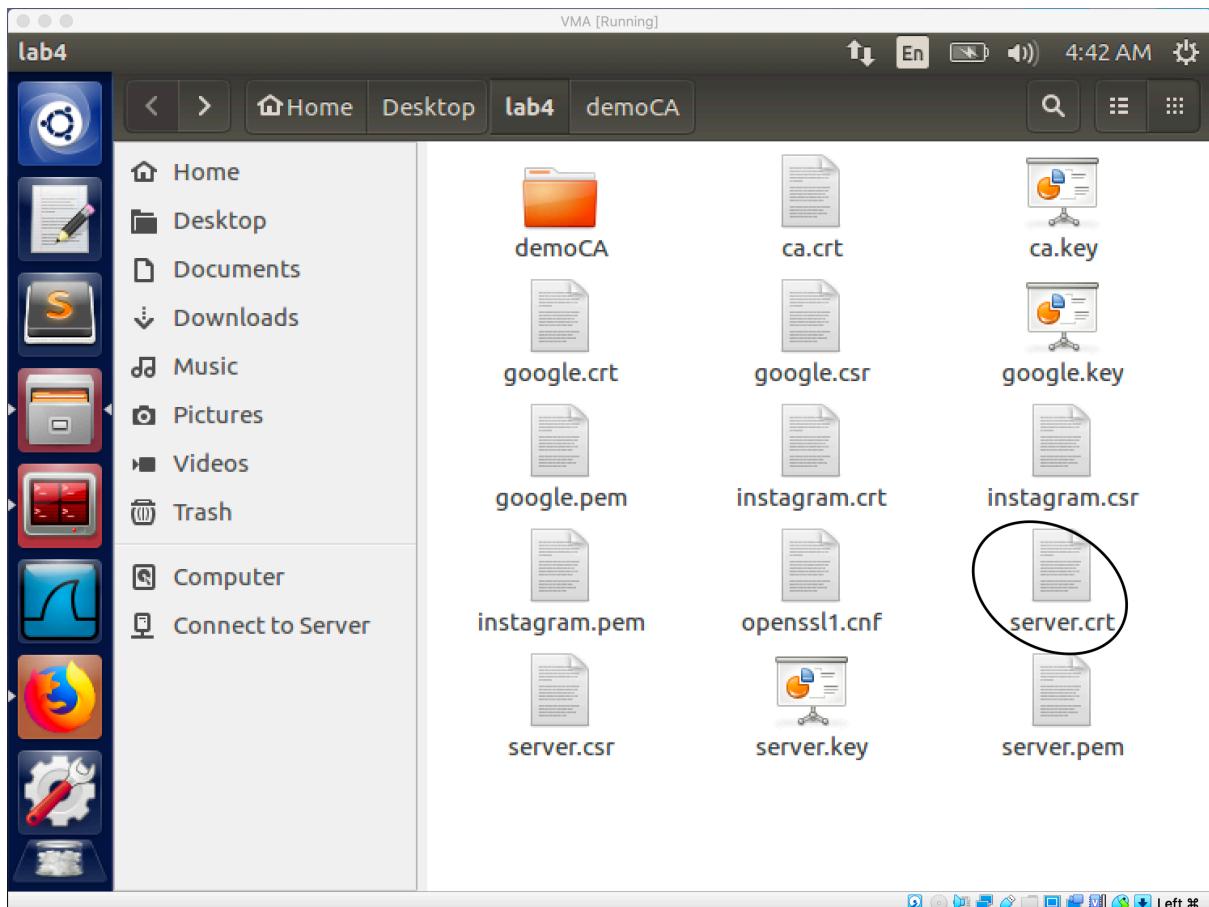
Step2:

I use command ‘openssl req -new -key server.key -out server.csr -config openssl1.cnf’ to generate a request to sign file for ‘SEEDPKILab2020.com’. A ‘server.csr’ file will be generated.



Step3:

I use command ' openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl1.cnf' to send the csr file to the CA and ask for signing the certificate. A file called server.crt will be generated which is the signed certificate of our server.



### Task3: Deploying Certificate in an HTTPS web server

#### Step1:

We choose SEEDPKILab202.com as the name of our website. To get the VM recognize this name, I add the ' 127.0.0.1 SEEDPKILab2020.com' entry to my '/etc/hosts' file.

VMA [Running] 4:45 AM

/bin/bash /bin/bash 61x22

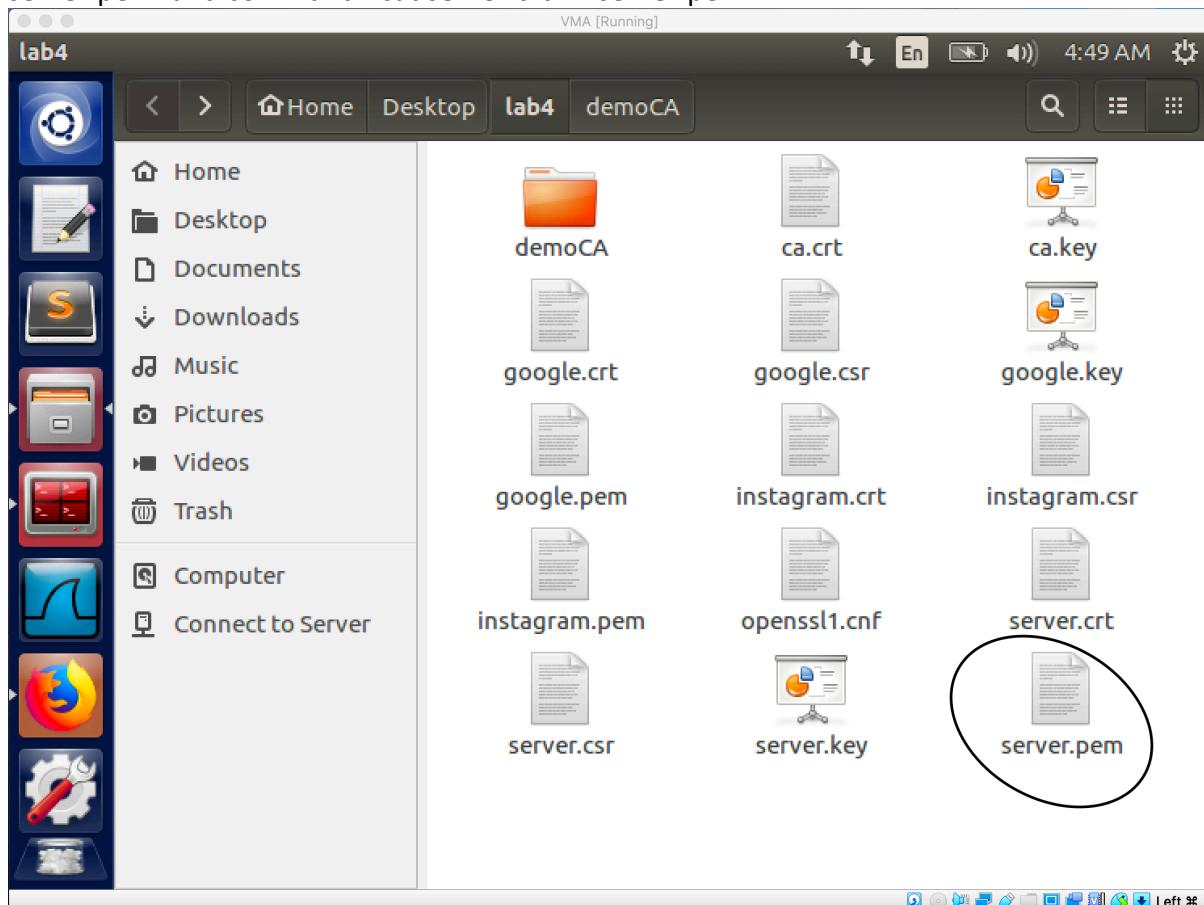
GNU nano 2.5.3 File: /etc/hosts

```
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      SEEDPKILab2020.com
127.0.0.1      instagram.com
```

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

Step2: Configuring the web server

I combine the secret key and certificate into one file by using command 'cp server.key server.pem' and command ' cat server.crt >> server.pem'.

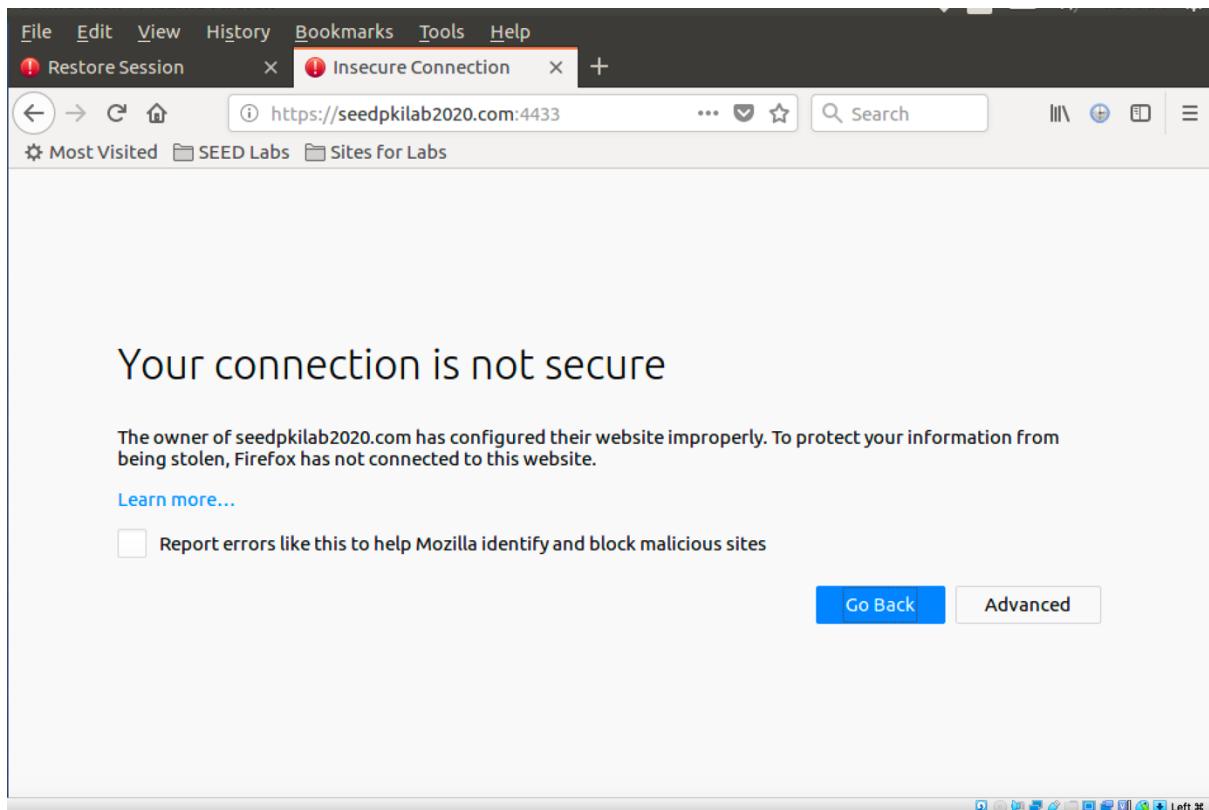


After that I use the command ' openssl s\_server -cert server.pem -www' to start a simple web server.

```
[02/27/21]seed@VM:~/.../lab4$ ls
ca.crt ca.key demoCA openssl1.cnf server.crt server.csr server.key
[02/27/21]seed@VM:~/.../lab4$ sudo nano /etc/hosts/
[02/27/21]seed@VM:~/.../lab4$ sudo nano /etc/hosts
[02/27/21]seed@VM:~/.../lab4$ cp server.key server.pem
[02/27/21]seed@VM:~/.../lab4$
[02/27/21]seed@VM:~/.../lab4$

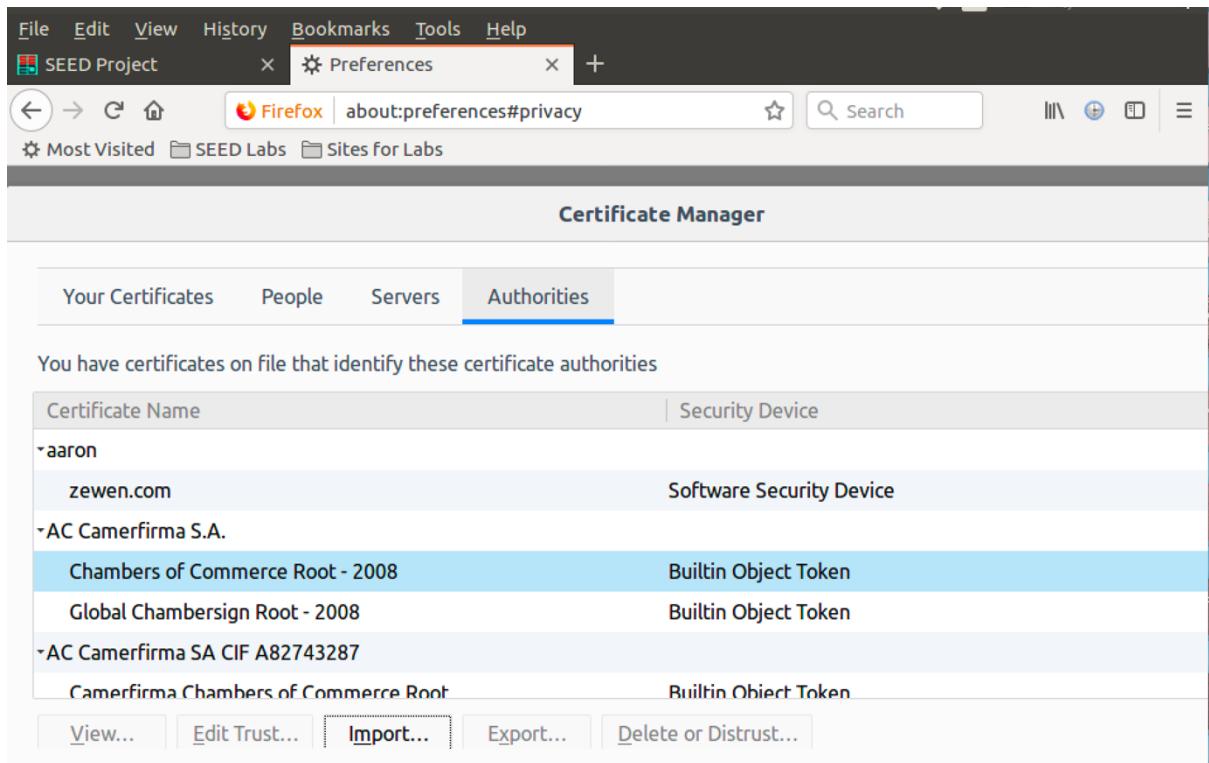
[02/27/21]seed@VM:~/.../lab4$ 
[02/27/21]seed@VM:~/.../lab4$ 
[02/27/21]seed@VM:~/.../lab4$ 
[02/27/21]seed@VM:~/.../lab4$ cat server.crt >> server.pem
[02/27/21]seed@VM:~/.../lab4$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
Using default temp DH parameters
ACCEPT
ACCEPT
ACCEPT
```

By searching the URL 'SEEDPKILab2020.com' on Firefox I was blocked because the certificate is self-signed.



### Step3: Getting the browser to accept our CA certificate

Since our self-signed is not include into our Firefox server, I manually add the ca.crt to the Firefox.



### Step4: Testing our HTTPS website

After adding the ca.crt to the Firefox server, I am able to visit the website.

Mozilla Firefox

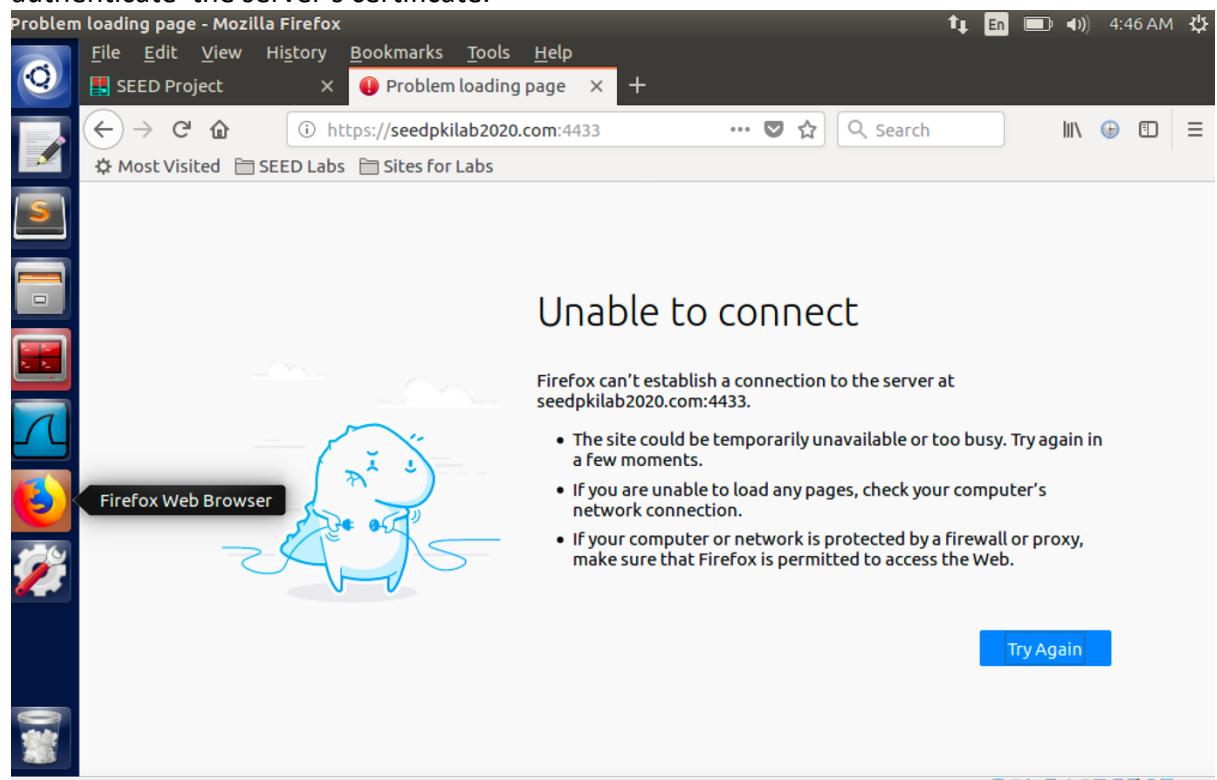
File Edit View History Bookmarks Tools Help

SEED Project seedpkilab2020.com:4433

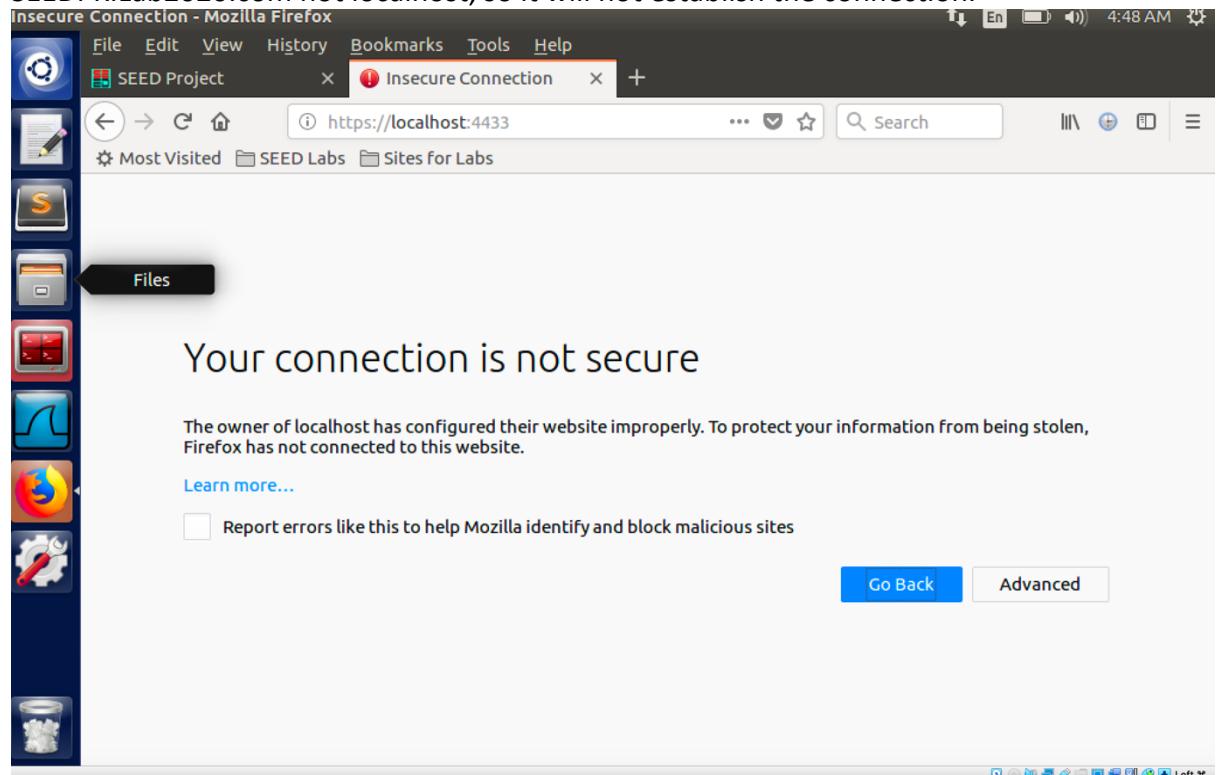
Most Visited SEED Labs Sites for Labs

```
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-SHA384TLSv1/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:DHE-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-DSS-AES256-GCM-SHA384TLSv1/SSLv3:DHE-RSA-AES256-SHA256
TLSv1/SSLv3:DHE-DSS-AES256-SHA256TLSv1/SSLv3:DHE-RSA-AES256-SHA256
TLSv1/SSLv3:DH-DSS-AES256-SHA256TLSv1/SSLv3:DHE-RSA-AES256-SHA
TLSv1/SSLv3:DH-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:DHE-RSA-CAMELLIA256-SHA
TLSv1/SSLv3:DHE-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-SHA TLSv1/SSLv3:AES256-GCM-SHA384
TLSv1/SSLv3:AES256-SHA256TLSv1/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-SHA256TLSv1/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:DHE-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:DHE-RSA-AES128-GCM-SHA256TLSv1/SSLv3:DHE-RSA-AES128-SHA256
TLSv1/SSLv3:DHE-DSS-AES128-SHA256TLSv1/SSLv3:DHE-RSA-AES128-SHA
TLSv1/SSLv3:DHE-DSS-AES128-SHA256TLSv1/SSLv3:DHE-RSA-AES128-SHA
TLSv1/SSLv3:DHE-DSS-AES128-SHA TLSv1/SSLv3:DHE-RSA-SEED-SHA
TLSv1/SSLv3:DHE-DSS-SEED-SHA TLSv1/SSLv3:DHE-RSA-SEED-SHA
```

- When I modify one byte of the server.pem file, I am not able to connect to the website. It is because the server.pem is corrupted so the web browser cannot authenticate the server's certificate.



- If I change the URL to 'localhost:4433' I am not able to connect to the website, since Firefox will check the CN in the certificate, it will find out the CN is SEEDPKILab2020.com not localhost, so it will not establish the connection.



Task4: Deploying Certificate in an Apache-Based HTTPS Website

In order to add a HTTPS website to the server, I add a HTTPS web server to the '/etc/apache2/sites-available/default-ssl.conf' directory.

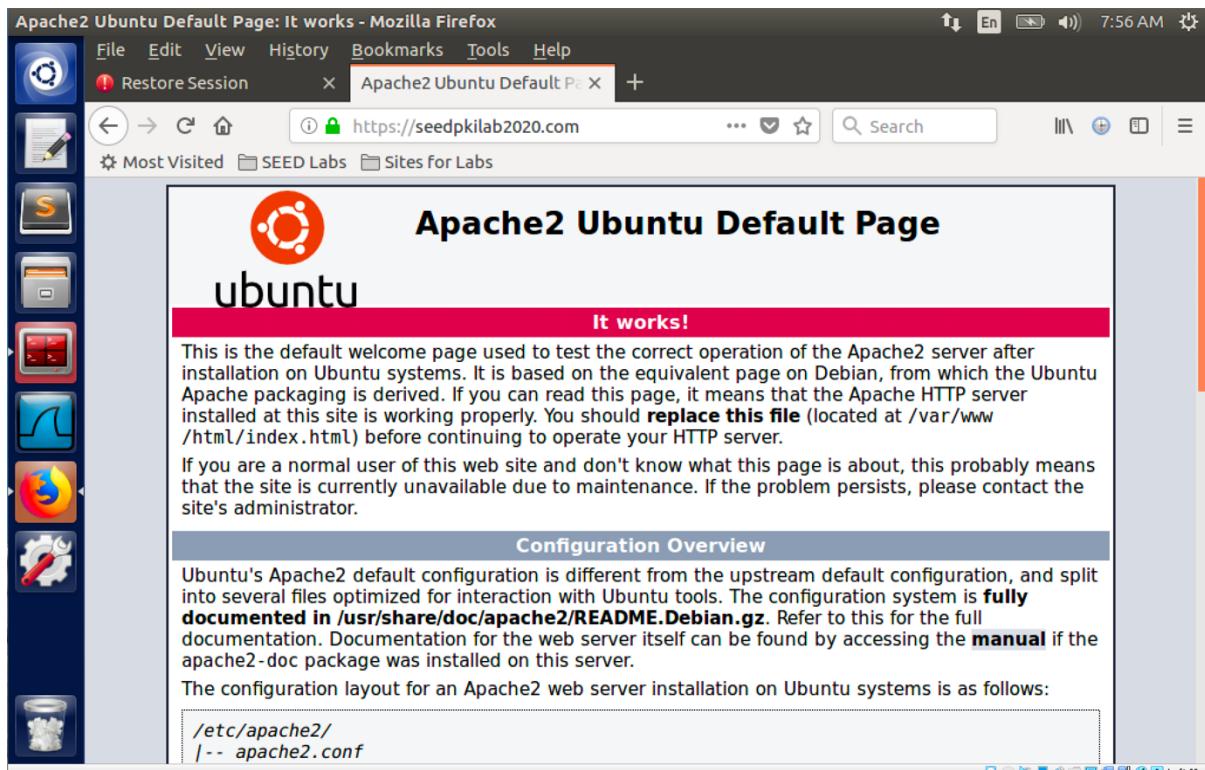
The screenshot shows a terminal window titled 'VMA [Running]' with the command '/bin/bash'. The title bar also shows the file path 'File: ...available/default-ssl.conf'. The terminal content is the Apache configuration file:

```
# Similarly, one has to force some clients$  
# their broken HTTP/1.1 implementation. Us$  
# "force-response-1.0" for this.  
# BrowserMatch "MSIE [2-6]" \  
# nokeepalive ssl-unclean-shut$  
# downgrade-1.0 force-response$  
  
</VirtualHost>  
<VirtualHost *:443>  
    ServerName SEEDPKILab2020.com  
    DocumentRoot /var/www/html  
    DirectoryIndex index.html  
  
    SSLEngine On  
    SSLCertificateFile      /home/seed/Desktop/$  
    SSLCertificateKeyFile   /home/seed/Desktop/$  
</VirtualHost>  
  
^G Get Help ^O Write Out^W Where Is ^K Cut Text ^J Justify  
^X Exit      ^R Read File^V Replace  ^U Uncut Tex^T To Spell
```

After add the server to default-ssl.conf file, I run server command to restart the service.

```
// Test the Apache configuration file for errors  
$ sudo apachectl configtest  
  
// Enable the SSL module  
$ sudo a2enmod ssl  
  
// Enable the site we have just edited  
$ sudo a2ensite default-ssl  
  
// Restart Apache  
$ sudo service apache2 restart
```

Then I am able to visit the SEEDPKILab2020.com under https request.



### Task5: Man-In-The-Middle Attack

Step1: Setting up the malicious website.

Followed the task 4 procedure, I add another HTTPS web server to the default-ssl.conf file for Instagram.com.

A screenshot of a terminal window titled "/bin/bash" running in a Virtual Machine (VMA [Running]). The window shows the command "GNU nano 2.5.3 File: ...ailable/default-ssl.conf". The content of the file is:

```
ServerName SEEDPKILab2020.com
DocumentRoot /var/www/html
DirectoryIndex index.html

SSLEngine On
SSLCertificateFile      /home/seed/Desktop/$
SSLCertificateKeyFile   /home/seed/Desktop/$
</VirtualHost>
<VirtualHost *:443>
    ServerName Instagram.com
    DocumentRoot /var/www/html
    DirectoryIndex index.html

    SSLEngine On
    SSLCertificateFile      /home/seed/Desktop/$
    SSLCertificateKeyFile   /home/seed/Desktop/$
</VirtualHost>
```

The terminal also shows a set of keyboard shortcuts at the bottom.

Step2:

After that I add the entry '10.0.2.7 instagram.com' to another vm. The 10.0.2.7 is the ip address of the man-in-the-middle.

The screenshot shows a Linux desktop environment with a terminal window open in the foreground. The terminal window title is "GNU nano 2.5.3" and the file path is "File: /etc/hosts". The content of the file is as follows:

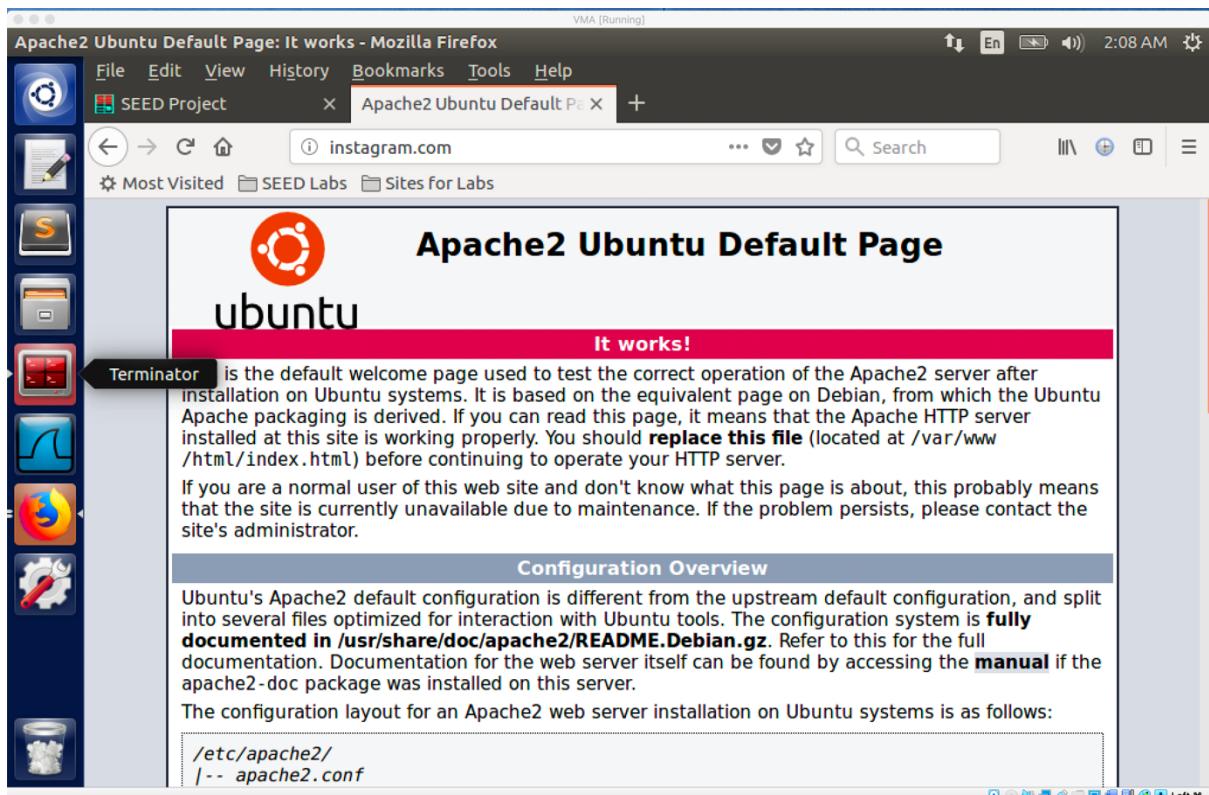
```
127.0.0.1      User
127.0.0.1      Attacker
127.0.0.1      Server
127.0.0.1      www.SeedLabSQLInjection.com
127.0.0.1      www.xsslabelgg.com
127.0.0.1      www.csrflabelgg.com
127.0.0.1      www.csrflabattacker.com
127.0.0.1      www.repackagingattacklab.com
127.0.0.1      www.seedlabclickjacking.com
10.0.2.7        google.com
10.0.2.7        instagram.com
```

The terminal window also displays a set of keyboard shortcuts at the bottom:

- ^G Get Help
- ^O Write Out
- ^W Where Is
- ^K Cut Text
- ^J Justify
- ^R Read File
- ^V Replace
- ^U Uncut Text
- ^T To Spell

Step3: Browse the target website

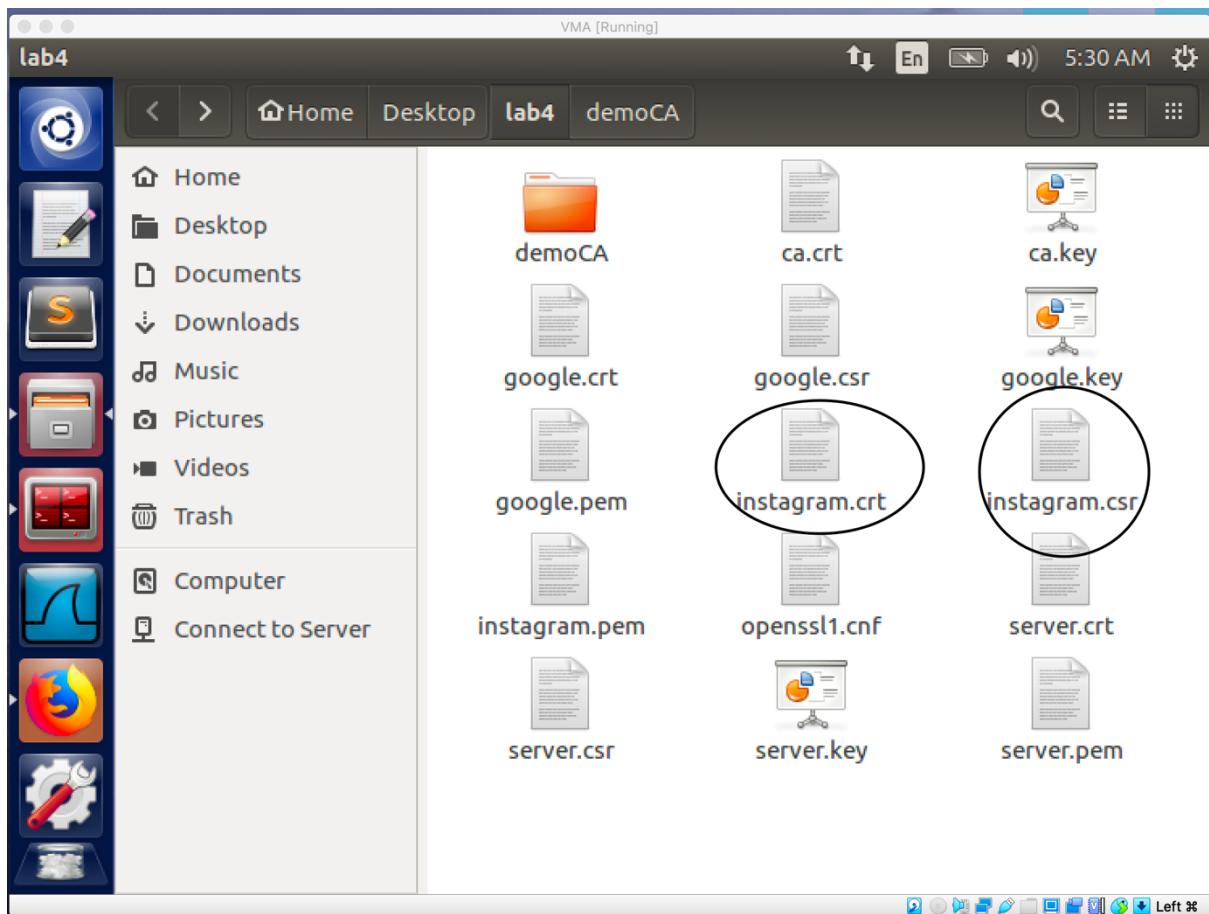
By searching the URL Instagram.com on the victim machine, I will go to the malicious website.



## Task6: Launching a Man-In-The-Middle Attack with a Compromised CA

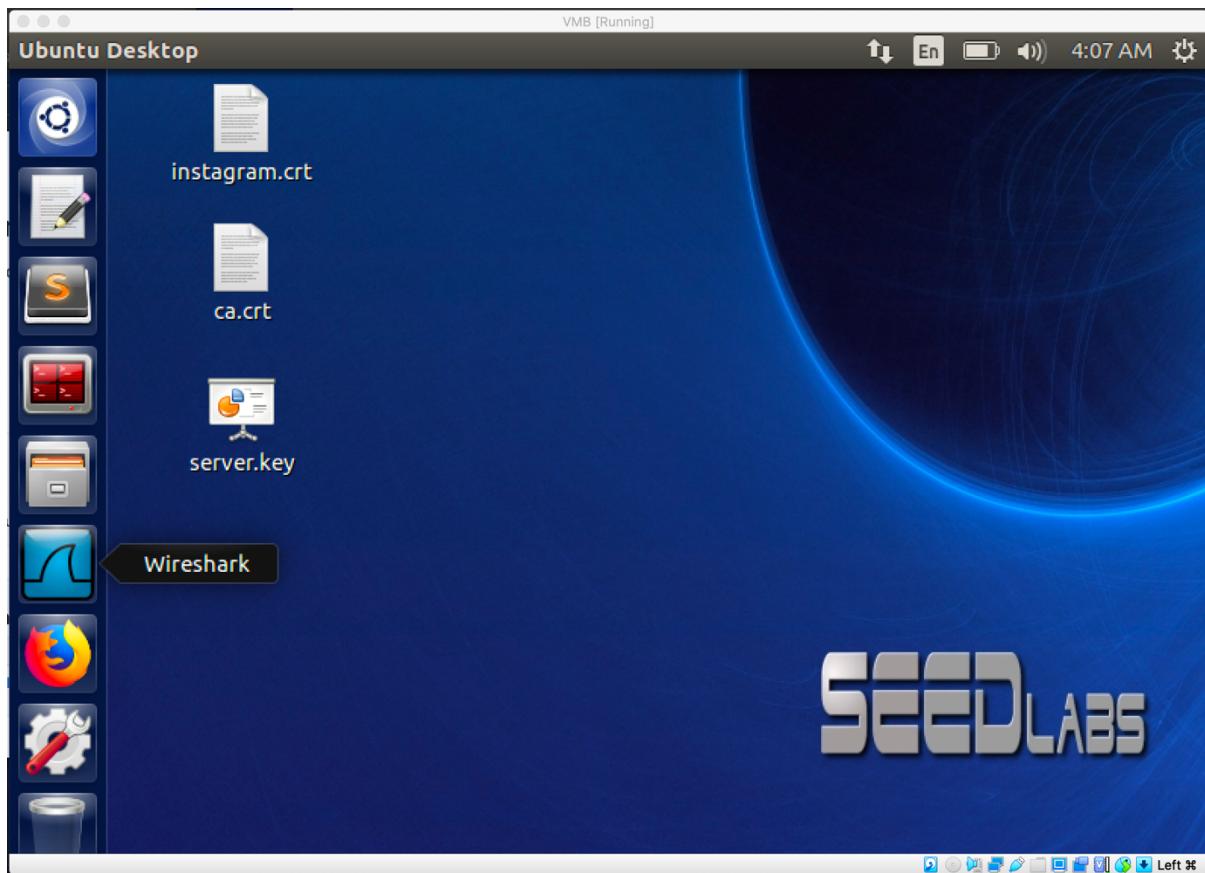
Step1:

I used my CA to sign a certificate for Instagram.com by using command 'openssl req -new -key server.key -out instagram.csr -config openssl1.cnf' and command 'openssl ca -in instagram.csr -out instagram.crt -cert ca.crt -keyfile ca.key -config openssl1.cnf'.



Step2:

I send the ca.crt file to the victim machine by using command 'scp <seed@10.0.2.7:/home/seed/Desktop/lab4/ca.crt> /home/seed/Desktop'



Step3:

I add the ca.crt file to the trusted ca list of Firefox.

The screenshot shows the Firefox certificate manager interface. It displays a list of certificate authorities under the heading 'YOU HAVE CERTIFICATES ON FILE THAT IDENTIFY THESE CERTIFICATE AUTHORITIES'. The list includes:

Certificate Name	Security Device
aaron	Software Security Device
zewen.com	Software Security Device
AC Camerfirma S.A.	Builtin Object Token
Chambers of Commerce Root - 2008	Builtin Object Token
Global Chambersign Root - 2008	Builtin Object Token

Step4:

After configuration, I am able to visit the target website under https request at the victim machine. Which means, if the DNS attack is successful I am able to set up the man-in-the-middle attack.

