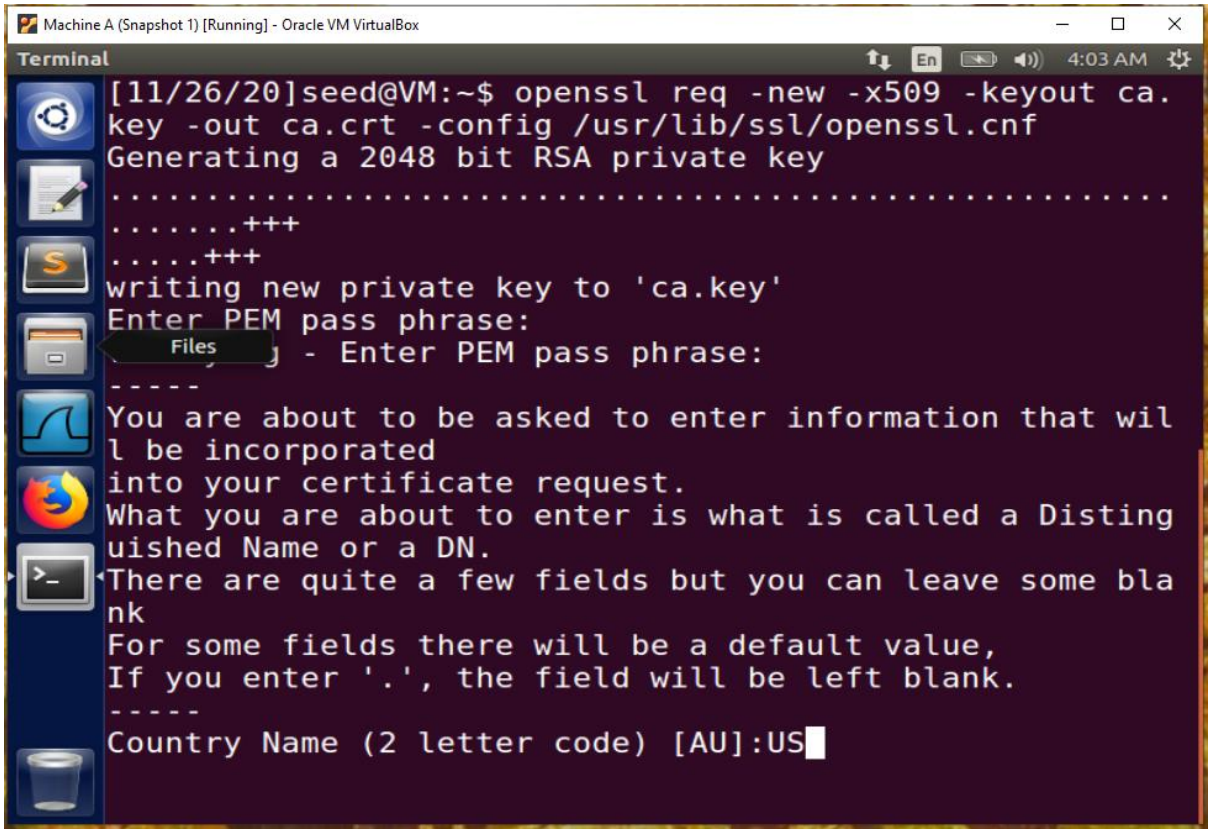
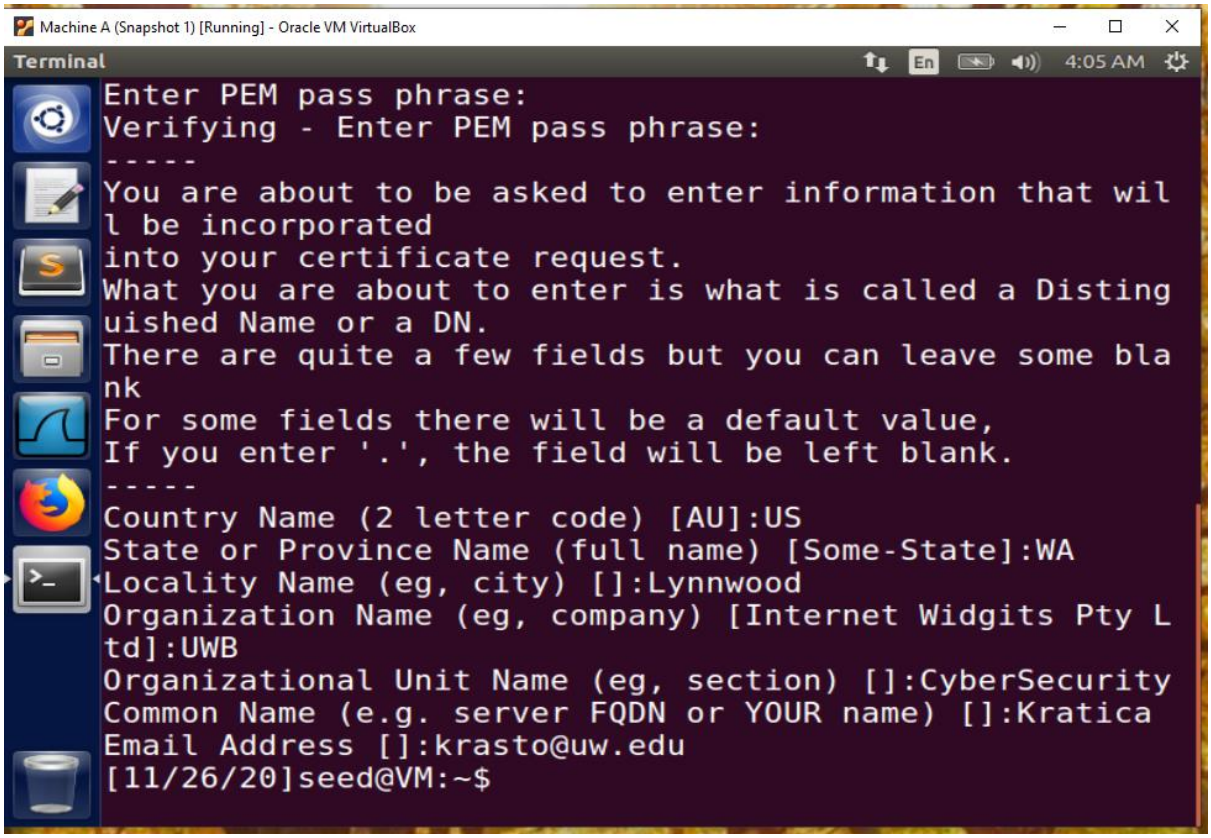
# Public-Key Infrastructure (PKI) Lab

## Task 1: Becoming a Certificate Authority (CA)

- To generate a self-signed certificate for CA. This CA is totally trusted, and its certificate will serve as the root certificate. After running the given command, the self-signed certificate for the CA can be generated.
  *openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf*






## Task 2: Creating a Certificate for SEEDPKILab2018.com

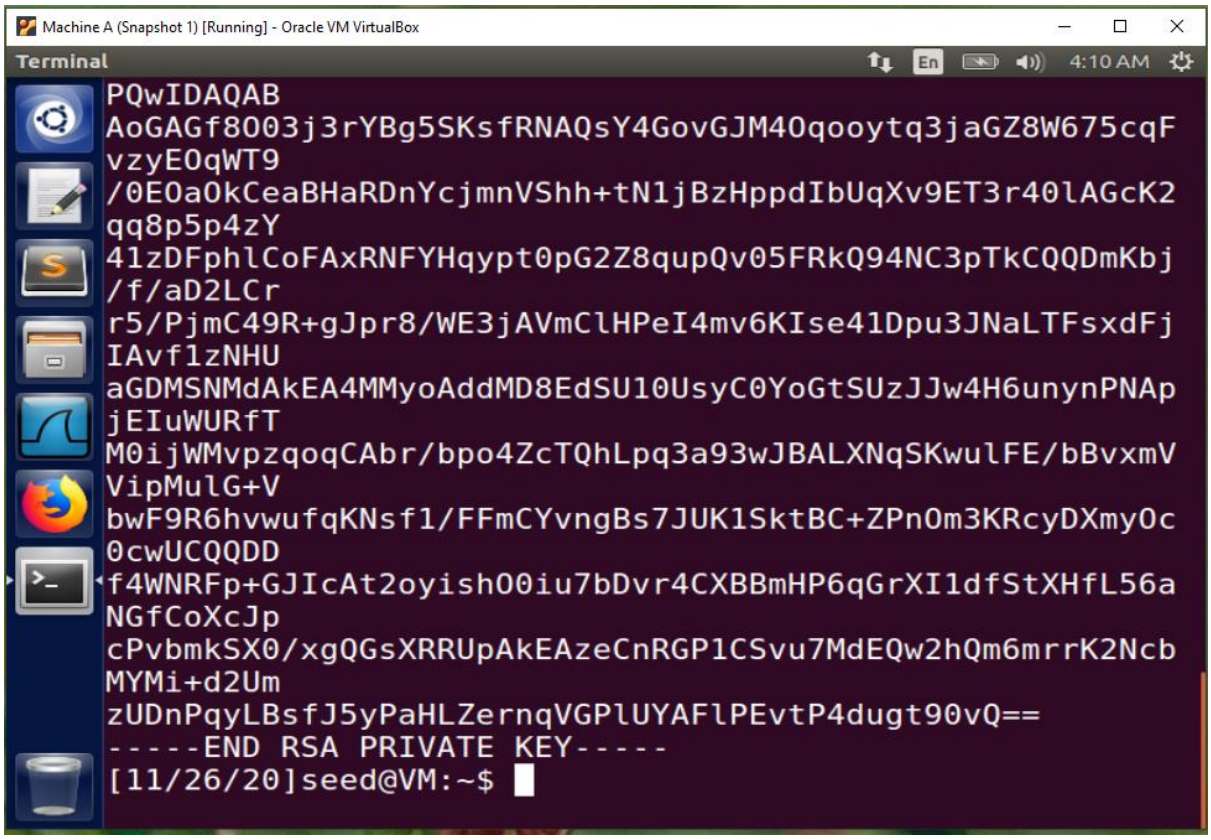
### Step 1: Generate public/private key pair

- The company is required to create its own public/private key pair. The command given below will generate an RSA key pair (both private and public keys). The private key password will be encrypted using AES-128 encryption algorithm.
  *openssl genrsa -aes128 -out server.key 1024*
- The keys will be stored in the file server.key

```
[11/26/20]seed@VM:~$ ls
android  ca.key         Documents  Music      source
bin      Customization  Downloads  Pictures   Templates
ca.crt   Desktop        lib        Public     Videos
[11/26/20]seed@VM:~$ 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:
3070600896:error:28069065:lib(40):UI_set_result:result
too small:ui_lib.c:823:You must type in 4 to 1023 chara
cters
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
[11/26/20]seed@VM:~$ ls
android  Customization  lib        server.key
bin      Desktop        Music      source
ca.crt   Documents      Pictures   Templates
ca.key   Downloads      Public     Videos
[11/26/20]seed@VM:~$
```

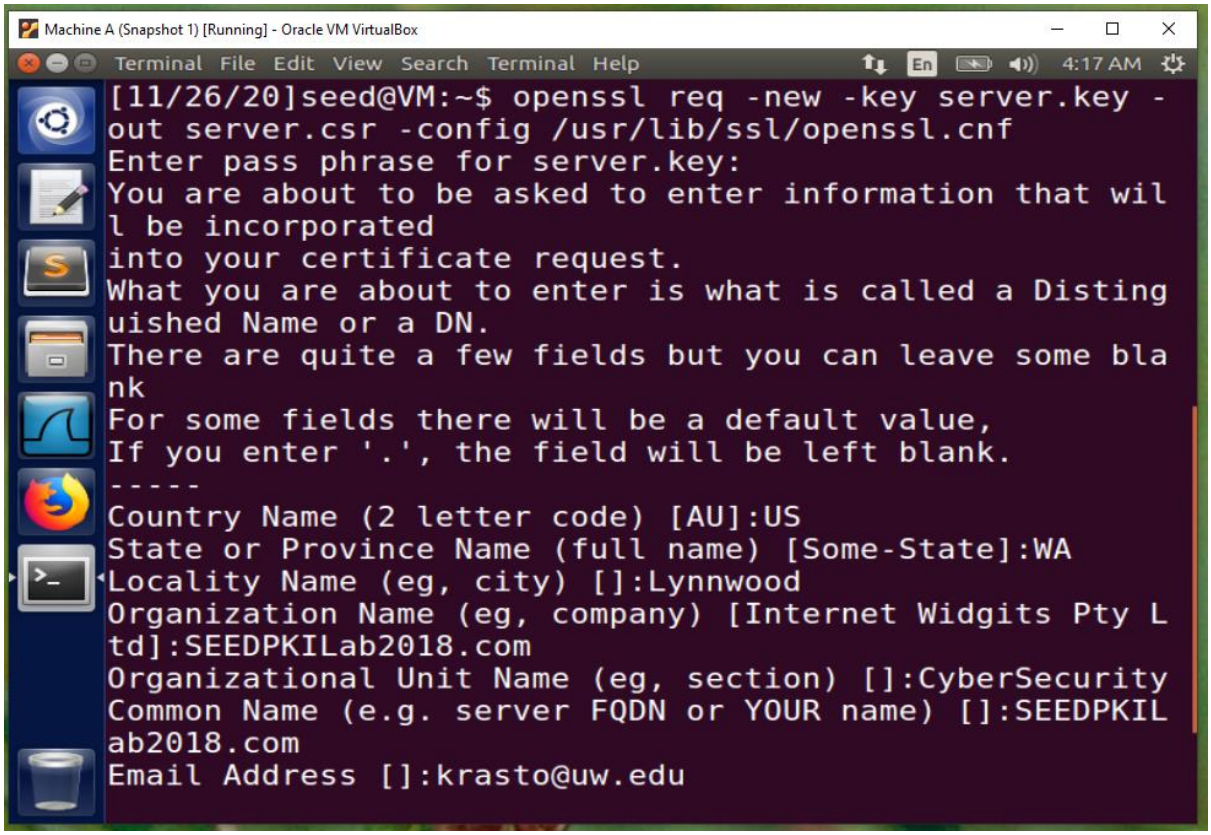
- The key stored generated file server.key is an encrypted text file. Therefore, the content in the file are modulus and exponents. The following command will show the modulus and private exponent in file:
  *openssl rsa -in server.key -text*

```
[11/26/20]seed@VM:~$ openssl rsa -in server.key -text
Enter pass phrase for server.key:
Private-Key: (1024 bit)
modulus:
    00:ca:14:01:2b:81:6a:9f:69:d4:bf:28:d4:0e:33:
    d7:7e:b9:5e:8b:f9:95:70:81:55:cb:fb:6c:36:cb:
    90:54:59:2c:10:41:41:aa:66:66:b5:6b:f0:29:e4:
    59:a4:74:29:b8:24:0d:cd:fb:99:2a:ca:e8:1f:d4:
    87:db:2c:fe:d6:e5:db:ee:74:a9:f1:94:cf:3b:03:
    42:55:1d:c6:fb:97:93:39:2e:6b:0a:04:82:5a:57:
    9b:82:b1:9f:de:05:12:e1:0b:ec:3d:71:d4:64:95:
    ee:5c:6d:39:18:85:ed:e9:fd:a2:c8:db:1b:19:d1:
    f8:ff:c3:61:1e:65:aa:4f:43
publicExponent: 65537 (0x10001)
privateExponent:
    19:ff:0e:d3:78:f7:ad:80:60:e5:22:ac:7d:13:40:
    42:c6:38:1a:8b:c6:24:ce:0e:aa:8a:32:b6:ad:e3:
    68:66:7c:5b:ae:f9:72:a1:6f:cf:21:0e:a9:64:fd:
    ff:41:0e:68:e9:02:79:a0:47:69:10:e7:61:c8:e6:
    9d:54:a1:87:eb:4d:d6:30:73:1e:9a:5d:21:b5:2a:
    5e:ff:44:4f:7a:f8:d2:50:06:70:ad:aa:ab:ca:79:
    a7:8c:d8:e3:5c:c3:16:98:65:0a:81:40:c5:13:45:
```



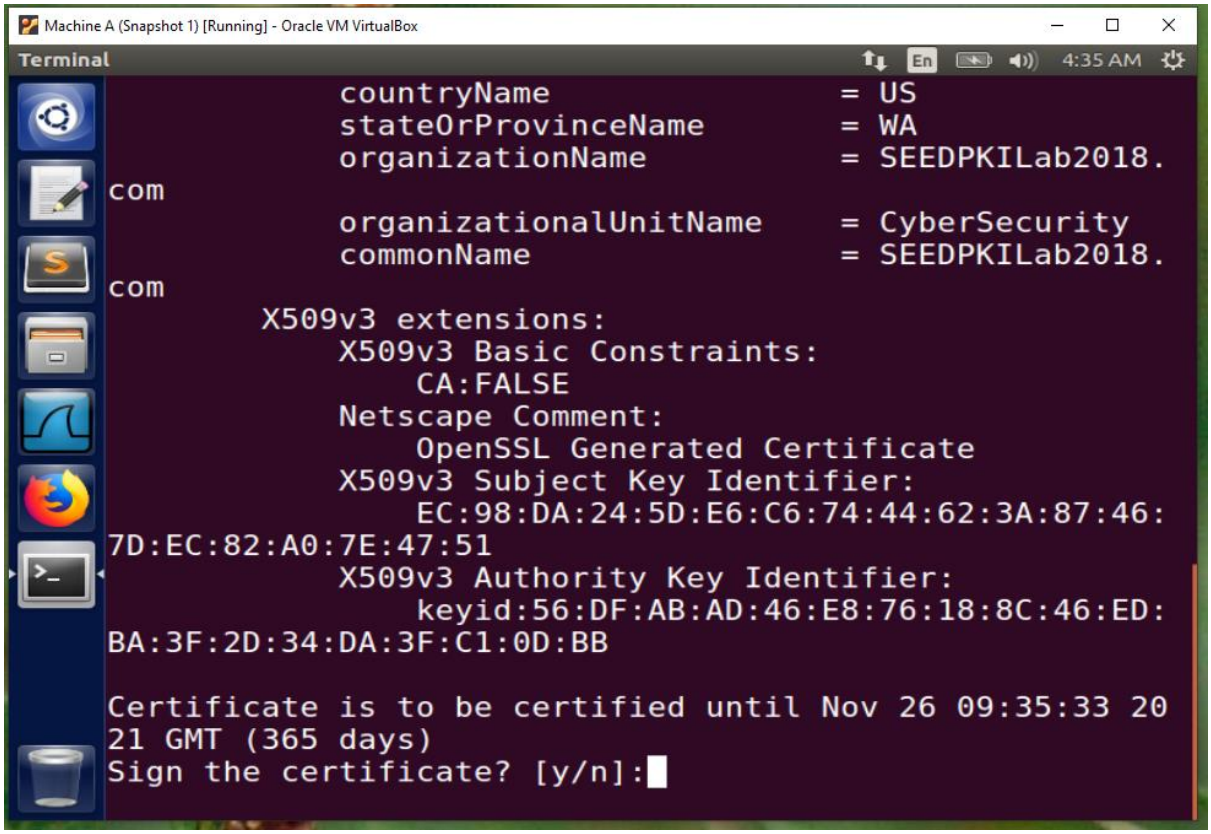

**Step 2: Generate a Certificate Signing Request (CSR)**

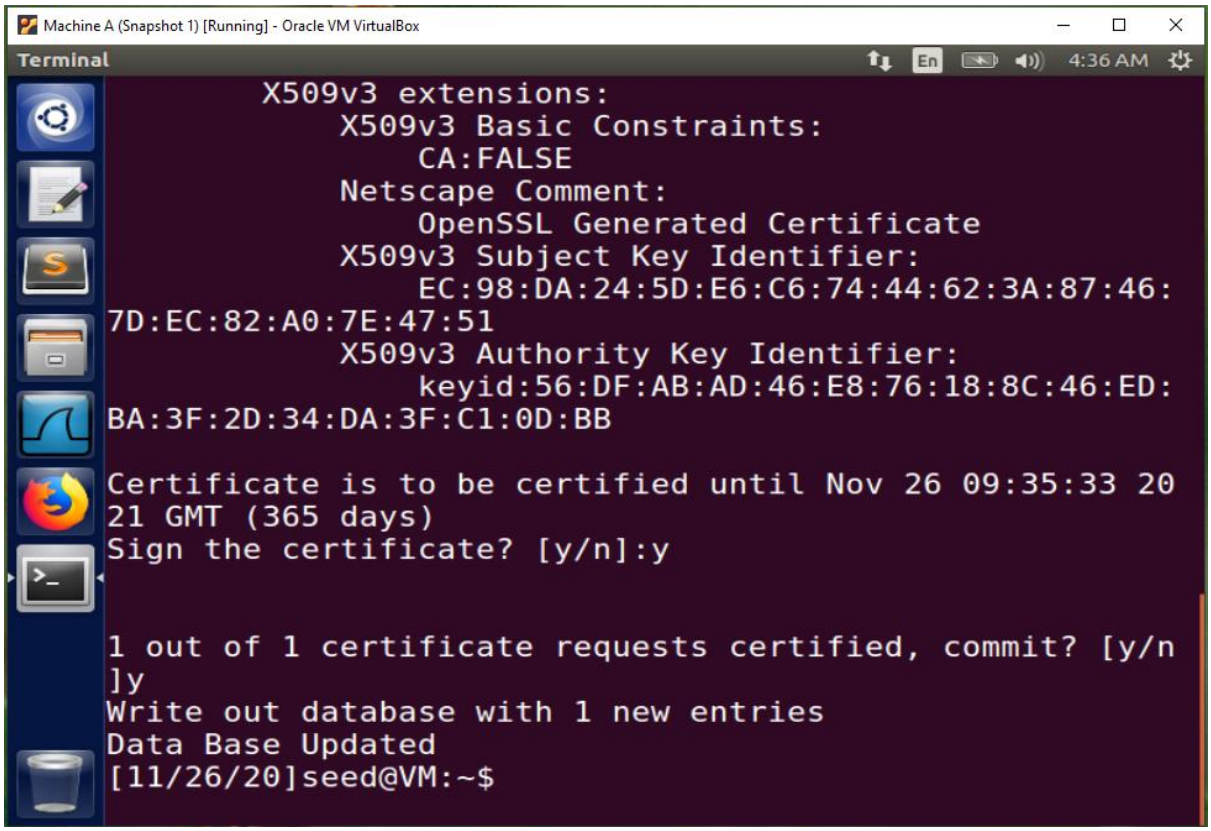
- Now the company has the server.key file. By using given command, the server.key file will generate the Certificate Signing Request (CSR) which contains SEEDPKILab2018.com public key.
  *openssl req -new -key server.key -out server.csr -config openssl.cnf*




**Step 3: Generating Certificates**

- Now to generate a certificate, the CSR file needs to have the CA's signature. Here I am using own trusted CA to generate certificates. The command given below will turns the certificate signing request (server.csr) into an X509 certificate (server.crt), using the CA's ca.crt and ca.key.
  *openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf*
- I observed that the certificate has been generated successfully. Refer below snapshot:

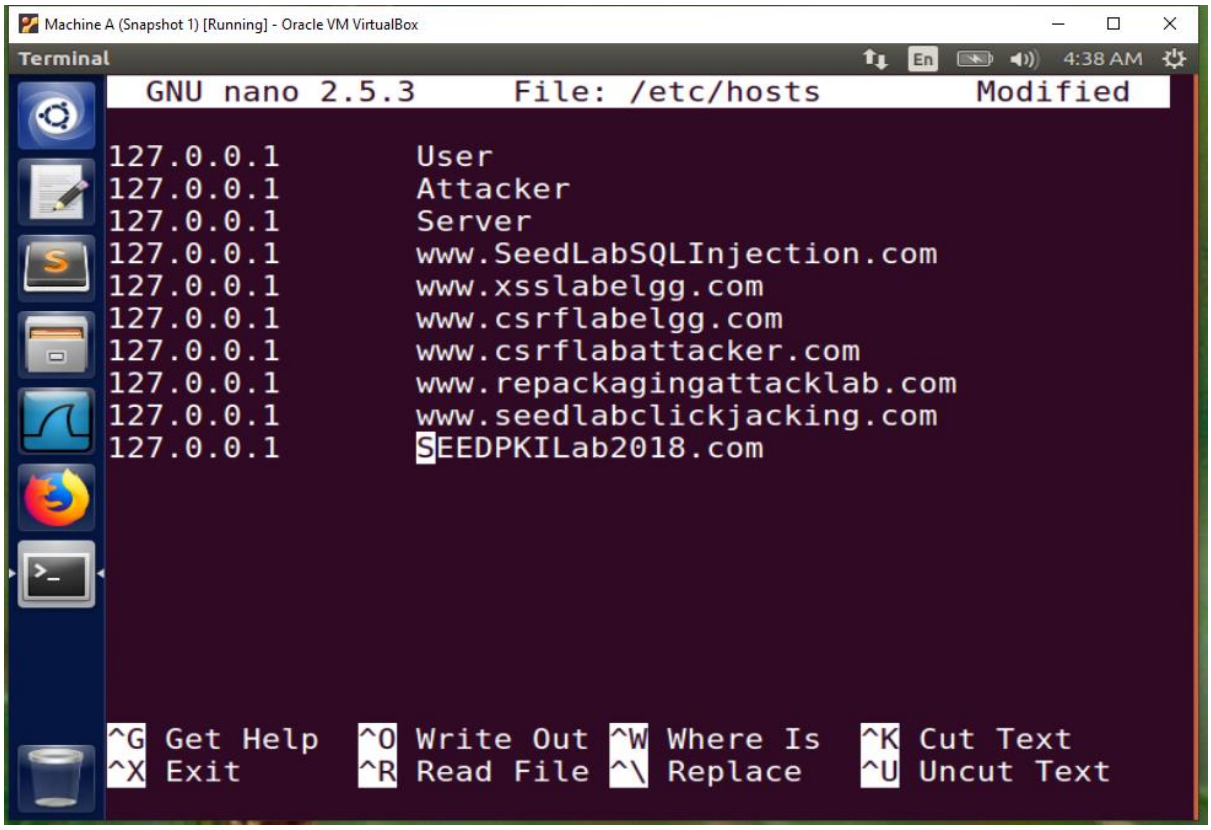



**Task 3: Deploying Certificate in an HTTPSWeb Server**

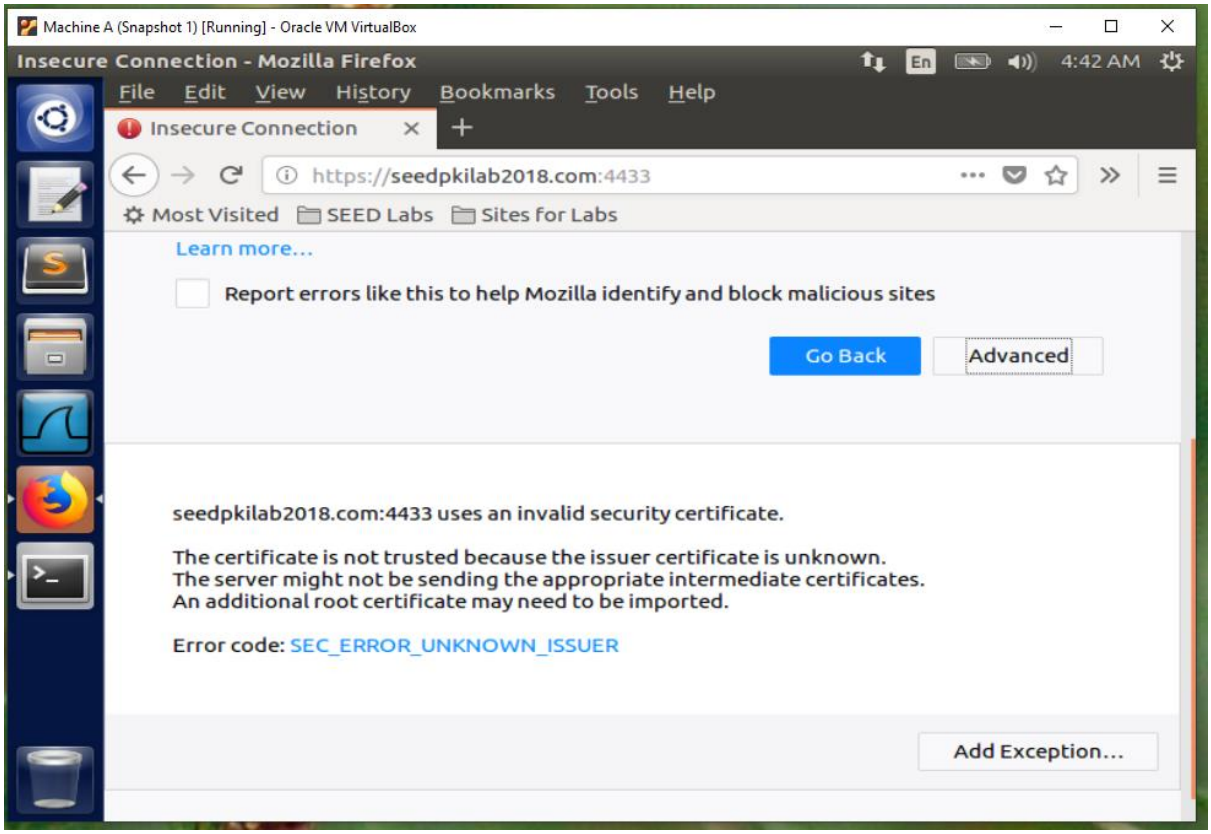
**Step 1: Configuring DNS**

Since SEEDPKILab2018.com website is being used in the experiment. Now configuring the entry of this website into /etc/hosts to localhost (127.0.01) as shown in below snapshot:



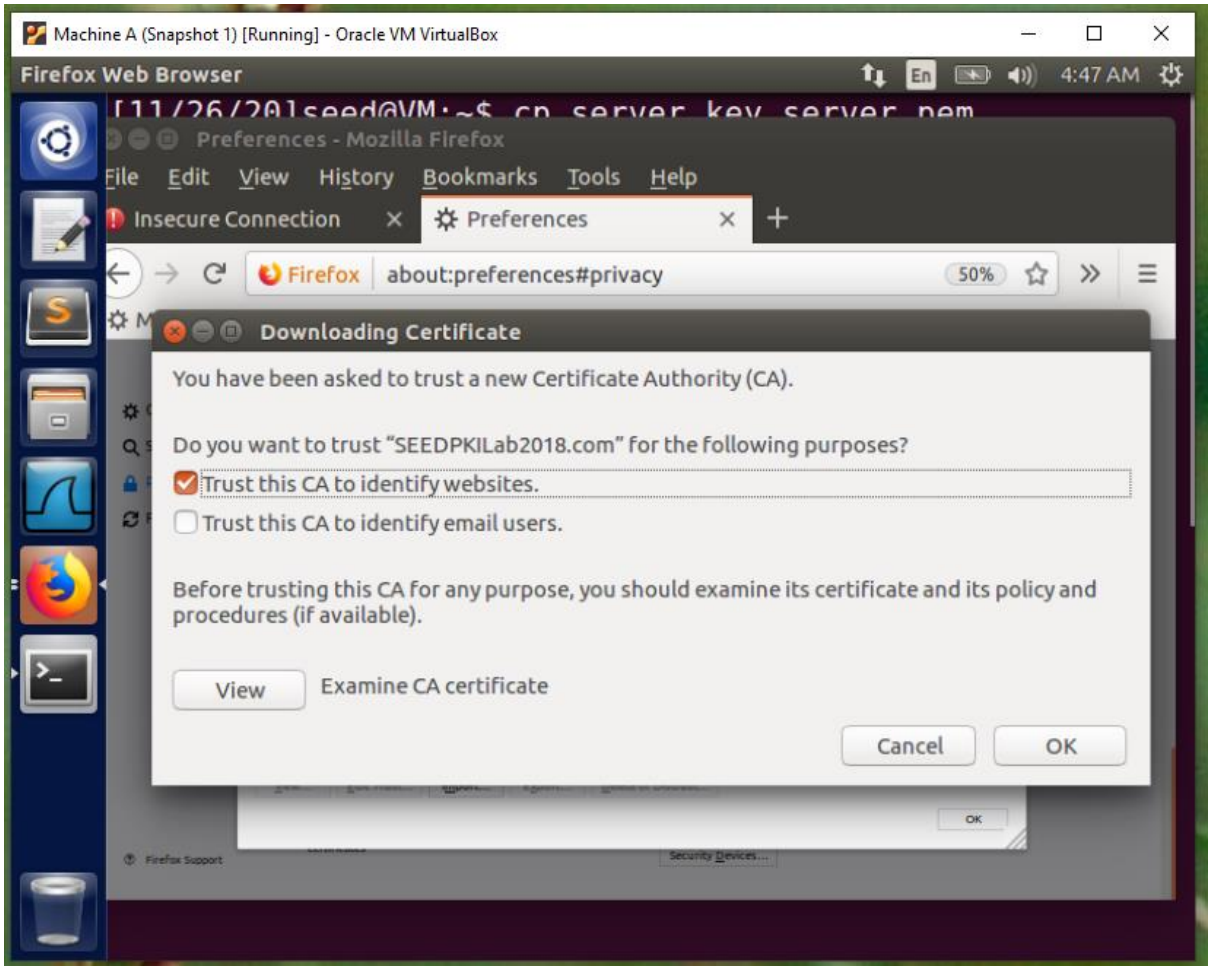

**Step 2: Configuring the web server**

- Firstly, combining the secret key and the certificate into one file by using given command:
  *cp server.key server.pem*
  *cat server.crt >> server.pem*
- By executing given OpenSSL command, the simple web server with generated certificate will launch.
  *openssl s_server -cert server.pem -www*
- When I hit https://seedpkilab2018.com:4433/ on Firefox browser, I observed error message. *"seedpkilab2018.com:4433 uses an invalid security certificate. The certificate is not trusted because the issuer certificate is unknown".*
  Refer below snapshot:



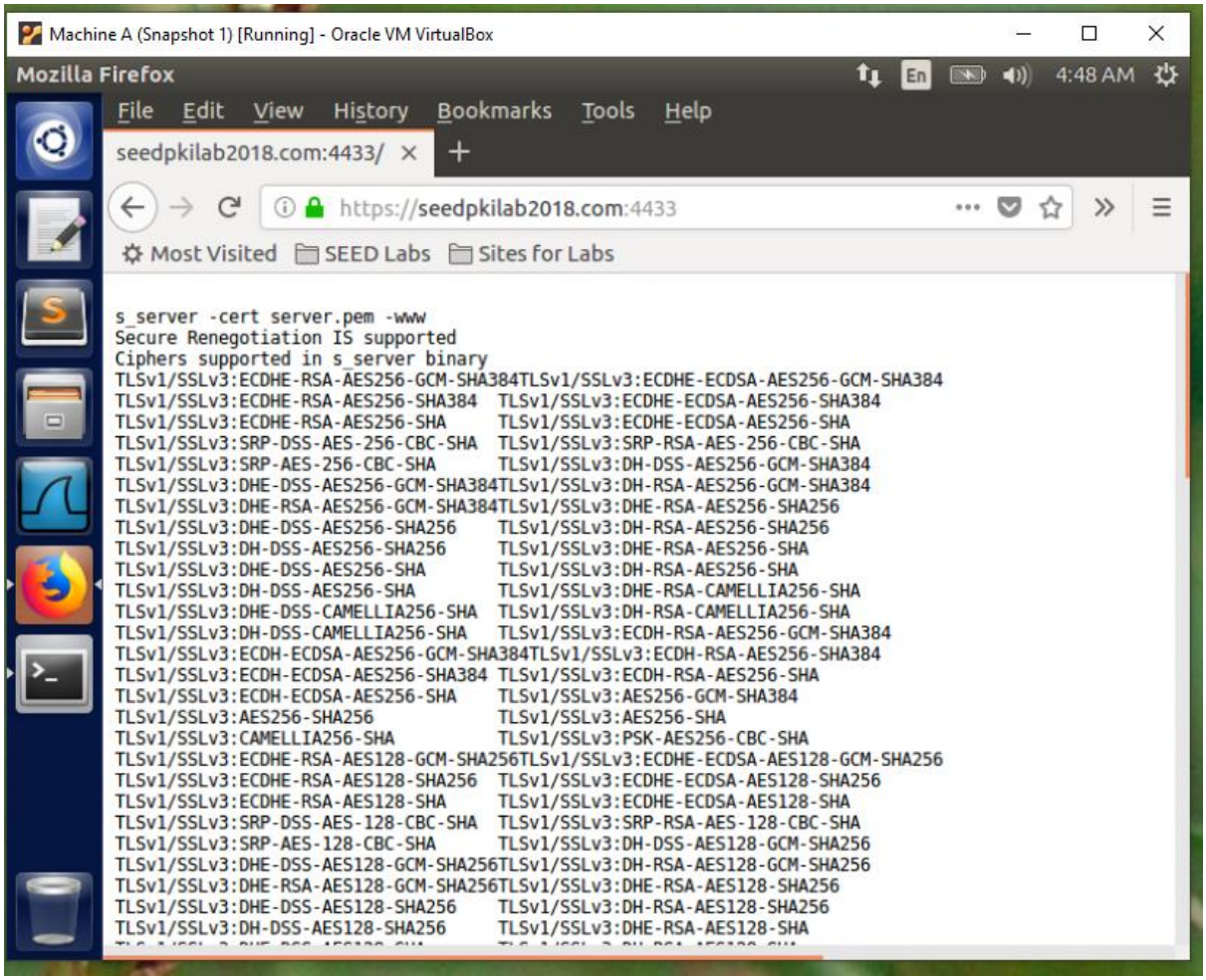

**Step 3: Getting the browser to accept our CA certificate**

- SEEDPKILab2018.com certificate is signed by CA (i.e., ca.crt), and Firefox does not accept this CA. So, we will manually add the ca.crt to the Firefox browser.
- Enabling “Trust this CA to identify websites” option while importing the certificate.
- Refer below snapshot showing ca.crt added to Firefox browser.



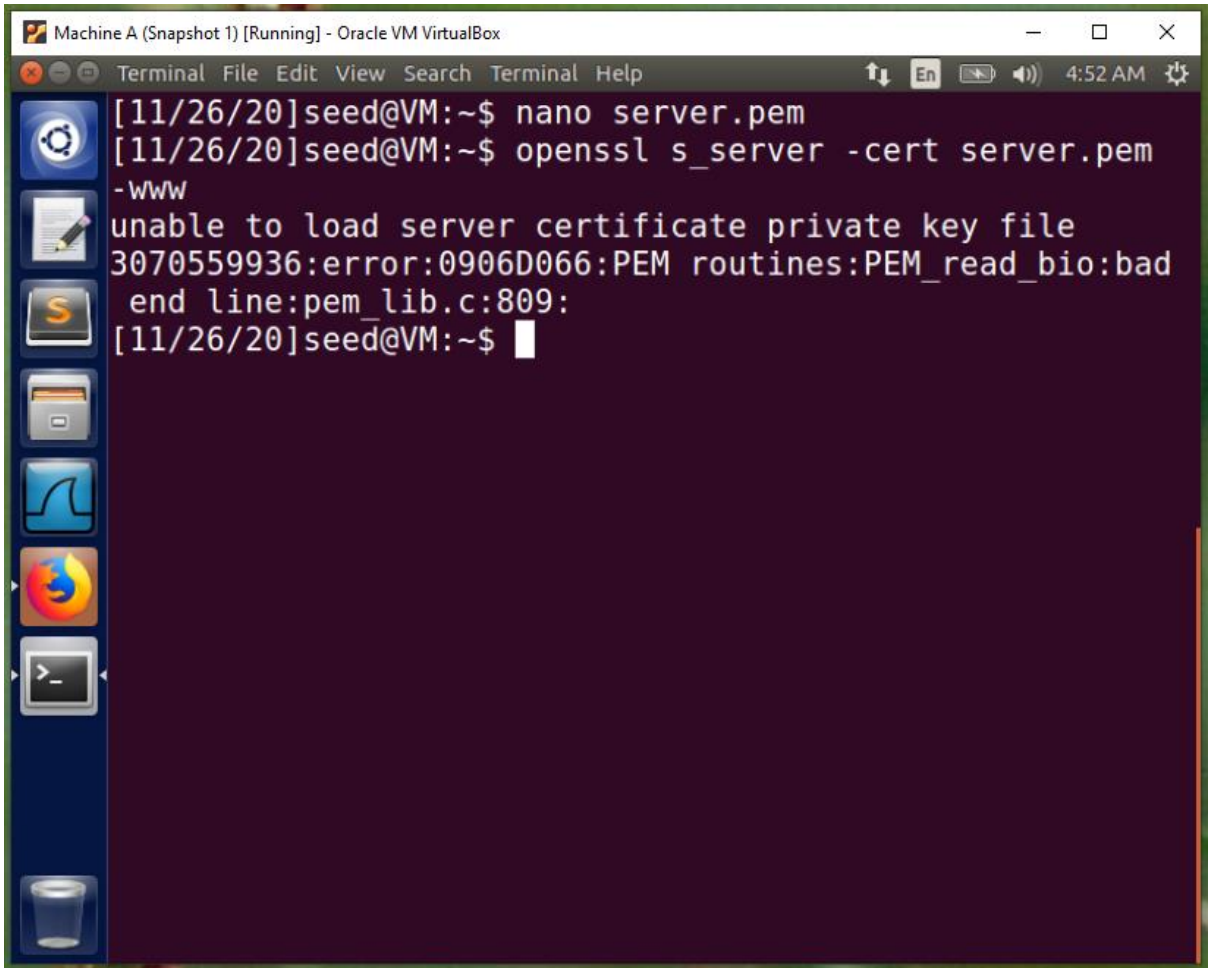

**Step 4: Testing our HTTPS website**

- After importing the CA certificate in Firefox browser. It will how the CA certificate in the certificate accepted list. When I browsed https://SEEDPKILab2018.com:4433 website on Firefox browser, it didn't give error. Refer given snapshot:

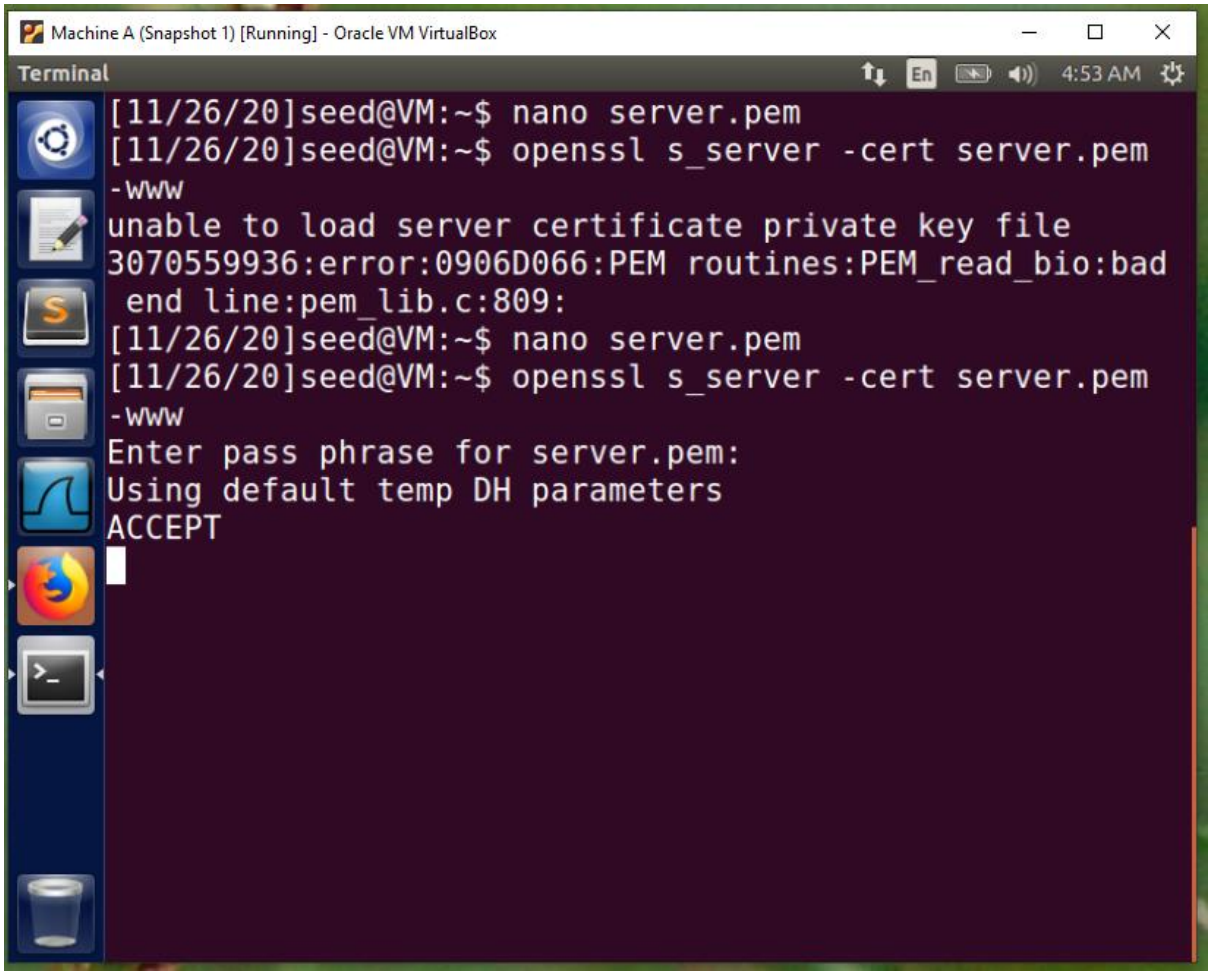



**Step 4: (1) Modify a single byte of server.pem, and restart the server, and reload the URL. What do you observe? Make sure you restore the original server.pem afterward. Note: the server may not be able to restart if certain places of server.pem is corrupted; in that case, choose another place to modify.**

- After modifying the bytes in server.pem file, I restarted the server using below command: *openssl s_server -cert server.pem -www*
- Observed error "unable to load server certificate private key file". Hence, https://SEEDPKILab2018.com:4433 did not worked.

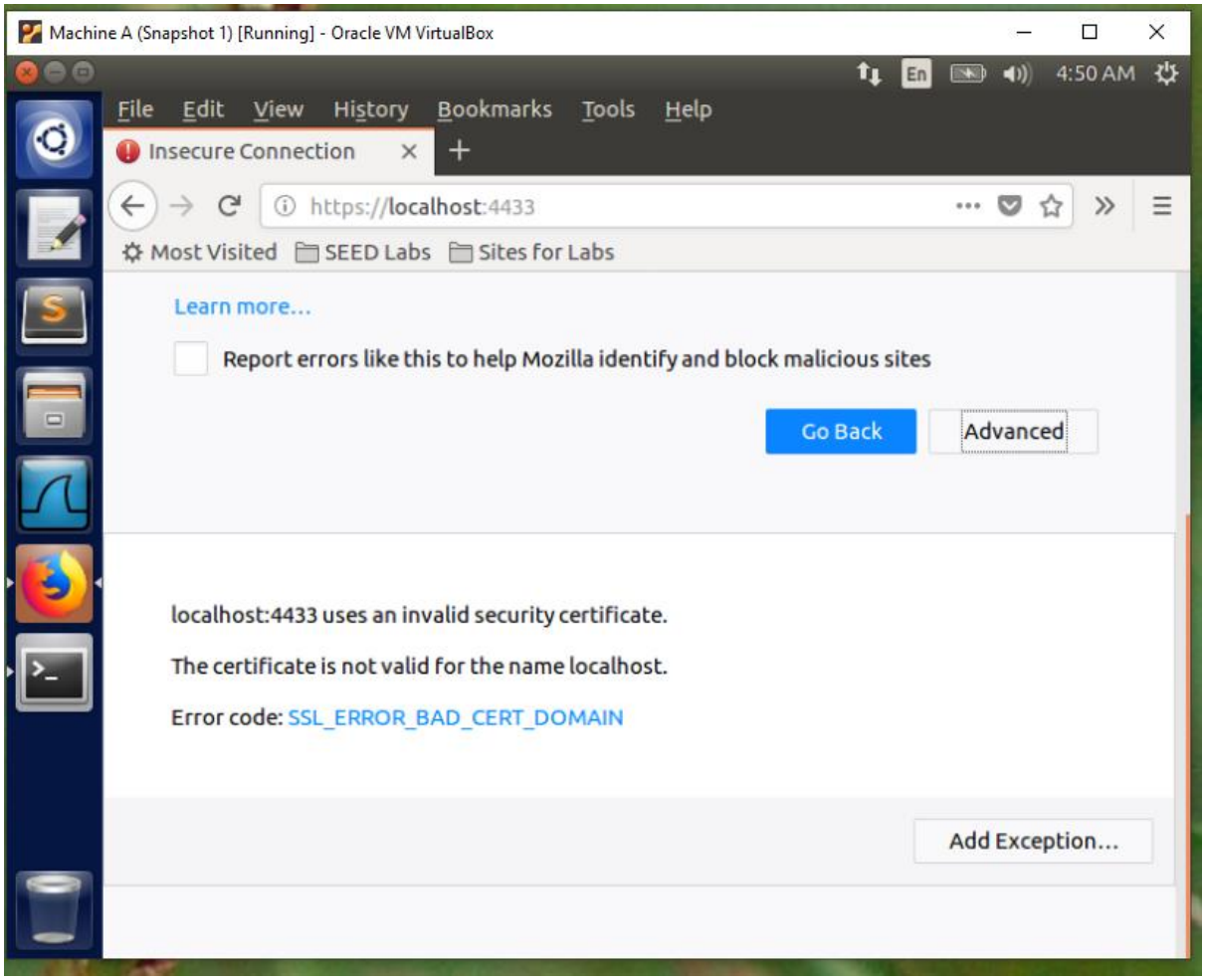



- Restoring back the original server.pem file. Then restarted the server using command *openssl s_server -cert server.pem -www*
- The given snapshot is showing ACCEPT which means that the server has been restarted successfully.



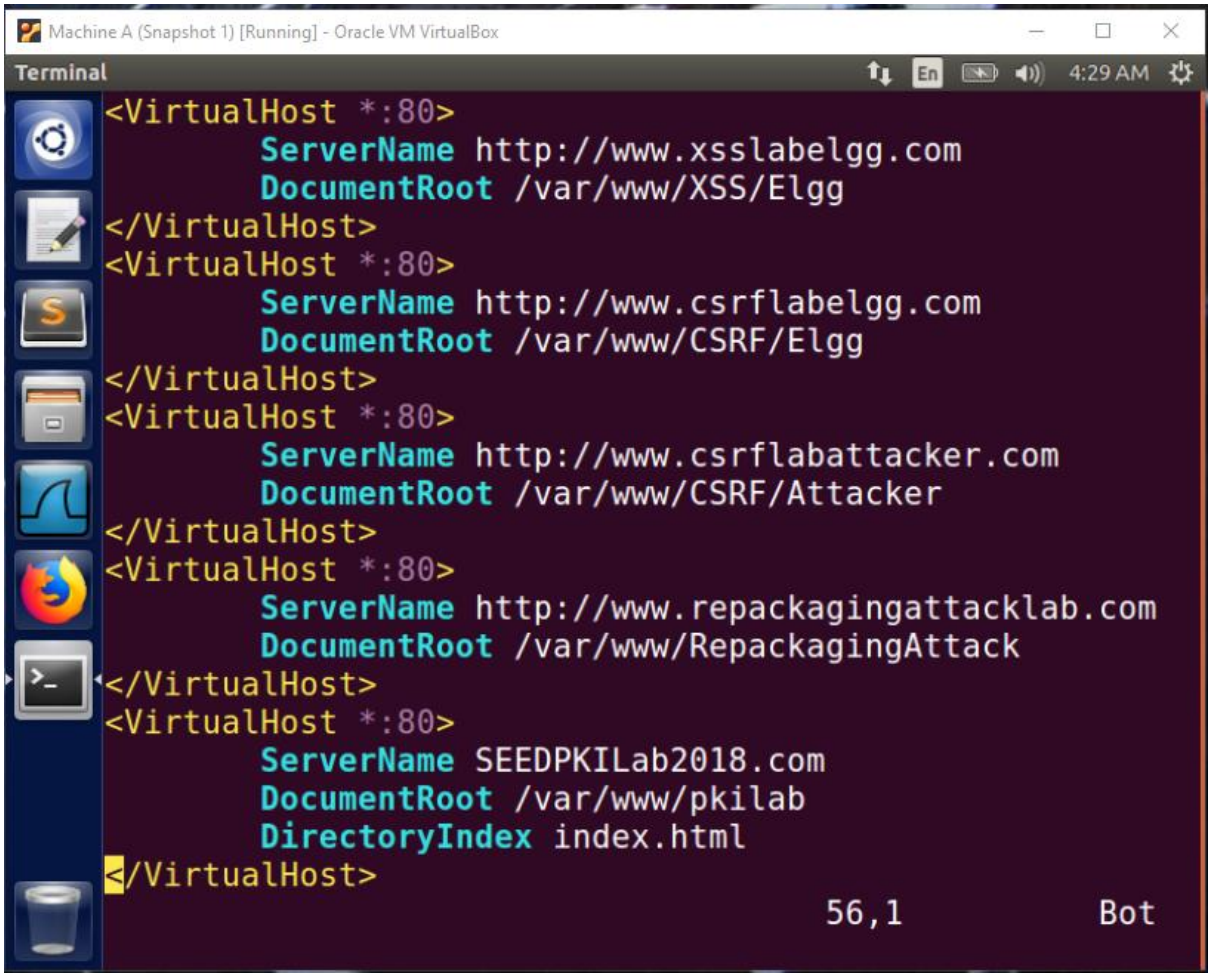

**Step 4: (2) Since SEEDPKILab2018.com points to the localhost, if we use https://localhost:4433 instead, we will be connecting to the same web server. Please do so, describe and explain your observations.**

After browsing https://localhost:4433 on Firefox browser, I observed that the browser gave Bad cert domain "The certificate is not valid for the name localhost". The certificate is tied to SEEDPKIlab2018.com not to localhost. Refer below snapshot showing error message:

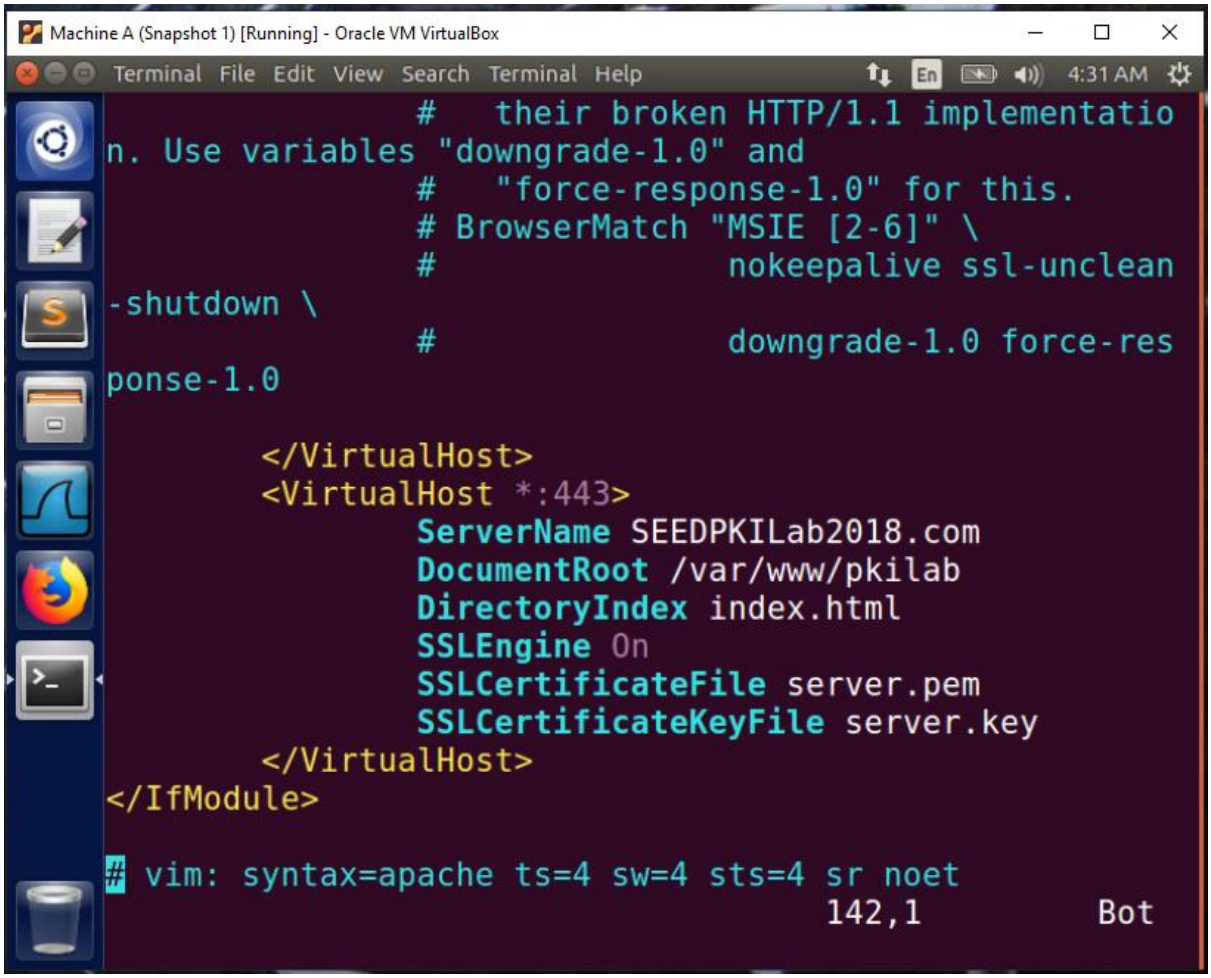



**Task 4: Deploying Certificate in an Apache-Based HTTPS Website**

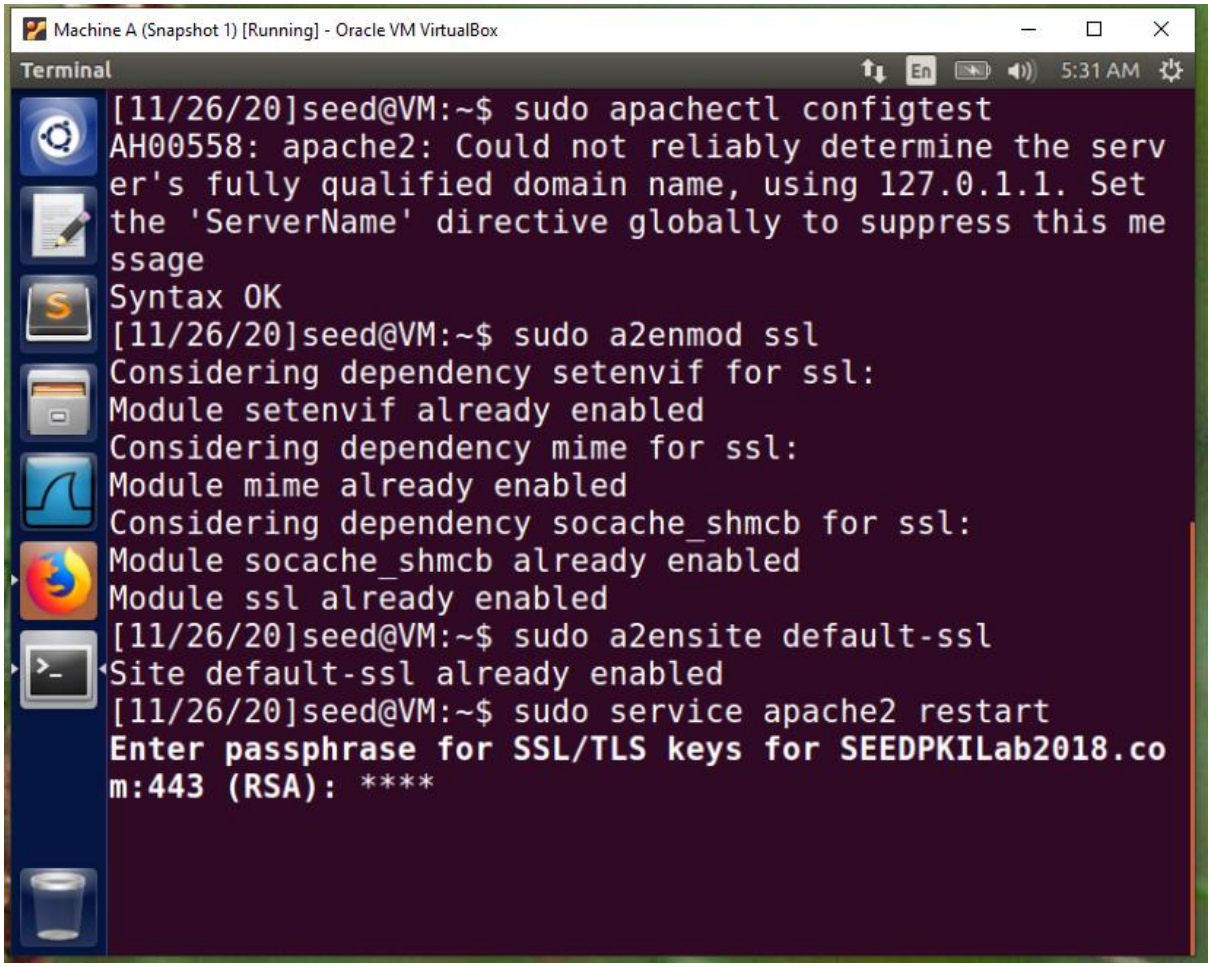
- Configuring an Apache server to build an HTTPS website i.e SEEDPKIlab2018.com, so it knows where to get the private key and certificates.
- An Apache server can host multiple websites simultaneously. The directory where a website's files are stored needs to be identified via its VirtualHost file. Adding HTTP website entry in /etc/apache2/sites-available directory and 000-default.conf file.



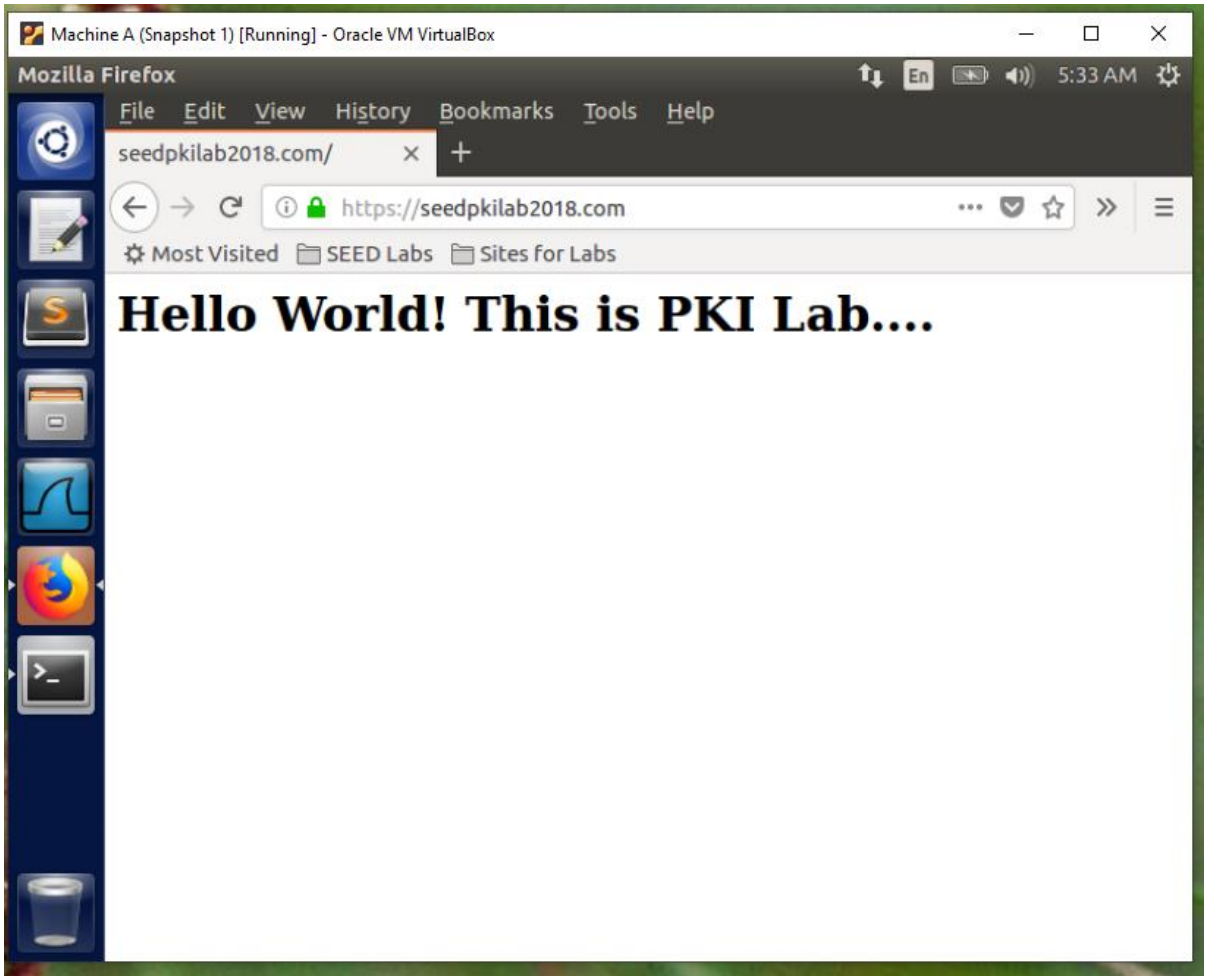

- Now adding HTTPS website's virtual host entry in default-ssl.conf file of /etc/apache2/sites-available directory.




- After modifying both the files in /etc/apache2/sites-available folder, executed below commands to enable SSL. So, that the traffic between the browser and the server will be encrypted while browsing the site.
  *sudo apachectl configtest*
  *sudo a2enmod ssl*
  *sudo a2ensite default-ssl*
  *sudo service apache2 restart*



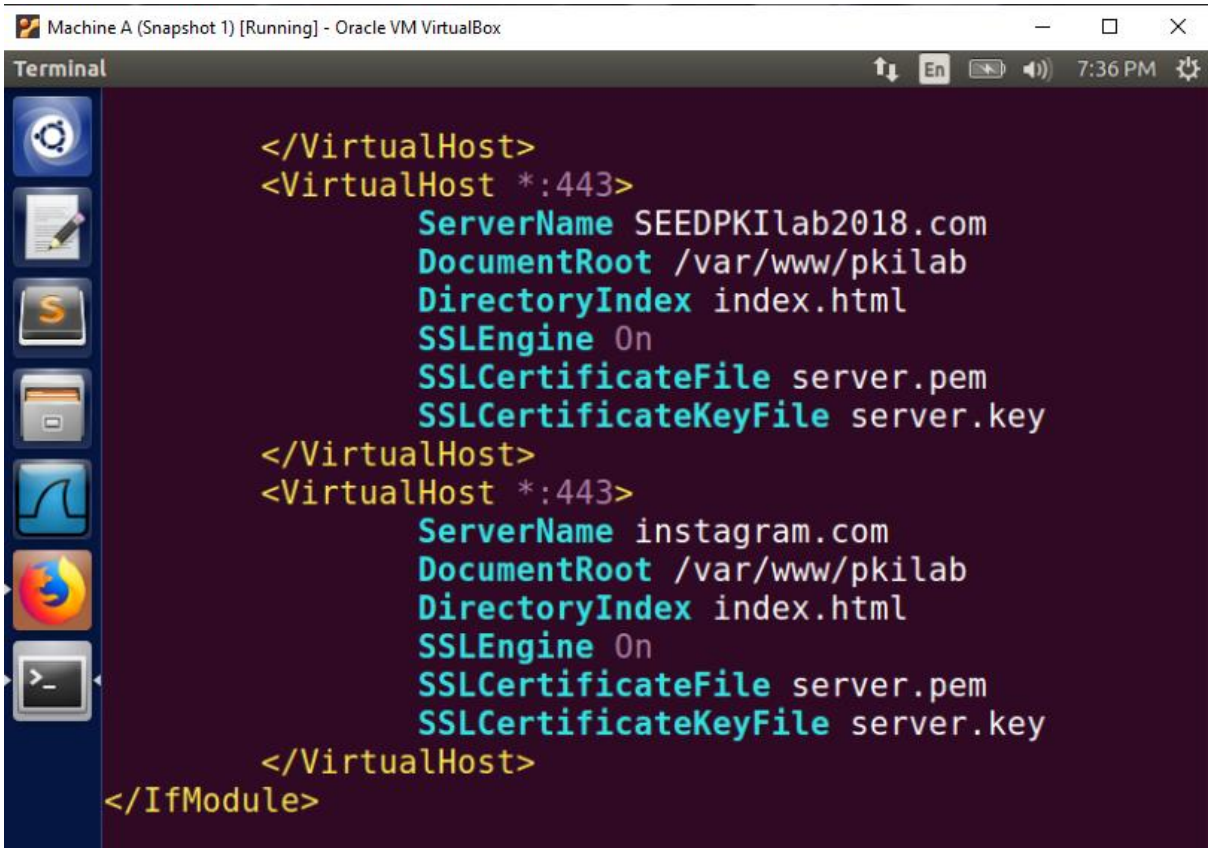

- Now again browse the website https://seedpkilab2018.com: 443 on Firefox browser and the site was successfully browsed. Refer below snapshot:



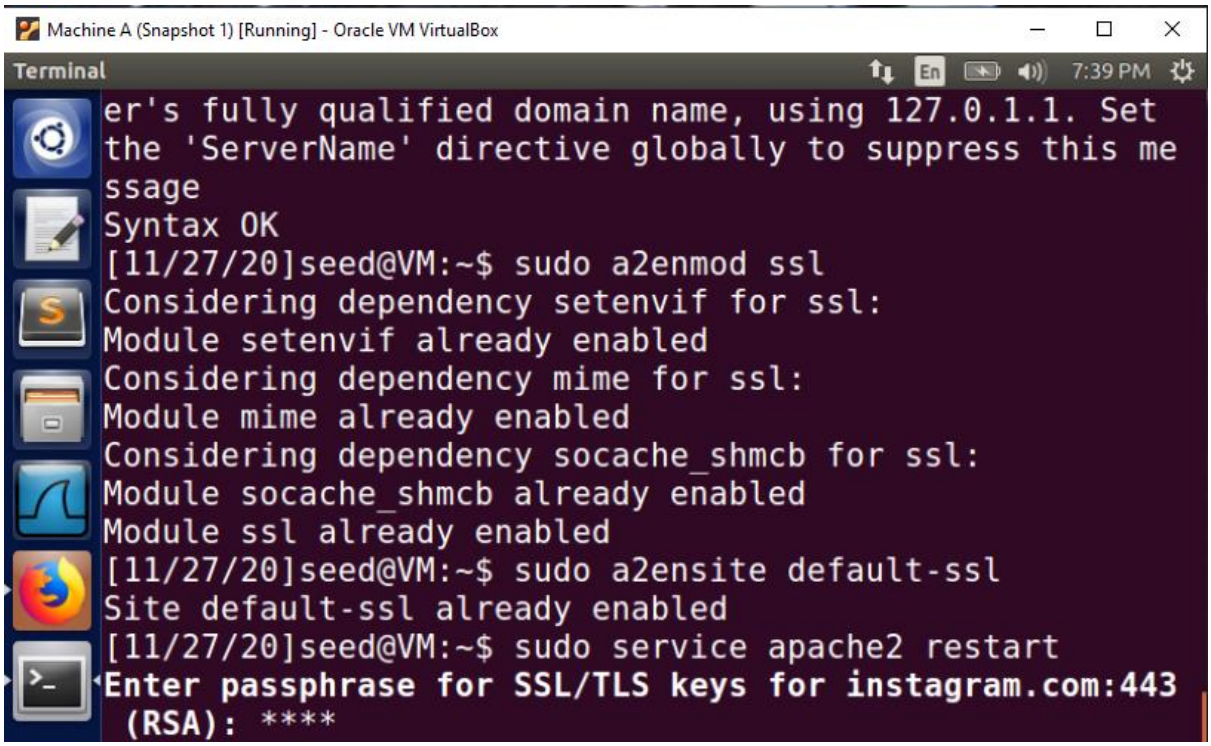

**Task 5: Launching a Man-In-The-Middle Attack**

**Step 1: Setting up the malicious website**

- I have used target website as Instagram.com. Therefore, made virtual host entry in /etc/apache2/sites-available/default-ssl.conf

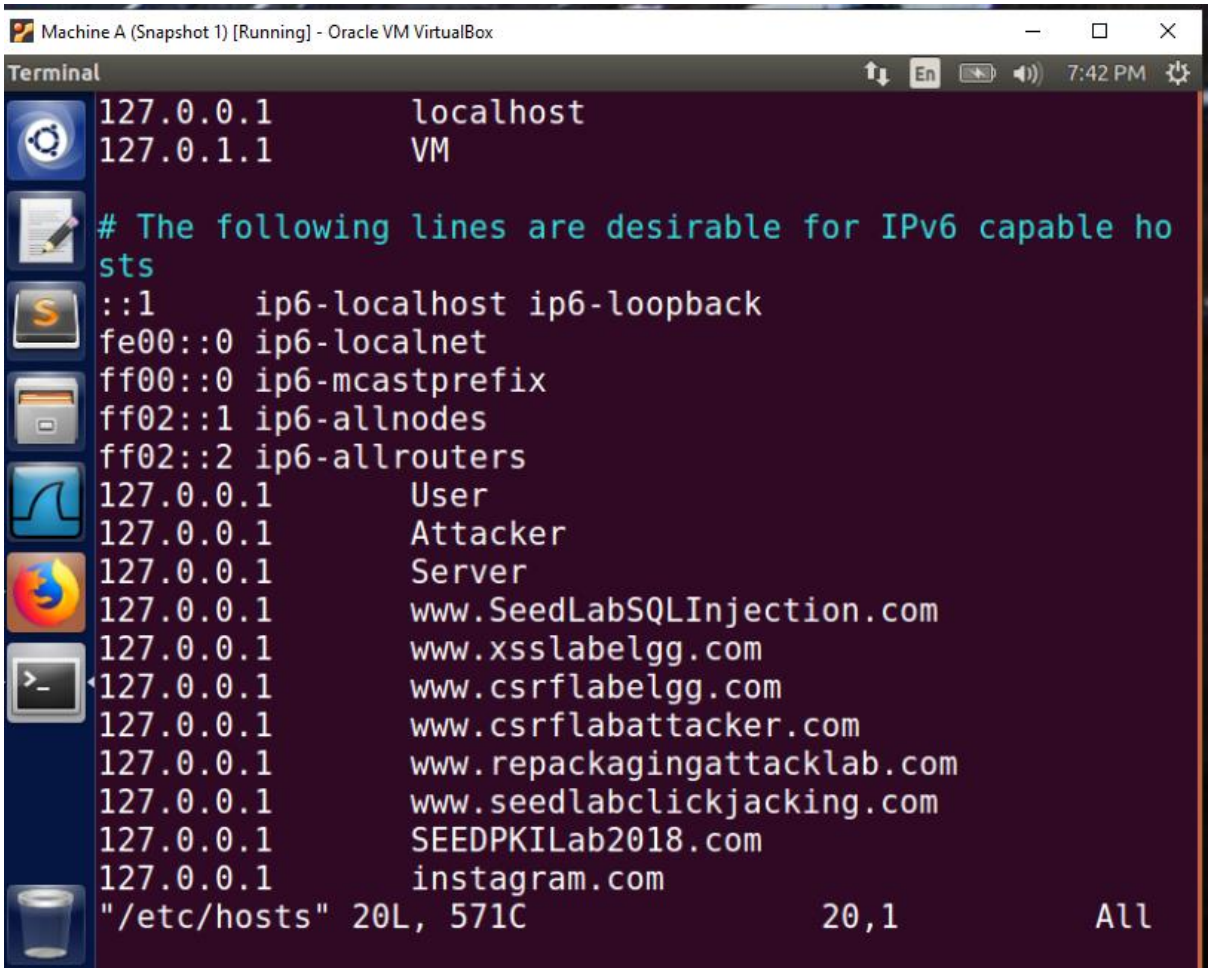



- After adding virtual host, restarted apache server using below commands:
  *sudo apachectl configtest*
  *sudo a2enmod ssl*
  *sudo a2ensite default-ssl*
  *sudo service apache2 restart*



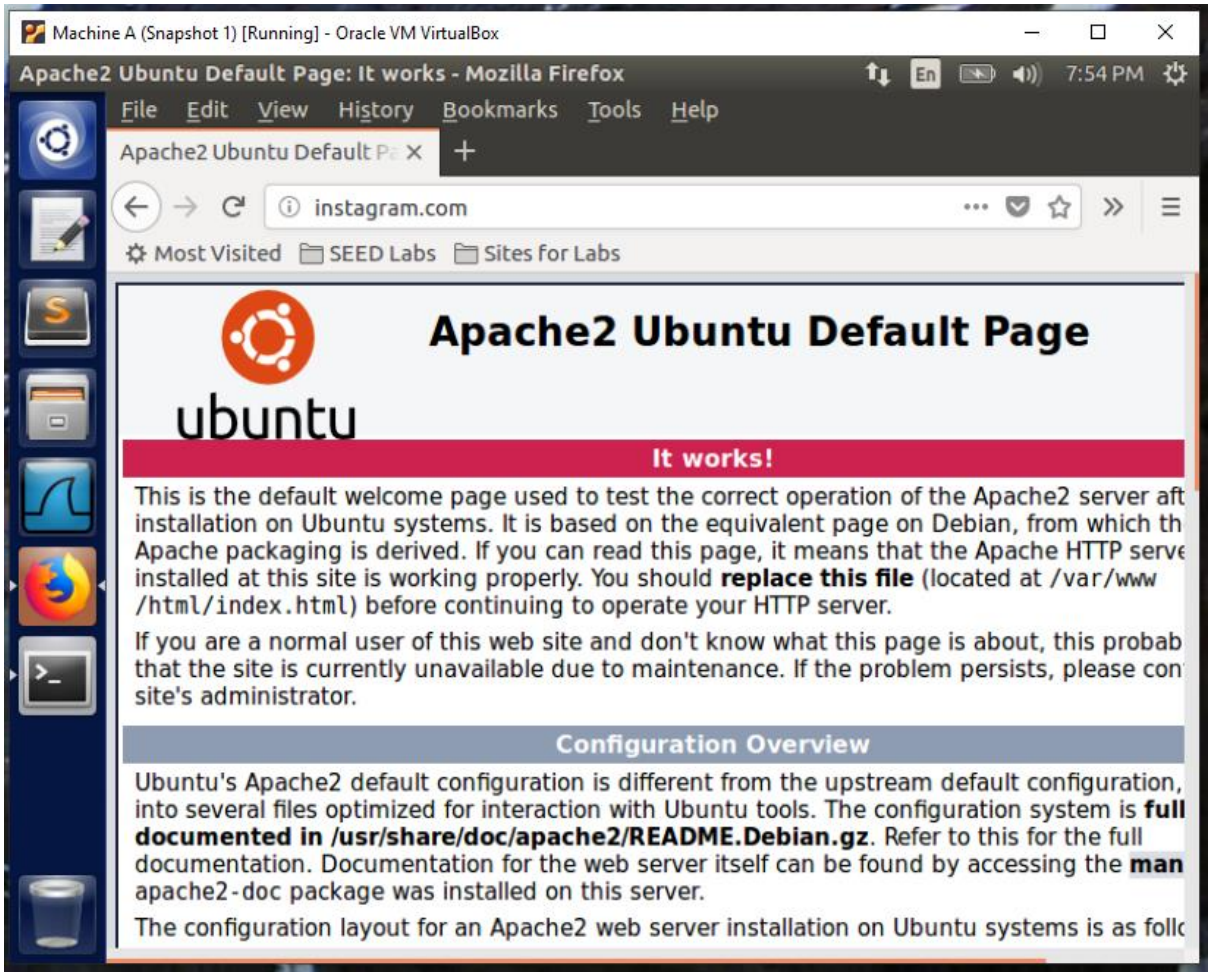
**Step 2: Becoming the man in the middle**

- Modified the host entry in /etc/hosts of victim's machine by emulating the result of DNS cache positing attack. Added Instagram.com entry as shown in snapshot:



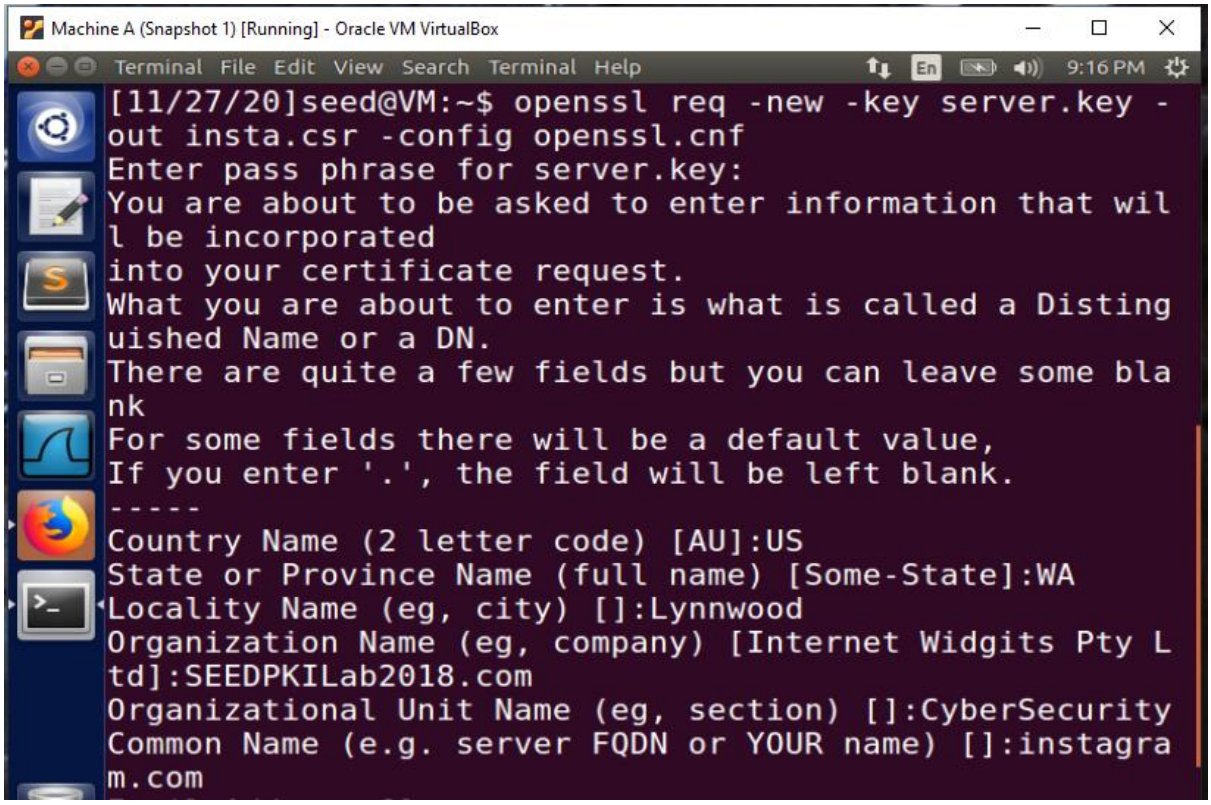

**Step 3: Browse the target website**

- After restarting the apache server using command:
  *sudo service apache2 restart*
- When I browsed https://instagram.com on Firefox browser, observed the page of "apache2 ubuntu default page". Refer below snapshot:

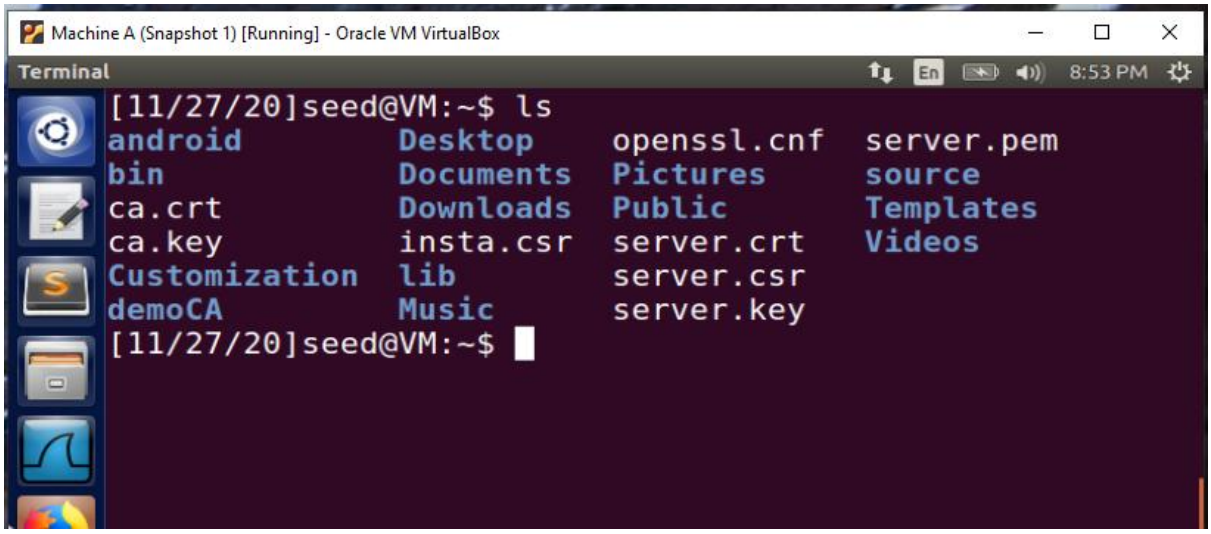



**Task 6: Launching a Man-In-The-Middle Attack with a Compromised CA**

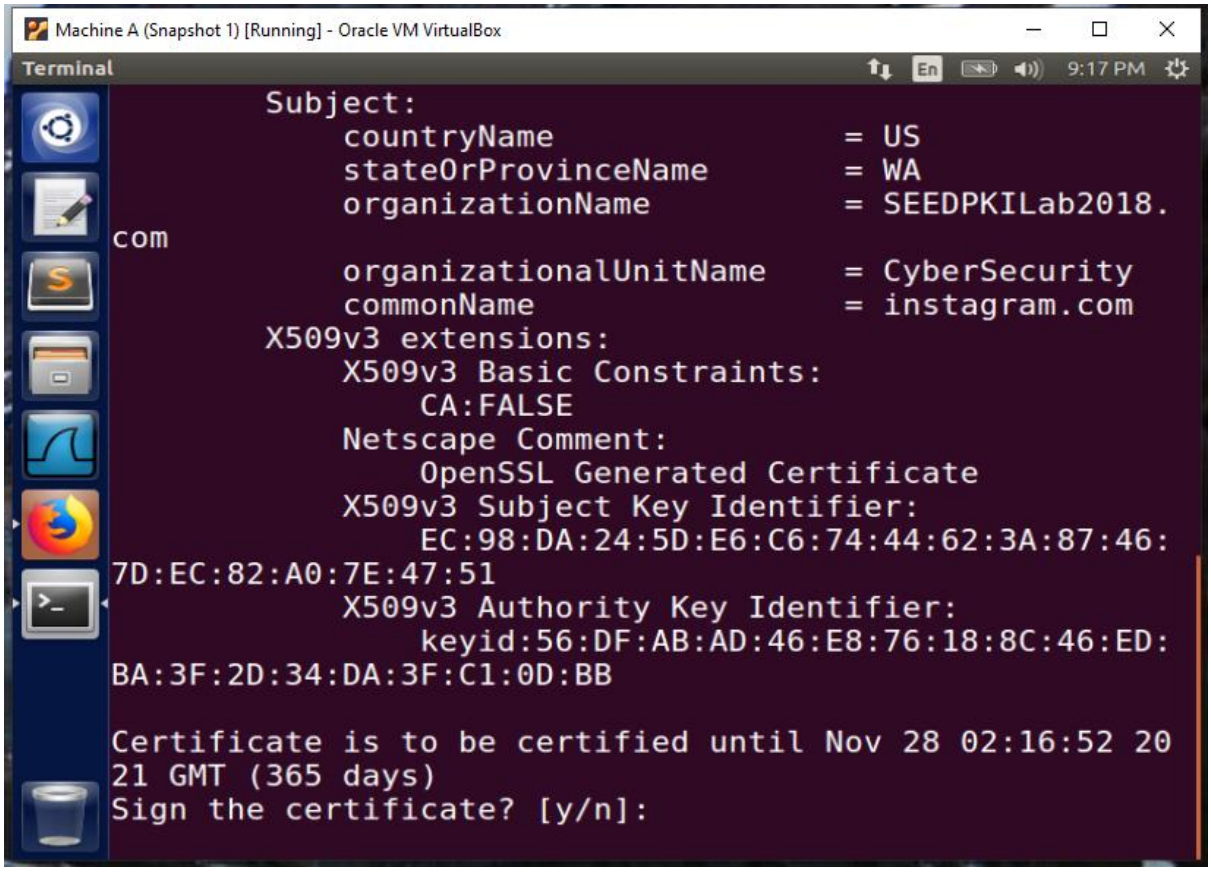
- Generating certificate signing request (csr) for instagram.com by using given command: *openssl req -new -key server.key -out insta.csr -config openssl.cnf*

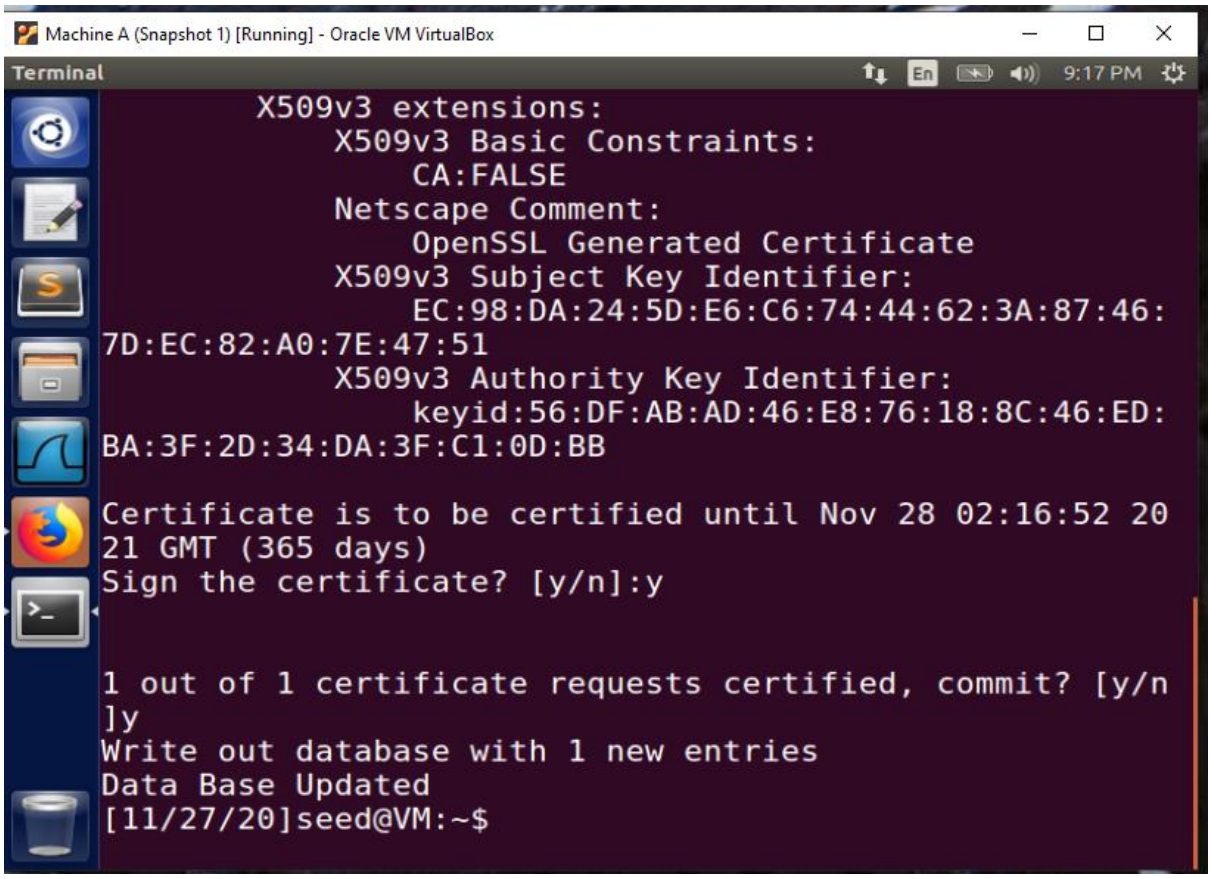



- Certificate signing request (csr) has been generated → insta.csr

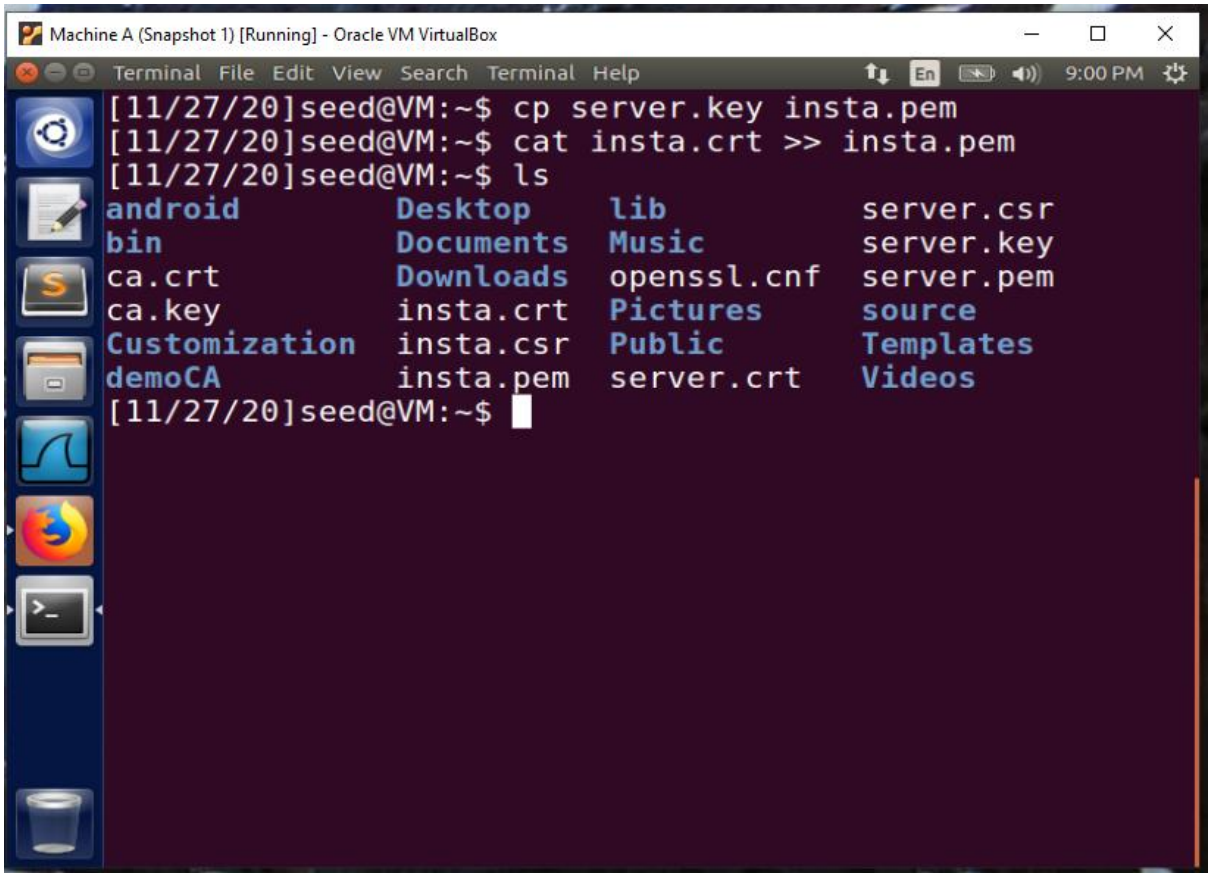



- Now generating the CA certificate for Instagram.com using following command:
  *openssl ca -in instagram.csr -out instagram.crt -cert ca.crt -keyfile ca.key -config openssl.cnf*

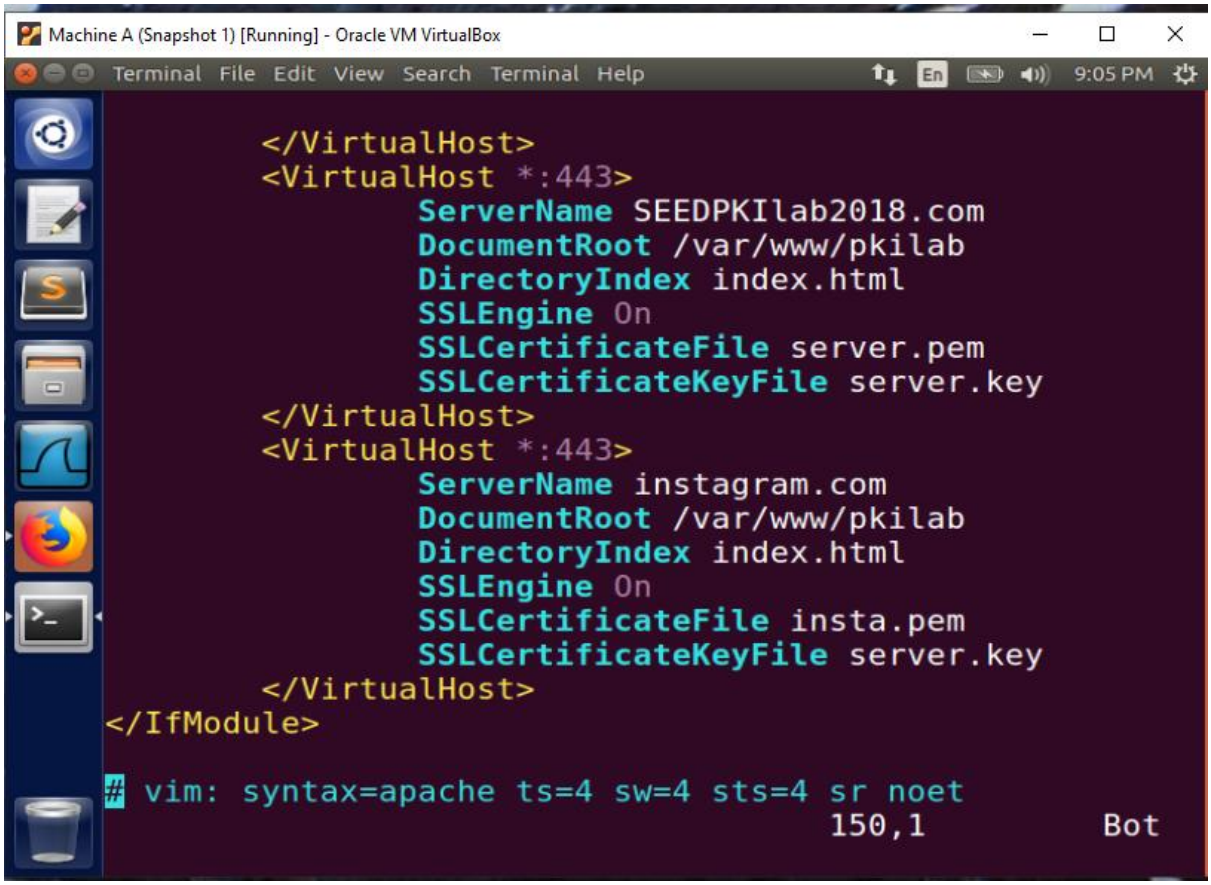


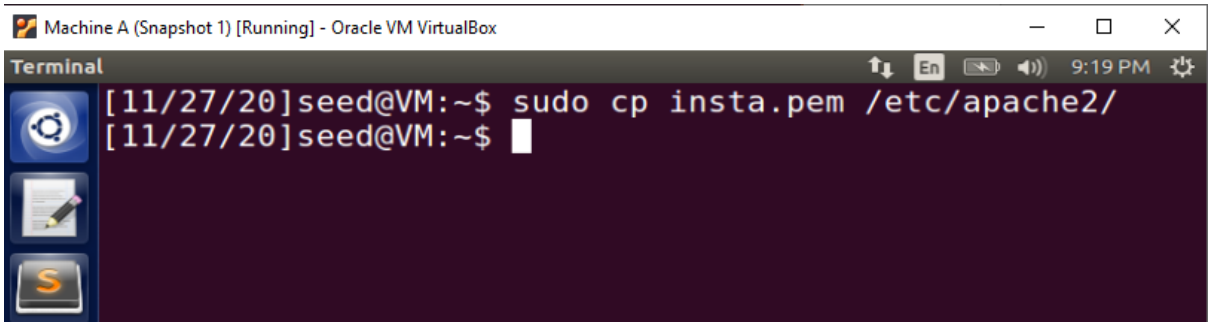

- Configuring the web server by combining the secret key and certificate in one file for Instagram.com



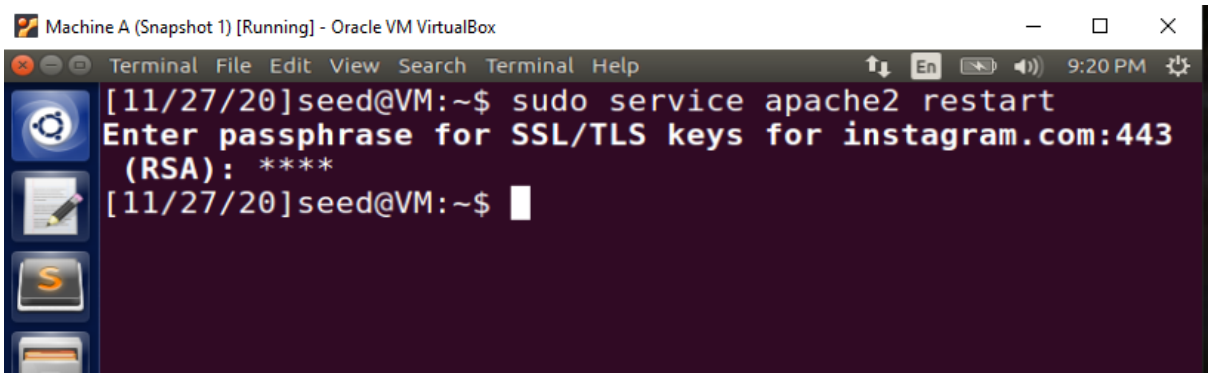

- Modified server certificate from server.pem to insta.pem



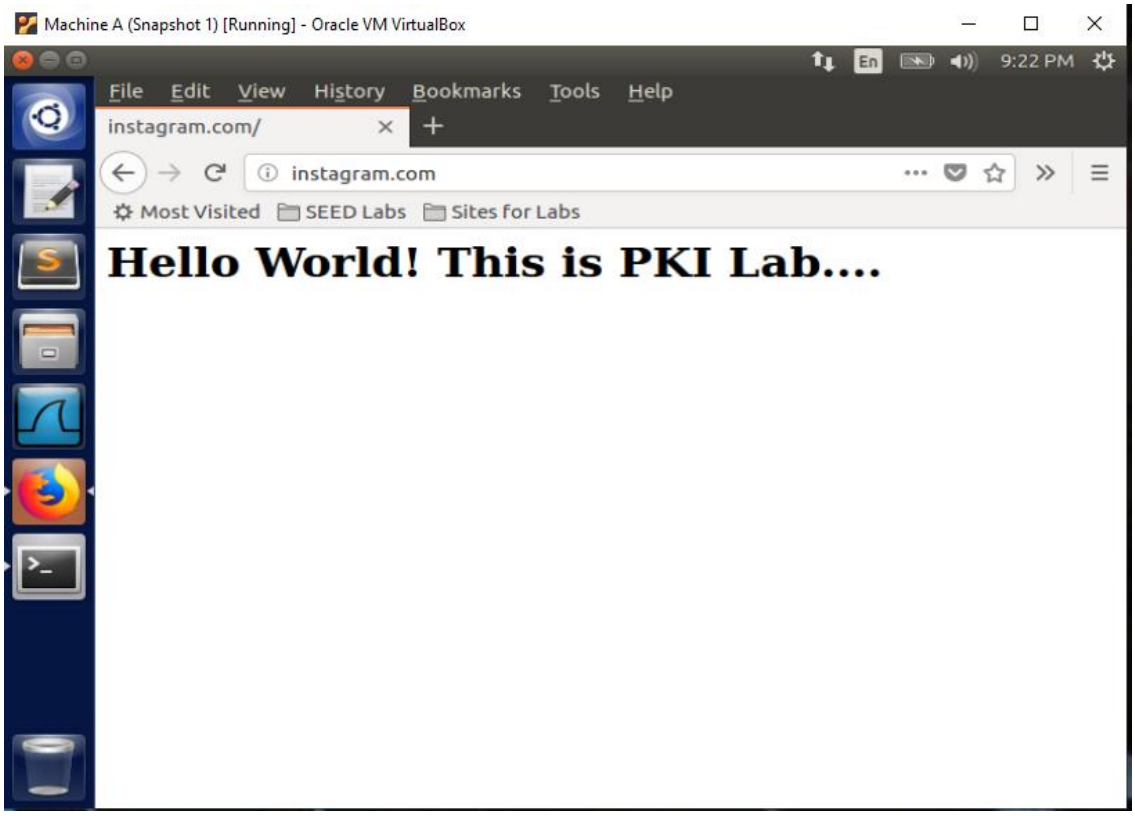
- Copying insta.pem file in /etc/apache2/ directory.




- Now restart the apache server using given command:
  *service apache2 restart*




- After browsing https://instagram.com on Firefox browser, I observed that MITM attack was successful. Refer below snapshot of Firefox browser showing the MITM attack:




In overall lab, I found MITM attack was very interesting and surprising.