Author: Wong Tin Kit
Student ID: 1003331

This task was done on SEED Ubuntu Clone of IP 10.0.2.10

Task 1: Becoming a Certificate Authority (CA)

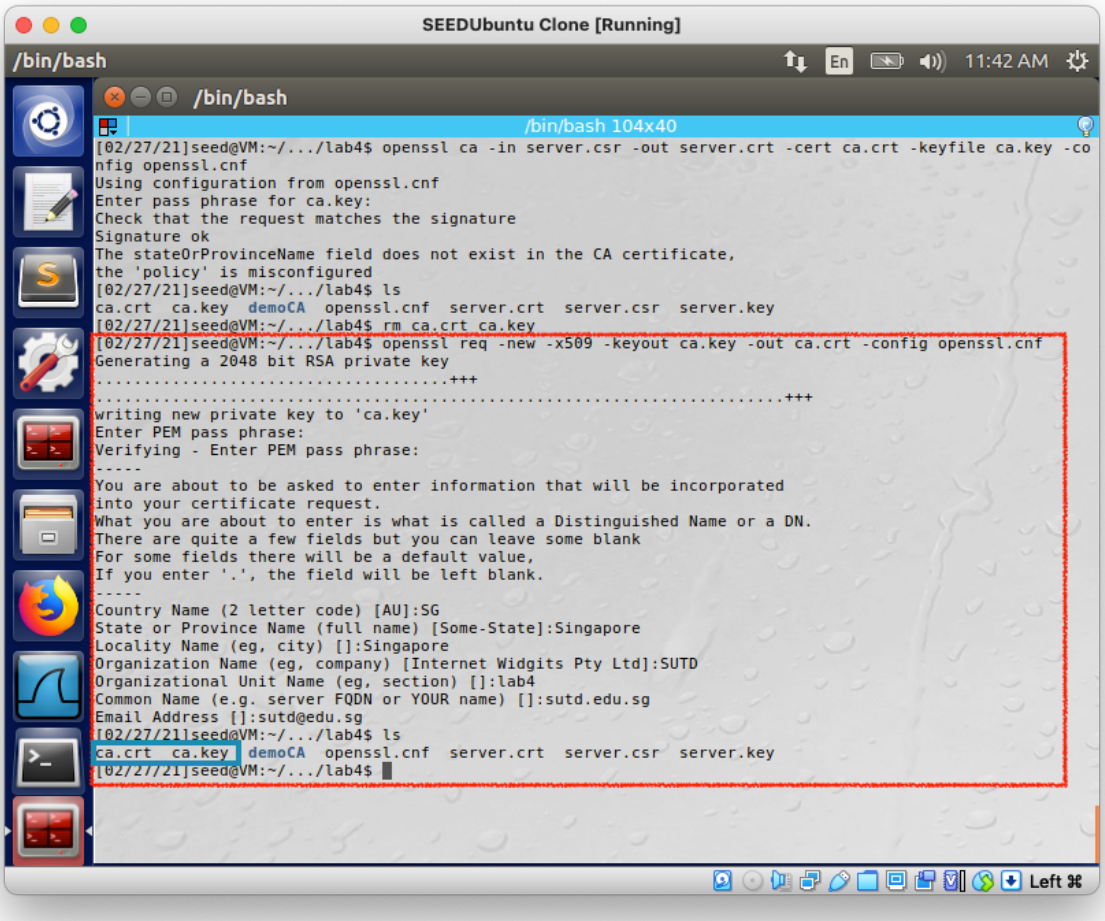

Fig. 1 Getting CA certificate and key

The PEM pass phrase used is ***tinkit***.
Fig. 1 shows the process in creating ***ca.crt*** and ***ca.key*** in the red highlighted box. We confirm the creation with ***ls*** and see that the ***ca.crt*** and ***ca.key*** are created.

Task 2: Creating a Certificate for **SEEDPKILab2020.com**

Step 1: Generate public/private key pair.

Passphrase for server.key: seedpkilab2020

The image below depicts the text output of ***server.key***.

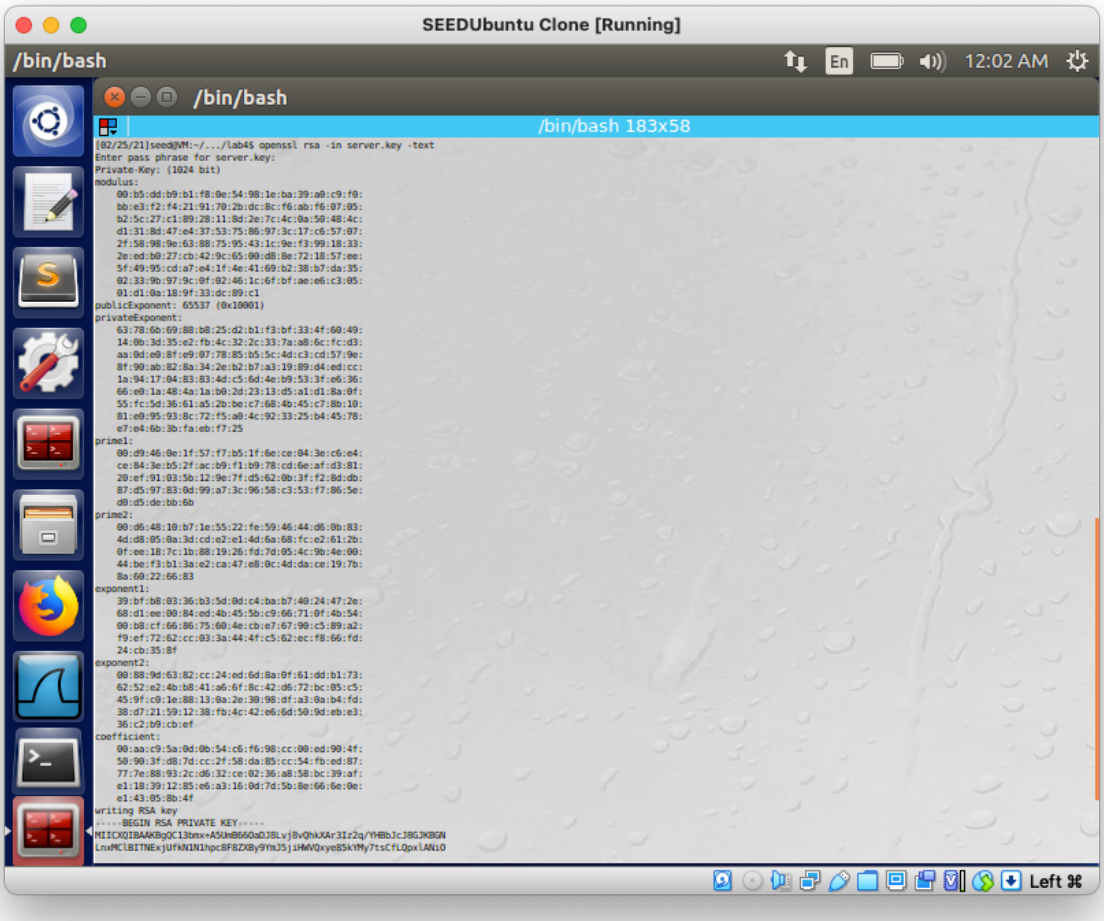

Fig. 2.1 Output of ***serer.key***

Step 2: Generate a Certificate Signing Request (CSR)

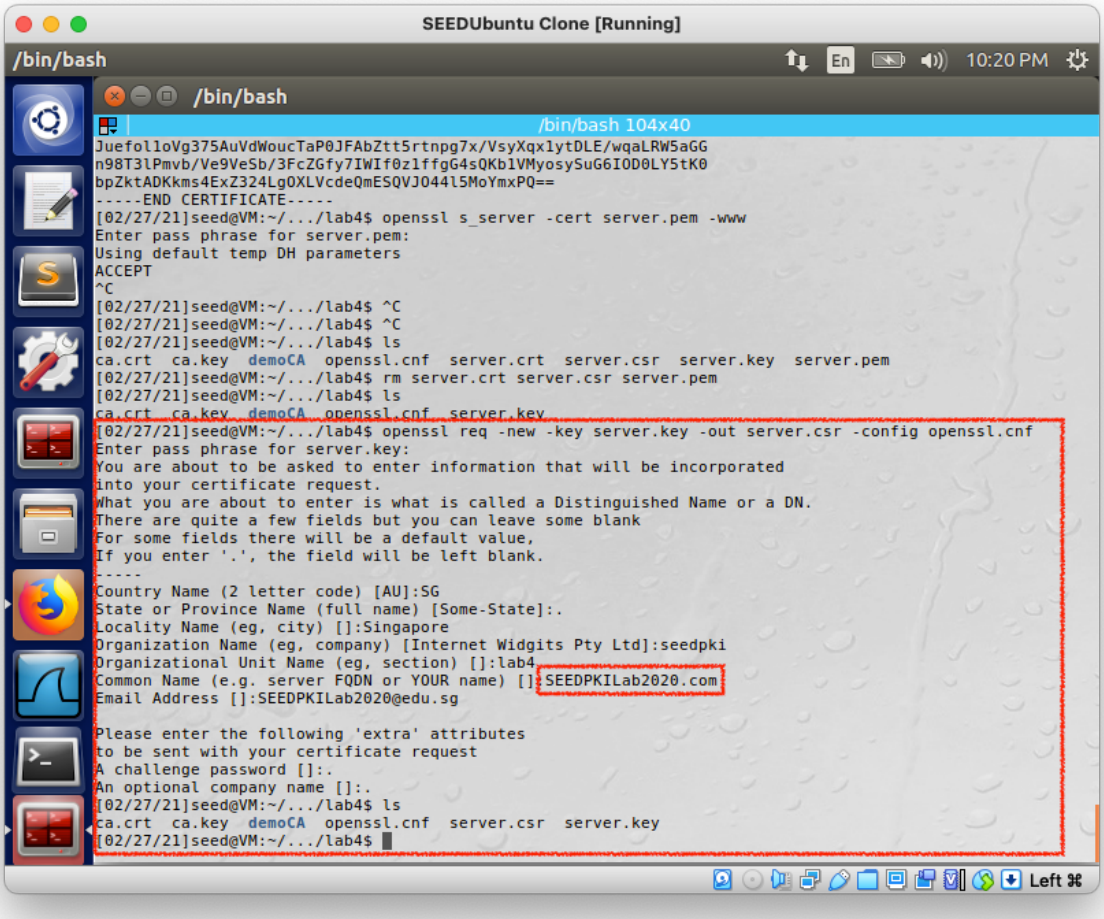

Fig. 2.2 Configuration in generating the certificate signing request.

As requested, I used SEEDPKILab2020.com as the Common Name in this CSR.

Step 3: Generating Certificates.

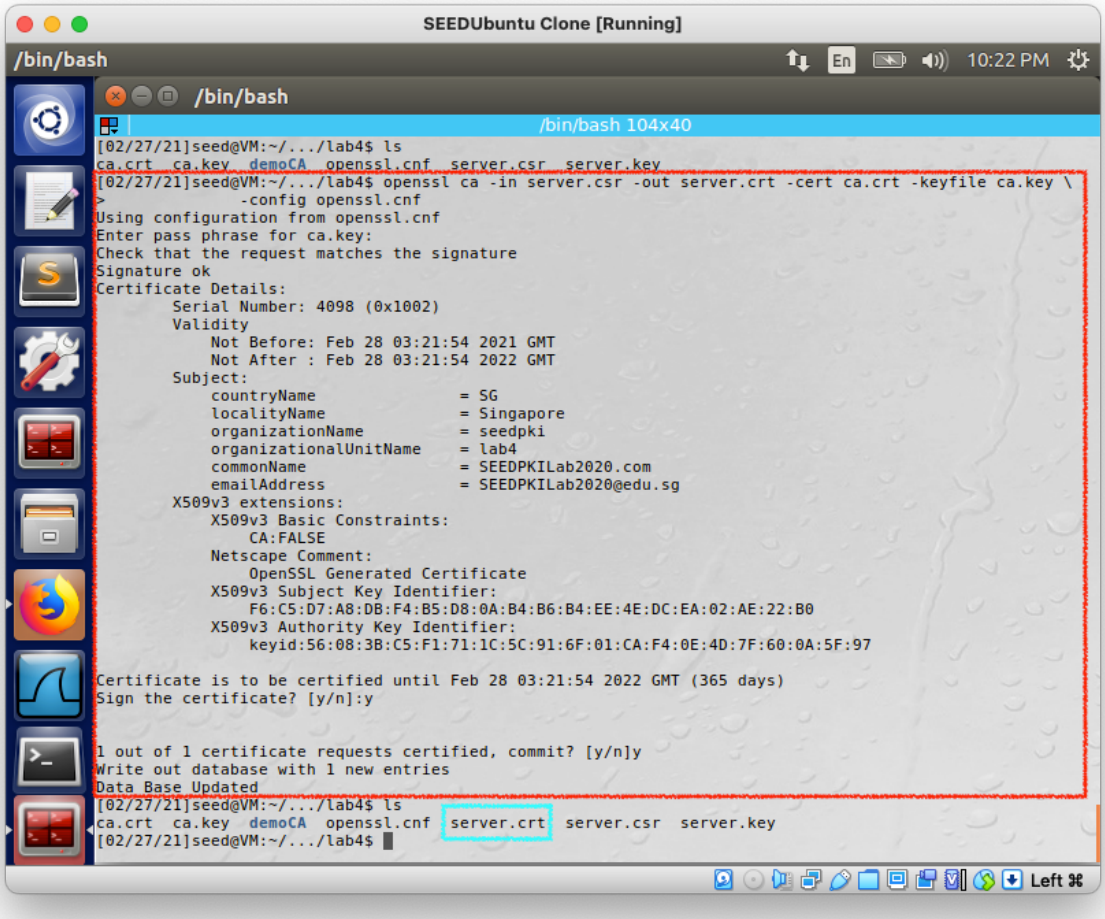

Fig. 2.3 Completion of CSR

In this case, I configured the policy section ***openssl.cnf*** to match anything. Fig. 2.3 shows the successful completion of the CA signing a CSR. We see ***server.crt*** as a result (highlighted in the cyan box).

Task 3: Deploying Certificate in an HTTPS Web Server

Step 1: Configuring DNS.

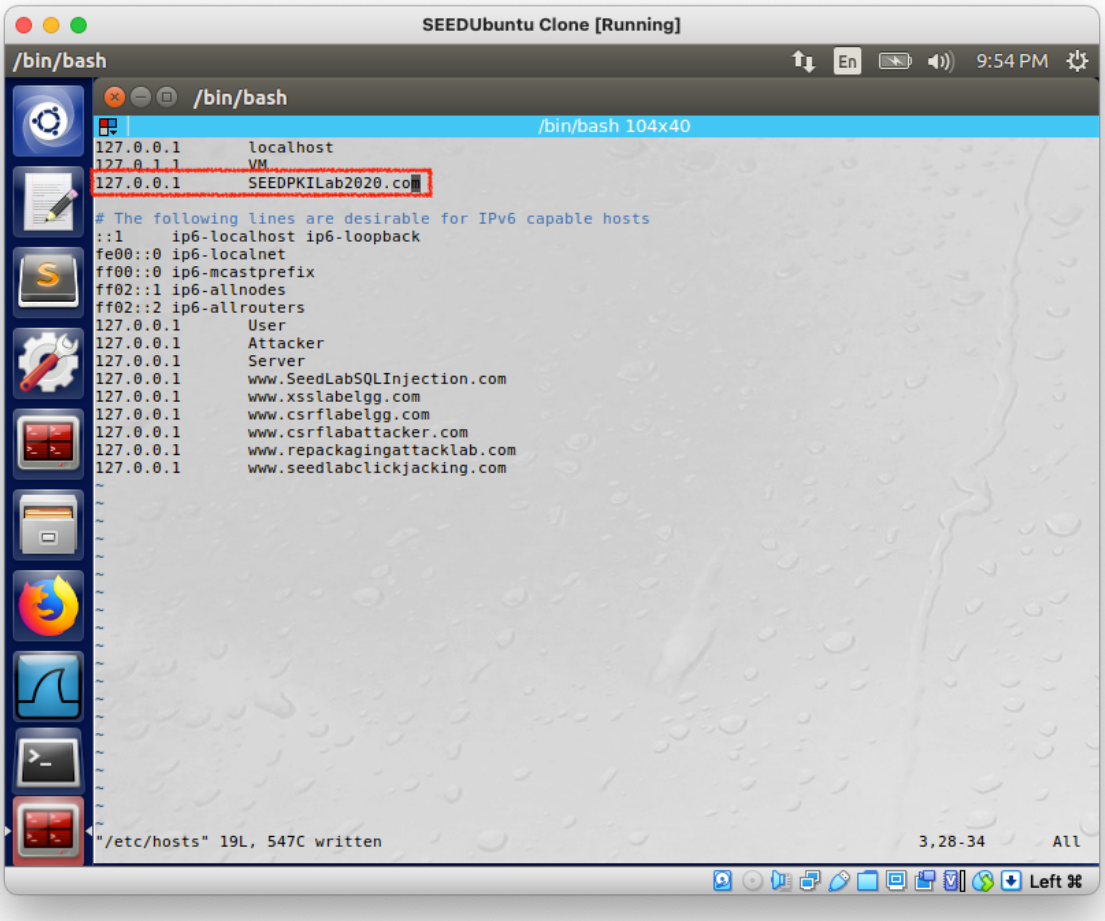

Fig. 3.1 Adding SEEDPKILab2020.com to */etc/hosts*

Fig. 3.1 shows the addition of the mapping of localhhost to ***SEEDPKILab2020.com***.

Step 2: Configuring the web server.

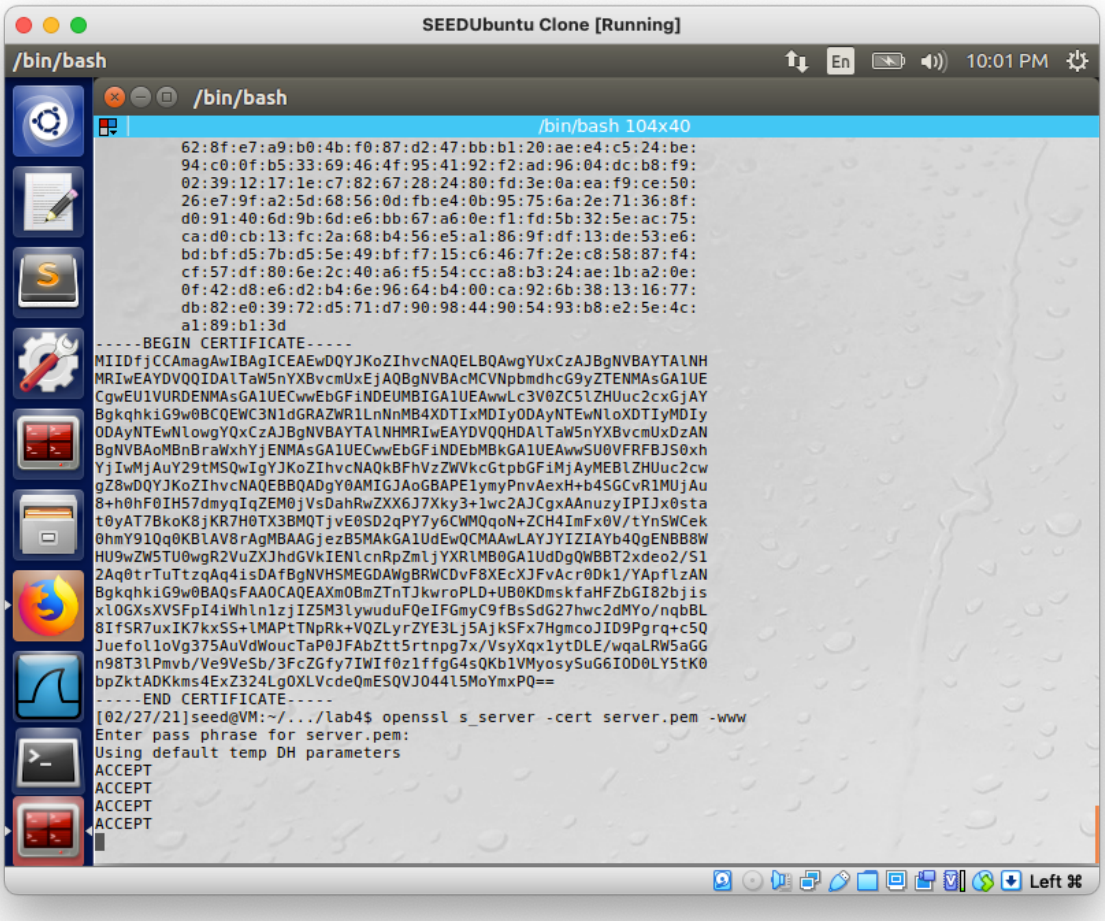

Fig. 3.2 Running OpenSSL HTTPS server

Fig. 3.2 shows the stdout of running the openssl https server and mozilla attempts to connect to it. We run the command as such ***openssl s_server -cert server.pem -www.*** The ***www*** flag helps in debugging and sends a status message back to the client when it connects. This includes lots of information about the ciphers used and various session parameters. The output is in HTML format so this option will normally be used with a web browser.

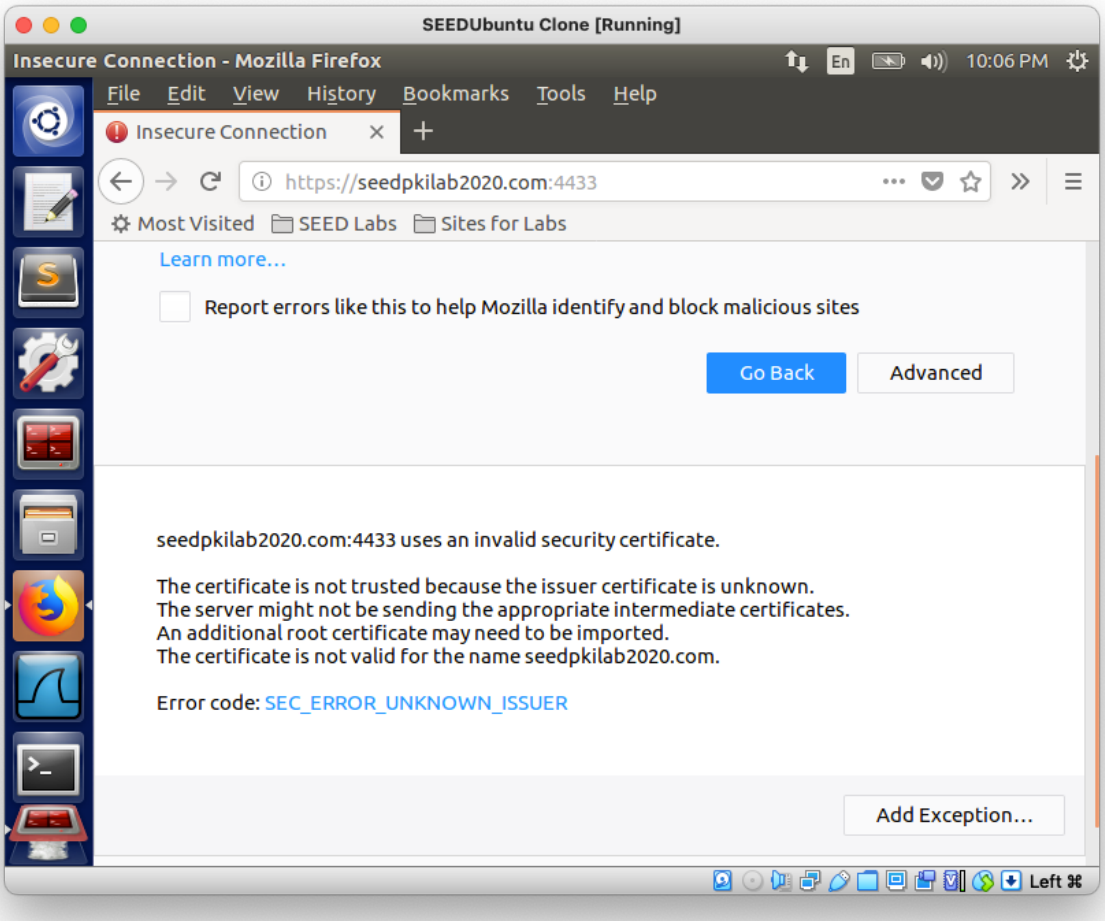

Fig. 3.3 Mozilla Browser Connection to ***https://seedpkilab2020.com***

Fig. 3.3 shows an attempt by Mozilla Browser to connect to ***https://seedpkilab2020.com***. This warning is expected as the browser does not recognise the Issuer of the website's certificate after checking it against Mozilla's list of trusted CAs.

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

Load ***ca.crt*** into Mozilla Firefox Browser by doing the following:

Edit -> Preference -> Privacy & Security -> View Certificates. -> Authorities -> Import -> Load ***ca.crt***

Step 4. Testing our HTTPS website.

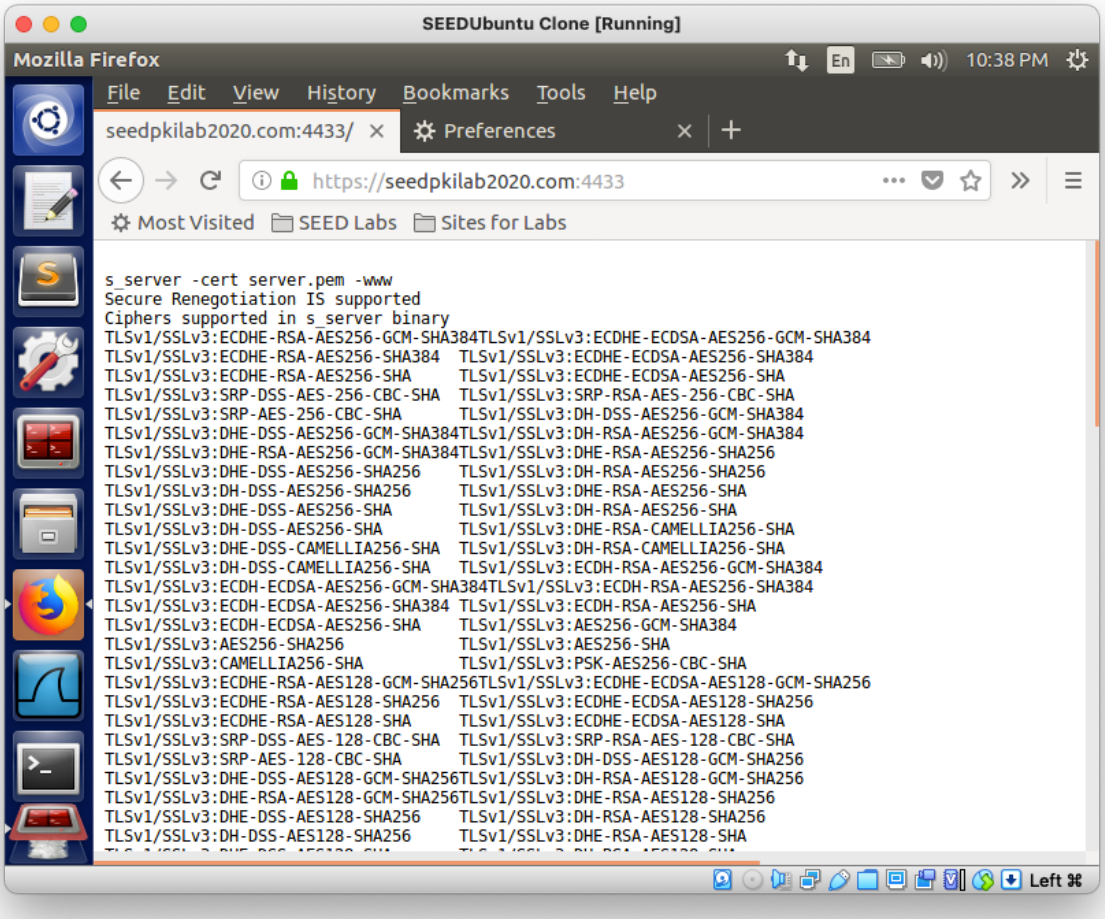

Fig. 3.4 First portion of information sent back to Mozilla Client

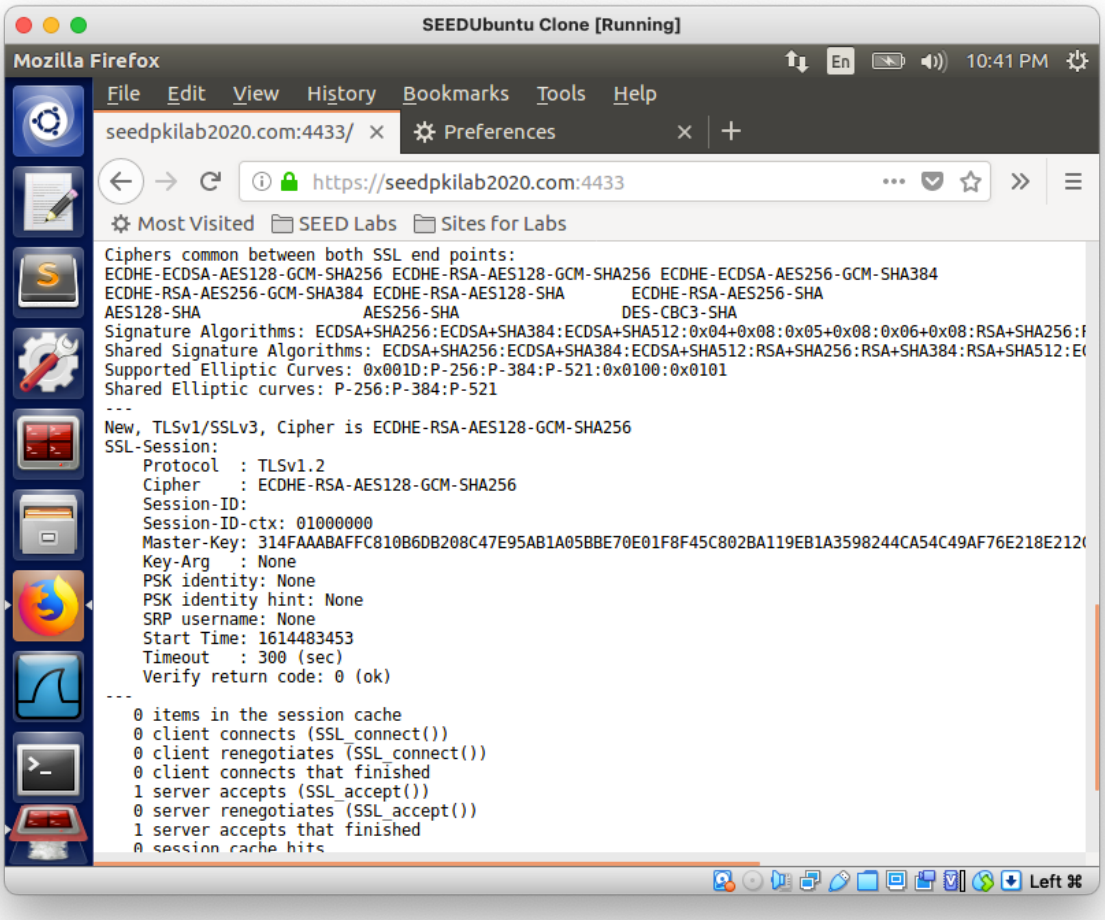

Fig. 3.5 Second portion of information sent back to Mozilla Client

Figures 3.4 and 3.5 show the entire set of status messages sent back to Mozilla Client from the openssl HTTPS server. Fig. 3.4 shows the available cipher suites supported by the openssl server. The start of Fig. 3.5 shows the cipher suits common to both SSL endpoints (openssl server and Mozilla Client). The second half of Fig. 3.5 shows the new SSL-Session setup between Mozilla Client and OpenSSL server.

Additional Tasks

1. Modify a single byte of server.pem, and restart the server, and reload the URL. What do you observe?

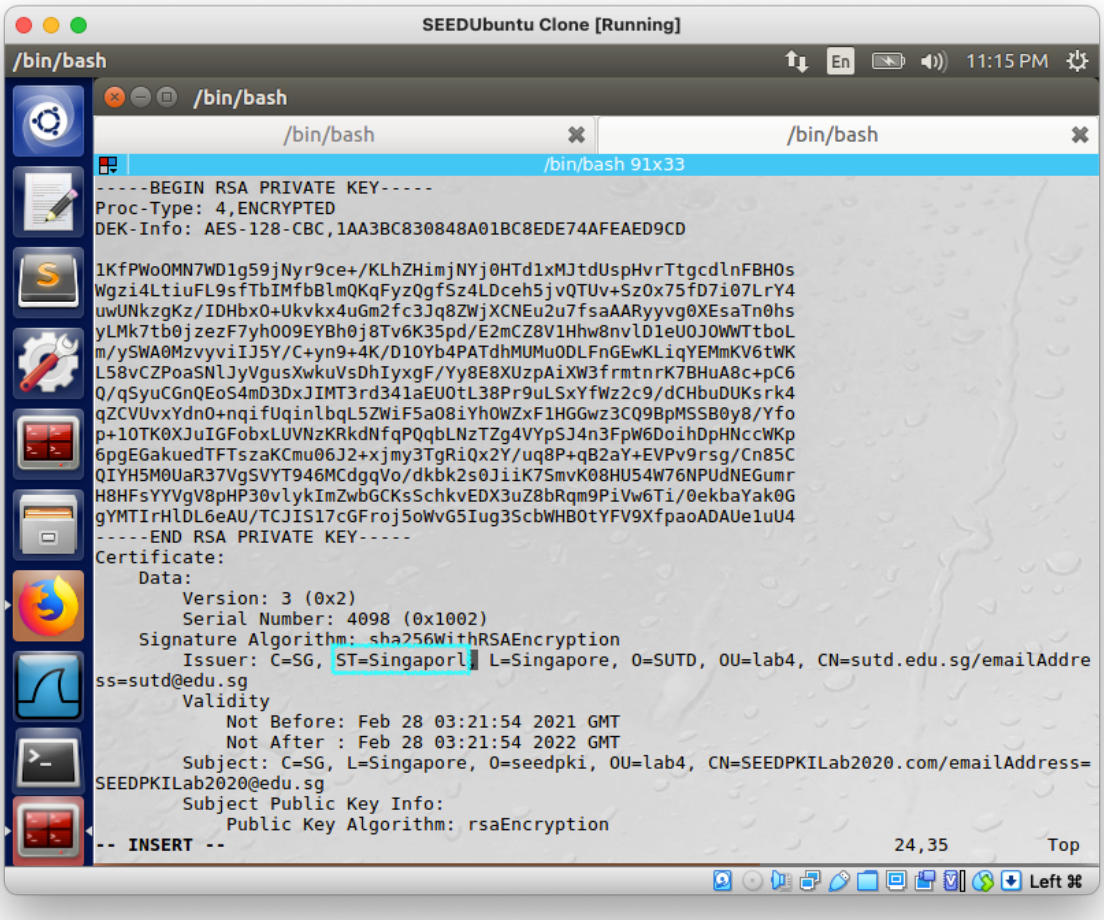

Fig. 3.4.1 Changing ST field in ***server.pem***

In Fig. 3.4.1, we changed the ST field to ***Singapor*l** and reloaded the openssl server before connecting to it via Mozilla as shown in Fig. 3.4.2.

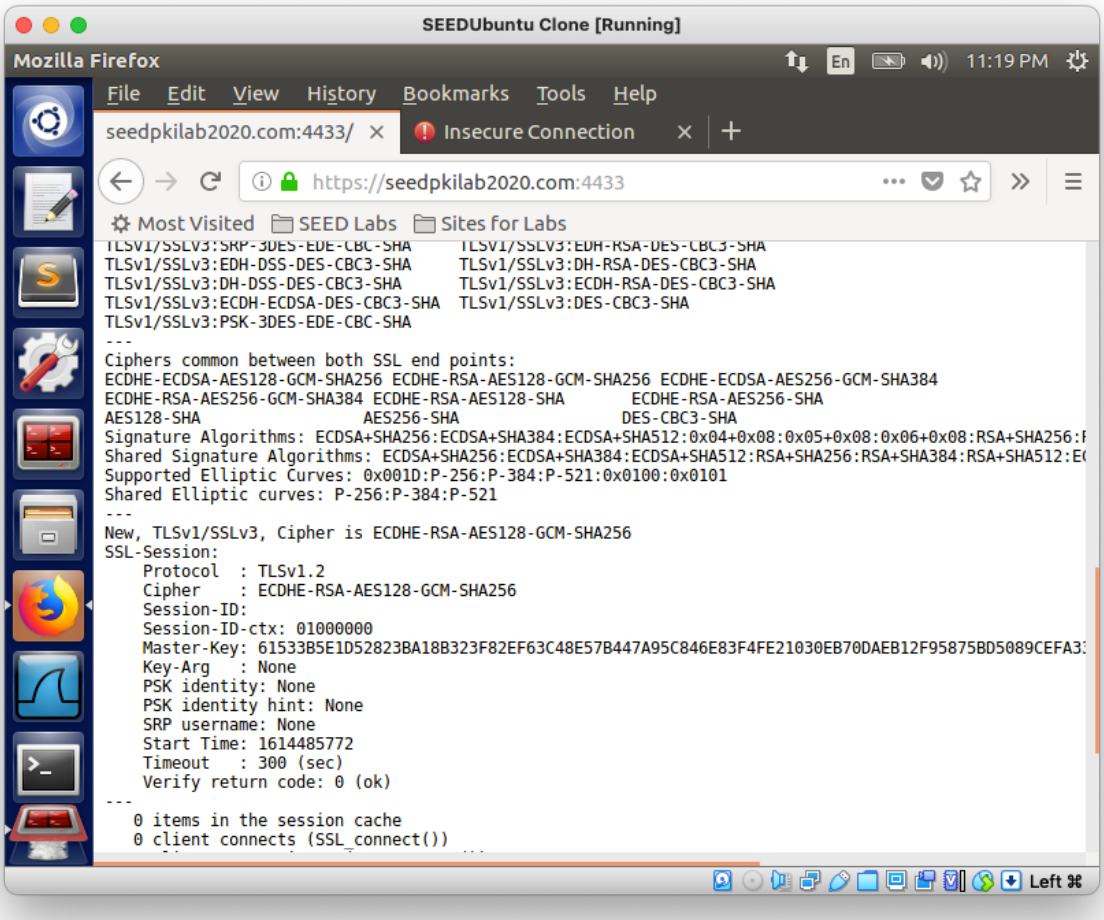

Fig. 3.4.2 Verbose Connection to ***https://seedpkilab2020.com***

However, changing a byte in the server key portion in ***server-cert.pem*** (server.pem) causes openssl to throw errors as shown in Fig. 3.4.3.

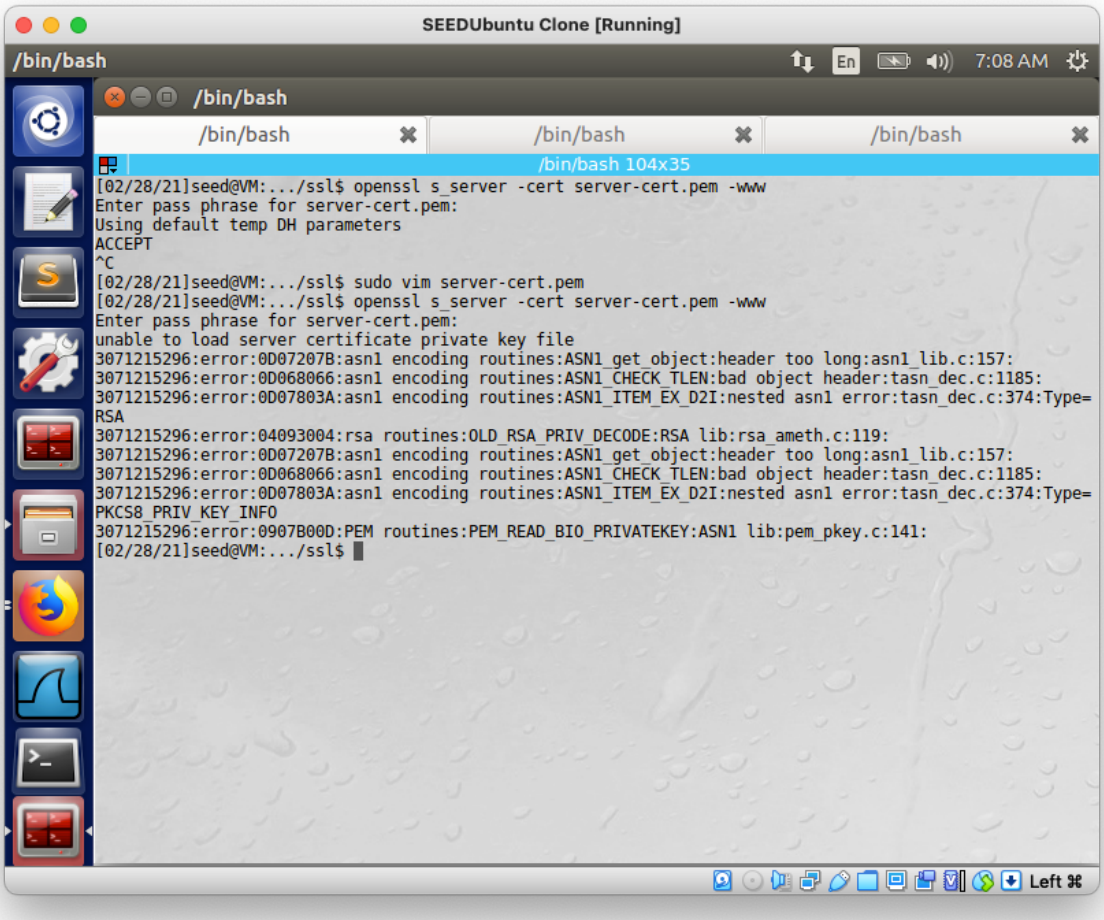

Fig. 3.4.3 Error thrown after modifying a single byte in key inside server.pem

2. Since SEEDPKILab2020.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.

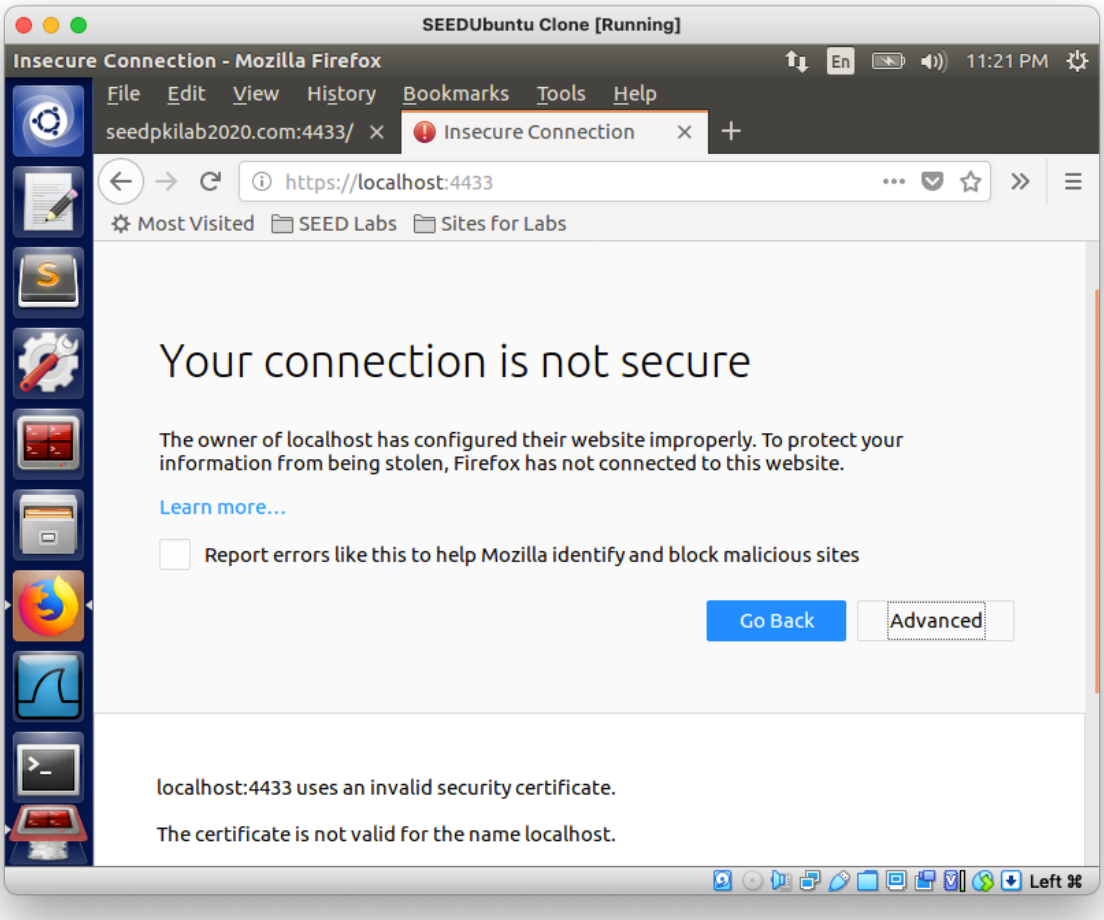

Fig. 3.4.3 Connection to ***https://localhost:4433***

This is expected because our certificate is only for ***seedpkilab2020.com*** (CN) and not ***localhost*** (hostname)***.*** This fails the second validation and hence the browser shows a warning to the user, explicitly asking for permission to trust this website.

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

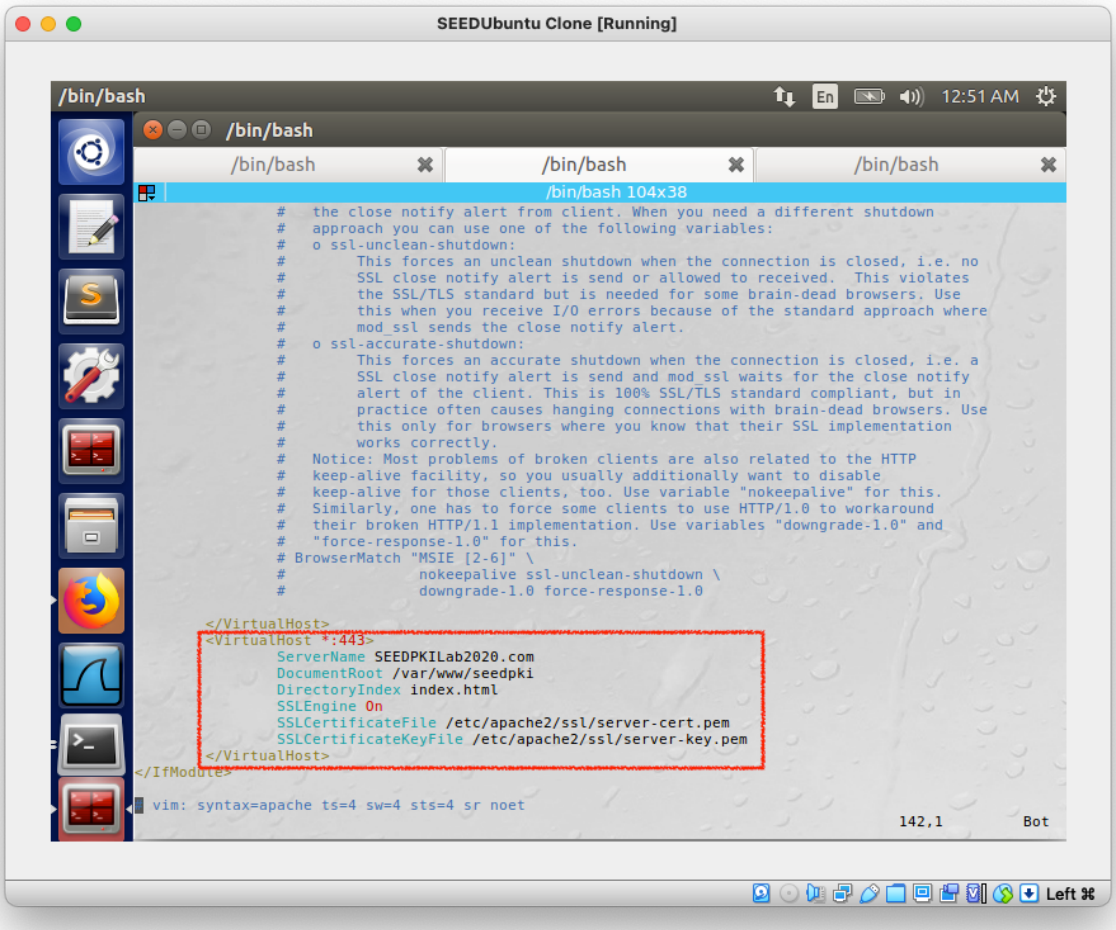

Fig. 4.1 ***/etc/apache2/sites-available/default-ssl.conf***

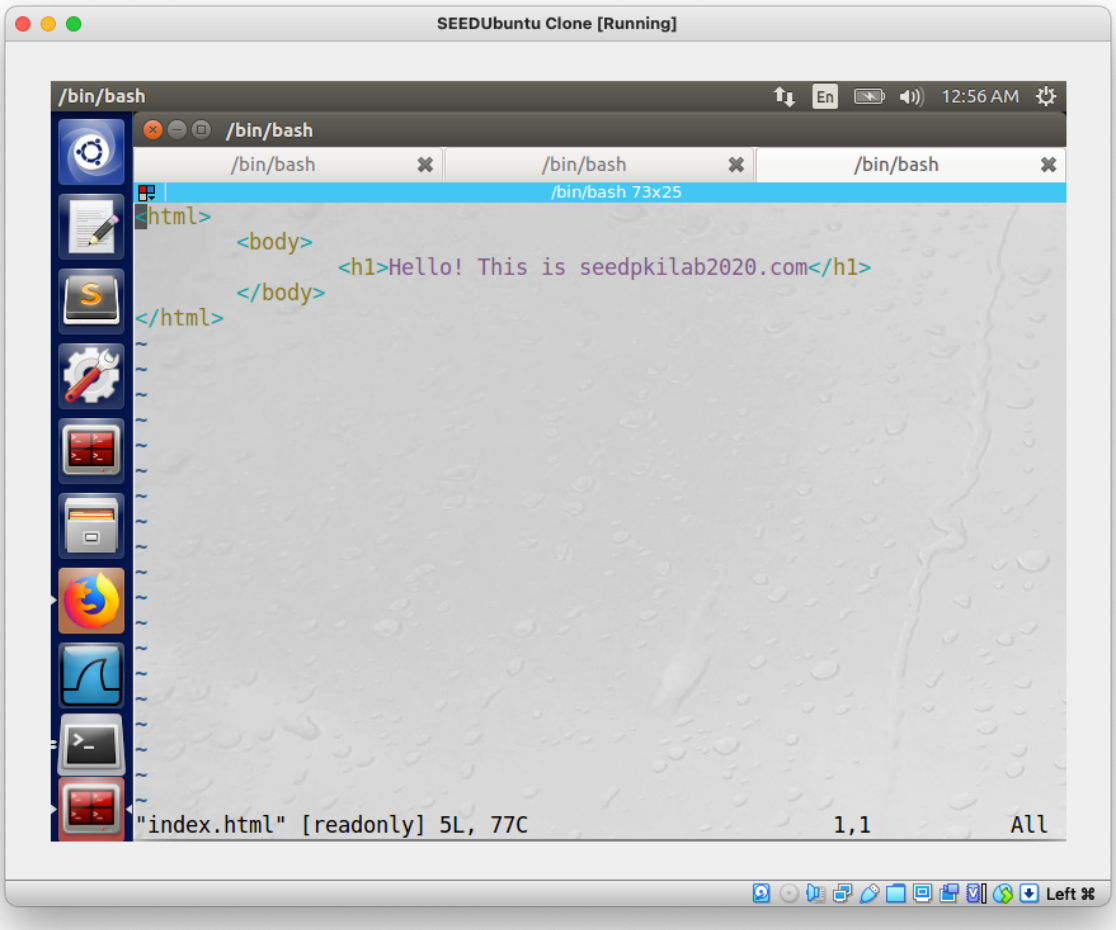

Fig. 4.2 Landing Page of ***httsp://seedpkilab.com***

First, we add ***SEEDPKILab2020.com*** as a VirtualHost in ***/etc/apache2/sites-available/default-ssl.conf*** as seen in Fig. 4.1. Here, we specify the following:

1. All files related to this website to be kept in ***/var/www/seedpki*** (specified by DocumentRoot)
2. ***https://seedpkilab2020.com***'s landing page to be ***index.html*** (DirectoryIndex).
    a. This is a simple html file as shown in Fig. 4.2 which loads when a client visits the website.
3. Apache will find ***seedpkilab2020***'s certificate and key in ***/etc/apache2/ssl/***
    a. Server key : ***server-key.pem***
    b. Server Certificate : ***server-cert.pem***

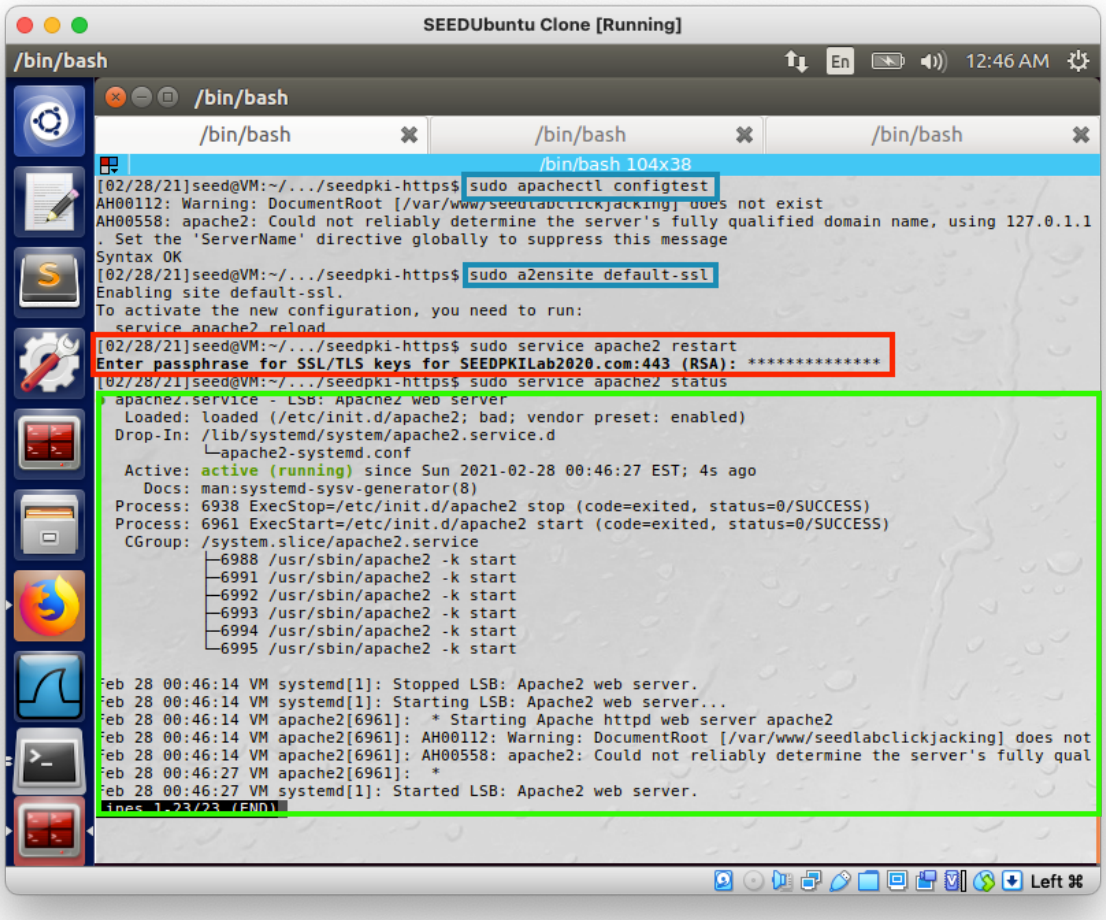

Fig. 4.1 Commands Used to restart Apache Server

Now that we have properly configured the necessary files for ***https://seedpkilab2020.com***. We will follow the following steps to configure ***apache2*** itself. The four steps are shown in Fig. 4.2. We are sure that ***apache2*** is able to find the appropriate files when I was prompted to input the passphrase for SSL keys for the website (highlighted in red box). After restarting ***apache2*** service, we verify that there are no errors by checking its status with the command ***sudo service apache2 status***. We further confirm we have successfully deployed the certificate in Apache-Based HTTPS Website by visiting the site itself as shown in Fig. 4.2 below. As expected, it loads the ***index.html*** that we created.

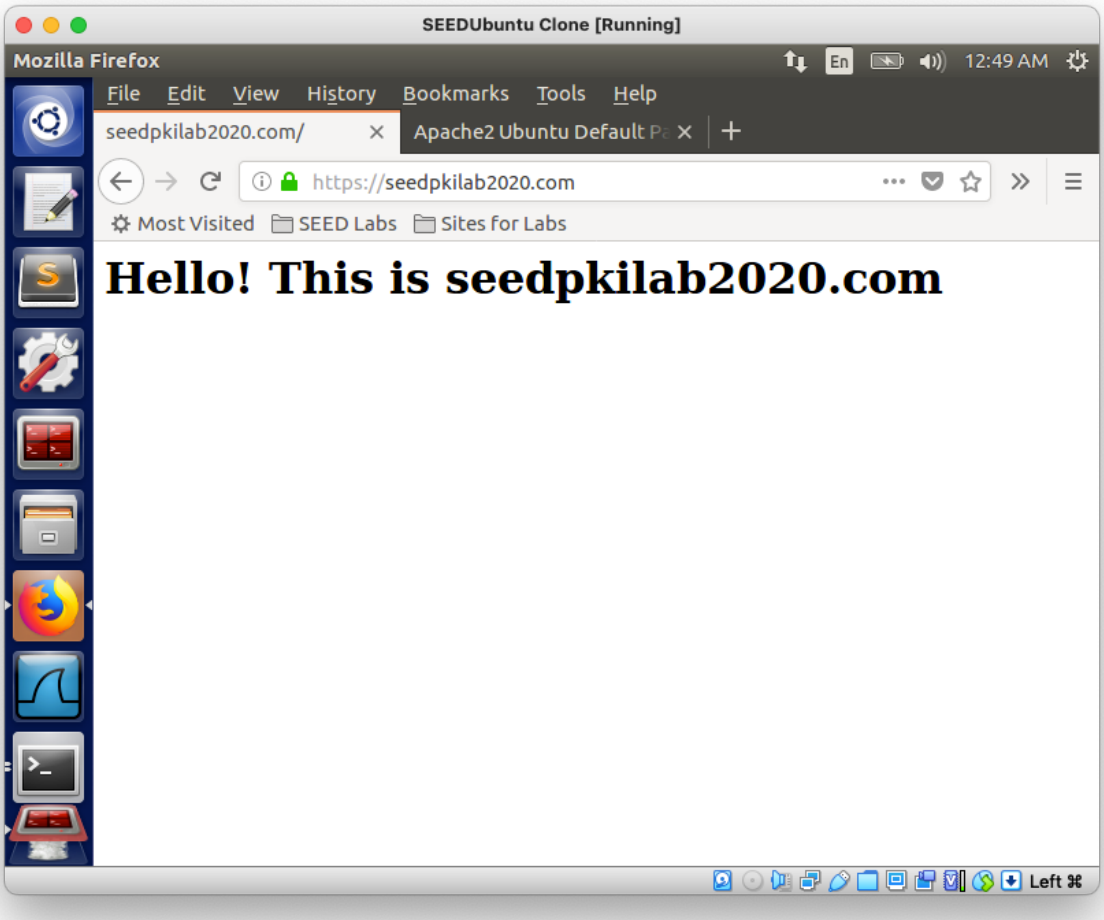

Fig. 4.2 ***index.html*** of seedpkilab2020.com

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

Step 1: Setting up the malicious website.

I will choose to impersonate ***facebook.com*** and as instructed, I first add the corresponding VirtualHost code block into ***/etc/apache2/sites-available/default-ssl.conf*** as shown in Fig 5.1 below.

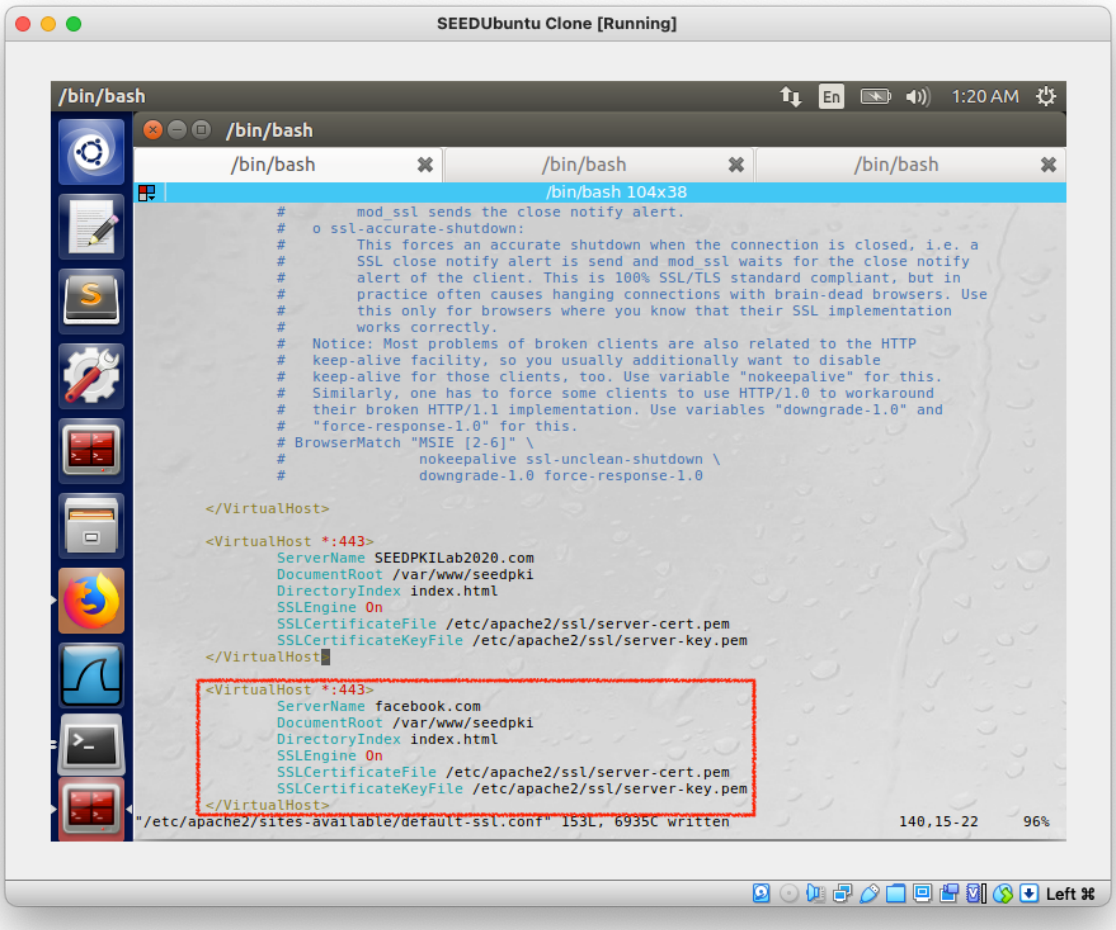

Fig. 5.1 Addition of VirtualHost (facebook.com)

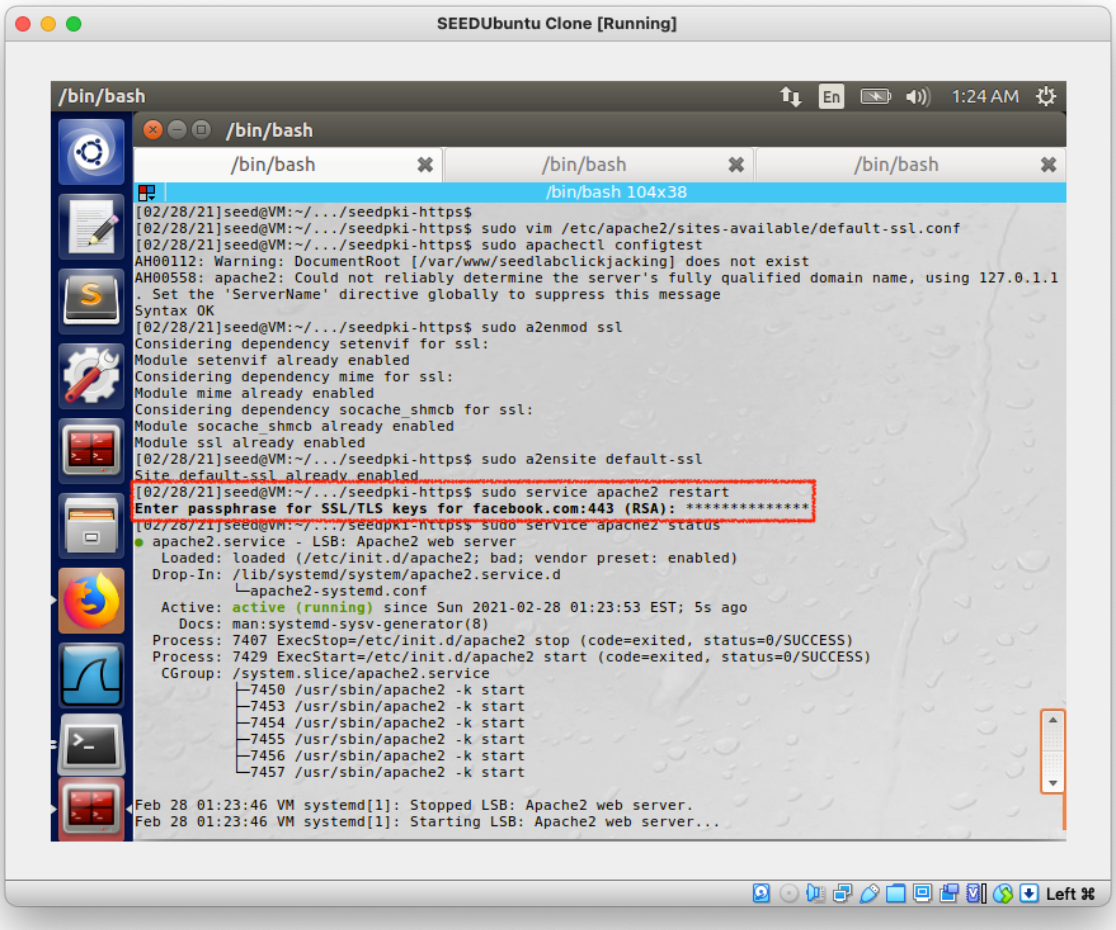

Fig. 5.2 Restarting apache2

Next, we add the IP-hostname mapping to ***/etc/hosts*** with the following line ***127.0.0.1 facebook.com*** as depicted in Fig. 5.5 (last image). Figure 5.3 shows the attempt to visit ***https://facebook.com*** on ***apache2*** on the attacker VM itself and Figure 5.4 shows the attempt to visit ***facebook.com***.

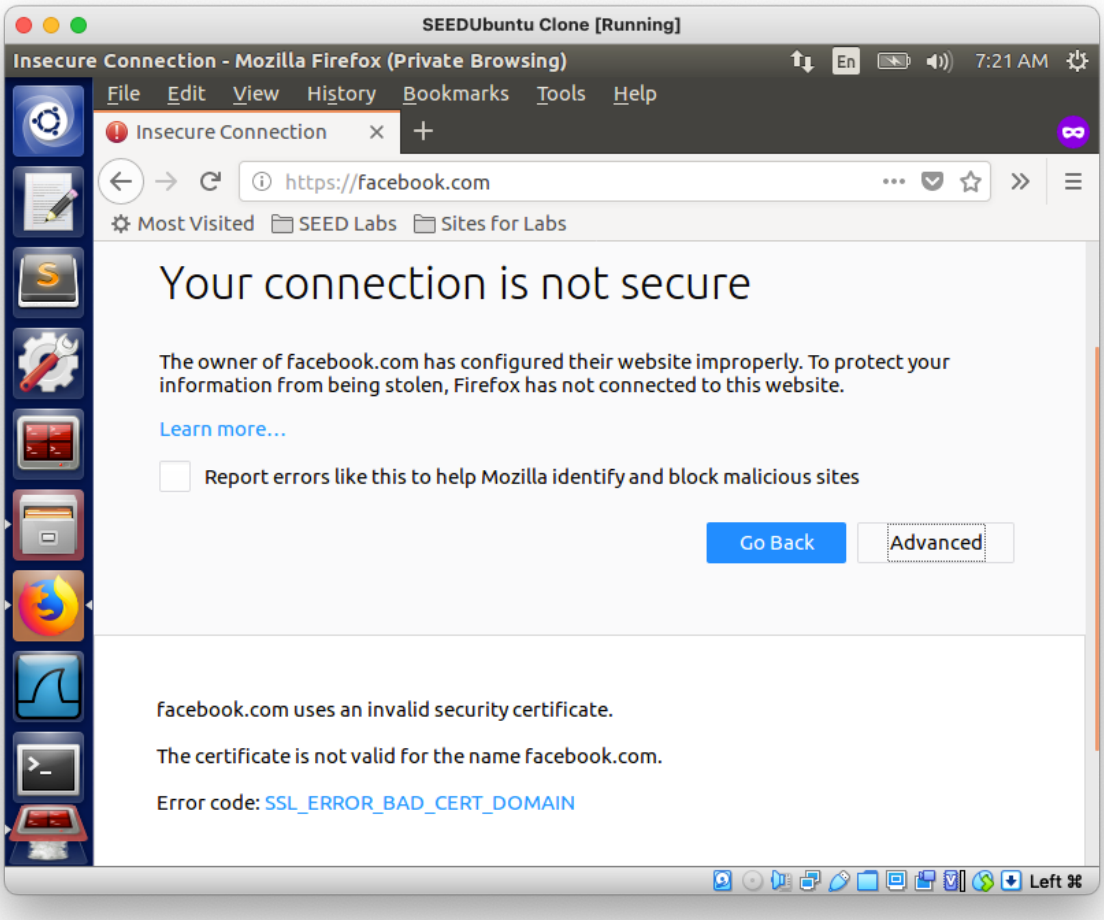

Fig. 5.3 Attempt to visit https://facebook.com

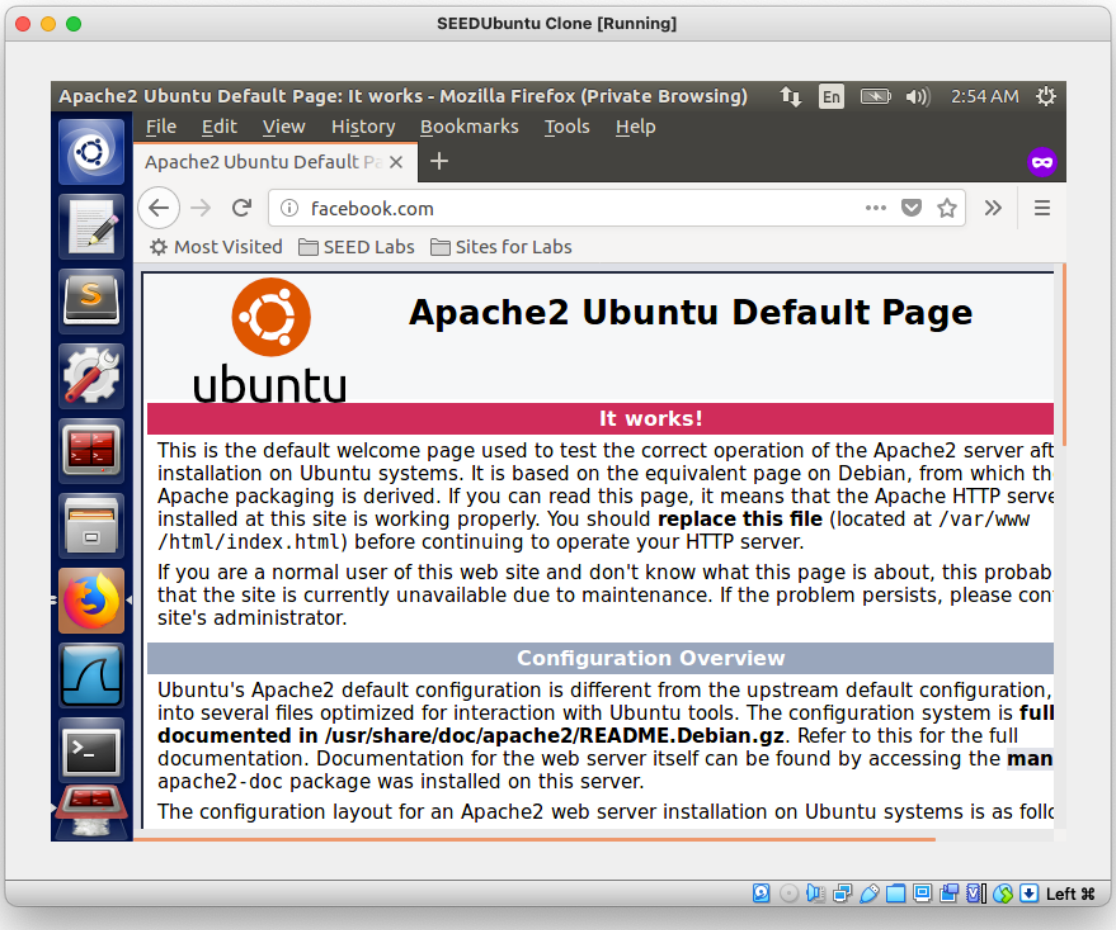

Fig. 5.4 Attempt to visit facebook.com

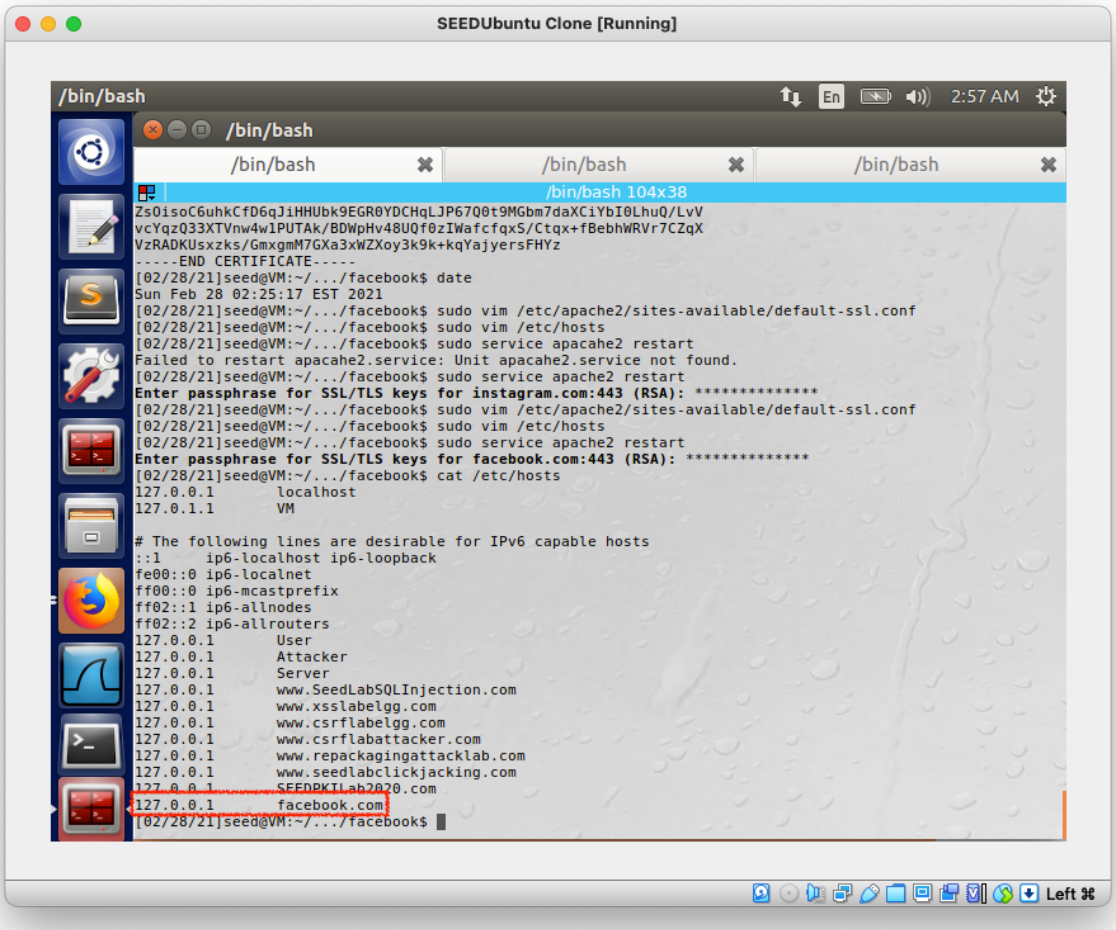

Fig. 5.5 Addition to ***/etc/hosts***

Step 2: Becoming the man in the middle

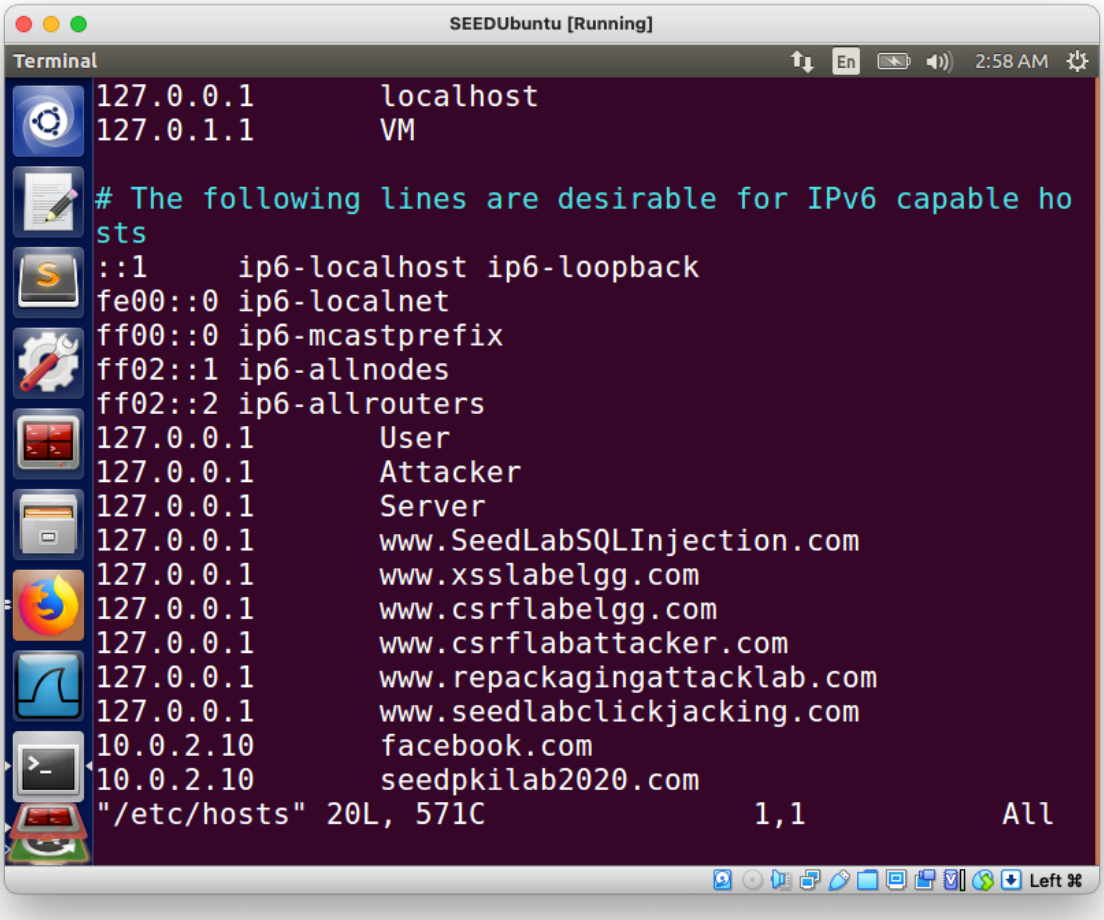

Fig 5.6 ***/etc/hosts*** file on victim VM

On the victim VM, we map the attacker IP (***10.0.2.10***) to the malicious site ***facebook.com***.

Step 3: Browse the target website.

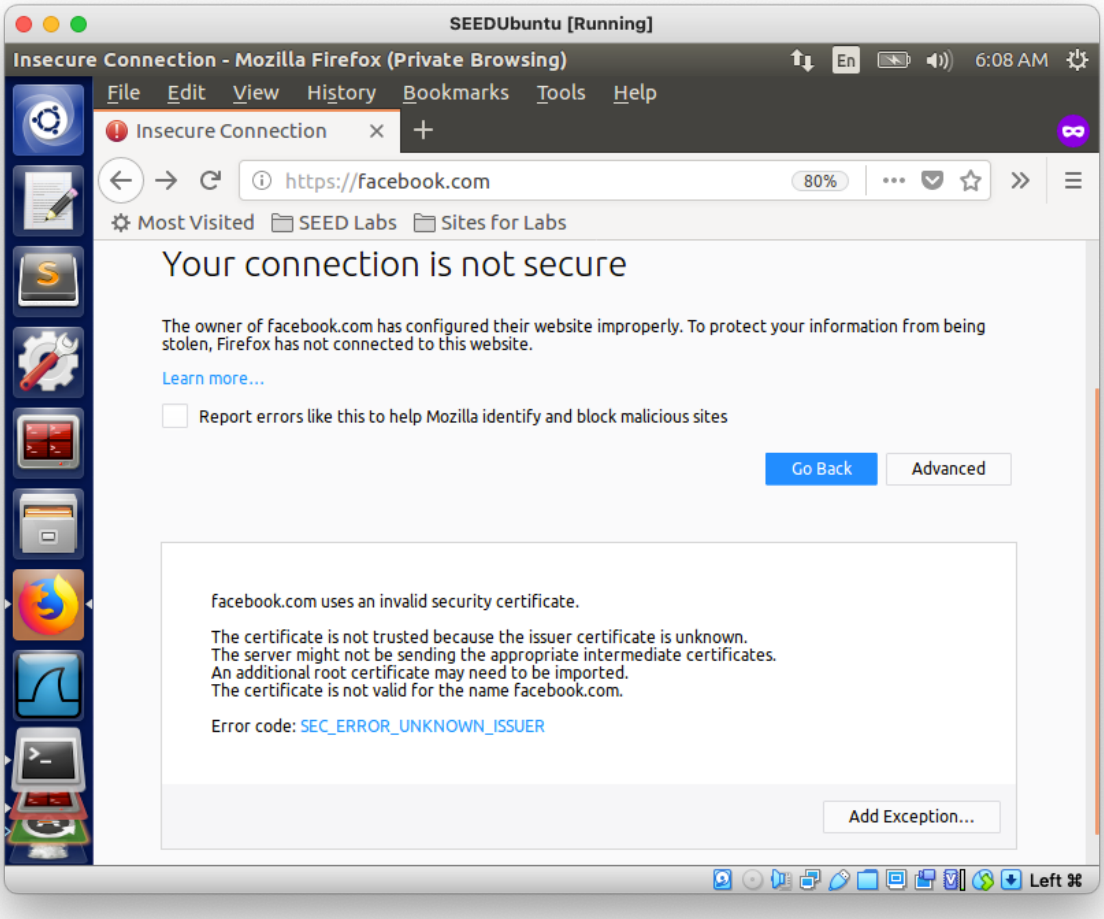

Fig. 5.7 Victim VM's attempt to connect to facebook.com

This is expected because the subject on the Certificate (sent by the attacker) is for seedpkilab2020.com instead of facebook. This explains the error ***The certificate is not valid for the name facebook.com***. The first 3 warnings are due to the attacker certificate's CA not being inside the victim brower's list of trusted CAs.

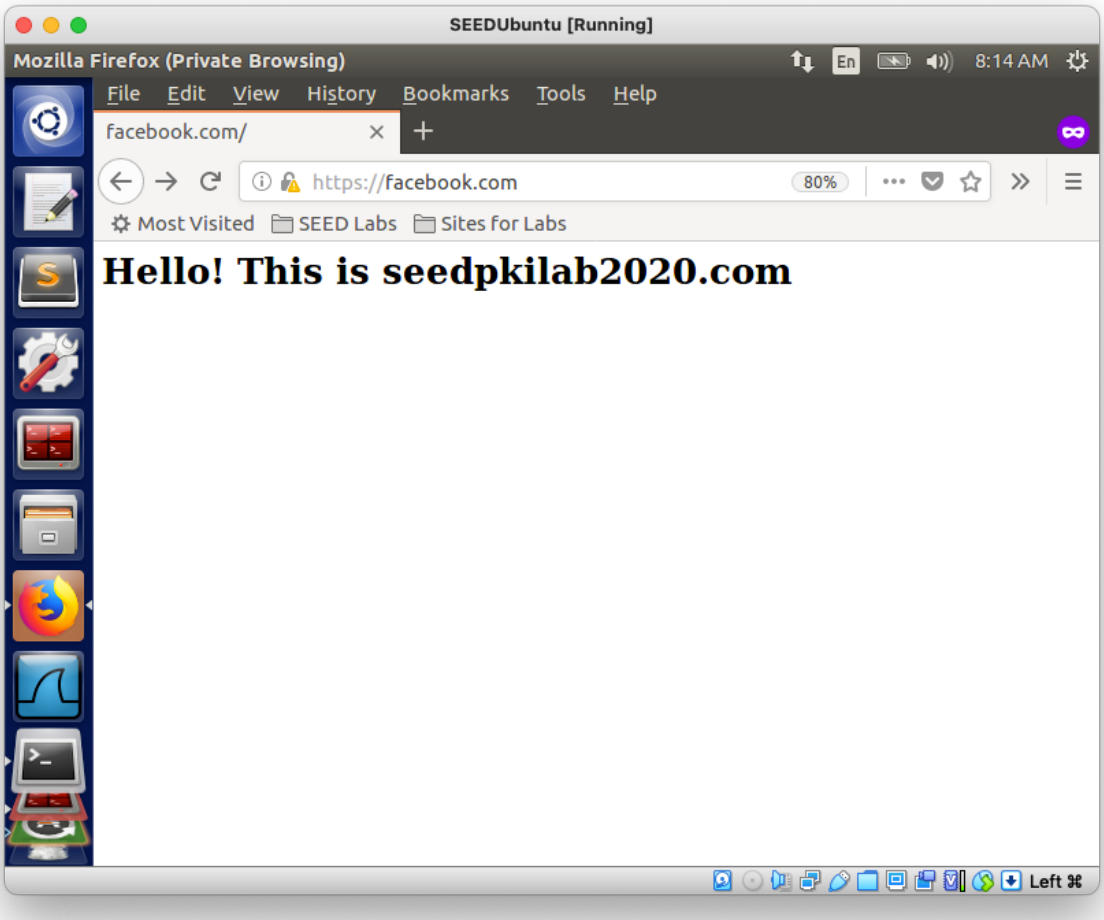

Fig 5.8 Victim ignores warning

Fig 5.8 shows the victim rerouted to the malicious site after ignoring the warning.

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

Let us first understand why the MITM (emulating a DNS attack) in task 5 fails:

1. Common Name in the certificate (seedpkilab2020) is not the same as the malicious site (facebook.com)
2. The CA that signed this malicious site's certificate is not in the list of trusted CAs.

In this task, we assume the root CA has been compromised by an attacker, and its private key is stolen. I will follow the steps detailed in Task 5 with several changes:

1. Generate ***facebook.com*** CSR with the common name explicitly set to ***facebook.com***
    a. Refer to Fig. 6.1's cyan highlighted box
2. Signing ***facebook.com*** CSR with CA key as shown in Fig 6.2

We then visit ***https://facebook.com*** and we are successfully rerouted to the malicious ***facebook.com***. Since we did not change ***index.html*** used in task 5, it is expected for us to see the landing page message as **Hello! This is seedpkilab2020.com** as shown in Fig. 6.4. This works because the CA is in the machine's trusted list of CA, thus suppressing the warning.

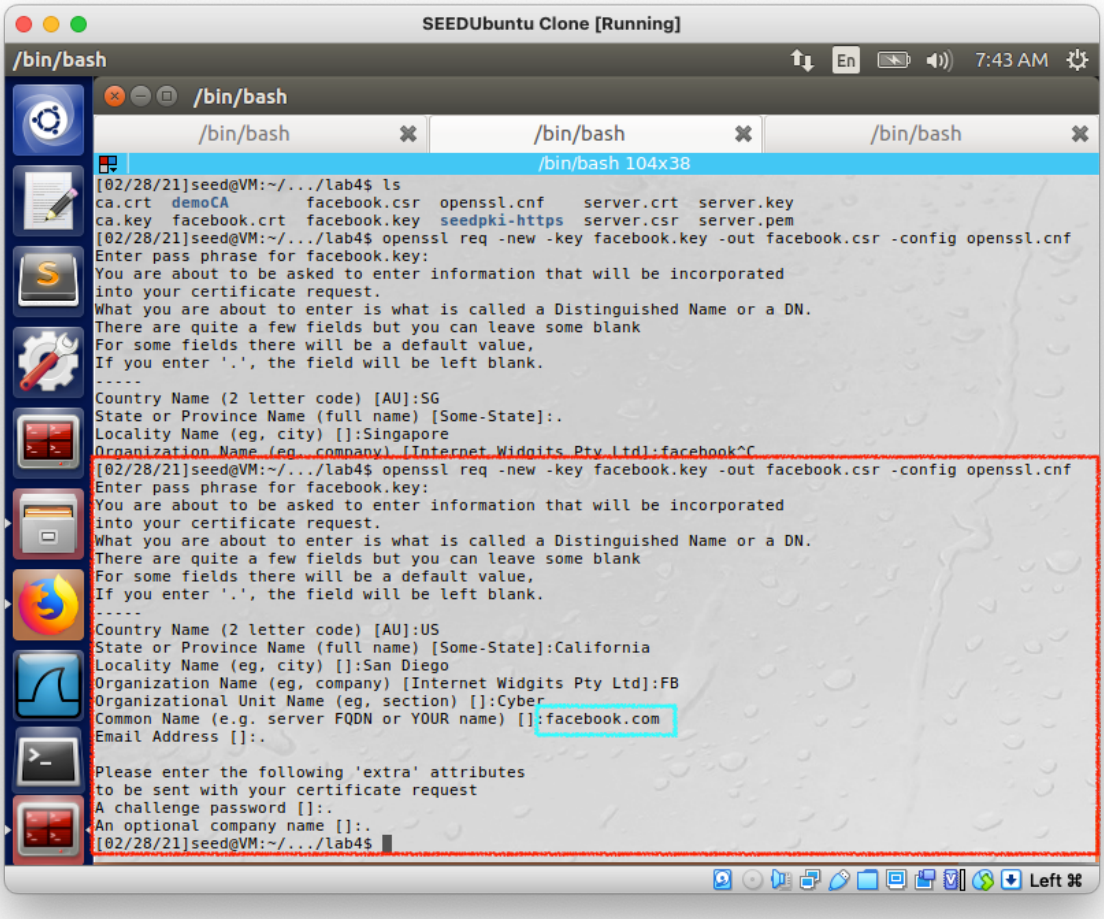

Fig 6.1 Creating *facebook.com* CSR

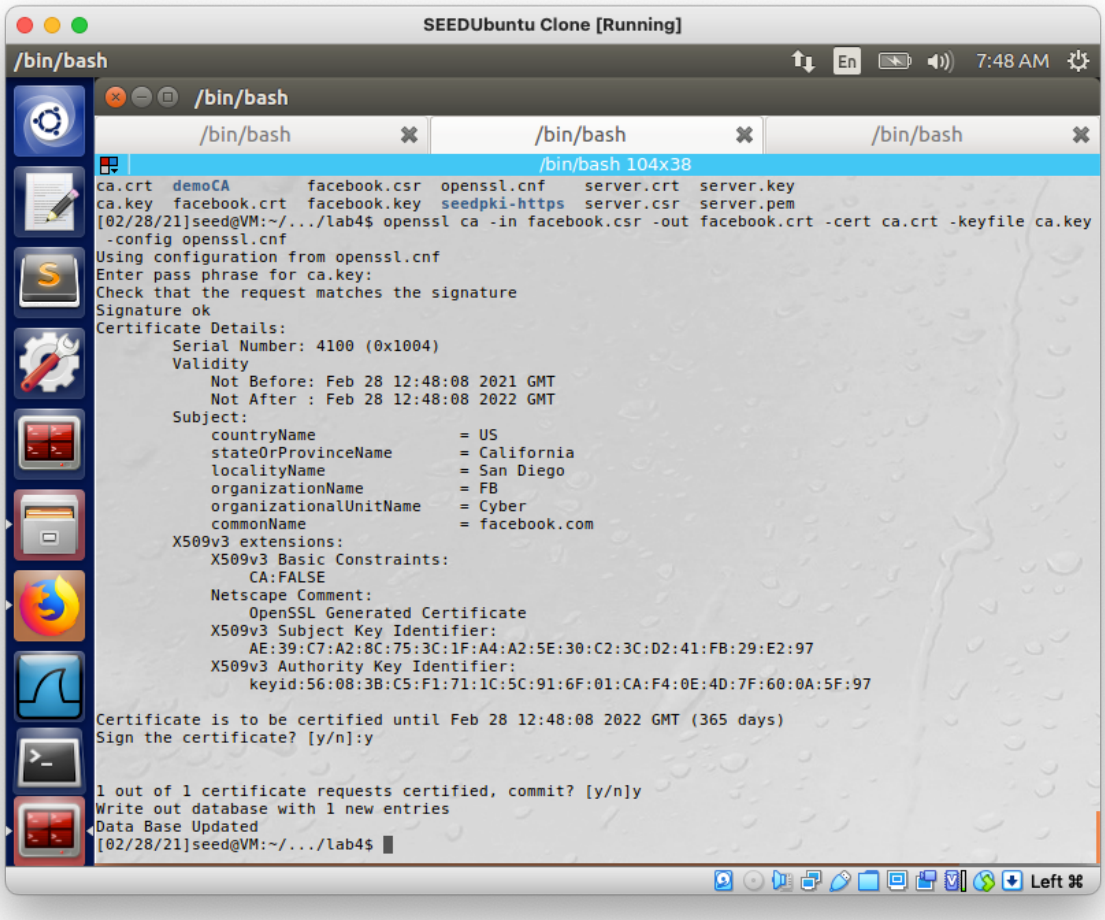

Fig 6.3 Signing of *facebook.com* CSR

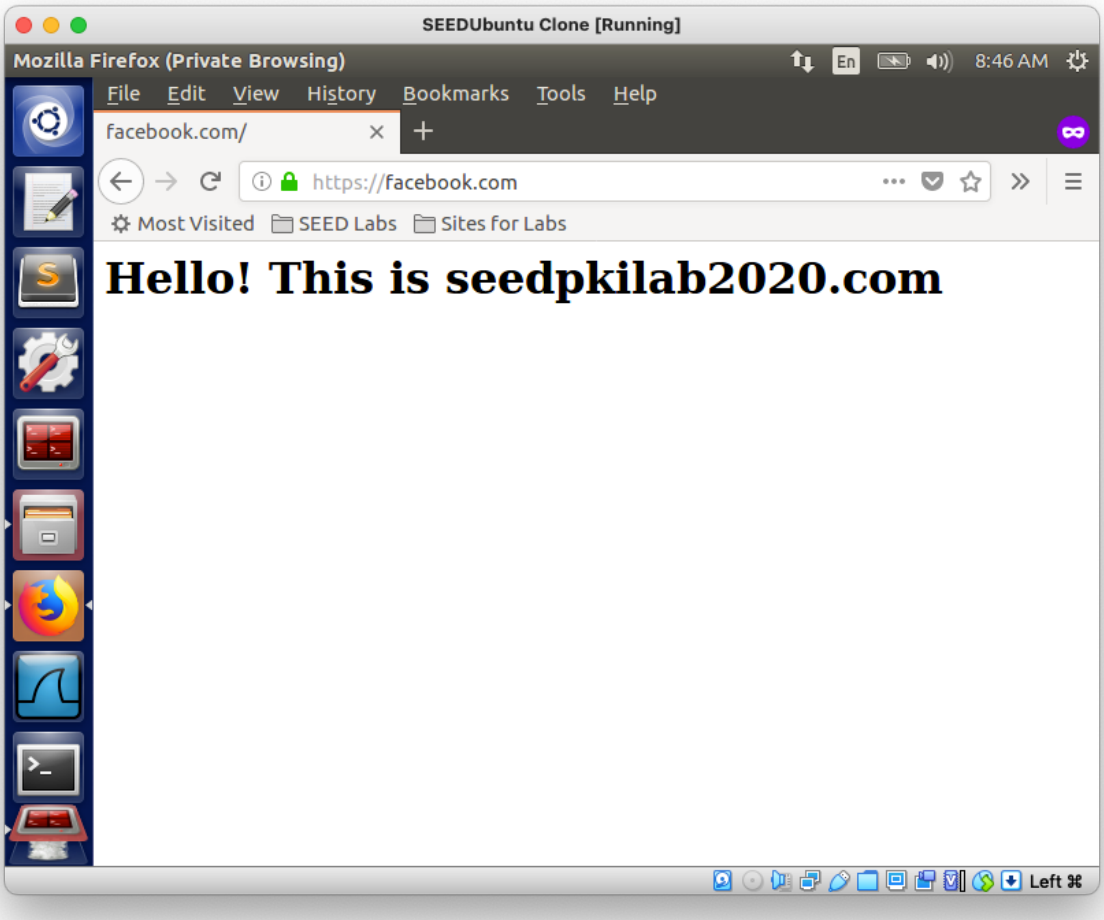

Fig 6.4 Malicious facebook.com