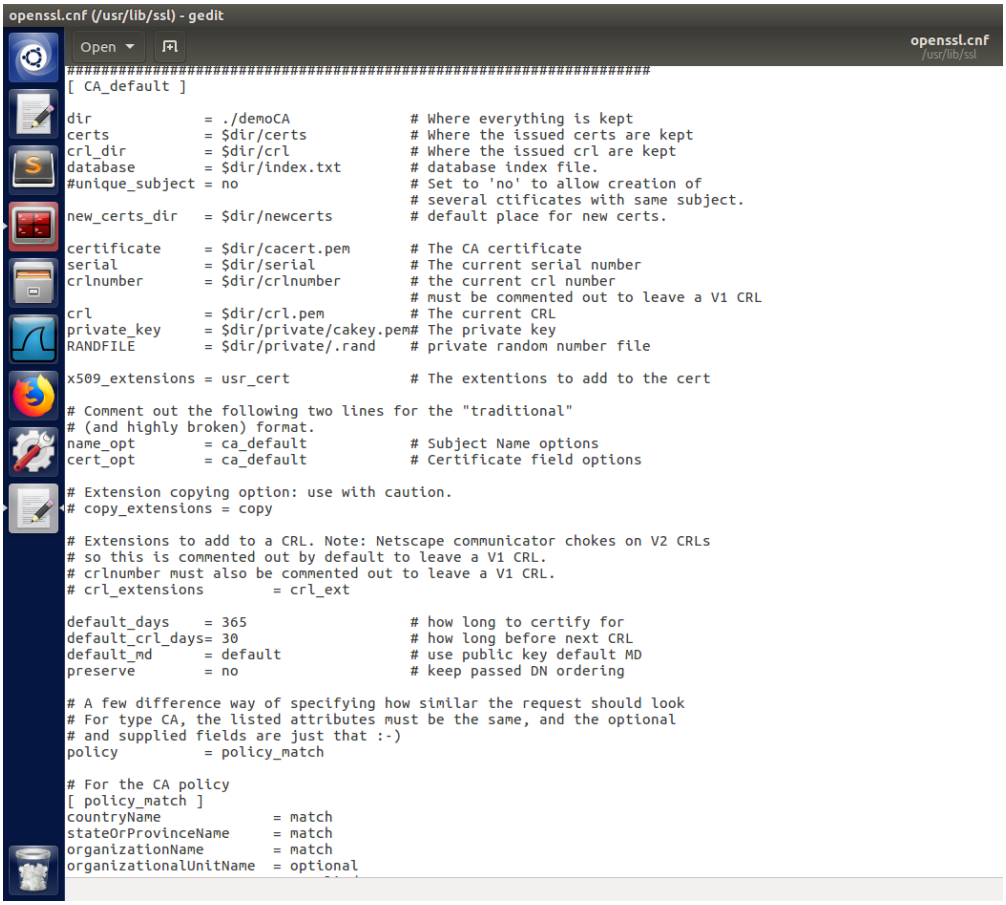
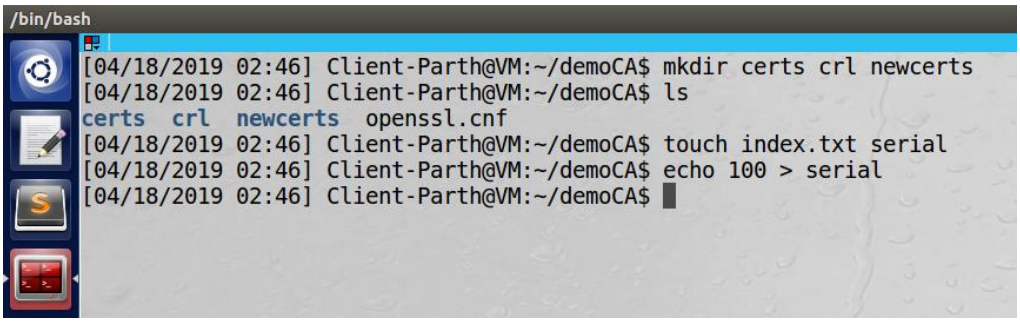
Public-Key Infrastructure (PKI) Lab

Task 1: Becoming a Certificate Authority (CA)

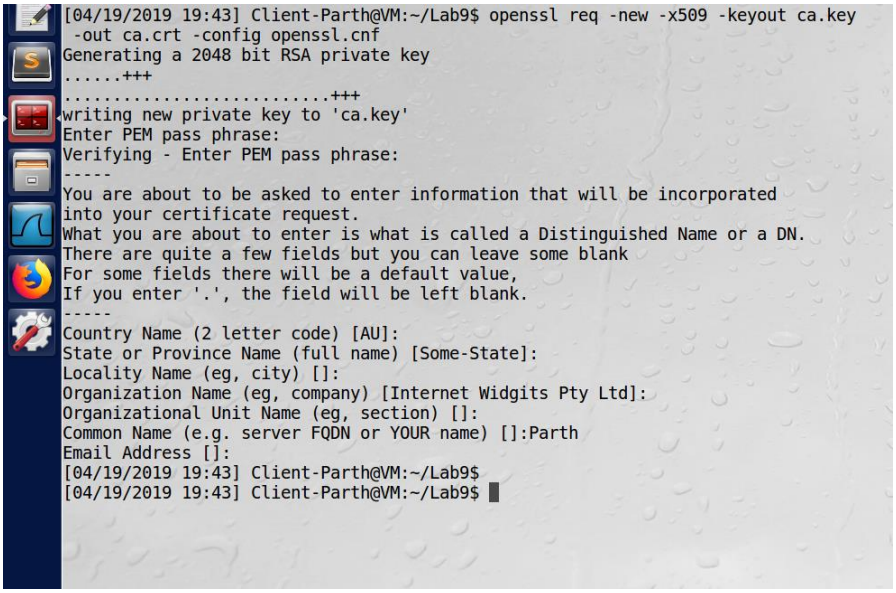
The Configuration File



We can see from the above screenshot that we have successfully set up the configuration file.



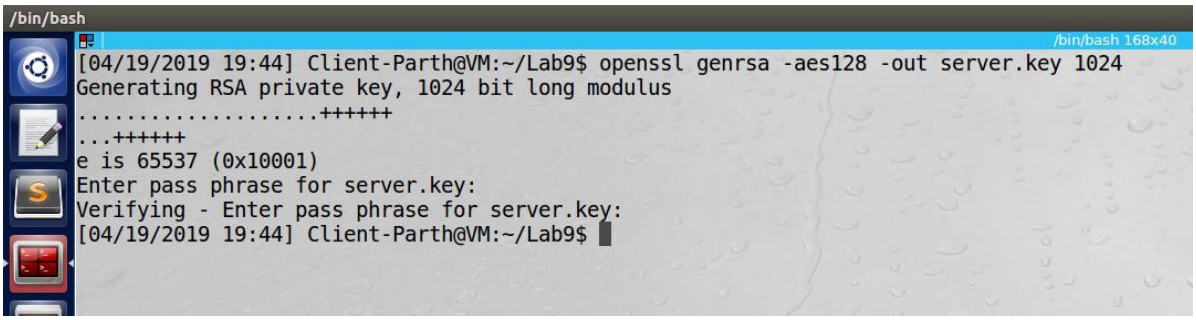
We first created a directory named demoCA and in that directory we created 3 other folder named certs, cerl and newcerts. And 2 files – index.txt and serial. The serial file is the number for the next certificate which we have put to be 1000.



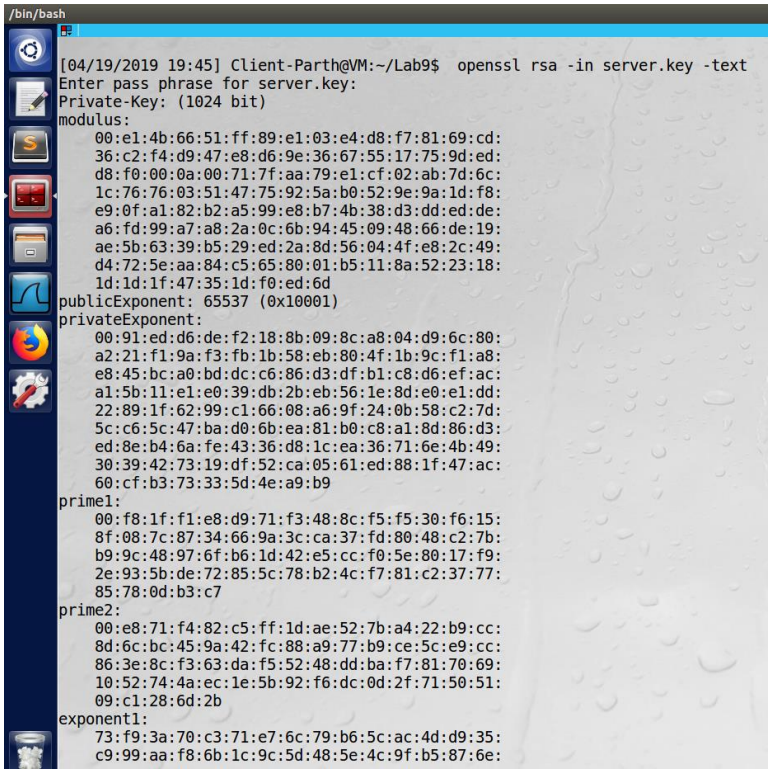
The above command helps us to generate a self signed certificate. In the above command ca.key is the CA's private key, ca.crt is the CA's public key and x509 is the public key certificate.

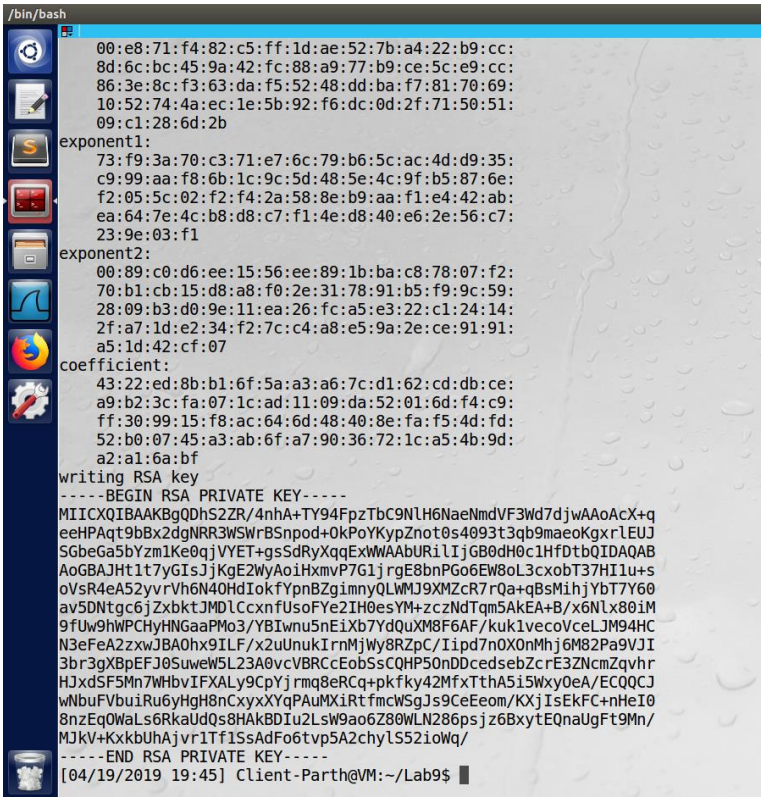
Task 2: Creating a Certificate for SEEDPKILab2018.com

Step 1: Generate public/private key pair.



For a company to have x509 digital certificate, it needs to first generate its own public and private key pair. This is done using the command as seen from the above screenshot. The above command generates a file named server.key. The output of this file is an encoded file. And it contains both private and public key.

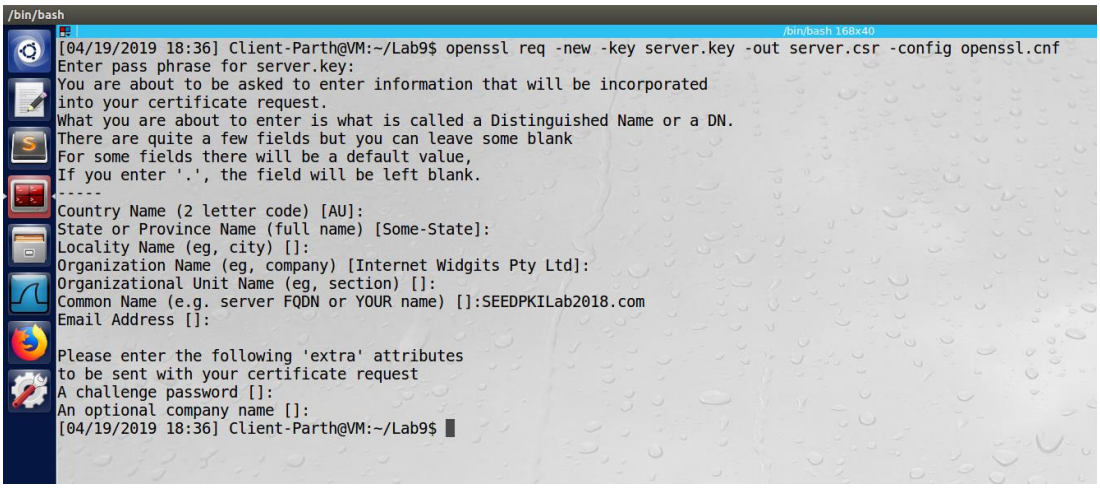




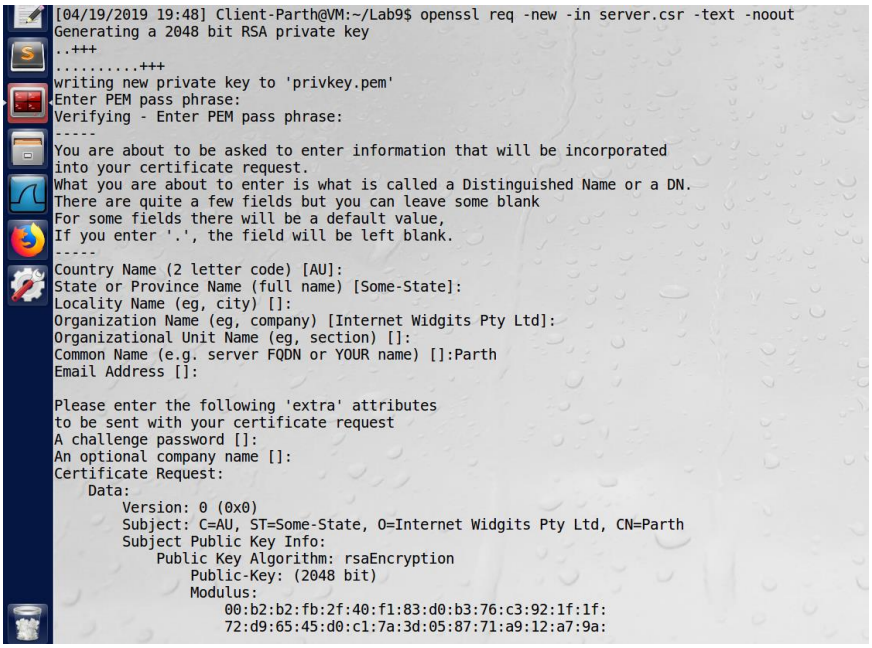
We use the above command to see the content of server.key file which is protected by password provided by users during the key generation. From the above screenshot we can see that the file

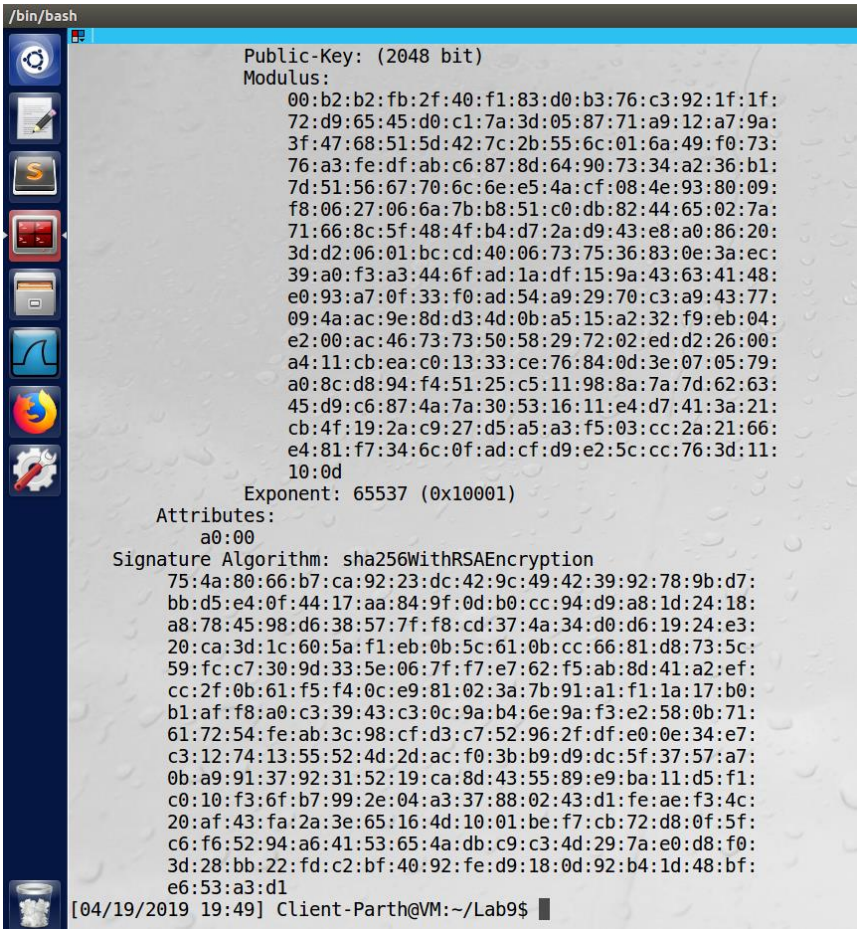
contains both private and public keys. Moreover, it also contains the prime 1 , prime 2, exponent 1, exponent 2 and coefficient that are useful for optimizing the decryption.

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



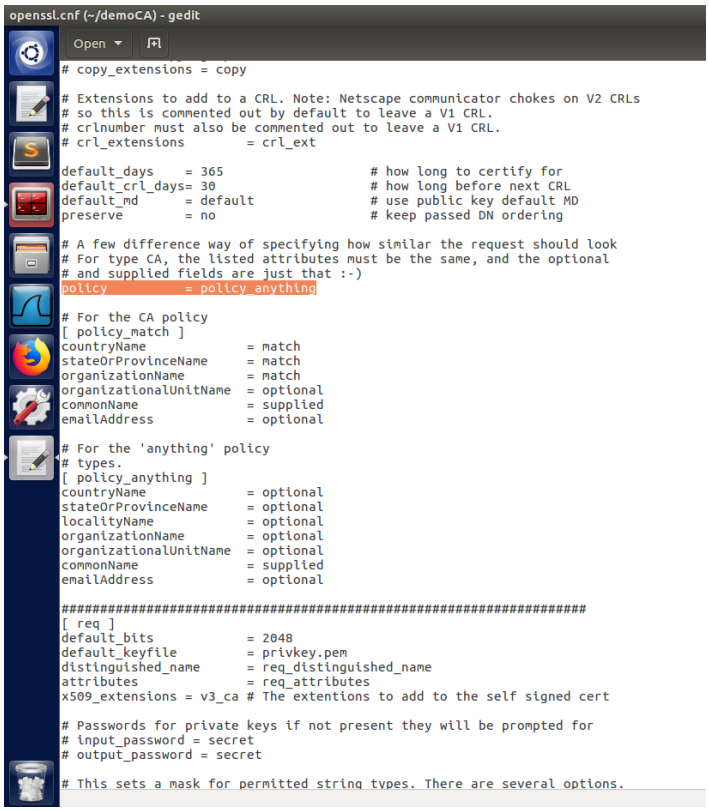
The above command generates a certificate signing request (CSR) that includes the company's public key. The generated signing request is stored in a CSR file server.csr. This is sent to root CA for verification.

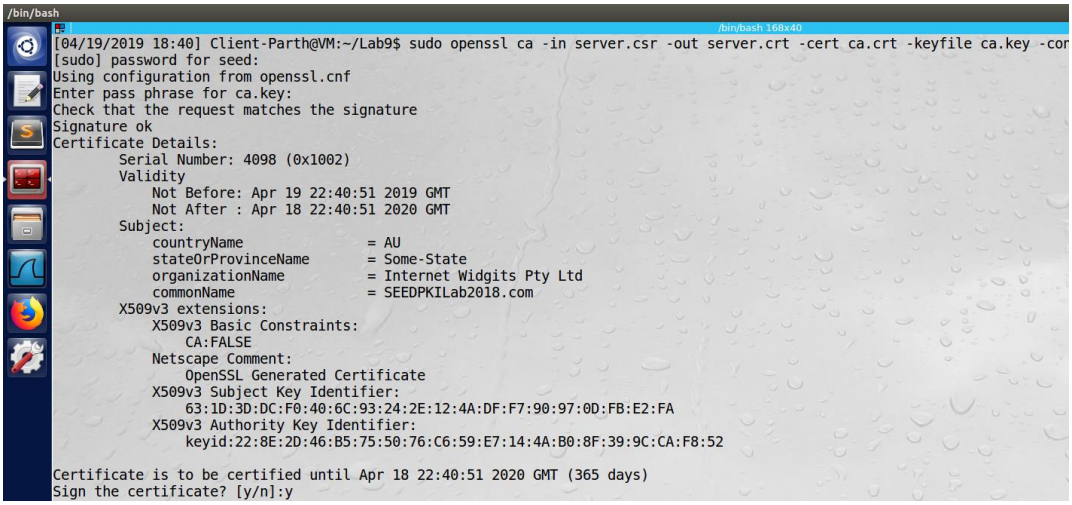




The above command lets us see the content of server.csr file.

Step 3: Generating Certificates.

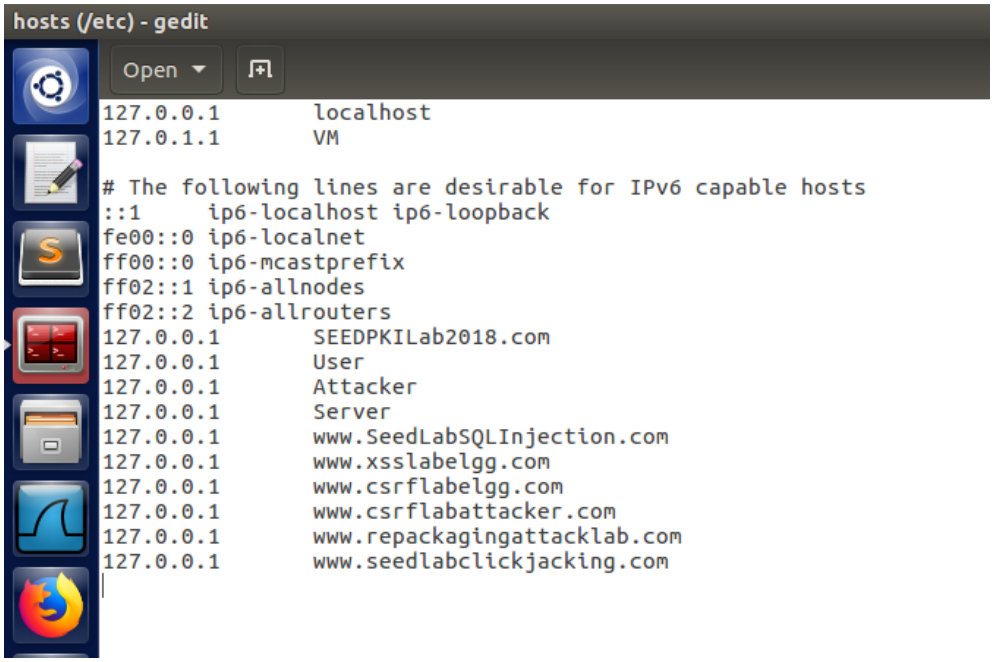




The above command converts the server.csr to server.crt which means that certificate signing request to X509 certificate with the help of ca.crt and ca.key which is the root CA's certificate and private key.

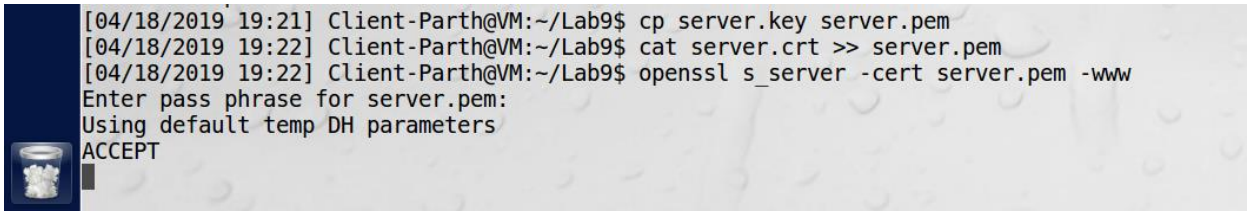
Task 3: Deploying Certificate in an HTTPS Web Server

Step 1: Configuring DNS



In this step we add the SEEDPKILab2018.com to map it to our local host.

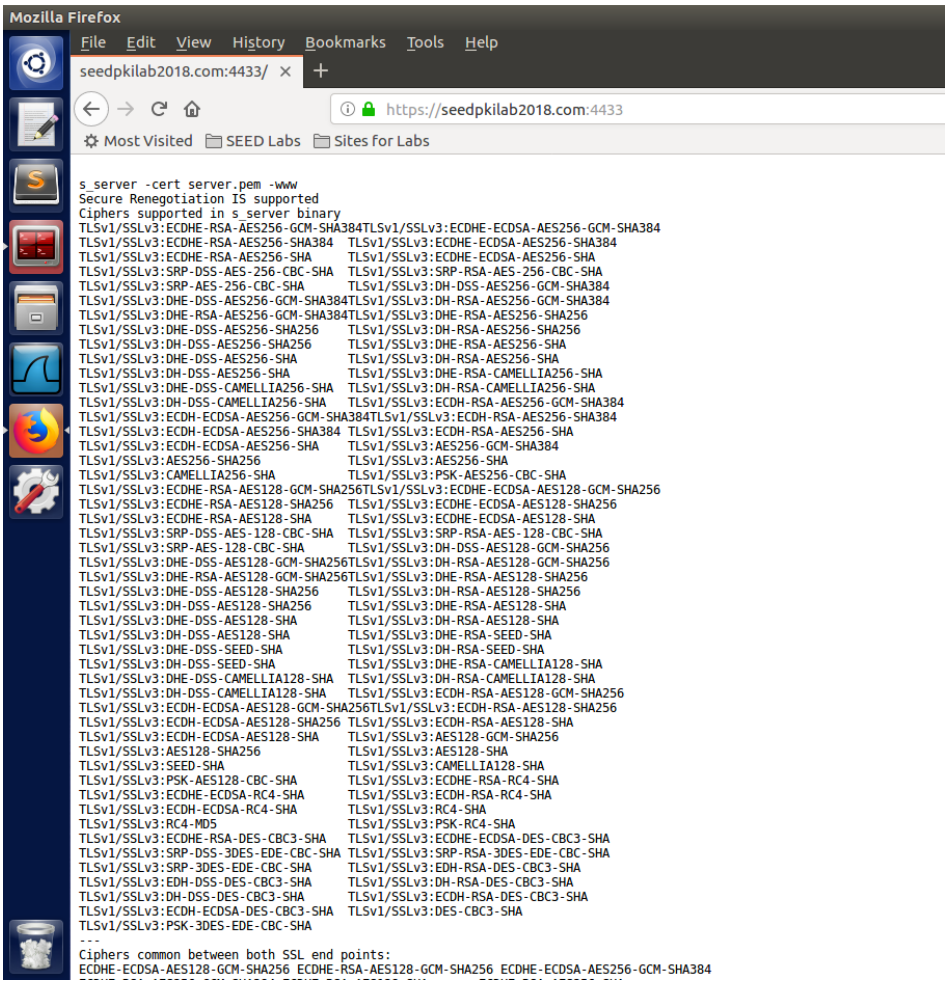
Step 2: Configuring the web server



After the certificate is received, the SEEDPKILab2018.com can host that in its HTTPS web site. Therefore, in this step we first combine the secret key and certificate into one file named server.pem. And then start the server.

Step 3: Getting the browser to accept our CA certificate

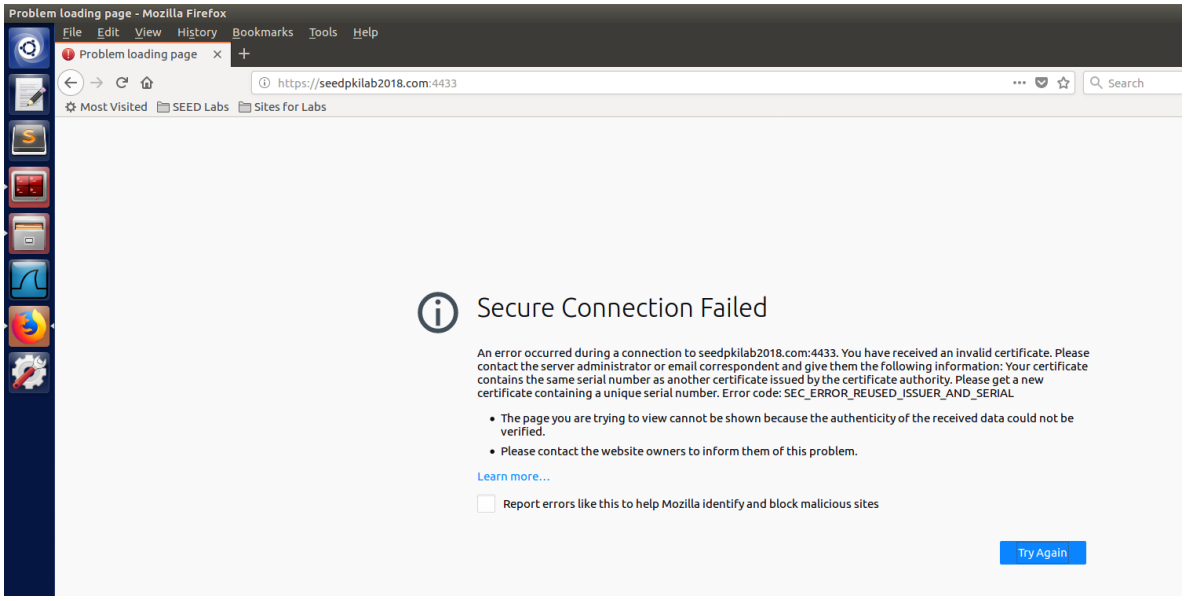
After adding the ca.crt to our firefox browser and then selecting 'trust this CA to identify this web site.' We can see that it is added in the list of the accepted certificate. After which when we visit the SEEDPKILab2018.com we can see that it is considered to be secure since, as per firefox the site now has valid certificate.



Step 4. Testing our HTTPS website

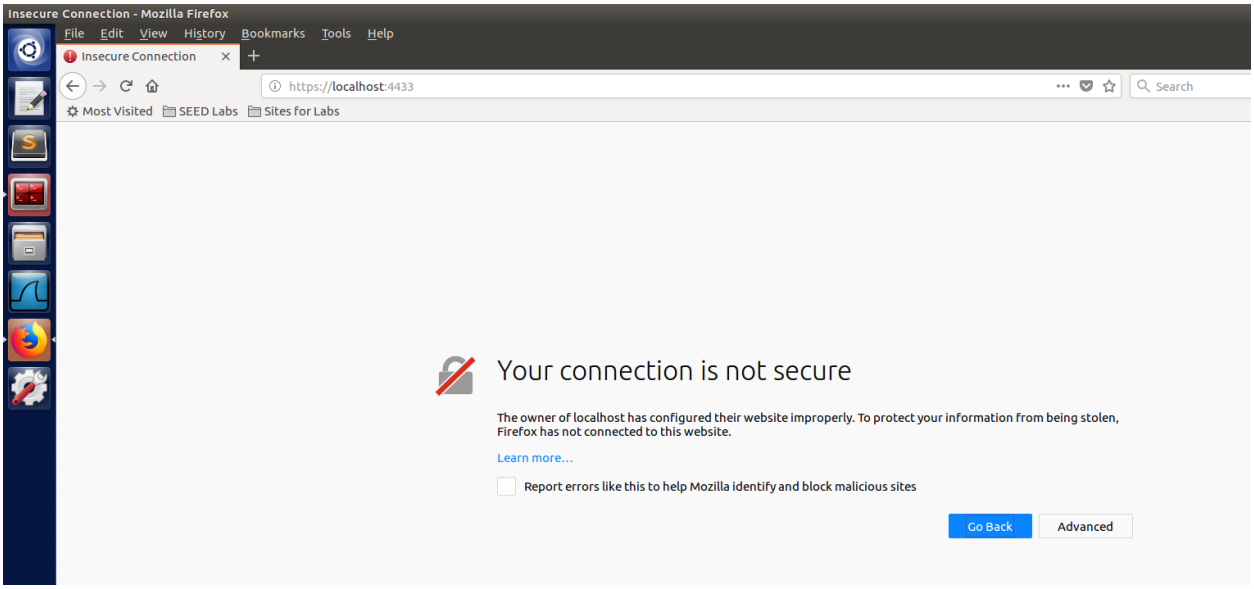
1.

In this step we first corrupted the server.pem file and then restarted the server. After which we reloaded the URL. We can see from the screenshot below that the browser is showing secure Connection failed that is because it now as an invalid certificate.



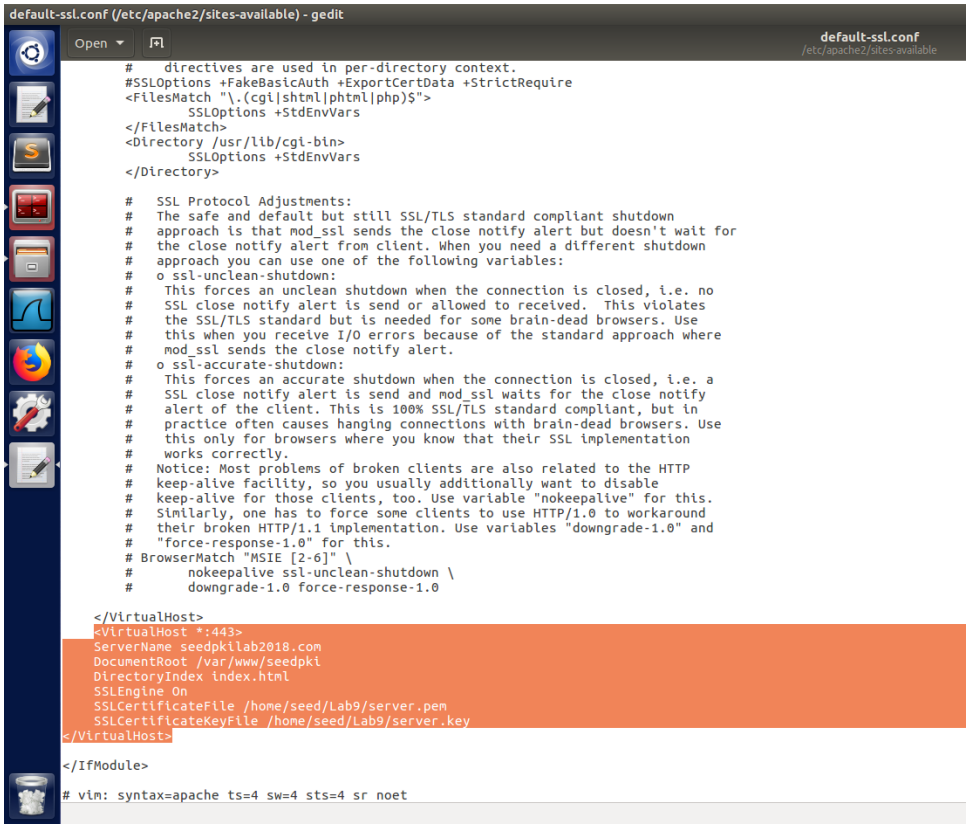
After this step we restore the original server.pem file.

2.

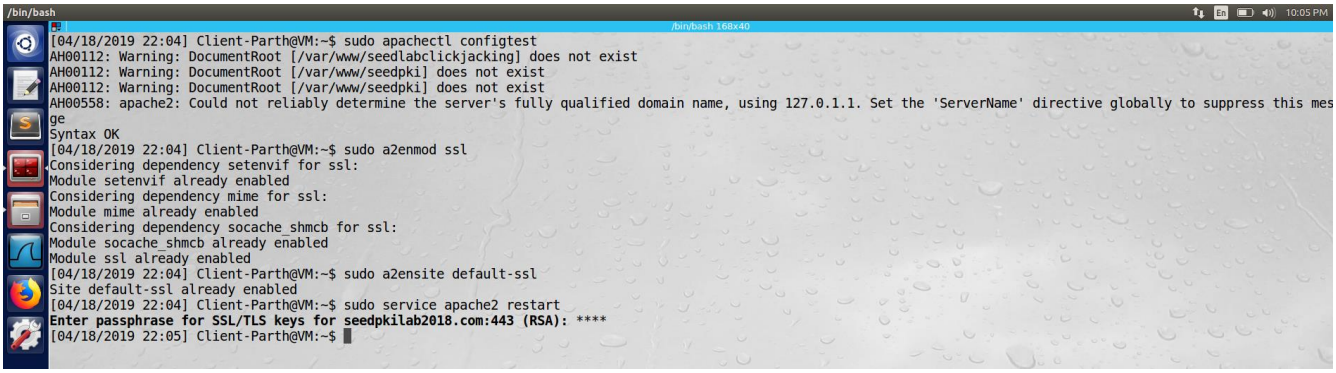


In this step since, the SEEDPKILab2018.com is pointed to localhost. Therefore, we try to access the same website with the URL as localhost. But from the above screenshot we can see that we are not able to access the website that is because the certificate has common name as SEEDPKILab2018.com and not localhost and therefore, the browser shows as the connection is not secure.

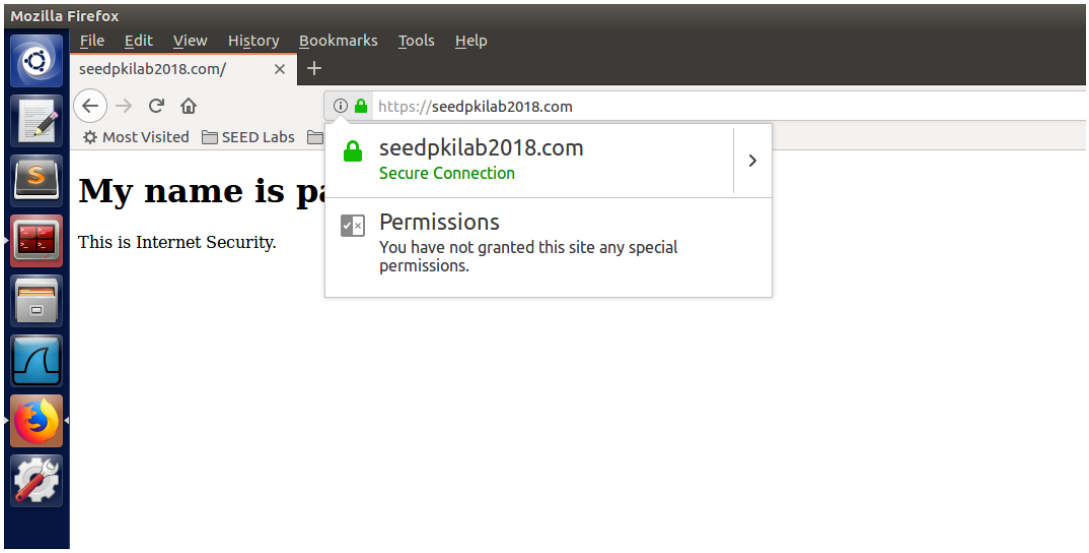
Task 4: Deploying Certificate in an Apache-Based HTTPS Website



The above setup tells Apache about hosting the seedpkilab2018.com. With the index page that we created inside the /var/www/seedpki. The path that is mentioned for SSLCertificatefile and SSLCertificateKeyFile is to tell the apache about the location of servers certificate and private key.

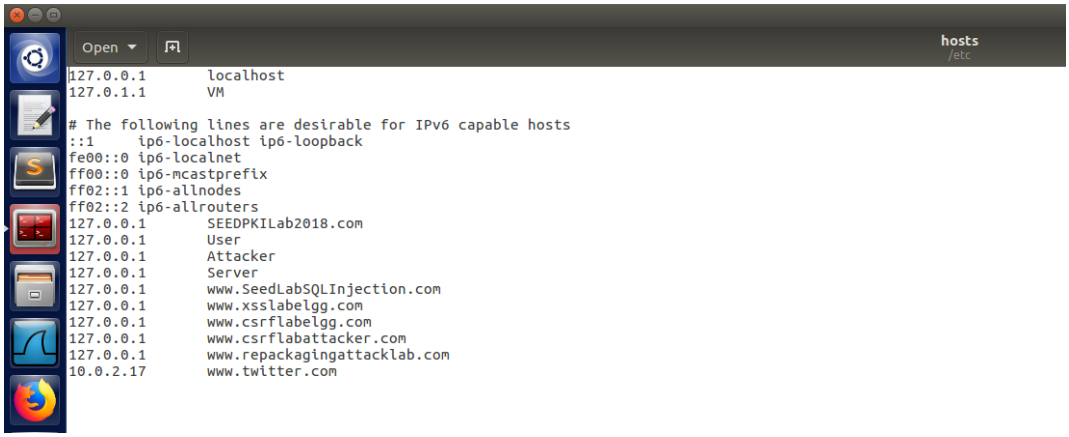


After the above configuration we restart the server to enable the SSL. Now when we visit the website all the traffic between the server and the browser will be encrypted

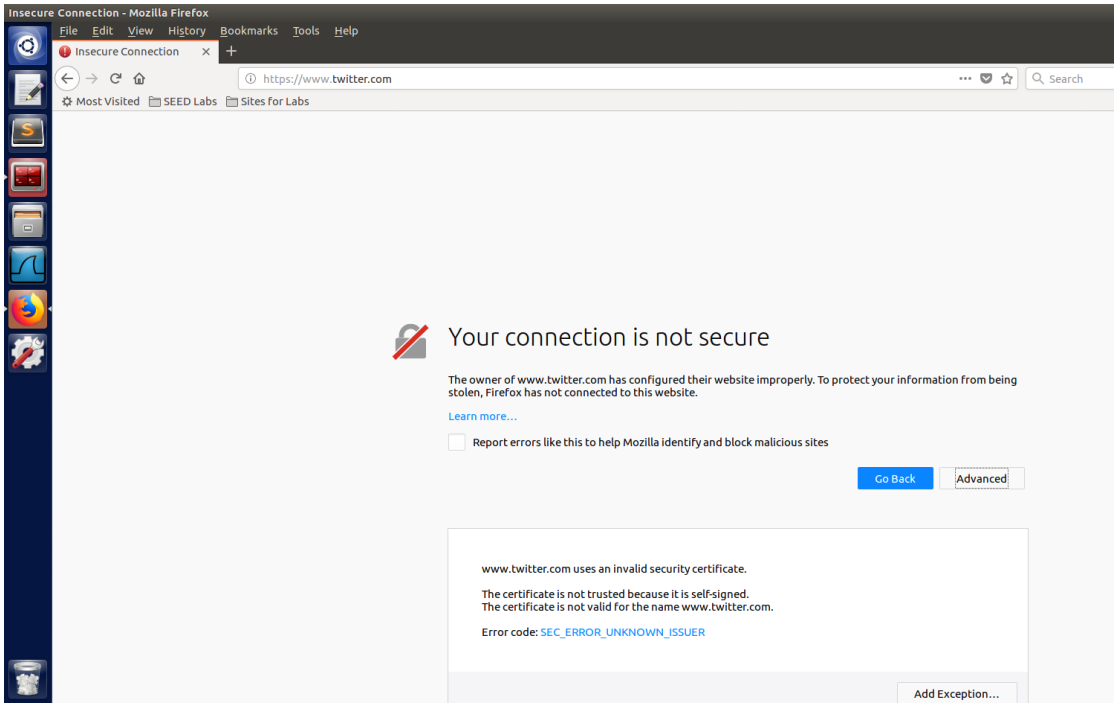


From the above screenshot we can see that when we visit the website seedpkilab2018.com the greenlock states that the connection is secure which means our setup is successful.

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

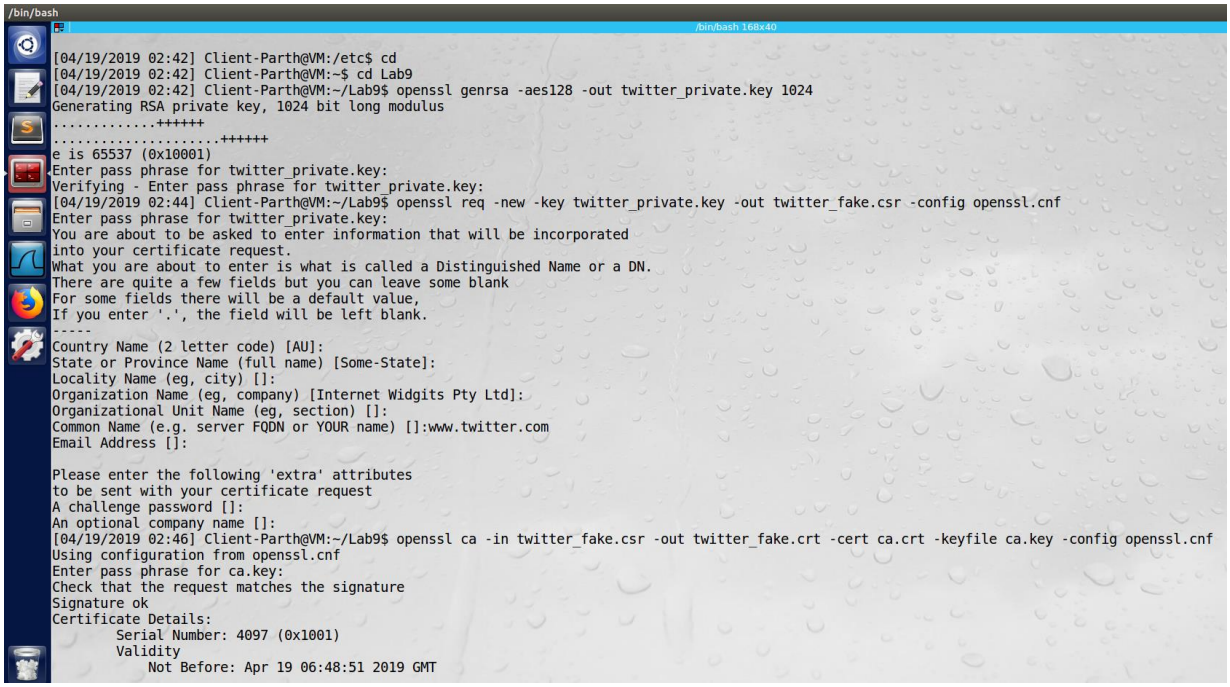


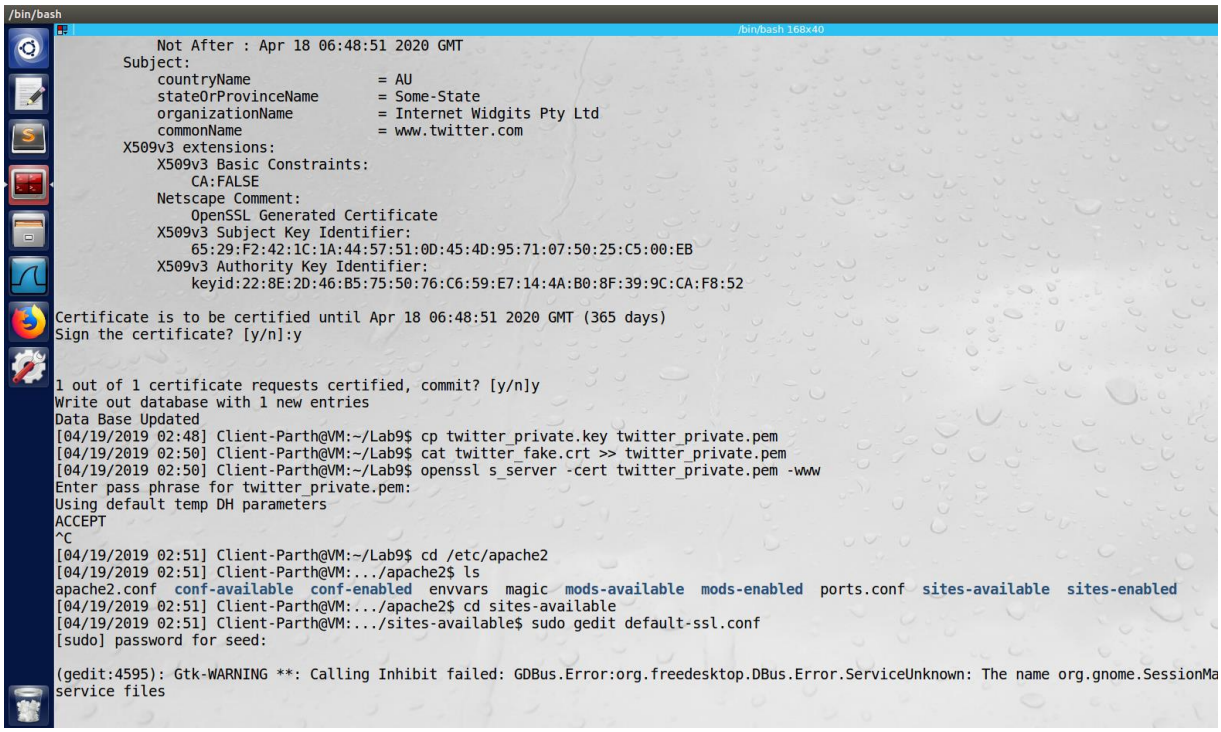
In the /etc/hosts file we put the ip address of the server against www.twitter.com as seen above

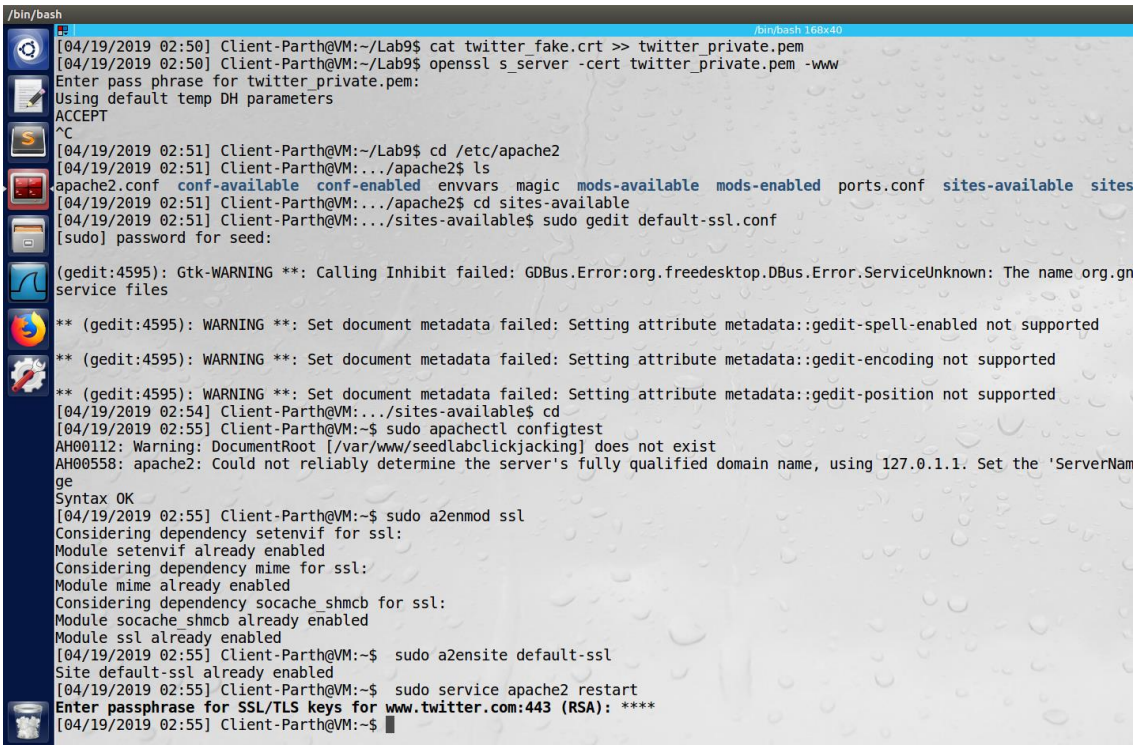


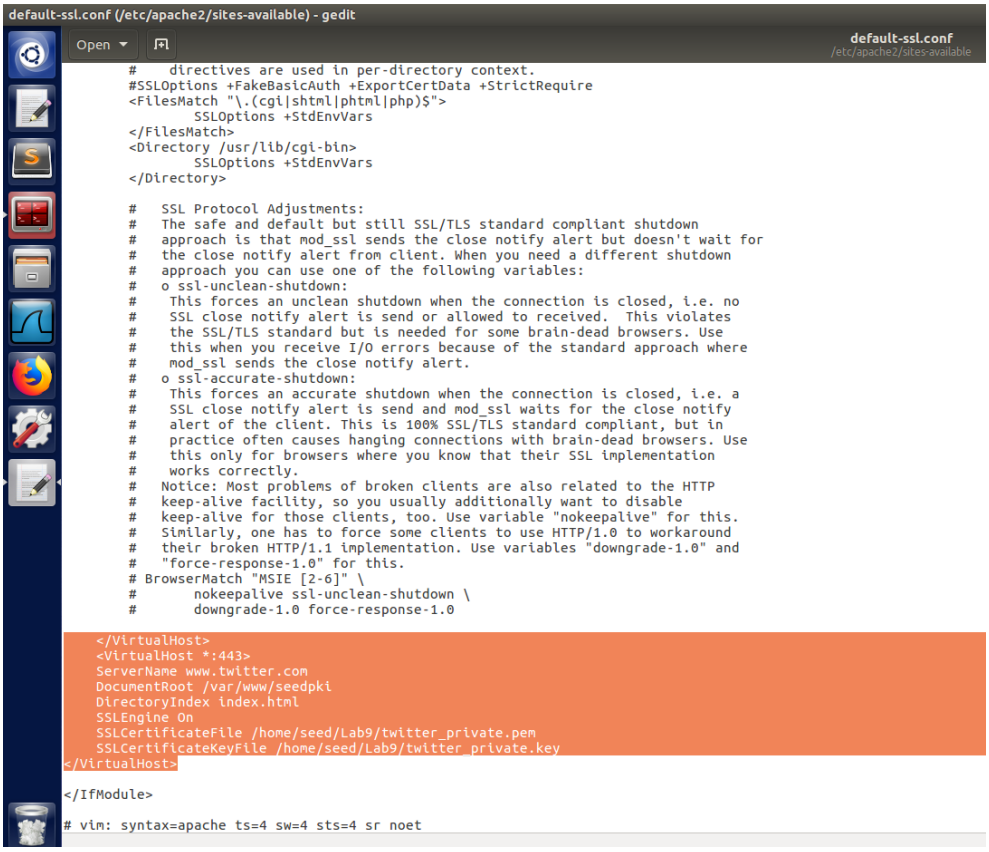
From the above screenshot we can see that the attack was not successful. This is because we do not have the twitter's certificate. The attack is restricted as the Common Name field does not contain twitter.com. And we cannot get the CN as twitter.com because we do not have its private attack

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




/bin/bash
/bin/bash 168x40
Not After : Apr 18 06:48:51 2020 GMT
Subject:
countryName = AU
stateOrProvinceName = Some-State
organizationName = Internet Widgits Pty Ltd
commonName = www.twitter.com
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
Netscape Comment:
OpenSSL Generated Certificate
X509v3 Subject Key Identifier:
65:29:F2:42:1C:1A:44:57:51:0D:45:4D:95:71:07:50:25:C5:00:EB
X509v3 Authority Key Identifier:
keyid:22:8E:2D:46:B5:75:50:76:C6:59:E7:14:4A:B0:8F:39:9C:CA:F8:52
Certificate is to be certified until Apr 18 06:48:51 2020 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
[04/19/2019 02:48] Client-Parth@VM:~/Lab9$ cp twitter_private.key twitter_private.pem
[04/19/2019 02:50] Client-Parth@VM:~/Lab9$ cat twitter_fake.crt >> twitter_private.pem
[04/19/2019 02:50] Client-Parth@VM:~/Lab9$ openssl s_server -cert twitter_private.pem -www
Enter pass phrase for twitter_private.pem:
Using default temp DH parameters
ACCEPT
^C
[04/19/2019 02:51] Client-Parth@VM:~/Lab9$ cd /etc/apache2
[04/19/2019 02:51] Client-Parth@VM:.../apache2$ ls
apache2.conf conf-available conf-enabled envvars magic mods-available mods-enabled ports.conf sites-available sites-enabled
[04/19/2019 02:51] Client-Parth@VM:.../apache2$ cd sites-available
[04/19/2019 02:51] Client-Parth@VM:.../sites-available$ sudo gedit default-ssl.conf
[sudo] password for seed:
(gedit:4595): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionMa
service files


/bin/bash
/bin/bash 168x40
[04/19/2019 02:50] Client-Parth@VM:~/Lab9$ cat twitter_fake.crt >> twitter_private.pem
[04/19/2019 02:50] Client-Parth@VM:~/Lab9$ openssl s_server -cert twitter_private.pem -www
Enter pass phrase for twitter_private.pem:
Using default temp DH parameters
ACCEPT
^C
[04/19/2019 02:51] Client-Parth@VM:~/Lab9$ cd /etc/apache2
[04/19/2019 02:51] Client-Parth@VM:.../apache2$ ls
apache2.conf conf-available conf-enabled envvars magic mods-available mods-enabled ports.conf sites-available sites
[04/19/2019 02:51] Client-Parth@VM:.../apache2$ cd sites-available
[04/19/2019 02:51] Client-Parth@VM:.../sites-available$ sudo gedit default-ssl.conf
[sudo] password for seed:
(gedit:4595): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gn
service files
** (gedit:4595): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-enabled not supported
** (gedit:4595): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported
** (gedit:4595): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-position not supported
[04/19/2019 02:54] Client-Parth@VM:.../sites-available$ cd
[04/19/2019 02:55] Client-Parth@VM:~$ sudo apachectl configtest
AH00112: Warning: DocumentRoot [/var/www/seedlabclickjacking] does not exist
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerNam
ge
Syntax OK
[04/19/2019 02:55] Client-Parth@VM:~$ 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
Module ssl already enabled
[04/19/2019 02:55] Client-Parth@VM:~$ sudo a2ensite default-ssl
Site default-ssl already enabled
[04/19/2019 02:55] Client-Parth@VM:~$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for www.twitter.com:443 (RSA): ****
[04/19/2019 02:55] Client-Parth@VM:~$



Assuming that we have stolen the private key of twitter. Therefore, we can now generate a public/private key pair. We can see from the above screenshots that we generated all the certificates and private key of twitter with CN as twitter.com. And then we restarted the apache server. And then we visit the twitter.com. We can see that the connection showing is secure and our attack is successful.