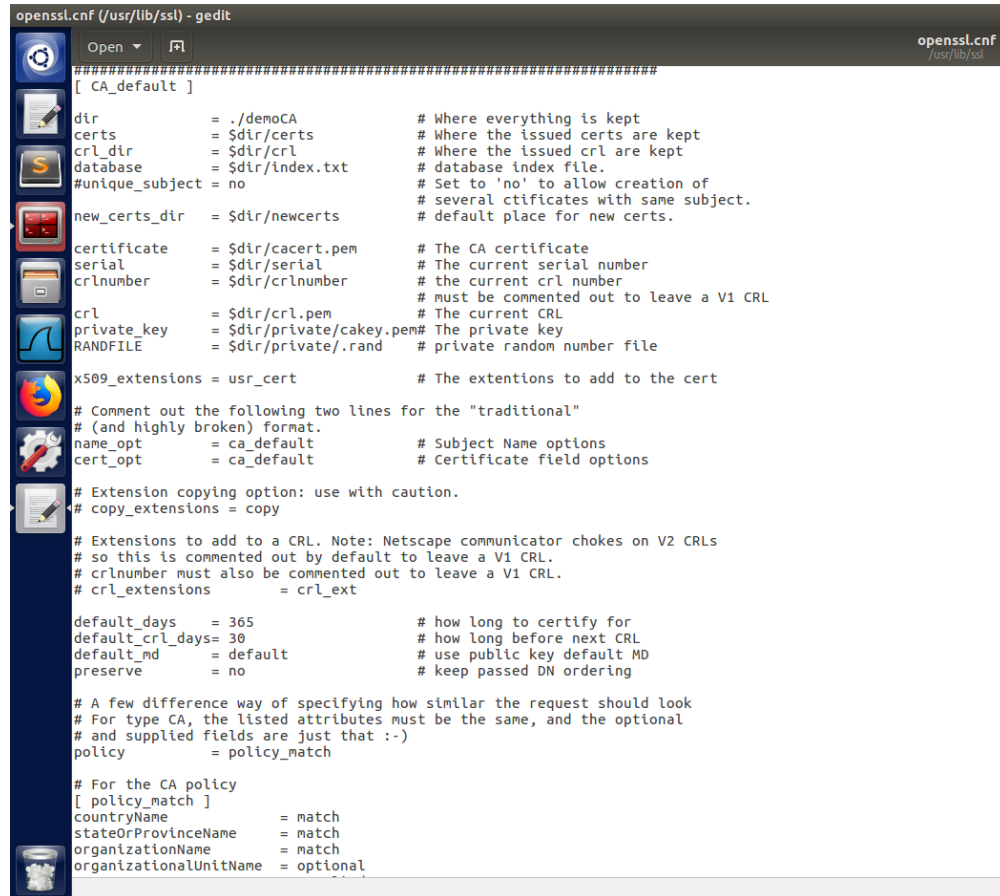


## Public-Key Infrastructure (PKI) Lab

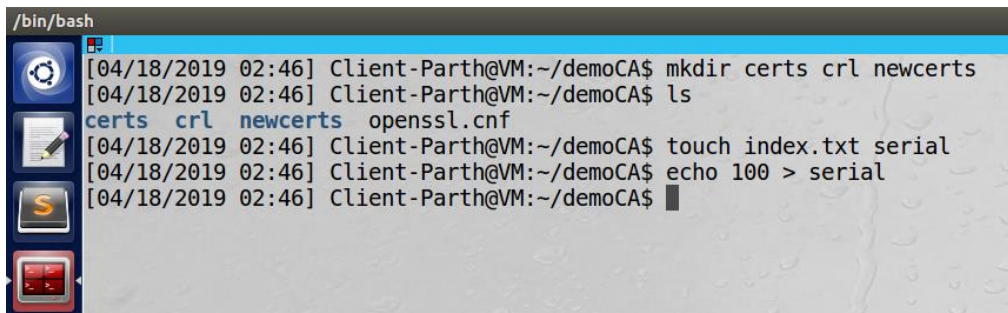
### Task 1: Becoming a Certificate Authority (CA)

#### The Configuration File

A screenshot of a gedit text editor window titled 'openssl.cnf (/usr/lib/ssl) - gedit'. The editor shows the contents of the 'openssl.cnf' file, which is a configuration file for the OpenSSL Certificate Authority. The file is divided into sections by square brackets. The first section is '[ CA\_default ]', which contains various settings for the CA. These include paths for certificates, CRLs, and private keys, as well as options for certificate creation and extension copying. The second section is '[ policy\_match ]', which contains settings for matching policy attributes. The third section is '[ policy\_match ]', which contains settings for matching policy attributes. The file is commented with many lines of text explaining the purpose of each setting.

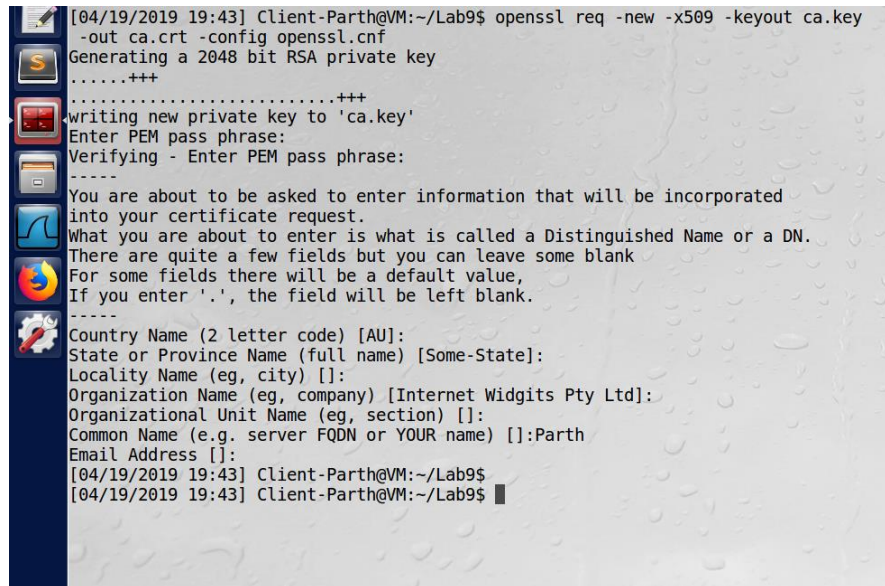
```
#####  
[ CA_default ]  
  
dir               = ./demoCA           # Where everything is kept  
certs             = $dir/certs         # Where the issued certs are kept  
crl_dir           = $dir/crl           # Where the issued crl are kept  
database          = $dir/index.txt     # database index file.  
#unique_subject   = no                 # Set to 'no' to allow creation of  
# several ctificates with same subject.  
# default place for new certs.  
  
new_certs_dir     = $dir/newcerts      # The CA certificate  
certificate       = $dir/cacert.pem    # The current serial number  
serial            = $dir/serial         # the current crl number  
crlnumber         = $dir/crlnumber     # must be commented out to leave a V1 CRL  
crl               = $dir/crl.pem       # The current CRL  
private_key       = $dir/private/cakey.pem# The private key  
RANDFILE         = $dir/private/.rand  # private random number file  
  
x509_extensions  = usr_cert           # The extensions to add to the cert  
  
# Comment out the following two lines for the "traditional"  
# (and highly broken) format.  
name_opt         = ca_default         # Subject Name options  
cert_opt         = ca_default         # Certificate field options  
  
# Extension copying option: use with caution.  
# copy_extensions = copy  
  
# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs  
# so this is commented out by default to leave a V1 CRL.  
# crlnumber must also be commented out to leave a V1 CRL.  
# crl_extensions = crl_ext  
  
default_days     = 365                 # how long to certify for  
default_crl_days= 30                  # how long before next CRL  
default_md       = default            # use public key default MD  
preserve        = no                   # keep passed DN ordering  
  
# A few difference way of specifying how similar the request should look  
# For type CA, the listed attributes must be the same, and the optional  
# and supplied fields are just that :~)  
policy          = policy_match  
  
# For the CA policy  
[ policy_match ]  
countryName     = match  
stateOrProvinceName = match  
organizationName = match  
organizationalUnitName = optional
```

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

A screenshot of a terminal window titled '/bin/bash'. The terminal shows a series of commands and their outputs. The user creates a directory named 'demoCA' and then lists its contents. The output shows that the directory is empty. The user then creates three subdirectories: 'certs', 'crl', and 'newcerts'. The user then creates two files: 'index.txt' and 'serial'. The output shows that the files are created successfully.

```
[04/18/2019 02:46] Client-Parth@VM:~/demoCA$ mkdir certs crl newcerts  
[04/18/2019 02:46] Client-Parth@VM:~/demoCA$ ls  
certs crl newcerts openssl.cnf  
[04/18/2019 02:46] Client-Parth@VM:~/demoCA$ touch index.txt serial  
[04/18/2019 02:46] Client-Parth@VM:~/demoCA$ echo 100 > serial  
[04/18/2019 02:46] Client-Parth@VM:~/demoCA$
```

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.

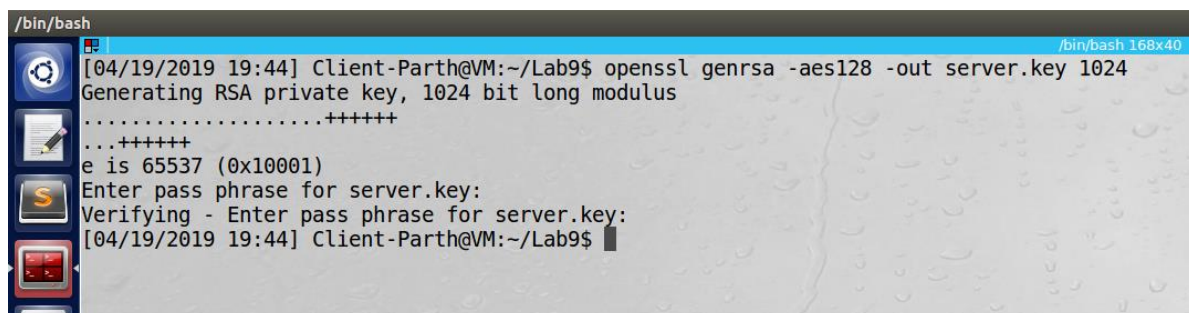


```
[04/19/2019 19:43] Client-Parth@VM:~/Lab9$ openssl req -new -x509 -keyout ca.key
-out ca.crt -config openssl.cnf
Generating a 2048 bit RSA private key
.....+++
writing new private key to 'ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information 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]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:Parth
Email Address []:
[04/19/2019 19:43] Client-Parth@VM:~/Lab9$
[04/19/2019 19:43] Client-Parth@VM:~/Lab9$
```

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.



```
/bin/bash
[04/19/2019 19:44] Client-Parth@VM:~/Lab9$ 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:
[04/19/2019 19:44] Client-Parth@VM:~/Lab9$
```

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.

```
/bin/bash
[04/19/2019 19:45] Client-Parth@VM:~/Lab9$ openssl rsa -in server.key -text
Enter pass phrase for server.key:
Private-Key: (1024 bit)
modulus:
 00:e1:4b:66:51:ff:89:e1:03:e4:d8:f7:81:69:cd:
 36:c2:f4:d9:47:e8:d6:9e:36:67:55:17:75:9d:ed:
 d8:f0:00:0a:00:71:7f:aa:79:e1:cf:02:ab:7d:6c:
 1c:76:76:03:51:47:75:92:5a:b0:52:9e:9a:1d:f8:
 e9:0f:a1:82:b2:a5:99:e8:b7:4b:38:d3:dd:ed:de:
 a6:fd:99:a7:a8:2a:0c:6b:94:45:09:48:66:de:19:
 ae:5b:63:39:b5:29:ed:2a:8d:56:04:4f:e8:2c:49:
 d4:72:5e:aa:84:c5:65:80:01:b5:11:8a:52:23:18:
 1d:1d:1f:47:35:1d:f0:ed:6d
publicExponent: 65537 (0x10001)
privateExponent:
 00:91:ed:d6:de:f2:18:8b:09:8c:a8:04:d9:6c:80:
 a2:21:f1:9a:f3:fb:1b:58:eb:80:4f:1b:9c:f1:a8:
 e8:45:bc:a0:bd:dc:c6:86:d3:df:b1:c8:d6:ef:ac:
 a1:5b:11:e1:e0:39:db:2b:eb:56:1e:8d:e0:e1:dd:
 22:89:1f:62:99:c1:66:08:a6:9f:24:0b:58:c2:7d:
 5c:c6:5c:47:ba:d0:6b:ea:81:b0:c8:a1:8d:86:d3:
 ed:8e:b4:6a:fe:43:36:d8:1c:ea:36:71:6e:4b:49:
 30:39:42:73:19:df:52:ca:05:61:ed:88:1f:47:ac:
 60:cf:b3:73:33:5d:4e:a9:b9
prime1:
 00:f8:1f:f1:e8:d9:71:f3:48:8c:f5:f5:30:f6:15:
 8f:08:7c:87:34:66:9a:3c:ca:37:fd:80:48:c2:7b:
 b9:9c:48:97:6f:b6:1d:42:e5:cc:f0:5e:80:17:f9:
 2e:93:5b:de:72:85:5c:78:b2:4c:f7:81:c2:37:77:
 85:78:0d:b3:c7
prime2:
 00:e8:71:f4:82:c5:ff:1d:ae:52:7b:a4:22:b9:cc:
 8d:6c:bc:45:9a:42:fc:88:a9:77:b9:ce:5c:e9:cc:
 86:3e:8c:f3:63:da:f5:52:48:dd:ba:f7:81:70:69:
 10:52:74:4a:ec:1e:5b:92:f6:dc:0d:2f:71:50:51:
 09:c1:28:6d:2b
exponent1:
 73:f9:3a:70:c3:71:e7:6c:79:b6:5c:ac:4d:d9:35:
 c9:99:aa:f8:6b:1c:9c:5d:48:5e:4c:9f:b5:87:6e:
```

```
/bin/bash
 00:e8:71:f4:82:c5:ff:1d:ae:52:7b:a4:22:b9:cc:
 8d:6c:bc:45:9a:42:fc:88:a9:77:b9:ce:5c:e9:cc:
 86:3e:8c:f3:63:da:f5:52:48:dd:ba:f7:81:70:69:
 10:52:74:4a:ec:1e:5b:92:f6:dc:0d:2f:71:50:51:
 09:c1:28:6d:2b
exponent1:
 73:f9:3a:70:c3:71:e7:6c:79:b6:5c:ac:4d:d9:35:
 c9:99:aa:f8:6b:1c:9c:5d:48:5e:4c:9f:b5:87:6e:
 f2:05:5c:02:f2:f4:2a:58:8e:b9:aa:f1:e4:42:ab:
 ea:64:7e:4c:b8:d8:c7:f1:4e:d8:40:e6:2e:56:c7:
 23:9e:03:f1
exponent2:
 00:89:c0:d6:ee:15:56:ee:89:1b:ba:c8:78:07:f2:
 70:b1:cb:15:d8:a8:f0:2e:31:78:91:b5:f9:9c:59:
 28:09:b3:d0:9e:11:ea:26:fc:a5:e3:22:c1:24:14:
 2f:a7:1d:e2:34:f2:7c:c4:a8:e5:9a:2e:ce:91:91:
 a5:1d:42:cf:07
coefficient:
 43:22:ed:8b:b1:6f:5a:a3:a6:7c:d1:62:cd:db:ce:
 a9:b2:3c:fa:07:1c:ad:11:09:da:52:01:6d:f4:c9:
 ff:30:99:15:f8:ac:64:6d:48:40:8e:fa:f5:4d:fd:
 52:b0:07:45:a3:ab:6f:a7:90:36:72:1c:a5:4b:9d:
 a2:a1:6a:bf
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQDhS2ZR/4nhA+TY94FpzTbC9NLH6NaeNmdVF3Wd7djwAAoAcX+q
eeHPAqt9bBx2dgNRR3WSWrBSnpod+OkPoYKypZnot0s4093t3qb9maeoKgxrLEUJ
SGbeGa5bYzmlKe0qjVYET+gsSdRyXqQExWAAABURiLIjGB0dH0c1HfDtBQIDAQAB
AoGBAJHt1t7yGI5JjKgE2WYAOiHxmvP7G1jrgE8bnPG06EW8oL3cxobT37HI1u+s
oVsR4eA52yvrVh6N40HdIokfYpnBZgimnyQLWJ9XMZC7rQa+qBsMihjYbT7Y60
av5DNtgc6jZxbktJMDLCcxnfUsoFYe2IH0esYM+zcNdTqm5AkEA+B/x6NLx80iM
9fUw9hWPCHyHNGaaPMo3/YBIwnu5nE1Xb7YdQuXM8F6AF/kuk1vecovceLJM94HC
N3eFeA2zxwJBA0hx9ILF/x2uUnukIrnMjWY8RZpC/Iipd7n0X0nMhj6M82Pa9VJI
3br3gXBpEFJ0SuweW5L23A0vcVBRCcEobScQHP50nDDcedsebZcrE3ZncmZqvhr
HJxdSF5Mn7WHbvIFXALy9CpYjrmq8eRCq+pkfky42MfxTthA5i5Wxy0eA/ECQOCJ
wNbuFVbuiRu6yHgH8nCxxyXYqPAuMXiRtmcWSgJ5c9eEom/KXjIsEkFC+nHeI0
8nzEq0WaLs6RkaUdQs8HAKBDIu2LsW9ao6Z80WLN286psjz6BxytEQnaUgFt9Mn/
MJkV+KxkbUhAjr1Tf1SAdFo6tvp5A2chylS521oWq/
-----END RSA PRIVATE KEY-----
[04/19/2019 19:45] Client-Parth@VM:~/Lab9$
```

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).

```
/bin/bash
[04/19/2019 18:36] Client-Parth@VM:~/Lab9$ 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]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:SEEDPKILab2018.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[04/19/2019 18:36] Client-Parth@VM:~/Lab9$
```

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.

```
[04/19/2019 19:48] Client-Parth@VM:~/Lab9$ openssl req -new -in server.csr -text -noout
Generating a 2048 bit RSA private key
.....+++
writing new private key to 'privkey.pem'
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]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:Parth
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Certificate Request:
Data:
  Version: 0 (0x0)
  Subject: C=AU, ST=Some-State, O=Internet Widgits Pty Ltd, CN=Parth
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (2048 bit)
    Modulus:
      00:b2:b2:fb:2f:40:f1:83:d0:b3:76:c3:92:1f:1f:
      72:d9:65:45:d0:c1:7a:3d:05:87:71:a9:12:a7:9a:
```

```
/bin/bash
Public-Key: (2048 bit)
Modulus:
  00:b2:b2:fb:2f:40:f1:83:d0:b3:76:c3:92:1f:1f:
  72:d9:65:45:d0:c1:7a:3d:05:87:71:a9:12:a7:9a:
  3f:47:68:51:5d:42:7c:2b:55:6c:01:6a:49:f0:73:
  76:a3:fe:df:ab:c6:87:8d:64:90:73:34:a2:36:b1:
  7d:51:56:67:70:6c:6e:e5:4a:cf:08:4e:93:80:09:
  f8:06:27:06:6a:7b:b8:51:c0:db:82:44:65:02:7a:
  71:66:8c:5f:48:4f:b4:d7:2a:d9:43:e8:a0:86:20:
  3d:d2:06:01:bc:cd:40:06:73:75:36:83:0e:3a:ec:
  39:a0:f3:a3:44:6f:ad:1a:df:15:9a:43:63:41:48:
  e0:93:a7:0f:33:f0:ad:54:a9:29:70:c3:a9:43:77:
  09:4a:ac:9e:8d:d3:4d:0b:a5:15:a2:32:f9:eb:04:
  e2:00:ac:46:73:73:50:58:29:72:02:ed:d2:26:00:
  a4:11:cb:ea:c0:13:33:ce:76:84:0d:3e:07:05:79:
  a0:8c:d8:94:f4:51:25:c5:11:98:8a:7a:7d:62:63:
  45:d9:c6:87:4a:7a:30:53:16:11:e4:d7:41:3a:21:
  cb:4f:19:2a:c9:27:d5:a5:a3:f5:03:cc:2a:21:66:
  e4:81:f7:34:6c:0f:ad:cf:d9:e2:5c:cc:76:3d:11:
  10:0d
Exponent: 65537 (0x10001)
Attributes:
  a0:00
Signature Algorithm: sha256WithRSAEncryption
  75:4a:80:66:b7:ca:92:23:dc:42:9c:49:42:39:92:78:9b:d7:
  bb:d5:e4:0f:44:17:aa:84:9f:0d:b0:cc:94:d9:a8:1d:24:18:
  a8:78:45:98:d6:38:57:7f:f8:cd:37:4a:34:d0:d6:19:24:e3:
  20:ca:3d:1c:60:5a:f1:eb:0b:5c:61:0b:cc:66:81:d8:73:5c:
  59:fc:c7:30:9d:33:5e:06:7f:f7:e7:62:f5:ab:8d:41:a2:ef:
  cc:2f:0b:61:f5:f4:0c:e9:81:02:3a:7b:91:a1:f1:1a:17:b0:
  b1:af:f8:a0:c3:39:43:c3:0c:9a:b4:6e:9a:f3:e2:58:0b:71:
  61:72:54:fe:ab:3c:98:cf:d3:c7:52:96:2f:df:e0:0e:34:e7:
  c3:12:74:13:55:52:4d:2d:ac:f0:3b:b9:d9:dc:5f:37:57:a7:
  0b:a9:91:37:92:31:52:19:ca:8d:43:55:89:e9:ba:11:d5:f1:
  c0:10:f3:6f:b7:99:2e:04:a3:37:88:02:43:d1:fe:ae:f3:4c:
  20:af:43:fa:2a:3e:65:16:4d:10:01:be:f7:cb:72:d8:0f:5f:
  c6:f6:52:94:a6:41:53:65:4a:db:c9:c3:4d:29:7a:e0:d8:f0:
  3d:28:bb:22:fd:c2:bf:40:92:fe:d9:18:0d:92:b4:1d:48:bf:
  e6:53:a3:d1
[04/19/2019 19:49] Client-Parth@VM:~/Lab9$
```

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

### Step 3: Generating Certificates.

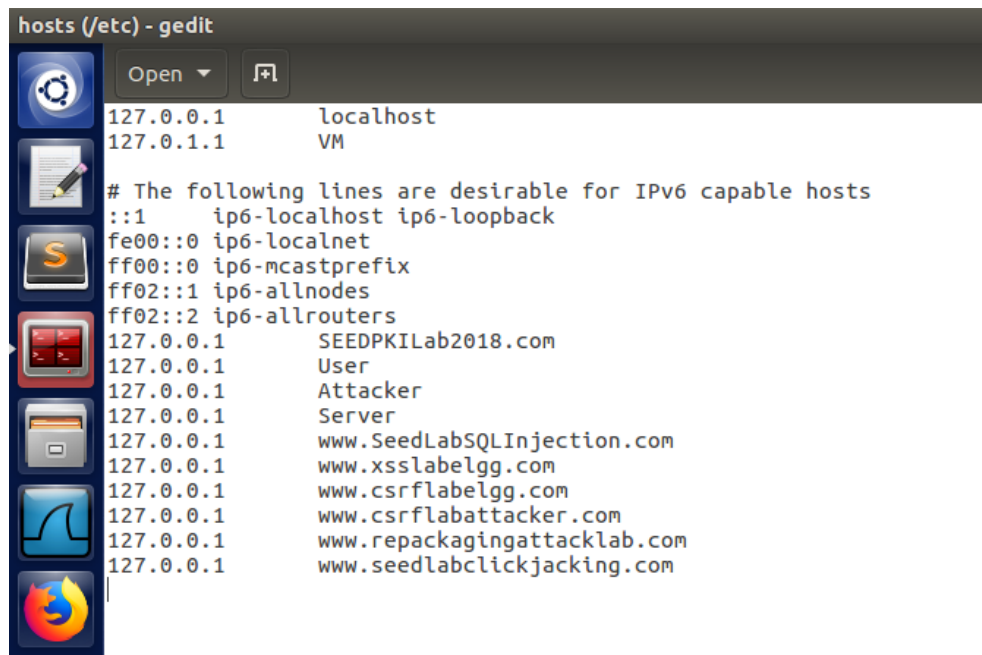
```
openssl.cnf (~/.demoCA) - gedit
# copy_extensions = copy
# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crlnumber must also be commented out to leave a V1 CRL.
# crl_extensions = crl_ext
default_days = 365 # how long to certify for
default_crl_days = 30 # how long before next CRL
default_md = default # use public key default MD
preserve = no # keep passed DN ordering
# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy = policy_anything
# For the CA policy
[ policy_match ]
countryName = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
# For the 'anything' policy
# types.
[ policy_anything ]
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
#####
[ req ]
default_bits = 2048
default_keyfile = privkey.pem
distinguished_name = req_distinguished_name
attributes = req_attributes
x509_extensions = v3_ca # The extensions to add to the self signed cert
# Passwords for private keys if not present they will be prompted for
# input_password = secret
# output_password = secret
# This sets a mask for permitted string types. There are several options.
```

```
/bin/bash
[04/19/2019 18:40] Client-Parth@VM:~/Lab9$ sudo openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -con
[sudo] password for seed:
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 4098 (0x1002)
  Validity
    Not Before: Apr 19 22:40:51 2019 GMT
    Not After : Apr 18 22:40:51 2020 GMT
  Subject:
    countryName = AU
    stateOrProvinceName = Some-State
    organizationName = Internet Widgits Pty Ltd
    commonName = SEEDPKILab2018.com
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
  Netscape Comment:
    OpenSSL Generated Certificate
  X509v3 Subject Key Identifier:
    63:1D:3D:DC:F0:40:6C:93:24:2E:12:4A:DF:F7:90:97:0D:FB:E2:FA
  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 22:40:51 2020 GMT (365 days)
Sign the certificate? [y/n]:y
```

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.

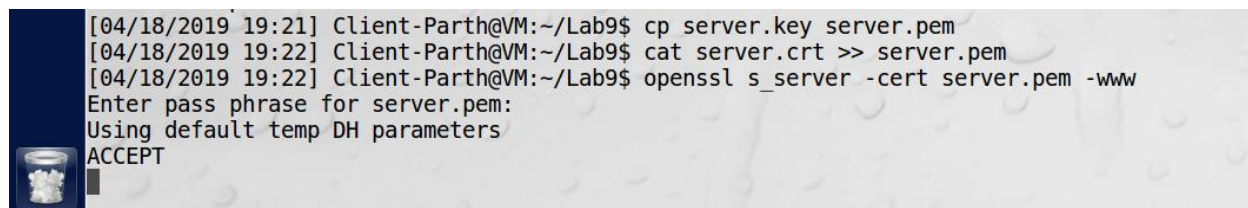
### Task 3: Deploying Certificate in an HTTPS Web Server

#### 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.

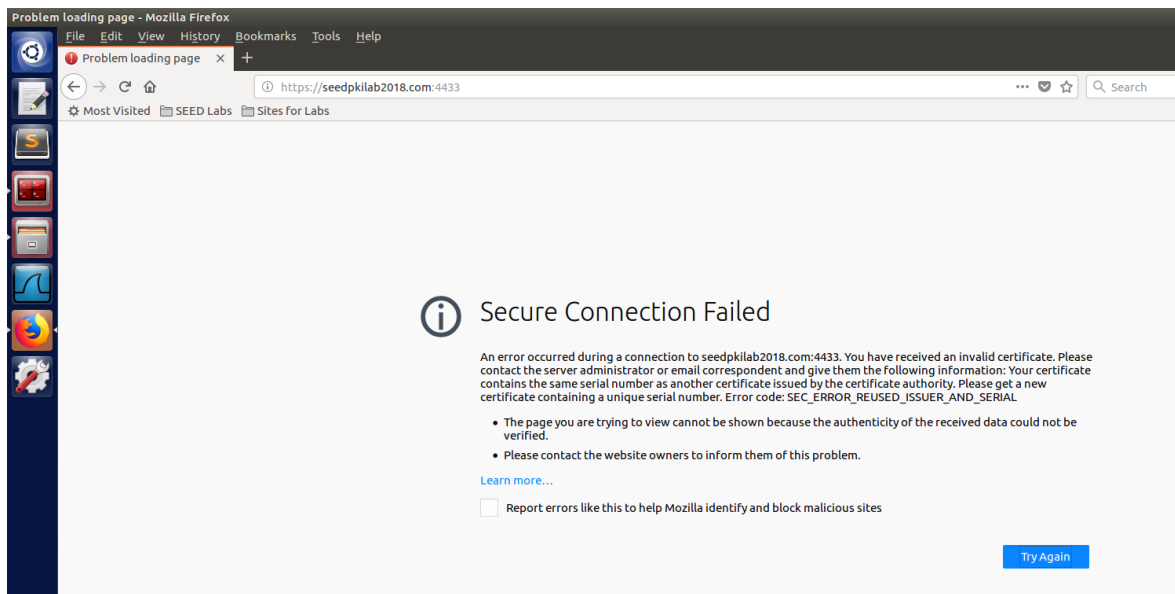
## Step 3: Getting the browser to accept our CA certificate

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.



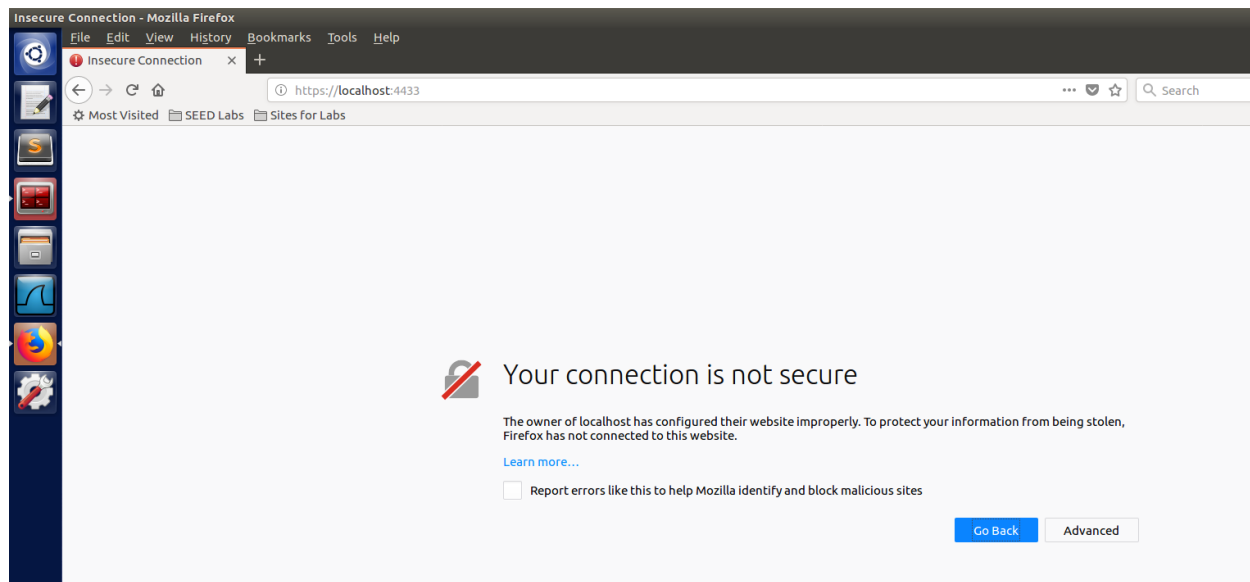
In this step we first corrupted the `server.pem` file and then restarted the server. After which we reloaded the URL. We can see from the screenshot below that the browser is showing secure Connection failed that is because it now as an invalid certificate.





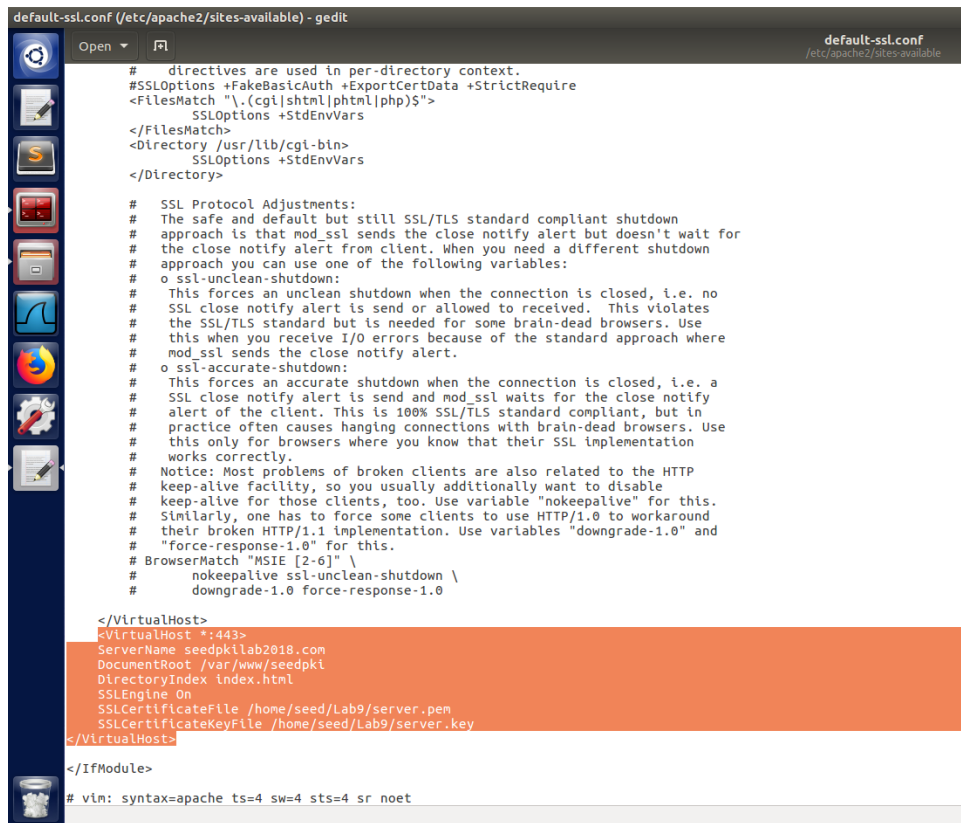
After this step we restore the original server.pem file.

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



```
default-ssl.conf (/etc/apache2/sites-available) - gedit
# directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
    SSLOptions +StdEnvVars
</Directory>

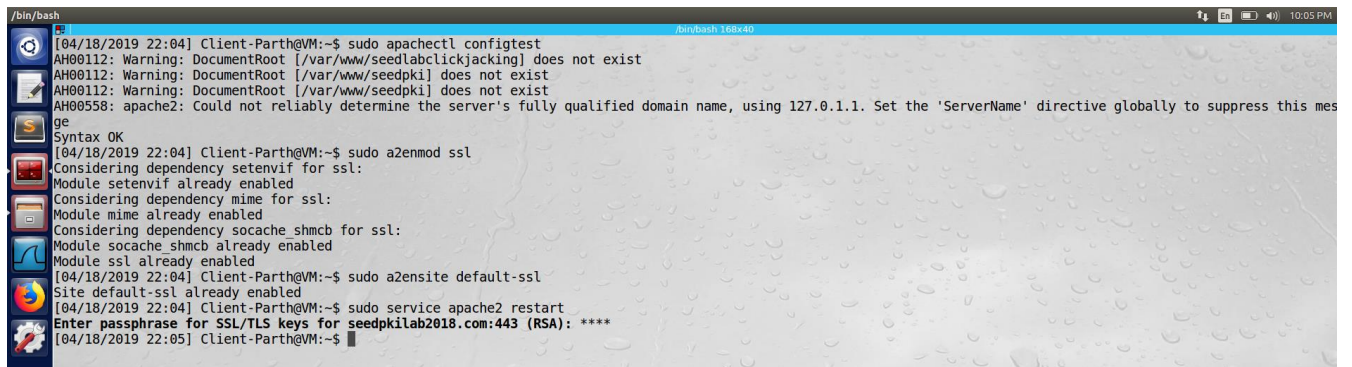
# SSL Protocol Adjustments:
# The safe and default but still SSL/TLS standard compliant shutdown
# approach is that mod_ssl sends the close notify alert but doesn't wait for
# the close notify alert from client. When you need a different shutdown
# approach you can use one of the following variables:
# o ssl-unclean-shutdown:
# This forces an unclean shutdown when the connection is closed, i.e. no
# SSL close notify alert is send or allowed to received. This violates
# the SSL/TLS standard but is needed for some brain-dead browsers. Use
# this when you receive I/O errors because of the standard approach where
# mod_ssl sends the close notify alert.
# o ssl-accurate-shutdown:
# This forces an accurate shutdown when the connection is closed, i.e. a
# SSL close notify alert is send and mod_ssl waits for the close notify
# alert of the client. This is 100% SSL/TLS standard compliant, but in
# practice often causes hanging connections with brain-dead browsers. Use
# this only for browsers where you know that their SSL implementation
# works correctly.
# Notice: Most problems of broken clients are also related to the HTTP
# keep-alive facility, so you usually additionally want to disable
# keep-alive for those clients, too. Use variable "nokeepalive" for this.
# Similarly, one has to force some clients to use HTTP/1.0 to workaround
# their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
# "force-response-1.0" for this.
# BrowserMatch "MSIE [2-6]" \
#     nokeepalive ssl-unclean-shutdown \
#     downgrade-1.0 force-response-1.0

</VirtualHost>
<VirtualHost *:443>
    ServerName seedpkilab2018.com
    DocumentRoot /var/www/seedpki
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /home/seed/Lab9/server.pem
    SSLCertificateKeyFile /home/seed/Lab9/server.key
</VirtualHost>

</IfModule>

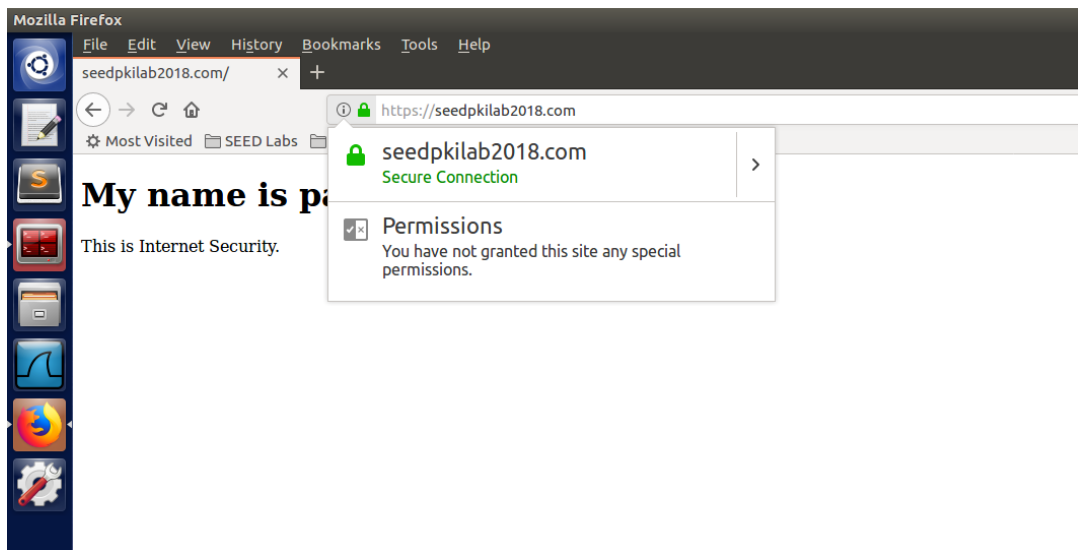
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

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.



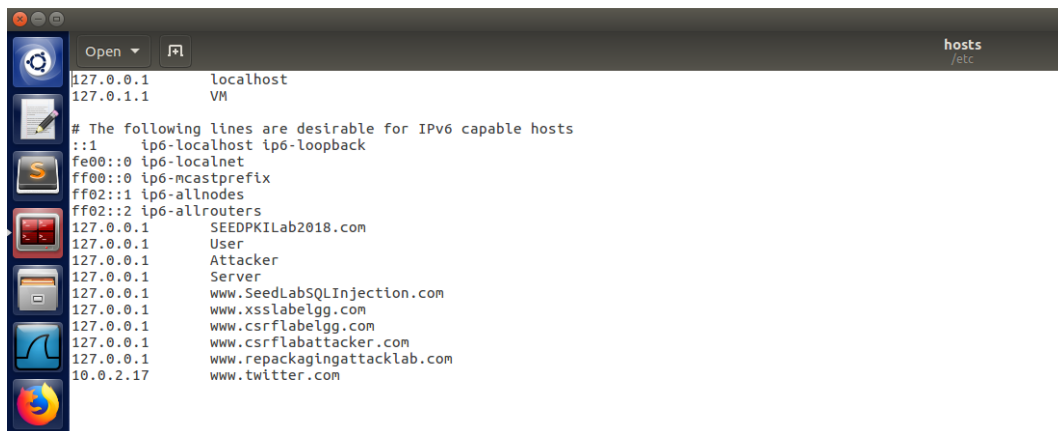
```
/bin/bash
[04/18/2019 22:04] Client-Parth@VM:~$ sudo apachectl configtest
AH00112: Warning: DocumentRoot [/var/www/seedlabclickjacking] does not exist
AH00112: Warning: DocumentRoot [/var/www/seedpki] does not exist
AH00112: Warning: DocumentRoot [/var/www/seedpki] does not exist
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
[04/18/2019 22:04] 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
[04/18/2019 22:04] Client-Parth@VM:~$ sudo a2ensite default-ssl
Site default-ssl already enabled
[04/18/2019 22:04] Client-Parth@VM:~$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for seedpkilab2018.com:443 (RSA): ****
[04/18/2019 22:05] Client-Parth@VM:~$
```

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

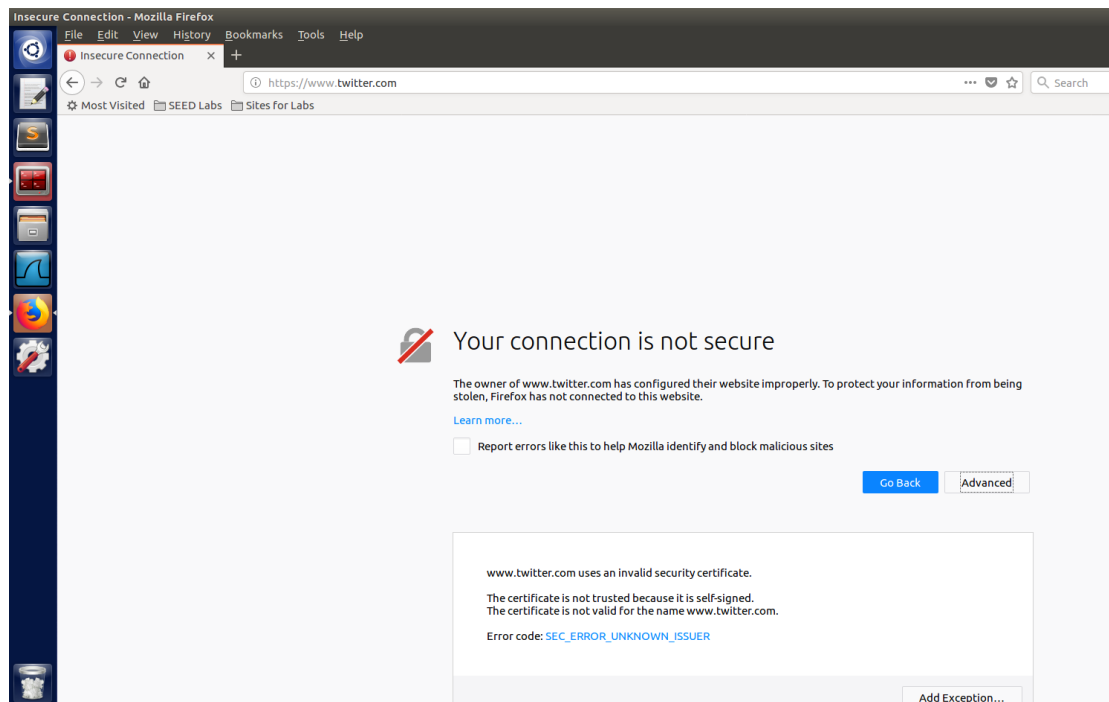


From the above screenshot we can see that when we visit the website seedpkilab2018.com the greenlock states that the connection is secure which means our setup is successful.

### 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](http://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
[04/19/2019 02:42] Client-Parth@VM:/etc$ cd
[04/19/2019 02:42] Client-Parth@VM:~$ cd Lab9
[04/19/2019 02:42] Client-Parth@VM:~/Lab9$ openssl genrsa -aes128 -out twitter_private.key 1024
Generating RSA private key, 1024 bit long modulus
.....+++++
e is 65537 (0x10001)
Enter pass phrase for twitter_private.key:
Verifying - Enter pass phrase for twitter_private.key:
[04/19/2019 02:44] Client-Parth@VM:~/Lab9$ openssl req -new -key twitter_private.key -out twitter_fake.csr -config openssl.cnf
Enter pass phrase for twitter_private.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]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:www.twitter.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[04/19/2019 02:46] Client-Parth@VM:~/Lab9$ openssl ca -in twitter_fake.csr -out twitter_fake.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: 4097 (0x1001)
    Validity
        Not Before: Apr 19 06:48:51 2019 GMT

```



```
/bin/bash
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

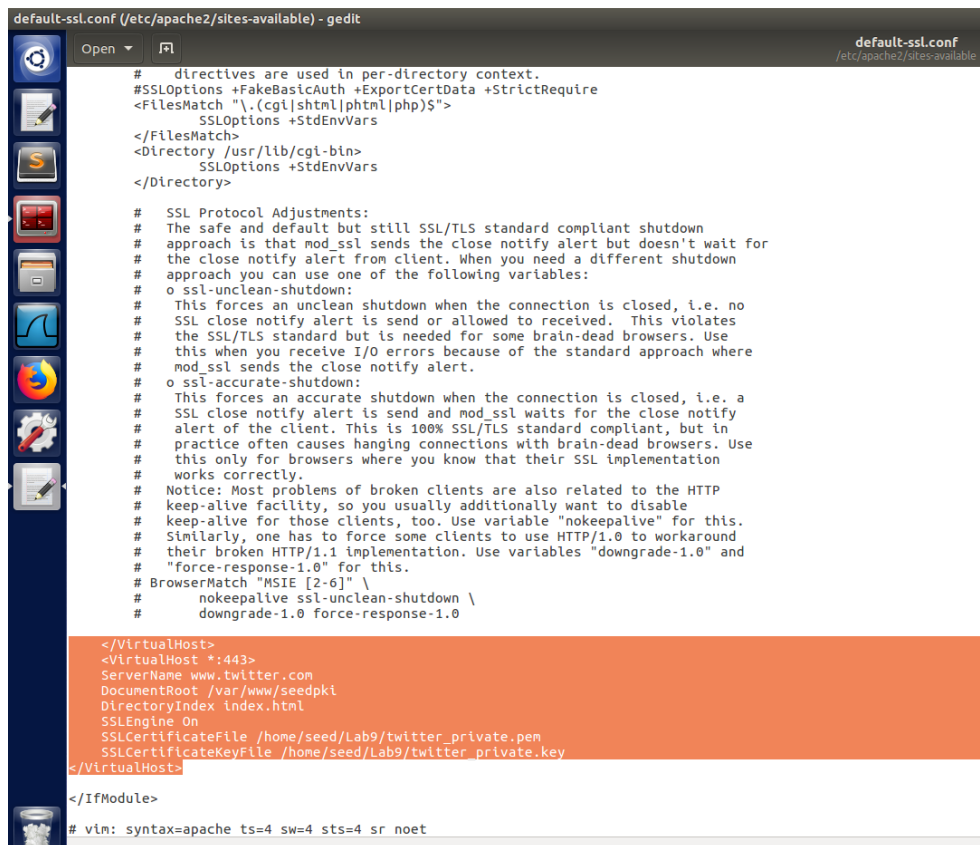
1 out of 1 certificate requests certified, commit? [y/n]
Write out database with 1 new entries
Data Base Updated
[04/19/2019 02:48] Client-Parth@VM:~/Lab9$ cp twitter_private.key twitter_private.pem
[04/19/2019 02:50] Client-Parth@VM:~/Lab9$ cat twitter_fake.crt >> twitter_private.pem
[04/19/2019 02:50] 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:~/Lab9$ 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:~/Lab9$ cd sites-available
[04/19/2019 02:51] Client-Parth@VM:~/Lab9$ 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.SessionManager is not registered for name service files
```

```
/bin/bash
[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:~/Lab9$ 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:~/Lab9$ cd sites-available
[04/19/2019 02:51] Client-Parth@VM:~/Lab9$ 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.SessionManager is not registered for name 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:~/Lab9$ cd
[04/19/2019 02:55] Client-Parth@VM:~/Lab9$ 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 'ServerName' directive manually to avoid this warning.
Syntax OK
[04/19/2019 02:55] Client-Parth@VM:~/Lab9$ 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:~/Lab9$ sudo a2ensite default-ssl
Site default-ssl already enabled
[04/19/2019 02:55] Client-Parth@VM:~/Lab9$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for www.twitter.com:443 (RSA): ****
[04/19/2019 02:55] Client-Parth@VM:~/Lab9$
```



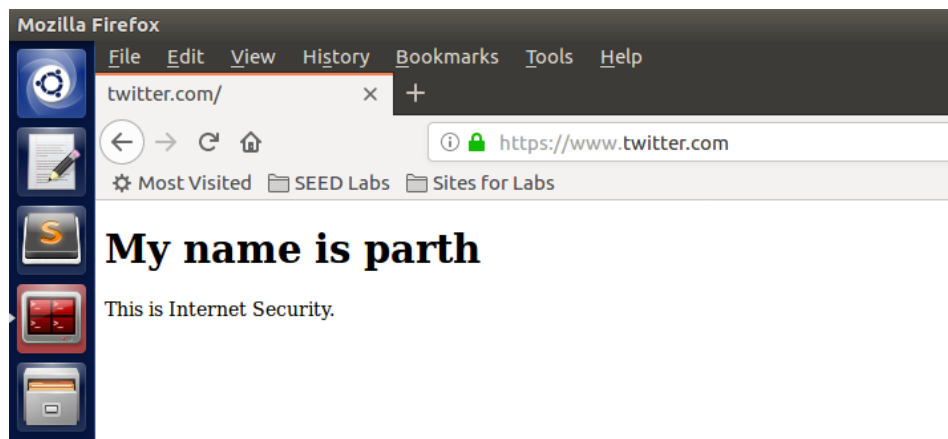
```
default-ssl.conf (/etc/apache2/sites-available) - gedit
# directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|sh|html|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
    SSLOptions +StdEnvVars
</Directory>

#
# SSL Protocol Adjustments:
# The safe and default but still SSL/TLS standard compliant shutdown
# approach is that mod_ssl sends the close notify alert but doesn't wait for
# the close notify alert from client. When you need a different shutdown
# approach you can use one of the following variables:
# o ssl-unclean-shutdown:
# This forces an unclean shutdown when the connection is closed, i.e. no
# SSL close notify alert is send or allowed to received. This violates
# the SSL/TLS standard but is needed for some brain-dead browsers. Use
# this when you receive I/O errors because of the standard approach where
# mod_ssl sends the close notify alert.
# o ssl-accurate-shutdown:
# This forces an accurate shutdown when the connection is closed, i.e. a
# SSL close notify alert is send and mod_ssl waits for the close notify
# alert of the client. This is 100% SSL/TLS standard compliant, but in
# practice often causes hanging connections with brain-dead browsers. Use
# this only for browsers where you know that their SSL implementation
# works correctly.
# Notice: Most problems of broken clients are also related to the HTTP
# keep-alive facility, so you usually additionally want to disable
# keep-alive for those clients, too. Use variable "nokeepalive" for this.
# Similarly, one has to force some clients to use HTTP/1.0 to workaround
# their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
# "force-response-1.0" for this.
# BrowserMatch "MSIE [2-6]" \
#     nokeepalive ssl-unclean-shutdown \
#     downgrade-1.0 force-response-1.0
#

</VirtualHost>
<VirtualHost *:443>
    ServerName www.twitter.com
    DocumentRoot /var/www/seedpki
    DirectoryIndex index.html
    SSLEngine On
    SSLCertificateFile /home/seed/Lab9/twitter_private.pem
    SSLCertificateKeyFile /home/seed/Lab9/twitter_private.key
</VirtualHost>

</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```



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.