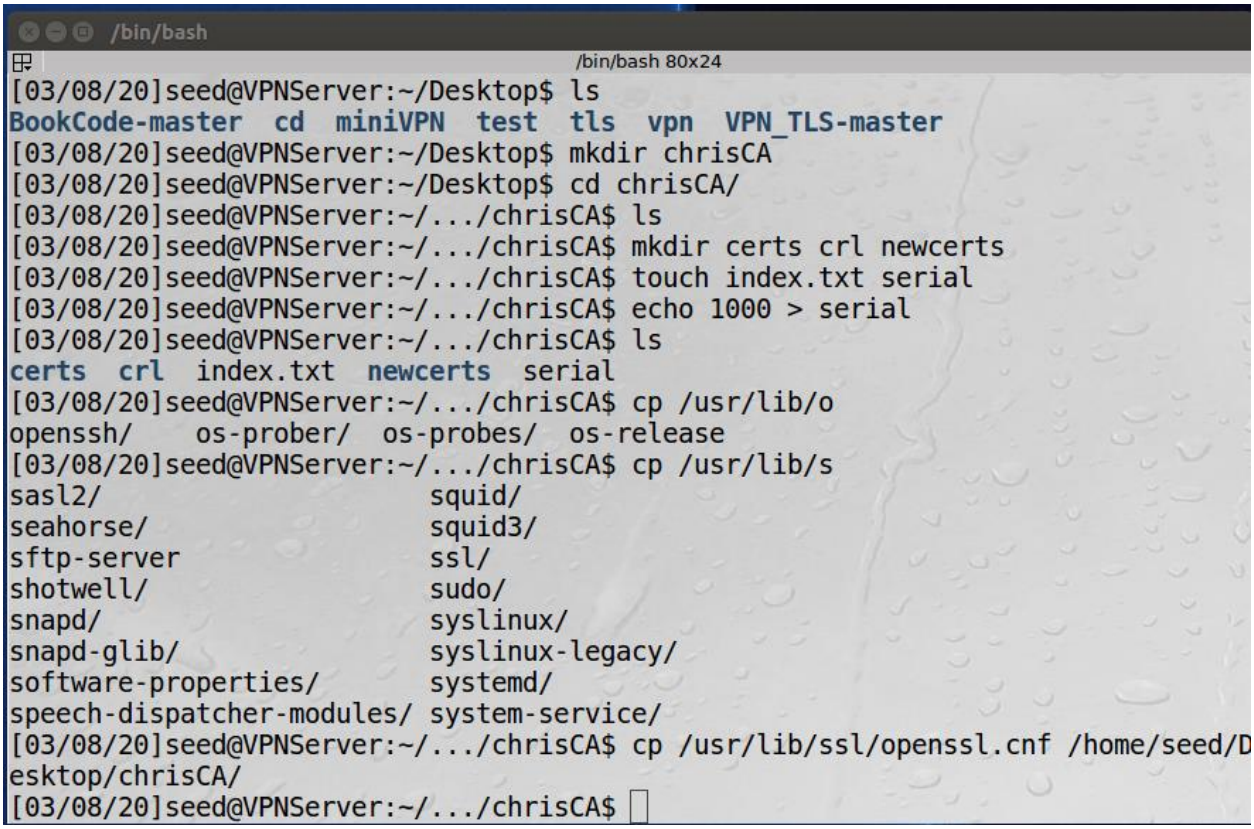
Task 1:

Creating our own Certificate Authority that issues digital certificates for other systems that can authenticate to other system by being verified by the CA.

Checking for /usr/lib/ssl/openssl.cnf file and to store within my own cert directory, customize the configuration when I used for command openssl.



Created a directory to store the CA files

create folder to store certs to issue the certs, crl issued the crl and new certs to place new certs..

create file index.txt store the database index file, and serial will contain the serial number for the next certificate.

Task 2

Creating self-signed certification adding a PEM pass phrase each time you want to use this CA to sign certificates

```
[03/08/20]seed@VPNServer:~/.../CAcerts$ sudo openssl req -new -x509 -keyout ca.k
ey -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]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:NYC
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Chris Desir INC
Organizational Unit Name (eg, section) []:cd2902
Common Name (e.g. server FQDN or YOUR name) []:cd2902ca.com
Email Address []:cd2902@nyu.edu
[03/08/20]seed@VPNServer:~/.../CAcerts$
```

The certificate needs to know the information about the CA so clients can verify if the CA is legit or not.

create its own public/private key pair.

```
ca.crt  ca.key  chrisCA  openssl.cnf
[03/08/20]seed@VPNServer:~/.../CAcerts$ sudo 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:
```

The company will create public and private key to attach their key for a CA

```
verifying - Enter pass phrase for server.key:
[03/08/20]seed@VPNServer:~/.../CAcerts$ sudo openssl rsa -in server.key -text
Enter pass phrase for server.key:
Private-Key: (1024 bit)
modulus:
    00:dd:49:d1:8d:11:13:9f:1f:d2:24:02:c8:04:17:
    32:67:01:44:aa:0d:ef:95:fc:55:13:64:c3:d4:01:
    4b:d7:9e:57:f2:5e:68:bf:c2:58:17:d5:4d:0e:2a:
    df:5b:db:63:b1:25:f0:ab:be:d5:e9:fa:aa:1d:f1:
    bd:14:48:9c:d6:f2:4f:dc:9a:d2:43:93:5c:57:24:
    68:19:4d:63:1c:07:ec:69:22:fb:4f:b8:5d:7e:78:
    21:c3:25:9d:fc:1a:db:00:da:dd:1a:16:d2:53:cf:
    46:d4:1c:4f:18:49:60:71:eb:dd:75:dc:dd:cb:e3:
    a3:d6:d9:69:a1:86:d2:de:4b
publicExponent: 65537 (0x10001)
privateExponent:
    00:9e:16:15:59:04:e1:12:a5:a5:f7:49:f1:60:52:
    be:14:2e:28:c2:9f:f0:10:23:53:17:e6:bd:ab:a0:
    3d:71:4d:52:a0:f8:67:36:ce:fd:26:11:a0:d9:c5:
    31:0a:2b:62:3d:f4:cf:c9:b2:2c:99:da:14:5c:0e:
    0d:b9:06:e2:db:ab:ee:86:fc:fe:ab:72:a4:58:ef:
    31:61:fd:34:46:ac:cc:ef:63:82:2f:2a:2b:64:d4:
    bc:8c:50:10:a0:b5:ff:99:4e:c1:d3:2e:6d:e7:2c:
    3e:56:07:ef:62:ad:11:e9:3c:84:b1:96:a1:8b:75:
    e6:02:45:1a:c8:c8:1e:9d:09
prime1:
    00:ef:17:73:49:c5:bd:8a:48:fb:19:74:76:bb:6f:
    e2:33:1d:c4:d0:13:ab:82:b8:d1:47:f2:c9:6e:d0:
    fa:52:53:6e:1f:1b:6d:eb:ed:3c:de:06:49:ef:9f:
    fb:09:38:a9:9d:1a:31:25:f6:d8:bb:c2:64:75:af:
    7e:db:1a:fa:15
prime2:
    00:ec:f0:0e:39:7d:60:be:2e:f2:7e:5d:9c:ba:2b:
    96:dd:d9:03:6d:f2:1e:a2:14:34:5e:9a:85:ca:be:
    10:84:0a:f9:a9:ff:05:b6:a1:95:e4:fd:88:f6:f4:
    5b:20:c1:75:ea:5f:85:e5:26:0c:6a:36:d7:d1:5e:
    51:d4:e6:6e:df
exponent1:
    00:d1:eb:b0:00:94:ca:6f:10:d6:ce:a9:af:d9:b0:
    49:55:82:0c:9c:78:b8:bc:a4:92:b3:6e:1a:a7:de:
```

```
    87:30:b8:7c:4b:80:84:a8:85:8c:13:86:80:f7:c7:
    27:45:2e:f1:65
exponent2:
    59:e5:26:70:77:7a:5a:0d:f3:e1:2f:e1:43:dd:3f:
    eb:41:e8:04:31:e5:9f:76:ec:fb:ab:c2:f1:4e:35:
    73:f3:ba:0f:de:7b:2b:74:99:2b:7d:6a:16:d0:55:
    36:d3:4f:ca:b1:9d:5a:78:9a:23:3e:55:99:d1:20:
    59:17:68:9f
coefficient:
    42:9a:3e:af:00:e3:5a:59:f5:27:d1:41:08:9c:b1:
    9a:cc:8e:02:f7:82:51:7f:49:51:b0:f1:8f:8c:96:
    8b:38:86:77:51:f7:48:5b:ea:7b:85:b2:ae:fb:92:
    c3:07:0b:91:0a:98:99:2a:d7:a4:b8:bd:e4:f4:84:
    1e:96:bc
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDdSdGNEROfH9IkAsgEFzJnAUSqDe+V/FUTZMPUAUvXnlfyXmi/
wlgX1U0OKt9b22OxJfCrvtXp+qod8b0USJzW8k/cmtJDk1xXJGgZTWMcB+xpIvtP
uF1+eCHDJZ38GtsA2t0aFtJTz0bUHE8YSWBx69113N3L46PW2WmhhtLeSwIDAQAB
AoGBAJ4WFVkE4RKlpfdJ8WBSvhQuKMKf8BAjUxfmvaugPXFNUqD4ZzbO/SYRoNnF
MQorYj30z8myLJnaFFwODbkG4tur7ob8/qtypFjvMWH9NEaszO9jgi8qK2TUvIxQ
EKC1/5lOwdMubecsPlYH72KtEek8hLGWoYt15gJFGsjIHp0JAkEA7xdzScW9ikj7
GXR2u2/iMx3E0BOrgrjRR/LJbtD6UlNuHxtt6+083gZJ75/7CTipnRoxJfbYu8Jk
da9+2xr6FQJBAOzwDjl9YL4u8n5dnLorlt3ZA23yHqIUNF6ahcq+EIQK+an/Bbah
leT9iPb0WyDBdepfheUmDGo219FeUdTmbt8CQQDR67AAlMpvENbOqa/ZsElVggyc
eLi8pJKzbhqn3nn6MGBelZG0dG5drf3ak4cwuHxLgISohYwThoD3xydFLvFlAkBZ
5SZwd3paDfPhL+FD3T/rQegEMeWfduz7q8LxTjVz87oP3nsrdJkrfWoW0FU200/K
sZ1aeJojPlWZ0SBZF2ifAj9Cmj6vAONaWfUn0UEInLGazI4C94JRf0lRsPGPjJaL
OIZ3UfdIW+p7hbKu+5LDBwuRCpiZKtekuL3k9IQelrw=
-----END RSA PRIVATE KEY-----
[03/08/20]seed@VPNServer:~/.../CAcerts$ sudo openssl rsa -in server.key -text
Enter pass phrase for server.key:
Private-Key: (1024 bit)
```

generates a Certificate Signing Request (CSR), which basically includes the company's public key as a identification of the system that wants to request certificate from a CA.

```
[03/08/20]seed@VPNServer:~/.../CAcerts$ sudo openssl req -new -key server.key -o
ut 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]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:NYC
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SEED
Organizational Unit Name (eg, section) []:PKI
Common Name (e.g. server FQDN or YOUR name) []:SEEDPKILAB2020.com
Email Address []:cd2902@nyu.edu

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:seed2
An optional company name []:.
[03/08/20]seed@VPNServer:~/.../CAcerts$
```

The CSR file needs to have the CA's signature to form a certificate. In the real world, the CSR files are usually sent to a trusted CA for their signature. In this lab, we will use our own trusted CA to generate certificates. The following command turns the certificate signing request (server.csr) into an X509 certificate (server.crt), using the CA's ca.crt and ca.key

```
[03/08/20]seed@VPNServer:~/.../CAcerts$ sudo openssl ca -in server.csr -out server.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
The organizationName field needed to be the same in the
CA certificate (Chris Desir INC) and the request (SEED)
```

```
[03/08/20]seed@VPNServer:~/.../CAcerts$ sudo openssl ca -in server.csr -out server.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: 4096 (0x1000)
        Validity
            Not Before: Mar  8 18:02:41 2020 GMT
            Not After : Mar  8 18:02:41 2021 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = New York
            localityName              = NYC
            organizationName          = SEED
            organizationalUnitName    = PKI
            commonName                = SEEDPKILAB2020.com
            emailAddress              = cd2902@nyu.edu
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                D7:E9:5A:C7:FB:08:0A:1E:5C:8D:2D:3B:20:C2:F5:19:B0:E8:CA:09
            X509v3 Authority Key Identifier:
                keyid:A0:A5:8B:C0:6F:67:52:F6:DD:7F:72:24:BE:AF:8A:4E:FD:50:AF:B2

Certificate is to be certified until Mar  8 18:02:41 2021 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
```

Had to edit policy to policy anything so the entries within the certificate and CA certificate doesn't give wrong entries if its different.

```
[03/08/20]seed@VPNServer:~/.../CAcerts$ sudo cat openssl.cnf | grep policy
tsa_policy1 = 1.2.3.4.1
tsa_policy2 = 1.2.3.4.5.6
tsa_policy3 = 1.2.3.4.5.7
policy          = policy_anything
# For the CA policy
[ policy_match ]
# For the 'anything' policy
[ policy_anything ]
proxyCertInfo=critical,language:id-ppl-anyLanguage,pathlen:3,policy:foo
default_policy  = tsa_policy1           # Policy if request did not specify it
other_policies  = tsa_policy2, tsa_policy3      # acceptable policies (optional)
```

Task 3

```
  GNU nano 2.5.3                                File: /etc/hosts

127.0.0.1       localhost
127.0.1.1       VM

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
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
192.168.85.137   vpnlabserver.com
127.0.0.1       SEEDPKILAB2020.com
```

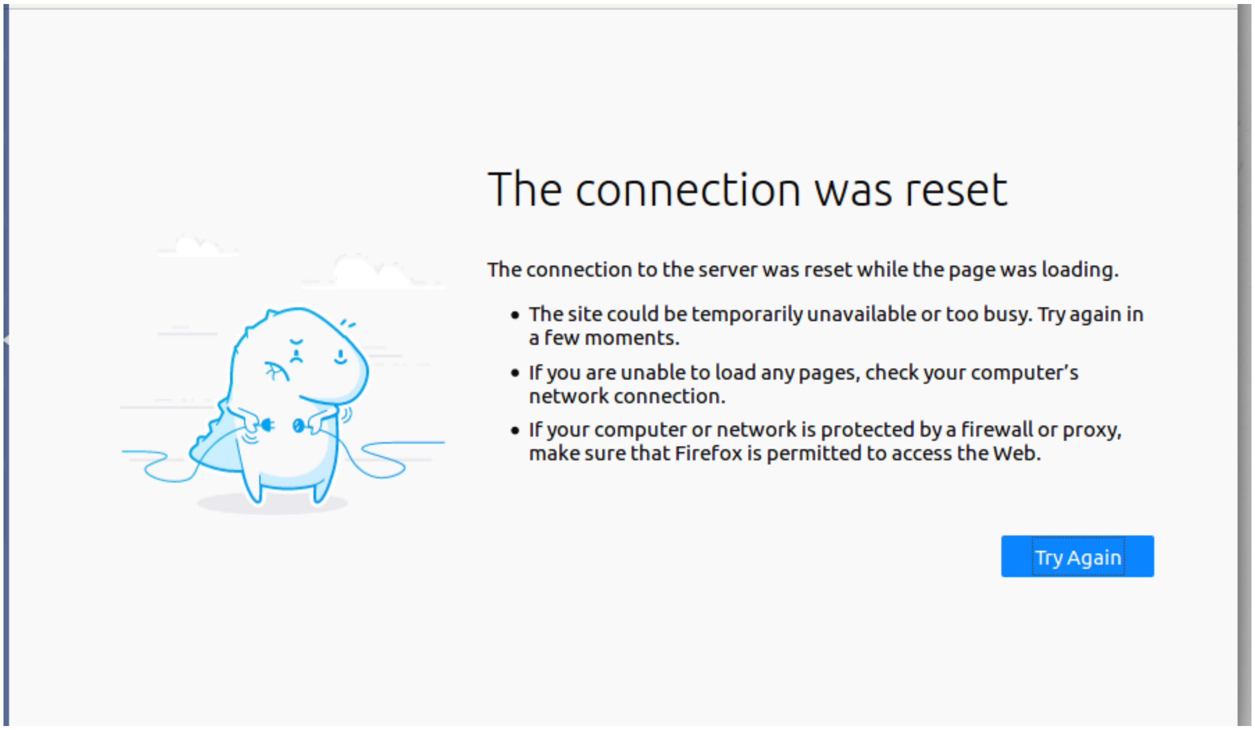
In the host file to map my hostname to my local IP so I can enter the domain name instead of IP.

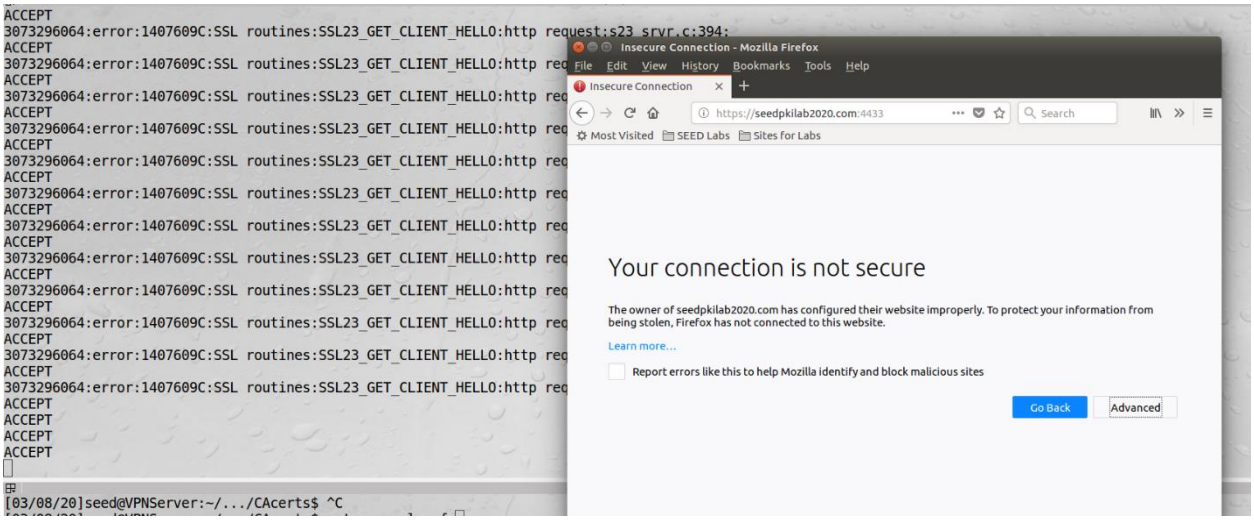
```
[03/08/20]seed@VPNServer:~/.../CAcerts$ ls
ca.crt  ca.key  chrisCA  openssl.cnf  server.crt  server.csr  server.key
[03/08/20]seed@VPNServer:~/.../CAcerts$ cp server.key server.pem
[03/08/20]seed@VPNServer:~/.../CAcerts$ cat server.crt >> server.pem
[03/08/20]seed@VPNServer:~/.../CAcerts$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
Using default temp DH parameters
ACCEPT
```

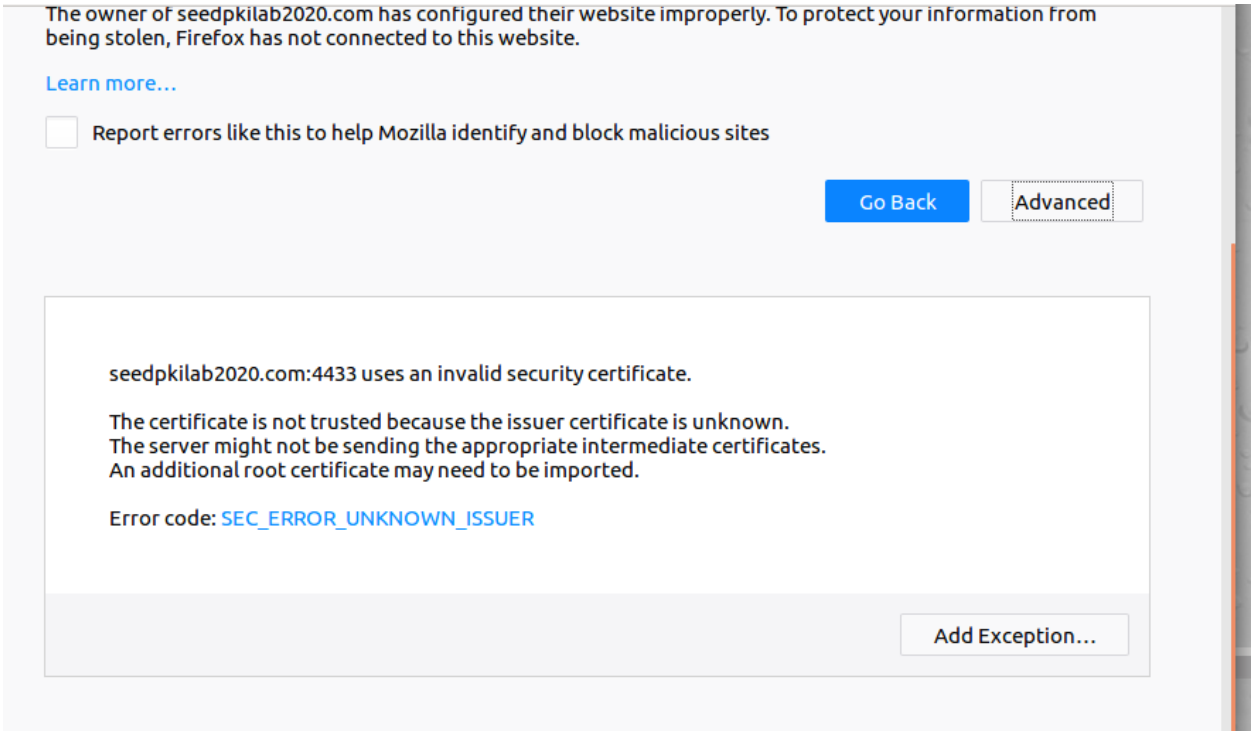
Authenticating myself with my key to the CA.

```
Using default temp DH parameters
ACCEPT
3070535360:error:1407609C:SSL routines:SSL23_GET_CLIENT_HELLO:http request:s23_srvr.c:394:
ACCEPT
3070535360:error:1407609C:SSL routines:SSL23_GET_CLIENT_HELLO:http request:s23_srvr.c:394:
ACCEPT
3070535360:error:1407609C:SSL routines:SSL23_GET_CLIENT_HELLO:http request:s23_srvr.c:394:
ACCEPT
3070535360:error:1407609C:SSL routines:SSL23_GET_CLIENT_HELLO:http request:s23_srvr.c:394:
ACCEPT
3070535360:error:1407609C:SSL routines:SSL23_GET_CLIENT_HELLO:http request:s23_srvr.c:394:
ACCEPT
3070535360:error:1407609C:SSL routines:SSL23_GET_CLIENT_HELLO:http request:s23_srvr.c:394:
ACCEPT
3070535360:error:1407609C:SSL routines:SSL23_GET_CLIENT_HELLO:http request:s23_srvr.c:394:
ACCEPT
3070535360:error:1407609C:SSL routines:SSL23_GET_CLIENT_HELLO:http request:s23_srvr.c:394:
ACCEPT
3070535360:error:1407609C:SSL routines:SSL23_GET_CLIENT_HELLO:http request:s23_srvr.c:394:
ACCEPT
3070535360:error:1407609C:SSL routines:SSL23_GET_CLIENT_HELLO:http request:s23_srvr.c:394:
ACCEPT
```

Browser could not see it because it needs to import the CA and key for me to authenticate myself through the privacy setting in Firefox.

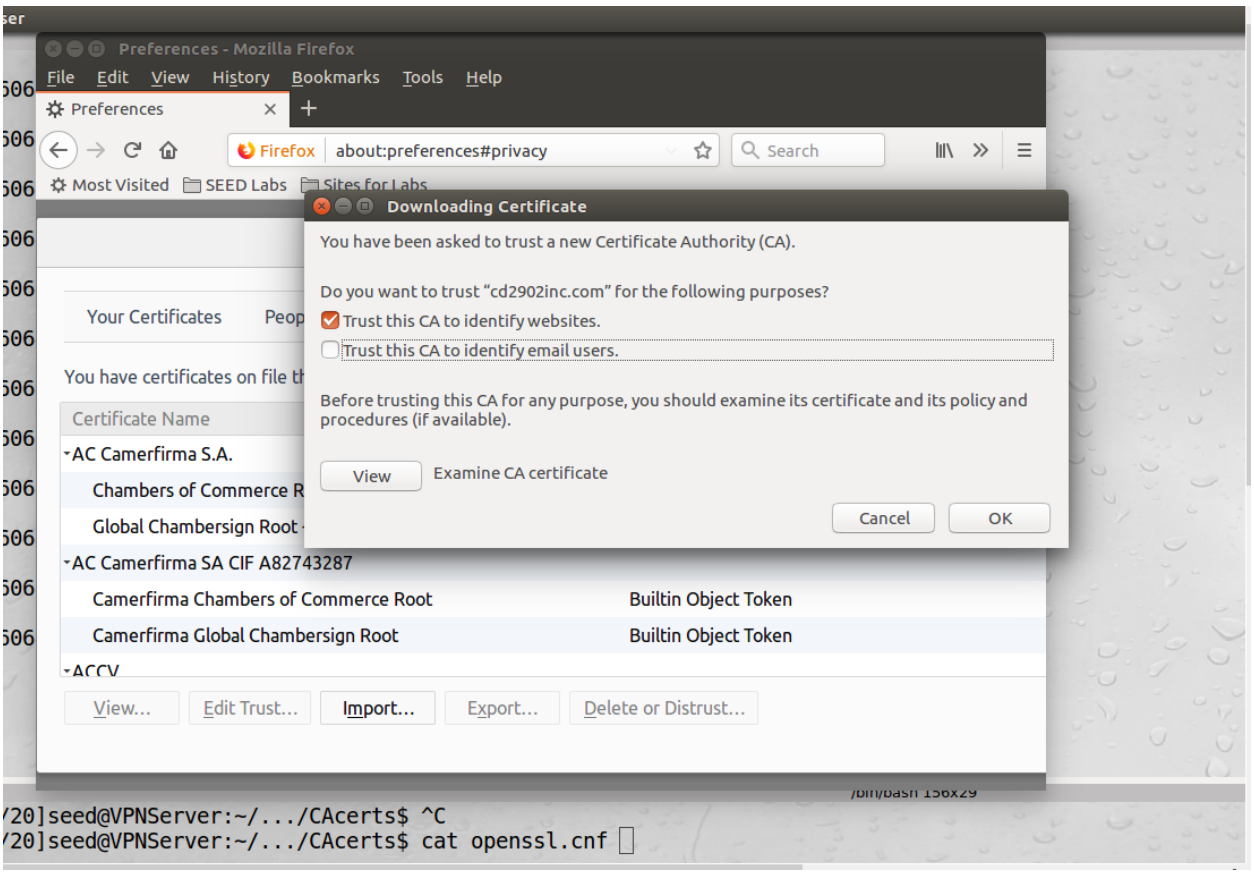


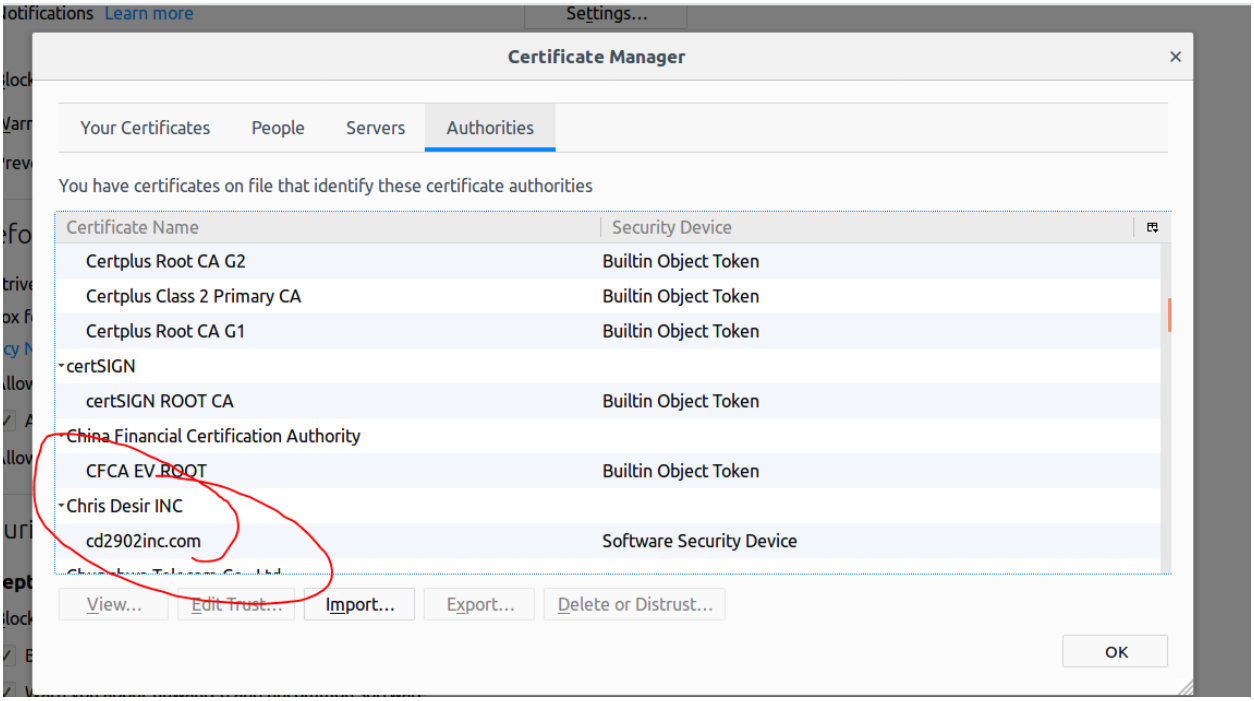




One of the cool part about this is that the browser wants to make sure if you want to trust that CA because if you import a CA that's compromise than the attack will act as a middle between the

client and the server.

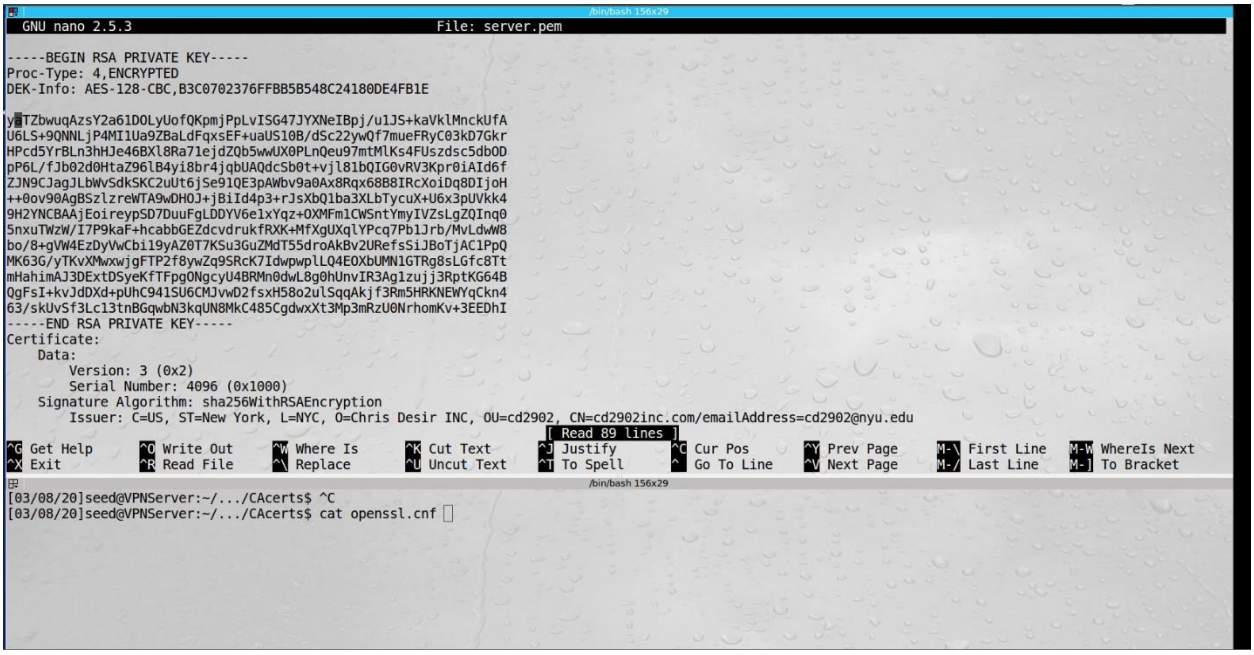




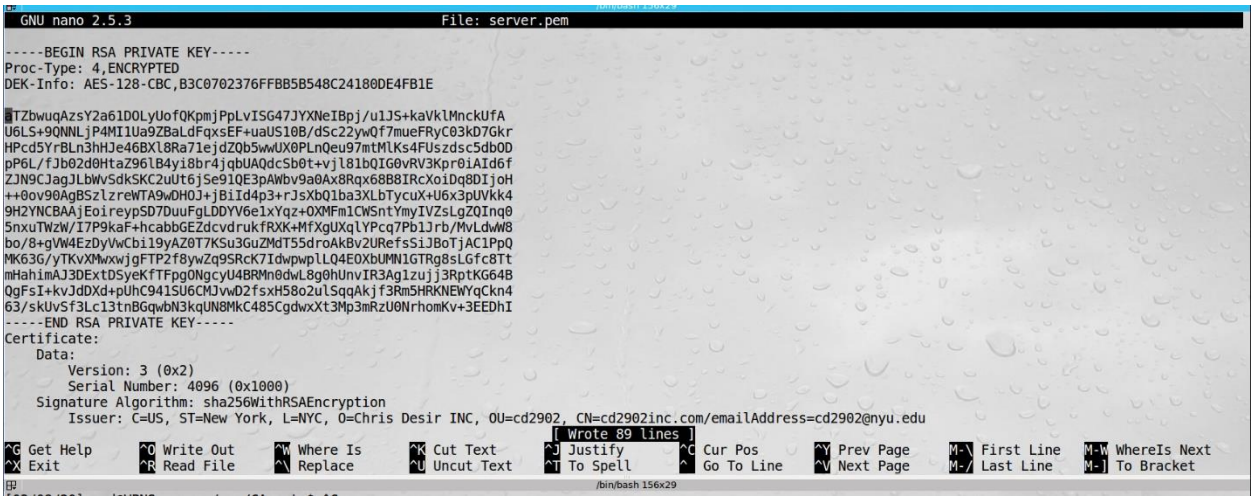
Now I enter the site and was able look through I was able to enter the website by importing the CA and ke and shows that im in a secure request because im on the transport layer security protocol

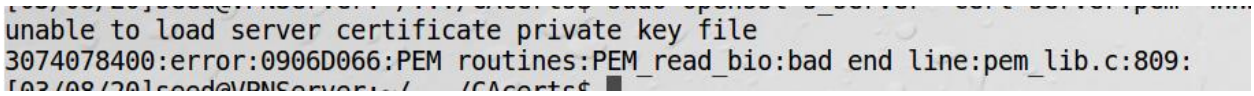
```
s_server -cert server.pem -www
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1/SSLv3:ECDHE-RSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDHE-ECDSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDHE-RSA-AES256-SHA384  TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA384
TLSv1/SSLv3:ECDHE-RSA-AES256-SHA     TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA
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:DH-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-RSA-AES256-GCM-SHA384TLSv1/SSLv3:DHE-RSA-AES256-SHA256
TLSv1/SSLv3:DHE-DSS-AES256-SHA256    TLSv1/SSLv3:DH-RSA-AES256-SHA256
TLSv1/SSLv3:DH-DSS-AES256-SHA256     TLSv1/SSLv3:DHE-RSA-AES256-SHA
TLSv1/SSLv3:DHE-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:DH-RSA-CAMELLIA256-SHA
TLSv1/SSLv3:DH-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-SHA384 TLSv1/SSLv3:ECDH-RSA-AES256-SHA
TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA    TLSv1/SSLv3:AES256-GCM-SHA384
TLSv1/SSLv3:AES256-SHA256            TLSv1/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-SHA256  TLSv1/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:DH-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:DHE-RSA-AES128-GCM-SHA256TLSv1/SSLv3:DHE-RSA-AES128-SHA256
TLSv1/SSLv3:DHE-DSS-AES128-SHA256    TLSv1/SSLv3:DH-RSA-AES128-SHA256
TLSv1/SSLv3:DH-DSS-AES128-SHA256     TLSv1/SSLv3:DHE-RSA-AES128-SHA
TLSv1/SSLv3:DHE-DSS-AES128-SHA       TLSv1/SSLv3:DH-RSA-AES128-SHA
TLSv1/SSLv3:DH-DSS-AES128-SHA        TLSv1/SSLv3:DHE-RSA-SEED-SHA
TLSv1/SSLv3:DHE-DSS-SEED-SHA         TLSv1/SSLv3:DH-RSA-SEED-SHA
TLSv1/SSLv3:DH-DSS-SEED-SHA          TLSv1/SSLv3:DHE-RSA-CAMELLIA128-SHA
TLSv1/SSLv3:DHE-DSS-CAMELLIA128-SHA  TLSv1/SSLv3:DH-RSA-CAMELLIA128-SHA
TLSv1/SSLv3:DH-DSS-CAMELLIA128-SHA   TLSv1/SSLv3:ECDH-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:ECDH-ECDSA-AES128-GCM-SHA256TLSv1/SSLv3:ECDH-RSA-AES128-SHA256
TLSv1/SSLv3:ECDH-ECDSA-AES128-SHA256 TLSv1/SSLv3:ECDH-RSA-AES128-SHA
TLSv1/SSLv3:ECDH-ECDSA-AES128-SHA    TLSv1/SSLv3:AES128-GCM-SHA256
TLSv1/SSLv3:AES128-SHA256            TLSv1/SSLv3:AES128-SHA
TLSv1/SSLv3:SEED-SHA                 TLSv1/SSLv3:CAMELLIA128-SHA
TLSv1/SSLv3:PSK-AES128-CBC-SHA       TLSv1/SSLv3:ECDHE-RSA-RC4-SHA
TLSv1/SSLv3:ECDHE-ECDSA-RC4-SHA      TLSv1/SSLv3:ECDH-RSA-RC4-SHA
TLSv1/SSLv3:ECDH-ECDSA-RC4-SHA       TLSv1/SSLv3:RC4-SHA
TLSv1/SSLv3:RC4-MD5                  TLSv1/SSLv3:PSK-RC4-SHA
TLSv1/SSLv3:ECDHE-RSA-DES-CBC3-SHA   TLSv1/SSLv3:ECDHE-ECDSA-DES-CBC3-SHA
TLSv1/SSLv3:SRP-DSS-3DES-EDE-CBC-SHA TLSv1/SSLv3:SRP-RSA-3DES-EDE-CBC-SHA
TLSv1/SSLv3:SRP-3DES-EDE-CBC-SHA     TLSv1/SSLv3:EDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:EDH-DSS-DES-CBC3-SHA     TLSv1/SSLv3:DH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:DH-DSS-DES-CBC3-SHA      TLSv1/SSLv3:ECDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:ECDH-ECDSA-DES-CBC3-SHA  TLSv1/SSLv3:DES-CBC3-SHA
TLSv1/SSLv3:PSK-3DES-EDE-CBC-SHA
Signature Algorithms: ECDSA+SHA256:ECDSA+SHA384:ECDSA+SHA512:0x04+0x08:0x05+0x08:0x06+0x08:RSA+SHA256:RSA+SHA384:RSA+SHA512:ECDSA+SHA1:RSA+SHA1
---
Reused, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256

TLSv1/SSLv3:ECDH-ECDSA-DES-CBC3-SHA  TLSv1/SSLv3:DES-CBC3-SHA
TLSv1/SSLv3:PSK-3DES-EDE-CBC-SHA
Signature Algorithms: ECDSA+SHA256:ECDSA+SHA384:ECDSA+SHA512:0x04+0x08:0x05+0x08:0x06+0x08:RSA+SHA256:RSA+SHA384:RSA+SHA512:ECDSA+SHA1:RSA+SHA1
---
Reused, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID: F950AB09831FAA79ADD5EB6D211B6A8266660B261742B66F0957EF66069053FC
    Session-ID-ctx: 01000000
    Master-Key: EBA911139C732FD86B398F5722BEE0491676EB1FFC12F026607FB1CA0892EB9DB12BD83BAB8EE9B8D1AE18957DAEADE8
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1583691849
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
   0 items in the session cache
   0 client connects (SSL_connect())
   0 client renegotiates (SSL_connect())
   0 client connects that finished
  27 server accepts (SSL_accept())
   0 server renegotiates (SSL_accept())
   7 server accepts that finished
   3 session cache hits
   4 session cache misses
   0 session cache timeouts
   0 callback cache hits
   0 cache full overflows (128 allowed)
---
no client certificate available
```

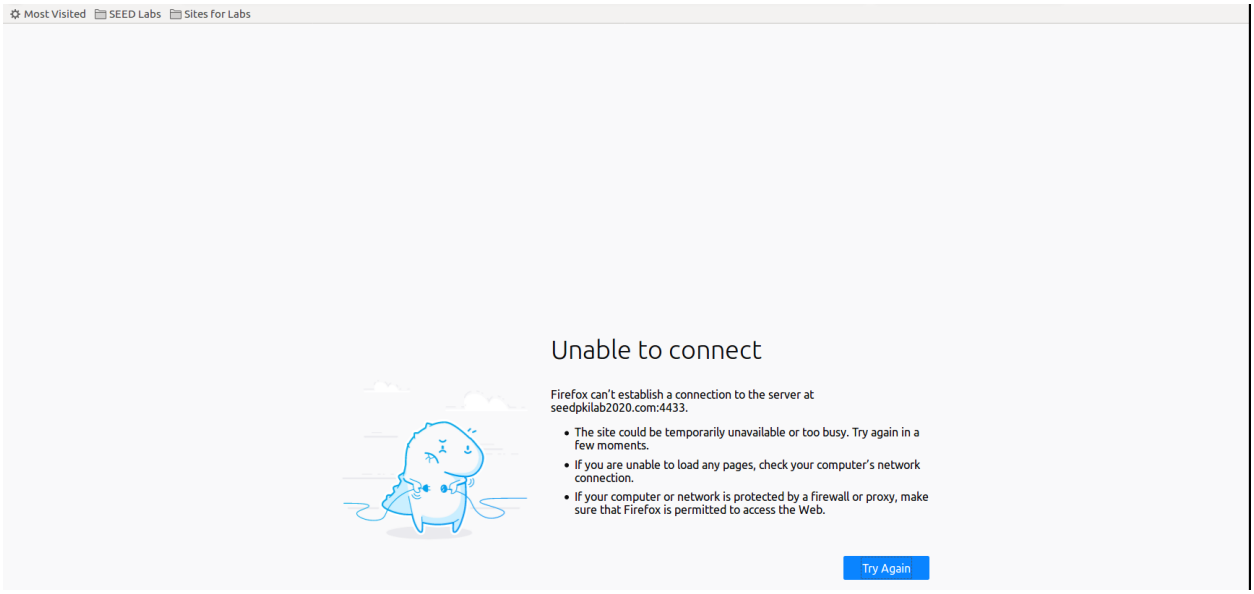
Here I'm going to change my key and see If I'm able to get in.



Changing



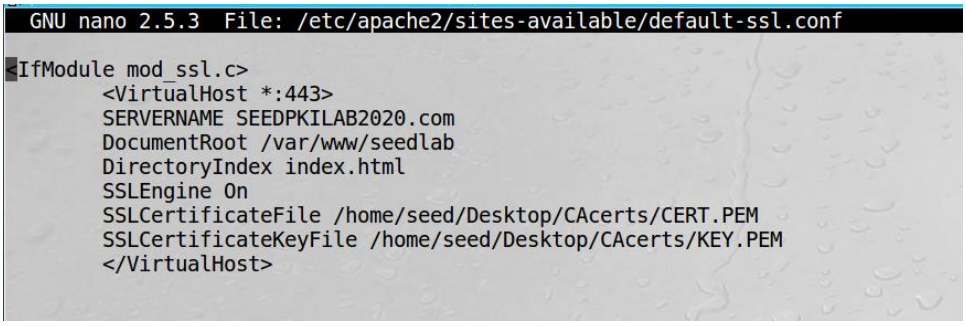




Was able to not get in because my key was not recognized by the CA.

Task 4

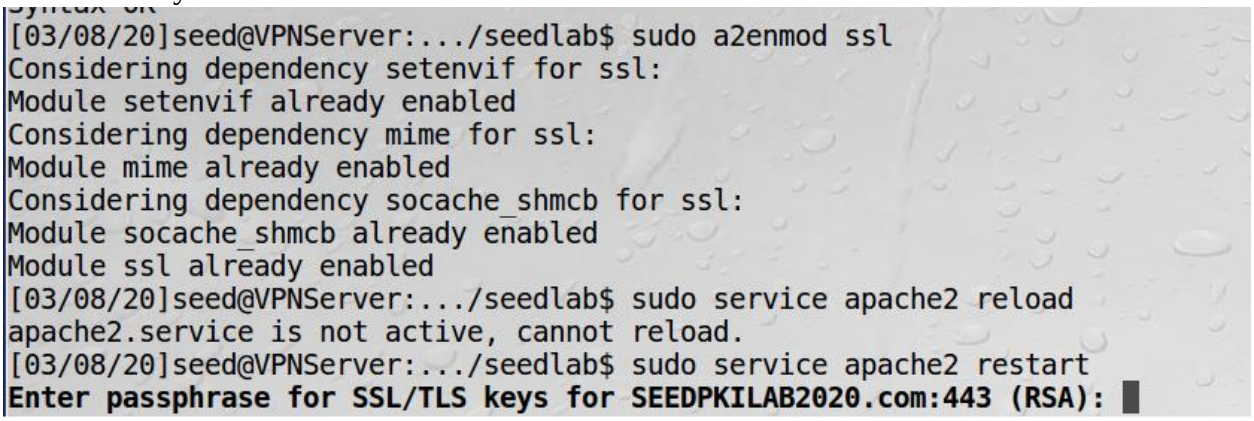
Setting up a real HTTPS SSL web server based on Apache web server by adding https protocol value within the ssl config file.



```
 GNU nano 2.5.3  File: /etc/apache2/sites-available/default-ssl.conf

<IfModule mod_ssl.c>
        <VirtualHost *:443>
        SERVERNAME SEEDPKILAB2020.com
        DocumentRoot /var/www/seedlab
        DirectoryIndex index.html
        SSLEngine On
        SSLCertificateFile /home/seed/Desktop/CAcerts/CERT.PEM
        SSLCertificateKeyFile /home/seed/Desktop/CAcerts/KEY.PEM
        </VirtualHost>
```

Restarting the web services, which ask for the key for the server to be verify by the CA

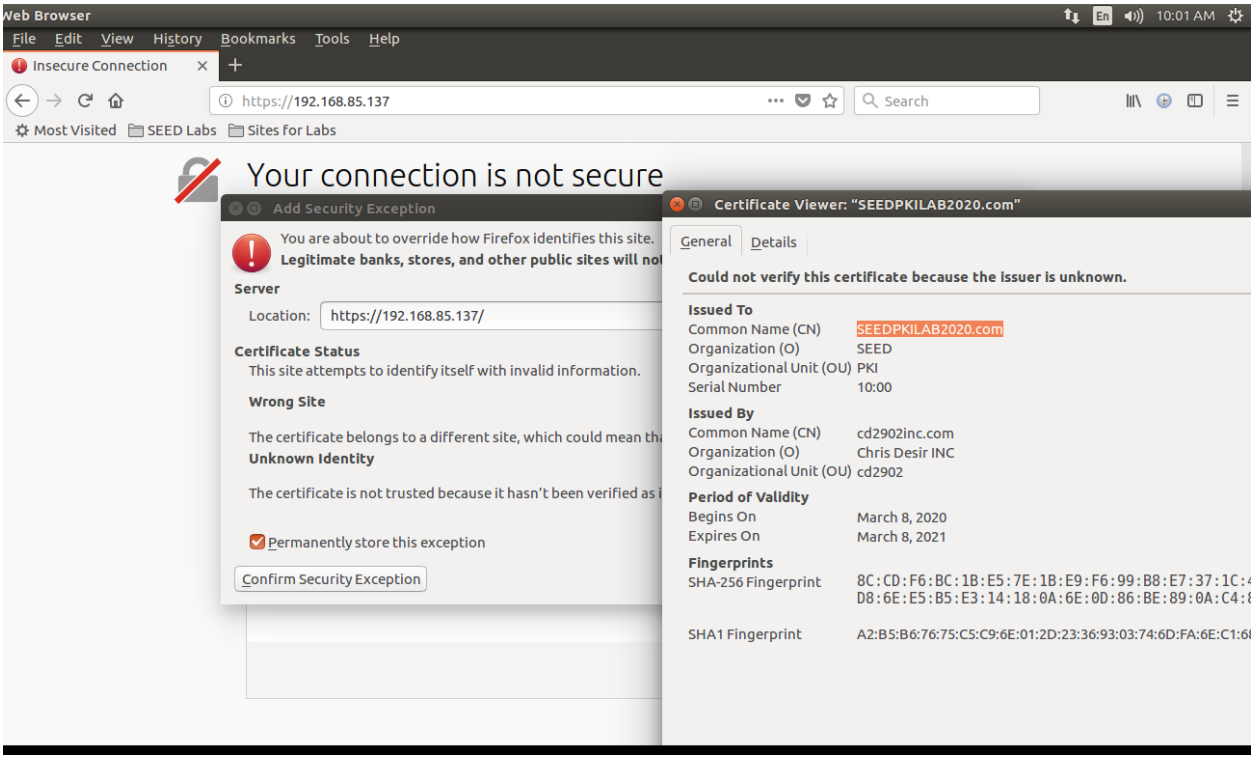
Added the keys



```
[03/08/20]seed@VPNServer:.../seedlab$ 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
[03/08/20]seed@VPNServer:.../seedlab$ sudo service apache2 reload
apache2.service is not active, cannot reload.
[03/08/20]seed@VPNServer:.../seedlab$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for SEEDPKILAB2020.com:443 (RSA):
```

restarting the required service linking with apache

Had to import the server certificate for my browser to manages the process of the CA verifying the server certificate.

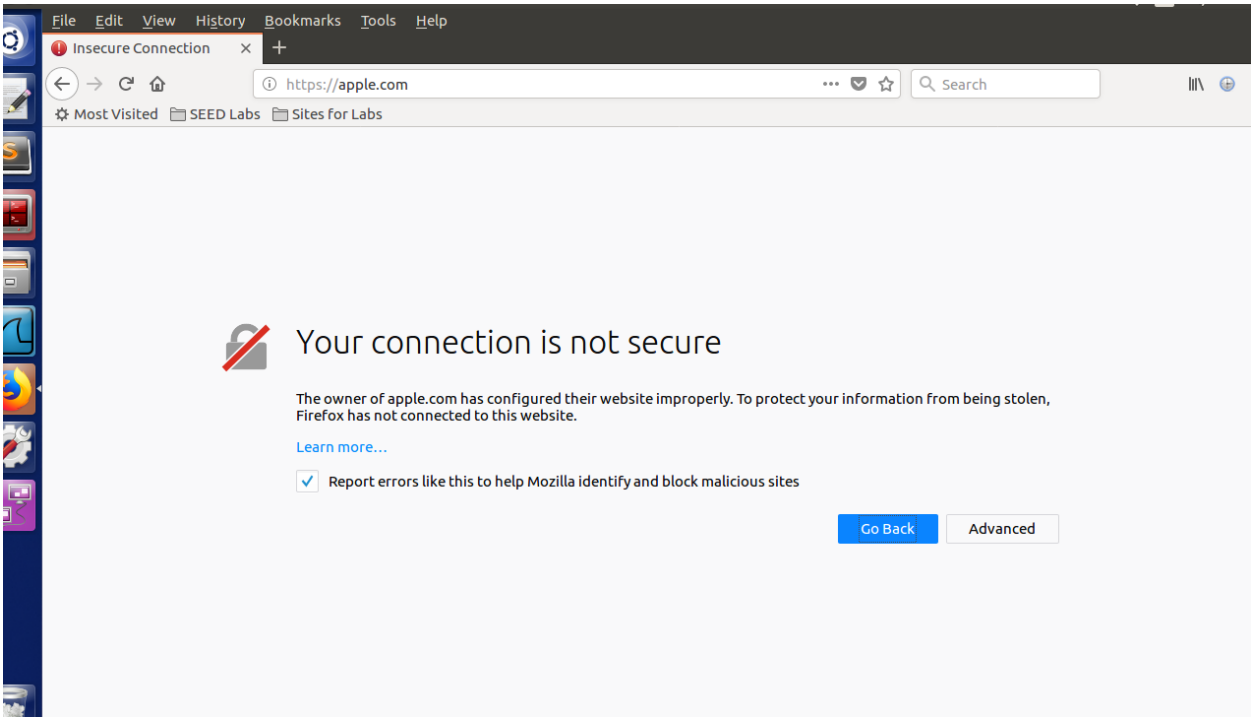


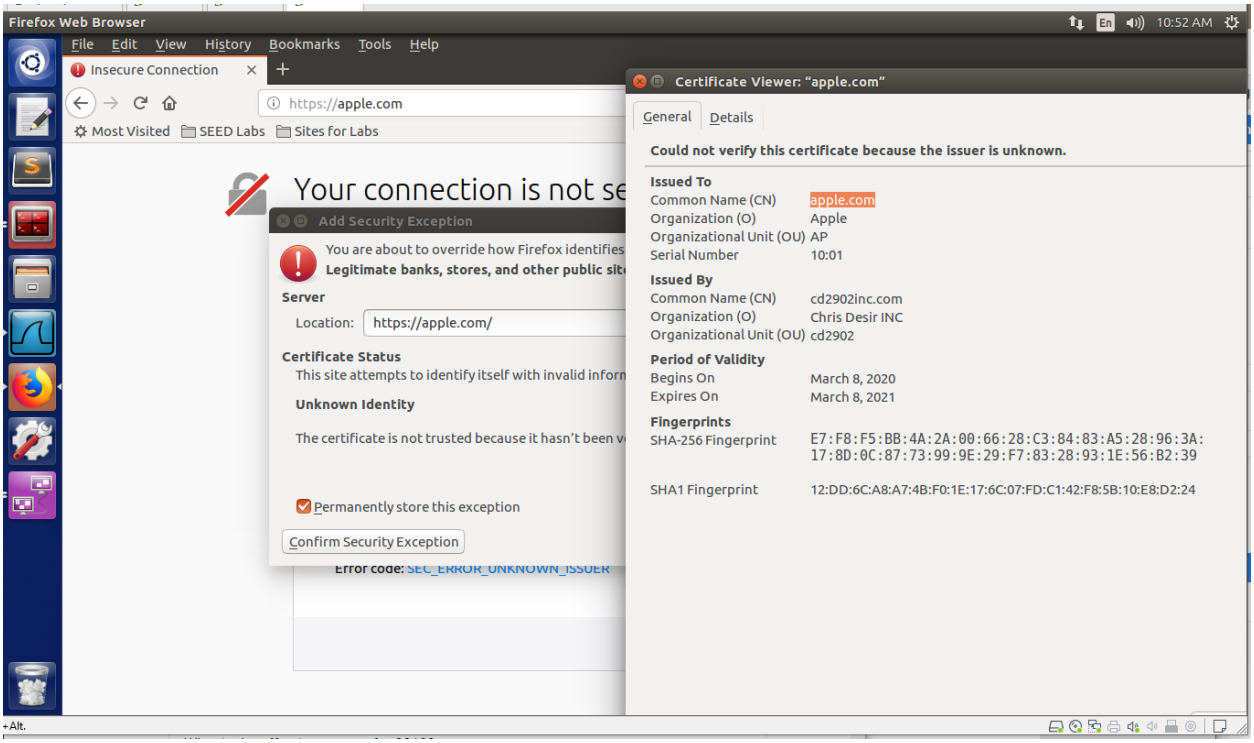
TESTED Complete

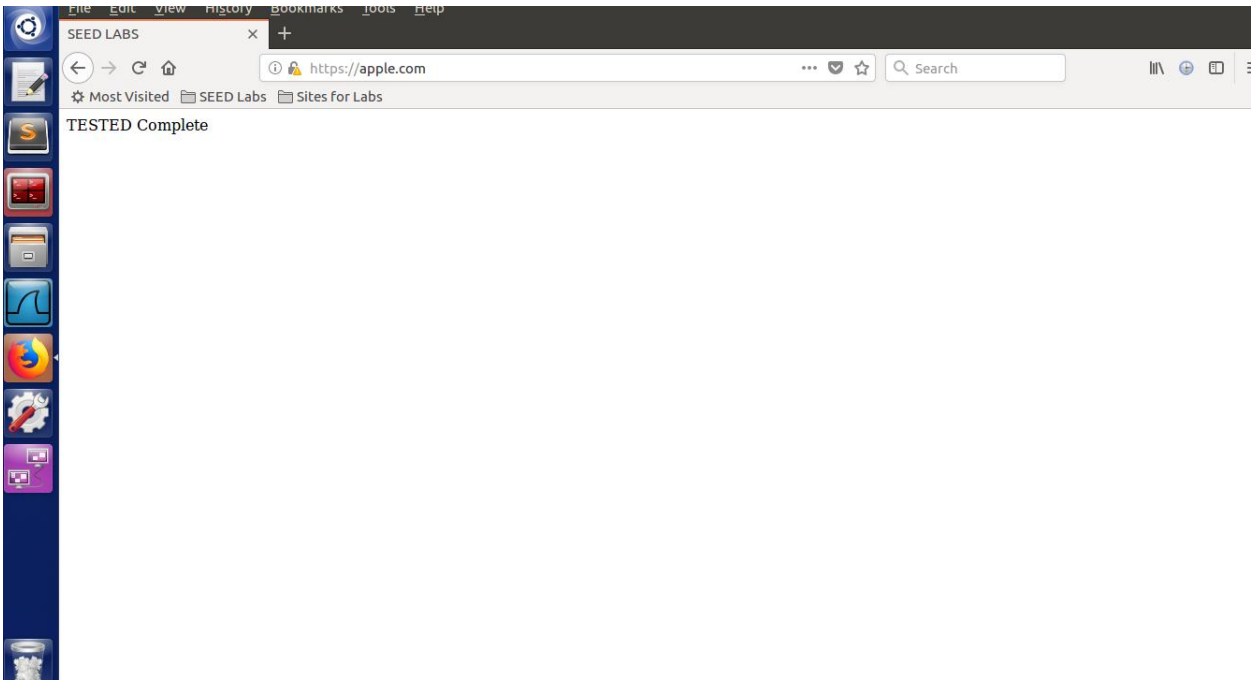
Task 5

Tried with the other lab same result but instead of using my apple key I used seedlab2 key and was able to get in with no issue by typing apple.com but the different I noticed is that the lock was not green as before which is suspicious.

```
<VirtualHost *:443>
SERVERNAME apple.com
DocumentRoot /var/www/seedlab
DirectoryIndex index.html
SSLEngine On
SSLCertificateFile /home/seed/Desktop/CAcerts/CERT2.pem
SSLCertificateKeyFile /home/seed/Desktop/CAcerts/KEY.PEM
```




Firefox Web Browser
10:52 AM
File Edit View History Bookmarks Tools Help
Insecure Connection
https://apple.com
Most Visited SEED Labs Sites for Labs
Your connection is not se
Add Security Exception
You are about to override how Firefox identifies
Legitimate banks, stores, and other public sit
Server
Location: https://apple.com/
Certificate Status
This site attempts to identify itself with invalid inform
Unknown Identity
The certificate is not trusted because it hasn't been v
Permanently store this exception
Confirm Security Exception
Error code: SEC_ERROR_UNKNOWN_ISSUER
Certificate Viewer: "apple.com"
General Details
Could not verify this certificate because the issuer is unknown.
Issued To
Common Name (CN) apple.com
Organization (O) Apple
Organizational Unit (OU) AP
Serial Number 10:01
Issued By
Common Name (CN) cd2902inc.com
Organization (O) Chris Desir INC
Organizational Unit (OU) cd2902
Period of Validity
Begins On March 8, 2020
Expires On March 8, 2021
Fingerprints
SHA-256 Fingerprint E7:F8:F5:BB:4A:2A:00:66:28:C3:84:83:A5:28:96:3A:
17:8D:0C:87:73:99:9E:29:F7:83:28:93:1E:56:B2:39
SHA1 Fingerprint 12:DD:6C:A8:A7:4B:F0:1E:17:6C:07:FD:C1:42:F8:5B:10:E8:D2:24
+Alt.


File Edit View History Bookmarks Tools Help
SEED LABS
https://apple.com
Search
Most Visited SEED Labs Sites for Labs
TESTED Complete

,

Task 6

Was able to authenticate apple.com by adding attacker IP to the hosts file, steal the SEED CA and requested a certificate for apple.com. By doing that I can make multiple machine authenticate them self because I took possession of the CA by creating an apple certificate.

this man in the middle (MITM) attack works when someone impersonates someone else.

```
sudo openssl req -new -key server.key -out apple.csr -config openssl.cnf
sudo openssl ca -in apple.csr  -out apple.crt
sudo openssl ca -in apple.csr  -out apple.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
cp server.key apple.pem
cat instagram.crt >> apple.pem
cat apple.crt  >> apple.pem
cp apple.crt CERT2.pem
sudo openssl ca -in apple.csr  -out apple.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
sudo nano /etc/apache2/sites-available/default-ssl.conf
sudo service apache2 restart
systemctl status apache2.service
ls
sudo nano /etc/apache2/sites-available/default-ssl.conf
sudo service apache2 restart
```

```
U6LS+9QNNLjP4MI1Ua9ZBaLdFqxsEF+uaUS10B/dSc22ywQf7mueFRyC03kD7Gkr
HPcd5YrBLn3hHJe46BXl8Ra71ejdZQb5wwUX0PLnQeu97mtMlKs4FUszdsc5dbOD
pP6L/fJb02d0HtaZ96lB4yi8br4jqbUAQdcSb0t+vjl81bQIG0vRV3Kpr0iAId6f
ZJN9CJagJLbWvSdkSKC2uUt6jSe91QE3pAWbv9a0Ax8Rqx68B8IRcXoiDq8DIjoH
++0ov90AgBSzlzreWTA9wDHOJ+jBiId4p3+rJsXbQ1ba3XLbTycuX+U6x3pUVkk4
9H2YNCBAAjEoireypSD7DuuFgLDDYV6e1xYqz+OXMFm1CWSntYmyIVZsLgZQInq0
5nxuTWzW/I7P9kaF+hcabbGEZdcvdrukfRXK+MfXgUXqlYPcq7Pb1Jrb/MvLdwW8
bo/8+gVW4EzDyVwCbi19yAZ0T7KSu3GuZMdT55droAkBv2URefsSiJBoTjAC1PpQ
MK63G/yTKvXMwxwjgFTP2f8ywZq9SRcK7IdwpwplLQ4EOXbUMN1GTRg8sLGfc8Tt
mHahimAJ3DExtDSyeKfTFpgONgcyU4BRMn0dwL8g0hUnvIR3Ag1zujj3RptKG64B
QgFsI+kvJdDXd+pUhC941SU6CMJvwD2fsxH58o2ulSqqAkjf3Rm5HRKNEWYqCkn4
63/skUvSf3Lc13tnBGqwbN3kqUN8MkC485CgdwxXt3Mp3mRzU0NrhomKv+3EEDhI
-----END RSA PRIVATE KEY-----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4097 (0x1001)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=New York, L=NYC, O=Chris Desir INC, OU=cd2902, CN=
2inc.com/emailAddress=cd2902@nyu.edu
        Validity
            Not Before: Mar  8 22:37:26 2020 GMT
            Not After : Mar  8 22:37:26 2021 GMT
        Subject: C=US, ST=NY, L=NYC, O=Apple, OU=AP, CN=apple.com
```