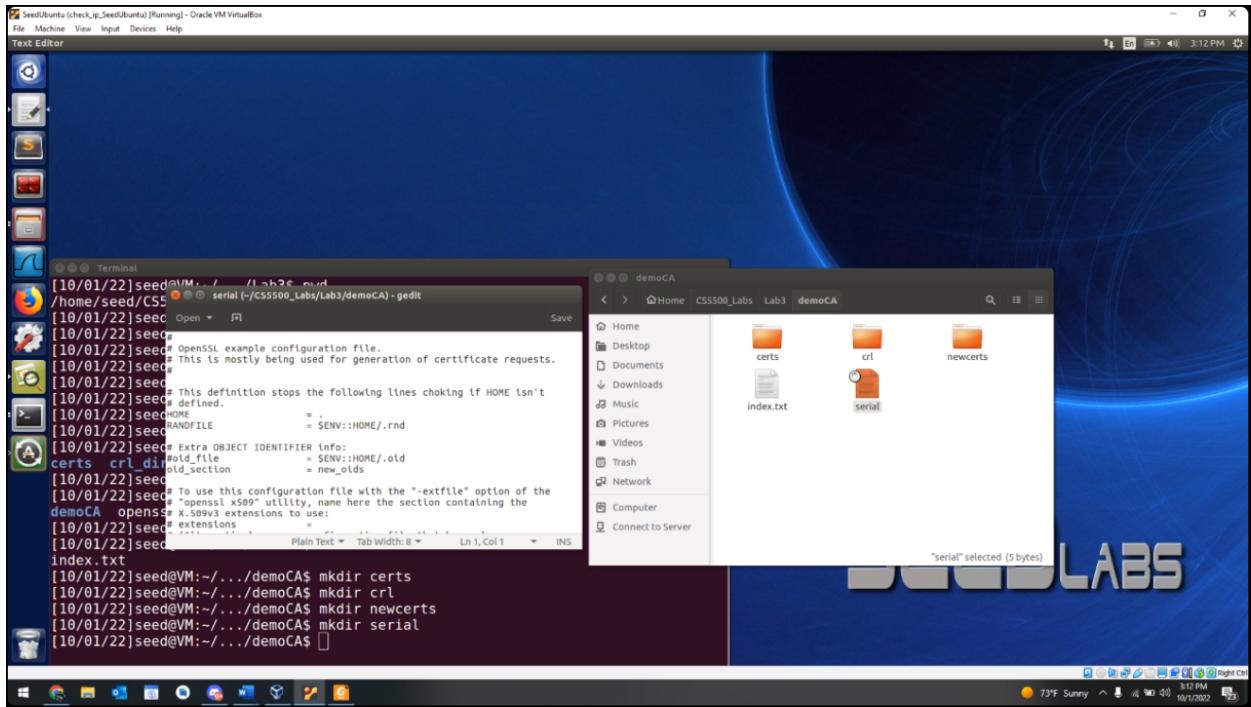


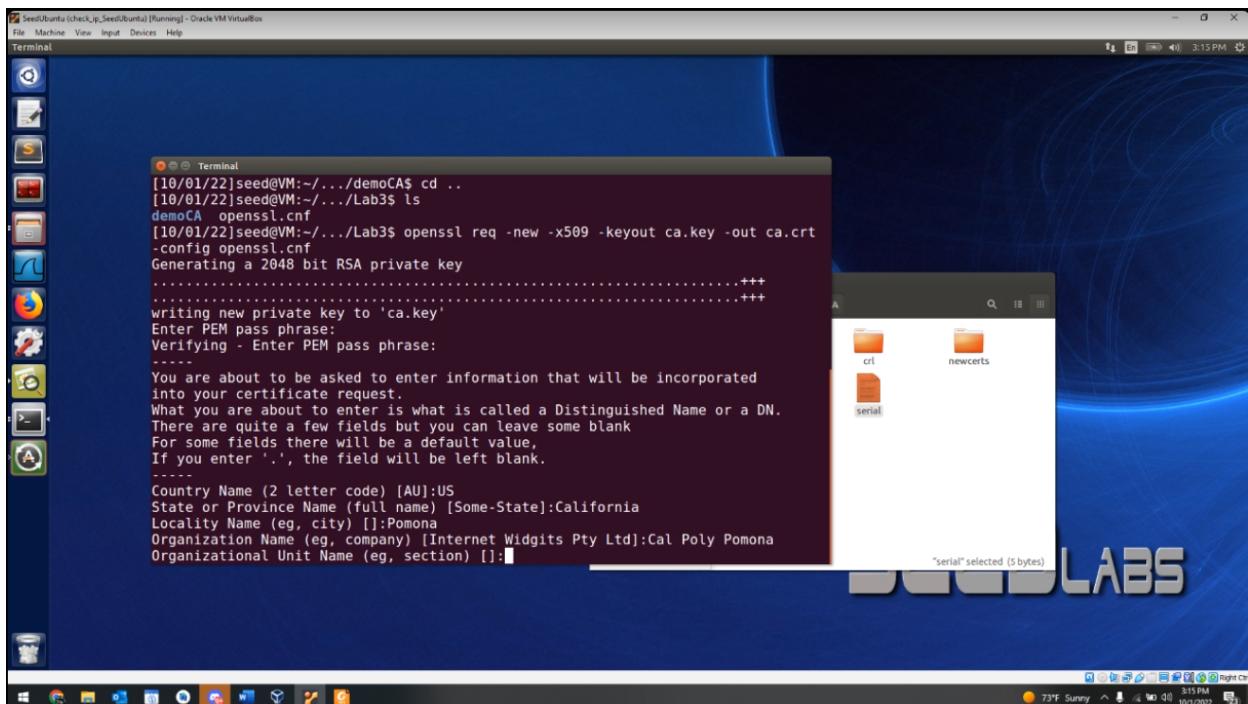
Group 8: Kevin Vo, Mason Godfrey, Thanh Le

CS 5500 – Lab 3: Public Key Infrastructure

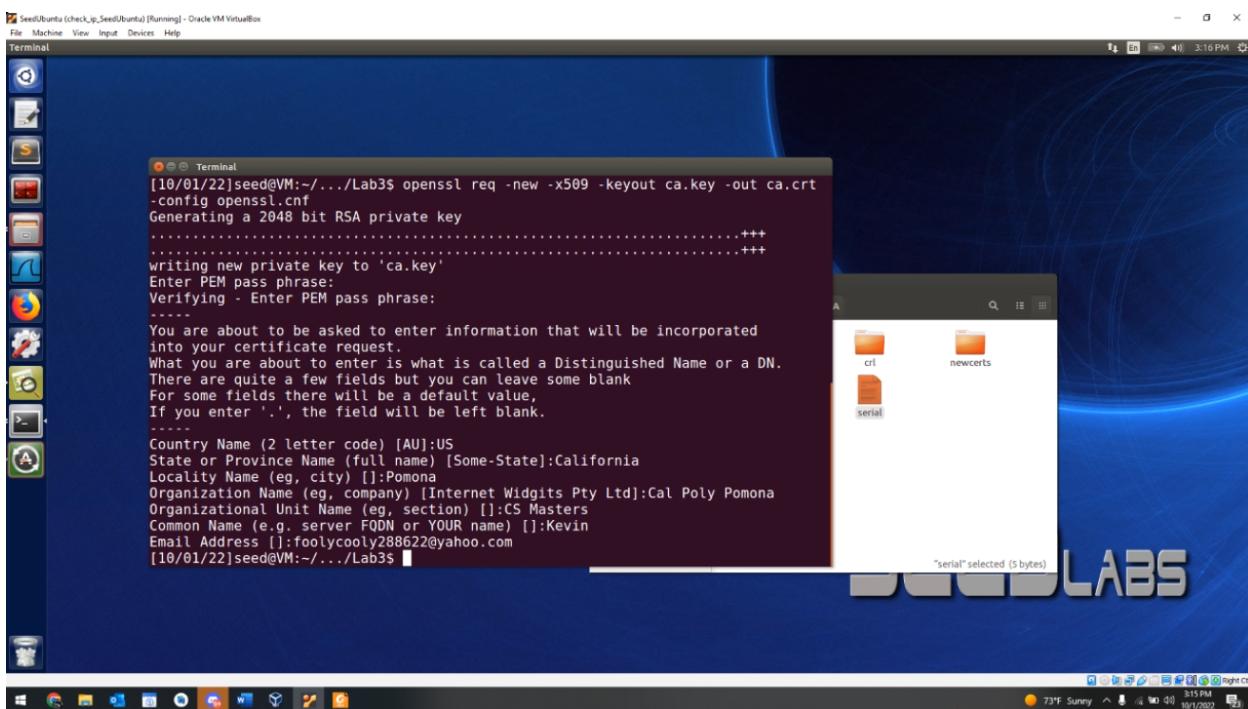
Task 1: Becoming a Certificate Authority



Here we created the proper directories that we were instructed to do. We were curious as the rest of the instructions did not have us use them beyond this point. It was only at the end of the lab that we were able to find auto-generated files from the instructions in this lab. Beyond this, our serial file had the necessary commands to proceed further.



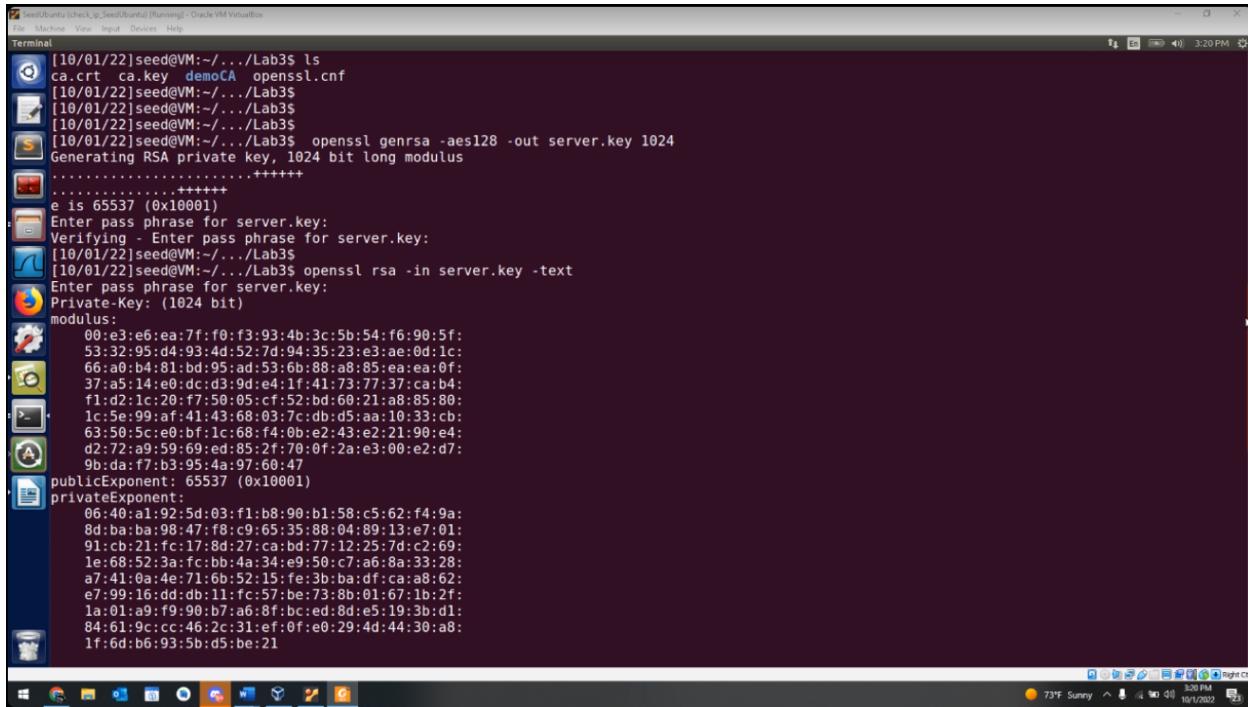
At this point we took the liberty to fill out the Country Name, State, Locality Name, Organization Name, its unit, and so on. We felt this part of the lab is crucial as we learn by mistake that Common Name is a significant value.



Here is the private key and our CA with our key within it.

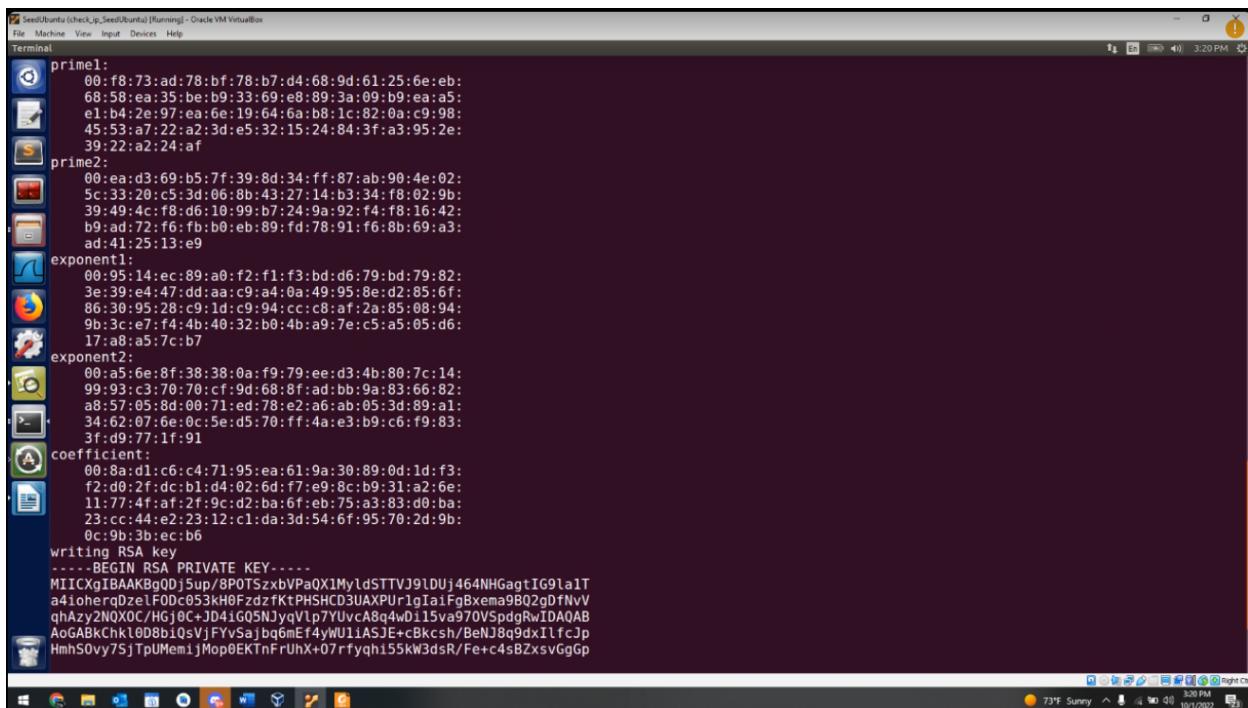
Task 2: Creating a Certificate for SEEDPKILab2020.com

Step 1



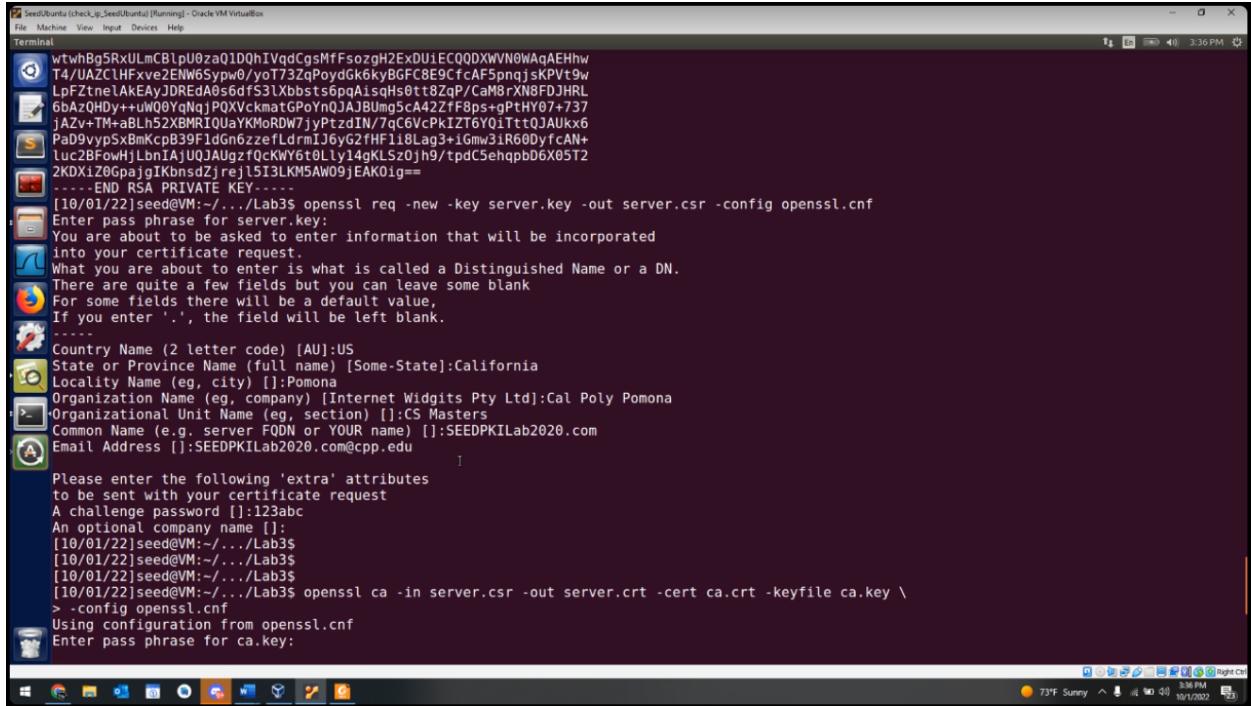
```
[10/01/22]seed@VM:~/.../Lab3$ ls
ca.crt  ca.key  demoCA  openssl.cnf
[10/01/22]seed@VM:~/.../Lab3$ [10/01/22]seed@VM:~/.../Lab3$ [10/01/22]seed@VM:~/.../Lab3$ [10/01/22]seed@VM:~/.../Lab3$ openssl genrsa -aes128 -out server.key 1024
Generating RSA private key, 1024 bit long modulus
.....+++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
[10/01/22]seed@VM:~/.../Lab3$ [10/01/22]seed@VM:~/.../Lab3$ openssl rsa -in server.key -text
Enter pass phrase for server.key:
Private-Key: (1024 bit)
modulus:
 00:e3:e6:ea:7f:f0:f3:93:4b:3c:5b:54:f6:90:5f:
 53:32:95:d4:93:4d:52:7d:94:35:23:e3:ae:0d:1c:
66:a0:b4:81:bd:95:ad:53:6b:88:a8:85:ea:ea:0f:
37:a5:14:e0:dc:d3:9d:e4:1f:41:73:77:37:cb:4:
f1:d2:1c:20:f7:50:05:cf:52:bd:60:21:a8:85:80:
1c:5e:99:af:41:43:68:03:7c:db:d5:aa:10:33:cb:
63:50:5c:e0:bf:1c:68:f4:0b:e2:43:e2:21:90:e4:
d2:72:a9:59:69:ed:85:f1:70:0f:2a:e3:00:e2:d7:
9b:da:f7:b3:95:4a:97:60:47
publicExponent: 65537 (0x10001)
privateExponent:
 06:40:a1:92:5d:03:f1:b8:90:b1:58:c5:62:f4:9a:
8d:ba:ba:98:47:f8:c9:65:35:88:04:89:13:e7:01:
91:cb:21:fc:17:8d:27:ca:bd:77:12:25:7d:c2:69:
1e:68:52:3a:fc:bb:4a:34:e9:50:c7:a6:8a:33:28:
a7:41:0a:4e:71:6b:52:15:fe:3b:ba:df:ca:a8:62:
e7:99:16:dd:db:11:fc:57:be:73:80:01:67:1b:2f:
1a:01:a9:f9:90:b7:a6:8f:bc:ed:8d:e5:19:3b:d1:
84:61:9c:cc:46:2c:31:ef:0f:e0:29:4d:44:30:a8:
1f:6d:b6:93:5b:d5:be:21
```

Now, we complete a similar process for the server certificate, filling out the information needed to tie the certificate to a website.



```
primal:
 00:f8:73:ad:78:bf:78:b7:d4:68:9d:61:25:6e:eb:
68:58:ea:35:be:b9:33:69:e8:89:3a:09:b9:caa5:
e1:b4:2e:97:ea:6e:19:64:6a:b8:1c:82:0a:c9:98:
45:53:a7:22:a2:3d:e5:32:15:24:84:3f:a3:95:2e:
39:22:a2:24:af
prime2:
 00:ea:d3:69:b5:7f:39:8d:34:ff:87:ab:90:4e:02:
5c:33:20:c5:3d:06:8b:43:27:14:b3:34:ff:80:92:9b:
39:49:4c:f8:d6:10:99:b7:24:9a:92:f4:f8:16:42:
b9:ad:72:f6:fb:b0:eb:89:fd:78:91:f6:8b:69:a3:
ad:41:25:13:e9
exponent:
 00:95:14:ec:89:a0:f2:f1:f3:bd:d6:79:bd:79:82:
3e:39:e4:47:dd:aa:c9:a4:0a:49:95:8e:d2:85:6f:
86:30:95:28:c9:1d:c9:94:cc:c8:af:2a:85:08:94:
9b:3c:e7:f4:4b:40:32:b0:4b:a9:7e:c5:a5:05:d6:
17:a8:a5:7c:b7
exponent2:
 00:a5:6e:8f:38:38:0a:f9:79:ee:d3:4b:80:7c:14:
99:93:c3:70:70:cf:9d:68:8f:ad:bb:9a:83:66:82:
a8:57:05:8d:00:71:ed:78:e2:a6:ab:05:3d:89:a1:
34:62:07:6e:0c:5e:d5:70:ff:4a:e3:b9:c6:f9:83:
3f:d9:77:1f:91
coefficient:
 00:8a:d1:c6:c4:71:95:ea:61:9a:30:89:0d:1d:f3:
f2:d0:2f:dc:b1:d4:02:6d:77:e9:8c:b9:31:a2:6e:
11:77:4f:a2:f2:9c:d2:ba:6f:eb:75:a3:83:00:ba:
23:cc:44:e2:23:12:c1:da:3d:54:6f:95:70:2d:9b:
0c:9b:3b:ec:b6
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgDj5up/BP0TSzbvPaqXIMyldTTVJ9lDUj464NHagtIG9laIT
a4ioherqDzelF0Dc053kH0FzdzKtPHSHCD3UAXPUrlqIa1FqBxema9BQ2gDfnvV
qhAzy2N0XOC/Hgj0C+JD4iG05NJyqVlp7YUvcA8q4wD115va970VSpdgRwIDAQAB
AoGABkChkl0D8b1oSvJFYoSajbq6mEt4ywU1ASJE+cBkcsH/BeNJBq9dx1lfcJp
HmhSOvy75jTpUMemijMop0EKTnFrUhX+07rfyqh155kW3dsR/Fe+c4sBZxsVggGp
```

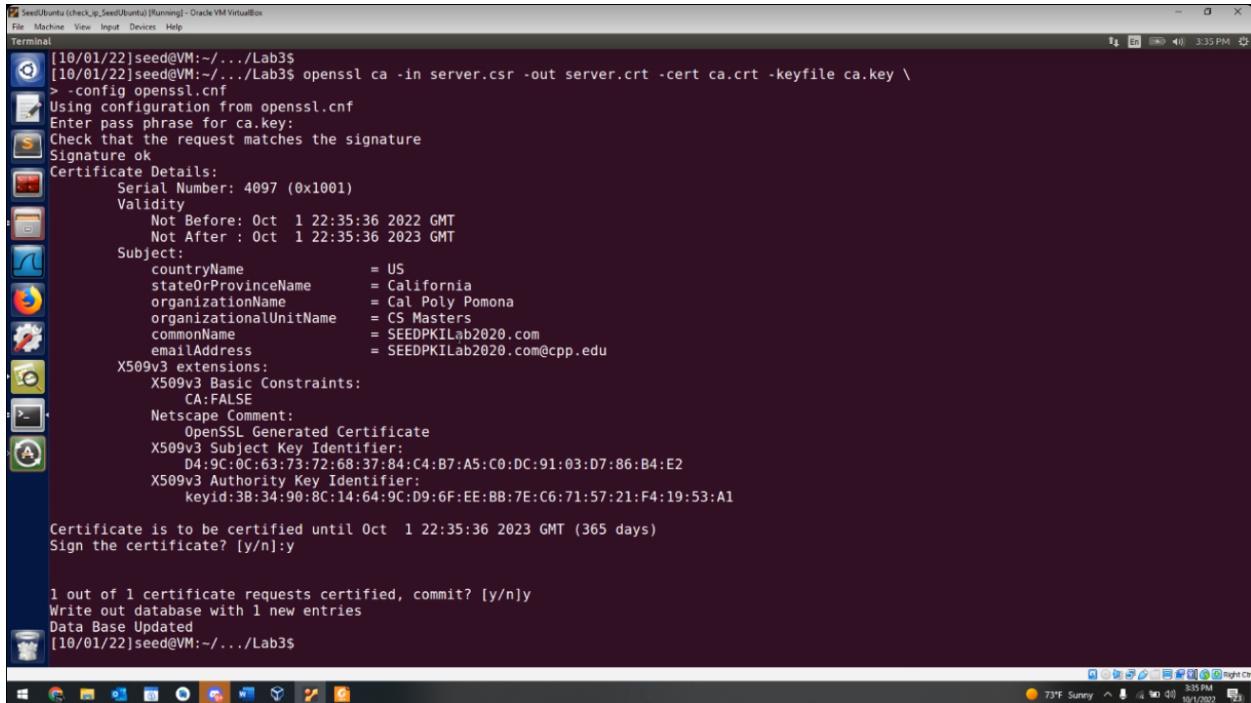
Step 2



```
Terminal
[10/01/22]seed@VM:~/.../Lab3$ 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]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Pomona
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Cal Poly Pomona
Organizational Unit Name (eg, section) []:CS Masters
Common Name (e.g. server FQDN or YOUR name) []:SEEDPKILab2020.com
Email Address []:SEEDPKILab2020.com@cpp.edu
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:123abc
An optional company name []:
[10/01/22]seed@VM:~/.../Lab3$ [10/01/22]seed@VM:~/.../Lab3$ [10/01/22]seed@VM:~/.../Lab3$ [10/01/22]seed@VM:~/.../Lab3$ 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:
```

We then request for the signature to be signed by our certificate authority to validate it.

Step 3



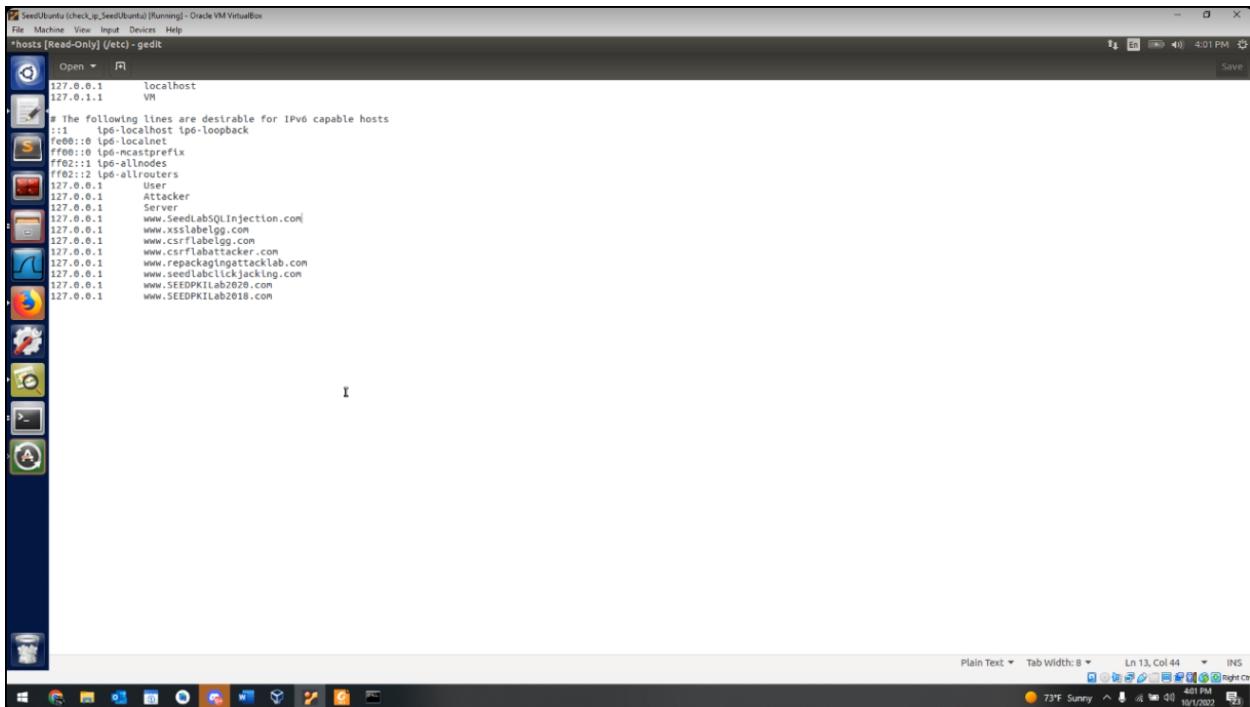
```
Terminal
[10/01/22]seed@VM:~/.../Lab3$ [10/01/22]seed@VM:~/.../Lab3$ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key \
> -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 4097 (0x1001)
    Validity
        Not Before: Oct  1 22:35:36 2022 GMT
        Not After : Oct  1 22:35:36 2023 GMT
    Subject:
        countryName          = US
        stateOrProvinceName  = California
        organizationName     = Cal Poly Pomona
        organizationalUnitName = CS Masters
        commonName            = SEEDPKILab2020.com
        emailAddress          = SEEDPKILab2020.com@cpp.edu
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
        Netscape Comment:
            OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
        D4:9C:63:73:72:68:37:84:C4:B7:A5:C0:DC:91:03:D7:86:B4:E2
    X509v3 Authority Key Identifier:
        keyid:3B:34:90:8C:14:64:9C:D9:FEE:BB:7E:C6:71:57:21:F4:19:53:A1
Certificate is to be certified until Oct  1 22:35:36 2023 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
[10/01/22]seed@VM:~/.../Lab3$
```

We then generate the certificate itself by signing the certificate into our certificate authority.

Task 3: Deploying Certificate in an HTTPS Web Server

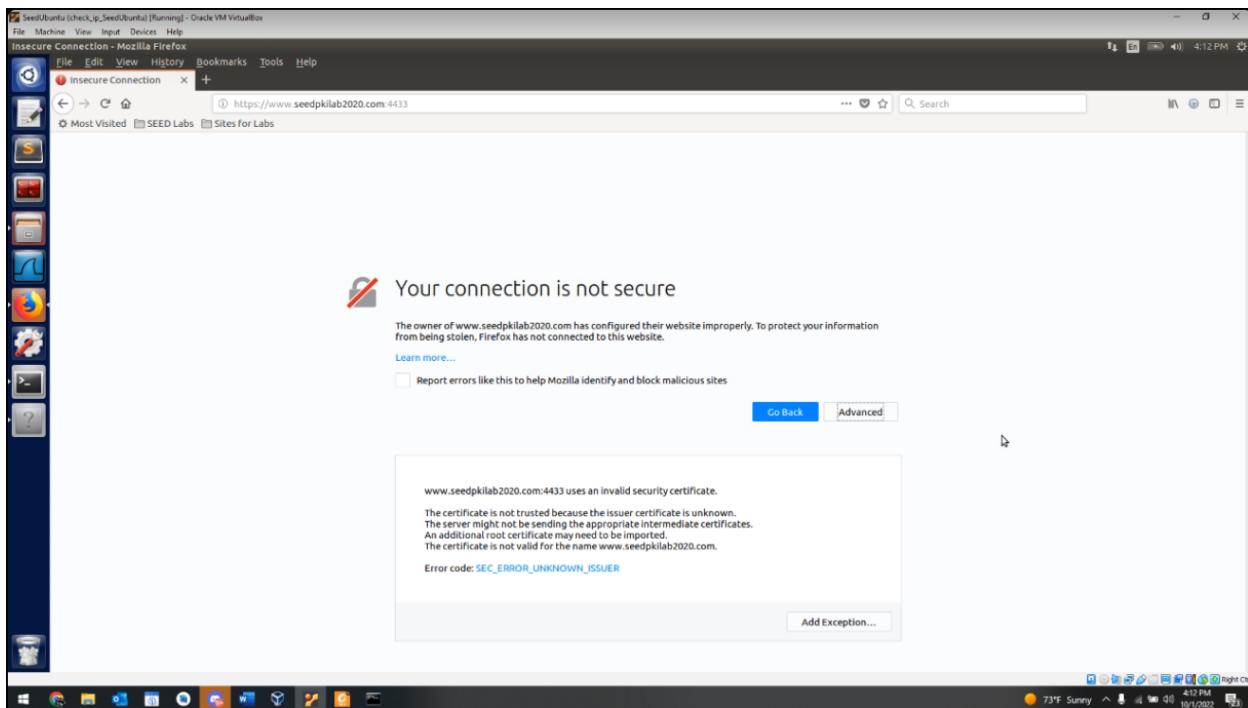
Step 1



```
SeedUbuntu [chck_ip_SeedUbuntu] (Running) - Oracle VM VirtualBox  
File Machine View Input Devices Help  
hosts [Read-Only] (etc) - gedit  
Open ▾ [ ]  
127.0.0.1 localhost  
127.0.1.1 VM  
  
# The following lines are desirable for IPv6 capable hosts  
::1:1 ip6-localhost ip6-loopback  
ff00::0 ip6-allnodes  
ff02::1 ip6-allrouters  
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.xsslabel0g.com  
127.0.0.1 www.cookieclicker.com  
127.0.0.1 www.csrflabattacker.com  
127.0.0.1 www.repackagingattacklab.com  
127.0.0.1 www.seedLabclickjacking.com  
127.0.0.1 www.SEEDPKILab2020.com  
127.0.0.1 www.SEEDPKILab2018.com
```

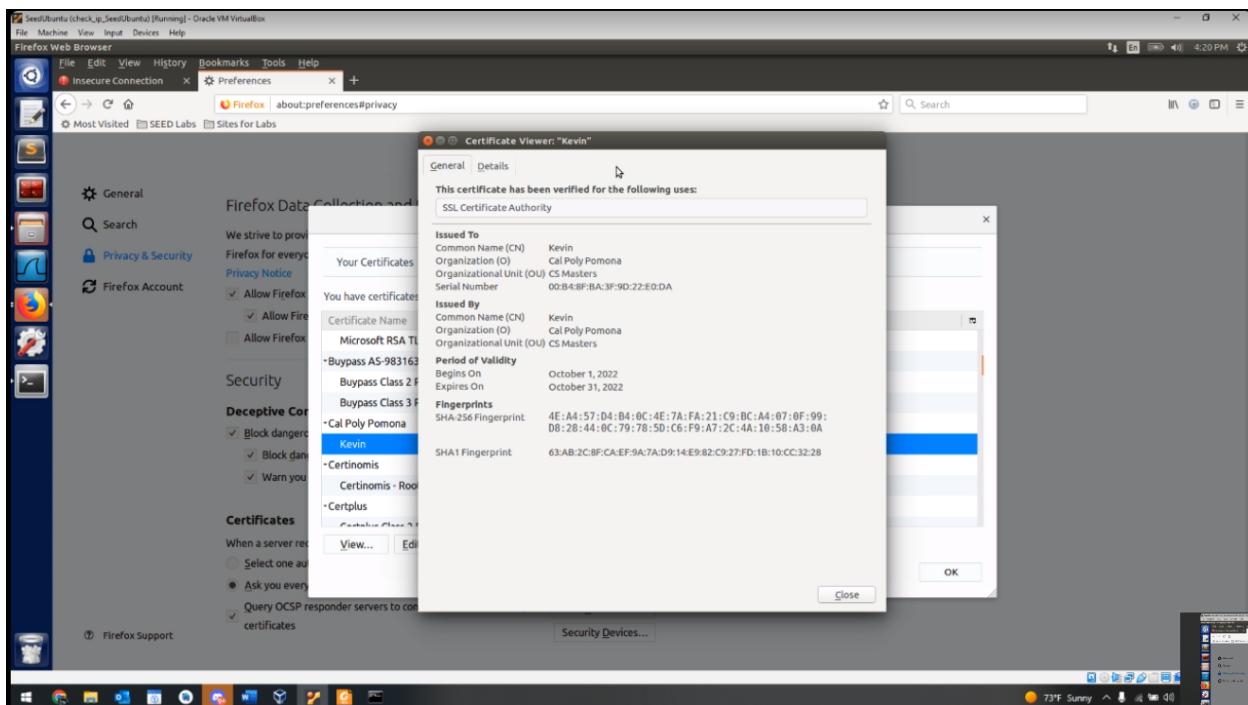
We included the required "www.SEEDPKILab2020.com" and "www.SEEDPKILab2018.com" just in case since the instructions were unclear and occasionally switches out these two URLs. This is done to have our computer recognize this URL/name.

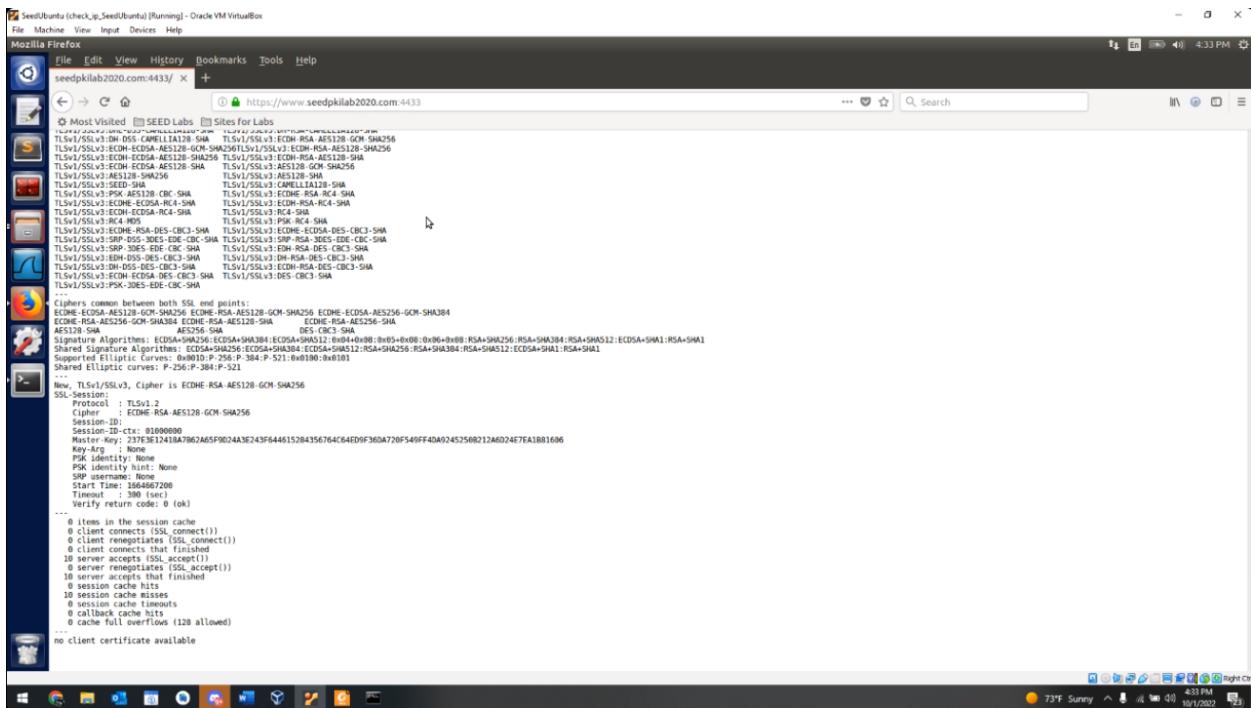
Step 2



This point it took us a while to figure out, but we had to include “www.” within the URL shown in the picture above for it to traverse to this page before you. It found that our connection was insecure since it doesn’t know our security certificate.

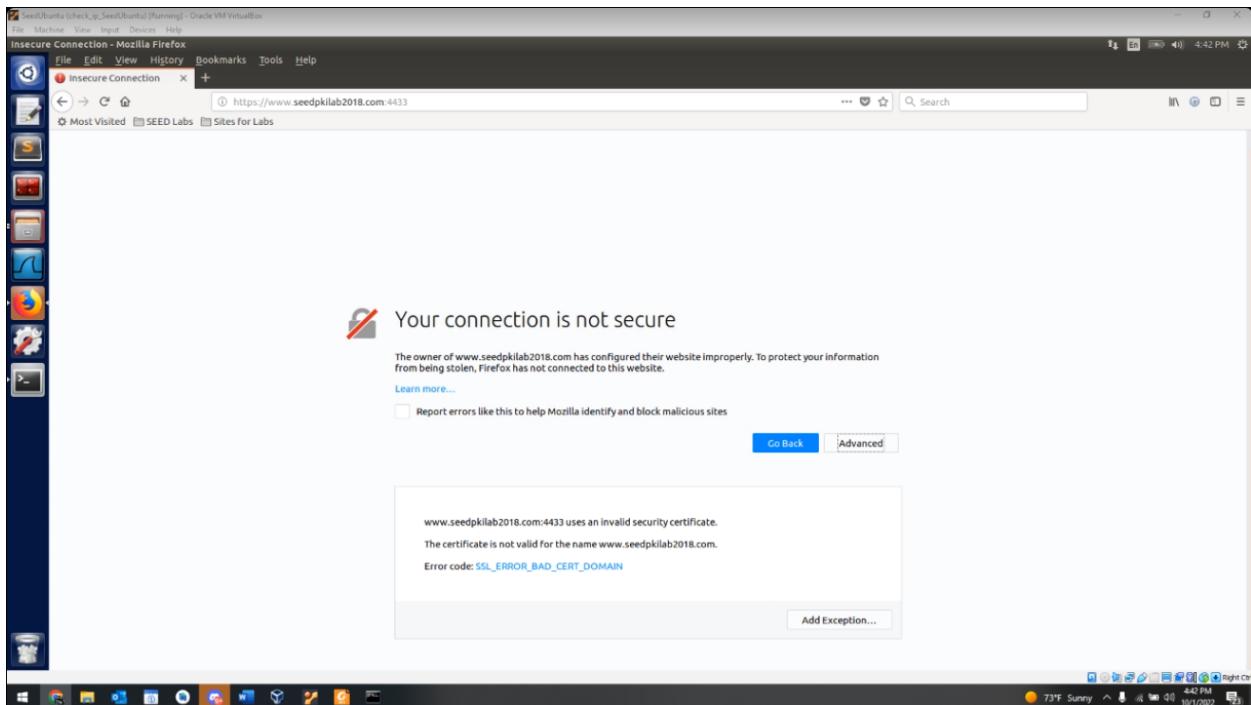
Step 3





To counter the SEC_ERROR_UNKNOWN_ISSUER, we imported our own certificate into FireFox for it to display the contents of the image above.

Step 4



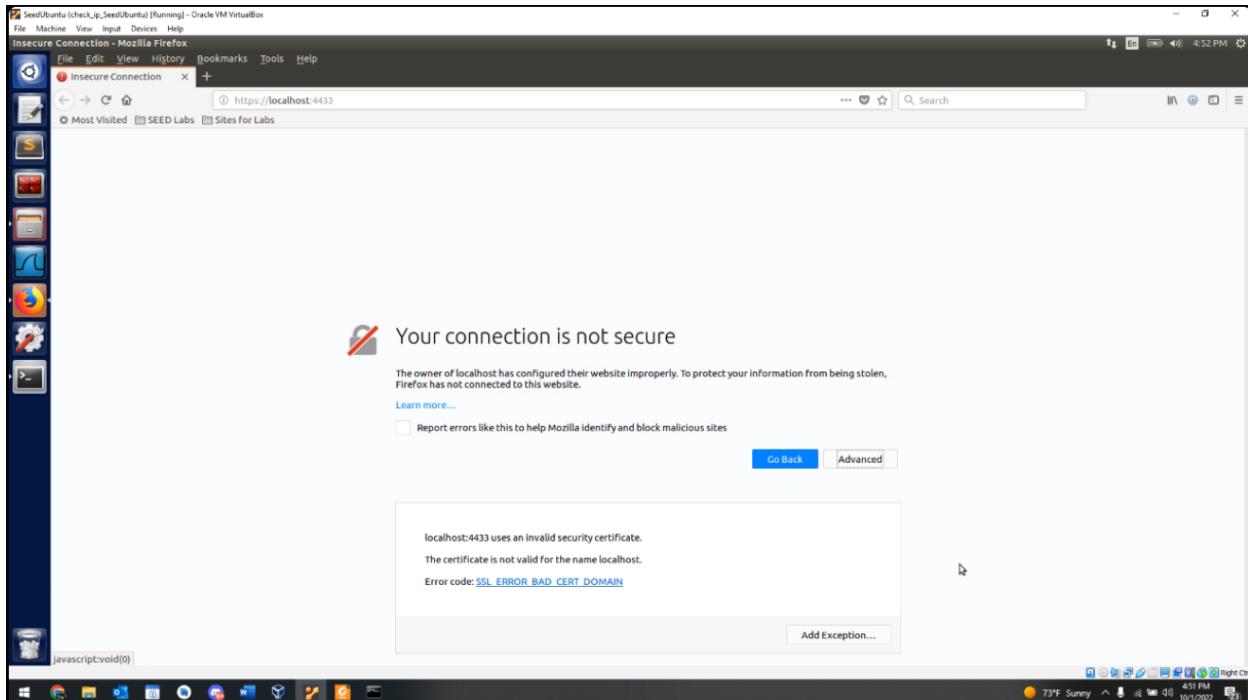
```

[10/01/22]seed@VM-.../Lab3$ [10/01/22]seed@VM-.../Lab3$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
unable to load server certificate private key file
3071080128:error:00680A8:asn1 encoding routines:ASN1_CHECK_TLEN:wrong tag:asn1 dec.c:1197:
3071080128:error:006C03A:asn1 encoding routines:ASN1_D2I_EX_PRIMITIVE:nested asn1 error:asn1_dec.c:765:
3071080128:error:00680A8:asn1 encoding routines:ASN1_CHECK_TLEN:wrong tag:asn1 dec.c:1197:
3071080128:error:04093004:rsa routines:OLD RSA PRIV DECODE:RSA lib:rsa ameth.c:119:
3071080128:error:00680A8:asn1 encoding routines:ASN1_CHECK_TLEN:wrong tag:asn1 dec.c:1197:
3071080128:error:007803A:asn1 encoding routines:ASN1_ITEM_EX_D2I:nested asn1 error:asn1_dec.c:374:Type=X509_ALGOR
3071080128:error:008303A:asn1 encoding routines:ASN1_TEMPLATE_NOEXP_D2I:nested asn1 error:asn1_dec.c:697:Field=pkeyalg, Type=PKCS8_PRIV_KEY_
INFO
3071080128:error:0907B00D:PEM routines:PEM_READ_BIO_PRIVATEKEY:ASN1 lib:pem_pkey.c:141:
[10/01/22]seed@VM-.../Lab3$ [10/01/22]seed@VM-.../Lab3$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
unable to load server certificate private key file
3070916288:error:007207B:asn1 encoding routines:ASN1_get_object:header too long:asn1_lib.c:157:
3070916288:error:0068066:asn1 encoding routines:ASN1_CHECK_TLEN:bad object header:asn1 dec.c:1185:
3070916288:error:006C03A:asn1 encoding routines:ASN1_TEMPLATE_NOEXP_D2I:nested asn1 error:asn1_dec.c:765:
3070916288:error:04093004:rsa routines:OLD RSA PRIV DECODE:RSA lib:rsa ameth.c:119:
3070916288:error:00680A8:asn1 encoding routines:ASN1_CHECK_TLEN:wrong tag:asn1 dec.c:1197:
3070916288:error:007803A:asn1 encoding routines:ASN1_ITEM_EX_D2I:nested asn1 error:asn1_dec.c:374:Type=X509_ALGOR
3070916288:error:008303A:asn1 encoding routines:ASN1_TEMPLATE_NOEXP_D2I:nested asn1 error:asn1_dec.c:697:Field=pkeyalg, Type=PKCS8_PRIV_KEY_
INFO
3070916288:error:0907B00D:PEM routines:PEM_READ_BIO_PRIVATEKEY:ASN1 lib:pem_pkey.c:141:
[10/01/22]seed@VM-.../Lab3$ [10/01/22]seed@VM-.../Lab3$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
Using default temp DH parameters
ACCEPT
ACCEPT
ACCEPT
ACCEPT
```

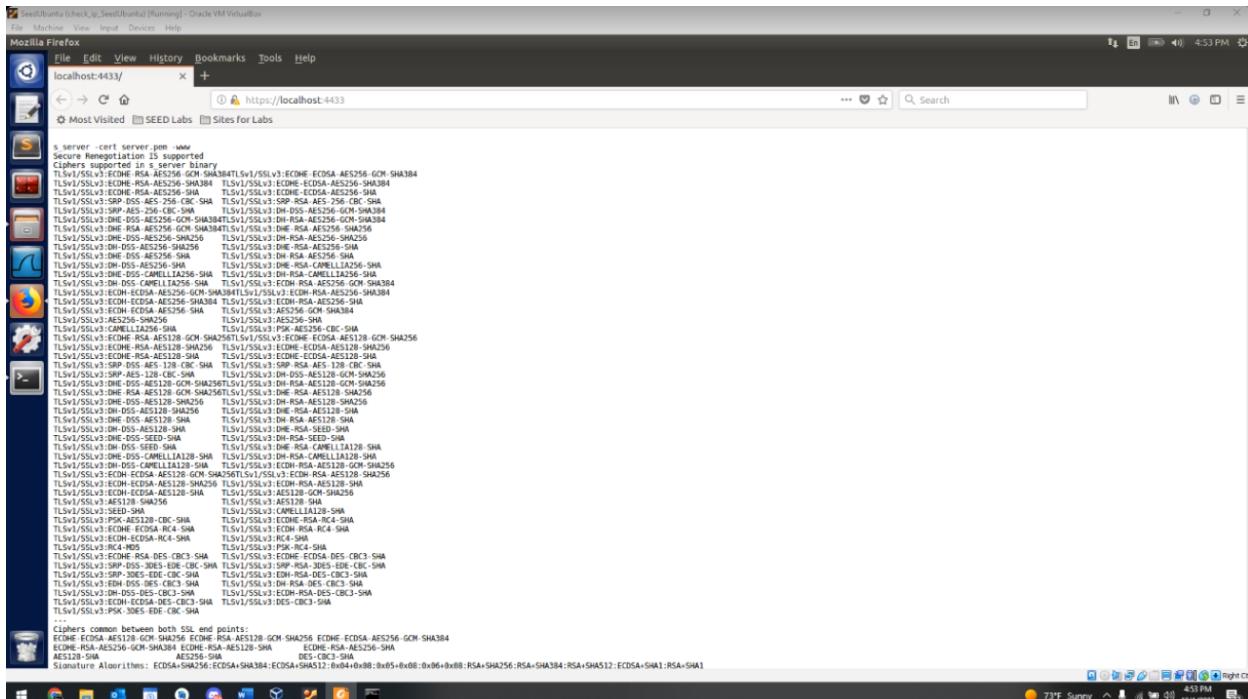
We found that Step 4 was most amusing since we discovered that altering certain characters within the server.pem file can have more weight than others. This can be seen in the image above as our initial change provided us an error when starting up the server with server.pem. However, when we went back and try to change a different character within it, the server worked.

Protocol	Cipher Suite	Key Exchange	Encryption	Hash	MAC
TLSv1/SSLv3	ECDSA-P384-SHA256	ECDSA	AES-256-GCM	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	ECDSA-P256-SHA256	ECDSA	AES-128-GCM	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	ECDSA-P384-SHA	ECDSA	AES-256-GCM	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	ECDSA-P256-SHA	ECDSA	AES-128-GCM	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	DHE-RSA-AES256-GCM-SHA256	DHE-RSA	AES-256-GCM	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	DHE-RSA-AES128-GCM-SHA256	DHE-RSA	AES-128-GCM	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	DHE-RSA-AES256-SHA256	DHE-RSA	AES-256-SHA	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	DHE-RSA-AES128-SHA256	DHE-RSA	AES-128-SHA	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	DSS-SEED-SHA		SEED	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	DSS-CAMELLIA128-SHA		CAMELLIA128-SHA	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	DSS-CAMELLIA128-SHA		CAMELLIA128-SHA	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	ECDH-ECDSA-AES128-GCM-SHA256	ECDH-ECDSA	AES-128-GCM	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	ECDH-ECDSA-AES256-GCM-SHA256	ECDH-ECDSA	AES-256-GCM	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	ECDH-ECDSA-AES128-SHA	ECDH-ECDSA	AES-128-SHA	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	ECDH-ECDSA-AES256-SHA	ECDH-ECDSA	AES-256-SHA	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	SEED-SHA		SEED-SHA	SHA-256	HKDF-SHA-256
TLSv1/SSLv3	PSK-AES128-CBC-SHA		PSK-AES128-CBC-SHA	SHA-256	HKDF-SHA-256

Above is evidence that we did what was asked of us to have our server running and that our connection is secure.



However, when we tried it under localhost, it was not secure as shown above.



Regardless, after bypassing the warning, we were able to visit the site and see that the contents were the same.

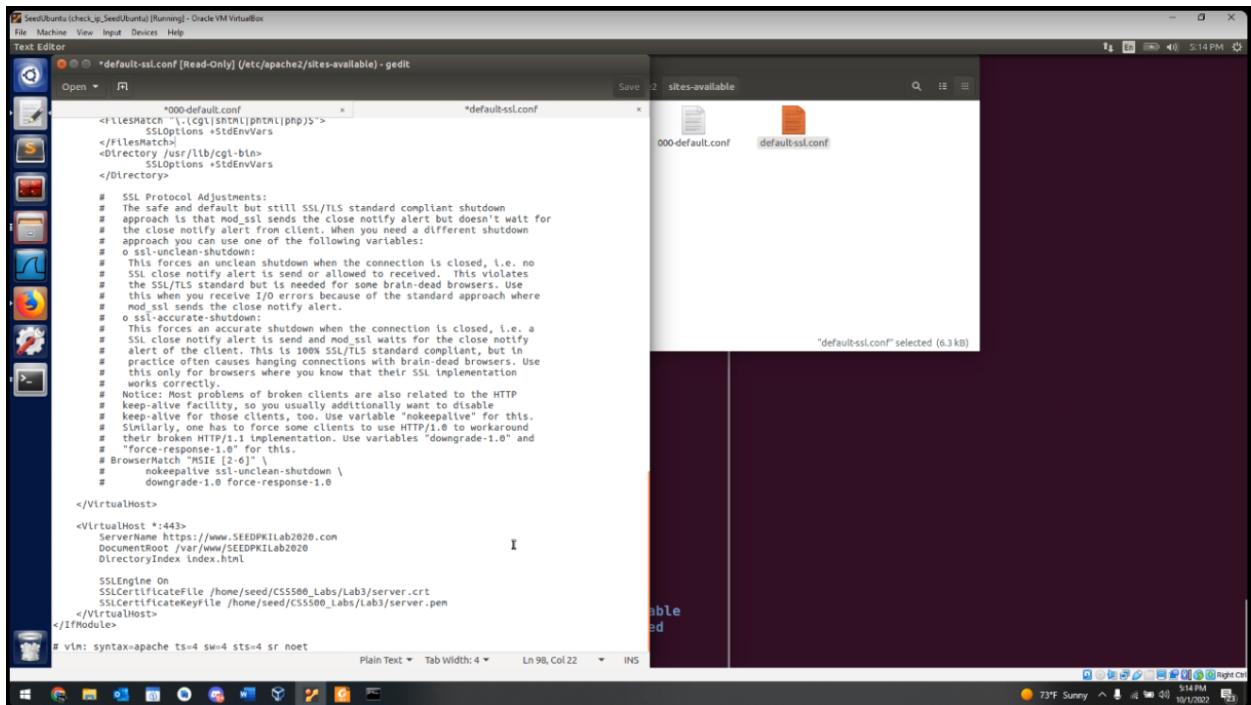
1. Modify a single byte of server.pem, and restart the server, and reload the URL. What do you observe? Make sure you restore the original server.pem afterward. Note: the server

may not be able to restart if certain places of server.pem is corrupted; in that case, choose another place to modify.

- a. Since we never applied the certificate to the 2018 site, modifying a byte of server.pem didn't affect it. While the server didn't start at first due to corruption, we were able to modify a new byte that didn't seem to affect the 2020 site's stability after restarting the server.
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.
 - a. When we connected via the localhost, we were told that our connection was insecure but were able to connect after bypassing the security warning.

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

1. Please use the above example as guidance to set up an HTTPS server for SEEDPKILab2020.com. Please describe the steps that you have taken, the contents that you add to Apache's configuration file, and the screenshots of the final outcome showing that you can successfully browse the HTTPS site.
 - a. We modified default-ssl.conf to add in a new entry, as this is how HTTPS connections can be generated. We created a new folder for the DocumentRoot path, and placed an index.html file within that had a short message about our group (group 8). We also linked our key and certification files inside the default-ssl.conf.



```
*000-default.conf *default-ssl.conf
<FilesMatch "\.(cgi|html|php|js)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
    SSLOptions +StdEnvVars
</Directory>

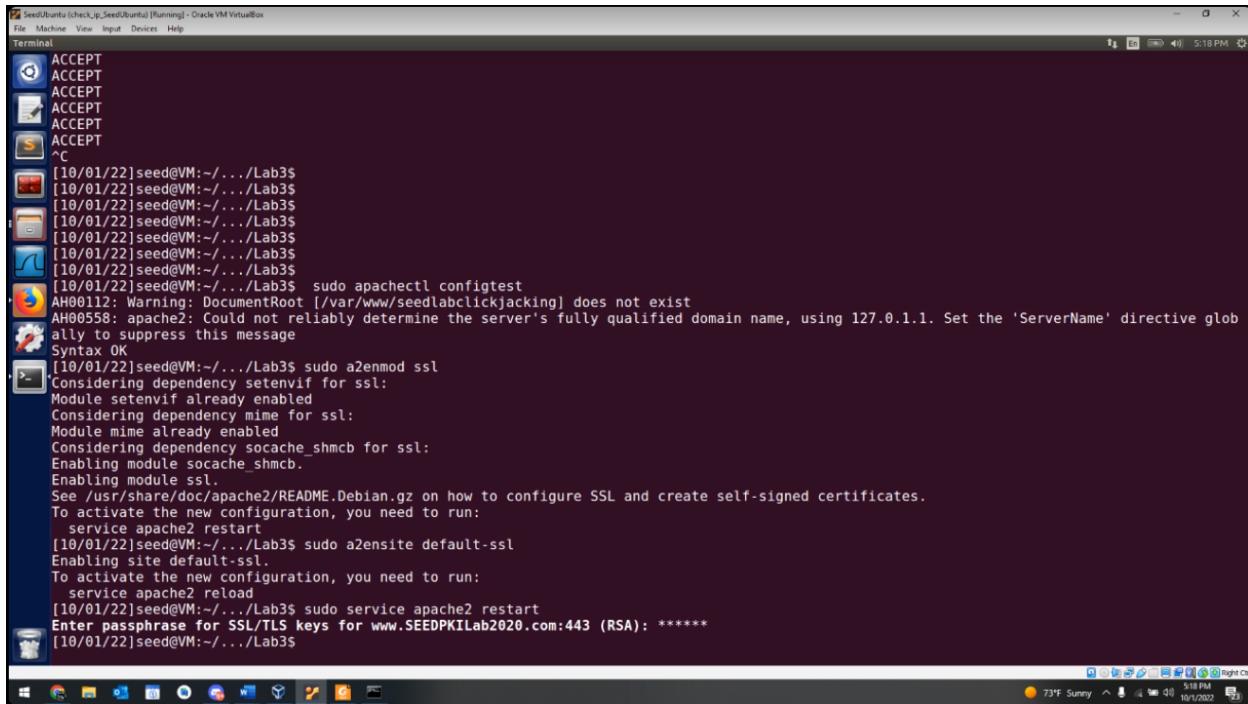
# SSL Protocol Adjustments:
# This safe and robust but still SSL/TLS standard compliant shutdown
# approach is that mod_ssl sends the close notify alert but doesn't wait for
# the close notify alert from client. When you need a different shutdown
# approach you can use one of the following variables:
# o ssl_stapling_force_iframes:
#     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 waits for 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 from client. This is the standard SSL/TLS shutdown behavior, 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 ability, 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 https://www.SEEDPKILab2020.com
    DocumentRoot /var/www/SEEDPKILab2020
    DirectoryIndex index.html

    SSLEngine On
    SSLCertificateFile /home/seed/c55500_Labs/Lab3/server.crt
    SSLCertificateKeyFile /home/seed/c55500_Labs/Lab3/server.pem
</VirtualHost>
</IfModule>

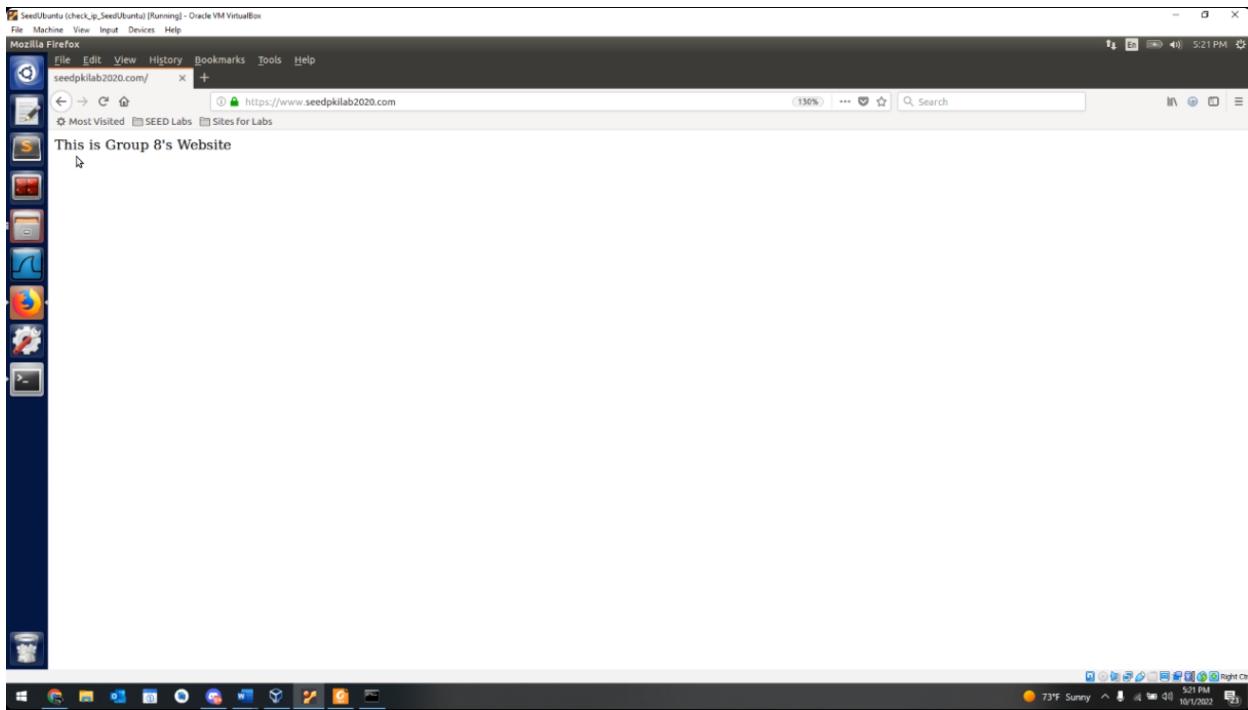
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

We wrote the website SSL information into default-ssl.conf, ensuring that the site with the expected URL is generated and that the contents of the website are from the index file.



```
SeedUbuntu (check_ip_SeedUbuntu) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
ACCEPT
ACCEPT
ACCEPT
ACCEPT
ACCEPT
ACCEPT
^c
[10/01/22]seed@VM:~/.../Lab3$ 
[10/01/22]seed@VM:~/.../Lab3$ 
[10/01/22]seed@VM:~/.../Lab3$ 
[10/01/22]seed@VM:~/.../Lab3$ 
[10/01/22]seed@VM:~/.../Lab3$ 
[10/01/22]seed@VM:~/.../Lab3$ 
[10/01/22]seed@VM:~/.../Lab3$ 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 globally to suppress this message
Syntax OK
[10/01/22]seed@VM:~/.../Lab3$ 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:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  service apache2 restart
[10/01/22]seed@VM:~/.../Lab3$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  service apache2 reload
[10/01/22]seed@VM:~/.../Lab3$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for www.SEEDPKILab2020.com:443 (RSA): *****
[10/01/22]seed@VM:~/.../Lab3$
```

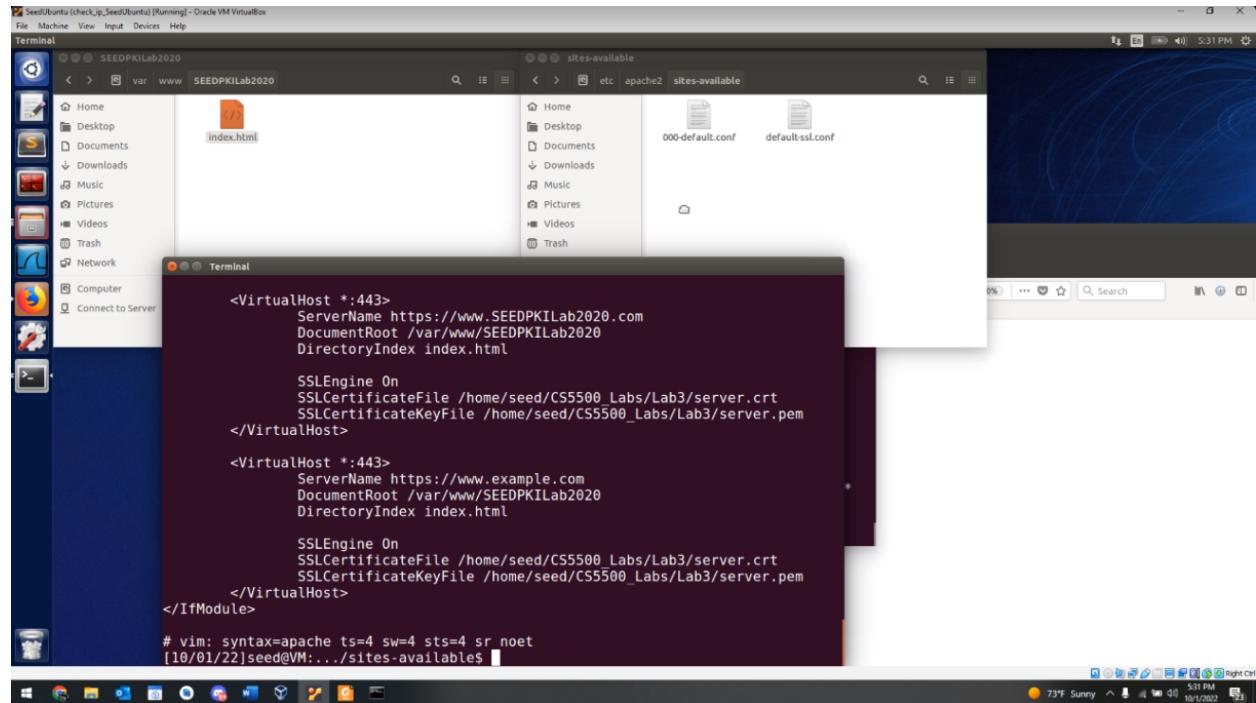
We prepared Apache to have it host the websites locally.



We went to the website, and found that it was being hosted locally and had the expected index contents.

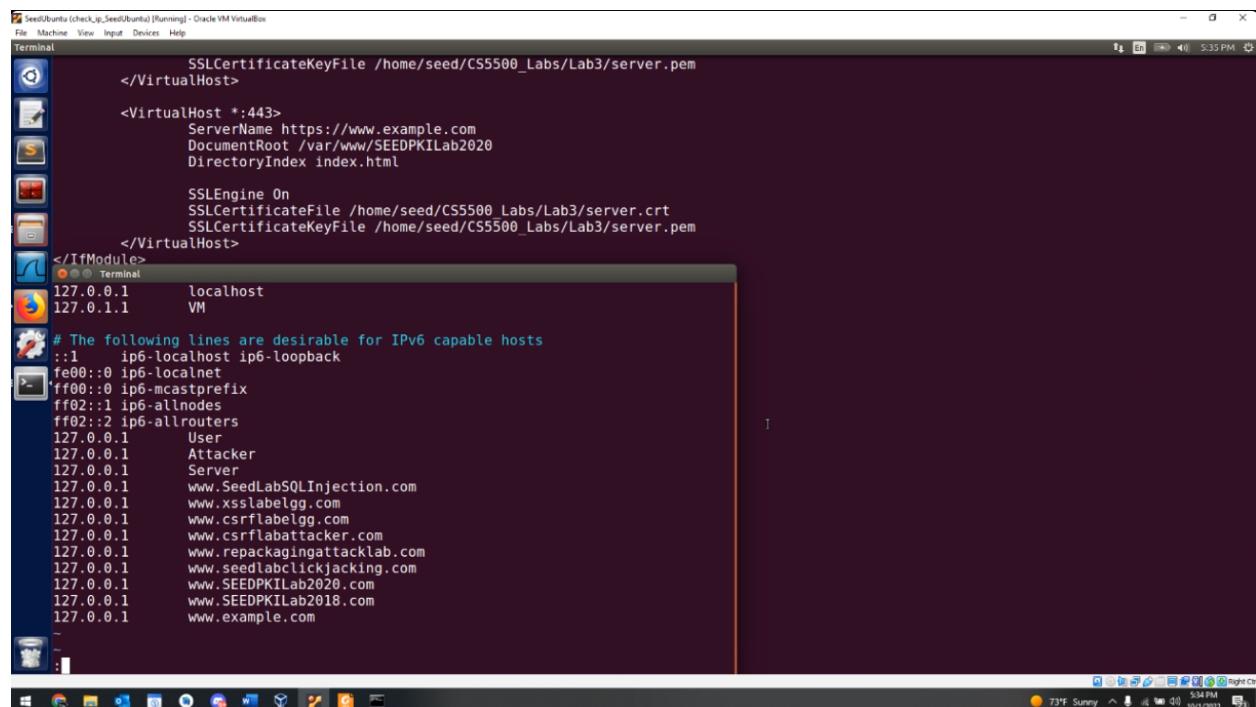
Task 5: Launching a Man-In-The-Middle Attack

Step 1



We modified a configuration file to tell Apache that example.com should have the same contents as the other website.

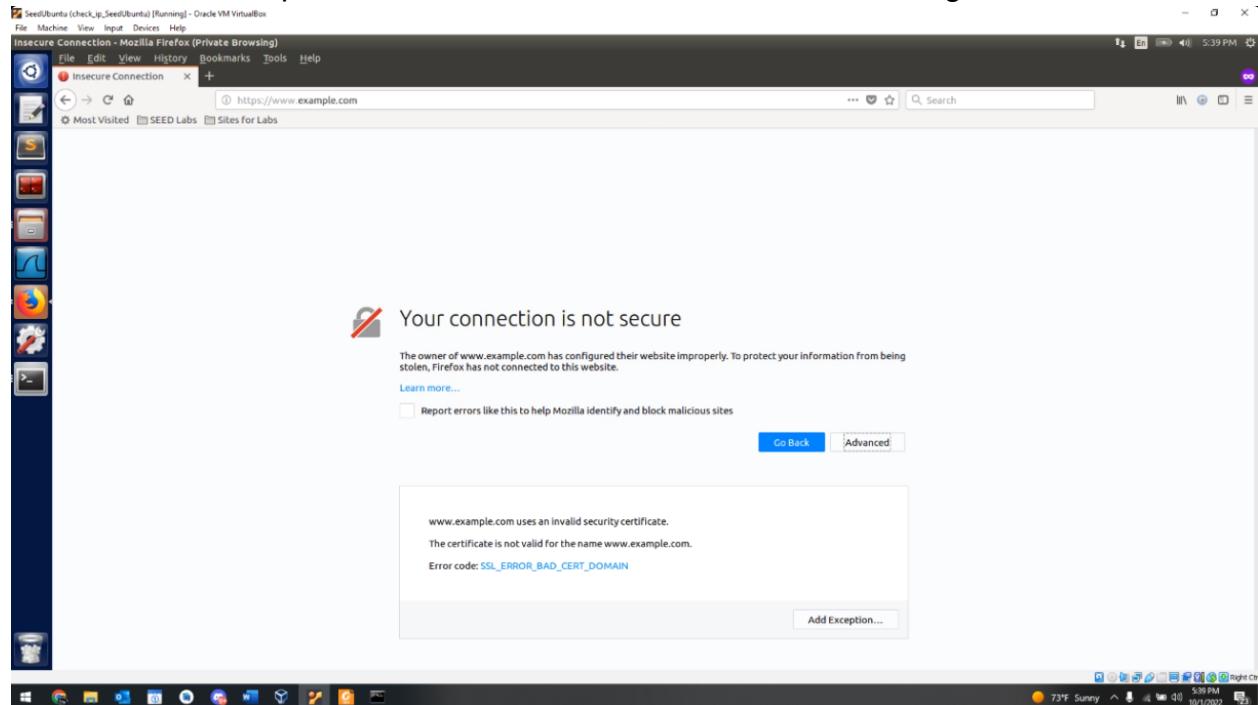
Step 2



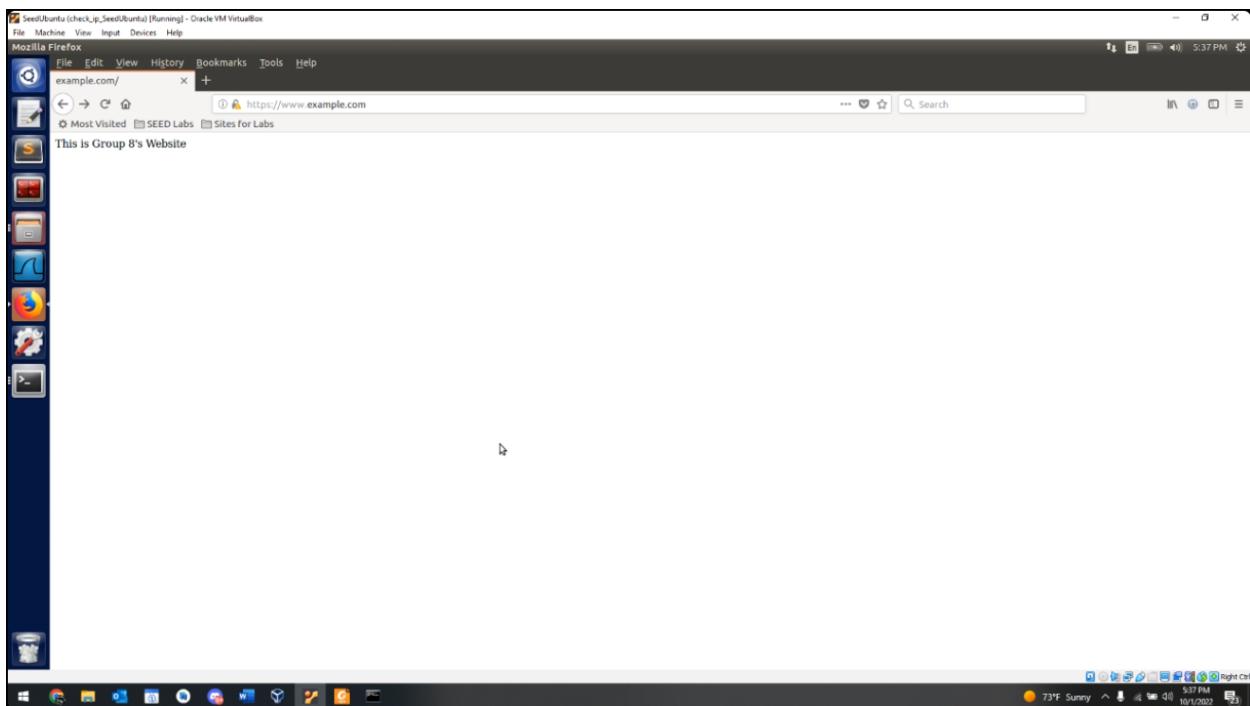
We overwrote the DNS information for example.com to redirect locally.

Step 3

1. With everything set up, now visit the target real website, and see what your browser would say. Please explain what you have observed.
 - a. Firefox realized that the certificate associated with our attack was incorrect, and gave a warning that the connection was not secure. However, when we bypassed this warning we were allowed to connect to example.com which showed us the malicious message.

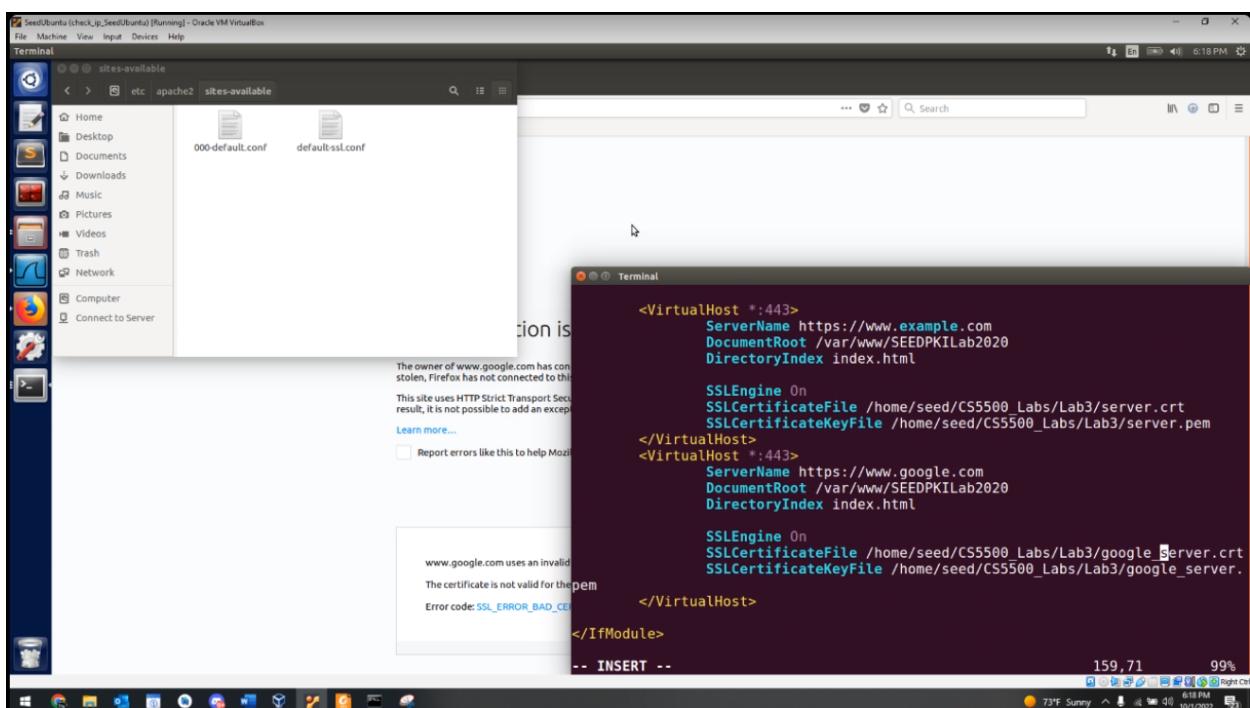


We were told that the certificate is invalid (it was signed for the incorrect site, not for example.com).

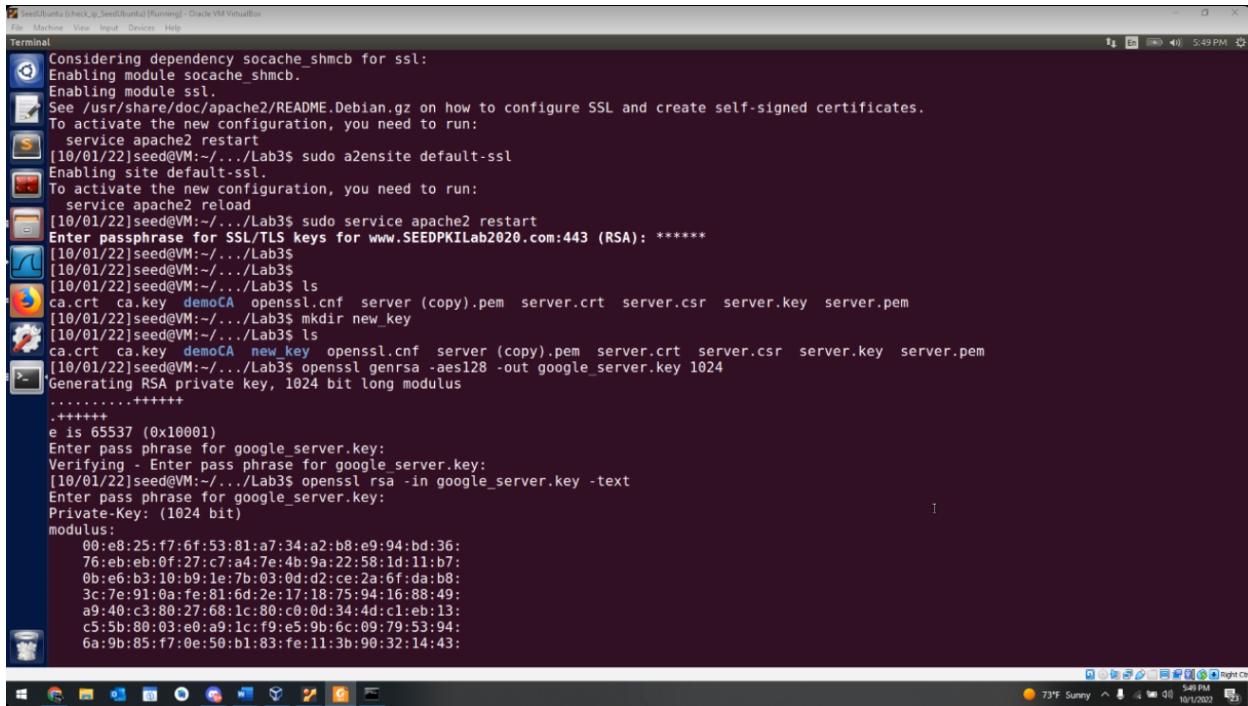


We were able to ignore the warning and view the site's contents, which have been overwritten on our own.

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



First, we added an additional host to the default-ssl configuration file to tell apache to redirect a local file.

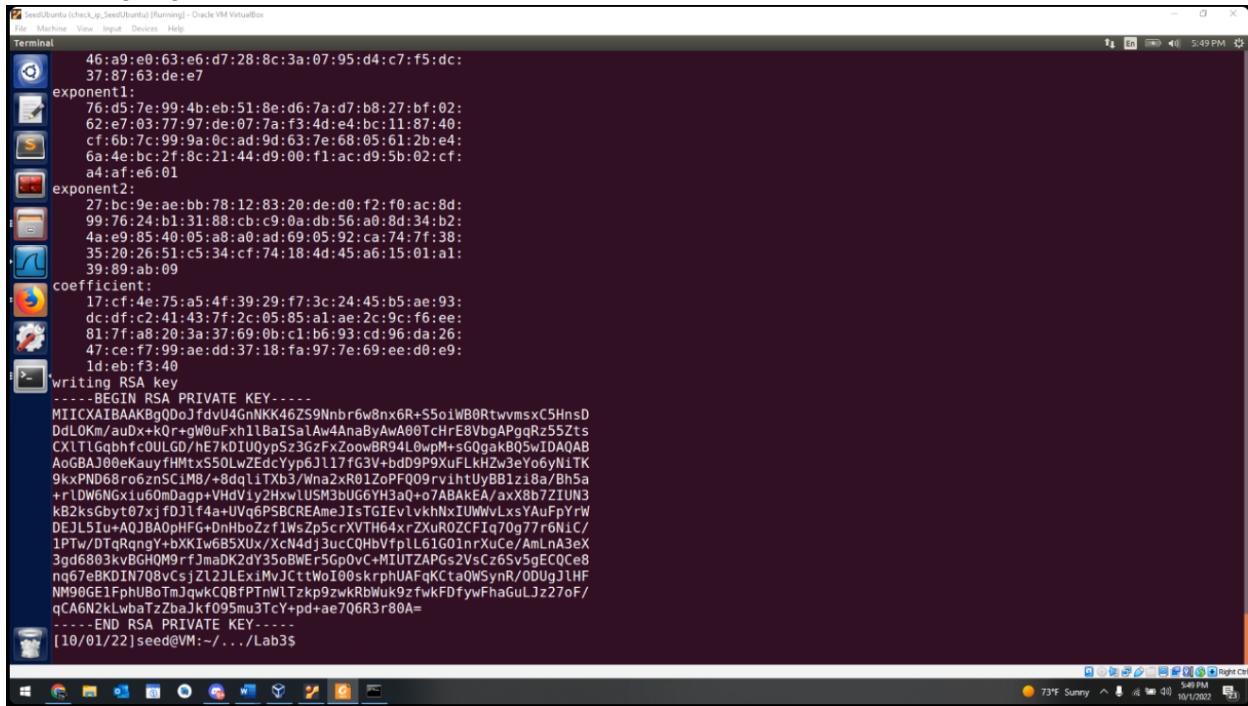


```

Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
    service apache2 restart
[10/01/22]seed@VM:~/.../Lab3$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
    service apache2 reload
[10/01/22]seed@VM:~/.../Lab3$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for www.SEEDPKILab2020.com:443 (RSA): *****
[10/01/22]seed@VM:~/.../Lab3$ 
[10/01/22]seed@VM:~/.../Lab3$ ca.crt ca.key demoCA openssl.cnf server (copy).pem server.crt server.csr server.key server.pem
[10/01/22]seed@VM:~/.../Lab3$ mkdir new_key
[10/01/22]seed@VM:~/.../Lab3$ ls
ca.crt ca.key demoCA openssl.cnf server (copy).pem server.crt server.csr server.key server.pem
[10/01/22]seed@VM:~/.../Lab3$ openssl genrsa -aes128 -out google_server.key 1024
[10/01/22]seed@VM:~/.../Lab3$ Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
Enter pass phrase for google_server.key:
Verifying - Enter pass phrase for google_server.key:
[10/01/22]seed@VM:~/.../Lab3$ openssl rsa -in google_server.key -text
Enter pass phrase for google_server.key:
Private-Key: (1024 bit)
modulus:
00:e8:25:f7:6f:53:81:a7:34:a2:b8:e9:94:bd:36:
76:eb:eb:0f:27:c7:a4:7e:4b:9a:22:58:id:11:b7:
0b:e6:b3:10:b9:1e:7b:03:0d:d2:ce:a6:f:d:a:b8:
3c:7e:91:0a:fe:81:6d:2e:17:18:75:94:16:88:49:
a9:40:c3:80:27:68:1c:80:0:0d:34:4d:c1:e:13:
c5:5b:80:03:e0:a8:1c:f9:e5:9b:6c:89:79:53:94:
6a:9b:85:f7:0e:50:b1:83:fe:11:3b:90:32:14:43:

```

Next, we generated a new key for google so we could have a certificate that tied directly to the spoofed google site.



```

46:a9:e0:63:e6:d7:28:8c:3a:07:95:d4:c7:f5:dc:
37:87:63:de:7
exponent:
76:d5:7e:99:4b:eb:51:8e:d6:7a:d7:b8:27:bf:02:
62:e7:03:77:97:de:07:7a:f3:4d:e4:bc:11:87:40:
cf:6b:7c:99:9a:0c:ad:9d:63:7e:68:05:61:2b:e4:
6a:4e:bc:2f:8c:21:44:d9:00:f1:ac:d9:5b:02:cf:
a4:af:e6:01
exponent2:
27:bc:9e:ae:bb:78:12:83:20:de:d0:f2:f0:ac:8d:
99:76:24:bl:31:88:cb:c9:0a:db:56:a0:8d:34:b2:
4a:e9:85:40:05:a8:a0:ad:69:05:92:ca:74:7f:38:
35:20:26:51:c5:34:cf:74:18:4d:45:a6:15:01:a1:
39:89:ab:09
coefficient:
17:cf:4e:75:a5:4f:39:29:f7:3c:24:45:b5:ae:93:
dc:df:c2:41:43:7f:2c:05:85:a1:ae:2c:9c:f6:ee:
81:7f:a8:20:3a:37:69:0b:c1:b6:93:cd:96:da:26:
47:ce:f7:99:ae:dd:37:18:fa:97:7e:69:ee:d0:e9:
1d:eb:f3:40
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICXAIQAKBgQD0JfdvU4GnKK46Z59Nnbr6w8nx6R+S5o1WB0RtwvmsxC5HnsD
OdlOKm/auDx+kOr+gW0uFxh1BaIsAla4naByAwA00TChE8VbgApQqrRz55Zts
CXlTLGobhfc0ULGD/hE7kDIU0ypS23GzFxZoopBR94Lwpm+sG0gakB05wIDAQAB
AoGBAAJ00eKauyfHmx550lwZEdcYyp63l17fG3V+bdD9P9XuFLKhZw3eY0gyNiTK
9kxPND68ro6znScIM8+8dq1iTxb3/Wha2xR01z0PfQ09rvihUyBB1zI8a/Bh5a
+rLDW6N6X1u60Ddagp+VhdViy2XwlUSM3bUG6YH3aQ+o/ABAKEA/axXb6/Z10N3
k82ksGbyt07xjfDlf4a+Uvg6PSBCREAmjeJlsTGIEvlvhNxUWvLxsYAuFpYrW
DEJL5Iu+A0JBAOpHFG+DnhboZf1WsZp5crXVTH64xrZXR0ZCF1q70g77r6NfC/
1PTw+DtqRqngY+bXXIw6B5XUx/XcN4dj3ucCQHbVfpll61G01nrXuCe/AmlnA3eX
3gd6803kvBGHQMrJma0K2dY35oBWEr5Gp0vC-MIUtzAPGs2VscZ65v5gECQce8
na67eBKDN708vCsjZL2JLExiMvJCTtWoI00skrphUAFqKCtaQWSynR/ODUgJlHF
NM90GE1FphUBoTmJqwkCQBfPTwltZkp9zwKRbwk9zfwKDfTywFhaGuLj270F/
qCAGN2kLwba1zzbaJkt095mu3tCY-pd+ae706R3r80A=
-----END RSA PRIVATE KEY-----
[10/01/22]seed@VM:~/.../Lab3$

```

```

yfPTFFkH40bpqkuK9DA8+sWaESIQwqFJdqj3iMYvcqMrM1VTjQUPukSt75ZMLgfo
HhP0ijyPaIIlMT0YbkSqjG3rWeEJryi20ieGYlrfxnpVgcdLm1dRjNmHywIDAQAB
AoGAcjhWjQNvasAjbfxE9AF36lWSXTfvAqatgehrN0V94Dt5jvv28/LarQAQw
BxoTOL7V+uxM3V/HhqL52ixDt0faPlgxluMudjbrzjACalcegXyxFKF01jJr1Mq
+8CbNifZda/HoRInai0VrxAcA6mwW/hxPJsnA7xalC0kg/5ECQODPt/RWtJ0600d
FycweCF8ZEHzo2X3xzjKQB0h7G/zN0FJW01cF4vNRTu5zxBw+hYdqTn2pH05
Whe2mnXdkAkeAx0Hijx0isXj0Ah7X7din0Wt1/hy2gm7+7FjbNzkofo/PvgiHa1
l2Y-3hAsrV75l1326ny+VPSoUhM0NPuPwJAYPREkjxXutaIh0SNWERabTLhvdu
/TySN2udRDR8NEB0CckZBYL+HOBH7nwQMr0h5CHzirLF7i+8s01+05mSQjAkGao
c1Qb2p3thRLIGysW91La5gEhlHKq0euElf1ptos/1/xn1JRiPaYZ9FCiptilzEv
MN80DjikAChLMHrQOJAK0+FRWaA9B65SNmDYdheBGfrwcIBFbznmnYRbE8kQ
YYqJT0+55G5vv8RFm5DzSnL8HcsjEs9cwbc049Yu6g==
-----END RSA PRIVATE KEY-----
[10/01/22]seed@VM:~/.../Lab3$ 
[10/01/22]seed@VM:~/.../Lab3$ openssl req -new -key google_server.key -out google_server.csr -config openssl.cnf
Enter pass phrase for google_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.
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Pomona
Organization Name (eg, company) [Internet Widgit Pty Ltd]:Cal Poly Pomona
Organizational Unit Name (eg, section) []:www.google.com
Common Name (e.g. server FQDN or YOUR name) []:www.google.com
Email Address []:www.google.com@aol.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:abc123
An optional company name []:www.google.com
[10/01/22]seed@VM:~/.../Lab3$ 
[10/01/22]seed@VM:~/.../Lab3$ openssl req -new -key google_server.key -out google_server.csr -config openssl.cnf

```

Afterwards, we created the new certificate signing request.

```

[10/01/22]seed@VM:~/.../Lab3$ cp google_server.key google_server.pem
[10/01/22]seed@VM:~/.../Lab3$ cat google_server.crt >> google_server.pem
[10/01/22]seed@VM:~/.../Lab3$ 

lines 1-20/20 (END)
[10/01/22]seed@VM:/etc$ 
[10/01/22]seed@VM:/etc$ 
[10/01/22]seed@VM:/etc$ 
[10/01/22]seed@VM:/etc$ 
[10/01/22]seed@VM:/etc$ 

```

We then placed the key and certificate into a pem file.

```

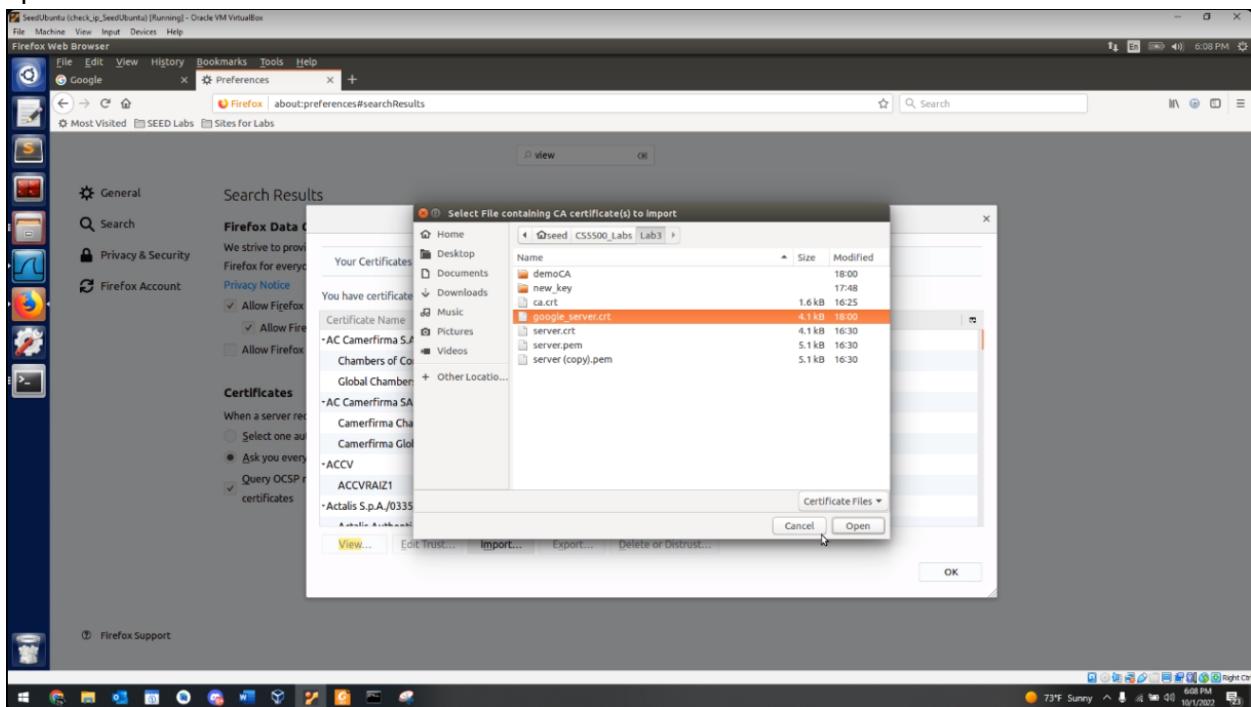
SeedUbuntu (check_ip_SeedUbuntu) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
A challenge password []:abc123
An optional company name []:www.google.com
[10/01/22]seed@VM:~/.../Lab3$ openssl ca -in google_server.csr -out google_server.crt -cert ca.crt -keyfile ca.key -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: 4099 (0x1003)
    Validity
        Not Before: Oct 2 01:00:12 2022 GMT
        Not After : Oct 2 01:00:12 2023 GMT
    Subject:
        countryName          = US
        stateOrProvinceName  = California
        organizationName     = www.SEEDPKILab2020.com
        organizationalUnitName = www.google.com
        commonName            = www.google.com
        emailAddress         = www.google.com@aol.com
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
        86:32:A7:0F:08:89:B6:5F:C1:A4:07:4A:B2:CE:41:84:F0:F0:3D:AA
    X509v3 Authority Key Identifier:
        keyid:24:50:E0:E5:0C:5C:CB:5A:DD:77:4C:C3:CF:6E:D1:6C:91:5F:27:A4

Certificate is to be certified until Oct 2 01:00:12 2023 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
[10/01/22]seed@VM:~/.../Lab3$ 

```

We then signed the certificate, creating a new entry in the ca certificate that had google's spoofed information.



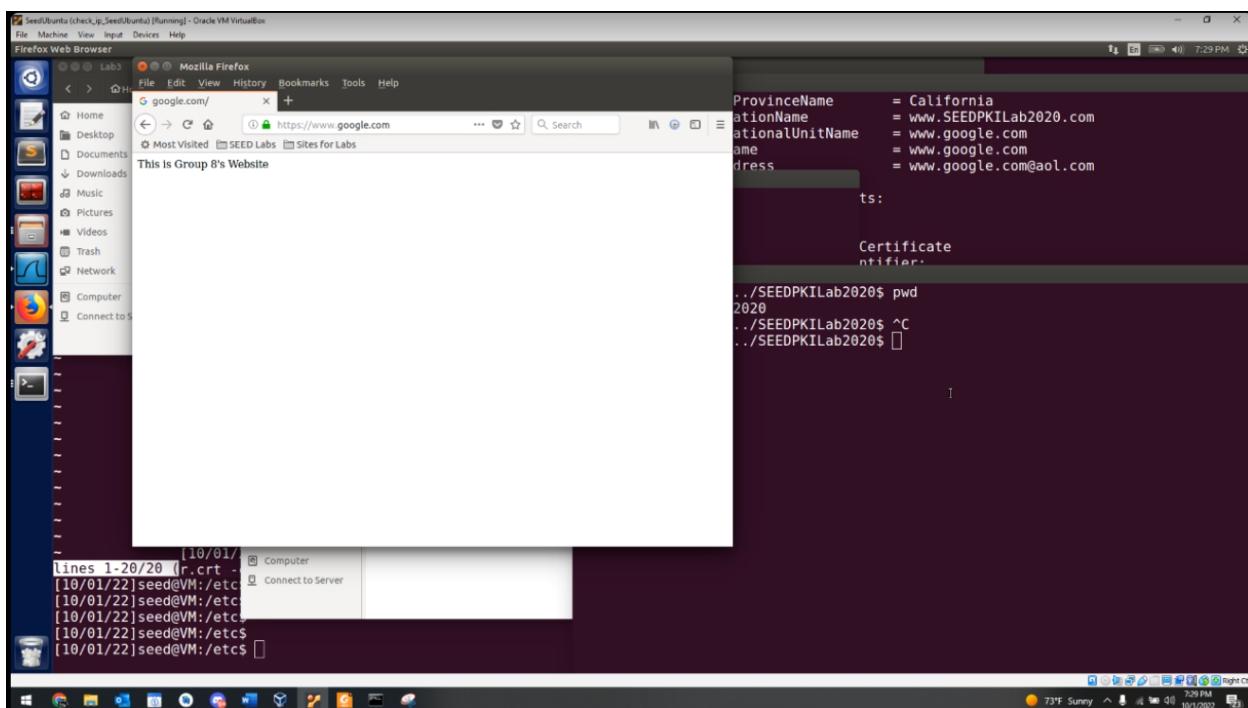
We updated the certificate within firefox to ensure it was accurate.

```
[10/01/22]seed@VM:~/.Lab3$ openssl ca -in google_server.csr -out google_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: 4099 (0x1003)
    Validity
        Not Before: Oct 2 01:00:12 2022 GMT
        Not After : Oct 2 01:00:12 2023 GMT
    Subject:
        countryName          = US
        stateOrProvinceName = California
        organizationName    = www.SEEDPKILab2020.com
        organizationalUnitName = www.google.com
        commonName            = www.google.com
        emailAddress         = www.google.com@aol.com
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
X509v3 Subject Key Identifier:
    86:32:A7:0F:D8:89:B6:5F:C1:A4:07:4A:B2:CE:41:84:F0:F0:3D:AA
X509v3 Authority Key Identifier:
    keyid:24:50:E0:E5:0C:5C:CB:5A:DD:77:4C:C3:CF:6E:D1:6C:91:5F:27:A4

Certificate is to be certified until Oct 2 01:00:12 2023 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
[10/01/22]seed@VM:~/.Lab3$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for www.google.com:443 (RSA): *****
[10/01/22]seed@VM:~/.Lab3$
```

We then restarted apache to ensure the changes would be reflected.



Finally, when we went to google.com we were told we had a secure connection even though the site was compromised.