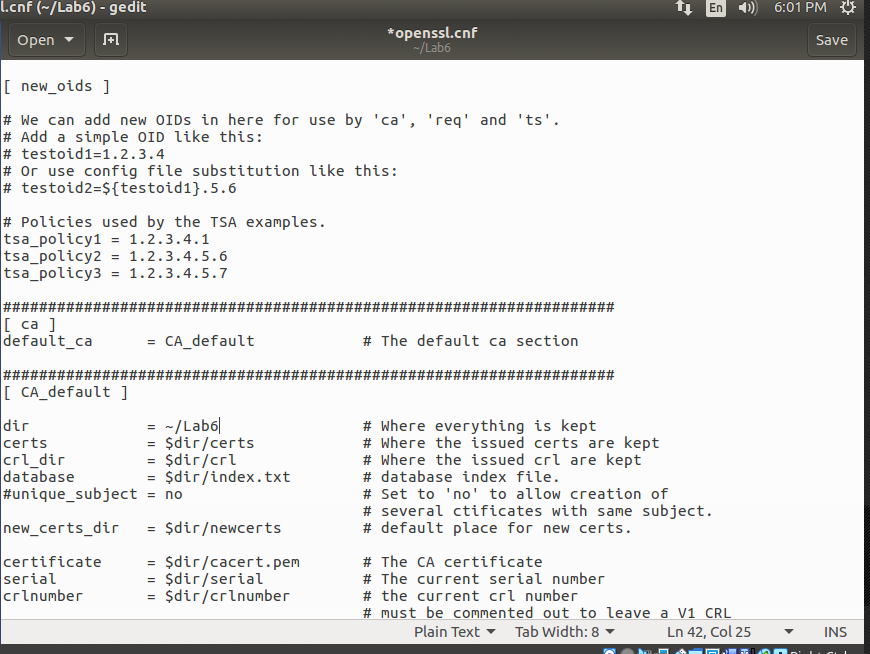
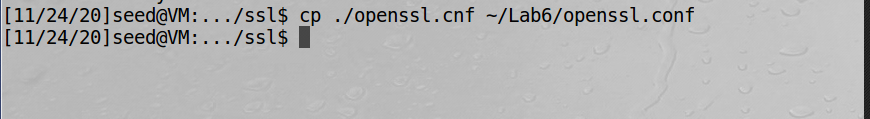
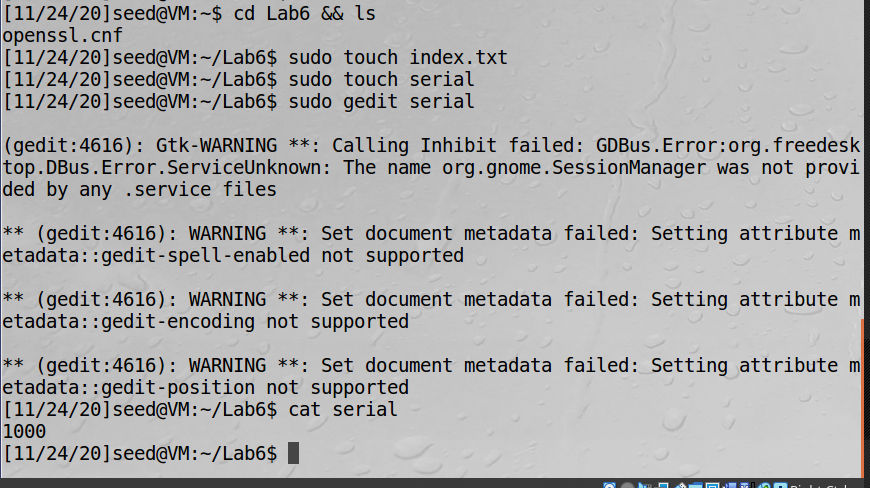
Lab 6 – Kevin Martin

**Task 1 – Becoming a Certificate Authority (CA)**

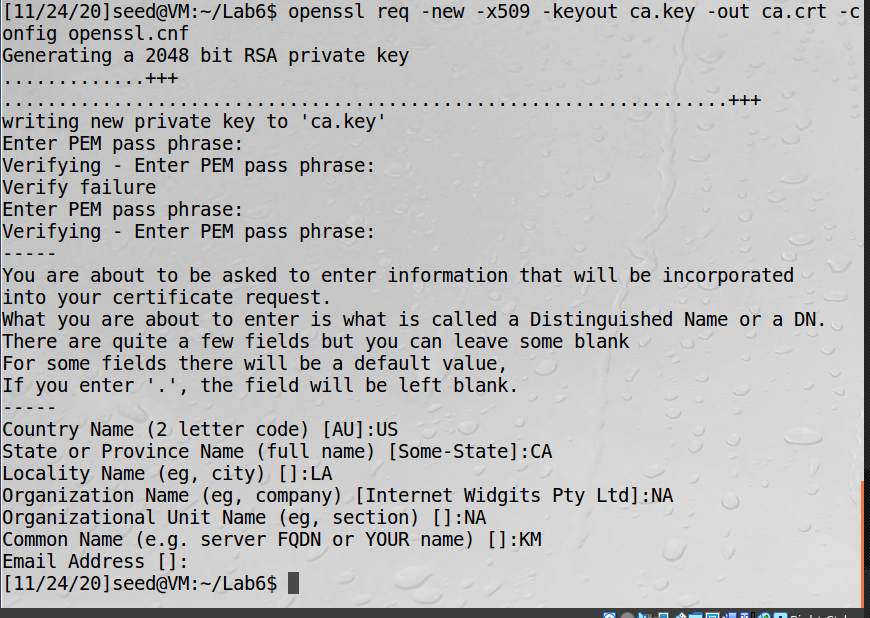
To become a CA, we must first copy a the conf template file into the machine we will be using, VM A in my case. Then, edit the file to show current directory:



Next the required files, an empty **index.txt** and a **serial** file with the number 1000:



Finally, the self-signed certificate:

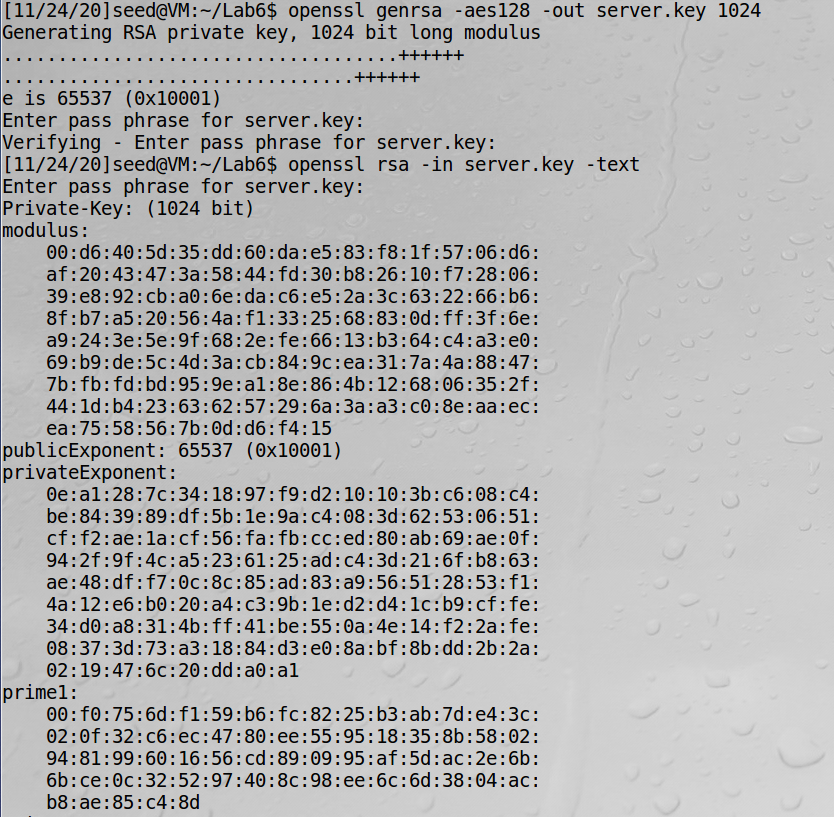


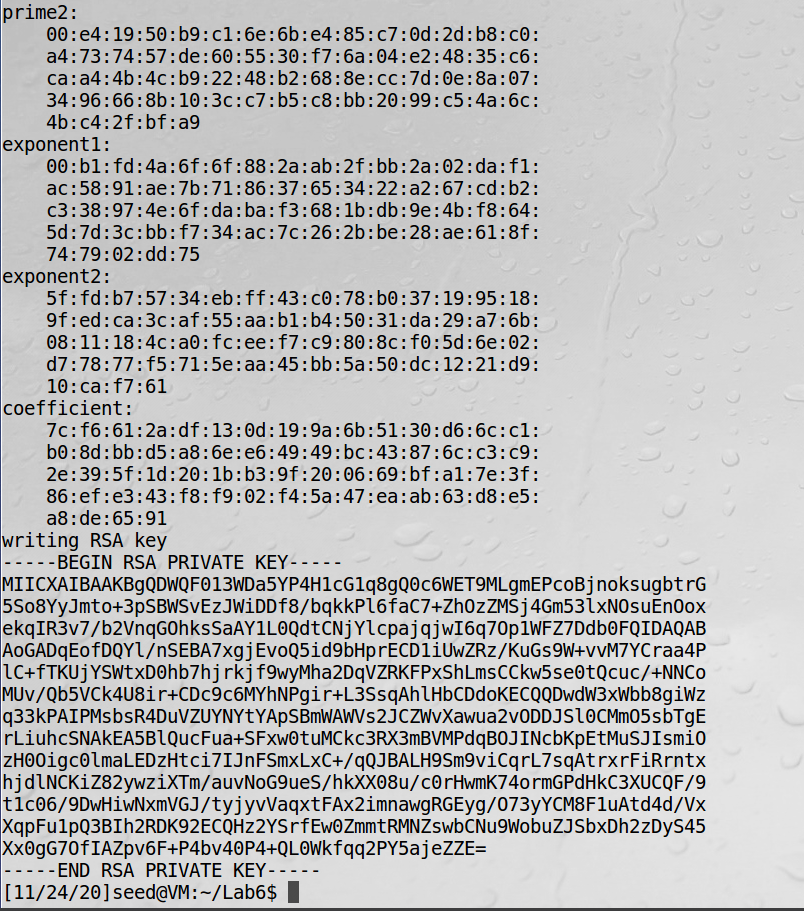
**Observation:** We used openssl to become a CA by utilizing the stock template and then modifying for our cusom configuration. The openssl command provides all the components necessary to create a new 509 certificate so long as the user enters in the all the proper identifying data.

**Explanation:** To utilize the PKI encryption system, a working certificate is needed. Instead of utlizing a third party, we can generate one ourselves and continue on with the encryption process.

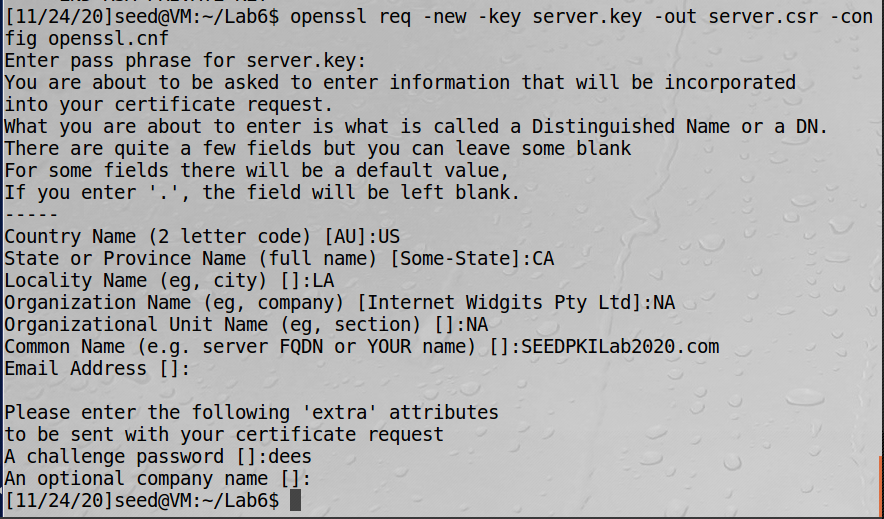
**Task 2 – Creating a Certificate for SEEDPKILab2020.com**

Next we generate the RSA public and private keys. We will use the same password as the previously generated certificate:

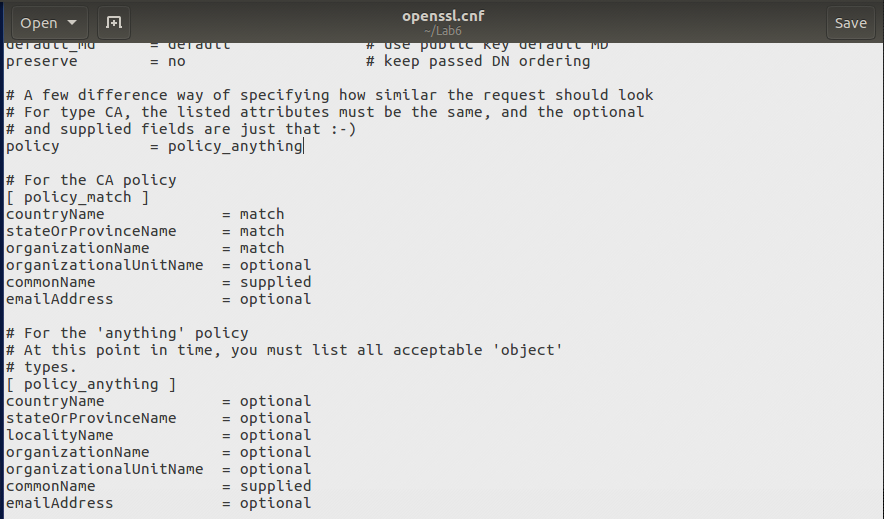




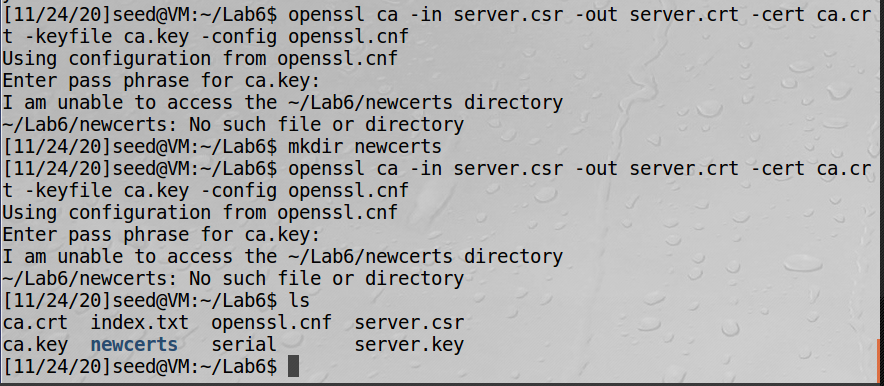
The second step is to generate the certificate signing request (CSR) with the common name specified:



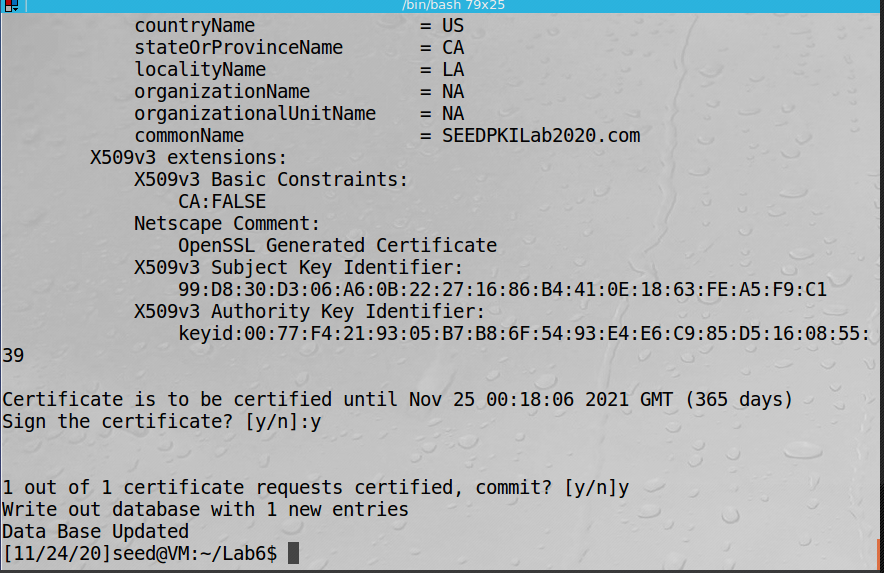
In the last step, we can actually generate the certificate. However, I had an error with the standard settings so I modified the **openssl.cnf** document to allow for “policy = policy\_anything”:



Unfortunately, I am still getting an error where the openssl program cannot access a “newcerts” directory. I created one manually and still see the error:



I finally realized that my main directory was incorrect, so after changing that I could create the certificate:



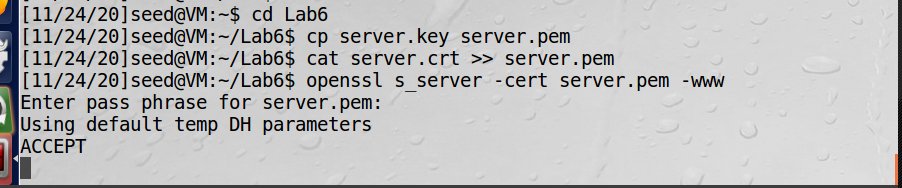
**Observation:** Similar to the first task, we were using openssl to create a certificate. We established ourselves as an authority first, and then recreated the events for a new “client”. This is now a valid certificate for the PKI process.  
  
**Explanation:** In order to create a valid certificate, there are three requirements from the previously established CA: first, generate the public and private keys, then create a certificate signing request with the **public key**, and finally issuing the certificate.

**Task 3 – Deploying Certificate in an HTTPS Web Server**

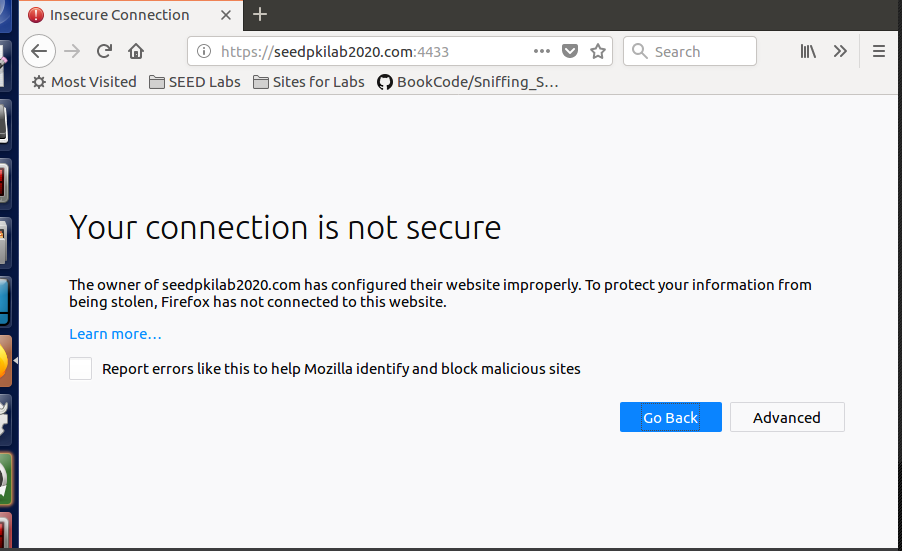
To actually use the certificate to securely browse the web, we must first add the domain to the recognized host list, which is in /etc/hosts (it shows as read only in the screenshot but I fixed it after):



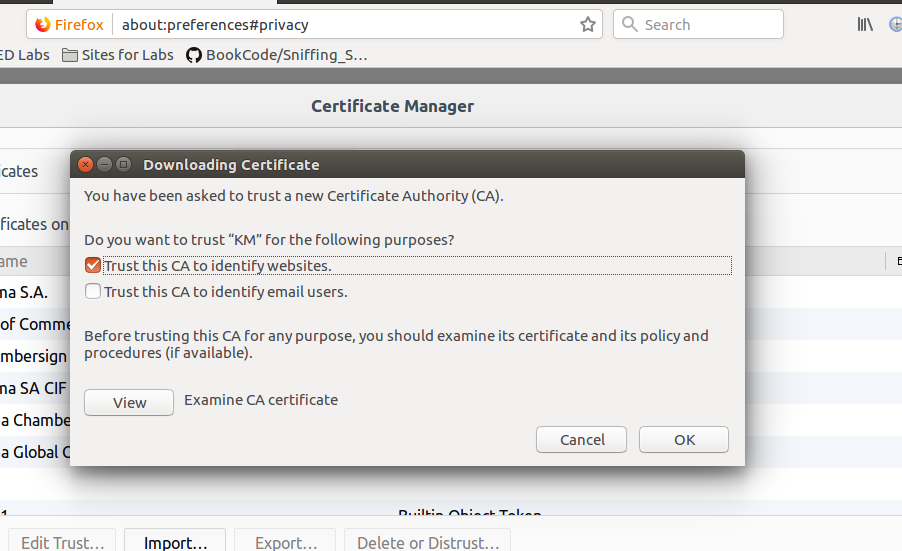
Next we will combine the secret key and certificate into one file and launch the web server:

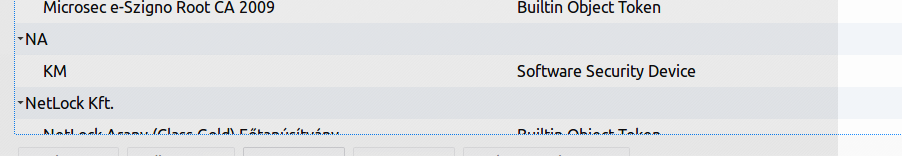


Before enabling our certificate, we get an error from the browser:

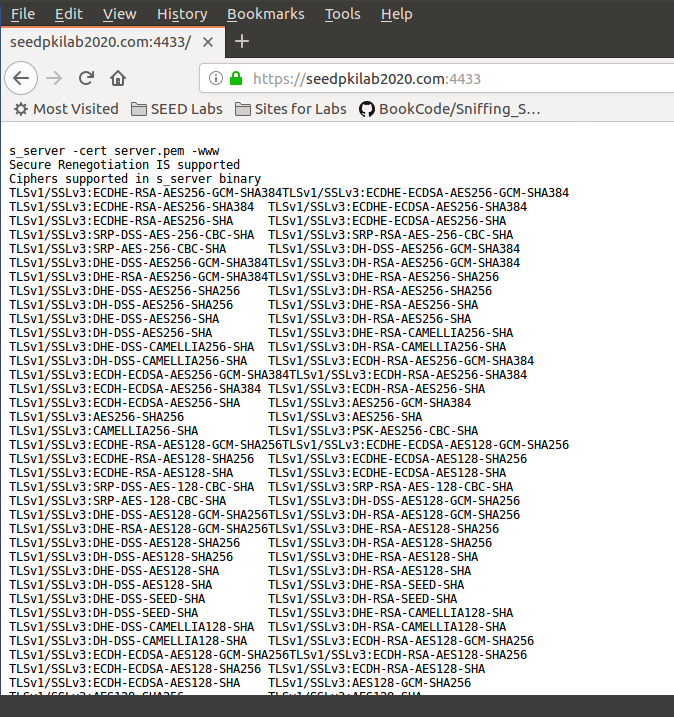


To fix, we enable **ca.crt** in Firefox:

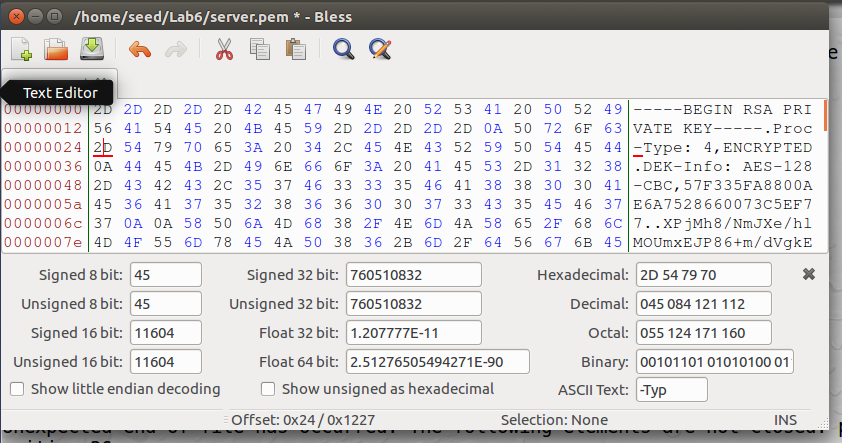




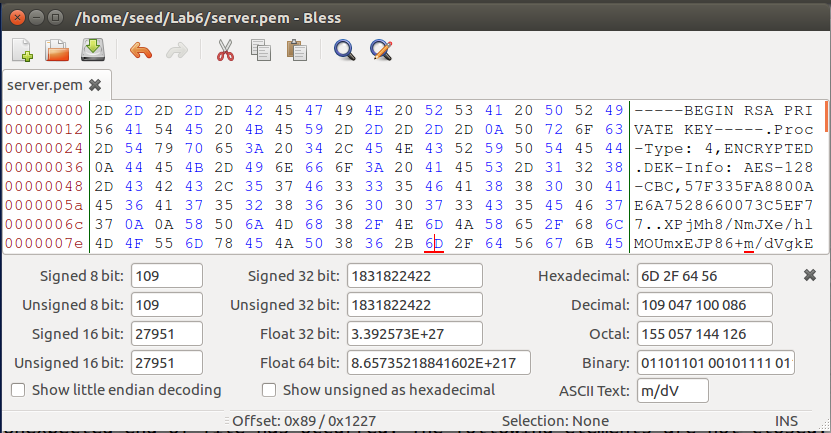
Finally, we can see the website:



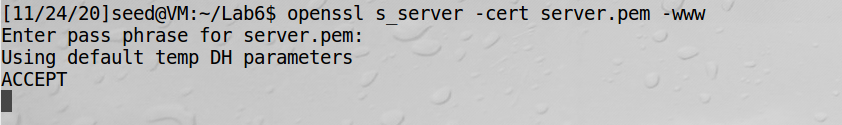
To test the HTTPS website, we will modify a single byte of **server.pem**, restart the server, and reload the URL. First, the modified bit, going from “2D” to “3D” using **bless**:



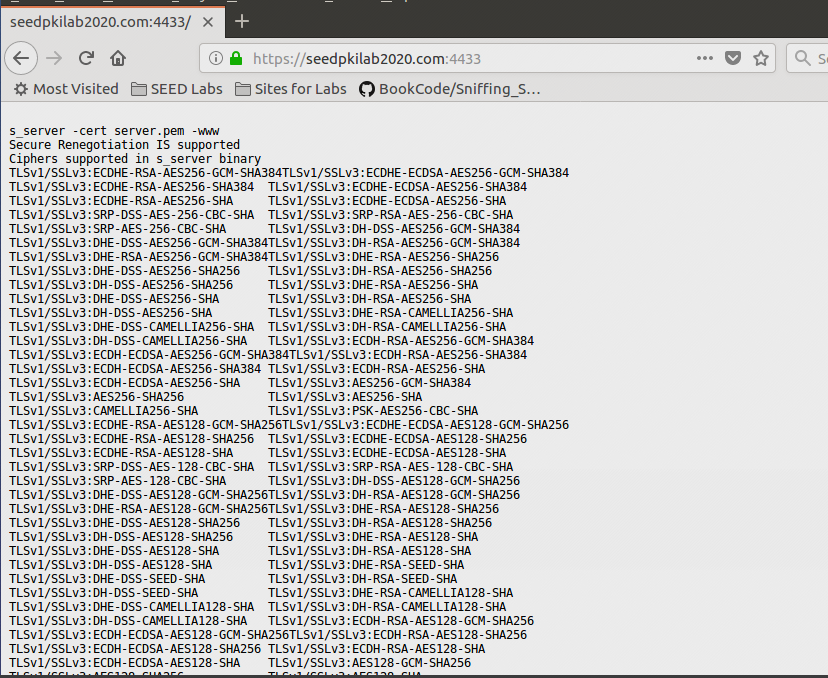
This bit caused an error, so I modifed the 6D (originally a 7D):



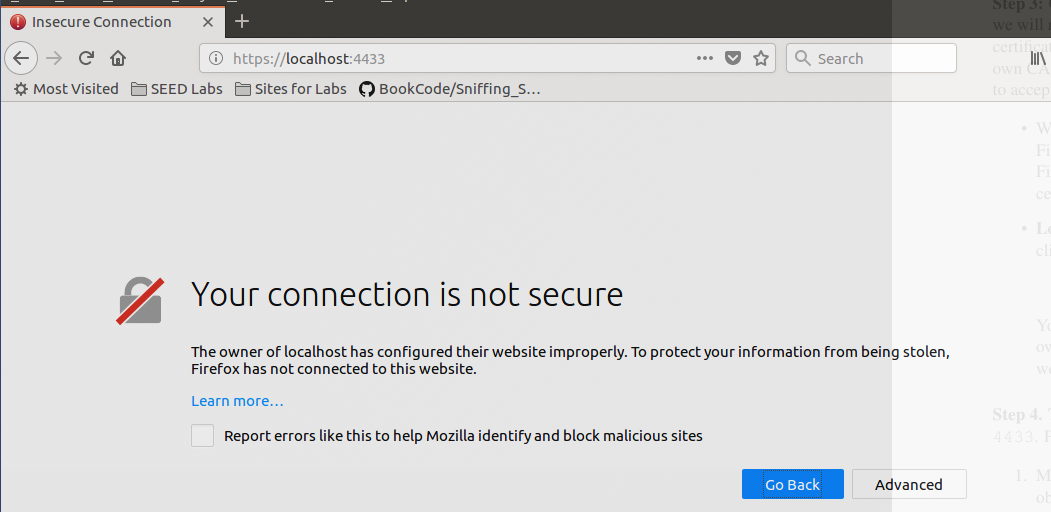
Next, restart the server:



Finally, retry the URL:



I expected to see an error, but perhaps the certificate/website was previously cached. I do however see the intended result upon trying to locate [https://localhost:4433](https://localhost:4433/):



**Observation:** We were able to successfully deploy our certificate for the identified client. However, upon corrupting even a single bit, we invalidate the entire process. Additionally, to use the certificate, we needed to add our previously created CA to the list of trusted CA’s in Firefox. This allowed the request to be permitted.

**Explanation:** By first establishing a valid CA, we can issue certificates that will be recognized via web browsers. This is all done using the openssl program, and correct entries/passwords must be used in order for the certificate to work.

**Deploying Certificate in an Apache-Based HTTPS Website**

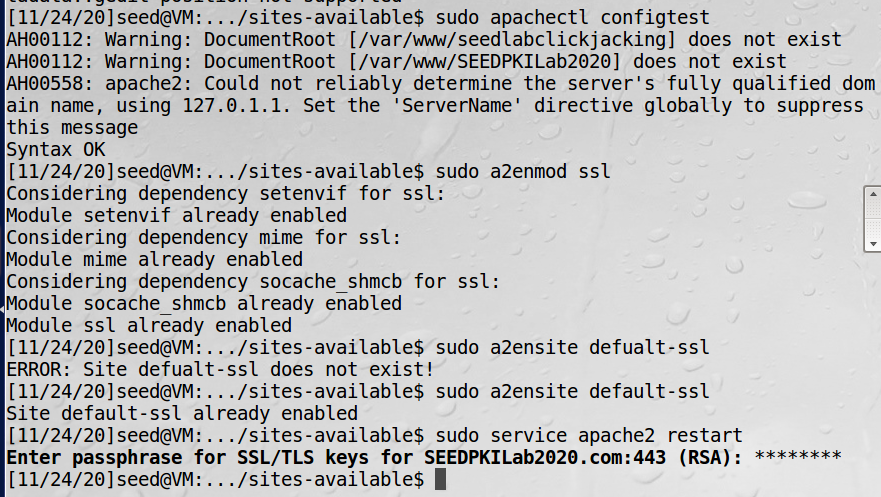
To test our certificate on a real webserver, we configure the pre-installed Apache server for our new domain. The first step is to modify the **open-ssl.conf** file:



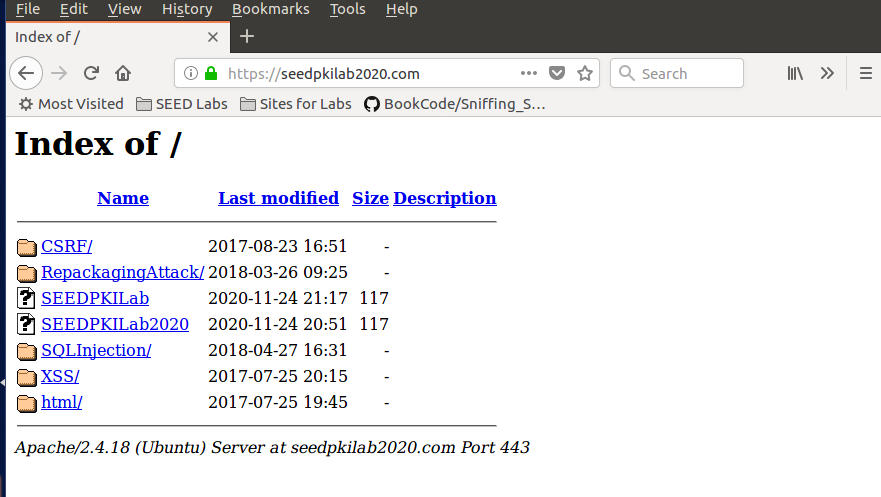
Then we have to create the website specified in the virutal host:



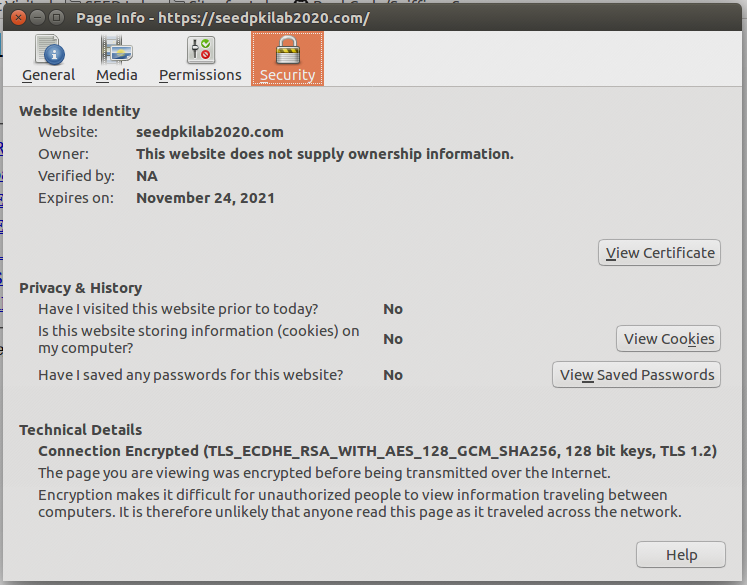
Next, we enable SSL:



Finally, test the webiste. Initially, I was getting error saying it could not find the document root. I changed it to be just **/var/www/** which allowed me to access the following:



And we can confirmthat this site is secure per Firefox:



**Observation:** After setting up an actual website via Apache, we can have Firefox independently confirm that the site is secure. Note that my directory at **/var/www/** is shown, and *clearly* the missing SEEDPKILab file is there. Regardless, we have achieved a secure connection.  
  
**Explanation:** By combining Apache with the previous openssl steps, we can create our own secure websites.