Project 2

**Task 1:**

Running the command: ‘openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf’

This command created the ca.crt, ca.key files. The command uses the openssl.cnf copied from /usr/lib/ssl/openssl.cnf.

**Task 2:**

Running the command: ‘openssl genrsa -aes128 -out server.key 1024’

This command creates the server.key file in the root directory of our folder.

We then ran the command: ‘openssl rsa -in server.key -text’

Showing us the server.key file was created successfully.



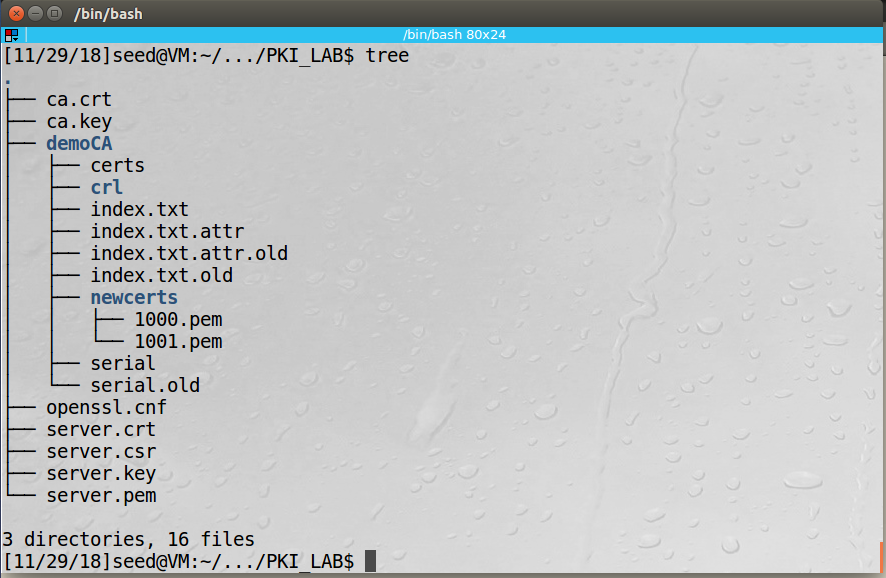
We then ran the command: ‘openssl req -new -key server.key -out server.csr -config openssl.cnf’

This generated the Generate a Certificate Signing Request (CSR). This contains the company’s public key.

Next, we needed to generate the X509 certificate file using the command: ‘openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key \ -config openssl.cnf’

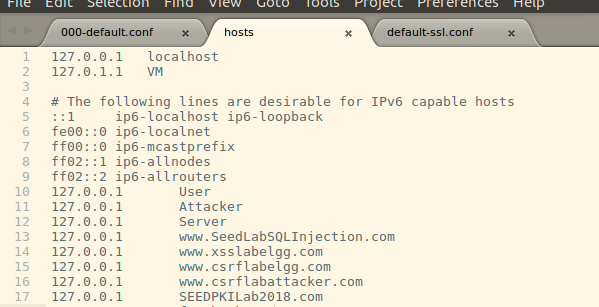
When making this file you need to make sure that all the fields from the server.key file command. However, if you don’t you can bypass the check for matching fields with the "policy = policy\_match" change to "policy = policy\_anything". This option is located in the openssl.conf in the current directory.

This is what our current working directory looks like now with all the files we have generated:



**Task 3:**

For the next part to work, we needed to add SEEDPKILab2018.com to the hosts file on linux. This is located at /etc/hosts.

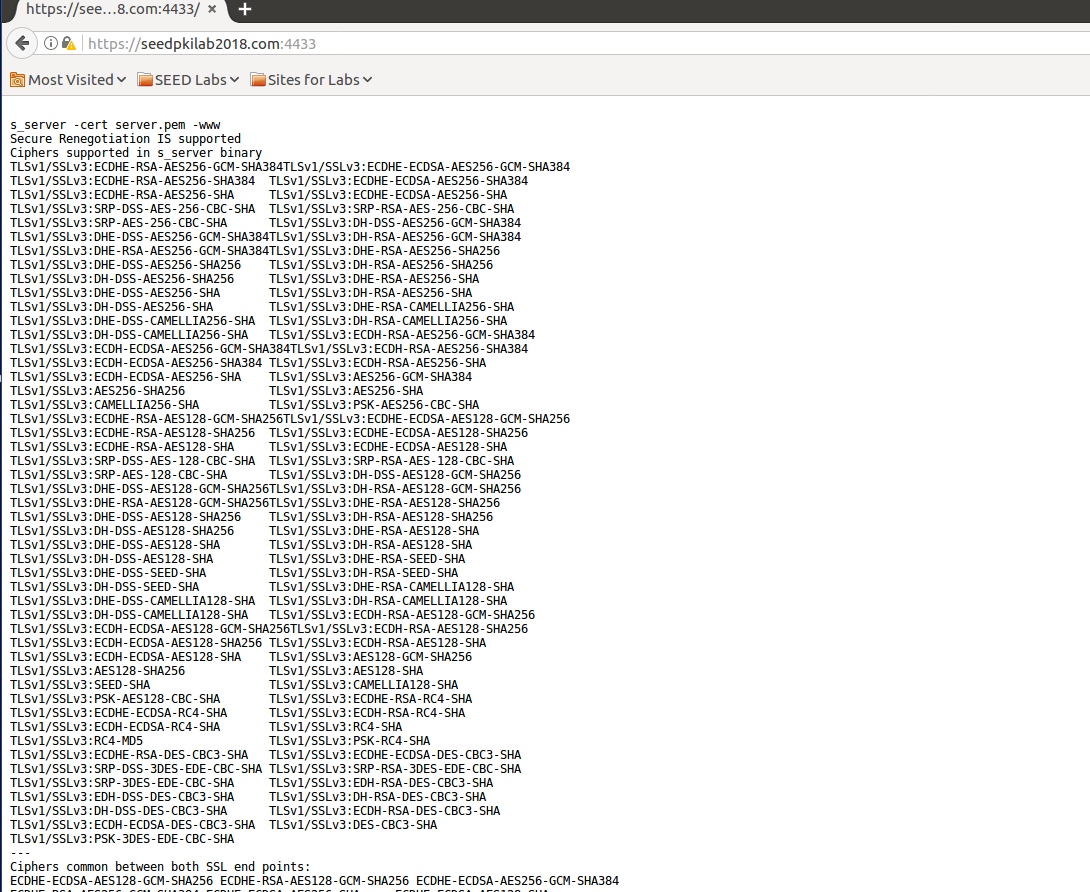
****

This makes it so if you enter SEEDPKILab2018.com into the browser, it will redirect you to 127.0.0.1 or localhost.

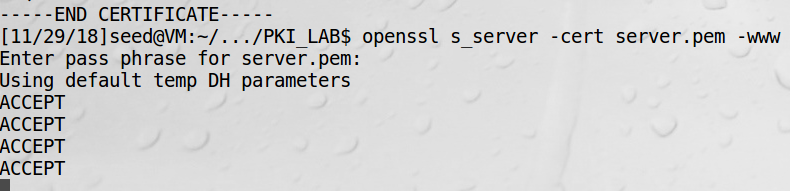
We make the .pem files for the server using the commands: ‘cp server.key server.pem’ and ‘cat server.crt >> server.pem’

This combines both the server.key and server.pem files into one file called server.pem.

Next we opened the openssl server using the command: ‘openssl s\_server -cert server.pem -www’

Navigating to <http://seedpkilab2018.com:4433> gave the following output

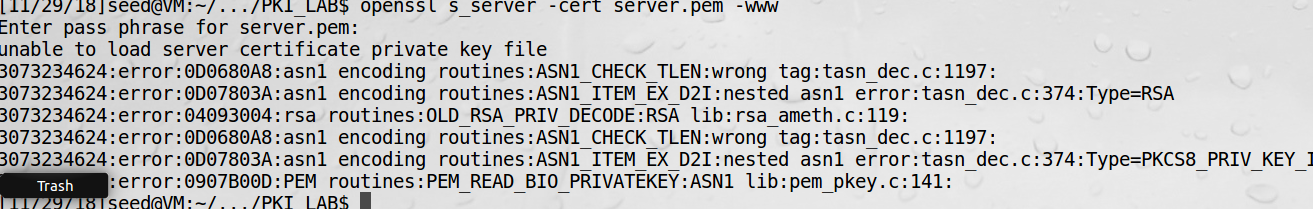
The command line shows that the browser is accepting out certificate files.



This will not work until we add the ca.crt file to Firefox, because otherwise the browser will not accept self-signed certificates.

Navigating to <https://localhost:4433> gives the same output.

IF we were to modify the server.pem file the server would not start and openssl would throw errors about the validity of the certificate.



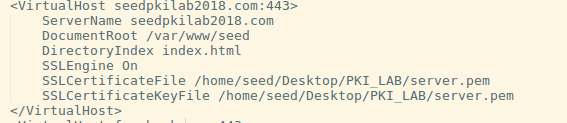
Changing the modification to the file makes the server accept the file again.

**Task 4:**

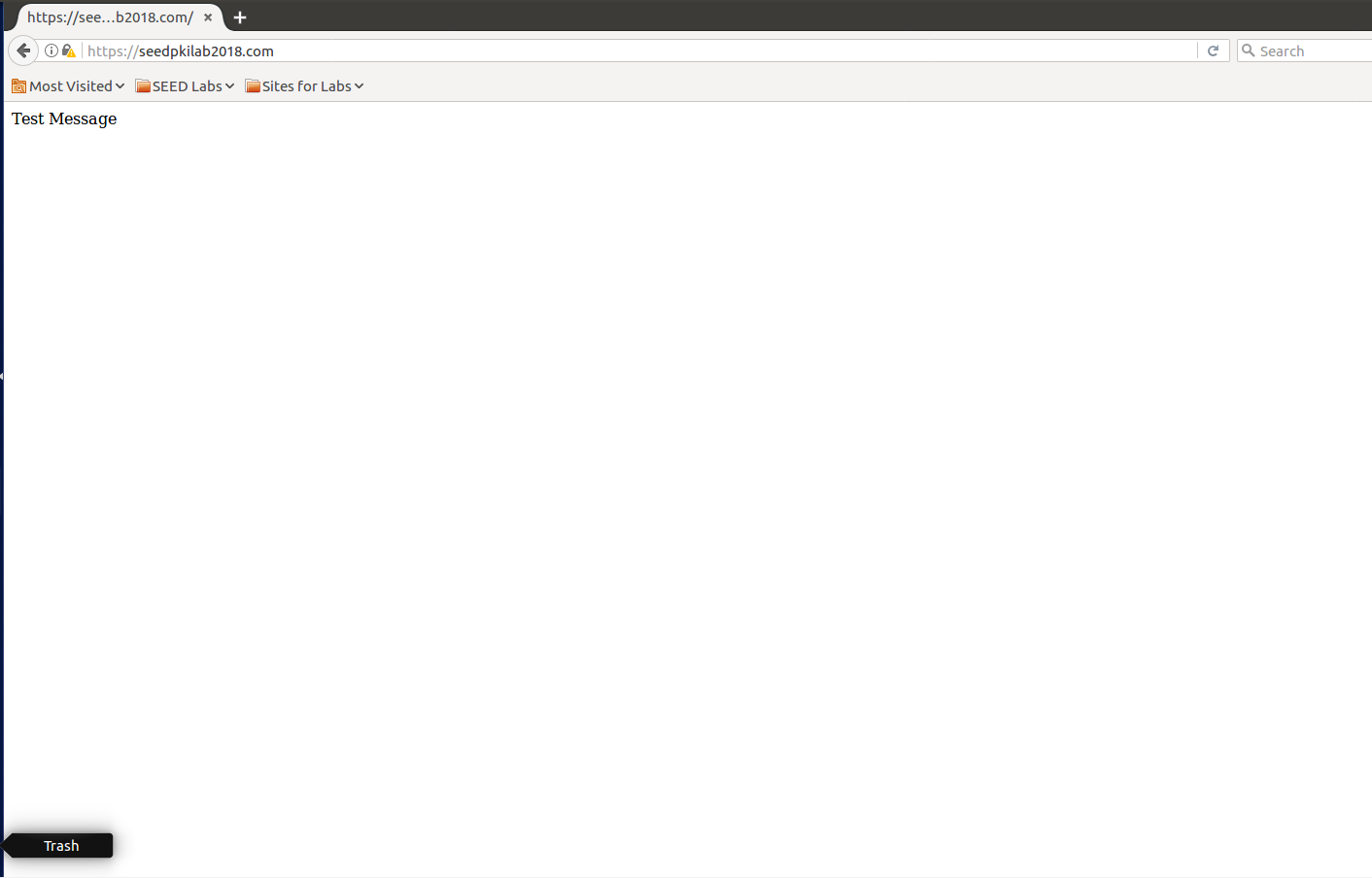
The file 000-default.conf file is used to modify the configuration of the non-secure sites for the apache2 server. Here is our config for seedpkilab2018.com:



The file default-ssl.conf file is used to modify the configuration of the secure sites (using port 443) for the apache2 server. Here is our config for seedpkilab2018.com:



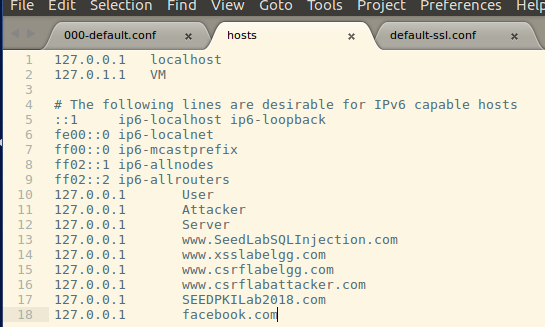
After checking the apache configuration with the command: ‘sudo apachectl configtest’ then enabling the sites and ssl we restarted the server using the command: ‘sudo service apache2 restart’.

Going to the browser and entering seedpkilab2018.com or <https://seedpkilab2018.com> returns this website now:

We don’t see the apache default page because we created a new directory for the seedplikab2018.com website under in /var/www/seed. Here we just added a new index.html file that had Test Message in a paragraph tag.

**Task 5:**

Now we needed to redirect users to our index.html file, even if they were trying to go to a different website. We decided to redirect users going to facebook.com. First we needed to add facebook.com to our hosts file.

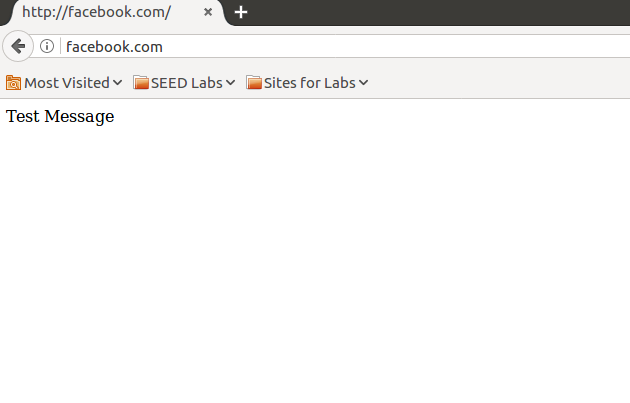


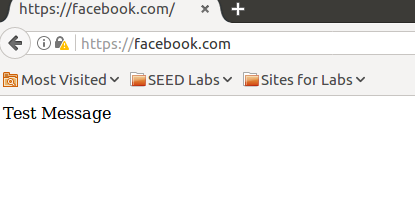
Then we added Facebook to both the port 80 config and the port 443 config in apache2.





We are redirecting the user to the directory /var/www/seed if they enter facebook.com in the browser now.





**Task 6:**

We had tried to create new certificates Facebook.com so the browser would raise no suspicion, but it looks as if Firefox is smart enough to know that these certificates are self-signed and not to be trusted for websites already in its database. So, it would not be possible to launch a man in the middle attack using a fake website.