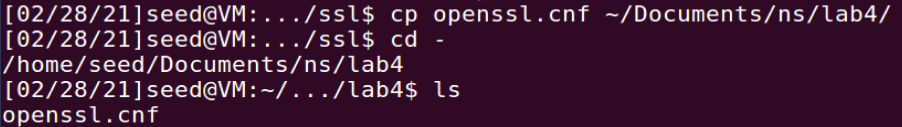
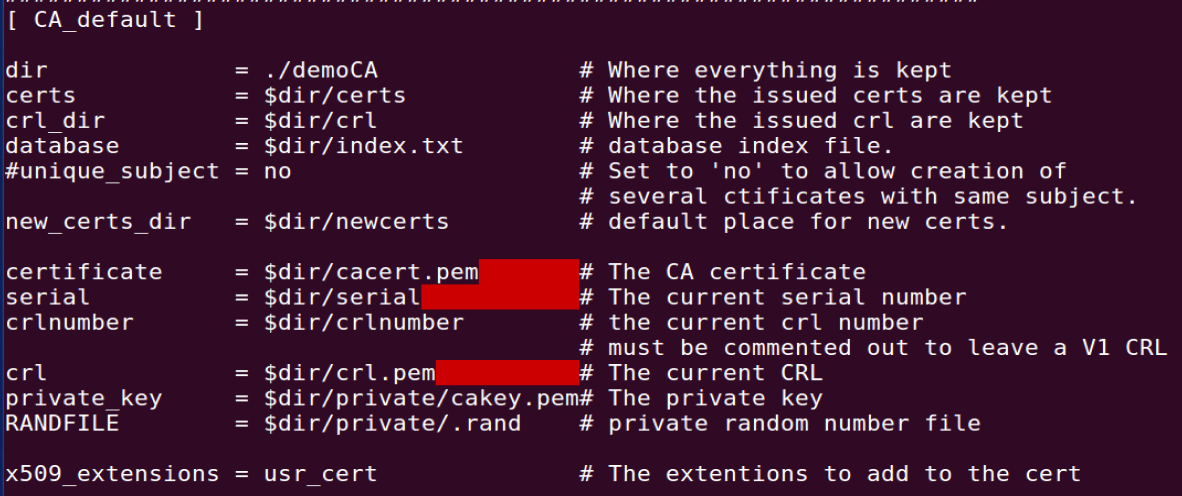
# 50.020 Network Security Lab 4: Public-Key Infrastructure (PKI)

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

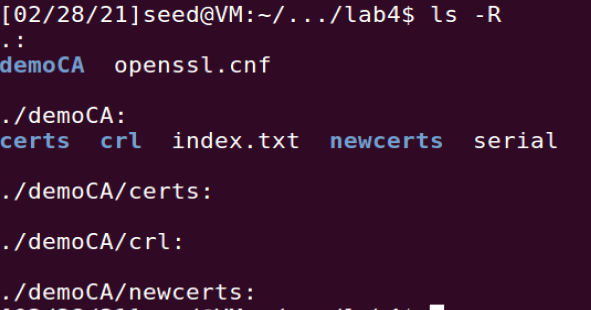
The OpenSSL configuration file is copied from /usr/lib/ssl/ into the current directory ~/Documents/lab4/



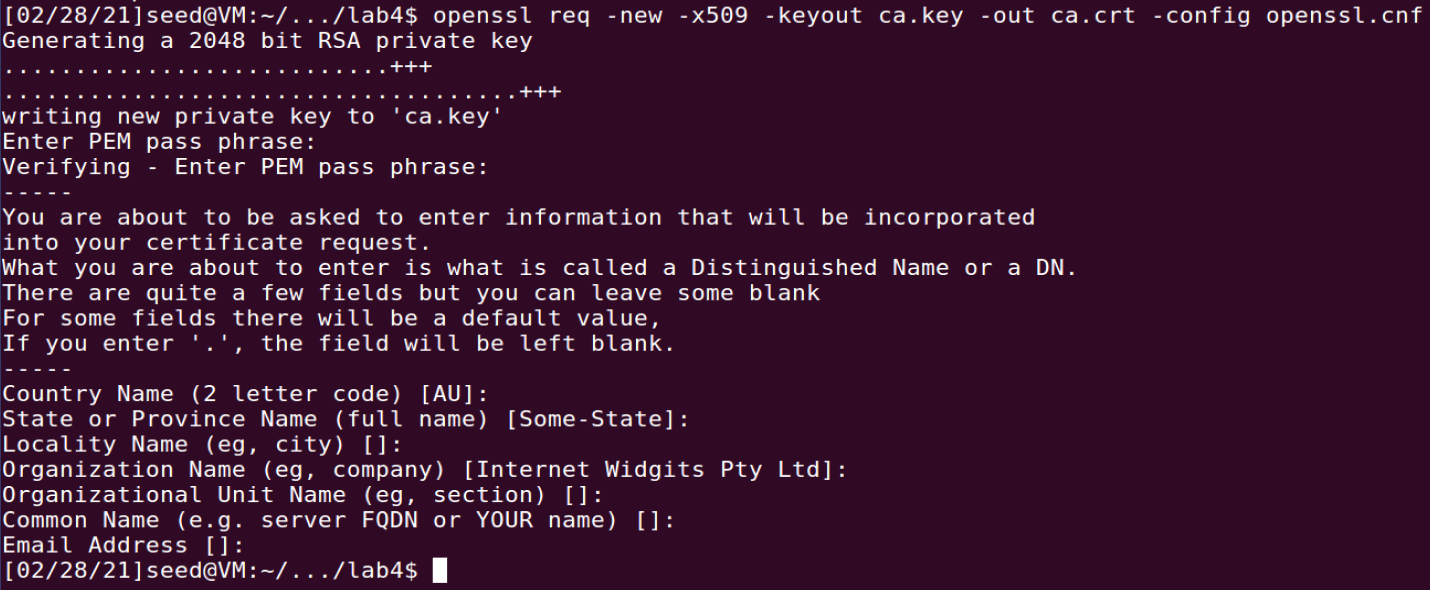
From the configuration file, these are the subdirectories that need to be created:



The following subdirectories are created accordingly:



The following command is executed to generate the self-signed certificate. The passphrase used is *liying*. The rest of the fields are left as default:



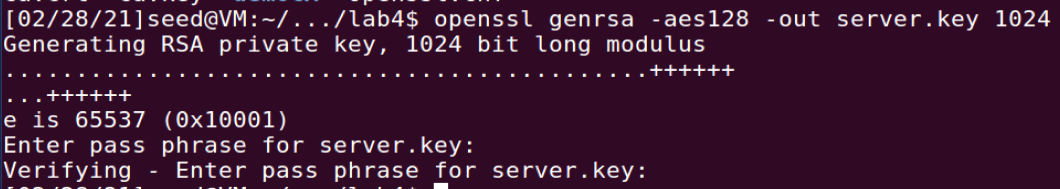
The private key and certificate are generated in the current directory as ca.key and ca.crt respectively:



## Task 2: Creating a Certificate for SEEDPKILab2020.com

Step 1: Generate public/private key pair

The following command is run to create the company’s public/private key pair. The password *liyingcompany* is used:

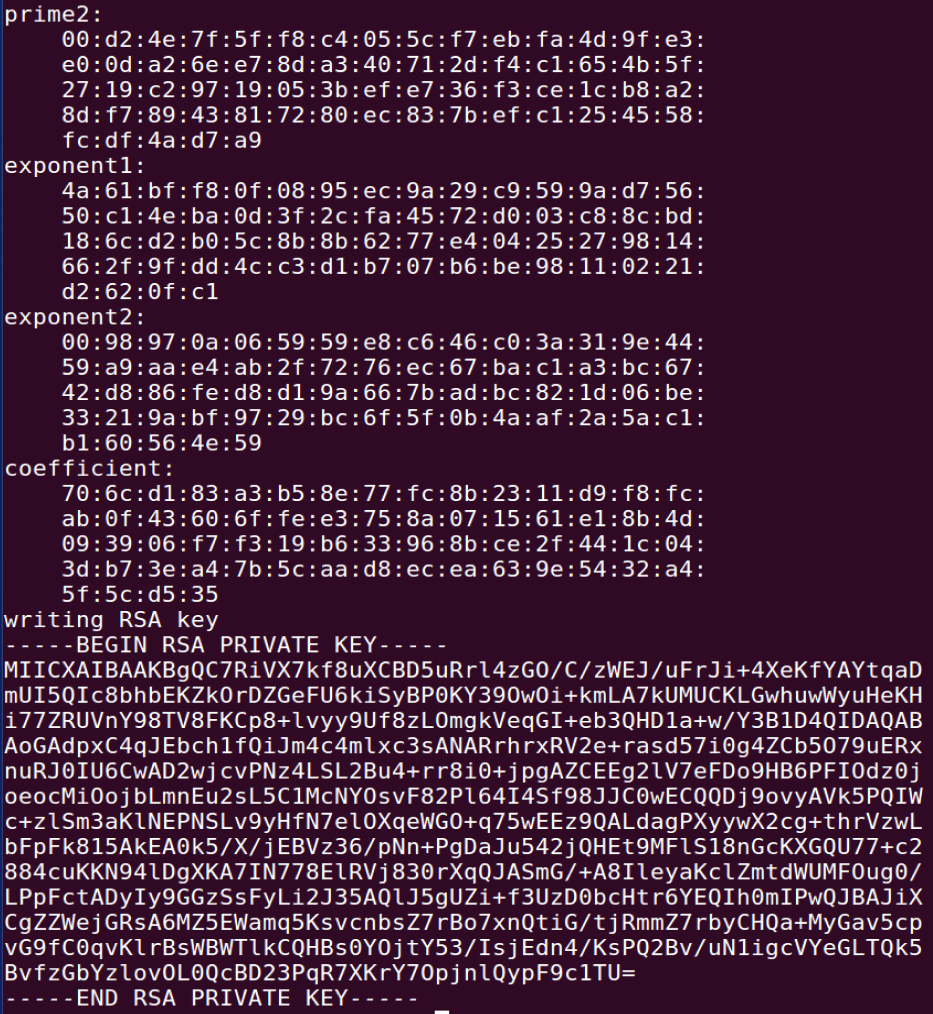


The keys are store in the file server.key in the current directory:



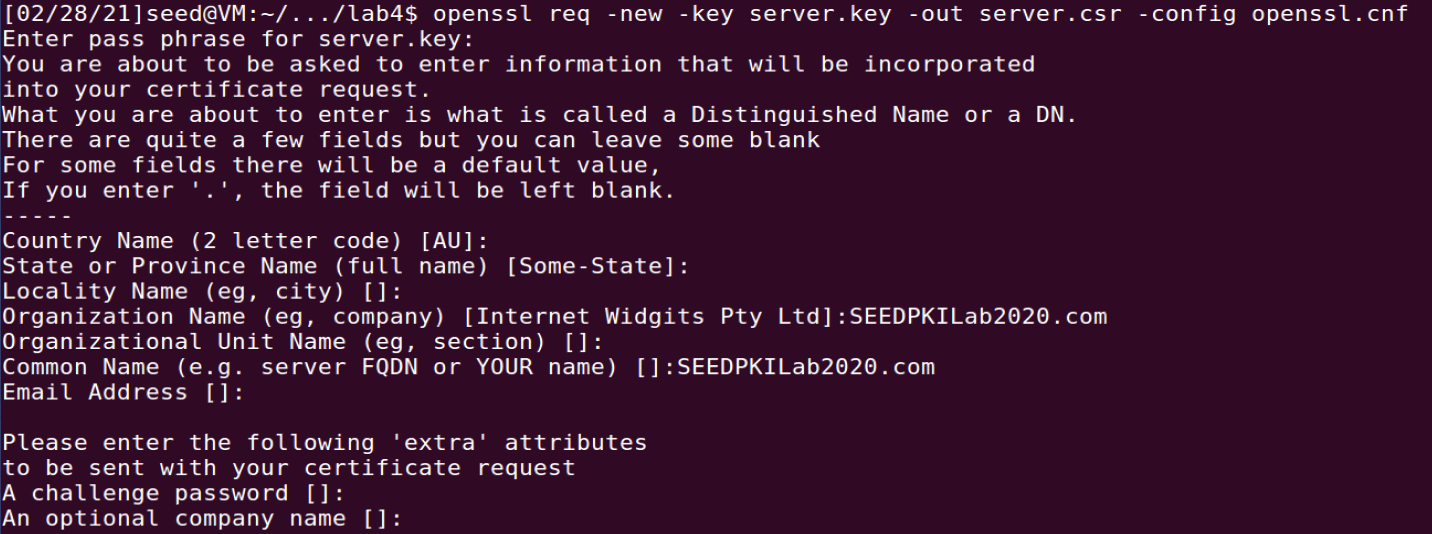
To view the actual content of the encrypted key-file, the following command is run:





Step 2: Generate a Certificate Signing Request (CSR)

The following command is run to generate a certificate signing request for the company. The Organization Name and the Common Name are both specified SEEDPKILab2020.com. The rest of the fields are left as default:

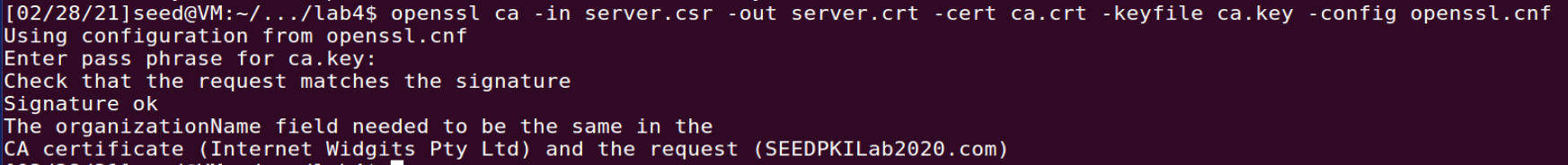


The CSR is generated as server.csr in the current directory:

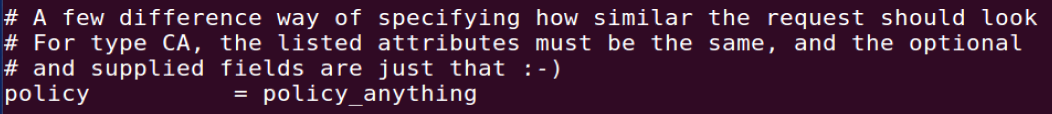


Step 3: Generating Certificates

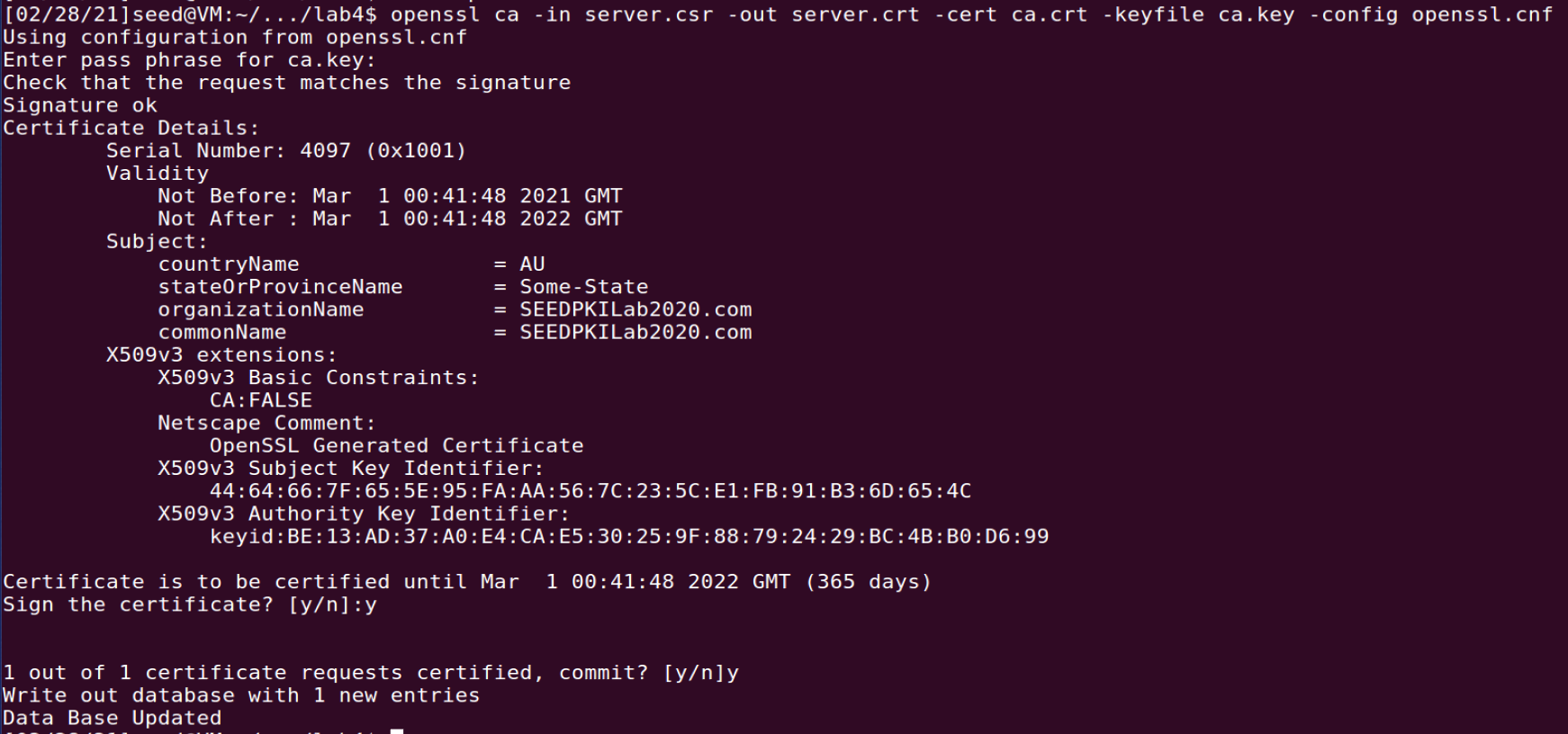
The following command is run to turn the CSR into an X509 certificate using the CA’s certificate and key. As expected, OpenSSL refuses to generate the certificate as the name in the request (SEEDPKILab2020.com) is not the same as that of the CA (Internet Widgits Pty Ltd):



The matching rule policy is then changed in the configuration file from policy\_match to policy\_anything:



The command is re-run and the certificate generation succeeds after the policy is changed:



The company’s certificate is generated as server.crt in the current directory:



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

Step 1: Configuring DNS

The /etc/hosts file is edited to map the hostname SEEDPKILab2020.com to our localhost IP:

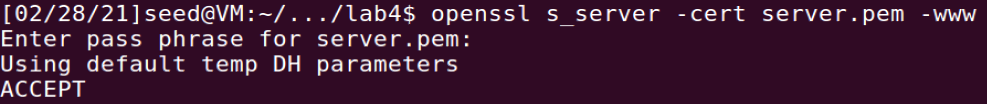


Step 2: Configuring the web server

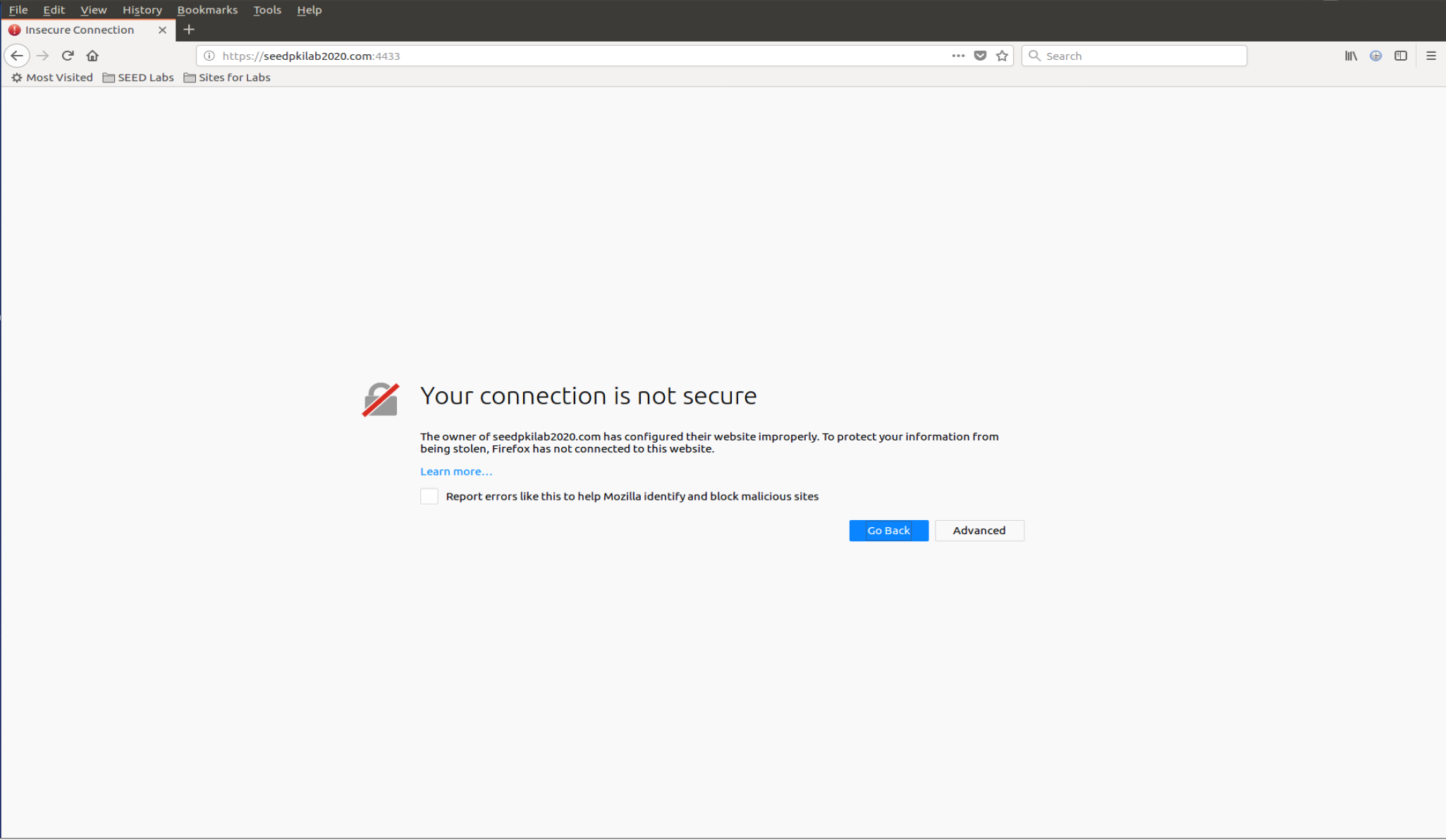
The server’s key and certificate is combined into one file, named server.pem:



The following command is run to start a simple web server using server.pem:

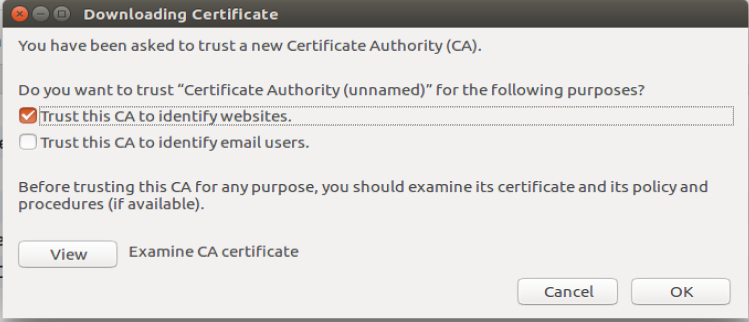


As expected, we get an error message from Firefox telling us that “The owner of seedpkilab2020.com has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.”

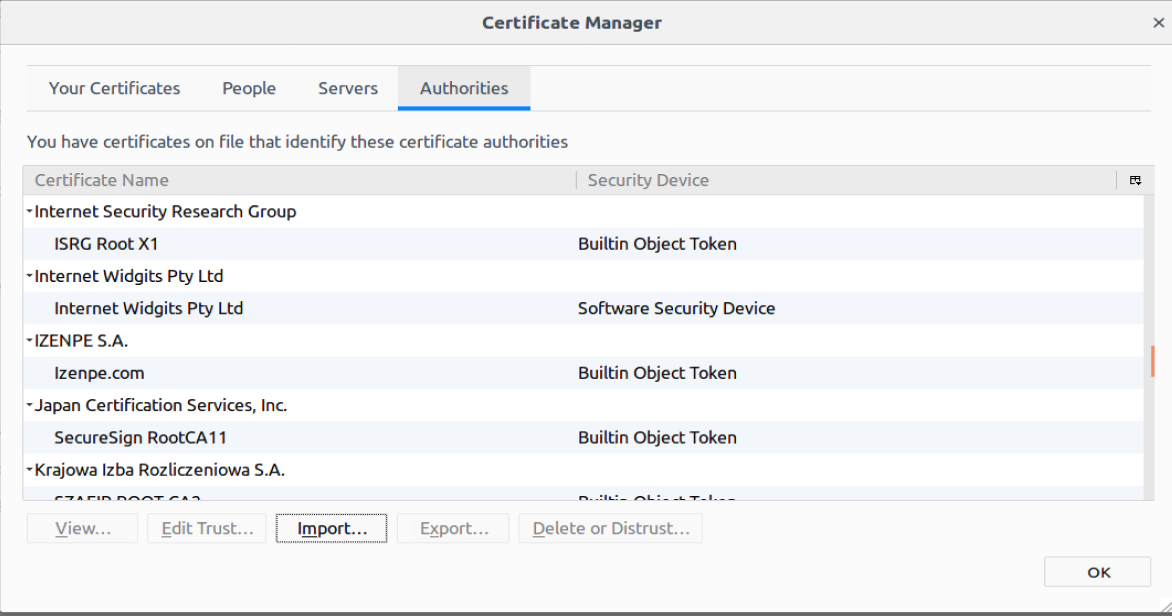


Step 3: Getting the browser to accept our CA certificate

To get the Firefox browser to accept our CA certificate, we manually add this certificate to the Firefox browser:

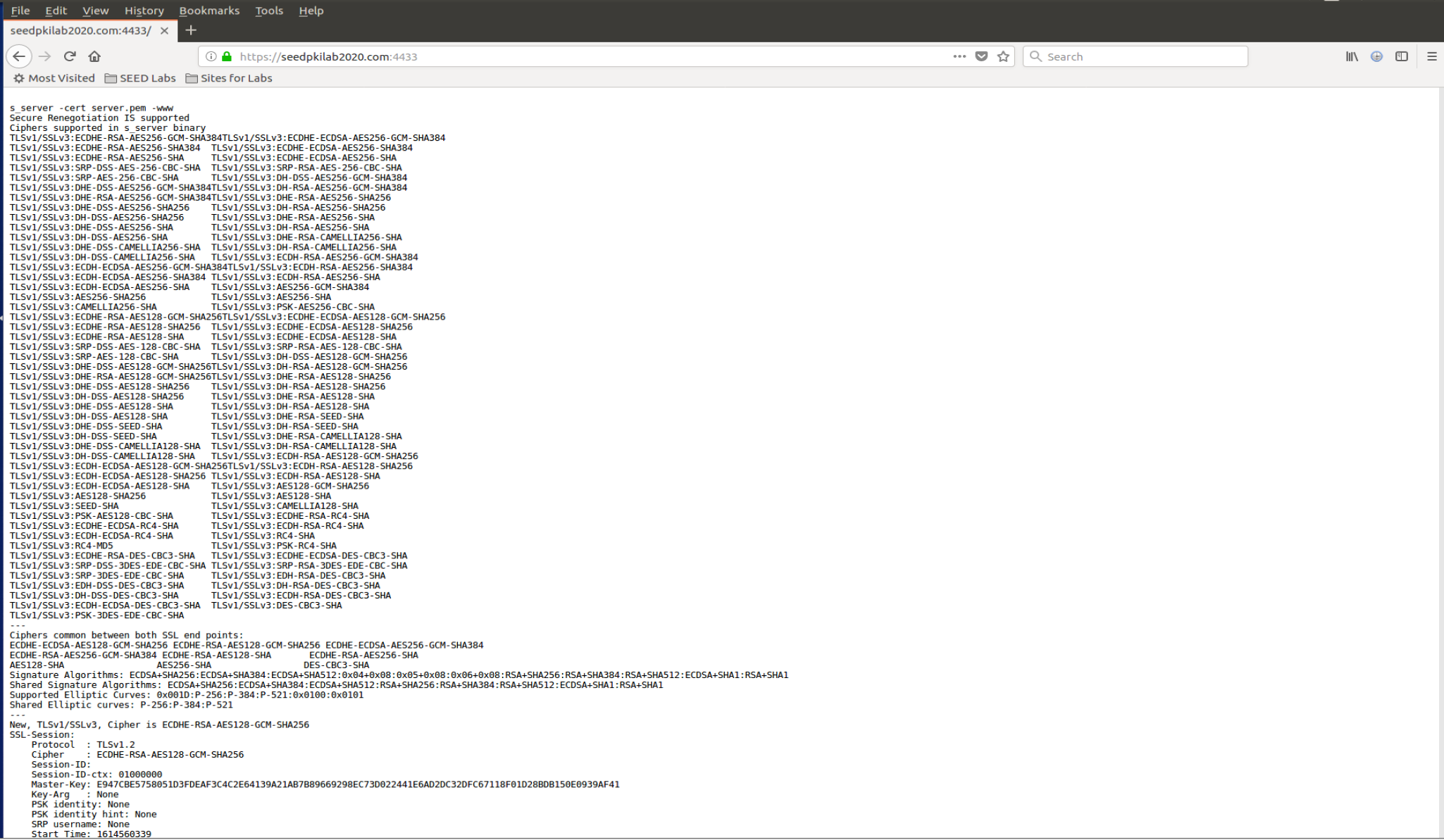


Our CA’s certificate is now in Firefox’s list of the accepted certificates:

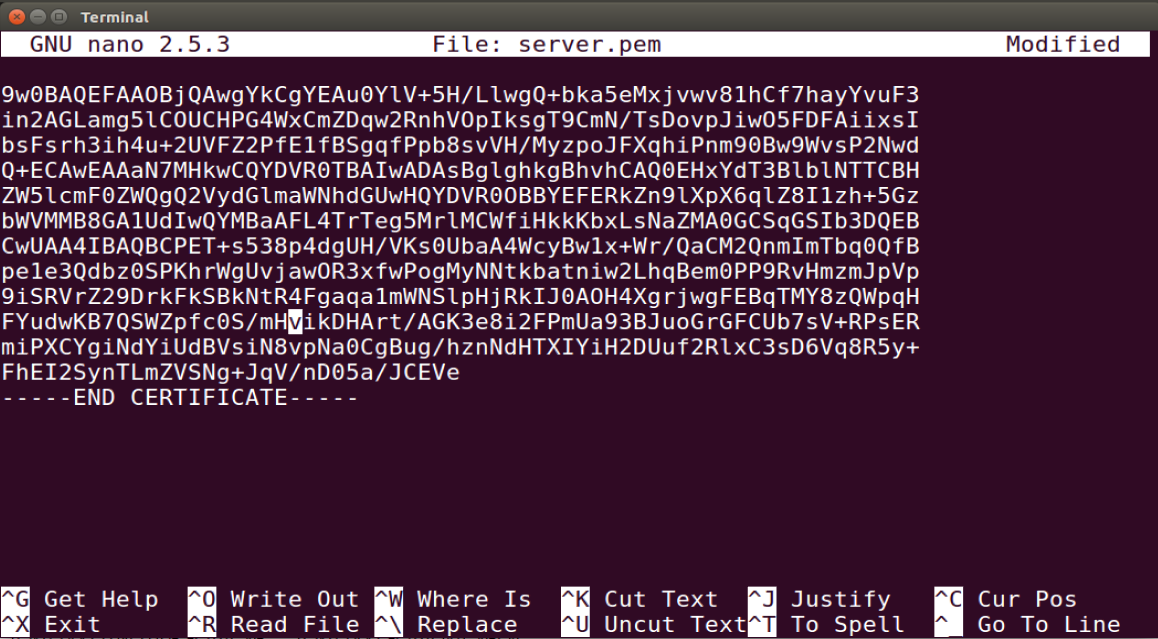


Step 4: Testing our HTTPS website

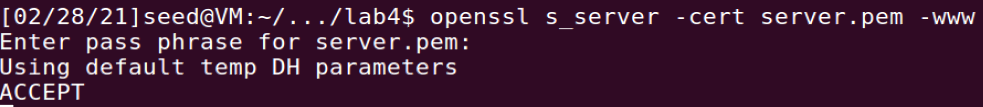
The webpage is refreshed and the page loads successfully:



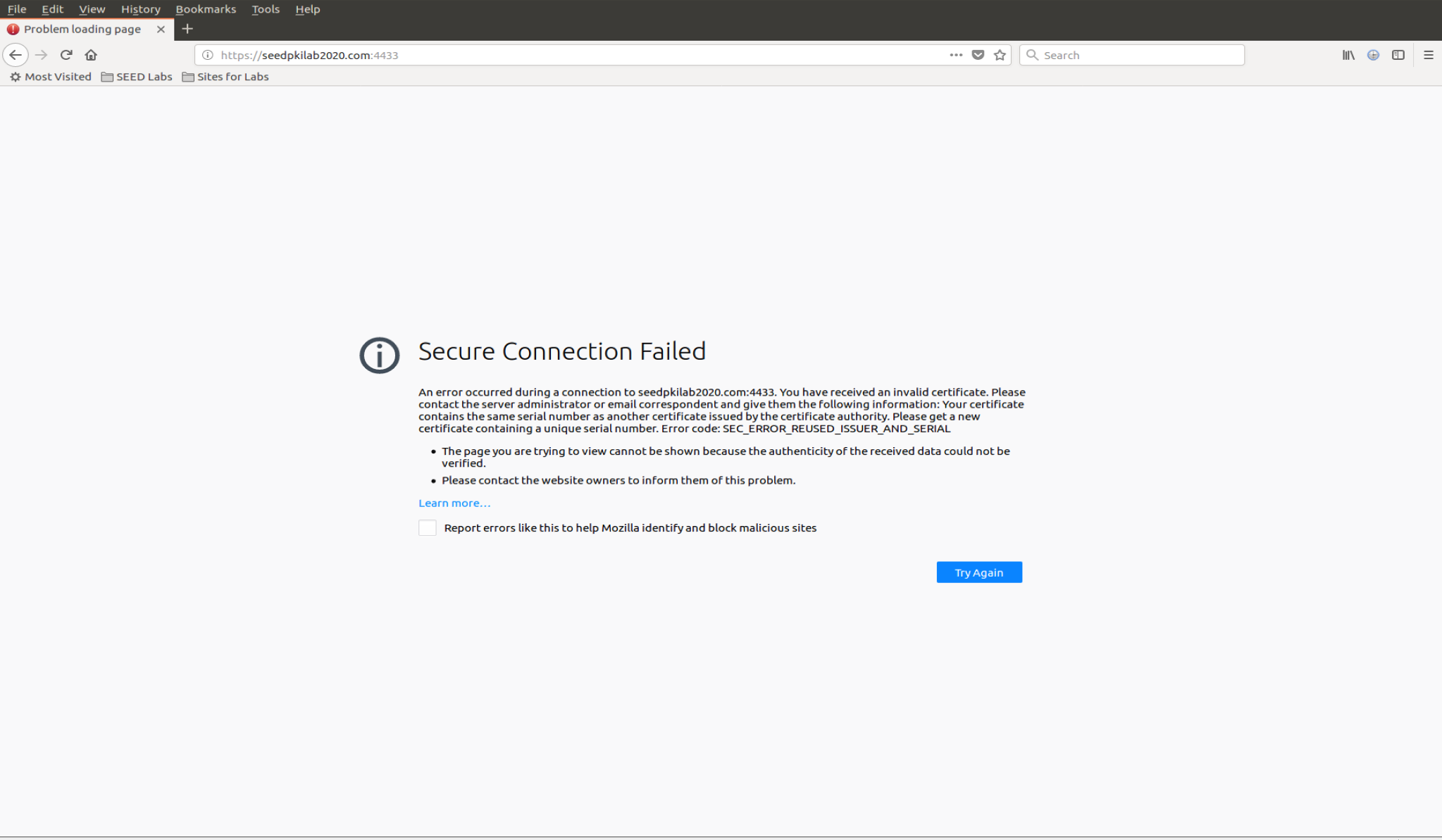
A single byte of server.pem is modified as shown (G is changed to H):



The server is restarted:

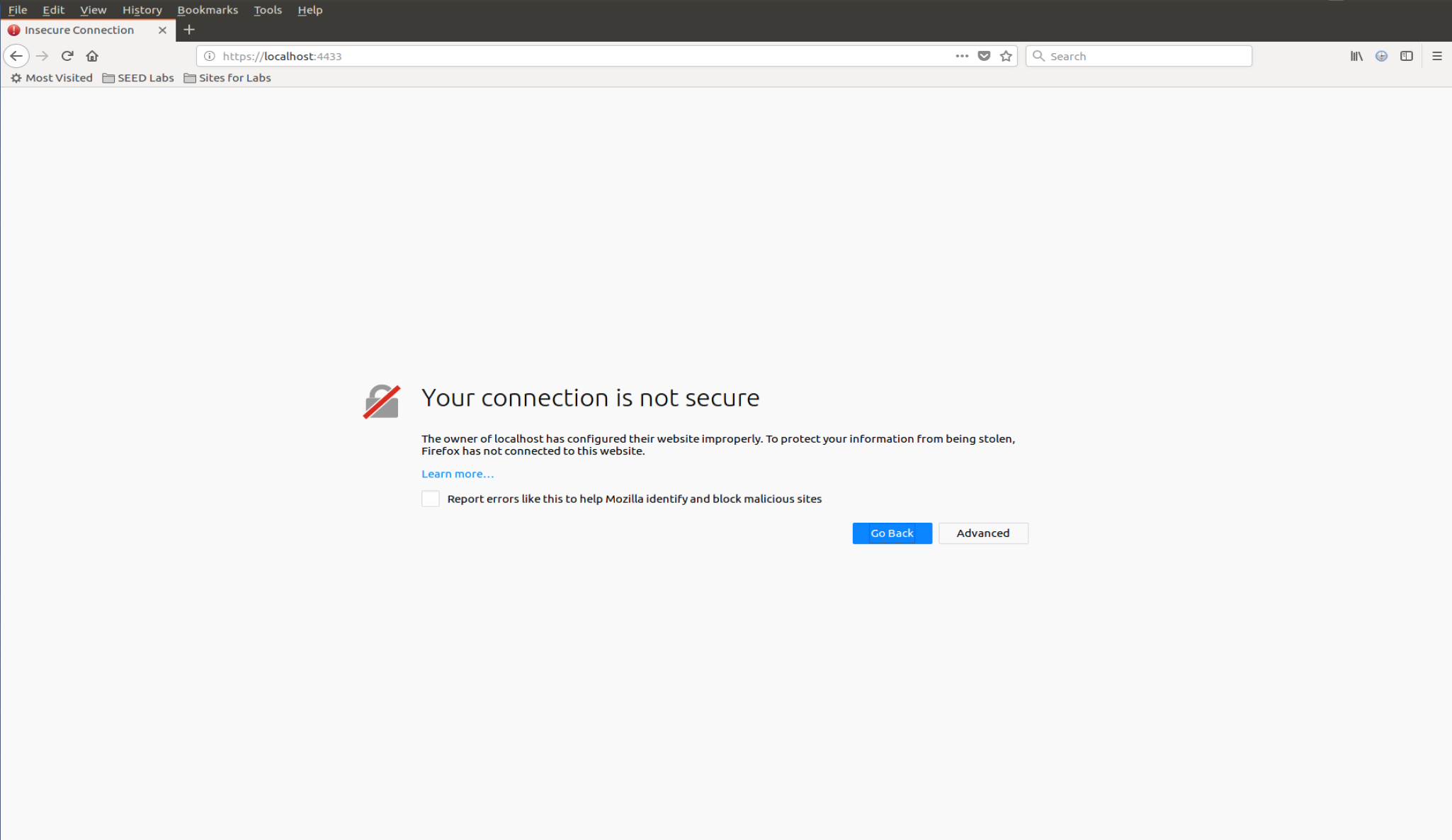


The URL of the browser is reloaded and the webpage fails to load. The message given is “An error occurred during a connection to seedpkilab2020.com:4433. You have received an invalid certificate. Please contact the server administrator or email correspondent and give them the following information: Your certificate contains the same serial number as another certificate issued by the certificate authority. Please get a new certificate containing a unique serial number. Error code: SEC\_ERROR\_REUSED\_ISSUER\_AND\_SERIAL”:



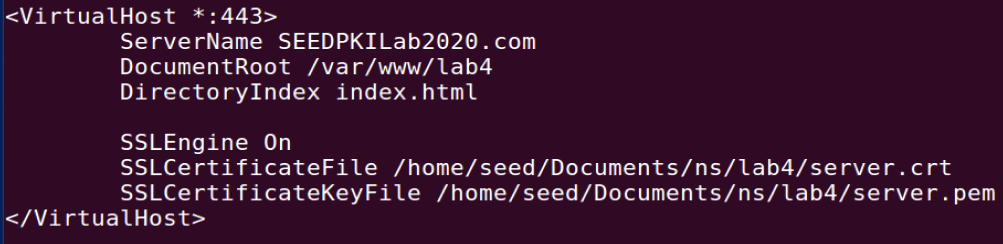
The original server.pem is restored afterwards.

We attempt to connect to SEEDPKILab2020 by using <https://localhost:4433> as the URL. The webpage fails to load as we get an error message from Firefox telling us that “The owner of localhost has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.”

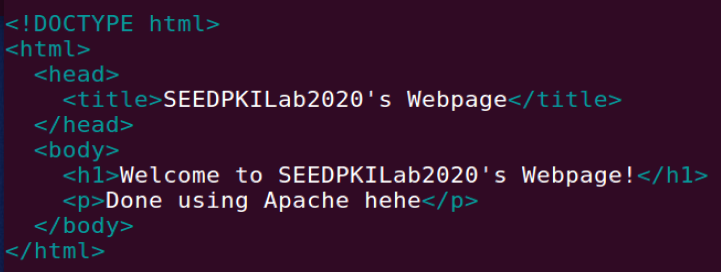


## Task 4: Deploying Certificate in an Apache-Based HTTPSWebsite

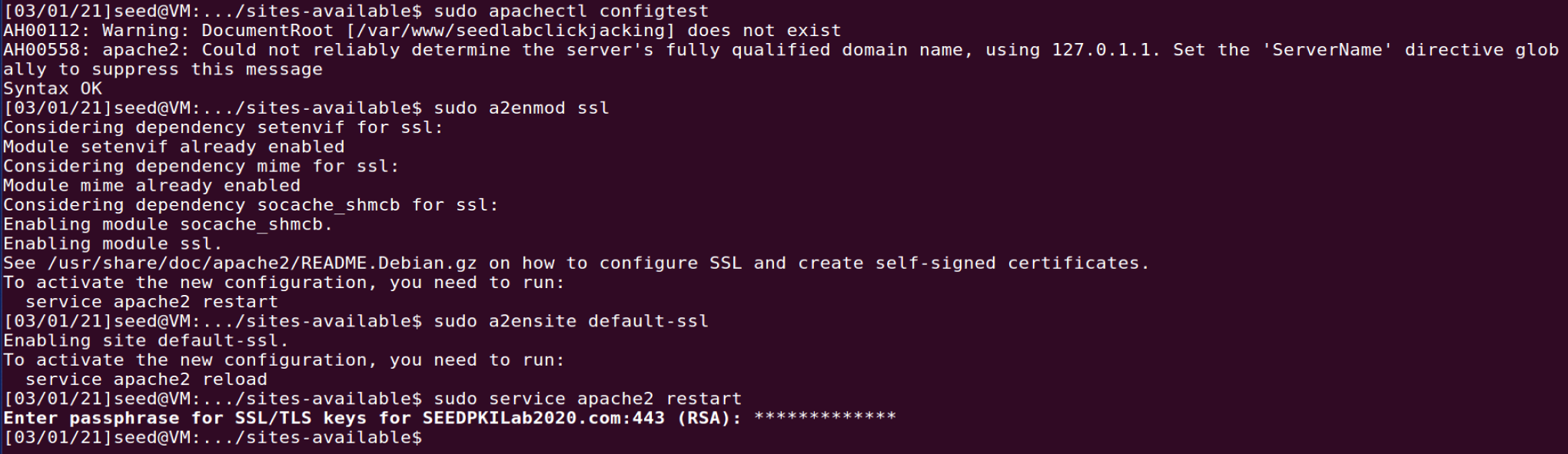
The Apache’s configuration file default-ssl.conf is edited to include the VirtualHost entry as shown below:



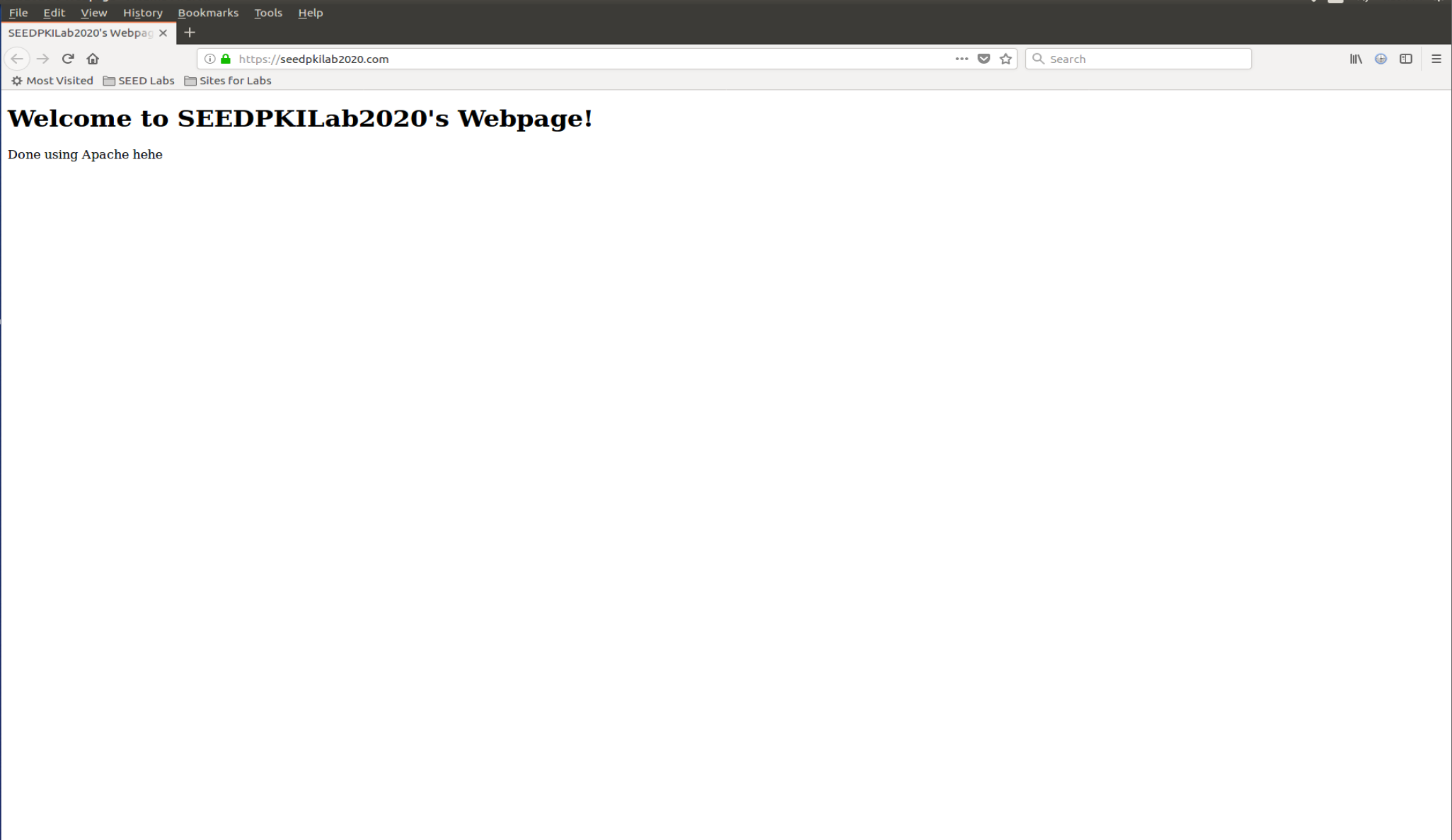
A simple index.html is done in the DocumentRoot folder specified:



A series of commands is run to enable SSL:



We proceed to visit <https://seedpkilab2020.com> on Firefox:



Since the contents of the webpage matches the contents of index.html, we have successfully browsed the HTTPS site.

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

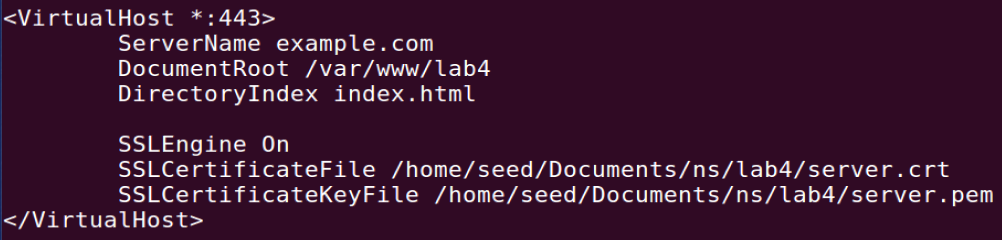
Network Addresses

Malicious Server: 10.0.2.128

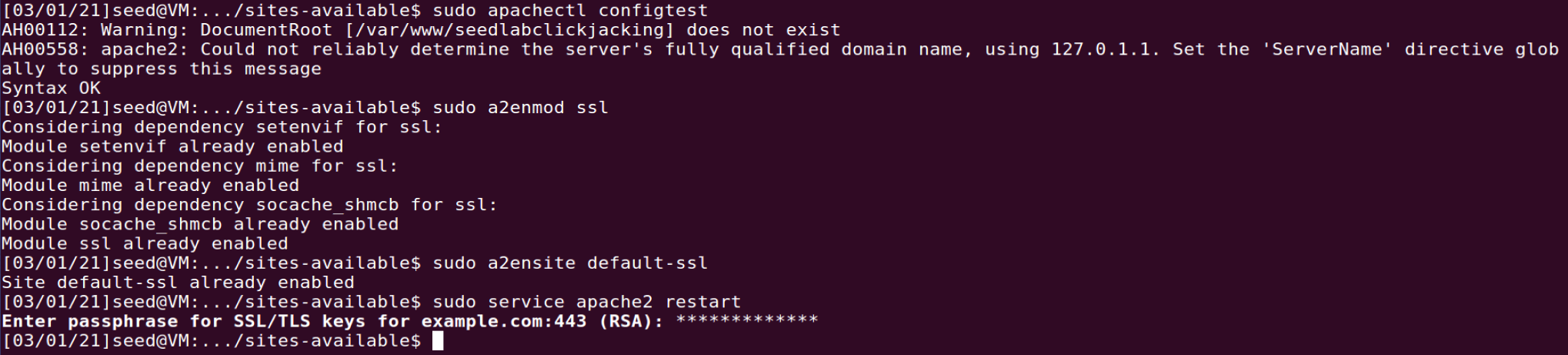
Victim Machine: 10.0.0.129

Step 1: Setting up the malicious website

The VirtualHost entry set up in Task 4 is edited to have the ServerName changed to example.com:



The series of commands are run again:



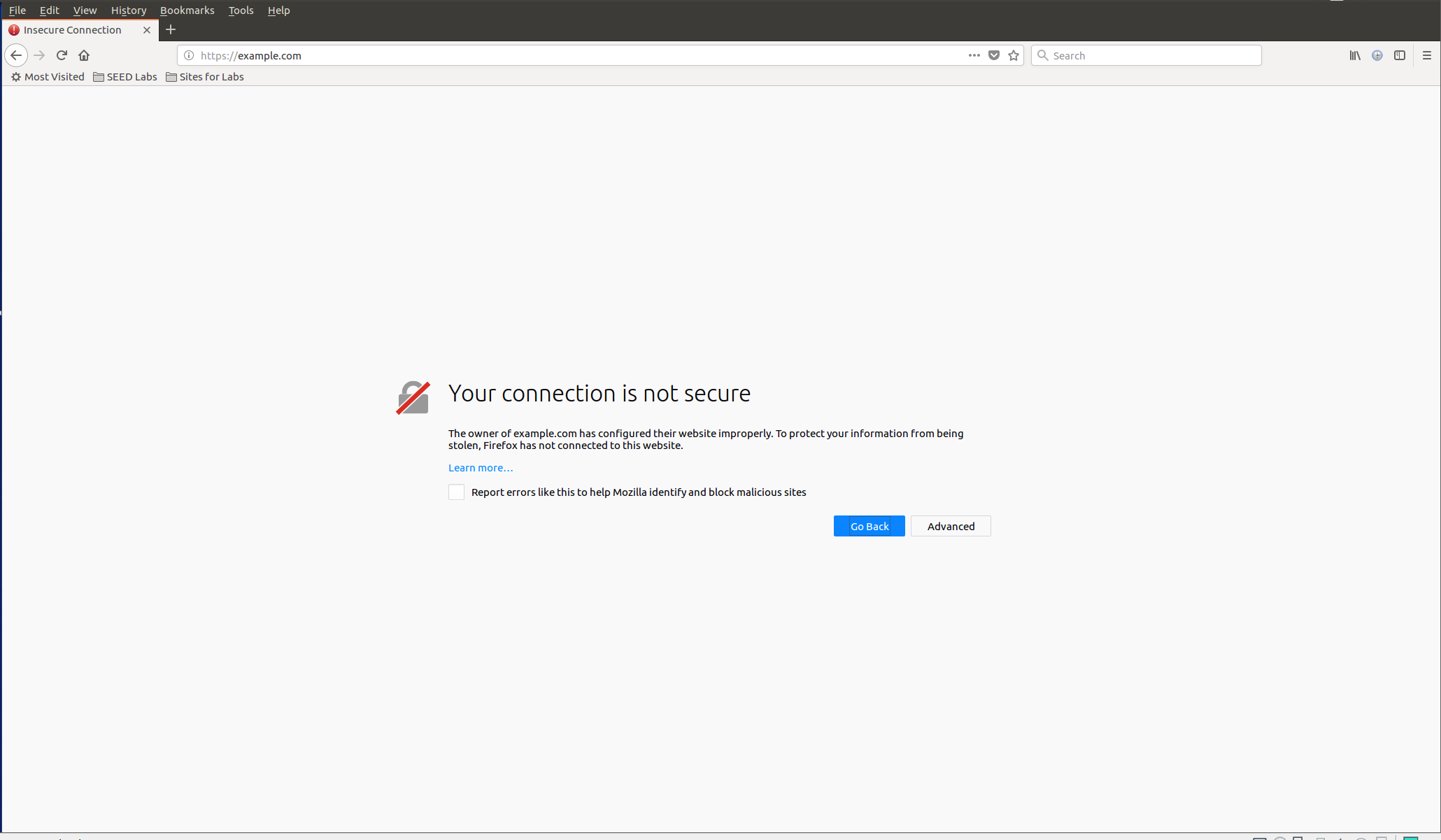
Step 2: Becoming the man in the middle

The victim machine (10.0.2.129)’s /etc/hosts file is edited to have the following entry:

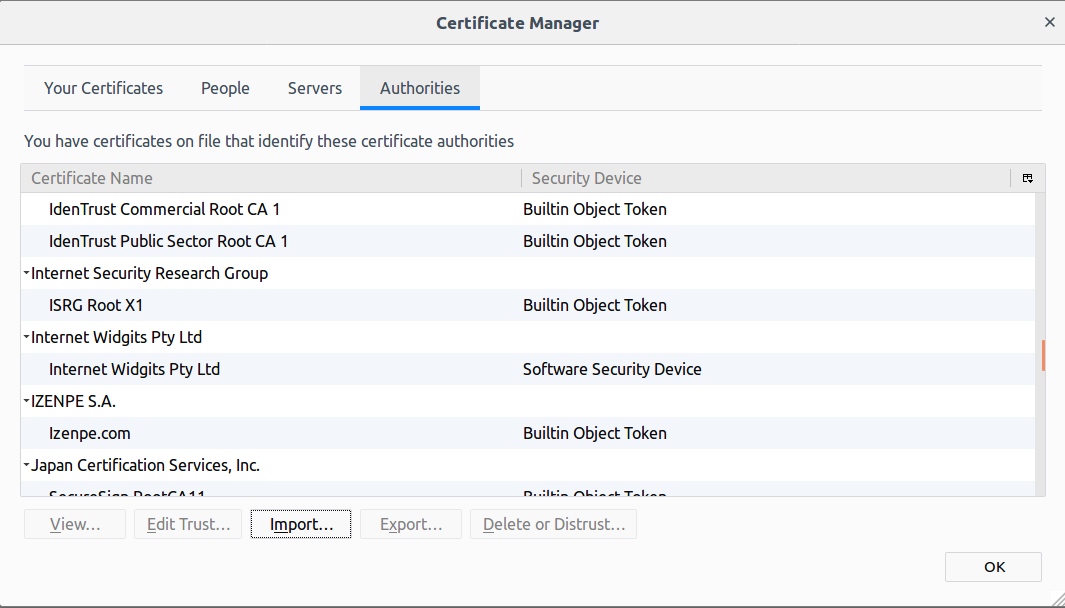


Step 3: Browse the target website

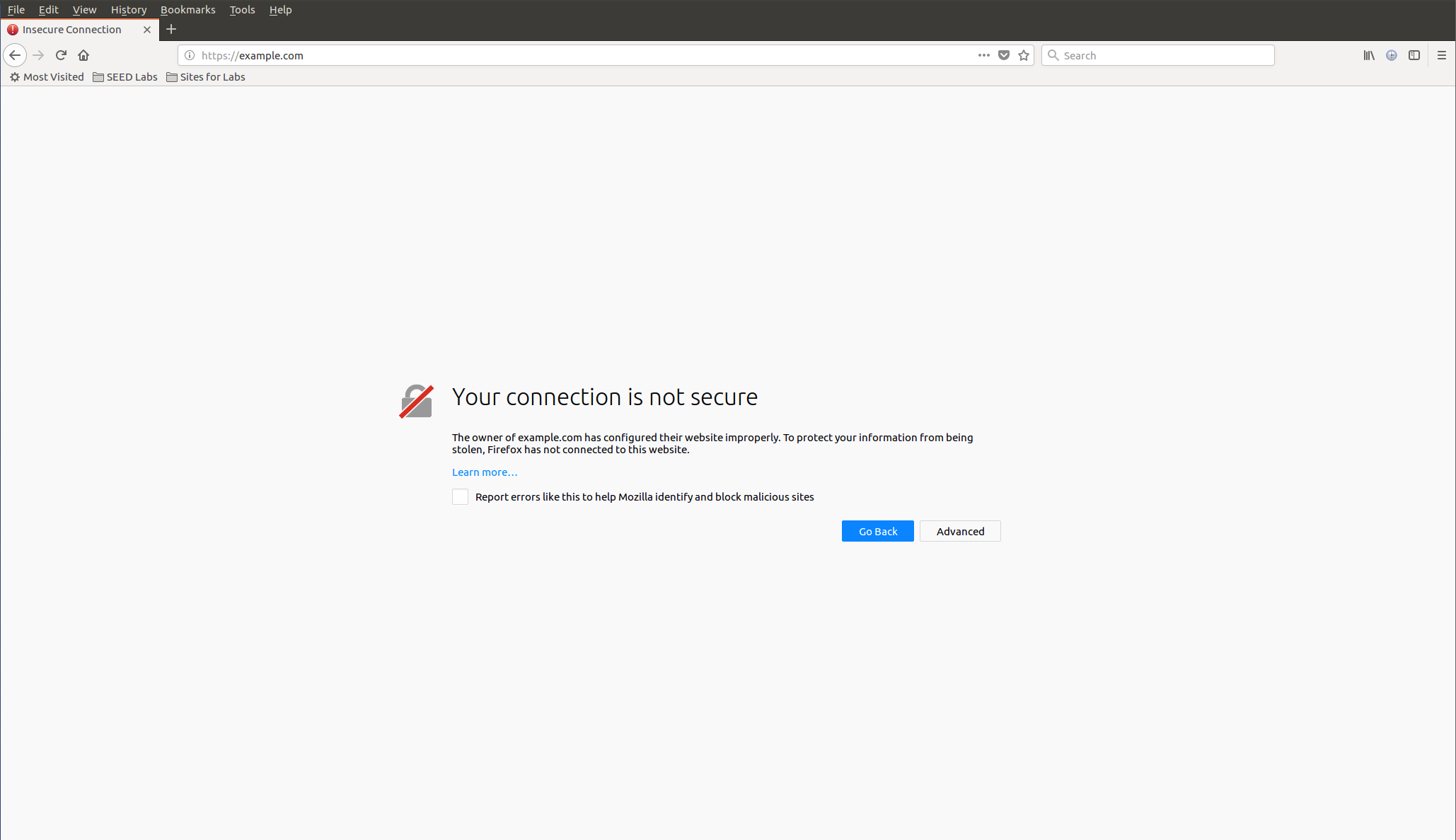
On the victim machine, we proceed to visit <https://example.com> on Firefox:



This is due to the fact that we have not added our root CA’s certificate to the list of recognised CA’s in the victim machine’s browser. We copy ca.crt into this victim machine and add it to the list of recognised CAs:



The changes are saved and the webpage is refreshed but the victim machine’s browser still shows the same message:



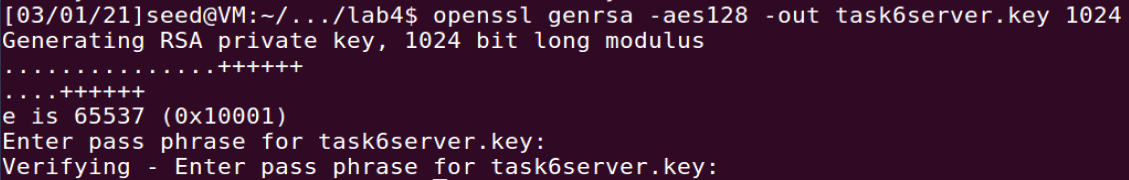
This happens because we are visiting the website with the domain name example.com but the server’s certificate and key-pair were created using the SEEDPKILab2020.com domain name. This mismatch in the certificate’s subject (common name) and the hostname of the server causes this to happen.

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

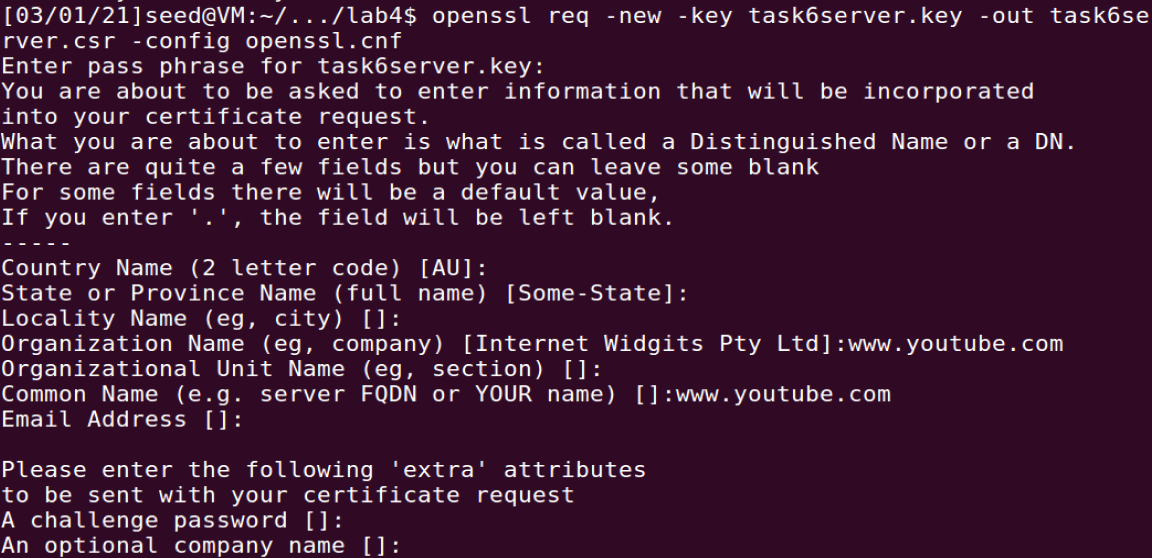
In task 5, the attack fails because the same certificate for SEEDPKILab2020.com is used for example.com. For the MITM attack to succeed, the attacker has to generate a valid certificate with the common name specified as the actual domain name he would like to impose as. For this experiment, we shall use youtube.com.

The steps to create the certificate for the fake website using the root CA’s private key is repeated, as in Task 2.

First, the following command is run to create the company’s public/private key pair in task6server.key. The password *task6company* is used:



The following command is run to generate a certificate signing request for the company. The CSR is saved as task6server.csr. The Organization Name and the Common Name are both specified as www.youtube.com. The rest of the fields are left as default:



After this, the following command is run to turn the CSR into an X509 certificate task6server.crt using the CA’s certificate and key:

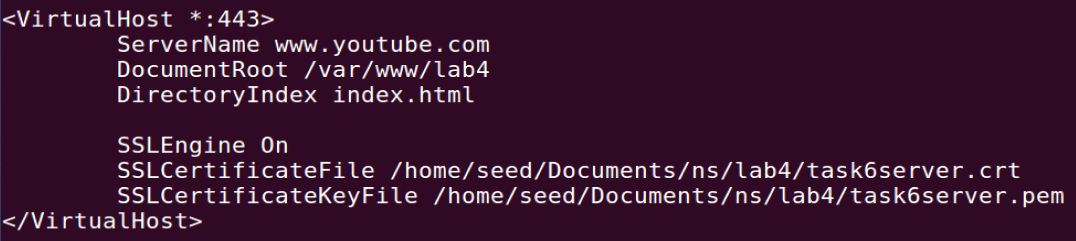


The key and certificate is then combined into one file called task6server.pem:

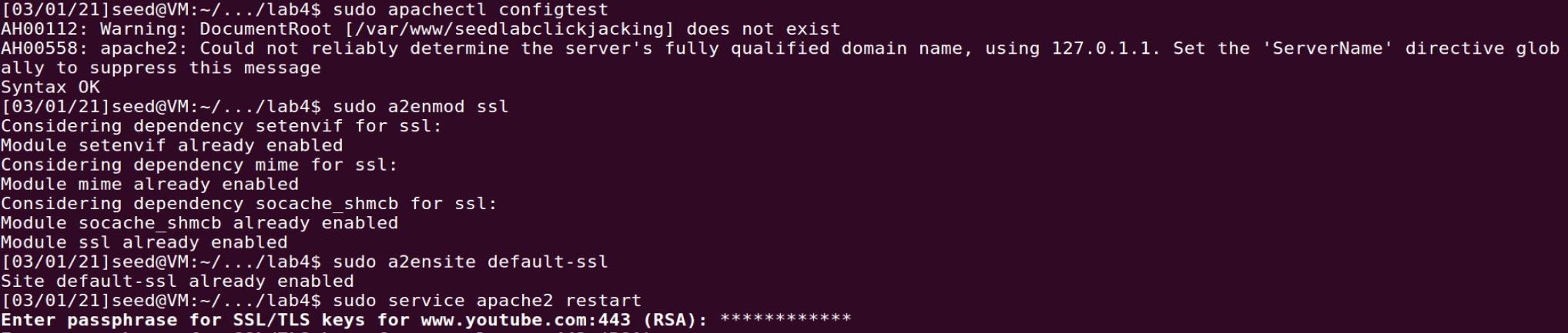


Now that we have created the key and the certificate for the fake [www.youtube.com](http://www.youtube.com) domain, we proceed to set up the Apache server to host this fake webpage.

The VirtualHost entry in the /etc/apache2/sites-available/default-ssl.conf configuration file is edited as shown:



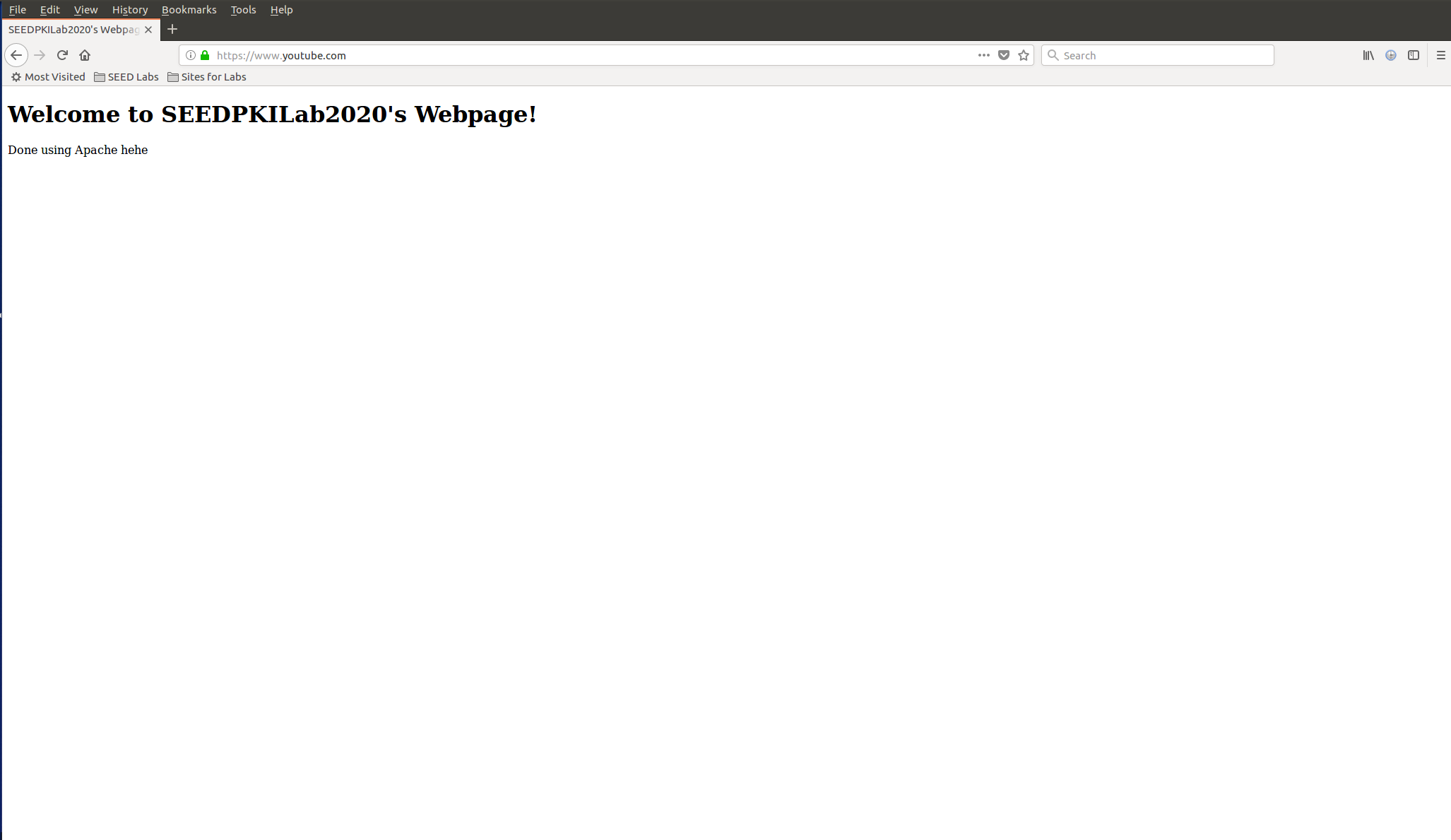
The following commands are run for the Apache server to restart with the changes applied:



Now that the server is up and running, we edit the /etc/hosts file on the victim machine to simulate a DNS poisoning attack:



Finally, we attempt to visit <https://www.youtube.com> on the victim machine:



The MITM attack is successful! ☺