

Public-Key Infrastructure (PKI) Lab Report

Mudit Vats
mpvats@syr.edu
3/2/2019

Table of Contents

Overview	3
Task 1: Becoming a Certificate Authority (CA).....	3
Observations / Explanations	4
Task 2: Creating a Certificate for SEEDPKILab2018.com	4
Step 1: Generate public/private key pair.	5
Step 2: Generate a Certificate Signing Request (CSR)	5
Step 3: Generating Certificates	6
Observations / Explanations	7
Task 3: Deploying Certificate in an HTTPS Web Server	8
Step 1: Configuring DNS	8
Step 2: Configuring the Web Server.....	8
Step 3: Getting the browser to accept our CA certificate	9
Step 4: Testing our HTTPS website	12
Modify a single byte of server.pem.....	13
Observations / Explanations	15
Task 4: Deploying Certificate in an Apache-Based HTTPS Website.....	15
Observations / Explanations	21

Overview

This lab report presents observations and explanations for the tasks described in the [Public-Key Infrastructure \(PKI\) Lab.](#)

Please note, I updated the icons and some theming elements to make the UI a little more pleasing to the eye. It's still the Ubuntu 16.04 Seed VM.

Task 1: Becoming a Certificate Authority (CA)

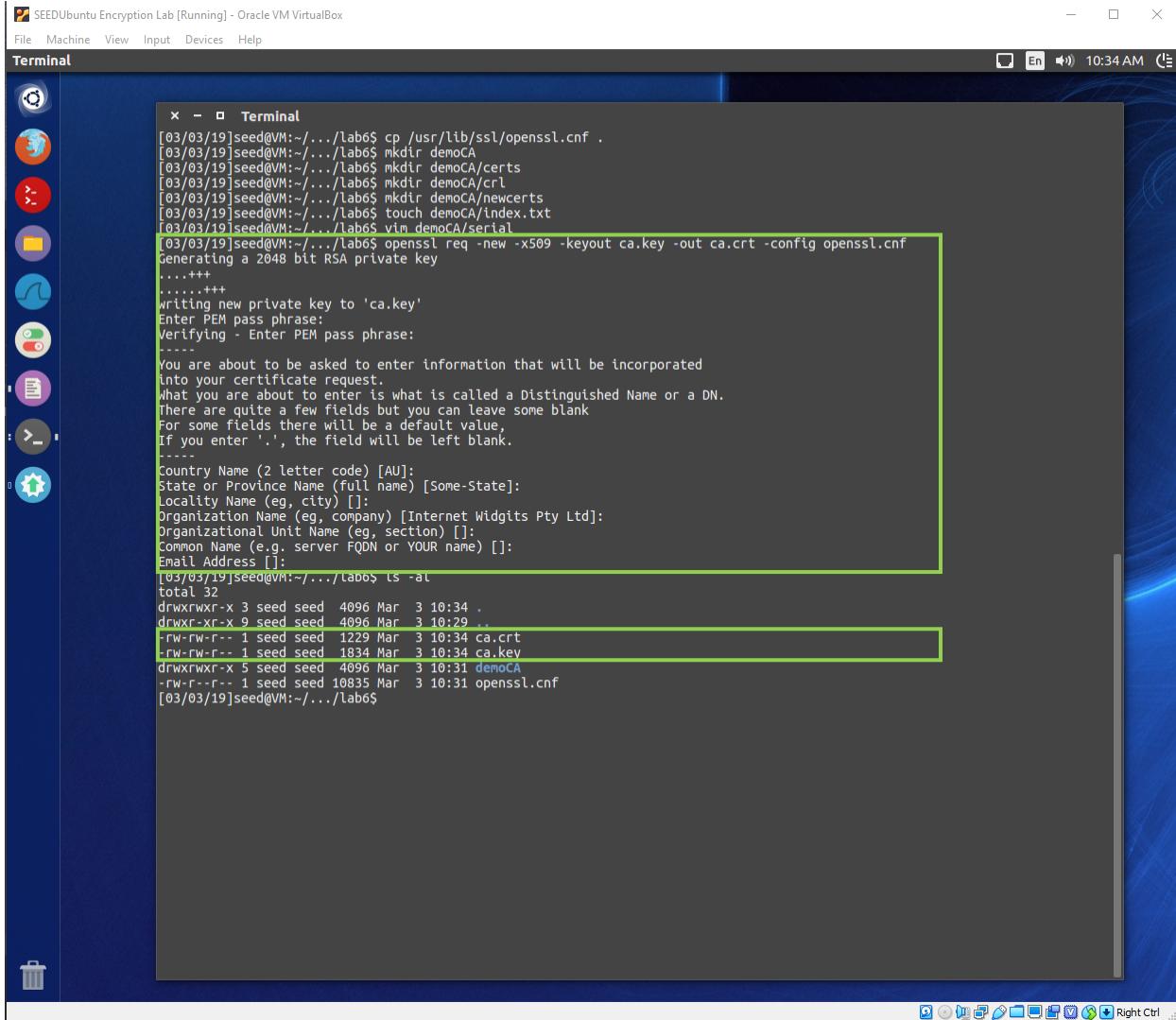
In the figure below, we show the steps to create a self-signed certificate. This self-signed certificate and private key will be used as our Certificate Authority to sign our server's certificate in the next task.

The steps below are as follows:

1. We copy the openssl.cnf file which contains various configuration settings used by the OpenSSL command.
2. We create the demoCA and several subdirectories for the new certs we create, serial number and database file.
3. We then execute the command – “openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf”. This command does the follows:
 - a. “openssl req -new” – Instructs OpenSSL that we have a new request.
 - b. “-x509” – Output x509 certificate format.
 - c. “-keyout ca.key” – Private key is ca.key.
 - d. “-out ca.crt” – Public key is ca.crt.
 - e. “-config openssl.cnf” – Use this configuration file.

The end result is that we have a ca.key, which is our private key and a ca.crt, which is a self-signed certificate that contains our public key.

We can now use the private key to sign other certificates and the public certificate can be used to verify the signed certificate.



The screenshot shows a terminal window within a virtual machine titled "SEEDUbuntu Encryption Lab [Running] - Oracle VM VirtualBox". The terminal session is as follows:

```
[03/03/19]seed@VM:~/.../lab6$ cp /usr/lib/ssl/openssl.cnf .
[03/03/19]seed@VM:~/.../lab6$ mkdir demoCA
[03/03/19]seed@VM:~/.../lab6$ mkdir demoCA/certs
[03/03/19]seed@VM:~/.../lab6$ mkdir demoCA/crl
[03/03/19]seed@VM:~/.../lab6$ mkdir demoCA/newcerts
[03/03/19]seed@VM:~/.../lab6$ touch demoCA/Index.txt
[03/03/19]seed@VM:~/.../lab6$ vim demoCA/serial
[03/03/19]seed@VM:~/.../lab6$ openssl req -new -x509 -keyout ca.key -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]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
[03/03/19]seed@VM:~/.../lab6$ ls -al
total 32
drwxrwxr-x 3 seed seed 4096 Mar  3 10:34 .
drwxr-xr-x 9 seed seed 4096 Mar  3 10:29 ..
-rw-rw-r-- 1 seed seed 1229 Mar  3 10:34 ca.crt
-rw-rw-r-- 1 seed seed 1834 Mar  3 10:34 ca.key
drwxrwxr-x 5 seed seed 4096 Mar  3 10:31 demoCA
-rw-r--r-- 1 seed seed 10835 Mar  3 10:31 openssl.cnf
[03/03/19]seed@VM:~/.../lab6$
```

Observations / Explanations

This task focused on creating a self-signed certificate to act as our Certificate Authority that will sign our server's public key; i.e. create a certificate which the server will send down to the web browser (client) when the https session is initiated.

This certificate is self-signed, meaning that not "official" CA is vouching for it, we are vouching for our own certificate, which is fine for experimentation.

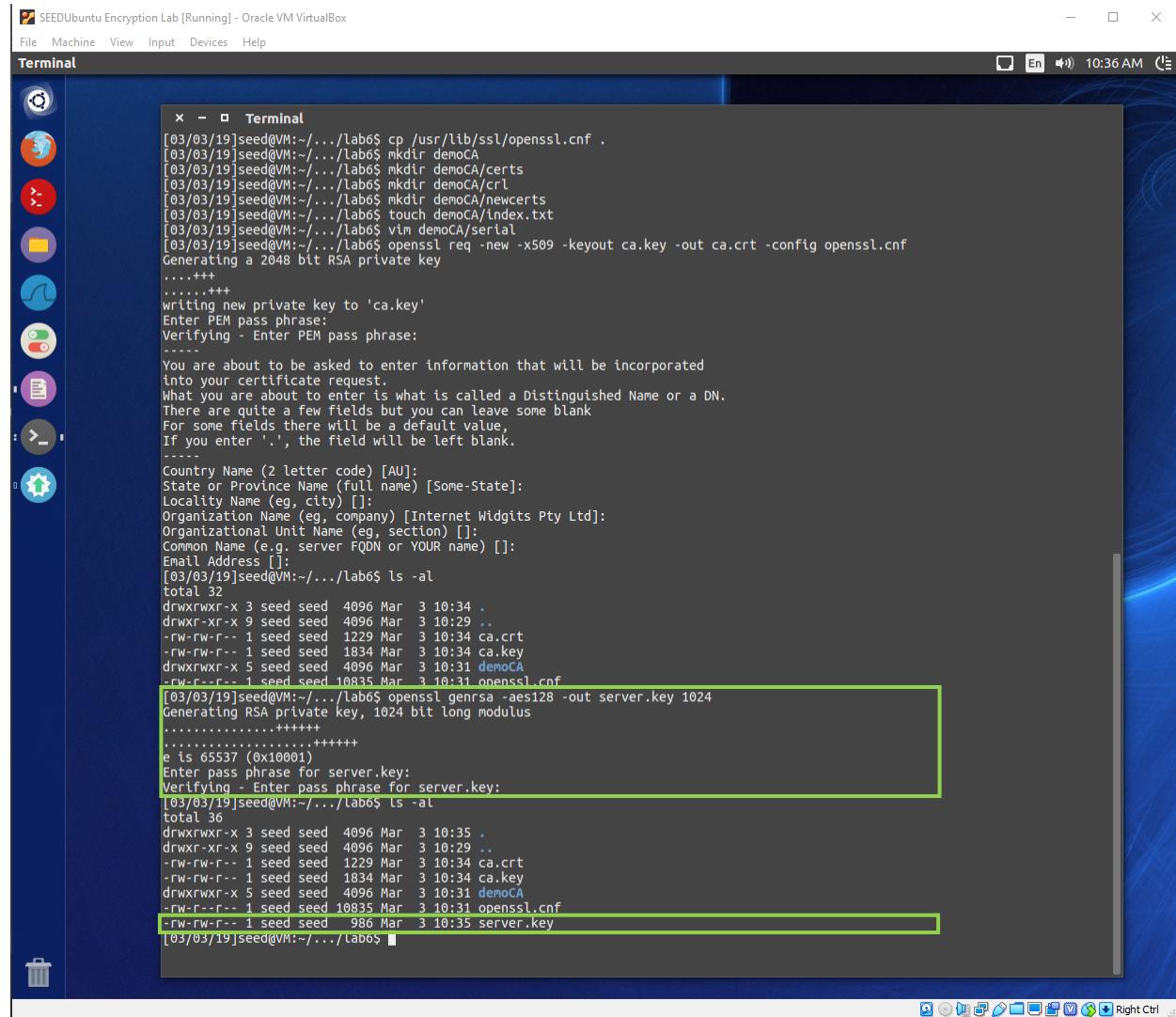
Task 2: Creating a Certificate for SEEDPKILab2018.com

In the following sections, we will go through the three steps to create the certificate for our web site.

Step 1: Generate public/private key pair.

In the figure below, we can see the generation of the server public/private key pair. The file server.key contains both the public and private key.

We execute the command “openssl genrsa -aes128 -out server.key 1024” which generates an RSA public / private key pair using the AES128 algorithm.



The screenshot shows a terminal window titled "Terminal" running on a SEEDUbuntu Encryption Lab [Running] - Oracle VM VirtualBox. The terminal output is as follows:

```
[03/03/19]seed@VM:~/.../lab6$ cp /usr/lib/ssl/openssl.cnf .
[03/03/19]seed@VM:~/.../lab6$ mkdir demoCA
[03/03/19]seed@VM:~/.../lab6$ mkdir demoCA/certs
[03/03/19]seed@VM:~/.../lab6$ mkdir demoCA/crl
[03/03/19]seed@VM:~/.../lab6$ mkdir demoCA/newcerts
[03/03/19]seed@VM:~/.../lab6$ touch demoCA/index.txt
[03/03/19]seed@VM:~/.../lab6$ vim demoCA/serial
[03/03/19]seed@VM:~/.../lab6$ openssl req -new -x509 -keyout ca.key -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]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
[03/03/19]seed@VM:~/.../lab6$ ls -al
total 32
drwxrwxr-x 3 seed seed 4096 Mar  3 10:34 .
drwxr-xr-x 9 seed seed 4096 Mar  3 10:29 ..
-rw-rw-r-- 1 seed seed 1229 Mar  3 10:34 ca.crt
-rw-rw-r-- 1 seed seed 1834 Mar  3 10:34 ca.key
drwxrwxr-x 5 seed seed 4096 Mar  3 10:31 demoCA
-rw-r--r-- 1 seed seed 10835 Mar  3 10:31 openssl.cnf
[03/03/19]seed@VM:~/.../lab6$ 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:
[03/03/19]seed@VM:~/.../lab6$ ls -al
total 36
drwxrwxr-x 3 seed seed 4096 Mar  3 10:35 .
drwxr-xr-x 9 seed seed 4096 Mar  3 10:29 ..
-rw-rw-r-- 1 seed seed 1229 Mar  3 10:34 ca.crt
-rw-rw-r-- 1 seed seed 1834 Mar  3 10:34 ca.key
drwxrwxr-x 5 seed seed 4096 Mar  3 10:31 demoCA
-rw-r--r-- 1 seed seed 10835 Mar  3 10:31 openssl.cnf
-rw-rw-r-- 1 seed seed 986 Mar  3 10:35 server.key
[03/03/19]seed@VM:~/.../lab6$
```

Step 2: Generate a Certificate Signing Request (CSR)

In the figure below, we generated the CSR. We see the server.csr in the current directory. We can also see, above that, that we specified the Common Name to be SEEDPKILab2018.com.

We execute the command “openssl req -new -key server.key -out server.csr -config openssl.cnf” to create the new csr (server.csr). As mentioned, we take care to specify the common name so it matches our website's name.

```

[03/03/19]seed@VM:~/.../lab6$ ls -al
total 32
drwxrwxr-x 3 seed seed 4096 Mar  3 10:34 .
drwxr-xr-x 9 seed seed 4096 Mar  3 10:29 ..
-rw-rw-r-- 1 seed seed 1229 Mar  3 10:34 ca.crt
-rw-rw-r-- 1 seed seed 1834 Mar  3 10:34 ca.key
drwxrwxr-x 5 seed seed 4096 Mar  3 10:31 demoCA
-rw-r--r-- 1 seed seed 10835 Mar  3 10:31 openssl.cnf
[03/03/19]seed@VM:~/.../lab6$ 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:
[03/03/19]seed@VM:~/.../lab6$ ls -al
total 36
drwxrwxr-x 3 seed seed 4096 Mar  3 10:35 .
drwxr-xr-x 9 seed seed 4096 Mar  3 10:29 ..
-rw-rw-r-- 1 seed seed 1229 Mar  3 10:34 ca.crt
-rw-rw-r-- 1 seed seed 1834 Mar  3 10:34 ca.key
drwxrwxr-x 5 seed seed 4096 Mar  3 10:31 demoCA
-rw-r--r-- 1 seed seed 10835 Mar  3 10:31 openssl.cnf
-rw-rw-r-- 1 seed seed 986 Mar  3 10:35 server.key
[03/03/19]seed@VM:~/.../lab6$ 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]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:SEEDPKILab2018.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[03/03/19]seed@VM:~/.../lab6$ ls -al
total 40
drwxrwxr-x 3 seed seed 4096 Mar  3 10:38 .
drwxr-xr-x 9 seed seed 4096 Mar  3 10:29 ..
-rw-rw-r-- 1 seed seed 1229 Mar  3 10:34 ca.crt
-rw-rw-r-- 1 seed seed 1834 Mar  3 10:34 ca.key
drwxrwxr-x 5 seed seed 4096 Mar  3 10:31 demoCA
-rw-r--r-- 1 seed seed 10835 Mar  3 10:31 openssl.cnf
-rw-rw-r-- 1 seed seed 643 Mar  3 10:38 server.csr
-rw-rw-r-- 1 seed seed 986 Mar  3 10:35 server.key
[03/03/19]seed@VM:~/.../lab6$ 
```

Step 3: Generating Certificates

In the figure below, we use the CSR and CA private key as input and generate a signed certificate for the server.

We execute the command “`openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf`” to generate the `server.crt` which is the server’s public key, plus some metadata (Common Name, certificate expiry etc) and signed with the CA’s key file.

```

SEEDUbuntu Encryption Lab [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
x - Terminal
A challenge password []:
An optional company name []:
[03/03/19]seed@VM:~/.../lab6$ ls -al
total 40
drwxrwxr-x 3 seed seed 4096 Mar  3 10:38 .
drwxr-xr-x 9 seed seed 4096 Mar  3 10:29 ..
-rw-rw-r-- 1 seed seed 1229 Mar  3 10:34 ca.crt
-rw-rw-r-- 1 seed seed 1834 Mar  3 10:34 ca.key
drwxrwxr-x 5 seed seed 4096 Mar  3 10:31 demoCA
-rw-r--r-- 1 seed seed 10835 Mar  3 10:31 openssl.cnf
-rw-rw-r-- 1 seed seed 643 Mar  3 10:38 server.csr
-rw-rw-r-- 1 seed seed 986 Mar  3 10:35 server.key
[03/03/19]seed@VM:~/.../lab6$ 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  3 17:40:17 2019 GMT
        Not After : Mar  2 17:40:17 2020 GMT
    Subject:
        countryName          = AU
        stateOrProvinceName = Some-State
        organizationName    = Internet Widgets Pty Ltd
        commonName           = SEEDPKILab2018.com
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
        Netscape Comment:
        OpenSSL Generated Certificate
        X509v3 Subject Key Identifier:
            CB:6D:09:CA:08:DE:62:3B:F0:4E:EE:29:B6:2B:81:D3:24:23:D6
        X509v3 Authority Key Identifier:
            keyid:E9:58:DA:B0:EF:BF:DB:9C:98:37:20:1D:C3:D2:6B:0B:ED:44:D3:20
Certificate is to be certified until Mar  2 17:40:17 2020 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
[03/03/19]seed@VM:~/.../lab6$ ls -al
total 44
drwxrwxr-x 3 seed seed 4096 Mar  3 10:40 .
drwxr-xr-x 9 seed seed 4096 Mar  3 10:29 ..
-rw-rw-r-- 1 seed seed 1229 Mar  3 10:34 ca.crt
-rw-rw-r-- 1 seed seed 1834 Mar  3 10:34 ca.key
drwxrwxr-x 5 seed seed 4096 Mar  3 10:40 demoCA
-rw-r--r-- 1 seed seed 10835 Mar  3 10:31 openssl.cnf
-rw-rw-r-- 1 seed seed 3681 Mar  3 10:40 server.crt
-rw-rw-r-- 1 seed seed 643 Mar  3 10:38 server.csr
-rw-rw-r-- 1 seed seed 986 Mar  3 10:35 server.key
[03/03/19]seed@VM:~/.../lab6$
```

Observations / Explanations

This task focused on ultimately generating a server certificate which is the server's public key and some additional metadata (e.g. Common Name) signed by the Certificate Authority's private key (ca.key).

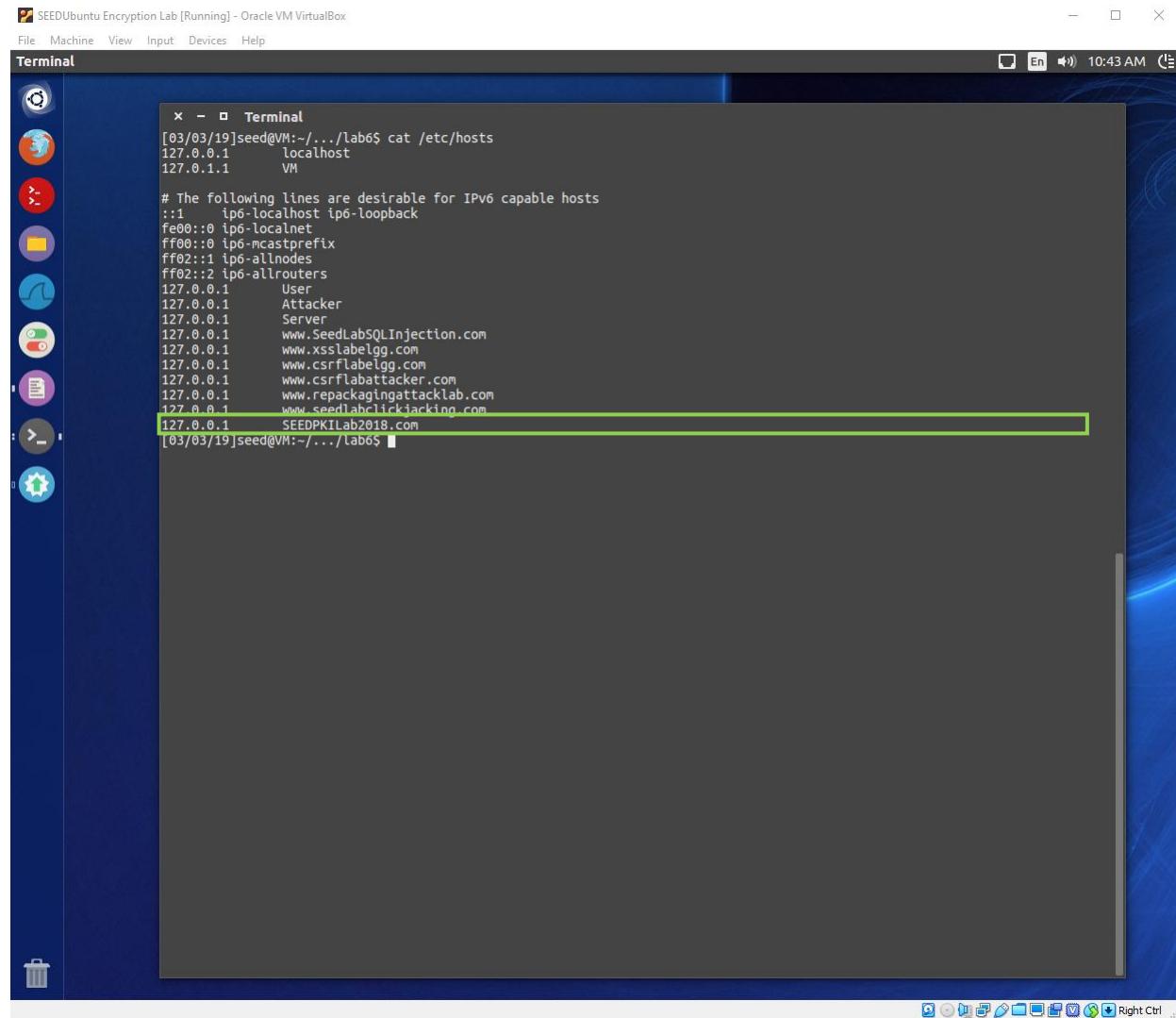
This server certificate is sent to the client when an https connection is made to the server. The client will then verify the server's certificate using the CA's certificate which we will load into the web browser.

Additionally, the web browser will use the Common Name to ensure the to double-check that the server certificate which was sent and the site that the user is accessing is the same (i.e. as specified in the URL and in the Common Name of the certificate). If so, the connection proceeds. If not, there may be an invalid certificate so the connection does not proceed.

Task 3: Deploying Certificate in an HTTPS Web Server

Step 1: Configuring DNS

In this step, we point the SEEDPKILab2018.com name to our localhost since we'll be running the website on the localhost.



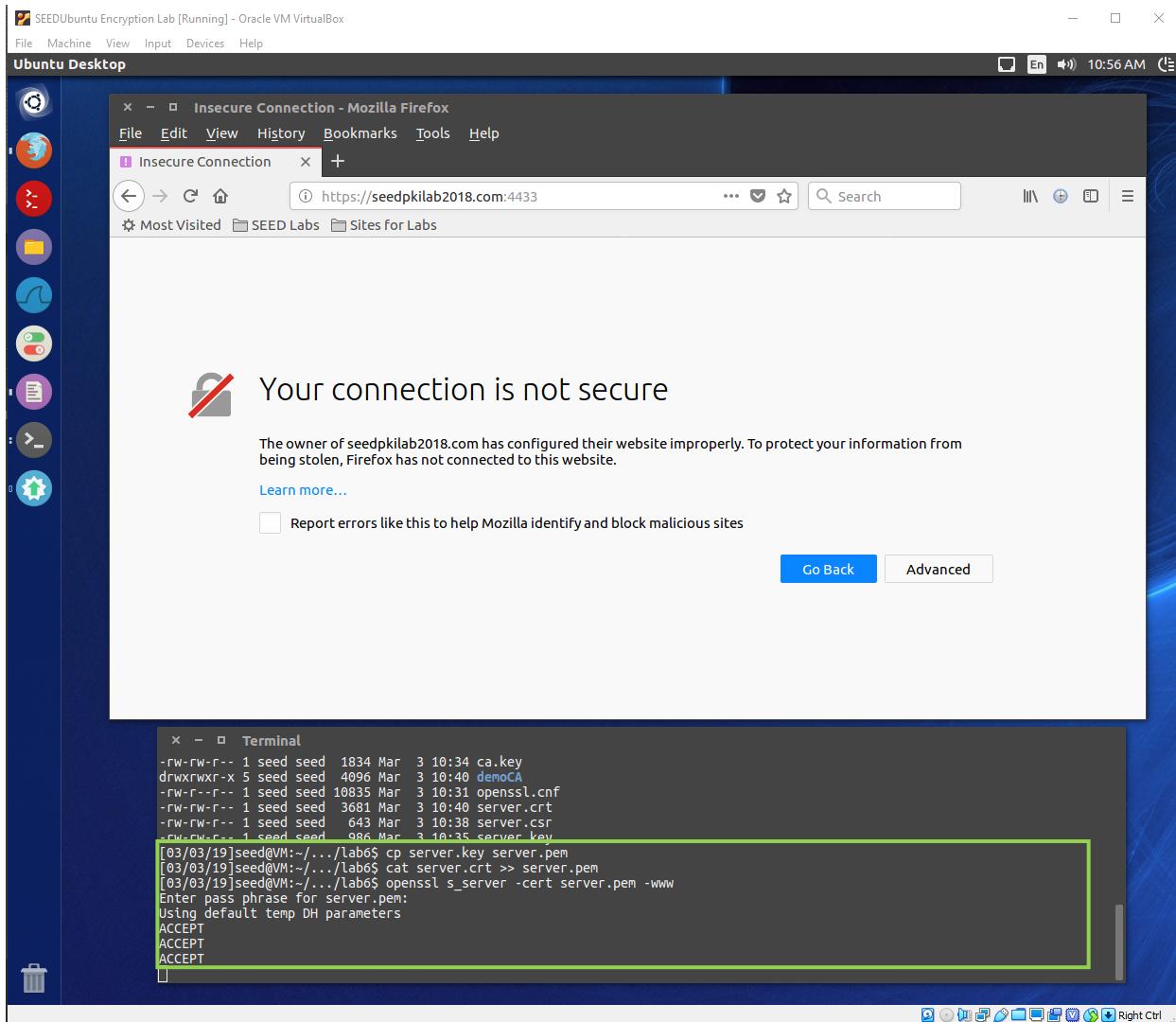
```
[03/03/19]seed@VM:~/.../lab6$ cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      VM

# 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
127.0.0.1      SEEDPKILab2018.com
[03/03/19]seed@VM:~/.../lab6$
```

Step 2: Configuring the Web Server

In the figure below, we see us combining the private key and public certificate into one file which is used by the web server to 1) send down the servers certificate and 2) use the private key to decrypt data from the client (web browser).

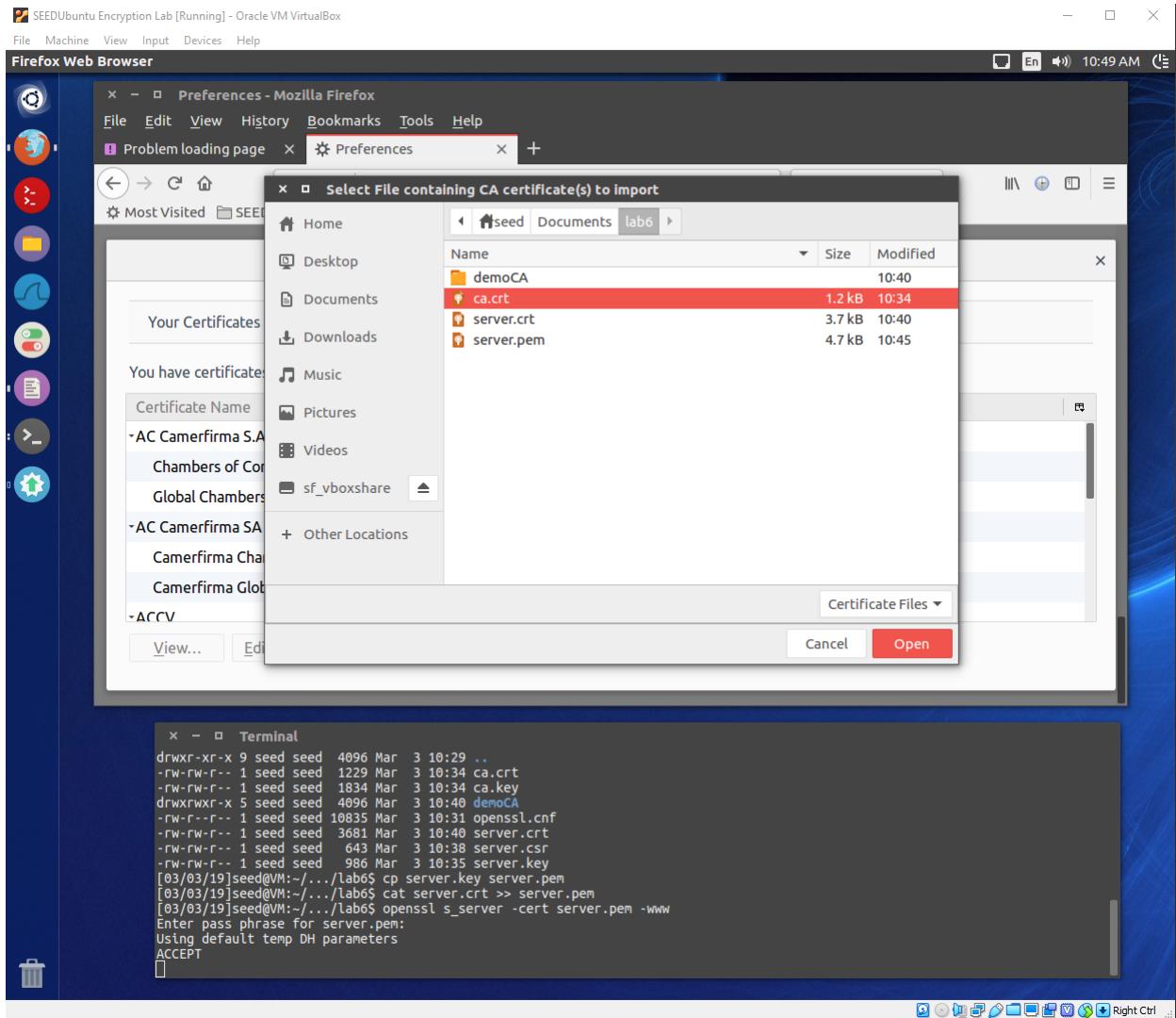
We can also see that without the Certificate Authority's certificate (ca.crt) added to the browsers certificates, we get an error when we try to access the <https://seedpkilab2018.com:4433> website.



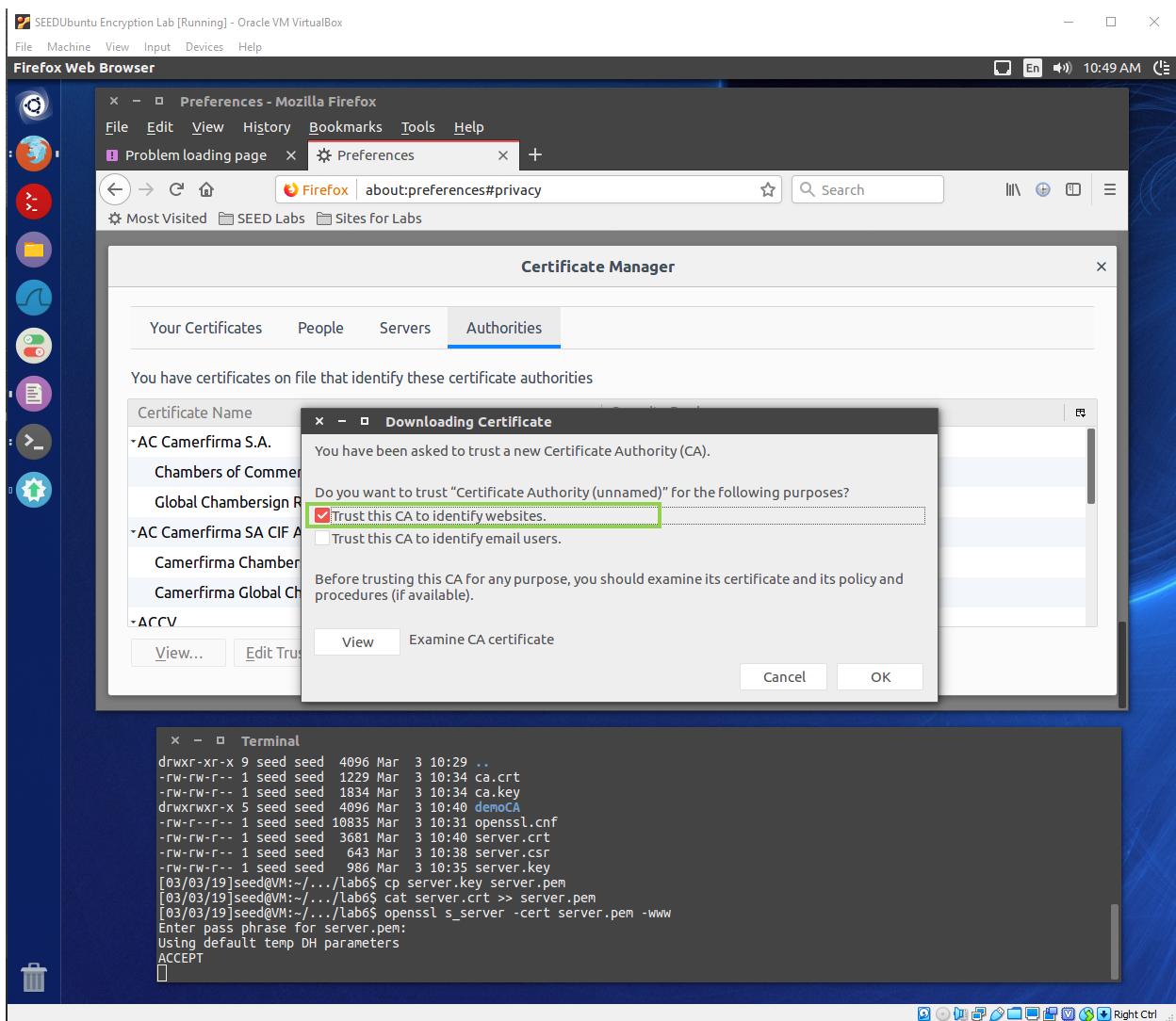
Step 3: Getting the browser to accept our CA certificate

In the next few figures, we see how the ca.crt is being added to the browser.

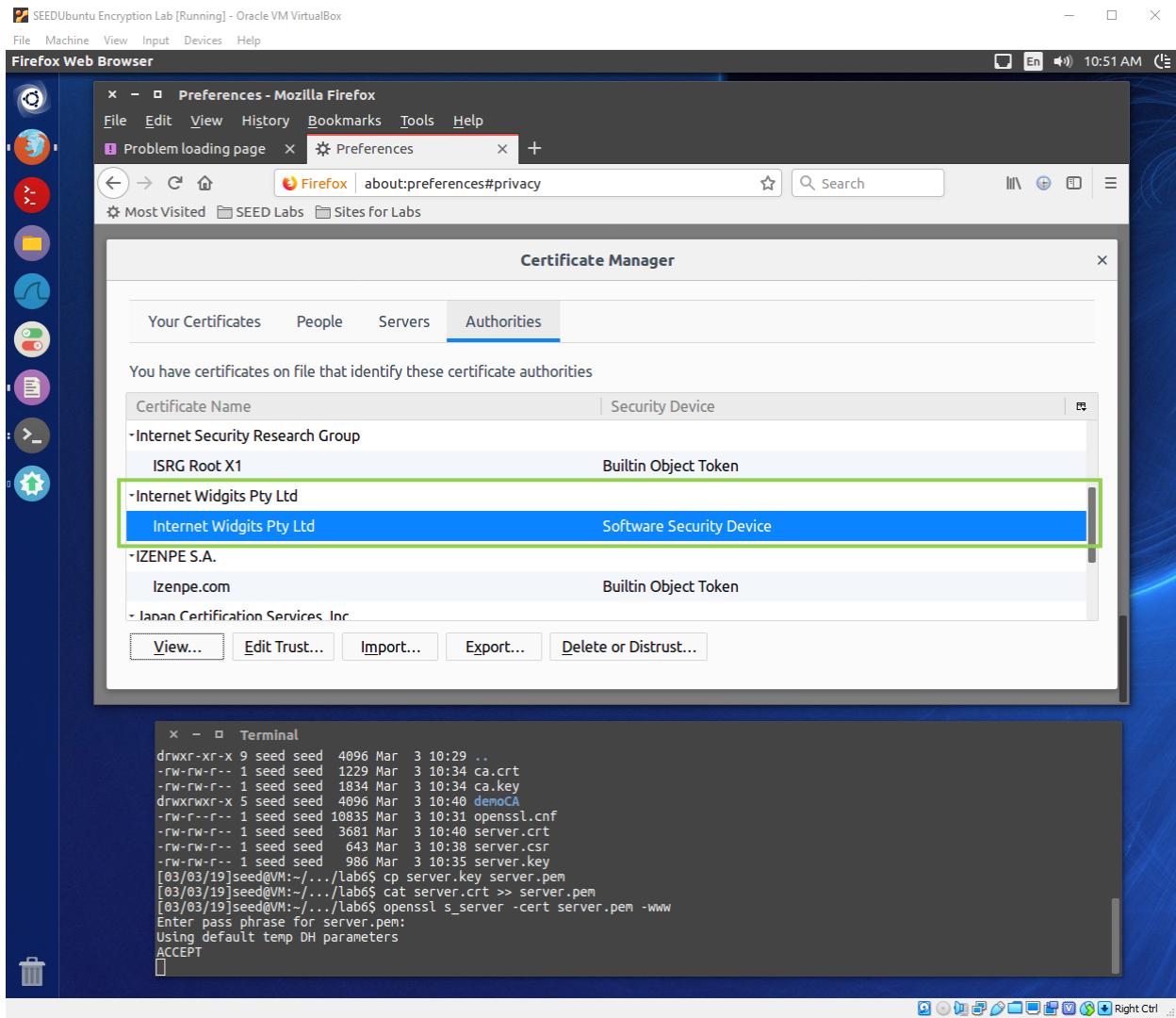
First, we select the certificate ca.crt.



We then specify that we should trust this certificate to identify websites. This means that if a website was signed by this CA, we should allow this CA certificate to verify the website's certificate.

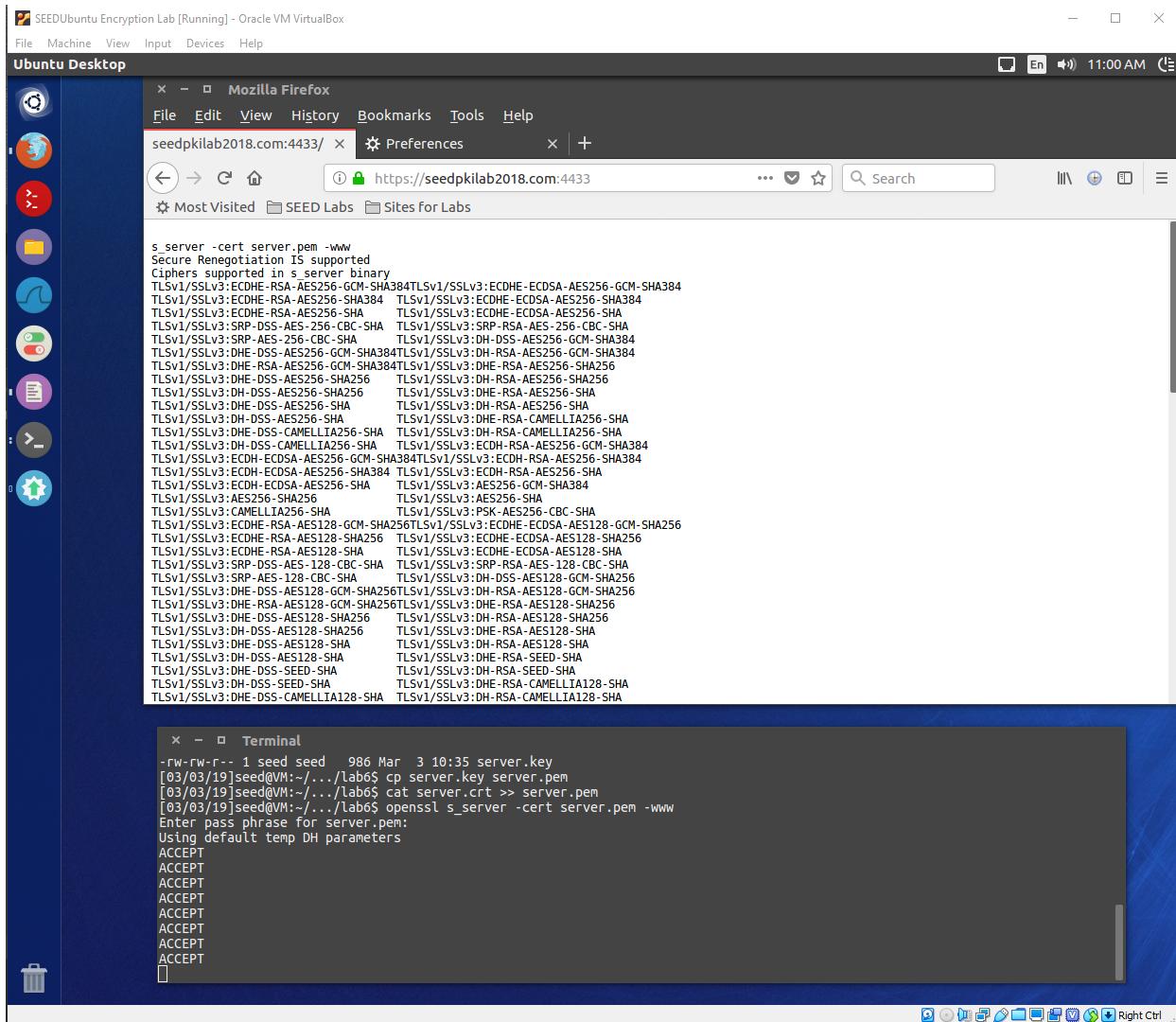


Finally, we see that our ca.crt is imported into the browser.



Step 4: Testing our HTTPS website

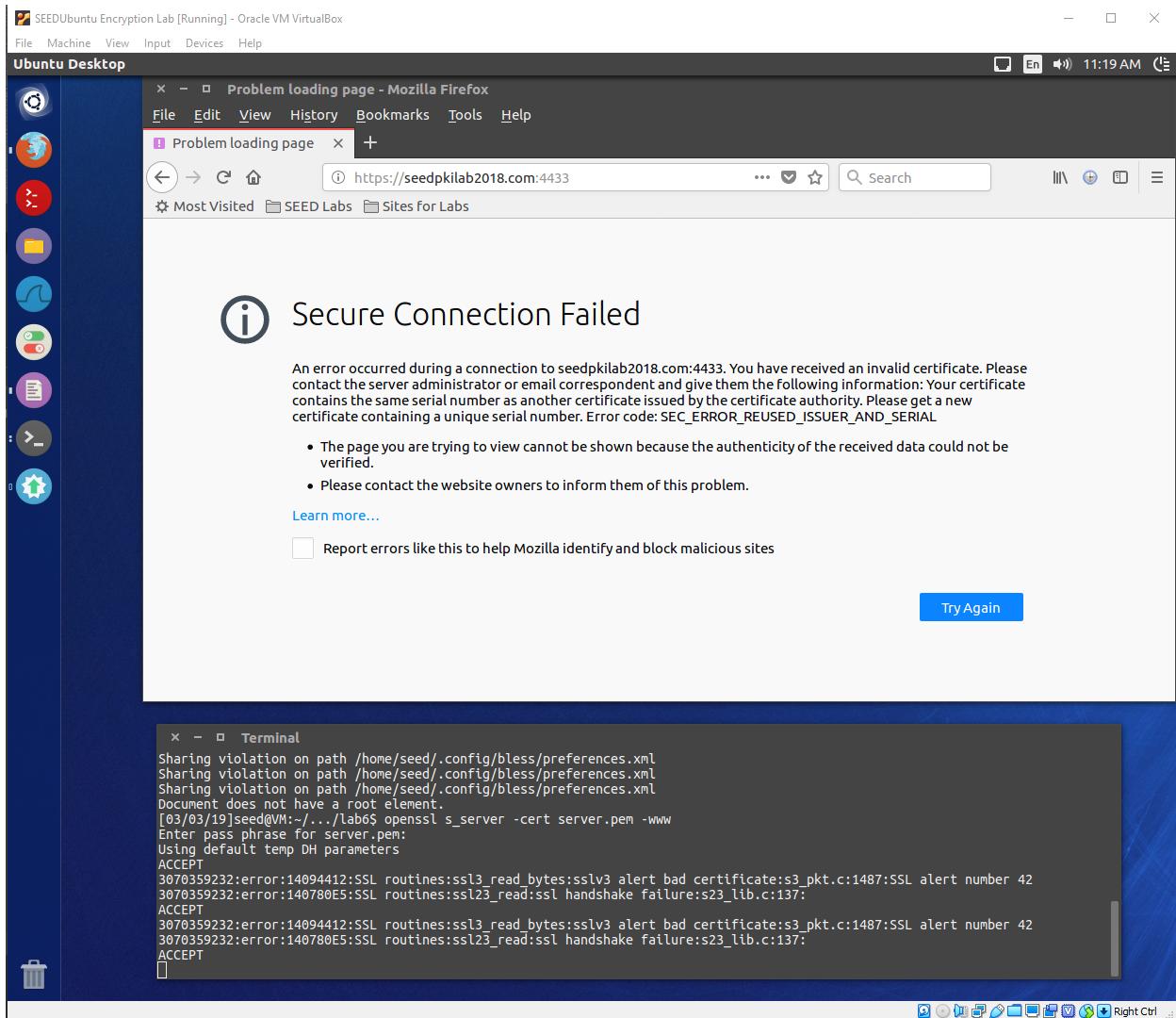
In the figure below, we can see that accessing our website through and https connection now works!



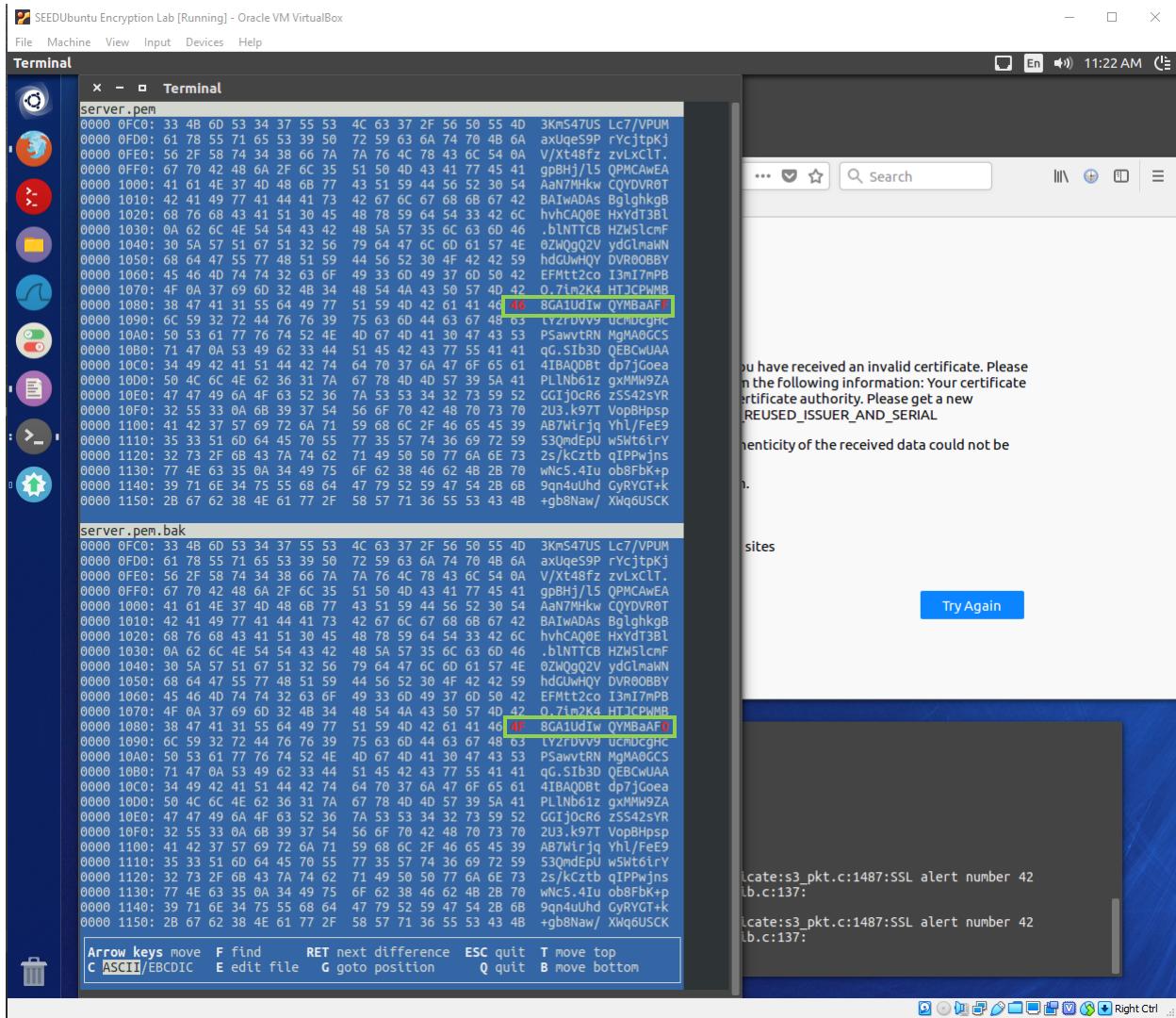
Modify a single byte of server.pem

We modify a single byte of the server.pem file and attempt access to the website.

Notice, we see the Secure Connection Failed / invalid certificate message.



In the figure below, we see the byte that was modified. I modified a single byte between the "----BEGIN CERTIFICATE----" and "----END CERTIFICATE----" section of the server.pem file. Please note, the server.pem is on top and the server.pem.bak (unmodified) is on the bottom.



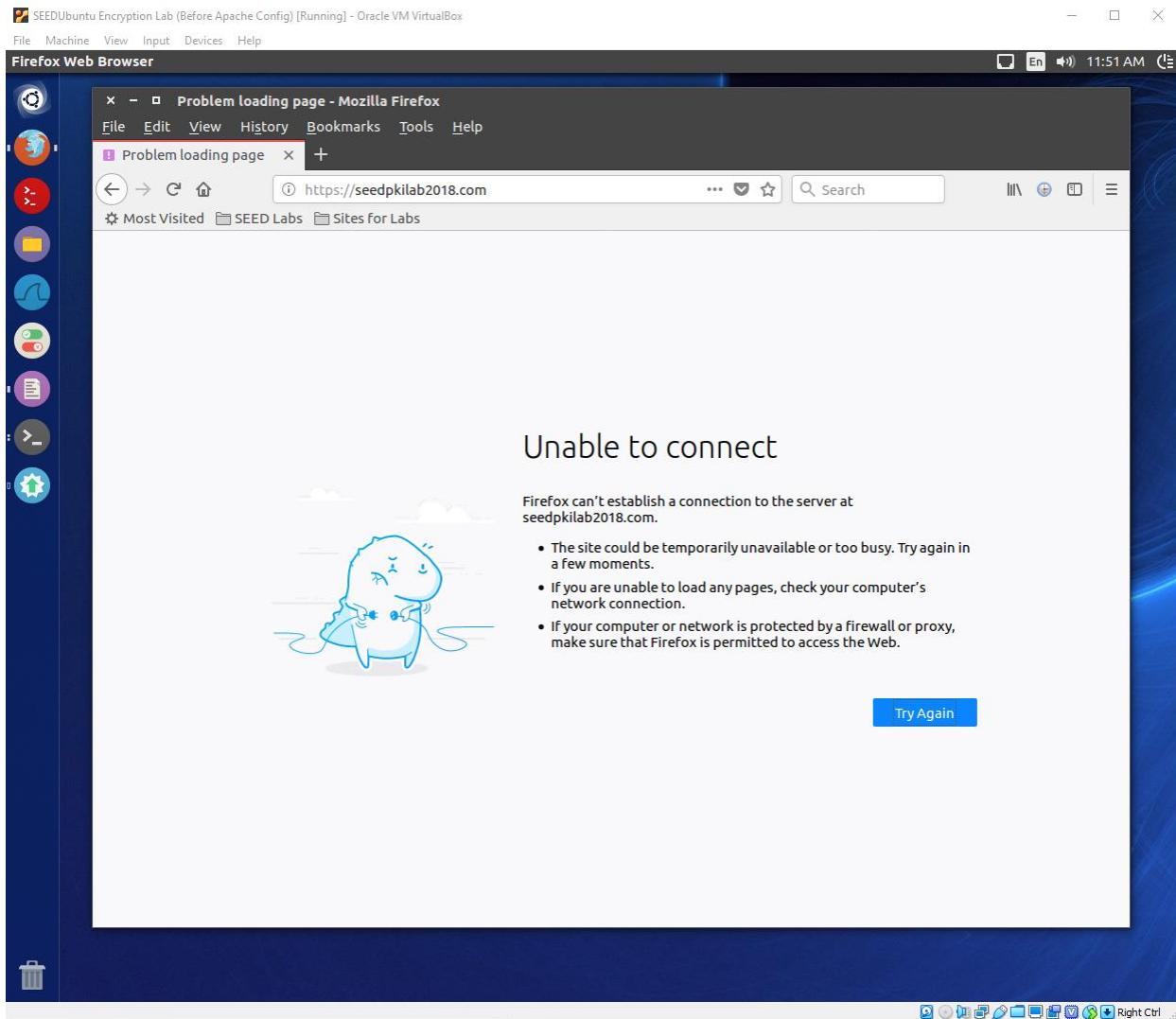
Observations / Explanations

We can see that when the web browser has the CA's certificate to verify the server's certificate, the https connection works without issue; i.e. we see the website ok.

When we modify a byte in the server's certificate signature section, we can start the website, but we get an invalid certificate error when the browser tries to use the CA certificate to verify the server's certificate. Since we modify the signature in the server's certificate, verification will fail... which is does.

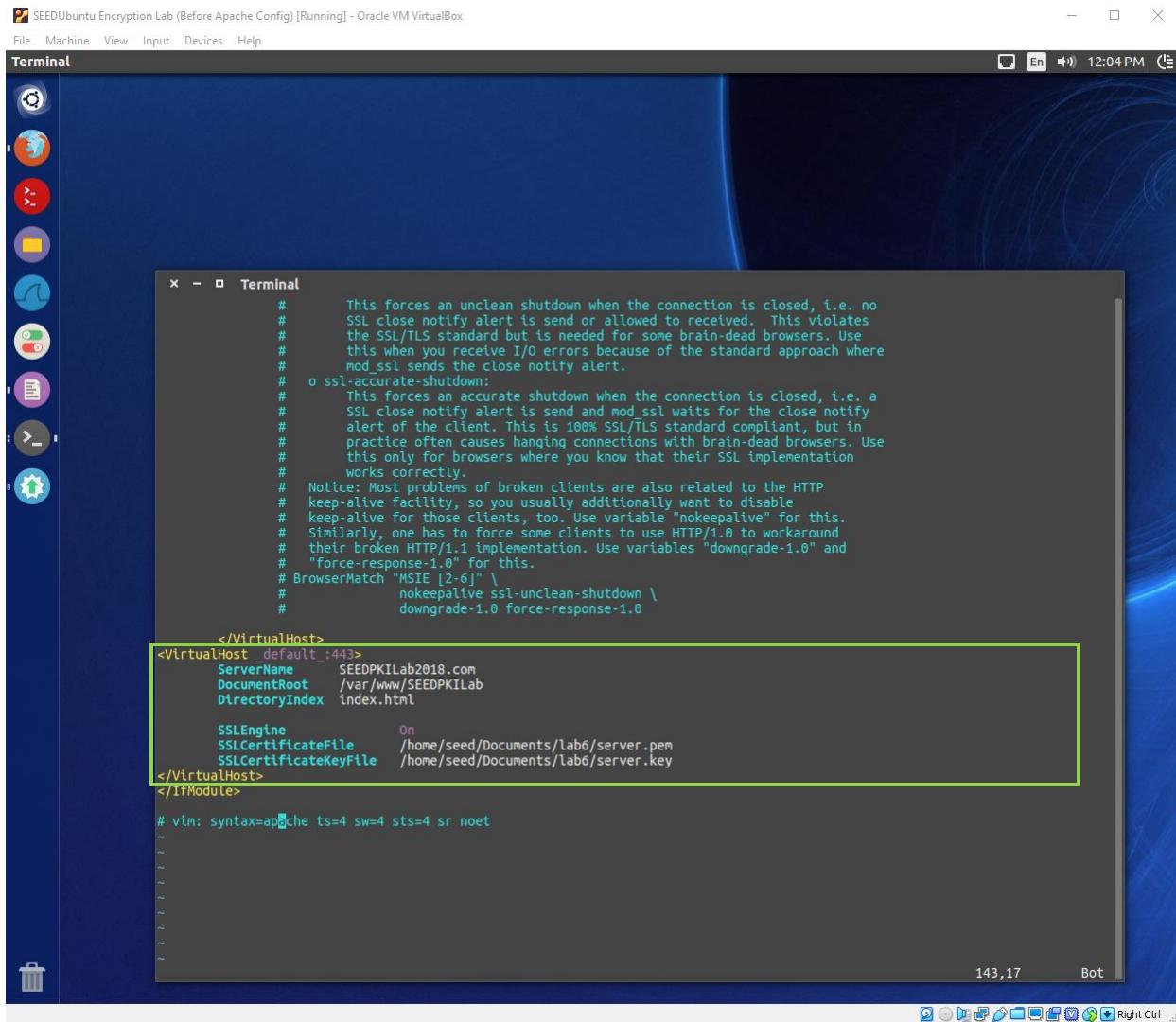
Task 4: Deploying Certificate in an Apache-Based HTTPS Website

In the figure below, when we try to access the <https://seedpkilab2018.com> website, it fails with error.



In the figure below, we see the edit of /etc/apache2/sites-available/default-ssl.conf to add this https website and associated information about the website:

- ServerName – Name of our site SEEDPKILab2018.com.
- DocumentRoot – Location of web files /var/www/SEEDPKILab.
- DirectoryIndex – HTML file that will load on website access.
- SSLEngine – SSL on for this site.
- SSLCertificateFile – Server's certificate file.
- SSLCertificateKeyFile – Server's private key file.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal". The content of the terminal shows an Apache configuration file (httpd.conf) with several commented-out sections. A specific section for a virtual host is highlighted with a green rectangle:

```
# 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 sends 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 of the client. This is 100% SSL/TLS standard compliant, 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 facility, 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 _default_:443>
  ServerName SEEDPKILab2018.com
  DocumentRoot /var/www/SEEDPKILab
  DirectoryIndex index.html

  SSLEngine On
  SSLCertificateFile /home/seed/Documents/lab6/server.pem
  SSLCertificateKeyFile /home/seed/Documents/lab6/server.key
</VirtualHost>
</IfModule>

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

In the figure below, we see the index.html file for the SEEDPKI2018.com website. This is located in /var/www/SEEKPKILab.

The only thing we do for this website is print the message "SEED PKI 2018 lab". I should've changed it to 2019, but since our site is "2018" per lab instructions, I kept it the same.

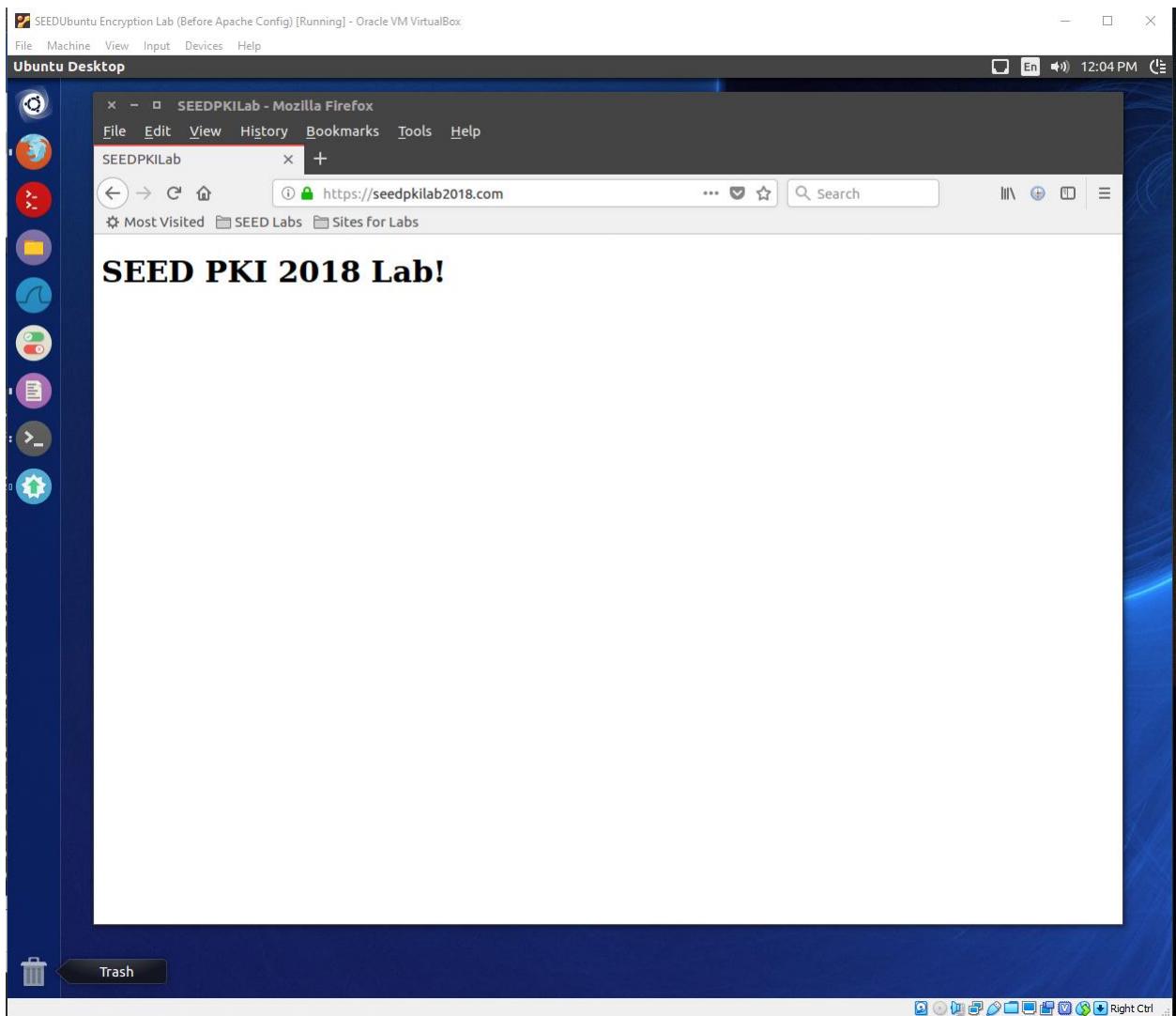
The screenshot shows a Linux desktop environment with a dark blue theme. A terminal window is open, displaying the following command-line session:

```
[03/03/19]seed@VM:.../SEEDPKILab$ pwd  
/var/www/SEEDPKILab  
[03/03/19]seed@VM:.../SEEDPKILab$ ls -al  
total 12  
drwxr-xr-x 2 root root 4096 Mar  3 11:55 .  
drwxr-xr-x 8 root root 4096 Mar  3 11:55 ..  
-rw-rw-r-- 1 root root 134 Mar  3 11:58 index.html  
[03/03/19]seed@VM:.../SEEDPKILab$ more index.html  
<!DOCTYPE html>  
<html>  
  <head>  
    <title>SEEDPKILab</title>  
  </head>  
  <body>  
    <h1>SEED PKI 2018 Lab!</h1>  
  </body>  
</html>  
[03/03/19]seed@VM:.../SEEDPKILab$
```

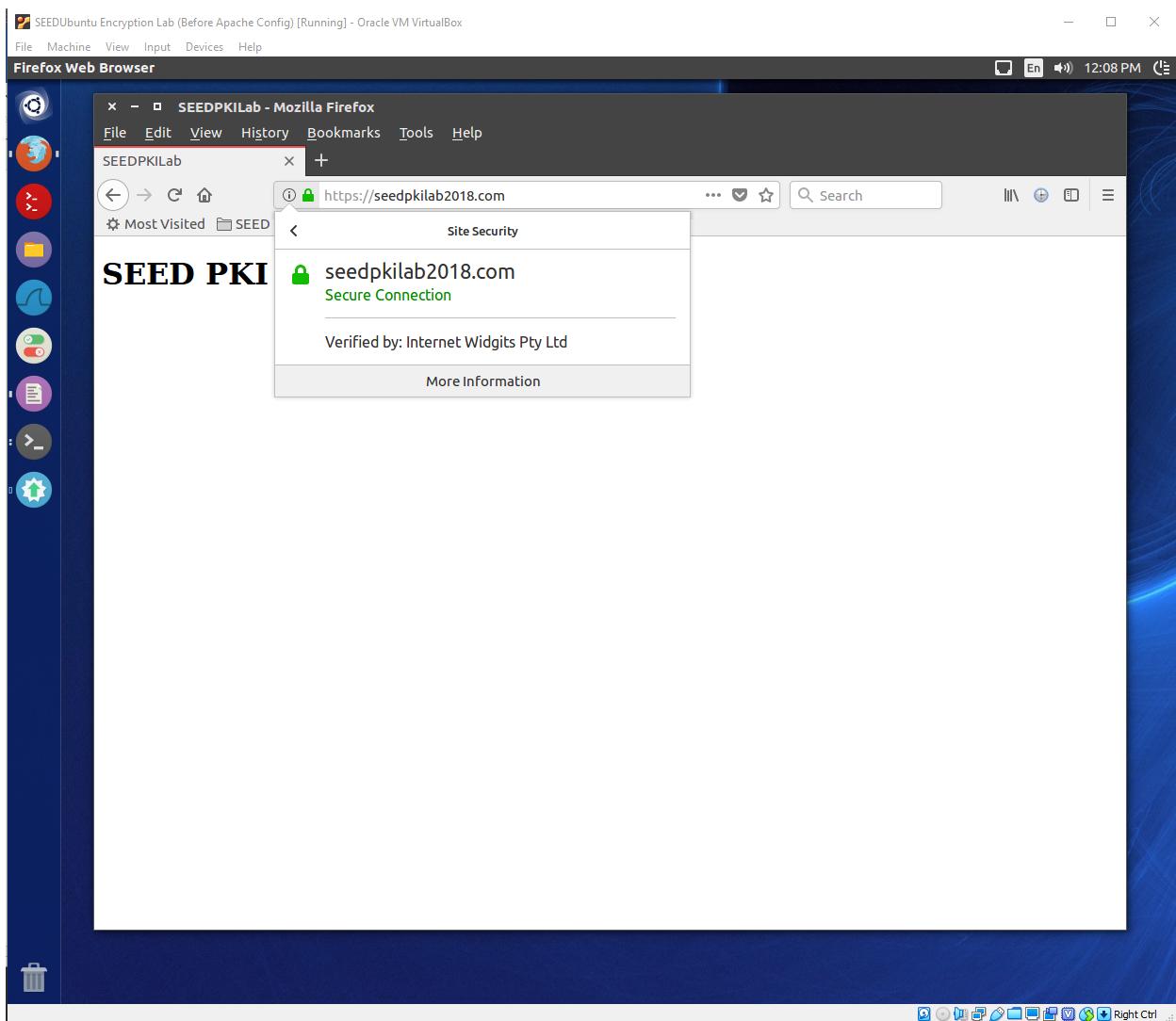
In the figure below, we see the commands to add/configure the website for SSL with the Apache web server.

```
[03/03/19]seed@VM:~/.../lab6$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
Using default temp DH parameters
ACCEPT
ACCEPT
^C
[03/03/19]seed@VM:~/.../lab6$ cd /etc/apache2/
[03/03/19]seed@VM:.../apache2$ ls
apache2.conf  conf-enabled  magic      mods-available  sites-available
conf-available  envvars     mods-available  ports.conf    sites-enabled
[03/03/19]seed@VM:.../apache2$ cd sites-available/
[03/03/19]seed@VM:.../sites-available$ ls
000-default.conf  default-ssl.conf
[03/03/19]seed@VM:.../sites-available$ ls -al
total 20
drwxr-xr-x  2 root root 4096 Apr 27 2018 .
drwxr-xr-x  8 root root 4096 Jul 25 2017 ..
-rw-r--r--  1 root root 6338 Apr  5 2016 000-default.conf
[03/03/19]seed@VM:.../sites-available$ vim default-ssl.conf
[03/03/19]seed@VM:.../sites-available$ sudo vim default-ssl.conf
[sudo] password for seed:
[03/03/19]seed@VM:.../sites-available$ 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
[03/03/19]seed@VM:.../sites-available$ 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
[03/03/19]seed@VM:.../sites-available$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  service apache2 reload
[03/03/19]seed@VM:.../sites-available$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for SEEDPKILab2018.com:443 (RSA): ****
[03/03/19]seed@VM:.../sites-available$
```

In the figure below, we see the successful display of the index.html file we created for the HTTPS SEEDPKI2018.com website!



In the figure below, when we click on the lock icon, we can see that the server's certificate is "Verified by: Internet Widgets Pty Ltd" which is the CA that we created.



Observations / Explanations

In this task, we enabled https for our web site in the Apache server. We created a simple website, added an entry in the VirtualHost section for 443 (TLS/SSL sites) for our website and configured Apache to accept the new configuration. Finally, when we accessed the website, we were able to successfully make an https connection and view the web page.

We also examined the certificate to ensure it's being verified by the correct CA – our CA certificate that we imported, which it is.