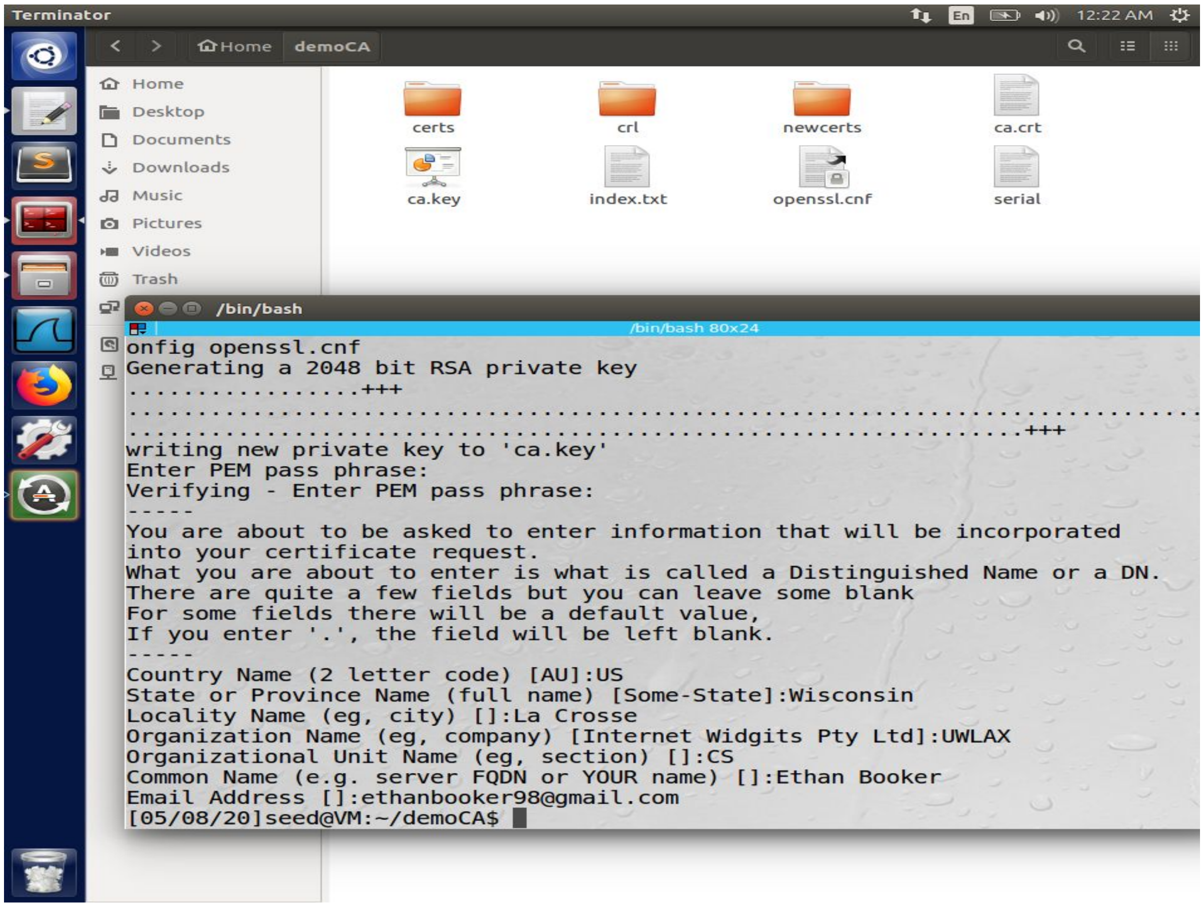
Ethan Booker

Public-Key Infrastructure (PKI) Lab

Spring 2020

Description of overall goals of lab:
To gain first-hand experience on PKI, especially on how it works, is used to protect the web, and how Man-in-the-middle attacks can be defeated by PKI.

2.1, Task 1
Description: Become a root CA
Evidence:



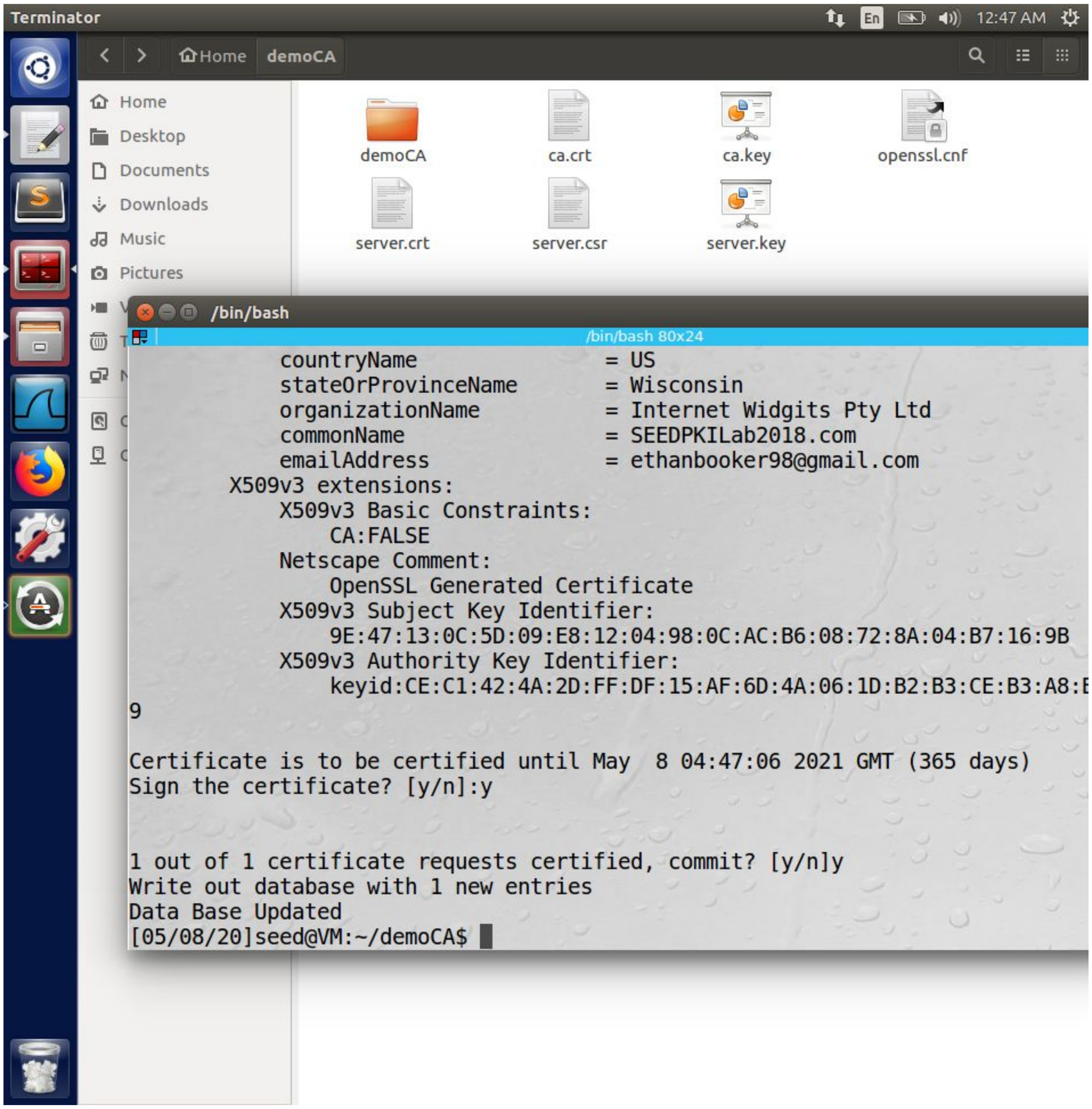
```
onfig openssl.cnf
Generating a 2048 bit RSA private key
.................+++
.....................................................................................
.............................................................+++
writing new private key to 'ca.key'
Enter PEM pass phrase:
Verifying - 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]:US
State or Province Name (full name) [Some-State]:Wisconsin
Locality Name (eg, city) []:La Crosse
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UWLAX
Organizational Unit Name (eg, section) []:CS
Common Name (e.g. server FQDN or YOUR name) []:Ethan Booker
Email Address []:ethanbooker98@gmail.com
[05/08/20]seed@VM:~/demoCA$
```

Analysis: I was able to create a self-signed by following the steps in which two files were created, the ca.crt and ca.key. The ca.key file contained an encrypted key and the ca.crt held my credentials that I entered.

2.2, Task 2

Description: Creating certificate for SEEDPKILab2018.com

Evidence:



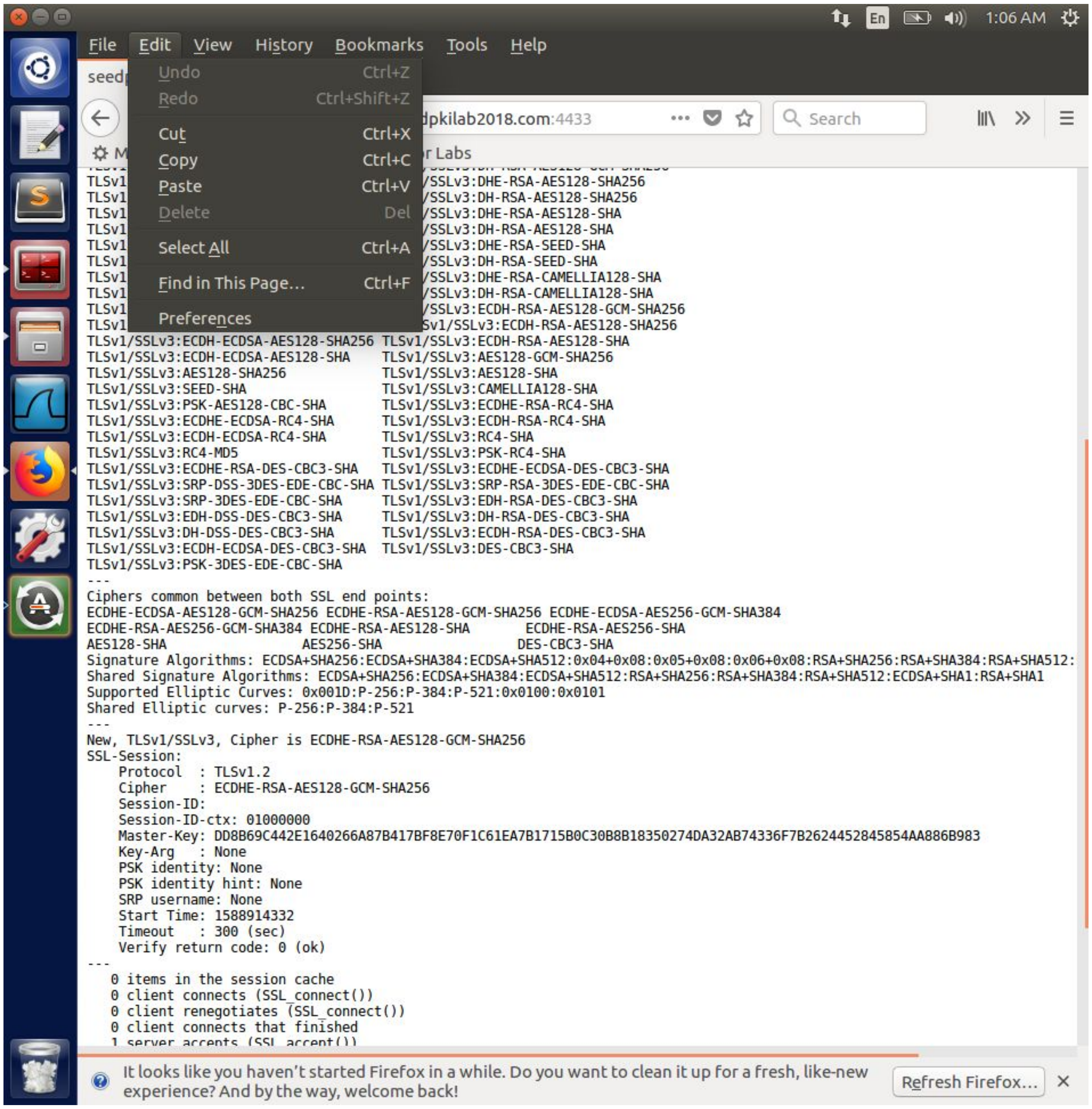
```
            countryName               = US
            stateOrProvinceName       = Wisconsin
            organizationName          = Internet Widgits Pty Ltd
            commonName                = SEEDPKILab2018.com
            emailAddress              = ethanbooker98@gmail.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                9E:47:13:0C:5D:09:E8:12:04:98:0C:AC:B6:08:72:8A:04:B7:16:9B
            X509v3 Authority Key Identifier:
                keyid:CE:C1:42:4A:2D:FF:DF:15:AF:6D:4A:06:1D:B2:B3:CE:B3:A8:E
9

Certificate is to be certified until May  8 04:47:06 2021 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
[05/08/20]seed@VM:~/demoCA$
```

Analysis: To create a certificate for the website, the website first needed to generate an RSA key pair then create an CSR which contains the website's public key. To be trusted, the CSR file needed a signature from the root CA in order to certify the website's certificate.

2.3, Task 3

Description: Deploying Certificate in an HTTPS Web Server

Evidence:



```
TLSv1                                    /SSLv3:DHE-RSA-AES128-SHA256
TLSv1                                    /SSLv3:DH-RSA-AES128-SHA256
TLSv1                                    /SSLv3:DHE-RSA-AES128-SHA
TLSv1                                    /SSLv3:DH-RSA-AES128-SHA
TLSv1                                    /SSLv3:DHE-RSA-SEED-SHA
TLSv1                                    /SSLv3:DH-RSA-SEED-SHA
TLSv1                                    /SSLv3:DHE-RSA-CAMELLIA128-SHA
TLSv1                                    /SSLv3:DH-RSA-CAMELLIA128-SHA
TLSv1                                    /SSLv3:ECDH-RSA-AES128-GCM-SHA256
TLSv1                                    Sv1/SSLv3:ECDH-RSA-AES128-SHA256
TLSv1/SSLv3:ECDH-ECDSA-AES128-SHA256 TLSv1/SSLv3:ECDH-RSA-AES128-SHA
TLSv1/SSLv3:ECDH-ECDSA-AES128-SHA    TLSv1/SSLv3:AES128-GCM-SHA256
TLSv1/SSLv3:AES128-SHA256            TLSv1/SSLv3:AES128-SHA
TLSv1/SSLv3:SEED-SHA                 TLSv1/SSLv3:CAMELLIA128-SHA
TLSv1/SSLv3:PSK-AES128-CBC-SHA       TLSv1/SSLv3:ECDHE-RSA-RC4-SHA
TLSv1/SSLv3:ECDHE-ECDSA-RC4-SHA      TLSv1/SSLv3:ECDH-RSA-RC4-SHA
TLSv1/SSLv3:ECDH-ECDSA-RC4-SHA       TLSv1/SSLv3:RC4-SHA
TLSv1/SSLv3:RC4-MD5                  TLSv1/SSLv3:PSK-RC4-SHA
TLSv1/SSLv3:ECDHE-RSA-DES-CBC3-SHA   TLSv1/SSLv3:ECDHE-ECDSA-DES-CBC3-SHA
TLSv1/SSLv3:SRP-DSS-3DES-EDE-CBC-SHA TLSv1/SSLv3:SRP-RSA-3DES-EDE-CBC-SHA
TLSv1/SSLv3:SRP-3DES-EDE-CBC-SHA     TLSv1/SSLv3:EDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:EDH-DSS-DES-CBC3-SHA     TLSv1/SSLv3:DH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:DH-DSS-DES-CBC3-SHA      TLSv1/SSLv3:ECDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:ECDH-ECDSA-DES-CBC3-SHA  TLSv1/SSLv3:DES-CBC3-SHA
TLSv1/SSLv3:PSK-3DES-EDE-CBC-SHA
---
Ciphers common between both SSL end points:
ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-ECDSA-AES256-GCM-SHA384
ECDHE-RSA-AES256-GCM-SHA384 ECDHE-RSA-AES128-SHA       ECDHE-RSA-AES256-SHA
AES128-SHA                  AES256-SHA                 DES-CBC3-SHA
Signature Algorithms: ECDSA+SHA256:ECDSA+SHA384:ECDSA+SHA512:0x04+0x08:0x05+0x08:0x06+0x08:RSA+SHA256:RSA+SHA384:RSA+SHA512:
Shared Signature Algorithms: ECDSA+SHA256:ECDSA+SHA384:ECDSA+SHA512:RSA+SHA256:RSA+SHA384:RSA+SHA512:ECDSA+SHA1:RSA+SHA1
Supported Elliptic Curves: 0x001D:P-256:P-384:P-521:0x0100:0x0101
Shared Elliptic curves: P-256:P-384:P-521
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID:
    Session-ID-ctx: 01000000
    Master-Key: DD8B69C442E1640266A87B417BF8E70F1C61EA7B1715B0C30B8B18350274DA32AB74336F7B2624452845854AA886B983
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1588914332
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
   0 items in the session cache
   0 client connects (SSL_connect())
   0 client renegotiates (SSL_connect())
   0 client connects that finished
   1 server accepts (SSL_accept())
```

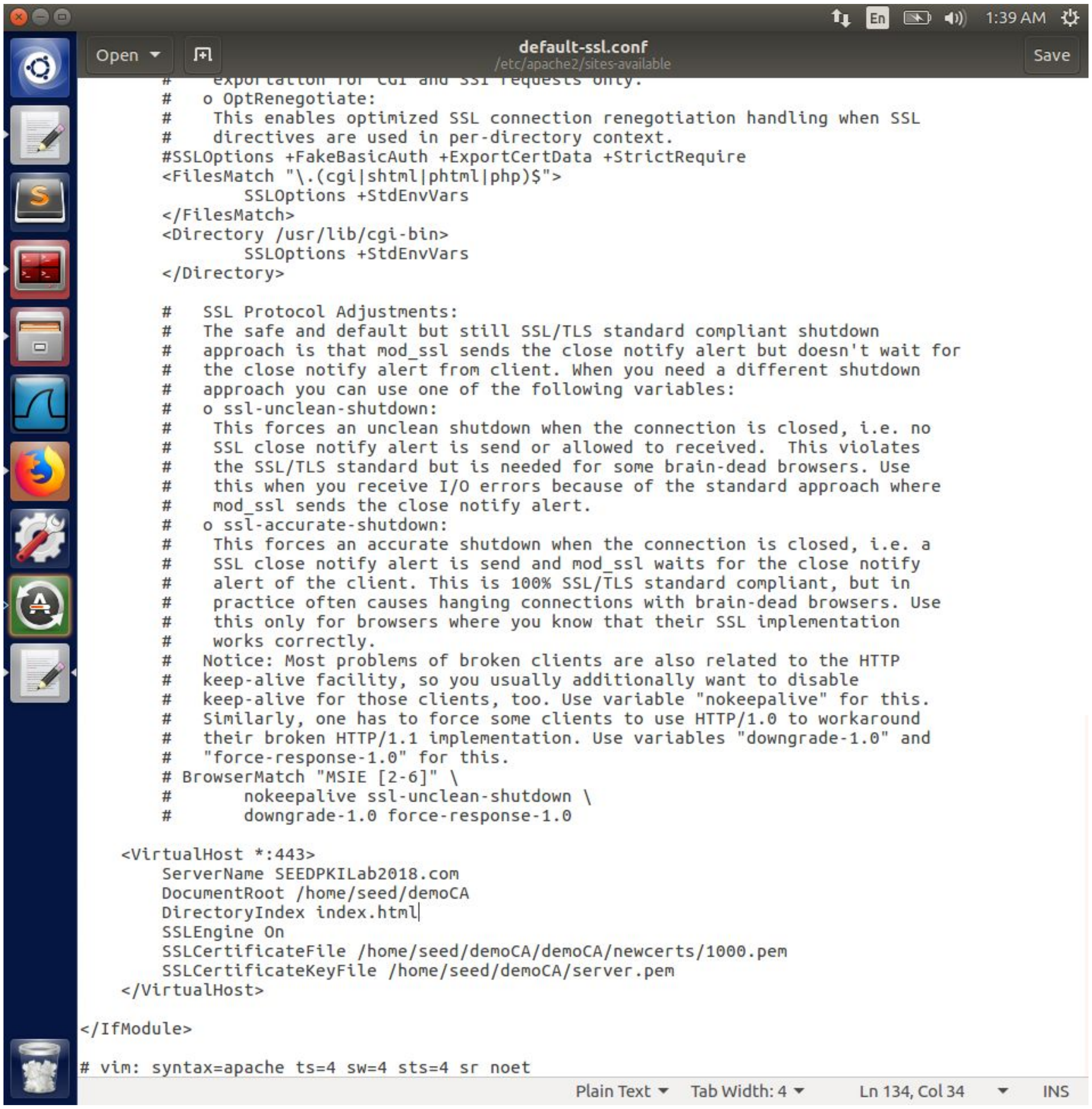
It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

Analysis: The website seemed to have posted the CA's certificate information after following the steps because I copied over the server certificate information to the .pem file which I ran. Modifying the pem file by a single bye displayed the changes however modifying the certificate caused an issue as it was no longer the certified root CA information. As for testing localhost, I for some reason was unable to access that page, so I assumed that I needed to change some sort of setting in order for it to work.

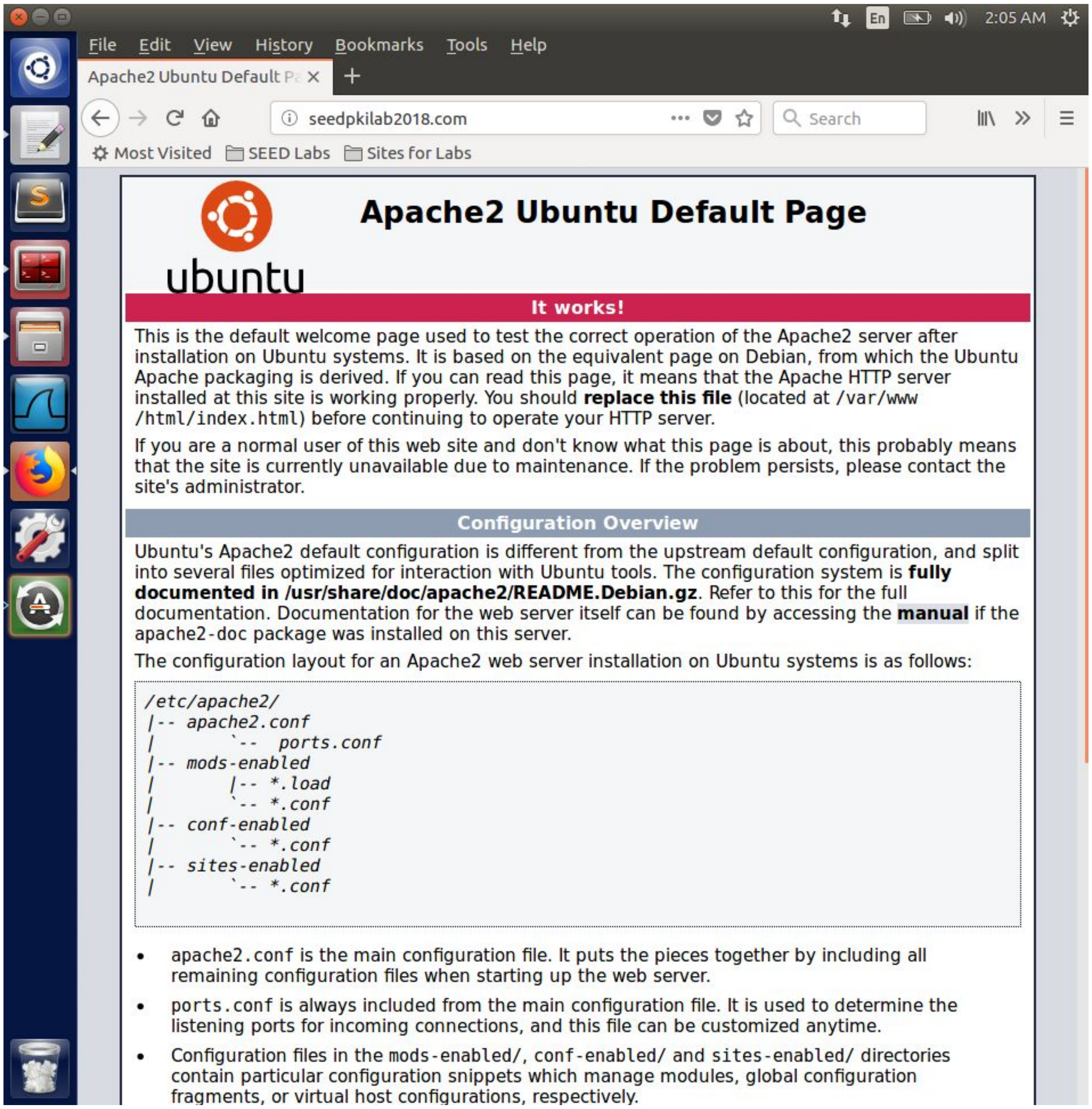
2.4, Task 4

Description: Deploying Certificate in an Apache-Based HTTPS Website

Evidence:


default-ssl.conf
/etc/apache2/sites-available

```
	#    exportation for CGI and SSI requests only.
	#   o OptRenegotiate:
	#    This enables optimized SSL connection renegotiation handling when SSL
	#    directives are used in per-directory context.
	#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
	<FilesMatch "\.(cgi|shtml|phtml|php)$">
		SSLOptions +StdEnvVars
	</FilesMatch>
	<Directory /usr/lib/cgi-bin>
		SSLOptions +StdEnvVars
	</Directory>

	#   SSL Protocol Adjustments:
	#   The safe and default 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-unclean-shutdown:
	#    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 *:443>
	ServerName SEEDPKILab2018.com
	DocumentRoot /home/seed/demoCA
	DirectoryIndex index.html
	SSLEngine On
	SSLCertificateFile /home/seed/demoCA/demoCA/newcerts/1000.pem
	SSLCertificateKeyFile /home/seed/demoCA/server.pem
    </VirtualHost>

</IfModule>

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



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

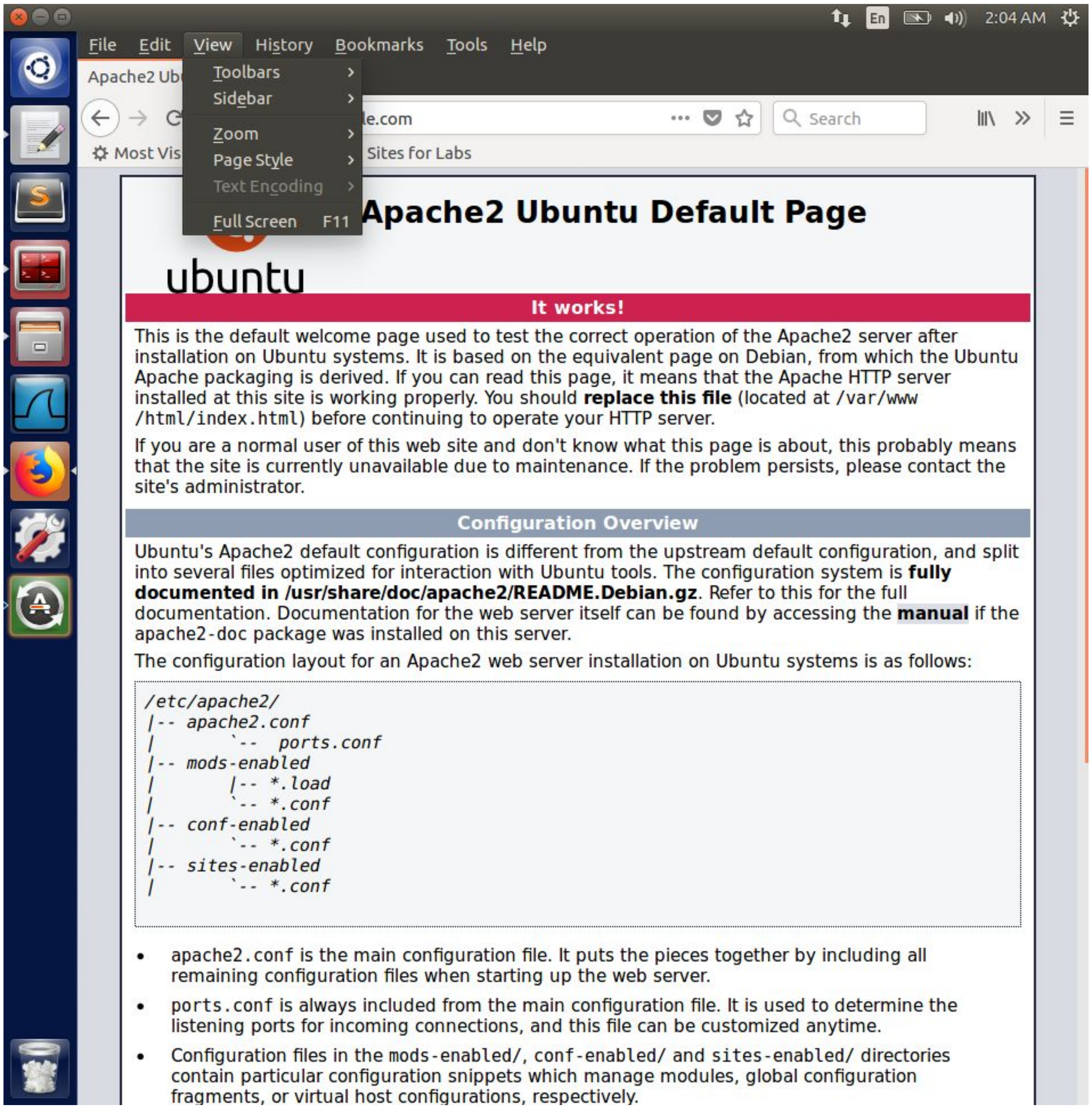
- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.

Analysis: After setting up the website configurations through the apache2 settings, I was able to access the website. The default page showed up because the index file was not configured but assumed to be correct due to the message.

2.5, Task 5

Description: Launching a Man-In-The-Middle attack

Evidence:



Apache2 Ubuntu Default Page

ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.

Analysis: I followed the same steps as task 4, but changed the ServerName to example.com. Unfortunately when I took a screenshot, the settings displayed over the url bar, but the url is short thus can assume it’s example.com. Once again, I did not change the index.html so the default page was displayed.

2.6, Task 6

Description: Launching a Man-In-The-Middle attack with a compromised CA

Evidence: Rather than showing pictures, I decided to just simply tell you what would happen as a result of being compromised. Basically in task 5 we were able to redirect users to our fake website. We could pretty much repeat these steps and replicate the website’s appearance to make it seem like they are on their targeted website but in reality they would be visiting our website.

Analysis: This is possible because the root CA is compromised and we are able to redirect users by changing their /etc/hosts file which is basically emulating a DNS cache poisoning attack. Thus we are able to reroute their target website to our server and display our fake website. The fake website can go a step further and have similar design to what they would normally see for later stages of stealing more information if needed.

Conclusion:

Throughout this lab, I basically gained a better insight on the process of PKI and each component involved in PKI. I was able to create a self CA then create a certificate for my website and deploy that website on a HTTPS web server. I also learned how to do a simple man-in-the-attack along with seeing the outcomes of a compromised CA. Personally I learned a lot about the process of becoming a CA and certifying certificate. I even learned some weakness of self certificates as you can put yourself in a vulnerability if our self-made root CA ever gets compromised and thus it’s best to pay for a commercial CA where we are not at fault.