Introduction

This document contains the Lab commands which can help students if they wish to copy and paste.

Exercise 1 – Public Key Infrastructure

Your Devices

You will be using the **PLABUBNT01** server to perform all exercises in this Lab. Please power this server on now.

Setting Up the Lab Environment

Power on your **PLABUBNT01** Linux server. In order to get the Lubuntu server, you may have to open one of the previous Practice Labs in the side panel to get a list of servers.

For this lab, you will clone a GitHub repository holding the setup scripts and instructions to the lab for easier copy and paste functionality. The setup script will automate many of the time-consuming setup tasks and free students up to focus on the Learning Objectives.

Open a terminal session on the Lubuntu server and perform the following commands. Any text that is in the brackets {} is a comment provided to help you understand what we are trying to accomplish with a command.

```
sudo git clone https://github.com/ndctdmh/pkilab
```

This command should have downloaded scripts into the **pkilab** directory under /home/administrator.
Go into that pkilab directory and change permissions and then run the **pkisetup.sh** script using sudo. Enter these commands to accomplish those tasks:

```
cd pkilab   {Change to pkilab dir}
sudo chmod 755 *   {Add execute permissions to the scripts}
sudo ./pkisetup.sh   {Run the setup script}
```

Press enter to run the script and watch for any errors. The script will first copy the lab instructions to your Lab server's desktop. Open this file and use it to make it easier to cut and paste some of the commands from the lab.

Becoming a Certificate Authority (CA)

A Certificate Authority (CA) is a trusted entity that issues digital certificates. The digital certificate certifies the ownership of a public key by the named subject of the certificate. A number of commercial CAs are treated as root CAs; VeriSign is the largest CA at the time of writing. Users who want to get digital certificates issued by the commercial CAs need to pay those CAs.

In this lab, we need to create digital certificates, but we are not going to pay any commercial CA. We will become a root CA ourselves, and then use this CA to issue certificate for others (e.g. servers). In this task, we will make ourselves a root CA, and generate a certificate for this CA. Unlike other certificates, which are usually signed by another CA, the root CA’s certificates are self-signed. Root CA’s certificates are usually pre-loaded into most operating systems, web browsers, and other software that rely on PKI. Root CA’s certificates are unconditionally trusted.

Note: Screen shots in the instructions will not always be an exact match to your activities.

Click **Next** to proceed to the first exercise.

Task 1 – Generate a Self-Signed Certificate

Step 1

As we described before, we need to generate a self-signed certificate for our CA. This means that this CA is totally trusted, and its certificate will serve as the root certificate. You can run the following command to generate the self-signed certificate for the CA:
Make sure to run these commands from /home/administrator

```
cd /home/plabadmin
```

```
openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf
```

You will be prompted for information and a password. Do not lose this password, because you will have to type the passphrase each time you want to use this CA to sign certificates for others. You will also be asked to fill in some information, such as the Country Name, Common Name, etc.

The results of the command are stored in two files: **ca.key** and **ca.crt**. The file ca.key contains the CA's private key, while ca.crt contains the public-key certificate.

Task 1 Deliverables

1.1 - **SCREENSHOT** the outputs of your new CA private Key and the Public Key Cert. Your output should be a Linux cat or more command of both the cert and key file. We only need to see the top portion of the outputs (Lines 1- 10 are fine). **Save this screenshot for your lab report.**

1.2 – **QUESTION AND OBSERVATION**: **In your lab report**, list the importance of the **ca.key** and the **ca.crt** files, be sure to include the purpose of each for a Certificate Authority.

Task 2 – Creating a Certificate for sra221.com

Now, that we are a root CA, we are ready to sign digital certificates for our customers. Our first customer is a company called **sra221.com**. For this company to get a digital certificate from a CA, it needs to go through three steps.

Step 1

Generate public/private key pair. The company needs to first create its own public/private key pair. We can run the following command to generate an RSA key pair (both private and public keys). You will also be required to provide a password to encrypt the private key (using the AES-128 encryption algorithm, as is specified in the command option). The keys will be stored in the file server.key:

```
openssl genrsa -aes128 -out server.key 1024
```

The server.key is an RSA Private KEY and encoded text file (also encrypted), so you will not be able to see the actual content, such as the modulus, private exponents, etc. To see those, you can run the following command:

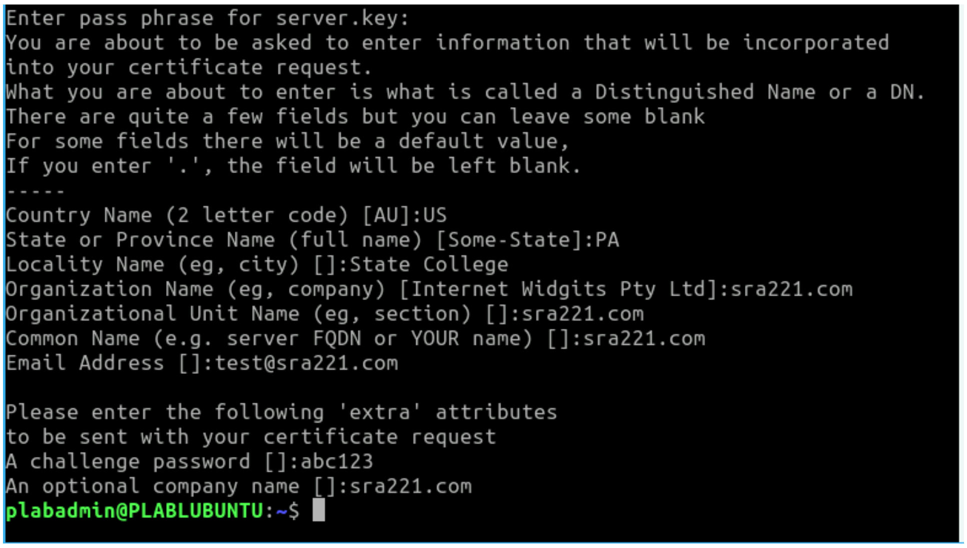
```
openssl rsa -in server.key -text | more
```

Step 2

Generate a Certificate Signing Request (CSR). Once the company has the key file, it should generate a Certificate Signing Request (CSR), which basically includes the company's public key. The CSR will be sent to the CA, who will generate a certificate for the key (usually after ensuring that identity information in the CSR matches with the server's true identity). Please use **sra221.com** as the common name of the certificate request.

```
openssl req -new -key server.key -out server.csr -config openssl.cnf
```

Keep in mind that this is the CSR for our company sra221.com so keep the names consistent when filling out the CSR, see the screen shot below:



```
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]:US
State or Province Name (full name) [Some-State]:PA
Locality Name (eg, city) []:State College
Organization Name (eg, company) [Internet Widgits Pty Ltd]:sra221.com
Organizational Unit Name (eg, section) []:sra221.com
Common Name (e.g. server FQDN or YOUR name) []:sra221.com
Email Address []:test@sra221.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:abc123
An optional company name []:sra221.com
plabadmin@PLABLUBUNTU:~$ 
```

Figure 2.1: CSR Naming Consistency

It should be noted that the above command is quite similar to the one we used in creating the self-signed certificate for the CA. The only difference is the -x509 option. Without it, the command generates a request; with it, the command generates a self-signed certificate.

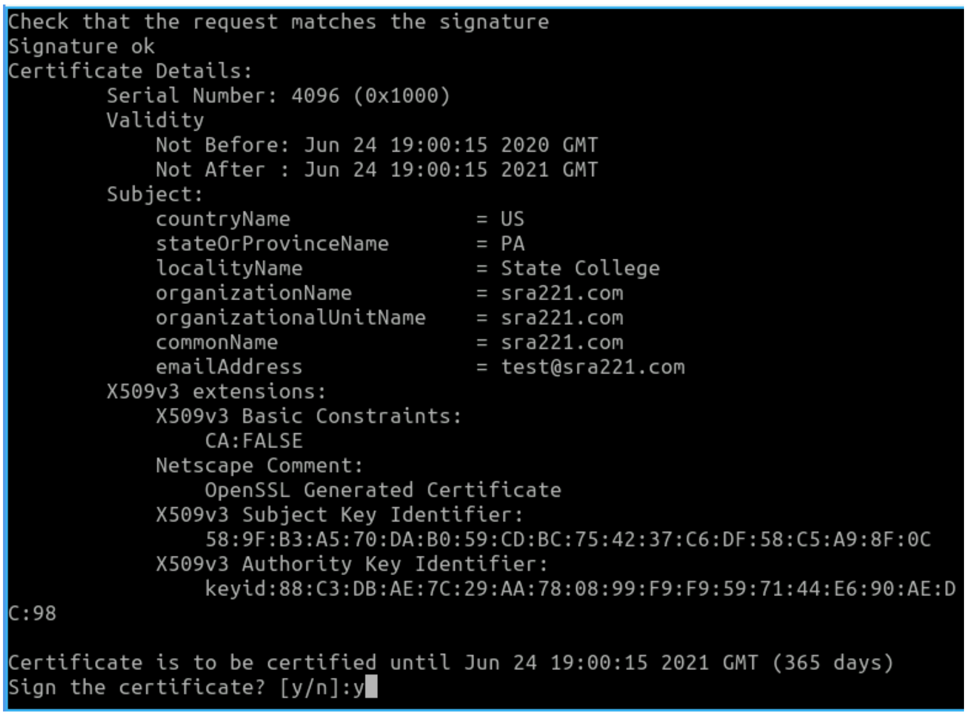
Remember to periodically reset the Auto Logout timer.

Step 3

Generating Certificates. The CSR file needs to have the CA's signature to form a certificate. In the real world, the CSR files are usually sent to a trusted CA for their signature. In this lab, we will use our own trusted CA to generate certificates. The following command turns the certificate signing request (server.csr) into an X.509 certificate (server.crt), using the CA's ca.crt and ca.key: (Put this command on **ONE LINE**)

```
openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile
ca.key -config openssl.cnf
```

The openssl command will generate the Certificate. It should prompt you to sign. Note our identifying information, like sra221.com in the CSR.



```
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Jun 24 19:00:15 2020 GMT
            Not After : Jun 24 19:00:15 2021 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = PA
            localityName              = State College
            organizationName          = sra221.com
            organizationalUnitName    = sra221.com
            commonName                = sra221.com
            emailAddress              = test@sra221.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                58:9F:B3:A5:70:DA:B0:59:CD:BC:75:42:37:C6:DF:58:C5:A9:8F:0C
            X509v3 Authority Key Identifier:
                keyid:88:C3:DB:AE:7C:29:AA:78:08:99:F9:F9:59:71:44:E6:90:AE:D
C:98

Certificate is to be certified until Jun 24 19:00:15 2021 GMT (365 days)
Sign the certificate? [y/n]:y
```

Figure 2.2: Generating a Certificate

Step 4

View the Certificate.

You can view the new certificate with the following command. Run from your HOME Directory.

```
openssl x509 -in server.crt -text | more
```

{**Note**: the above command may fail, backspace over –text and retype it}

If it continues to fail, then it is possible your key never was generated. Did you see the Y/N prompt to sign the certificate like the screen shot above? If not redo the command.

Task 2 Deliverables

2.1 - **SCREENSHOT** the first lines of your certificate that show the following and save it for your **lab report**:

- **Signature Algorithm**
- **Issuer**
- **Subject**

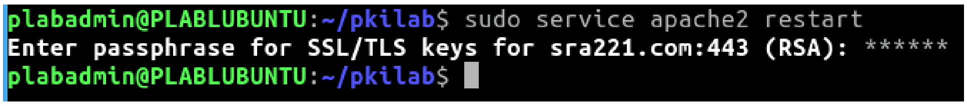
Task 3 – Deploying Certificate in an Apache Web Server

Step 1

Now we need to run a series of commands to enable SSL. Apache will ask us to type the password used for encrypting the private key. Once everything is set up properly, we can browse the web site, and all the traffic between the browser and the server will be encrypted.

Run these commands; If you receive an error about the **clickjacking** site just keep going, this is normal.

```
sudo a2enmod ssl                {NOTE:Enable the SSL module}
sudo a2ensite plabadmin-ssl     {NOTE:Enable the site}
sudo service apache2 restart    {NOTE:Restart Apache}
```



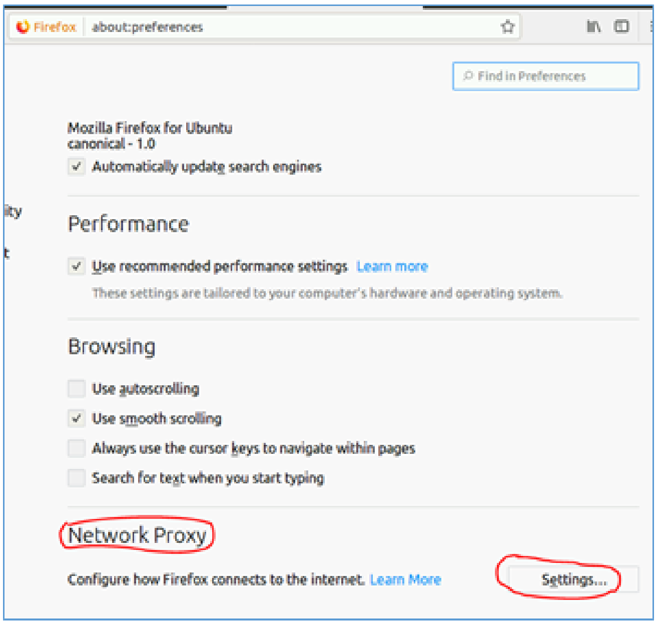
```
plabadmin@PLABLUBUNTU:~/pkilab$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for sra221.com:443 (RSA): ******
plabadmin@PLABLUBUNTU:~/pkilab$ 
```

Figure 3.1: Restarting the Apache service after enabling the site

Step 2

Change your Browsers Proxy to allow sra221.com

Open preferences in the Firefox browser and find Network Proxy Settings.



Firefox about:preferences

Find in Preferences

Mozilla Firefox for Ubuntu
canonical - 1.0

☑ Automatically update search engines

Performance

☑ Use recommended performance settings Learn more

These settings are tailored to your computer's hardware and operating system.

Browsing

□ Use autoscrolling

☑ Use smooth scrolling

□ Always use the cursor keys to navigate within pages

□ Search for text when you start typing

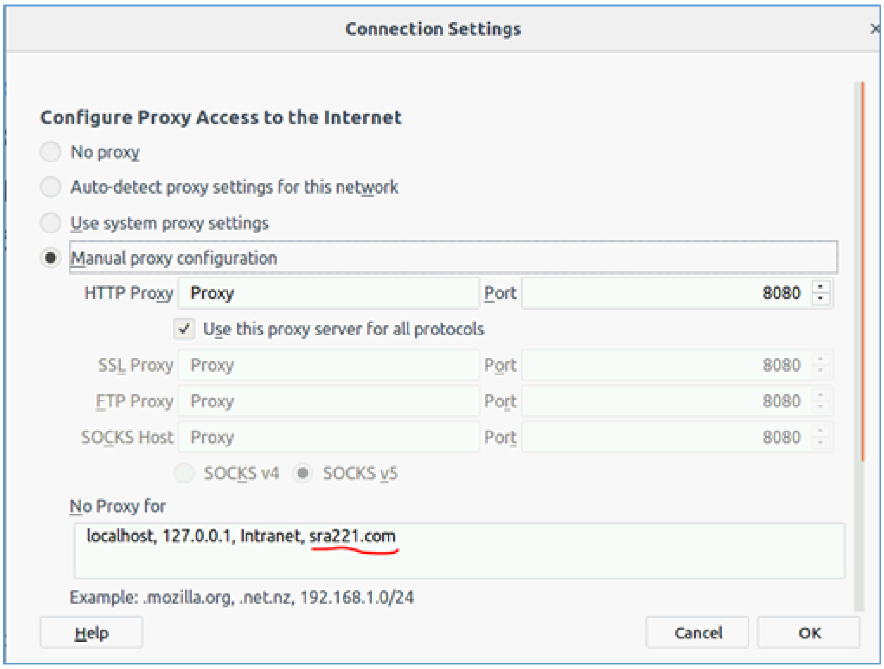
Network Proxy

Configure how Firefox connects to the internet. Learn More

Settings...

Figure 3.2: Network Proxy Settings in Firefox Preferences

Add your site "sra221.com" to the No Proxy list and **save**. We need to do this because the lab system is filtering through a proxy server and our site is not on the approved list.



Connection Settings

Configure Proxy Access to the Internet

○ No proxy

○ Auto-detect proxy settings for this network

○ Use system proxy settings

● Manual proxy configuration

HTTP Proxy: Proxy Port: 8080

☑ Use this proxy server for all protocols

SSL Proxy: Proxy Port: 8080

FTP Proxy: Proxy Port: 8080

SOCKS Host: Proxy Port: 8080

○ SOCKS v4 ● SOCKS v5

No Proxy for

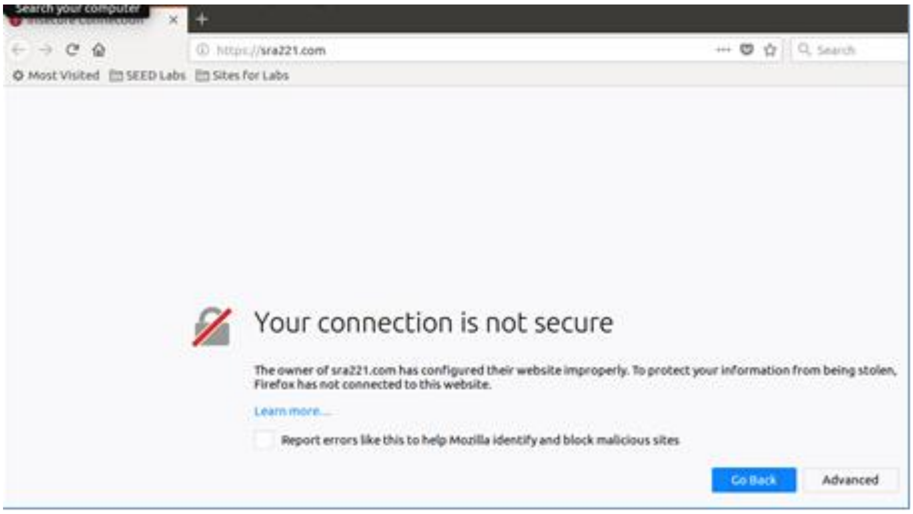
localhost, 127.0.0.1, Intranet, sra221.com

Example: .mozilla.org, .net.nz, 192.168.1.0/24

Figure 3.3: Site added to No Proxy list.

Step 3

Now, visit the site (https://sra221.com) from Firefox and observe.



Your connection is not secure

The owner of sra221.com has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

Learn more...

□ Report errors like this to help Mozilla identify and block malicious sites

Figure 3.4: Connection Not Secure

Challenge and Analysis

Perform some analysis of how browsers use certificates. **Fix the problem** so that sra221.com is trusted and displays a green is good lock.

Challenge and Analysis Deliverables

3.1 For your **lab report**, take a **SCREENSHOT** showing that the sra221 site is trusted, this should include the Green Lock and your Welcome Message.

3.2 For your **lab report**, take a **SCREENSHOT** after clicking on the Green Lock showing the name of the **Verified by: CA Name**. If you are using a snipping tool you might need to move the cursor off the screen so it does not move your output.

3.3 **Lab Report QUESTION: Explain what you had to do to enable the browser to trust the entity SRA221.com?** Your response should include the role of the CA and what has been done with the certificate to enable trust by the browser.

Keep all devices that you have powered on in their current state and proceed to the review section.

Review

Well done, you have completed the **PKI** Practice Lab.

Summary atom item:

```
aaaaaaaa-1111-1111-1111-193f35a24fe3
```

Summary

You have completed the following exercises:

- Task 1 – Generated a Self-Signed Certificate
- Task 2 – Created a Certificate for sra221.com
- Task 3 – Deployed a Certificate in an Apache Website
- Challenge – Fixed the Insecure Website

You should now have a better understanding of the following:

- Public-key encryption
- Public-Key Infrastructure (PKI)
- Certificate Authority (CA) and root CA
- X.509 certificate and self-signed certificate
- Apache, HTTP, and HTTPS
- Man-in-the-middle attacks

Shutdown all virtual machines used in this lab. Alternatively, you can log out of the lab platform.