Practical Networking

Practical TLS – Lab Guide

A deep dive into SSL and TLS – the protocols that secure the Internet

Lab 3.3 – Creating a CA and issuing Certificates

In this lab you will create your own Certificate Authority. Then you'll act as two difference servers and generate Certificate Signing Requests (CSRs), which your new CA will use to create Server Certificates.

This lab will involve using many instances of the **`openssl`** command. Pay particular attention to what files are provided as **input** to this command, and what is produced as **output**. Note that this lab was written for OpenSSL version 1.1.1 -- different versions may behave slightly differently.

Create working directory and acquire lab files

1. Navigate to the folder you created in Lab 0.0 named Practical-TLS
   - **`cd Practical-TLS`**

2. Create a new directory for Lab 3.2 files:
   - **`mkdir "LAB3.3 – Creating a CA and Issuing Certificates"`**

3. Download one of the following lab files and place it in the newly created folder:
   - **`LAB 3.3 Files - Creating a CA and Issuing Certificates - Practical TLS.tar.gz`**
   - **`LAB 3.3 Files - Creating a CA and Issuing Certificates - Practical TLS.zip`**

   The lab files are in `.zip` or `.tar.gz` format – the contents within are identical, use whichever file is easier for you

4. Unzip or Untar the file you downloaded into your LAB 3.3 working directory:
   - For the TAR.GZ file, you will use the Linux command: `tar -xvzf "LAB 3.3 ...`
   - For the ZIP file, it should be something like right clicking the file and selecting "Extract All..."

5. The following **two directories** and **five files** should exist in the LAB 3.3 directory::

```
.../LAB3.3 – Creating a CA and Issuing Certificates/RootCA/
.../LAB3.3 – Creating a CA and Issuing Certificates/RootCA/CA/
.../LAB3.3 – Creating a CA and Issuing Certificates/Pracnet_CSR-config.conf
.../LAB3.3 – Creating a CA and Issuing Certificates/RootCA/PracNet_CA-config.conf
.../LAB3.3 – Creating a CA and Issuing Certificates/RootCA/CA/serial
.../LAB3.3 – Creating a CA and Issuing Certificates/RootCA/CA/index.txt
.../LAB3.3 – Creating a CA and Issuing Certificates/RootCA/CA/index.txt.attr
```

Generate Root Certificate and Key

In this task you will start by generating an RSA Private Key file (note: generating a Private Key file generates both the Public and Private Keys).

Then, once the Public and Private keys have been generated them, you will use them to create your Root CA's Self-Signed certificate.

1. Change directories into the RootCA directory
   - `cd RootCA/`
2. Generate a 4096 Bit RSA Private Key for your Certificate Authority
   - `openssl genrsa -out ca.key 4096`
3. Examine the contents of the newly created Private Key file
   - `openssl rsa -in ca.key -noout -text`
4. Generate your Root CA's Self-Signed Certificate
   - `openssl req -new -x509 -days 3650 -sha256 -key ca.key -out ca.cert`
   - Enter the requested information:

     ```
     Country Name          = < Two letter Country Code, example: US GB RU IN >
     State or Province     = < The state you live in, example: Washington, New York >
     Locality Name         = < The city you live in, example: Seattle, Denver, London >
     Organization Name     = PracNet
     Organizational Unit = Practical TLS
     Common Name           = Root CA
     Email Address         = < Leave blank and Press Enter >
     ```

     - *What Key was used to sign this certificate?*
5. Examine the contents of the newly created Certificate file:
   - `openssl x509 -in ca.cert -noout -text`
6. Examine various fields within the newly created Certificate and Key File
   - `openssl rsa -in ca.key -noout -modulus`
   - `openssl x509 -in ca.cert -noout -modulus`
   - `openssl x509 -in ca.cert -noout -subject -issuer`
   - `openssl x509 -in ca.cert -noout -dates`
7. Return to the base LAB 3.3 directory:
   - `cd ..`

Verify CA files

At this point you should be in the base directory for LAB 3.3 and you should have **two directories** and **seven files** (total). Each file serves a specific purpose which is described below.

Inside the **base LAB 3.3 directory**, there should be **one file** and **one directory**:

```
$  ls -gh
total 4.0K
-rwxrwxrwx 1 user 1.2K Oct  4 16:16 PracNet_CSR-config.conf
drwxrwxrwx 1 user  512 Oct  4 19:04 RootCA
```

- **PracNet_CSR-config.conf** is a configuration files for **C**ertificate **S**igning **R**equests
- **RootCA** is a directory

Inside the **RootCA** directory, there should be **three files** and **one directory**:

```
$  ls -gh RootCA/
total 12K
drwxrwxrwx 1 user  512 Oct  4 19:00 CA
-rwxrwxrwx 1 user 1.3K Oct  4 17:21 PracNet_CA-config.conf
-rwxrwxrwx 1 user 2.1K Oct  4 19:04 ca.cert
-rwxrwxrwx 1 user 3.2K Oct  4 19:03 ca.key
```

- **CA** is a directory
- **PracNet_CA-config.conf** is a configuration file with **C**ertificate **A**uthority options
- **ca.cert** is the Certificate Authority certificate file which you created in the task prior
- **ca.key** is the Certificate Authority Private Key file which you created in the task prior

Inside the **CA** directory (which is inside the **RootCA** directory), there should be **three files**:

```
$  ls -gh RootCA/CA/
total 0
-rwxrwxrwx 1 user  0 Oct  4 18:59 index.txt
-rwxrwxrwx 1 user  0 Oct  4 18:59 index.txt.attr
-rwxrwxrwx 1 user 11 Oct  4 19:00 serial
```

- **index.txt** will contain a list of all issued certificates (at the moment this file is empty)
- **index.txt.attr** will control certain CA attributes (at the moment this file is empty)
- **serial** is the starting serial number in hexadecimal for all issued certificates

VERIFY YOU HAVE IDENTICAL FILES BEFORE CONTINUING

Generate a Private Key and CSR for Frodo.com

For this task you will pretend that you own the domain Frodo.com and you need to acquire a SSL certificate. You will generate a Public and Private Key. Then you will generate a CSR to request a certificate from the Root CA.

1. Generate a 2048 bit Public and Private Key for Frodo.com:
   - `openssl genrsa -out frodo.com.key 2048`
2. Generate a Certificate Signing Request (CSR):
   - `openssl req -new -sha256 -key frodo.com.key -out frodo.com.csr`
   - Enter the requested information:

```
Country Name              = < Two letter Country Code, example: US GB RU IN >
State or Province         = < The state you live in, ex: Washington, New York >
Locality Name             = < The city you live in, ex: Seattle, Denver, London >
Organization Name         = PracNet
Organizational Unit       = Practical TLS
Common Name               = frodo.com
Email Address             = < Leave blank and Press Enter >
Challenge Password        = < Leave blank and Press Enter >
Optional Company Name     = < Leave blank and Press Enter >
```

     - *What Key was used to sign this CSR?*
3. Examine the content of the CSR you just created:
   - `openssl req -in frodo.com.csr -noout -text`

The newly created CSR (`frodo.com.csr`) will be given to the CA in order to acquire a Certificate.

Use the RootCA to create a signed certificate for Frodo.com

To create a CA signed certificate, you will need the **CA configuration file**, the **CA's certificate**, the **CA's private key file**, and a **CSR**. At this point, we have all of the required files.

1. Create a CA signed certificate for Frodo.com
   - `openssl ca -config RootCA/PracNet_CA-config.conf -cert RootCA/ca.cert -keyfile RootCA/ca.key -in frodo.com.csr -out frodo.com.cert`
     *(NOTE: this entire command should be typed on **one line**)*
     - *You will be asked to confirm twice -- say **Y**es by typing "y" and pressing enter.*
2. Examine the contents of the Certificate you just created
   - `openssl x509 -in frodo.com.cert -noout -text`
     - What is the Subject
     - What is the Issuer?
     - What is the Validity?
     - What extensions were created?
     - What is the Serial number? What is the Serial number in hexadecimal?

Inspect the files which were modified when the Certificate was created

Creating the certificate modified many of the files in the **CA** directory (which is inside the **RootCA** directory). The contents of that directory should resemble the following:

```
$  ls -gh RootCA/CA/
total 8.0K
-rwxrwxrwx 1 user 5.8K Oct  4 20:17 ABCD000001.pem
-rwxrwxrwx 1 user   99 Oct  4 20:17 index.txt
-rwxrwxrwx 1 user   21 Oct  4 20:17 index.txt.attr
-rwxrwxrwx 1 user    0 Oct  4 18:59 index.txt.attr.old
-rwxrwxrwx 1 user    0 Oct  4 18:59 index.txt.old
-rwxrwxrwx 1 user   11 Oct  4 20:17 serial
-rwxrwxrwx 1 user   11 Oct  4 19:00 serial.old
```

1. Examine the contents of the files in the `RootCA/CA/` directory
   - `ls -gh RootCA/CA/`
2. Examine `index.txt` -- this file contains a log of each certificate which was issued
   - `cat RootCA/CA/index.txt`
3. Examine `serial` -- this file automatically increments by one after each issued certificate
   - `cat RootCA/CA/serial`
4. Examine `index.txt.attr` -- option which requires each certificate to have a unique subject DN
   - `cat RootCA/CA/index.txt.attr`
5. Examine the `old` versions of these files which were automatically created as a backup
   - `cat RootCA/CA/index.txt.old`
   - `cat RootCA/CA/index.txt.attr.old`
   - `cat RootCA/CA/serial.old`
6. Examine `ABCD000001.pem` -- This file is the serial number and content of the issued certificate
   - `cat RootCA/CA/ABCD000001.pem`

The CA configuration file (**`PracNet_CA-config.conf`**) contains many settings which are applied to the certificate. All the possible directives this file can contain are outside the scope of this course. But doing a quick look through the file to see the basic format is helpful:

```
$  cat RootCA/PracNet_CA-config.conf
```

Try to identify what the various directives do. For instance, find which directive identify the validity period. Or which directives determine the extensions that will be present in the final certificate.

Generate a Private Key and CSR for Bilbo.com

In this task you will pretend that you own the domain Bilbo.com, and that you need to acquire a SSL certificate. This will be similar to the task prior with Frodo.com, except the certificate for Bilbo.com will have additional extensions.

Certain extensions are added automatically by the CA when the certificate is created (as was the case in the prior task for Frodo.com). But other extensions can be added by the server's request by providing them in the CSR. The CA can then decide whether the extension requests in the CSR will be copied to the final Certificate.

Take a look at the content of the CSR Configuration file:

```
$  cat PracNet_CSR-config.conf
```

Notice the `req_extensions` directive, which loads the `requested_extensions` section. This section provides the directive for adding the Subject Alternative Names the server is requesting.

These configuration directives will be provided to the `openssl req` command using the `-config` argument when we generate a CSR for Bilbo.com.

1. Generate a 2048 bit RSA Private Key file for Bilbo.com
   - `openssl genrsa -out bilbo.com.key 2048`
2. Generate a CSR for Bilbo.com using the custom CSR configuration file
   - `openssl req -new -sha256 -config PracNet_CSR-config.conf -key bilbo.com.key -out bilbo.com.csr`
     *(NOTE: this entire command should be typed on **one line**)*
   - Enter the requested information:
     `Country Name` = *< Two letter Country Code, example: `US GB RU IN` >*
     `State or Province` = *< Accept the default by leaving it blank and pressing enter >*
     `Locality Name` = *< Accept the default by leaving it blank and pressing enter >*
     `Organization Name` = *< Accept the default by leaving it blank and pressing enter >*
     `Common Name` = `*.bilbo.com`
3. Examine the contents of the newly created CSR file
   - `openssl req -in bilbo.com.csr -noout -text`
     - *Notice the newly added Requested Extension*

Use the RootCA to create a signed certificate for Bilbo.com

Create a CA signed certificate using `openssl ca`. You will provide as input: the **CA configuration file**, the **CA's certificate**, and the **CA's private key file**, and the **CSR**. You will receive as output the **signed certificate**.

1. Create a CA signed certificate for Bilbo.com
   - `openssl ca -config RootCA/PracNet_CA-config.conf -cert RootCA/ca.cert -keyfile RootCA/ca.key -in bilbo.com.csr -out bilbo.com.cert`
     *(NOTE: this entire command should be typed on **one line**)*
     - *You will be asked to confirm twice -- say **Y**es by typing "y" and pressing enter.*
2. Examine the contents of the Certificate you just created
   - `openssl x509 -in bilbo.com.cert -noout -text`
     - What is the Serial number? What is the Serial number in hexadecimal?
     - What are the Subject, Issuer, and Validity?
     - What extensions were created? Which didn't exist in the Frodo.com certificate?
     - What domain(s) and sub-domain(s) does the Bilbo.com certificate protect?

Inspect the CA files after the Bilbo.com Certificate was created

As before, when the CA generated a certificate, many of the files in the CA directory changed.

```
$ ls -gh RootCA/CA/
total 16K
-rwxrwxrwx 1 user 5.8K Oct  4 20:17 ABCD000001.pem
-rwxrwxrwx 1 user 6.2K Oct  4 21:23 ABCD000002.pem
-rwxrwxrwx 1 user  207 Oct  4 21:23 index.txt
-rwxrwxrwx 1 user   21 Oct  4 21:23 index.txt.attr
-rwxrwxrwx 1 user   21 Oct  4 20:17 index.txt.attr.old
-rwxrwxrwx 1 user   99 Oct  4 20:17 index.txt.old
-rwxrwxrwx 1 user   11 Oct  4 21:23 serial
-rwxrwxrwx 1 user   11 Oct  4 20:17 serial.old
```

1. Examine the contents of the files in the `RootCA/CA/` directory
   - `ls -gh RootCA/CA/`
   - `cat RootCA/CA/index.txt`
   - `cat RootCA/CA/serial`
   - `cat RootCA/CA/index.txt.attr`
2. Examine the `old` versions of these files which were automatically created as a backup
   - `cat RootCA/CA/index.txt.old`
   - `cat RootCA/CA/index.txt.attr.old`
   - `cat RootCA/CA/serial.old`
3. Examine `ABCD000002.pem` -- This file is the serial number and content of the issued certificate
   - `cat RootCA/CA/ABCD000002.pem`