CS-4331-Special Topics in Security

Fall 2016

LAB # 4: Creating and Using Digital Certificates

In this lab, you will explore the process of issuing digital certificates. You will act as a Certificate Authority (CA) who will issue certificates to other entities. As a CA you will create a self-signed certificate (i.e., a root certificate) whose corresponding private key will be used to sign certificates for your customers. Your root certificate will be loaded into the browser so your client's certificates can be recognized. The lab will give you insights into how a real CA works.

A subset of the commands that you will use (e.g., ca, req, x509) require certain attributes to be specified in the openssl configuration file (openssl.conf). It's a good idea to make a configuration file for your CA separate from the system openssl configuration file. For this reason you should copy the system configuration file to your current working directory. This configuration file is likely at /etc/ssl/openssl.cnf. If it's not here, you can run *locate openssl.cnf* to find it.

With the openssl.cnf file now copied to your working directory, you can open it and observe some of the configurations inside it. The *CA-default* section of this file refers to several sub-directories that you will need to create inside your current directory. The main ones required for this project are as below. Note that "dir" is just a variable holding a directory name.

```
dir = ./demoCA                  # Where everything is kept
certs = $dir/certs              # Where the issued certs are kept
crl_dir = $dir/crl              # Where the issued crl are kept
new_certs_dir = $dir/newcerts   # default place for new certs.
database = $dir/index.txt       # database index file.
serial = $dir/serial            # The current serial number
```

Create an empty file for the index.txt file, and put a single number such as 1000 in the serial file. Having done these basic configurations, you are now good to go with certificate creation.

Step #1: Creating the root certificate (or self-signed certificate) for your CA.

Run the following command to create the root certificate.

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

-new denotes a new keypair, -x509 specifies an x509 certificate -keyout specifies the file where the new private key will go, -out determines where the certificate will go, and -config tells openssl to use our config rather than the default config.

The command will prompt you for a password which you will use to each time you have to create certificates for your customers. After entering the command, you will be asked to enter attributes of your CA, such as country name, state, email address, common name, etc.

Note that the –x509 attribute is critical. The same exact command without it will result into the creation of a certificate signing request (CSR) in the file specified under the switch "-out" instead of a certificate. In the ensuing sections we will reuse the same command but without the –X509 to create a CSR.

Step #2: Creating a Certificate Signing Request

We now have a fully-fledged CA ready to sign certificates. Our first client will be CS-4331.com. This client will have to first generate a private-public key-pair before creating a CSR and sending it to the CA. The CSR will basically contain the client's public key and its identifying information. The CA, on receiving the CSR will create and sign the certificate for the client.

(a) Here is generating the public-private key pair:

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

The keys will be stored in the file server.key. You might be surprised what aes is doing here. It is actually used to encrypt the private key. The command will prompt you for a password which will be used to generate the aes key to encrypt the private key.

Because the file "server.key" is encoded and encrypted, the following command is required to view its contents (i.e., modulus, exponents, etc)

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

(b) Here is generating the CSR:

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

Observe that this command is similar to the earlier described command albeit with the the –x509 option missing.

(c) Here is generating the certificate:

After passing the csr (file name: server.csr) to the CA, the CA then creates the certificate (in server.crt) using the following command

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

There is a chance that openssl will refuse to generate certificates at this stage. The most likely reason is that certain fields in your requests do not match those of the CA. You can address this problem by either changing the policy or changing you requests to match the

policy (see *policy_match* in configuration file). The simplest way to circumvent policy issues is by changing the line:

policy = policy_match" to "policy = policy_anything

Step #3: The CS-4331.com uses the certificate

We now have a public-key certificate generated for our client and we will use it to secure web browsing.

First we add the 127.0.0.1 CS-4331.com entry to /etc/hosts. This tells the client that CS-4331.com is located on 127.0.0.1 (i.e., we pre-empt a fully fledged DNS query).

We then follow the following steps to launch a simple web server with the certificate generated using OpenSSL (i.e., this server will use the created certificate to prove its identity to the browser):

Combine the secret key and certificate into one file
cp server.key server.pem
cat server.crt >> server.pem

Launch the web server using server.pem
openssl s_server -cert server.pem –www

By default, the server will listen on port 4433. To access the server, browse to https://CS-4331.com:4433/ in the Iceweasel browser.

What do you observe? Explain the reason behind the observation.

You may now add your certificate to the browser's certificate repository. To add it, Click "Edit", then "Preferences", then "Advanced", then "Certificates" tab and finally "View Certificates" button.

A list of certificates already accepted by the browser will be displayed. Click "Import" to import your CA's certificate ca.crt, browse to ca.crt and select the option "Trust this CA to identify websites". Now your CA's certificate is part of the list of accepted certificate.

Again browse to https://CS-4331.com:4433/ to view our client's secured website.

What do you observe? Explain the reason behind the observation.