INGI2347: Installing a web server supporting HTTPS

Xavier Carpent, Xiao Chen, Marco Canini

April 28, 2015

This lab's goal is to show you how to install your own webserver supporting HTTPS, and to get you more familiar with Public Key Infrastructure. It has to be done alone, either on your own laptop or the lab's computers. However, please team up in advance with a colleague because a part of the exercise will be done in pairs.

1 Setting up the environment

Typically, a webserver is run by root. Since some of you use machines from the lab (on which you are not root), this tutorial will cover the installation of a webserver from a regular user's perspective. For the sake of simplicity, we will use a lightweight and easily-configurable webserver, `lighttpd`. If you work on your laptop and/or if you are more comfortable with another webserver, feel free to install the webserver as root on your laptop, and/or use any alternative.

Start by downloading `lighttpd` into your home directory. Open a terminal and type in the following commands.

```
wget http://download.lighttpd.net/lighttpd/releases-1.4.x/lighttpd-1.4.35.tar.gz
tar -xf lighttpd-1.4.35.tar.gz
```

Next, install the software in your home directory by doing:

```
cd lighttpd-1.4.35/
./configure --prefix=$HOME/lighttpd --with-openssl \
--with-openssl-libs=/usr/lib64/openssl --without-pcre --without-bzip2
make
make install
```

You might encounter such errors, install the necessary software respectively:

```
configure: error: C compiler cannot create executables ...  --> sudo apt-get install libc6-dev.
configure: error: configure: error: pcre-config not found ...  ---> sudo apt-get install libpcre3-de
configure: error: zlib-headers and/or libs where not found ...  ---> sudo apt-get install zlib1g-dev
configure: error: bzip2-headers and/or libs where not found ...  ---> sudo apt-get install libbz2-de
```

Make sure it works as expected by typing:

```
cd
lighttpd/sbin/lighttpd -v
```

You should see the line "`lighttpd/1.4.35 (ssl) - a light and fast webserver`". Make sure the "`(ssl)`" bit is there, too. If there is not, check your openssl directory. You need to change /usr/lib64/openssl to wherever the openssl library is. Probably, the library is located in "/usr/lib/openssl".

2 Getting the webserver running

Create a sample HTML document, for example by doing:

```
cd
mkdir html/
echo '<html>Hello, world!</html>' > html/index.html
```

Then, create the configuration file for `lighttpd` (we will name it `server.conf`) in your home directory, and fill it with:

```
server.document-root = "/path/to/your/home/html/"
server.port = 64080
index-file.names = ("index.html")
mimetype.assign = (".html" => "text/html")
```

Notice that we set the server port to 64080. Normally, HTTP runs on port 80 but ports from 0 to 1023 are restricted to root. Moreover, only a few ports are open to students on machines of the Intel room.

Finally, run your server by doing:

```
cd
lighttpd/sbin/lighttpd -D -f server.conf
```

Open your favorite browser and go to `http://localhost:64080/`. You should be able to see your greeting message. Verify that you can access your buddy's webpage as well by replacing `localhost` with her/his IP or hostname.

3 Self-signed certificate and first HTTPS test

Start by generating a self-signed certificate. As a reminder, here's what your commands should look like:

```
openssl genrsa 1024 > server.key
openssl req -new -key server.key -out server.csr
openssl x509 -req -in server.csr -signkey server.key -out server.crt
cat server.crt server.key > server.pem
```

Be sure to know what each of these do. Think about what CN you should put in order to avoid errors. When you think it is correct, enable SSL on your webserver by editing `server.conf` and appending:

```
ssl.engine = "enable"
ssl.pemfile = "/path/to/server.pem"
```

Also, change the server port to `server.port = 64443`. Remember to restart your server whenever you make changes to the configuration files. If it seems changes have not taken effect, try emptying the cache of the testing browser.

Visit the page of your friend (in `https`). Is everything alright ? Observe the traffic with wireshark if possible, and identify the certificate.

4 A certificate signed by a CA

Instead of signing your own CSR, send it to a CA (represented for the sake of the exercise by `xiao.chen@uclouvain.be`) to receive your certificate. Please name your CSR explicitly. What is also necessary to complete the chain of trust ? Try it out on your server and the one of your friend. Make sure no error messages remain.

5 Client authentication

You will now create your own Certification Authority. Use the following commands:

```
openssl genrsa 2048 > ca.key
openssl req -new -key ca.key -out ca.csr
openssl x509 -req -in ca.csr -signkey ca.key -out ca.crt
```

What does each of them do ? For the purpose of signing CSR's as a CA, the default configuration file of SSL suggests using the following file structure. Make sure you create the following files and directories (`man ca`):

```
./demoCA                        - main CA directory
./demoCA/cacert.pem             - CA certificate
./demoCA/private/cakey.pem      - CA private key
./demoCA/serial                 - CA serial number file
./demoCA/index.txt              - CA text database file
./demoCA/newcerts/              - where new certificates will be placed
```

The file `index.txt` is empty, but the file `serial` should contain an integer, say `01`. Once this has been set up, you are ready to sign CSR's.

Import this authority into your browser, as well as the certificate of your friend's CA . Then, generate a client CSR by using:

```
openssl genrsa 1024 > client.key
openssl req -new -key client.key -out client.csr
```

Sign your own CSR as well as your friend's using the CA you created above:

```
openssl ca -in client.csr -out client.crt
```

Note: in some SSL installations, the default configuration file does not use `demoCA`. You may copy the default configuration file (usually `/etc/.../openssl.cnf`) locally, modify the `dir` option in the local copy to `demoCA` and finally use the `-config` option to specify the new configuration file.

For your own certificate, use the following command to export it to a format browsers use (and import it to your browser):

```
openssl pkcs12 -export -out client.p12 -in client.crt -inkey client.key
```

Finally, in order to enable client authentication, add the following to `server.conf`:

```
ssl.verify-peer = "enable"
ssl.verify-depth = 1
```

Browse your website and your colleague's website and make sure everything works properly.