MIECT: Security 2020-21

Practical Exercises: Mutual authentication with X.509 certificates, TLS and HTTPS

December 3, 2020 Due date: no date

Changelog

- v1.0 - Initial version.

1 Introduction

Smartcards can be used to authenticate users and this has been observed for the purpose of authenticating a user logging in to the local machine (with PAM). However, when using Web technologies the concept can be easily extended in order to allow authentication of users remotely. By remotely we consider the case where the server providing a Web service or Web page is not collocated with the client's browser, and the smartcard is not directly available to the server.

This laboratory guide will focus in the setup of a secure server, using a custom Certification Authority (CA). By importing the CA public key (self-signed public key certificate), clients can validate the authenticity of the server. At the same time, it can be considered that some contents are restricted to a limited number of individuals, and the access to those contents can be controlled by means of a smartcard.

For the execution of this guide we will use Linux and the following packages: the Portuguese Citizen Card tools, `apache2`, `php`, `xca`, and `Qt5`.

2 Creation of a Certification Authority

Certification Authorities are vital for authentication of services across the Internet. They are considered to be trusted and this trust is inherited by the certificates they issue (sign). Although CAs commonly have a global reach and are to be globally trusted, this is not really a requirement and custom CAs can be created. If clients install the custom CA and trust it, all certificates issued by the CA will also be trusted, just like any other commercial, widely deployed root CA. This is useful for services which have a limited number of users accessing it.

In this step we will create a custom CA in order to generate signed certificates for a personal Web server. For this purpose we will rely on the `xca` software.

The first step is to launch the `xca` application and create a new database. Do not forget to specify a password!

Afterwards, generate a new private key for the CA, named `CAKey`. Usually, CA keys are considerably stronger that those of server certificates. Consider 4096 bits if possible.

Then create a CA by selecting *Certificates→New Certificate*. Do not forget to select the key you just created, as well as the default CA template (you must select it and then choose *Apply all*. Also, you

must fill the identification data for the CA, define the key usage extension as critical and define the validity of the CA (a few years).

After having the CA up and running, we can generate a certificate for our server. For that, we need first to create a Certificate Signing Requests (CSR), and sign that request with the CA just created. Go to the tab *Certificate Signing Requests* and click *New Request*. This time, apply the *HTTP Server* template and fill the remaining fields. Use `localhost` as the common name, generate a new key and define the key usage extension as critical. You can select the TLS Web Server Authentication extended key usage.

With the CSR, the certificate can be finally generated by signing the CSR with the `CAKey`. Right-click on the CSR and select *sign*. Please make sure you sign the CSR with the CAKey!

The final step is to export the CA Certificate, and both the Server Certificate and Private Key. Export everything as PEM. Do not export the CA Private Key! This key is secret and should **NEVER** be distributed!

3 Installation of a server certificate

Installation of the certificates and keys requires copying the files to specific places. The CA's certificate should go to `/etc/ssl/certs`. The Server certificate and private key should go to `/etc/ssl/private`.

Clients should also install the CA certificate in their keystore. For this purpose, import CA certificate to a browser's list of root CAs.

For the server we will use `apache2`, which must also be installed. Afterwards, the `SSL` module must be enabled by issuing:

```
a2enmod ssl
```

In the folder `/etc/apache2/sites-available` there is a file named `default-ssl.conf`. Copy this file to `/etc/apache2/sites-enabled` (or do a symlink). This file defines the SSL configuration the server will use, and must be edited in order to consider the cryptographic material just created. Be sure to modify the following variables:

- `SSLCertificateFile`: This should refer the PEM file containing the server certificate.
- `SSLCertificateKeyFile`: This should refer the PEM file containing the server private key.
- `SSLCACertificatePath`: This should refer `/etc/ssl/certs`.

Edit all required variables and restart the server by executing (as administrator):

```
service apache2 restart
```

Verify that the browser can access the server by typing the URL `https://localhost`.

If everything is correct, there should be no warning or error, and the page should be secure. The practical result is that the server is authenticated and the connection is secure. The browser accepts the server's certificate because it was issued by a trusted CA. If the `common-name` field of the certificate is different from `localhost`, the browser will show a warning message.

4 Using smartcards for authenticating users

The next step is to authenticate the client, and in this particular case, identify the individual user by means of the Portuguese Citizen Card.

The first step is to download the complete certification chains of the Portuguese CC (only for authentication), and install their certificates into the `/etc/apache2/certs/PTEIDCC` directory. In the course Web page you can find te means to get these files (check for the practical class related with signatures with Citizen Card).

To do so, concatenate all PEM files related with authentication hierarquies into a single PEM file (e.g. `PT.pem`).

Note: test CCs need to use an alternative certification chain, see also the course Web page for getting them.

Next proceed to configure the server by editing the file `default-ssl.conf`. In this file, inside the `VirtualHost *:443` section, add the following content:

```
SSLCACertificateFile /etc/apache2/certs/PTEIDCC/PT.pem

<Location /secure>
    SSLVerifyClient require
    SSLVerifyDepth 10
    SSLOptions +OptRenegotiate +StdEnvVars +ExportCertData
</Location>
```

Create the directory `/var/www/html/secure` and put a file named `index.php` with an HTML success message.

Now try to access the URL `https://localhost/secure`. Without a smartcard installed, the browser should present and error as it is unable to provide a valid certificate for the server. In this case, the server will only allow certificates signed by a certificate present in the `PT.pem` file.

Using Wireshark, examine the packets exchanged between the browser and the server.

In order to successfully access the page you must go to Firefox preferences and choose:

Advanced→Security Devices→Load

Provide any name and specify the value `/usr/local/lib/libpteidpkcs11.so` as the filename. The smartcard should be available, and if the smartcard is inserted in the reader, your certificates should be listed when choosing the option *Advanced→View Certificates*.

Retry the access to the secure address; it should be granted without asking for a user certificate. Explain why.

Using Wireshark inspect the messages that are exchanged between the browser and the server. Start the browser after the Wireshark, in order to capture the main TLS authentication handshake. Repeat the access to the secure address to observe the session resume handshake.

An important aspect of authenticating users through a smartcard is to clearly identify the individual (Portuguese Citizen) at the application layer (e.g., PHP application). For this purpose, edit the `index.php` file that you previously created and add the following content:

```
<html>
    <head><meta charset="UTF-8"></head>
    <body>
        <pre>
        <?php print_r($_SERVER); ?>
        </pre>
    </body>
</html>
```

Access the secure Web page again and verify the variables that are exposed to the PHP application. It should be noticed that the application can further restrict access to the service by verifying any of these fields. However, it will have the assurance that the user was successfully authenticated by the server.

5 References

- Portuguese Citizen Card web site: `http://www.cartaodecidadao.pt`
- Apache2 ModSSL, `http://httpd.apache.org/docs/2.2/mod/mod_ssl.html`
- XCA, `http://sourceforge.net/projects/xca`