

Introduction to cybersecurity TP 4 - SSL/PKI

Maciej Korczynski & Simon Fernandez

Septembre 2021

Objectives

- Discover PKI
- Generate a certificate for AuthN with webserver

Certificate Authentication

We will see how to create a CA (Certificate Authentication) and how to use it. This CA will be used to later on produce user certificates. For more information, see the wikipedia page

First, create the directories where we will work

```
mkdir CA
cd CA
mkdir newcerts private
```

The `CA/` dir will contain all the files that reside on the Certification Agency side:

- The certificate
- The database of signed certificates
- The keys/requests and certificates that we will have generated

`CA/newcerts` will contain a copy of each new signed certificate.

`CA/private` will contain the private key of our CA. This key will be used to sign certificates.

Copy the configuration file from `/etc/ssl/openssl.cnf` to `CA/`.

This file contains the OpenSSL configuration values. You can modify it and change some default values. (like country codes or key size). It also sets the paths where some files will be searched. If some functions fail, check this configuration file !

First, change the `$dir` value so that the generated files are put in our `CA` folder.

Creating the Certificate Authentication

Copy `openssl.cnf` to `opensslCA.cnf` and take a look at the `[ca]` section of the file. Modify it if needed.

Then, create the CA by running this command and filling the different fields:

```
openssl req -new -x509
            -extensions v3_ca
            -keyout private/cakey.pem
            -out cacert.pem
            -days 3650
            -config ./opensslCA.cnf
```

This will produce two files:

- The private key of the CA, in `private/cakey.pem`

- The CA in `cacert.pem`

You can display the CA you just created with

```
openssl x509 -in cacert.pem -noout -text
```

- Who is the Issuer ? The Subject ?
- What are their roles ?
- Is this CA signed ? By who ?

Creation of a user certificate

We will create a certificate for a fictive company (that will be used by an application with SSL) and we will sign it with our CA.

The creation of a certificates works in two steps. First, generate a public/private key pair and a certificate request. Then, send this request to the CA that will sign it and generate the actual certificate.

To symbolize the fact that this step is done on another server, we will create a new folder that will hold the company data.

```
cd ../
mkdir Company
cd User
```

Creation of the request

Create the `opensslCompany.cnf` file from `openssl.cnf`, update the `$dir` field, take a look at the `[usr_cert]` section, then create the request with :

```
openssl req -new
            -nodes
            -out req.pem
            -config ./opensslCompany.cnf
```

Note: If you want to create a certificate for a website, then the **Common Name** field must be the domain name of the website.

This command will produce two files:

- A private key `privkey.pem`
- A certificate signature request `req.pem`

We can see the content of the request with:

```
openssl req -in req.pem -text -verify -noout
```

This step only create a certification request the certificate. It is not signed yet. This step can be done on a remote machine, then, in order to sign the certificate with the CA, we will need to send the `req.pem` file to the CA and get back the certificate once signed. But remember, the private key must NEVER be sent on the network

- What does the request contain ?
- What is the content of the **Signature Value** field ? Who signed it ? Why ?

We will “send” this request to the CA, by copying `req.pem` to the `CA/` folder

Signing the certificate with the CA

Back to the `CA/` folder

Create the database listing the signed certificates:

```
echo '01' > serial
touch index.txt
```

In the [`ca`] section of the config file, subsection [`policy_match`], chose which fields must match between the CA and the request. For example, if you want your CA to be able to sign certificates from a different company, change the `organizationName` value to `supplied`

Now, lets process the certification request and generate a signed certificate:

```
openssl ca -out cert.pem
           -config ./opensslCA.cnf
           -infiles req.pem
```

This will update the two files of the database (`serial` and `index.txt`) and will create two files:

- The signed certificate `cert.pem`
- A copy of it in `CA/newcerts`

Read the content of the certificate:

```
openssl x509 -in cert.pem -noout -text -purpose
```

- Who is the Issuer ? Who is the subject ?
- What is inside the `index.txt` and `serial` files ?

Now that the certificate is generated and signed, we can send it back to the company

Distributing the CA

Now, anyone can verify that the Company certificate (`Company/cert.pem`) is well signed by the CA (`CA/cacert.pem`), they just need both certificates.

The CA certificate can be added to web browsers, mail clients, ... like root certificate, and it will be used to validate the Company certificate

You can check if a certificate was signed by the CA with

```
openssl verify -CAfile cacert.pem cert.pem
```

Renew a certificate

A certificate can expire and become invalid if:

- The certificate itself expires (see the expiration date in the creation process)
- The signing CA expires

In the first case, we can either create a new certificate and sign it, or re-sign the old one

In the second case, a new CA must be generated and distributed, then the certificate must be re-created and signed

The CA can't sign two certificates with the same `Common Name`, we first need to revoke the old certificate.

`index.txt` contains the list of certificates, indexed by their `Common Name` (CN). For example, to revoke the certificate `02.pem` (in `newcerts/`), we can run :

```
openssl ca -revoke newcerts/02.pem -config ./opensslCA.cnf
```

- How does this affect the `index.txt` file ?

CRL

CRL (certificate revocation list) is a list of revoked certificates. When you revoke a certificate, you must update this list.

```
echo 01 > crlnumber
openssl ca -gencrl -crldays 31 -config ./openssl.cnf -out rootca.crl
```

We can check the content of the revocation list with :

```
openssl crl -in rootca.crl -noout -text
```

Formats

Depending on the application, you may have to convert the certificate to another format.

From PEM to DER :

```
openssl x509 -in input.crt -inform PEM -out output.crt -outform DER
```

From DER to PEM :

```
openssl x509 -in input.crt -inform DER -out output.crt -outform PEM
```

Apache – Authentication - SSL

For a web server, certificates can be used to authenticate the server to the user (so that the user knows it is talking to the right entity), or the other way around (so that the server can authenticate the user). We will see how to deploy both methods with Apache

Client authentication by certificate

We created our own CA. We can now create a certificate per client and sign it with the CA. After this we can add these lines to the Apache `Virtualhost` configuration to force the client authentication by certificate :

```
SSLCACertificateFile /etc/apache2/ssl/cacert.pem
SSLRequireSSL
SSLVerifyClient require
SSLVerifyDepth 1
```

This will only allow users authenticating with a certificate signed by `cacert.pem`

You may need to activate the Apache SSL module :

```
a2enmod ssl
```

Server certificat

You can also use a certificate to enable https on your website and sign it with your CA (or create a self-signed certificate). To configure your website you need to add these line, pointing to the signed certificate and the private key used to authenticate the server :

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/apache.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache.key
SSLProtocol ALL
```

You also need to active SSL mode on apache.

Note : A certificate can only be used to authenticate the server with a given **Common name**. To simulate a DNS server and give a name to your webserver, you can edit the `/etc/hosts` file on your client VM.

Bonus : What is `SSLProtocol` ? Give your opinion on the proper configuration of it

Practice

Establish a authentication for one of your website by certificate. You can also active https for one of your website (or for both).

You will find to some advice about the use of SSL in <https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices>

Bonus - Apache Hardening

`mod_security`

`mod_security` offers an open-source web application firewall (call WAF) and a intrusion detection and prevention system for web applications.

It is widely used. To use it you need to install the package `libapache2-mod-security2` and to active in apache the `mod mod-security` . As a good configuration pratice, you can follow the OWASP Rule Set.

`mod_evasive`

`mod_evasive` offers a open-source system to detect and prevent HTTP brute-force attack (DOS/DDOS). Like `mod_security`, it is widely used. To install it you need the package `libapache2-mod-evasive`.

Then create the file `/etc/apache2/mods-available/mod-evasive.conf` and activate the mod `mod-evasive`. An example of `mod-evasive.conf` file :

```
<ifmodule mod_evasive20.c>
    DOSHashTableSize 3097
    DOSPageCount 2
    DOSSiteCount 50
    DOSPageInterval 1
    DOSSiteInterval 1
    DOSBlockingPeriod 10
    DOSLogDir /var/log/mod_evasive
    DOSEmailNotify yourmail@mail.com
    DOSWhitelist 127.0.0.1
</ifmodule>
```

- What are those different parameters ?

You can specify a command with `DOSSystemCommand` in case of attack. For example block the IP with `iptables`. The problem is that the apache deamon should not have the right to modify `iptables` rules. A better way is to combine `mod_evasive` with `fail2ban`.

Practice :

Enhance the security of your website. For example you can active `mod_security` and `mod_evasive`. Do not hesitate to configure some other features (enable custom apache logging per virtualhost, disable following links, ...)