Configurar PKI

Luis Ángel Cotelo Leyva

1.- Ámbito del problema

Se quiere configurar una infraestructura de clave pública (Public Key Infrastructure) con el objetivo de autenticar a otros en la red.

Una PKI, resumiendo algo, es un conjunto de máquinas, que certifican ante el resto de usuarios de la red que un usuario o servicio es quien dice ser. Esto se hace mediante el uso de claves públicas y privadas.

En el problema que nos atañe, se creará una entidad certificadora raíz (root CA) que sea capaz de certificar a otras entidades certificadoras subordinadas (intermediate CA) para que estas hagan el trabajo de certificar a terceros. Una de las entidades certificadoras subordinadas, certificará a un servidor https con el que más tarde se realizará una conexión para ver si todo ha salido bien.

1.1.- ¿Por qué dos tipos de entidad?

Que haya dos entidades certificadoras, una raíz y otra subordinada supone una capa de seguridad más a la infraestructura, ya que si una de las entidades cae, por el motivo que sea, hay que revocar en cadena todos los certificados que haya firmado esta entidad.

Si se es precavido y se protege a la entidad certificadora más importante, la raíz, apagando y desconectando esta máquina de la red, al producirse un accidente, no habría que volver a montar la infraestructura. Bastaría con revocar los certificados a partir de la entidad comprometida, levantar la CA raíz, volver a certificar alguna entidad intermedia y volver a apagarla.

2.- Configurar CA Raíz

Voy a usar una máquina virtual en Microsoft Azure. No tiene mucha capacidad de cómputo ni especificaciones técnicas, pero tampoco es necesario, ya que en el momento que esta entidad certifique a una subordinada, se apagará y desconectará de la red. También voy a usar OpenSSL para configurar la entidad certificadora.

Una vez montada e iniciada la máquina virtual (Debian 10 en mi caso) hay que configurar el sistema de clave privada/pública.

Lo primero es actualizar los repositorios y actualizar el sistema.

```
cotelo@ca:~$ sudo apt-get update
Hit:1 http://debian-archive.trafficma
Hit:2 http://debian-archive.trafficma
Hit:3 http://debian-archive.trafficma
Hit:4 http://debian-archive.trafficma
Reading package lists... Done
cotelo@ca:~$ sudo apt-get upgrade
```

Una cosa a tener en cuenta tras actualizar el sistema, es que si las entidades raíz y subordinada están en husos horarios distintos, sus horas locales podrían diferir, por lo que podría haber problemas. Para solucionar este inconveniente, simplemente hay que instalar ntp en ambas máquinas.

```
cotelo@ca:~$ sudo apt-get install ntp
```

Ahora hay que crear la estructura de directorios y archivos para la CA raíz. Esta estructura la voy a crear en el directorio / por tanto voy a necesitar cambiar al usuario root por unos instantes.

```
cotelo@ca:/$ sudo -i
root@ca:~# mkdir /root/ca
root@ca:~# cd /root/ca
root@ca:~/ca# mkdir newcerts certs crl private requests
root@ca:~/ca# touch index.txt
root@ca:~/ca# touch index.txt.attr
root@ca:~/ca# echo '1000' > serial
root@ca:~/ca# ls
certs  crl  index.txt  index.txt.attr  newcerts  private  requests  serial
root@ca:~/ca#
```

Los archivos y directorios creados son necesarios para la posterior configuración de OpenSSL.

Ahora hay que llevar un archivo de configuración a la máquina. Este archivo es el siguiente: openssl_root.cnf. Para ello, puede hacerse con scp, o ayudarnos del repositorio ya que está subido ahí. Para descargarlo del repositorio, se requiere la visualización de los archivos sin el código html propio de la página.

Con el método que sea, se ubica el archivo *openssl_root.conf* en el directorio que hemos preparado /root/ca

```
root@ca:~/ca# wget raw.githubusercontent.com/cotelus/ConfigurePKI/main/configuracion_raiz/openssl_root.cnf
URL transformed to HTTPS due to an HSTS policy
--2020-12-05 12:37:46--  https://raw.githubusercontent.com/cotelus/ConfigurePKI/main/configuracion_raiz/openssl_root.cnf
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 199.232.56.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|199.232.56.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4259 (4.2K) [text/plain]
Saving to: 'openssl_root.cnf'

openssl_root.cnf            100%[=============================================>]   4.16K  --.-KB/s    in 0s

2020-12-05 12:37:46 (33.5 MB/s) - 'openssl_root.cnf' saved [4259/4259]

root@ca:~/ca# ls
certs  crl  index.txt  index.txt.attr  newcerts  openssl_root.cnf  private  requests  serial
root@ca:~/ca#
```

El archivo *openssl_root.conf* es una plantilla. En mi caso solo necesito modificar *dir* y DOMAINNAME para que coincida con mi estructura de directorios y su ubicación.

```
# The root key and root certificate.
private_key       = $dir/private/ca.DOMAINNAME.key.pem
certificate       = $dir/certs/ca.DOMAINNAME.crt.pem

# For certificate revocation lists.
crlnumber         = $dir/crlnumber
crl               = $dir/crl/ca.DOMAINNAME.crl.pem
crl_extensions    = crl_ext
```

```
[ CA_default ]
# Directory and file locations.
dir               = /root/ca
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt
serial            = $dir/serial
RANDFILE          = $dir/private/.rand

# The root key and root certificate.
private_key       = $dir/private/ca.COTELO.key.pem
certificate       = $dir/certs/ca.COTELO.crt.pem

# For certificate revocation lists.
crlnumber         = $dir/crlnumber
crl               = $dir/crl/ca.COTELO_crl.pem
crl_extensions    = crl_ext
```

2.1.- Generar clave privada CA raíz y firmar certificado

En el directorio anterior, donde está ubicada toda la estructura de ficheros y archivos, se genera con openssl la clave privada.

```
root@ca:~/ca# openssl genrsa -aes256 -out private/ca.COTELO.key.pem 4096
```

Ahora se genera el certificado de la CA raíz con la clave privada que se acaba de generar.

```
root@ca:~/ca# openssl req -config openssl_root.cnf -new -x509 -sha512 -extensions v3_ca -key /root/ca/private/ca.COTELO.
key.pem -out /root/ca/certs/ca.COTELO.crt.pem -days 3650 -set_serial 0
```

(Copio la orden aquí ya que la imagen se ve muy pequeña)

$ openssl req -config openssl_root.cnf -new -x509 -sha512 -extensions v3_ca -key /root/ca/private/ca.COTELO.key.pem -out /root/ca/certs/ca.COTELO.crt.pem -days 3650 -set_serial 0

La orden anterior, lo que hace es generar un certificado, pero con los siguientes requisitos:

- El archivo de configuración de OpenSSL va a ser *openssl_root.cnf*
- El algoritmo de encriptación va a ser *sha512*
- La clave para firmar el certificado va a ser */root/ca/private/ca.COTELO.key.pem*
- El certificado generado va a ser */root/ca/certs/ca.COTELO.crt.pem*
- El certificado va a durar 3650 días (aproximadamente 10 años)

3.- Configurar CA Subordinada

De forma muy similar a la CA Raíz, esta máquina va a ser configurada en Azure con las mismas características.

Después de actualizar la máquina e instalar ntp, se procede a crear la estructura de archivos y directorios de manera similar a la CA raíz.

```
root@subordinada:~# mkdir /root/subordinada
root@subordinada:~# cd /root/subordinada/
root@subordinada:~/subordinada# mkdir certs newcerts crl csr private
root@subordinada:~/subordinada# touch index.txt
root@subordinada:~/subordinada# touch index.txt.attr
root@subordinada:~/subordinada# echo 1000 > crlnumber
root@subordinada:~/subordinada# echo '1234' > serial
root@subordinada:~/subordinada# ls
certs  crl  crlnumber  csr  index.txt  index.txt.attr  newcerts  private  serial
```

En la dirección /root/subordinada voy a descargar el archivo openssl_intermediate.cnf

```
root@subordinada:~/subordinada# wget https://raw.githubusercontent.com/cotel
us/ConfigurePKI/main/configuracion_raiz/openssl_intermediate.cnf
```

De manera similar a la CA raíz, modifico los valores dir y DOMAINNAME para que se ajusten a mi problema.

```
[ CA_default ]
# Directory and file locations.
dir               = /root/subordinada
certs             = $dir/certs
crl_dir           = $dir/crl
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt
serial            = $dir/serial
RANDFILE          = $dir/private/.rand

# The root key and root certificate.
private_key       = $dir/private/int.COTELO.key.pem
certificate       = $dir/certs/int.COTELO.crt.pem

# For certificate revocation lists.
crlnumber         = $dir/crlnumber
crl               = $dir/crl/int.COTELO.crl.pem
crl_extensions    = crl_ext
```

3.1.- Generar clave privada y petición de firma CA subordinada

Al contrario que en la CA raíz, esta entidad subordinada no puede firmarse a sí misma, ya que depende de otra entidad. En este caso, se genera un certificado de petición de firma que se enviará a la CA raíz, esta lo firmará y lo devolverá ya firmado a la CA subordinada.

Aún así, lo primero es generar la clave privada de esta CA subordinada.

```
root@subordinada:~/subordinada# openssl req -config /root/subordinada/openss
l_intermediate.cnf -new -newkey rsa:4096 -keyout /root/subordinada/private/i
nt.COTELO.key.pem -out /root/subordinada/csr/int.COTELO.csr
Generating a RSA private key
..................++++
.....................++++
writing new private key to '/root/subordinada/private/int.COTELO.key.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:ES
State or Province Name []:GRANADA
Locality Name []:ALBOLOTE
Organization Name []:COTELO
Organizational Unit Name []:
Common Name []:caCOTELO
Email Address []:
```

3.2.- Enviar petición de CA subordinada a CA raíz

Uno de los archivos que generó la orden anterior fué */root/subordinada/csr/int.COTELO.csr*. Este archivo tiene una extensión .csr, esta extensión significa “solicitud de firma de certificado” en nuestro idioma. Este tipo de archivos se generan en el servidor donde van a ser utilizados para enviarlos a la entidad certificadora correspondiente para que acredite la identidad.

Para este caso, hay que enviar este archivo a la máquina CA raíz. Para ello voy a usar el comando scp, haciendo que la máquina CA raíz sea la que se encargue del tema del transporte.

```
root@ca:/# scp cotelo@13.69.136.200:/root/subordinada/csr/int.COTELO.csr /root/ca/
/root/ca/certs/             /root/ca/index.txt.attr     /root/ca/private/
/root/ca/crl/               /root/ca/newcerts/          /root/ca/requests/
/root/ca/index.txt          /root/ca/openssl_root.cnf   /root/ca/serial
root@ca:/# scp cotelo@13.69.136.200:/root/subordinada/csr/int.COTELO.csr /root/ca/requests
/
The authenticity of host '13.69.136.200 (13.69.136.200)' can't be established.
ECDSA key fingerprint is SHA256:3dzKvNDeFkwCMKtv6p9jXNhCy8F5lsZ7SqJmJlqGets.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '13.69.136.200' (ECDSA) to the list of known hosts.
cotelo@13.69.136.200's password:
int.COTELO.csr                                        100% 1675     1.5MB/s   00:00
root@ca:/# cd /root/ca/requests/
root@ca:~/ca/requests# ls
int.COTELO.csr
root@ca:~/ca/requests#
```

Ahora, desde la máquina CA raíz se firma el archivo .csr

```
root@ca:~/ca# openssl ca -config /root/ca/openssl_root.cnf -extensions v3_intermediate_ca
-days 3650 -notext -md sha512 -in /root/ca/requests/int.COTELO.csr -out /root/ca/requests/
int.COTELO.crt.pem
Using configuration from /root/ca/openssl_root.cnf
Enter pass phrase for /root/ca/private/ca.COTELO.key.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4097 (0x1001)
        Validity
            Not Before: Dec  5 17:53:41 2020 GMT
            Not After : Dec  3 17:53:41 2030 GMT
        Subject:
            countryName               = ES
            stateOrProvinceName       = GRANADA
            organizationName          = COTELO
            commonName                = caCOTELO
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                10:86:08:B3:07:91:5F:F4:7A:48:B1:1C:09:C9:9D:C9:0B:D0:8B:AE
            X509v3 Authority Key Identifier:
                keyid:3E:AB:E4:76:B4:54:07:62:4F:AC:A0:D3:F6:93:F4:33:F6:0F:20:1F

            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
Certificate is to be certified until Dec  3 17:53:41 2030 GMT (3650 days)
Sign the certificate? [y/n]:y
```

Las opciones de openssl son muy similares a cuando se autofirma la CA raíz, siendo la diferencia más notable la opción *"-extensions v3_intermediate_ca"* que es la que dirá que se está firmando un certificado para una CA subordinada.

3.3.- Cadena de certificados

Hecho esto hay que crear la cadena de certificados. Esto es un documento en el que se almacena también el certificado de la CA raíz para dar autoridad al documento. Al requerir del certificado de la CA raíz, esta operación se hace también en la CA raíz.

```
root@ca:~/ca# cat requests/int.COTELO.crt.pem certs/ca.COTELO.crt.pem > requests/chain.COT
ELO.crt.pem
root@ca:~/ca# ls requests/
chain.COTELO.crt.pem  int.COTELO.crt.pem  int.COTELO.csr
```

Finalmente, lo que hay que hacer es enviar los archivos */root/ca/requests/int.COTELO.crt.pem* y */root/ca/requests/chain.COTELO.crt.pem* a la máquina que va a ser la CA subordinada.

```
root@ca:~/ca# scp /root/ca/requests/chain.COTELO.crt.pem cotelo@13.69.136.200:/home/cotelo
/
```

```
root@ca:~/ca# scp /root/ca/requests/int.COTELO.crt.pem cotelo@13.69.136.200:/home/cotelo/
```

Como scp no puede ejecutarse como sudo en la máquina destino y el directorio /root/subordinada/ de este, pertenece a root, envío los certificados al directorio personal del usuario cotelo. Suponiendo que este usuario se ha creado sola y exclusivamente para el manejo de los certificados.

Una vez en la máquina CA subordinada, copio los certificados que se han transferido desde la CA raíz a la estructura de directorios que se montó anteriormente para ello.

```
cotelo@subordinada:/$ cd /home/cotelo
cotelo@subordinada:~$ ls
chain.COTELO.crt.pem  int.COTELO.crt.pem
cotelo@subordinada:~$ sudo mv chain.COTELO.crt.pem /root/subordinada/certs/
cotelo@subordinada:~$ sudo mv  /root/subordinada/certs/
.bash_history       .bashrc             .ssh/
.bash_logout        .profile            int.COTELO.crt.pem
cotelo@subordinada:~$ sudo mv int.COTELO.crt.pem /root/subordinada/certs/
cotelo@subordinada:~$ ls
cotelo@subordinada:~$ ls /root/subordinada/certs/
chain.COTELO.crt.pem  int.COTELO.crt.pem
```

En este punto, no va a ser necesaria la CA raíz hasta que haya que renovar los certificados de las CA subordinadas. Es por tanto, que puede apagarse la máquina.

4.- CA subordinada certifica servidor HTTPS

En este punto, ya se puede montar un servidor https con apache y que la entidad certificadora subordinada que se montó anteriormente, certifique al servidor.

Hacer esto es muy similar a que la CA raíz firme a la CA subordinada. De hecho, es el mismo procedimiento. En este caso es el servidor https el que crea el documento de petición de firma y se lo envía a la CA subordinada para que lo firme.

En la máquina que hará de servidor https, se crea la siguiente estructura de directorios:

```
cotelo@httpserver:~$ sudo mkdir /root/auth/private /root/auth/csr /root/auth/certs
```

Se genera la clave privada para el servidor https y la petición de firma:

```
cotelo@httpserver:~$ sudo openssl req -out /root/auth/csr/servidor_https.csr.pem -n
ewkey rsa:2048 -nodes -keyout /root/auth/private/servidor_https.key.pem
Generating a RSA private key
.........+++++
.........................................+++++
writing new private key to '/root/auth/private/servidor_https.key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:GRANADA
Locality Name (eg, city) []:ALBOLOTE
Organization Name (eg, company) [Internet Widgits Pty Ltd]:COTELO
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:httpsCOTELO
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

```
cotelo@httpserver:~$ sudo ls /root/auth/csr
servidor_https.csr.pem
```

Mediante scp se lleva el archivo *root/auth/csr/servidor_https.csr.pem* a la CA subordinada para que lo firme. Además he copiado de forma temporal el archivo al directorio /home/cotelo/ para que el usuario que hace el scp tenga acceso.

```
cotelo@subordinada:~$ sudo scp cotelo@40.113.66.206:/home/cotelo/servidor_ht
tps.csr.pem /root/subordinada/requests/
cotelo@40.113.66.206's password:
servidor_https.csr.pem                    100%  985     1.2MB/s   00:00
```

Ahora se firma la petición desde la CA subordinada.

```
cotelo@subordinada:~$ sudo openssl ca -config /root/subordinada/openssl_intermediate.cnf -extensions server_cert
-days 730 -notext -md sha512 -in /root/subordinada/requests/servidor_https.csr.pem -out /root/subordinada/request
s/servidor_https.crt.pem
Using configuration from /root/subordinada/openssl_intermediate.cnf
Enter pass phrase for /root/subordinada/private/int.COTELO.key.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4660 (0x1234)
        Validity
            Not Before: Dec  5 19:12:26 2020 GMT
            Not After : Dec  5 19:12:26 2022 GMT
        Subject:
            countryName               = ES
            stateOrProvinceName       = GRANADA
            localityName              = ALBOLOTE
            organizationName          = COTELO
            commonName                = httpsCOTELO
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
                SSL Server
            Netscape Comment:
                OpenSSL Generated Server Certificate
            X509v3 Subject Key Identifier:
                77:CD:E0:82:3A:5A:39:F3:58:E1:FC:2E:D1:13:2E:5A:A2:DA:26:E3
            X509v3 Authority Key Identifier:
                keyid:10:86:08:B3:07:91:5F:F4:7A:48:B1:1C:09:C9:9D:C9:0B:D0:8B:AE
                DirName:/C=ES/ST=GRANADA/L=ALBOLOTE/O=COTELO/CN=caCOTELO
                serial:10:01
```

Finalmente, se envía desde la CA subordinada al servidor HTTPS el certificado firmado.

```
cotelo@subordinada:~$ sudo scp /root/subordinada/requests/servidor_https.crt.pem cotelo@40.113.66.2
06:/home/cotelo/
cotelo@40.113.66.206's password:
servidor_https.crt.pem                                        100% 1866     2.0MB/
s   00:00
```

Se envía también el certificado de la CA subordinada (el certificado encadenado Ca raíz y CA subordinada).

```
cotelo@subordinada:~$ sudo scp /root/subordinada/certs/chain.COTELO.crt.pem cotelo@40.113.66.206:/h
ome/cotelo/
cotelo@40.113.66.206's password:
chain.COTELO.crt.pem                                          100% 3940     5.1MB/s   00:00
```

Se comprueba que el módulo del certificado y de la clave privada del servidor https son iguales.

```
root@httpserver:~# openssl x509 -in /root/auth/certs/servidor_https.crt.pem -noout
-modulus
Modulus=E042BD758B7701E02AD89FFB5534F77CF42C30CE5C0A305F1FF28BE490289AE2CF9E98A83F4
31512E286AB1FC4E5E86DC83F6B892E9E7AA0F5C07B6CCDB2A2E9C703799F8A59B027B991FFE9A1B417
6A0B4C870B5912328A838C202BB0DA5AEEE22FCB07F75ED696278949005BD67C0C46AD755E8AD8D3CCC
8A0CA5AF9C3039B97D3817B300B77A47FF42871D742B0613ECB5B7401CBA68B66FD8170AEFD11F35ABF
D1D569C15DF13D7269780B8A1B3E9A2D9EE098DD75AC11B9F43863642C3FF6F3EF7DA207F145E664BFD
2D8C4E1A3B1C9F441EFCD0F2CB6A3CE68E16C75DD5EDAB7FF0C559C39118223611BE885276924972EA1
FB30ABB937881BDEDA7C07
root@httpserver:~# sudo openssl rsa -in /root/auth/private/servidor_https.key.pem -
noout -modulus
Modulus=E042BD758B7701E02AD89FFB5534F77CF42C30CE5C0A305F1FF28BE490289AE2CF9E98A83F4
31512E286AB1FC4E5E86DC83F6B892E9E7AA0F5C07B6CCDB2A2E9C703799F8A59B027B991FFE9A1B417
6A0B4C870B5912328A838C202BB0DA5AEEE22FCB07F75ED696278949005BD67C0C46AD755E8AD8D3CCC
8A0CA5AF9C3039B97D3817B300B77A47FF42871D742B0613ECB5B7401CBA68B66FD8170AEFD11F35ABF
D1D569C15DF13D7269780B8A1B3E9A2D9EE098DD75AC11B9F43863642C3FF6F3EF7DA207F145E664BFD
2D8C4E1A3B1C9F441EFCD0F2CB6A3CE68E16C75DD5EDAB7FF0C559C39118223611BE885276924972EA1
FB30ABB937881BDEDA7C07
```

Si todo coincide, se sigue adelante. Desde el servidor https ahora se mueve el certificado firmado a la estructura de directorios que se creó.

```
cotelo@httpserver:~$ sudo mv /home/cotelo/servidor_https.crt.pem /root/auth/certs/
```

Sólo falta enviar el certificado de la CA subordinada, el certificado del servidor y la clave privada del servidor al directorio del que los vaya a cargar apache.

```
cotelo@httpserver:~$ sudo ls /etc/apache2/certificates/
chain.COTELO.crt.pem  servidor_https.crt.pem  servidor_https.key.pem
```

Se modifica el archivo de configuración de ssl de apache2.

tpserver: ~

```
3.2                                /etc/apache2/sites-available/default-ssl.conf

     SSLEngine on

     #   A self-signed (snakeoil) certificate can be created by installing
     #   the ssl-cert package. See
     #   /usr/share/doc/apache2/README.Debian.gz for more info.
     #   If both key and certificate are stored in the same file, only the
     #   SSLCertificateFile directive is needed.
     #SSLCertificateFile     /etc/ssl/certs/ssl-cert-snakeoil.pem
     #SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
     # CERTIFICADOS SSL PERSONALIZADOS
     SSLCertificateFile      /etc/apache2/certificates/servidor_https.crt.pem
     SSLCertificateKeyFile   /etc/apache2/certificates/servidor_https.key.pem
     SSLCACertificateFile    /etc/apache2/certificates/chain.COTELO.crt.pem
```

Se activa el módulo ssl de apache2 y se reinicia el servicio para que los cambios se apliquen.

```
cotelo@httpserver:~$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
cotelo@httpserver:~$ sudo systemctl restart apache2
```

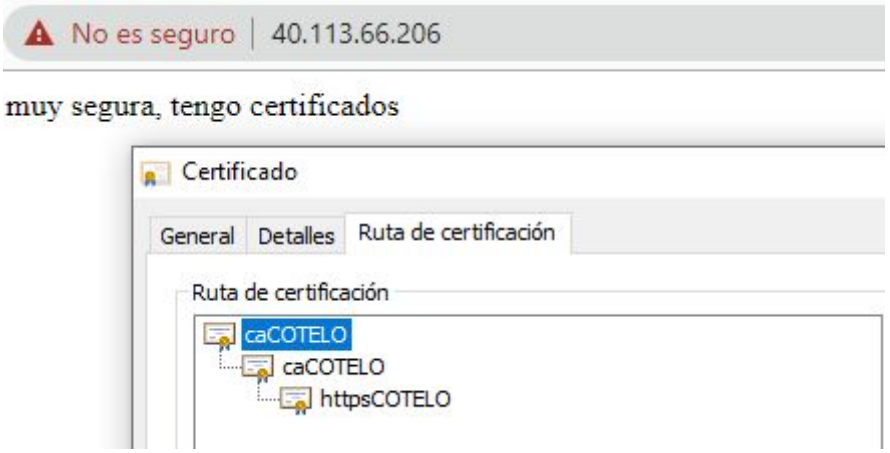
Se activa ssl en apache2 y se recarga el servicio

```
cotelo@httpserver:~$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
cotelo@httpserver:~$ sudo systemctl reload apache2
```

5.- Para finalizar

Si todo ha ido bien, el servidor apache2 va a enviar los certificados con las consultas https. No obstante, el navegador que se use no va a tener instalados los certificados de la PKI que se ha creado.

Para que el certificado sea válido en el navegador, habrá que instalarlo en los certificados de confianza del ordenador.



El certificado raíz puede encontrarse aqui: caCotelo