

# Práctica 2

## Seguridad Informática

Pedro Allué Tamargo (758267)  
Juan José Tambo Tambo (755742)

20 de octubre de 2020

### Índice

1. Tarea 1: Crear CA	1
2. Tarea 2: Crear certificado para <i>seginfo.es</i>	1
3. Tarea 3: Implementación de certificados en un servidor web <i>HTTPS</i>	1
4. Tarea 4: Implementación de certificados en un servidor web <i>HTTPS</i> basado en <i>Apache</i>	3
5. Tarea 5: Lanzando un ataque <i>MITM</i>	3
6. Tarea 6: Lanzando un ataque <i>MITM</i> con una CA comprometida	4

## 1. Tarea 1: Crear CA

Se va a proceder a crear una *CA* (*Certification Authority*). Para ello se va a crear un certificado y se va a firmar por nosotros mismos. Para ello se va a utilizar la herramienta `openssl`.

Lo primero será obtener una copia del fichero de configuración de `openssl` disponible en `/usr/lib/ssl/openssl.cnf`. Se va a modificar el fichero de configuración de tal forma que la variable `dir` almacene el valor `./ourCA`. Tras la modificación se van a crear los subdirectorios `certs`, `crl_dir`, `new_certs_dir` y los ficheros `database` y `serial`.

Ahora se podrá crear el *certificado auto-firmado* para la *CA*. Para ello se ejecutará el comando:

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

La opción `-x509` indica que el certificado es *auto-firmado*.

## 2. Tarea 2: Crear certificado para *seginfo.es*

Para crear un certificado para *seginfo.es* debemos generar un par de claves públicas/privadas. Para ello utilizaremos el siguiente comando:

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

El programa pedirá una contraseña para cifrar el fichero *server.key*. Ahora debe ser la entidad certificadora la que firme el certificado. Para ello se debe crear una solicitud de firma de certificado (*CSR*). Para generar el fichero *server.csr* se utilizará el comando:

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

Durante la creación de la petición de firma el programa pedirá datos como el *Common Name*. A este campo se le dará el valor de *seginfo.es*.

Una vez obtenido el fichero de petición de firma del certificado se pedirá a la *CA* que firme el certificado. Para ello la *CA* ejecutará el siguiente comando.

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

Ahora el fichero *server.crt* contendrá el certificado firmado por nuestra *CA* que demuestra su identidad frente a la entidad certificadora.

## 3. Tarea 3: Implementación de certificados en un servidor web *HTTPS*

Tras la generación del certificado para el servidor *seginfo.es* se va a proceder a implementar el certificado con un servidor web *HTTPS*. Para este paso se va a realizar una modificación sobre el *DNS* de la máquina para acceder al servidor

127.0.0.1	seginfo.es
-----------	------------

```
cp server.key server.pem
cat server.crt >> server.pem
```

```
openssl s_server -cert server.pem -www
# Si se quiere ejecutar el comando en
# segundo plano se utilizara:
# openssl s_server -cert server.pem -www \
# -pass pass:_password_ &
```

```

→ ↻ 🏠 🔒 https://192.168.56.200:4433
... 📄 🌟

s_server -cert server.pem -www
Secure Renegotiation IS supported

Ciphers supported in s_server binary

TLSv1.2 : ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 : ECDHE-RSA-AES256-GCM-SHA384
TLSv1.2 : DHE-RSA-AES256-GCM-SHA384 TLSv1.2 : ECDHE-ECDSA-CHACHA20-POLY1305
TLSv1.2 : ECDHE-RSA-CHACHA20-POLY1305 TLSv1.2 : DHE-RSA-CHACHA20-POLY1305
TLSv1.2 : ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2 : DHE-RSA-AES128-GCM-SHA256
TLSv1.2 : DHE-RSA-AES128-GCM-SHA256 TLSv1.2 : ECDHE-ECDSA-AES256-SHA384
TLSv1.2 : ECDHE-RSA-AES256-SHA384 TLSv1.2 : DHE-RSA-AES256-SHA256
TLSv1.2 : ECDHE-ECDSA-AES128-SHA256 TLSv1.2 : ECDHE-RSA-AES128-SHA256
TLSv1.2 : DHE-RSA-AES128-SHA256 TLSv1.0 : ECDHE-ECDSA-AES256-SHA
TLSv1.0 : ECDHE-RSA-AES256-SHA SSLv3 : DHE-RSA-AES256-SHA
TLSv1.0 : ECDHE-ECDSA-AES128-SHA TLSv1.0 : ECDHE-RSA-AES128-SHA
SSLv3 : DHE-RSA-AES128-SHA TLSv1.2 : RSA-PSK-AES256-GCM-SHA384
TLSv1.2 : DHE-PSK-AES256-GCM-SHA384 TLSv1.2 : RSA-PSK-CHACHA20-POLY1305
TLSv1.2 : DHE-PSK-CHACHA20-POLY1305 TLSv1.2 : ECDHE-PHK-CHACHA20-POLY1305
TLSv1.2 : AES256-GCM-SHA384 TLSv1.2 : PSK-AES256-GCM-SHA384
TLSv1.2 : PSK-CHACHA20-POLY1305 TLSv1.2 : RSA-PSK-AES128-GCM-SHA256
TLSv1.2 : DHE-PSK-AES128-GCM-SHA256 TLSv1.2 : AES128-GCM-SHA256
TLSv1.2 : PSK-AES128-GCM-SHA256 TLSv1.2 : AES256-SHA256
TLSv1.2 : AES128-SHA256 TLSv1.0 : ECDHE-PSK-AES256-CBC-SHA384
TLSv1.0 : ECDHE-PSK-AES256-CBC-SHA SSLv3 : SRP-RSA-AES256-CBC-SHA
SSLv3 : SRP-AES-256-CBC-SHA TLSv1.0 : RSA-PSK-AES256-CBC-SHA384
TLSv1.0 : DHE-PSK-AES256-CBC-SHA384 SSLv3 : RSA-PSK-AES256-CBC-SHA
SSLv3 : DHE-PSK-AES256-CBC-SHA SSLv3 : AES256-SHA
TLSv1.0 : PSK-AES256-CBC-SHA384 SSLv3 : PSK-AES256-CBC-SHA
TLSv1.0 : ECDHE-PSK-AES128-CBC-SHA256 TLSv1.0 : ECDHE-PSK-AES128-CBC-SHA
SSLv3 : SRP-RSA-AES128-CBC-SHA SSLv3 : SRP-AES128-CBC-SHA
SSLv3 : RSA-PSK-AES128-CBC-SHA256 TLSv1.0 : DHE-PSK-AES128-CBC-SHA256
SSLv3 : RSA-PSK-AES128-CBC-SHA SSLv3 : DHE-PSK-AES128-CBC-SHA
SSLv3 : AES128-SHA TLSv1.0 : PSK-AES128-CBC-SHA256
SSLv3 : PSK-AES128-CBC-SHA

---

Ciphers common between both SSL end points:
ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-ECDSA-CHACHA20-POLY1305
ECDHE-RSA-CHACHA20-POLY1305 ECDHE-AES256-GCM-SHA384 ECDHE-RSA-AES256-GCM-SHA384
ECDHE-ECDSA-AES256-SHA ECDHE-RSA-AES128-SHA
ECDHE-RSA-AES256-SHA AES128-GCM-SHA256 AES256-GCM-SHA384
AES128-SHA AES256-SHA

Signature Algorithms: ECDSA-SHA256:ECDSA-SHA384:ECDSA-SHA512:0x04+0x08:0x05+0x08:0x06+0x08:RSA-SHA256:RSA-SHA384:RSA-SHA512:ECDSA-SHA1:RSA-SHA1
Shared Signature Algorithms: ECDSA-SHA256:ECDSA-SHA384:ECDSA-SHA512:RSA-SHA256:RSA-SHA384:RSA-SHA512:ECDSA-SHA1:RSA-SHA1
Supported Elliptic Curves: X25519:P-256:P-384:P-521:0x010010x0101

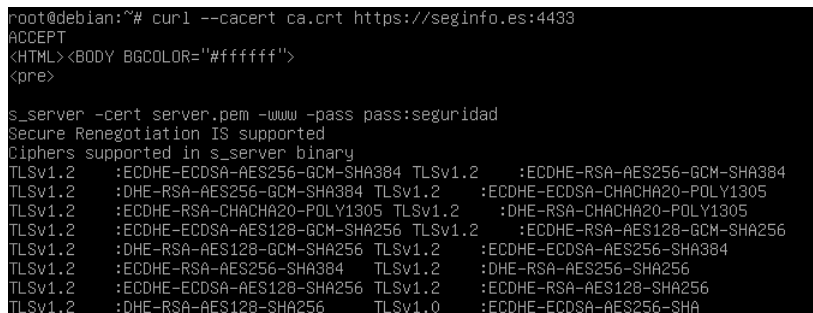
```

Si se intenta acceder utilizando *curl* se obtendrá un error (Figura 1.10). También por favor mete la figura que no tengo la MV) ya que no se puede establecer la veracidad del servidor.

2

```
curl --cacert ca.crt https://seginfo.es:4433
```

Tras esto se observaría que el resultado es el mostrado en la Figura 2.



```
root@debian:~# curl --cacert ca.crt https://seginfo.es:4433
ACCEPT
<HTML><BODY BGCOLOR="#ffffff">
<pre>

s_server -cert server.pem -www -pass pass:seguridad
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1.2 :ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 :ECDHE-RSA-AES256-GCM-SHA384
TLSv1.2 :DHE-RSA-AES256-GCM-SHA384 TLSv1.2 :ECDHE-ECDSA-CHACHA20-POLY1305
TLSv1.2 :ECDHE-RSA-CHACHA20-POLY1305 TLSv1.2 :DHE-RSA-CHACHA20-POLY1305
TLSv1.2 :ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2 :ECDHE-RSA-AES128-GCM-SHA256
TLSv1.2 :DHE-RSA-AES128-GCM-SHA256 TLSv1.2 :ECDHE-ECDSA-AES256-SHA384
TLSv1.2 :ECDHE-RSA-AES256-SHA384 TLSv1.2 :DHE-RSA-AES256-SHA256
TLSv1.2 :ECDHE-ECDSA-AES128-SHA256 TLSv1.2 :ECDHE-RSA-AES128-SHA256
TLSv1.2 :DHE-RSA-AES128-SHA256 TLSv1.0 :ECDHE-ECDSA-AES256-SHA
```

Figura 2: Captura de pantalla del acceso desde curl con certificado de CA

¿Qué pasaría si modificásemos un solo byte del fichero *server.pem*? Depende. Si se modifica un byte de la zona *Certificate* el servidor no se inicia (**METER CAPTURA DE PANTALLA**). Si se modifica un byte de la zona *signature-algorithm* el servidor se inicia sin problemas (**METER CAPTURA DE PANTALLA PERO ME EXTRAÑA QUE NO DE NINGÚN ERROR**).

¿Qué ocurre si nos intentamos conectar mediante `https://localhost:4433`? Que no podremos conectar ya que el certificado ha sido emitido para *seginfo.es* y no para *localhost*.

#### 4. Tarea 4: Implementación de certificados en un servidor web *HTTPS* basado en *Apache*

Configurar el servidor Apache2. Modificar ficheros 000-default y default-ssl. Crear directorio `/var/www/seginfo.es` y fichero `index.html` (y poner el contenido que le hemos metido).

Hacer los comandos de apache2. Introducir el passphrase de la clave al reiniciar. Hacer comando `curl --cacert ca.crt https://seginfo.es` y decir que sale todo bien.

Hay capturas de pantalla

#### 5. Tarea 5: Lanzando un ataque *MITM*

Explicar cual es la página a suplantar (`example.com`).

Crear certificado para el sitio que queremos suplantar.

Para lanzar un MITM hay que modificar el DNS del host. Explicar como lo has hecho en Windows.

Utilizando el navegador web del host ir a `example.com`.

Añadimos el certificado al navegador (firefox) y volvemos a entrar.

Hay capturas de pantalla

## 6. Tarea 6: Lanzando un ataque *MITM* con una CA comprometida

Aquí ya no tengo nada apuntado pero yo diría que el atacante ha obtenido la clave privada de la CA y puede firmar lo que quiera. Va a suplantar webs mediante ataque a DNS y ha suplantado la página de [www.pcbox.com](http://www.pcbox.com). Explicar un poco qué ha hecho apoyándose en las tareas 4 y 5.

Hay captura de pantalla