En modo *root*, crear el directorio `/var/log/apache2`

Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

SEGURIDAD EN REDES

Práctica 4: Configuración de un portal seguro

Autores: Josemaría Malgosa Sanahuja
María Dolores Cano Baños

SEGURIDAD EN REDES

Configuración de un portal seguro PRÁCTICA 4

El protocolo *ssl* permite establecer conexiones (*sockets*) TCP seguras. Su funcionamiento es muy parecido al del protocolo *ssh* aunque es totalmente independiente (*ssl* trabaja a nivel TCP mientras que *ssh* opera en el nivel de aplicación). Con *ssl* es posible dotar de comunicaciones seguras a la arquitectura cliente-servidor tradicional.

El intercambio de información (resumido) entre el cliente y el servidor sigue los siguientes pasos:

(a) El cliente inicia la conexión SSL con el servidor. Los puertos TCP-*ssl* más utilizados son el 443 (https), 465 (SMTP), 993 (imap), 995 (pop), etc.

(b) El servidor envía al cliente su certificado. Opcionalmente, pide el certificado del cliente

(c) El cliente verifica que el certificado ha sido emitido por una Autoridad de Certificación (CA) de confianza. Para ello, dispone de una base de datos con los certificados de las CAs más conocidas. Si la verificación no es posible, el usuario debe añadir el certificado de la CA utilizada por el servidor manualmente

(d) El cliente usa la clave pública del servidor para cifrar la clave de la sesión y se la manda al servidor. Si en el paso (b) el servidor ha pedido el certificado del cliente, éste debe mandárselo en este momento. Una vez recibido, el servidor comprobará que ha sido emitido por una CA de confianza (de lo contrario lo rechazará)

(e) Finalmente, se negocia el cifrado simétrico a utilizar y si es necesario, se intercambian claves simétricas nuevas.

De entre todos los modelos basados en la arquitectura cliente-servidor, el más conocido y utilizado es el de un portal web. Para crear un portal web seguro (*https*) es necesario adquirir -previo pago a una CA- un certificado de componentes. Por lo tanto, en esta práctica deberemos (1) erigirnos nosotros mismo como una CA válida, creando nuestro propio certificado de CA y (2) crear un certificado para el servidor y firmarlo con la clave privada de nuestra CA.

CREACIÓN DE UNA AUTORIDAD CERTIFICADORA (CA)

1. Crear el directorio *~/pr4* y situarse en él. Crear la clave privada RSA de la CA (*ca_priv.pem*)

```
alumno@localhost:~/pr4> openssl genrsa -out CA-priv.pem
```

2. Visualizar todos los campos de la clave privada ¿Contiene la clave generada toda la información del RSA (en definitiva, el par de claves privada-pública) o solo la relativa con la clave privada?

   Contiene todos los numeros necesarios tanto para deducir la clave privada como la pública

3. Ahora se debe generar el certificado de la CA firmado por la propia CA (autofirmado). Para ello, ejecutar el siguiente comando:

```
openssl req -new -x509 -days 1461 -key ca_priv.pem -out ca.crt
```

   Responder a las preguntas (opción –new) tal y como se muestra a continuación:

```
Country Name (2 letter code) [AU]:SP
State or Province Name (full name) [Some-State]:Murcia
Locality Name (eg, city) []:Cartagena
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UPCT
Organizational Unit Name (eg, section) []:grado IT
Common Name (e.g. server FQDN or YOUR name) []:CA-ser
Email Address []:ca@upct.es
```

4. Visualizar el certificado en modo texto y verificar que todos los campos se han establecido correctamente

```
alumno@localhost:~/pr4> openssl x509 -in CA.crt -text -noout
```

5. Ahora debe crearse un archivo donde se almacena el número de serie que la CA utilizará para crear y firmar certificados nuevos (*This file consist of one line containing an even number of hex digits with the serial number to use. After each use the serial number is incremented and written out to the file again. The default filename consists of the CA certificate file base name with ".srl" appended. For example if the CA certificate file is called "mycacert.pem" it expects to find a serial number file called "mycacert.srl"*)

```
echo 01 > ca.srl
```

Ahora nuestra CA ya está preparada para generar cualquier certificado. De momento, el único certificado que tenemos que generar es el del servidor web. Para ello, primero deberemos crear una petición de certificado (*certificate signing request, csr*) y posteriormente, el propio certificado firmado por nuestra CA.

6. Generar la clave privada del servidor apache (*srv_priv.pem*)

```
alumno@localhost:~/pr4> openssl genrsa -out srv_priv.pem
```

7. ¿Por qué es necesario generar una petición de certificado antes de generar el propio certificado?

Un certificado 509 tiene una clave pública, una clave publica tiene asociada una privada. Si la FNMT te hace un certificado de 0 tendrían que hacer una clave publica y una privada y después eliminarla. En vez de eso, las claves las generamos nosotros y enviamos la clave pública a la CA para que la certifique, de esta forma garantizamos que la clave privada solo la conocemos nosotros

8. Para generar la petición de certificado (*srv.csr*) ejecutar el siguiente comando:

```
openssl req -new -key srv_priv.pem -out srv.csr
```

Responder a las preguntas (opción –new) tal y como se muestra a continuación:

```
Country Name (2 letter code) [AU]:SP
State or Province Name (full name) [Some-State]:Murcia
Locality Name (eg, city) []:Cartagena
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UPCT
Organizational Unit Name (eg, section) []:grado IT
Common Name (e.g. server FQDN or YOUR name) []:ser_git.upct.es
Email Address []:webmaster@upct.es
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

9. Visualizar la petición de certificado en modo texto y verificar que todos los campos se han establecido correctamente ¿Contiene las claves privada y pública o solo la pública?
   Solo la pública
10. Finalmente, ya se puede proceder a crear el certificado del servidor ejecutando el siguiente comando:

```
openssl x509 -req -days 730 -in srv.csr -out srv.crt -CA ca.crt -CAserial ca.srl
-CAkey ca_priv.pem
```

11. Verificar que el número de serie almacenado en el archivo *ca.srl* se ha incrementado en una unidad Incrementa en una unidad

CONFIGURACIÓN DEL SERVIDOR APACHE

12. En modo root, copiar los siguientes archivos en los directorios de configuración de *ssl* de apache:

    srv.crt al directorio */etc/apache2/ssl.crt/*
    srv_priv.pem al directorio */etc/apache2/ssl.key/*

13. En modo root, crear en el directorio */srv/www/htdocs/ssl/* el fichero *index.html* que contenga la frase "*zona segura*".

14. En modo *root*, verificar que apache tiene el módulo SSL activado (`apache2ctl -M`). De no ser así, activarlo (`a2enmod ssl_module`). Verificar también que apache arranca con la opción SSL activada (variable `APACHE_SERVER_FLAGS="SSL"` en el archivo `/etc/sysconfig/apache2`). De no ser así, activarla. Finalmente, arrancar el servidor apache (`systemctl start apache2.service`).

15. Mediante un navegador, acceder a http://ser_git.upct.es y posteriormente a https://ser_git.upct.es ¿Funciona? No, salta una excepción

16. Eliminar la excepción de seguridad del navegador e incorporar el certificado de nuestra CA en el navegador y acceder de nuevo a https://ser_git.upct.es ¿Funciona mejor?

17. Verificar que el servidor apache escucha peticiones por el puerto 443 (`/etc/apache2/listen.conf`)

18. Verificar que la configuración SSL del servidor apache es correcta (`/etc/apache2/vhost/vhost-ssl.conf`):

```
DocumentRoot /srv/www/htdocs/ssl
SSLEngine on
SSLCertificateFile    /etc/apache2/ssl.crt/ srv.crt
SSLCertificateKeyFile /etc/apache2/ssl.key/ srv_priv.pem
```

```
<IfDefine SSL>
        <IfDefine !NOSSL>
        <IfModule mod_ssl.c>

                Listen 443

        </IfModule>
        </IfDefine>
</IfDefine>
```

GENERACIÓN DE CERTIFICADOS

19. Actuando como CA, genere su propia clave privada y su correspondiente certificado (*apellido_priv.pem*, *apellido.crt*). Guarde toda esta información en ~/pr4

20. El formato PKCS12 permite juntar en un mismo archivo el certificado X.509 (con la clave pública) y la clave privada cifrada mediante un algoritmo simétrico. Este formato es el que acostumbran a utilizar las CA para enviar a los usuarios los certificados solicitados. Utilizando *openssl*, genere un archivo PKCS12 (*apellido.p12*) a partir de los ficheros *apellido_priv.pem*, *apellido.crt*

```
alumno@localhost:~/pr4> openssl pkcs12 -export -out apellido.p12 -inkey apellido_priv.pem -in apellido.crt
```