Apache con SSL

http://www.kriptopolis.org/apache-con-ssl

Esta guía fue desarrollada por Kriptopolis

Textos bajo licencia Creative Commons

Como dicen que los consejos salen gratis y que siempre es mejor enseñar a pescar que regalar una caña, he dedicado de nuevo un rato a escribir una breve nota sobre cómo se añade a un Apache 2 la posibilidad de conexión cifrada con SSL. Así, quien quiera empezar a montarse en casa un pequeño "laboratorio de pruebas" ya tiene por donde empezar...

Los pasos que se detallan a continuación han sido realizados en Arch Linux 0.8 (Voodoo), con Apache 2.2.3-2 y OpenSSL 0.9.8d-1. No creo que hoy en día sea difícil instalar ambas cosas en cualquier distribución, y son los únicos requisitos previos. Tampoco creo que sea muy diferente en Windows.

Insisto: el artículo está actualizado y los resultados comprobados -en las condiciones antedichas- a día de hoy, algo que no suele ocurrir en los cientos de tutoriales al respecto que corren por Internet. De hecho, este minitutorial quedará defasado en pocos meses, a nada que evolucionen navegadores, servidores y demás, pero ahora mismo su frescura es su mayor ventaja. Por tanto, el momento ideal para ponerse manos a la obra es **ahora mismo**.

Debe quedar también claro que no se pretende enseñar a instalar un servidor seguro plenamente operativo y asegurado en un entorno de producción, sino solamente un arreglo rápido para utilizar en un localhost casero. Quien afronte metas más comprometidas habrá de investigar un poco por su cuenta.

Sin más, intentaremos lograr el objetivo en muy pocos pasos, descritos -espero- de la forma más sencilla posible.

1. Generar CLAVE PRIVADA:

Llamaremos a nuestro servidor p.ej. ciberia (sustituye "ciberia" por el nombre-url del tuyo). Para generar una clave privada RSA protegida por contraseña:

```
$ openssl genrsa -des3 > ciberia.key
```

La protección de la clave por contraseña (para lo que puedes utilizar también AES) presenta el inconveniente de que tendrás que teclear esta contraseña cada vez que arranques el servidor y que si la olvidas se acabó.

Por eso, si prefieres la comodidad a la máxima seguridad (por ejemplo, se trata de tu servidor de pruebas) y estás "seguro" de que la clave no va a caer en malas manos (o no importa), puedes optar por no proteger con contraseña:

```
$ openssl genrsa > ciberia.key
```

2. Generar CSR (CERTIFICATE SIGNING REQUEST):

Ahora generamos una petición para enviar a una Autoridad de Certificación y solicitar que firme nuestra clave y nos devuelva un certificado.

```
$ openssl req -new -key ciberia.key > ciberia.csr
```

Responde a las preguntas sobre país, provincia. ciudad, empresa, nombre y e-mail.

El fichero generado podrías enviarlo a una autoridad de certificación (como Verisign, de pago, o CACert, gratuita aunque no del agrado de Microsoft) para que lo firmaran y te enviaran tu certificado, pero haremos algo aún mejor: crearemos nuestra propia autoridad de certificación.

3. Crear nuestra propia AUTORIDAD DE CERTIFICACIÓN (CA):

```
$ /etc/ssl/misc/CA.pl -newca
```

(ojo: el path a CA.pl puede ser distinto en tu distribución).

Presiona Enter y propociona la que será contraseña de acceso a la clave privada de esta nueva autoridad. Proporciona país, provincia, ciudad, etc. para esta autoridad.

Enhorabuena. Ya eres el feliz propietario de una flamante "Autoridad de Certificación", contenida en el nuevo directorio demoCA. El fichero cacert.pem contiene su clave pública, mientras que en private/cakey.pem dispones de la clave privada que te permitirá crear certificados para tus "clientes".

4. CREAR CERTIFICADOS:

Como ejemplo, empecemos por firmar el CSR que generaste en el paso 2.

```
$ openssl ca -policy policy_anything -out ciberia.crt -infiles
ciberia.csr
```

Introduce la contraseña de tu autoridad cuando se te pida. Pulsa 'Y' cuandos se te pregunte si deseas firmar e 'Y' también para ejecutar (commit) el cambio en tu base de datos.

Vamos a generar también un certificado autofirmado para tu propia autoridad certificadora (en el apartado 6 verás para qué). Sitúate en el directorio de tu autoridad (demoCA) y teclea:

```
$ openssl x509 -req -days 365 -in careq.pem -out cacert.crt -signkey
./private/cakey.pem
```

5. AÑADIR CERTIFICADO AL SERVIDOR WEB (Apache 2):

```
cp ciberia.key /etc/httpd/conf
cp ciberia.crt /etc/httpd/conf
```

Editar /etc/httpd/conf/httpd.conf para descomentar la línea que permite incluir httpd-ssl.conf

Editar /etc/httpd/conf/extra/httpd-ssl.conf para añadir el nombre, email del administrador y puerto de nuestro servidor y editar las líneas de clave y certificado para que digan así:

```
SSLCertificateFile /etc/httpd/conf/ciberia.crt
SSLCertificateKeyFile /etc/httpd/conf/ciberia.key
```

Sólo resta arrancar de nuevo el servidor y conectarse a https://ciberia (o el nombre de tu servidor) para comprobar que funciona.

6. NAVEGADORES WEB:

Si ejecutaste los pasos anteriores (y todo ha ido bien) verás que al acceder por primera vez a tu servidor "seguro" el navegador duda de la autenticidad de tu flamante certificado, porque en última instancia duda de tu autoridad certificadora.

La solución consiste en colocar el certificado de tu autoridad (cacert.crt) en algún sitio del servidor, donde pueda ser accedido -o descargado- por los navegadores de tus visitantes. En el caso de Opera y Firefox, al pulsar el enlace el navegador preguntará si aceptas esa nueva autoridad para certificar sitios web, a lo que has de responder (obviamente) 'Sí'. En Explorer has de importar el certificado siguiendo los pasos que te propone el asistente del propio navegador.

Autenticación de usuarios mediante certificados

http://www.kriptopolis.org/autenticacion-de-usuarios-mediante-certificados

Suele ser cada vez más normal que se nos requiera a los administradores de sistemas una forma alternativa para acceder a los correos corporativos desde cualquier lugar del mundo, además de usando el cliente de correo. Normalmente esta opción suele ser usar algún tipo de webmail.

Bajo mi punto de vista ésta es sin duda una forma muy cómoda de acceder al correo electrónico, y al mismo tiempo un peligro en potencia. Los usuarios y muchos administradores de sistemas suelen dormir tranquilos cuando instalan este servicio usando https y protegido con un login y un password para cada usuario que tenga cuenta en el servidor. Sin embargo, en mi caso, son numerosos los dolores de barrigua que me produce el pensar que cualquiera desde un ciberbar puede ver los correos de la gente a la que se supone que protejo con el simple hecho de saber la clave del usuario...

Donde actualmente trabajo, existe una gran cantidad de consultores que viajan alrededor de todo el mundo y usan en ocasiones el webmail desde algún ordenador que les proporciona el cliente al que van a prestar la consultoría. Es muy fácil que a un cliente poco avanzado se le ocurra echar un ojo por encima del hombro de mi consultor cuando va a teclear la clave, para luego poder usarla él tranquilamente desde su casa. Un cliente un poco más avanzado podría incluso instalar un keylogger en esa máquina, con lo que el trabajo sería aún más fácil. Entonces... ¿cuál es la solución? La solución pasa por mejorar el sistema de autentificación. Hay que autentificarse no sólo por una clave que sabes, sino que también debes usar un certificado que tienes. Probablemente en tu pendrive, que debes guardar como si fuese tu tarjeta de crédito.

A continuación expongo, paso a paso, cómo instalar un servidor Apache y cómo generar los certificados para los clientes.

En este caso, nosotros mismos seremos la autoridad certificadora (CA) que firmará los certificados del cliente, autoridad que hay que guardar de forma segura y sólo poder acceder a ella los administradores de sistemas. Espero que este mini-manual os sirva de ayuda. Tened en cuenta que tendréis que adaptar los comandos ejecutados para que se ajusten a vuestra estructura de directorio. Esta instalación se ha hecho en una distribución Debian, pero sirve para cualquiera. Antes de instalar Apache tenéis que tener instalados openssl y wget.

Pasos a seguir:

Paso 1: Descargar el servidor web de un lugar de confianza.

wget http://apache.gva.es/httpd/httpd-2.0.55.tar.gz

Para comprobar que es el ejecutable que queremos nos bajamos el md5 y la firma.

wget http://www.apache.org/dist/httpd/httpd-2.0.55.tar.gz.md5
wget http://www.apache.org/dist/httpd/httpd-2.0.55.tar.gz.asc

Para comprobar la firma hay que bajar previamente las claves públicas. Los fuentes te los puedes bajar de la web principal o desde un mirror, pero las claves públicas y la firma debes bajártelas desde el sitio oficial, para estar seguro de que nadie las manipuló.

http://www.apache.org/dist/httpd/KEYS

Usando gnupg importamos las claves y luego comprobamos la firma:

```
        gpg --import KEYS

        gpg --verify httpd-2.0.55.tar.gz.asc

        jgomez@IT:~/apache-2.2-test$ gpg --verify httpd-
2.0.55.tar.gz.asc
        gpg: Signature made Mon 10 Oct 2005 03:35:15 AM CEST using RSA
key ID 10FDE075
        gpg: Good signature from "wrowe@covalent.net"
        gpg:                  aka "William A. Rowe, Jr. "
        gpg:                  aka "wrowe@lnd.com"
        gpg:                  aka "wrowe@apache.org"
        gpg: WARNING: This key is not certified with a trusted
signature!
        gpg:          There is no indication that the signature
belongs to the owner.
        Primary key fingerprint: 33 16 9B 46 FC 12 D4 01  CA 6D DB D7
DE EA 4F D7
```

Comprobamos que la firma es correcta. El mensaje que aparece abajo, diciendo que no podemos estar seguros que esa clave sea de esa persona, se debe a que nosotros no la hemos firmado.

Una vez llegados aquí, tenemos nuestros fuentes en nuestro ordenador y además sabemos que nadie los ha manipulado.

Paso 2: Compilación de Apache

Descomprimimos los fuentes:

```
        tar -zxvf httpd-2.0.55.tar.gz
        cd httpd-2.0.55
```

Tenemos que activar mod_ssl y mod_setenvif. Este último módulo es necesario para mantener compatibilidad con algunos navegadores como Internet Explorer.

Para compilar Apache vamos a pasar los siguientes parámetros que necesitamos al configure. Si necesitas tener soporte para más cosas es mejor ir a la documentación oficial de Apache en http://www.apache.org y ver qué módulos debes activar. Así, si quieres soporte para mysql necesitar poner --with-mysql

Yo voy a instalar Apache en un directorio dentro de mi HOME. Utilicé esta linea de configuración:

```
        ./configure --prefix=/home/jgomez/apache-installation --with-
mpm=prefork --enable-ssl --enable-setenvif
```

```
make

make install
```

Y ya tenemos nuestro Apache instalado con soporte para SSL.

Si te vas al directorio donde lo instalastes (que en este caso es /home/jgomez/apache-installation) y luego ejecutas ./bin/apachectl start ya tendrás tu servidor funcionando. Puedes conectarte desde esa misma máquina con un navegador escribiendo http://localhost y verás la página de bienvenida de Apache.

Paso 3: Configuración Apache mod_ssl

Ahora vamos a hacer que escuche en el puerto 443 para conexiones seguras. Es decir, para el protocolo https.

Añadimos lo siguiente al fichero conf/httpd.conf:

```
Listen 443
SSLEngine on
SSLOptions +StrictRequire

SSLProtocol -all +TLSv1 +SSLv3
SSLCipherSuite HIGH:MEDIUM:!aNULL:+SHA1:+MD5:+HIGH:+MEDIUM

SSLMutex file:/home/jgomez/apache-installation/logs/ssl_mutex

SSLRandomSeed startup file:/dev/urandom 1024
SSLRandomSeed connect file:/dev/urandom 1024

SSLSessionCache shm:/home/jgomez/apache-
installation/logs/ssl_cache_shm
SSLSessionCacheTimeout 600

SSLPassPhraseDialog builtin
SSLCertificateFile /home/jgomez/apache-
installation/conf/ssl.crt/server.crt
SSLCertificateKeyFile /home/jgomez/apache-
installation/conf/ssl.key/server.key

SSLVerifyClient none
SSLProxyEngine off

        AddType application/x-x509-ca-cert      .crt
        AddType application/x-pkcs7-crl         .crl

SetEnvIf User-Agent ".*MSIE.*" \
  nokeepalive ssl-unclean-shutdown \
  downgrade-1.0 force-response-1.0
```

NOTA: En el fichero de configuracion del apache, esa linea anterior debe ir sin los \.

Antes de arrancar de nuevo nuestro Apache tendremos que generar los certificados y para eso hacemos:

```
umask 022
```

```
        IT:/home/jgomez/apache-installation# mkdir conf/ssl.key
        IT:/home/jgomez/apache-installation# mkdir conf/ssl.crt
        IT:/home/jgomez/apache-installation# mkdir conf/ssl.crl
```

Ahora creamos nuestros certificados, que estarán firmados por nosotros mismos. Se supone que nos los tendria que dar alguna empresa tipo VeriSign, pero para lo que nosotros queremos, nos bastamos y nos sobramos.

```
        openssl req -new -x509 -days 30 -keyout /home/jgomez/apache-
installation/conf/ssl.key/server.key -out /home/jgomez/apache-
installation/conf/ssl.crt/server.crt -subj '/CN=JmGV-Test-Certificate'
```

(Para más información sobre la sintaxis ir a www.openssl.org)

```
        T:/home/jgomez/apache-installation# openssl req -new -x509 -
days 30 -keyout /home/jgomez/apache-
installation/conf/ssl.key/server.key -out /home/jgomez/apache-
installation/conf/ssl.crt/server.crt -subj '/CN=Kriptopolis-Test-
Certificate'
        Generating a 1024 bit RSA private key
        ..++++++
        ....................................++++++
        writing new private key to '/home/jgomez/apache-
installation/conf/ssl.key/server.key'
        Enter PEM pass phrase:
        Verifying - Enter PEM pass phrase:
```

Desde ahora cuando arranquéis Apache os pedirá la clave. En realidad es configurable, pero por defecto os lo pedirá.

Una vez arrancado ya podéis conectaros poniendo desde el mismo ordenador, en un navegador https://localhost

Os saldrán los datos del certificado.

Si es así es que vamos bien.

Ahora vamos a cambiar la configuración, para que cada cliente que se vaya a conectar a nuestro web tenga que tener un certificado instalado en su navegador.

Paso 4: Hacemos que el servidor nos pida un certificado

Para eso, cambiamos configuración en Apache cambiando los siguientes valores:

```
SSLVerifyClient require
SSLVerifyDepth 1
```

Ese valor a 1 significa el numero máximo de intermediarios que aceptamos en el certificado. Si lo ponemos a 1 significa que todos los certificados tienen que estar firmados directamente por el CA que indiquemos. En este caso somos nosotros mismos.

Una vez que hagamos esos cambios y volvamos a rearrancar el servidor, veremos que ya no nos
podemos conectar, y eso se debe a que no tenemos instalado ningún certificado en

nuestro
navegador; por lo tanto no tiene éxito la negociación y nos echa.

Tenemos que indicar Apache cuál es la entidad certificadora. En nuestro caso está en /home/jgomez/apache-installation/conf/ssl.crt/server.crt

Para ello, en httpd.conf añadimos:

```
SSLCACertificateFile /home/jgomez/apache-
installation/conf/ssl.crt/server.crt
```

Reiniciamos Apache para ver que va todo bien.

Paso 5: Generación de certificados para clientes

Generación de certificados para la gente que queramos que vea nuestra web.

La creación de certificados para el cliente es muy similar a la que seguimos para la generación de certificados para el servidor.

Mediante este comando creamos una clave cliente:

```
        openssl req -new -sha1 -newkey rsa:1024 -nodes -keyout
client.key -out request.pem -subj '/O=Secure/OU=LoGGiK/CN=JmGV'
```

Eso genera dos ficheros, client.key y request.pem (éste debe ser firmado por la CA, en este caso nosotros). La CA se supone que debe de asegurarse que la información es real antes de firmar nada.

Ahora toca firmarla. Te puedes poner a configurar el fichero openssl.cnf, pero yo lo que hice fue copiar el directorio /usr/lib/ssl/misc/demoCA a mi directorio CA, con lo que me quedó la siguiente estructura:

```
jgomez@IT:~$ tree CA
   CA
   |-- demoCA
   |   |-- cacert.pem
   |   |-- careq.pem
   |   |-- certs
   |   |-- crl
   |   |-- index.txt
   |   |-- index.txt.attr
   |   |-- index.txt.old
   |   |-- newcerts
   |   |   `-- 91213828FA1E8EA9.pem
   |   |-- private
   |   |   `-- cakey.pem
   |   `-- serial
   |-- requests
   |   `-- request.pem
   `-- signed
       `-- signed.pem
```

Una vez ejecutado:

```
        openssl ca -config /usr/lib/ssl/openssl.cnf -keyfile
/home/jgomez/apache-installation/conf/ssl.key/server.key -cert
/home/jgomez/apache-installation/conf/ssl.crt/server.crt -policy
policy_anything -out /home/jgomez/CA/signed/signed.pem -infiles
/home/jgomez/CA/requests/request.pem
```

Tenemos el fichero signed.pem listo para enviar al cliente.

El cliente por tanto, una vez recibido ese fichero, tendrá en un directorio los siguientes ficheros:

client.key request.pem signed.pem

Ahora tiene que guardarlo todo en un certificado con formato PKCS#12 y para eso hace:

```
        openssl pkcs12 -export -clcerts -in signed.pem -inkey
client.key -out client.p12
```

Y ya para finalizar hay que cargar el certificado client.p12 en el navegador. Cada navegador tendrá un procedimiento para hacer esto.

Próximamente realizaré otra serie de artículos donde se expliquen otros métodos que tienen aún más ventajas, como copias de certificados en el servidor, revocación de certificados, etc.

- Estoy absolutamente seguro de que la seguridad absoluta no existe -