

Instalación de un certificado digital en Apache

En esta guía, cubriremos cómo crear un **certificado SSL auto-firmado** para Apache en un servidor Linux, que nos permitirá cifrar el tráfico a nuestro servidor. Si bien esto no proporciona el beneficio de la validación por parte de terceros de la identidad del servidor, cumple los requisitos de aquellos que simplemente quieren transferir información de forma segura.

Paso Uno — Activar el Módulo SSL

El soporte para SSL viene de serie en el paquete. Simplemente necesitamos habilitarlo para aprovechar SSL en nuestro sistema.

Activar el módulo escribiendo:

```
sudo a2enmod ssl
```

Una vez activado SSL, hemos de reiniciar el servidor para que el cambio sea aplicado:

```
sudo service apache2 restart
```

Con esto, nuestro servidor web es ahora capaz de tratar SSL, si lo configuramos para hacerlo.

Paso Dos — Crear un Certificado SSL Auto-firmado

Comenzamos creando un subdirectorio dentro de la jerarquía de configuración de Apache para colocar allí los archivos de certificado que hagamos:

```
sudo mkdir /etc/apache2/ssl
```

Ahora que tenemos una ubicación para colocar nuestra clave y certificado, podemos crearlos en un solo paso escribiendo:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key
```

Veamos qué significa este comando realmente:

- **openssl**: Esta es la herramienta de línea de comandos básica proporcionada por OpenSSL para crear y administrar certificados, claves, solicitudes de firma, etc.

- **req:** Esto especifica un subcomando para la gestión de solicitudes de firma de certificados X.509 (CSR) . X.509 es un estándar de infraestructura de clave pública al que SSL se adhiere para su gestión de claves y certificados. Ya que estamos deseando *crear* un nuevo certificado X.509, esto es lo que queremos.
- **-x509:** Esta opción especifica que queremos crear un archivo de certificado autofirmado en lugar de generar una solicitud de certificado.
- **-nodes:** Esta opción indica a OpenSSL que no deseamos proteger nuestro archivo de clave con una frase de contraseña. Tener un archivo de clave protegida por contraseña impediría que Apache arrancara automáticamente, ya que tendríamos que introducir la contraseña cada vez que el servicio se reiniciase.
- **-days 365:** Esto especifica que el certificado que estamos creando será válido por un año.
- **-newkey rsa:2048:**Esta opción creará la solicitud de certificado y una nueva clave privada al mismo tiempo. Esto es necesario ya que no creamos una clave privada por adelantado. El **rsa: 2048** le dice a OpenSSL que genere una clave RSA de 2048 bits.
- **-keyout:** Este parámetro nombra el archivo de salida para el archivo de clave privada que se está creando.
- **-out:** Esta opción nombra el archivo de salida para el certificado que estamos generando.

Cuando pulsemos **Enter**, se nos hará una serie de preguntas.

El elemento más importante que se solicita es la línea que dice **Common Name** (e.g. **server FQDN or YOUR name**). Debemos introducir el nombre de dominio que deseamos asociar con el certificado.

La sección de preguntas se parece a esto:

```
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Castellon
Locality Name (eg, city) []:Castellon de la Plana
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Caminas Web Projects
Organizational Unit Name (eg, section) []:Departament of Server Development
Common Name (e.g. server FQDN or YOUR name) []:www.midomimioseguero.com
Email Address []:your_email@midomimioseguero.com
```

La clave y el certificado serán creados y colocados en el directorio `/etc/apache2/ssl`.

```
root@pedrablava:/etc/apache2/ssl# ls -la
total 16
drwxr-xr-x 2 root root 4096 oct 11 17:38 .
drwxr-xr-x 9 root root 4096 oct 11 17:36 ..
-rw-r--r-- 1 root root 1493 oct 11 17:38 apache.crt
-rw-r--r-- 1 root root 1704 oct 11 17:38 apache.key
```

Paso Tres — Configurar Apache para que use SSL

Una vez tenemos nuestro certificado y clave disponible, podemos configurar Apache para usar estos ficheros en nuestro host virtual.

En vez de basar nuestro fichero de configuración en `000-default.conf`, usaremos el fichero `default-ssl.conf` que contiene la configuración mínima de SSL.

Hay que abrir el archivo con privilegios de root (como siempre que tocamos ficheros de configuración importantes, hacemos un backup antes de modificarlos)

```
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Con los comentarios quitados, el fichero se parece a esto:

```
<IfModule mod_ssl.c>
    <VirtualHost _default_:443>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        SSLEngine on
        SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        <FilesMatch "\.(cgi|shtml|phtml|php)$">
            SSLOptions +StdEnvVars
        </FilesMatch>
        <Directory /usr/lib/cgi-bin>
            SSLOptions +StdEnvVars
        </Directory>
        BrowserMatch "MSIE [2-6]" \
            nokeepalive ssl-unclean-shutdown \
            downgrade-1.0 force-response-1.0
        BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
    </VirtualHost>
</IfModule>
```

Esto parece un poco complicado, pero con suerte, no nos tenemos que preocupar de la mayoría de opciones.

Sólo queremos fijar las cosas normales que hemos configurado para el resto de host (ServerAdmin, ServerName, ServerAlias, DocumentRoot, etc.) así como cambiar la localización donde Apache busca el certificado y la clave SSL.

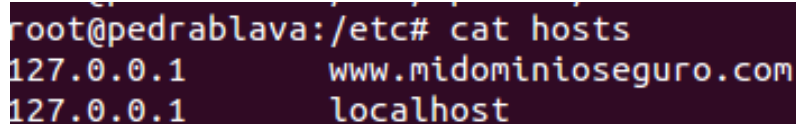
Al final, parecerá algo así.

```
<IfModule mod_ssl.c>
  <VirtualHost _default_:443>
    ServerAdmin admin@midominioseguro.com
    ServerName www.midominioseguro.com
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/apache.crt
    SSLCertificateKeyFile /etc/apache2/ssl/apache.key
    <FilesMatch "\.(cgi|shtml|phtml|php)$">
      SSLOptions +StdEnvVars
    </FilesMatch>
    <Directory /usr/lib/cgi-bin>
      SSLOptions +StdEnvVars
    </Directory>
    BrowserMatch "MSIE [2-6]" \
      nokeepalive ssl-unclean-shutdown \
      downgrade-1.0 force-response-1.0
    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
  </VirtualHost>
</IfModule>
```

Guarda el fichero y sal cuando hayas terminado.

Paso Cuatro — Configurar /etc/hosts

Hay que añadir una línea a /etc/hosts para que encuentre www.midominioseguro.com



```
root@pedrablava:/etc# cat hosts
127.0.0.1 www.midominioseguro.com
127.0.0.1 localhost
```

Paso Cinco — Activar el Host Virtual con SSL

Cuidado

Como tenemos muchos host virtuales activados, id a sites-enabled y

desactivad todos los que aparezcan allí.

Ahora que lo hemos configurado, hemos de activarlo.
Lo hacemos escribiendo en la consola:

```
sudo a2ensite default-ssl.conf
```

Y ahora reiniciamos apache:

```
sudo service apache2 restart
```

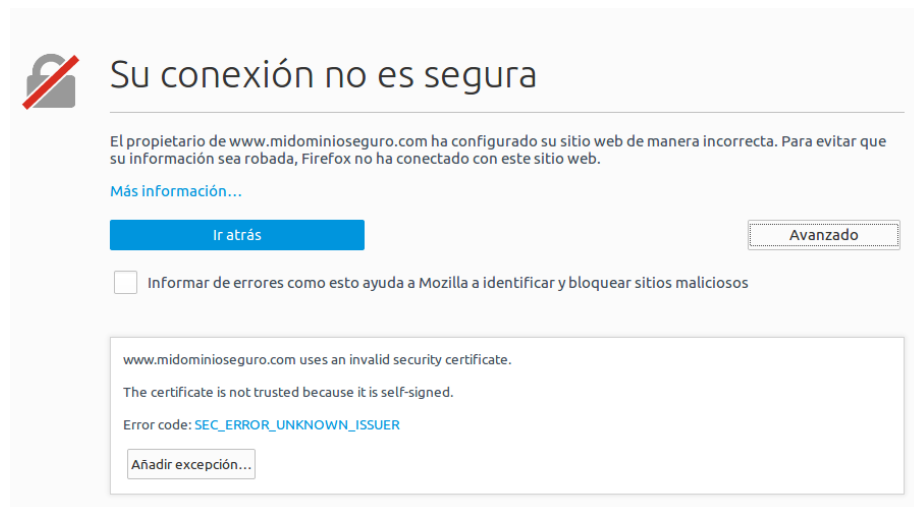
Esto debería activar nuestro nuevo virtual host, que servirá contenido encriptado usando el certificado SSL que hemos creado.

Paso Seis — Probar nuestra configuración

Ahora que ya está preparado, podemos probar nuestra configuración visitando nuestro nombre de dominio del servidor después de introducir el protocolo `https://`

`https://www.midominioseguro.com`

Obtendremos un aviso de que el navegador no puede verificar la identidad de nuestro servidor porque no ha sido firmado por ninguna de las autoridades de certificación en las que él confía



Esto es lo esperado ya que hemos generado un certificado auto-firmado. Evidentemente, nosotros mismos no nos podemos validar porque no tenemos interacción con ninguna de estas autoridades, pero sí que encriptará todas las comunicaciones entre el navegador y el cliente.

Podemos pulsar el botón *Avanzado*, o cualquier opción similar que muestre el navegador, y luego *Añadir Excepción*. En el diálogo que aparece, hacemos clic en *Confirmar excepción de seguridad*.

Ahora seremos dirigidos al contenido de `DocumentRoot` que hemos configurado en nuestro host virtual, pero ahora este tráfico está encriptado. Puedes comprobarlo haciendo clic en el icono de candado que aparece en el navegador en la barra de navegación.



Si visualizamos el Certificado en el navegador, veremos lo siguiente:



Conclusión

Ahora debemos tener SSL activado en nuestro sitio web. Esto asegurará las comunicaciones entre los visitantes y nuestro servidor, pero *mostrará un aviso* a cada usuario ya que el navegador no puede verificar la validez del certificado. Por tanto este tipo de certificados sólo sirven para intranets. Si tenemos planificado lanzar un sitio público con SSL, necesitaremos comprar un certificado emitido por una autoridad de certificados de confianza.

Redirigir el tráfico de nuestro sitio

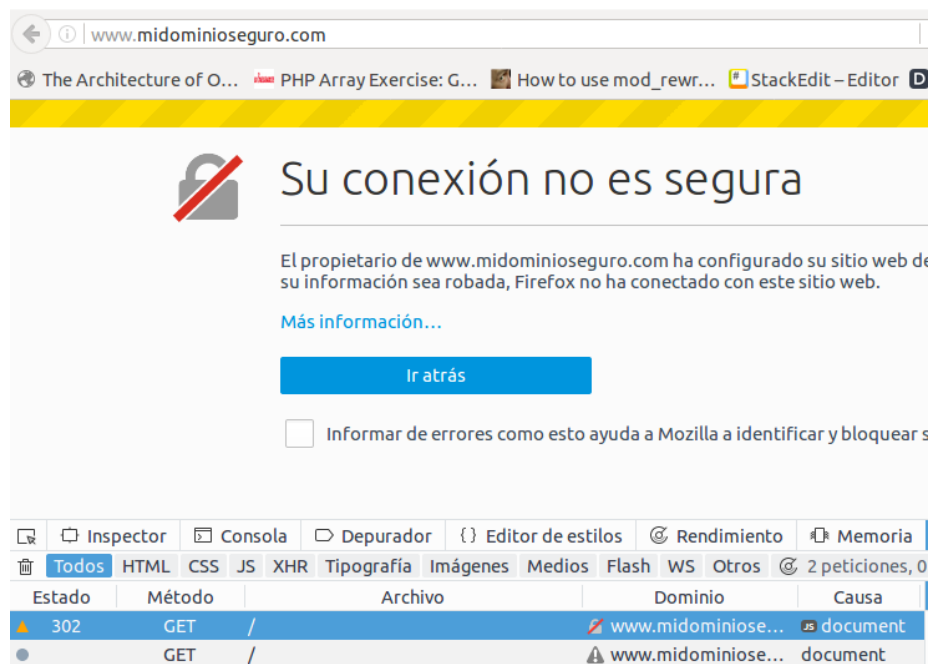
Ahora que ya tenemos instalado el certificado y configurado nuestro sitio para que funcione con **https**, nos interesa que todos aquellos clientes que accedan a la versión no segura, sean redirigidos a la versión segura.

Pensemos que tal vez algún usuario se guardó en favoritos una URI de nuestro sitio cuando todavía no habíamos implementado el protocolo seguro, o alguien nos menciona en un blog, o en un buscador, etc.

La forma más sencilla es modificando el archivo de configuración para el puerto 80 (el que corresponde con http)

Para ello, editamos el archivo `default-ssl.conf` y añadimos lo siguiente

```
<IfModule mod_ssl.c>
  <VirtualHost *:80>
    ServerName www.midominioseguro.com
    Redirect / https://www.midominioseguro.com/
  </VirtualHost>
...
</IfModule>
```



Cuando hayamos probado que todo nuestro sitio funciona como es esperado, haremos la redirección permanente.

```
<IfModule mod_ssl.c>
  <VirtualHost *:80>
    ServerName www.midominioseguro.com
    Redirect permanent / https://www.midominioseguro.com/
  </VirtualHost>
```


...
</IfModule>

Certificado provisto por una autoridad de confianza

Con este método estamos generando un certificado en el que el navegador no confía. Es por ello que nos da alertas de seguridad.

En una situación real, debemos instalar un certificado digital obtenido de una entidad certificadora reconocida como, por ejemplo, letsEncrypt.org