

ASEGURAR APACHE



G.I.I. E. Tecnologías de la Información
Servidores Web Altas Prestaciones
Curso 14/15

Rafael Ortiz Cáceres
Adrián Álvarez Sáez
Álvaro Pérez Luque

Índice

[Introducción](#)

[Primeros pasos](#)

[Usuarios](#)

[Permisos del directorio raíz del servidor](#)

[Deshabilitar módulos](#)

[Configuración mediante directivas](#)

[Archivos de configuración](#)

[Ocultamiento de información](#)

[Información del Servidor](#)

[Deshabilitar el listado de ficheros](#)

[Limitar el acceso a archivos por su extensión](#)

[Control de acceso a los archivos publicados](#)

[Denegar el acceso por defecto](#)

[Evitar la resolución de enlaces simbólicos.](#)

[Autenticación, Autorización y Control de Acceso](#)

[Autenticación](#)

[Control de Acceso](#)

[Seguridad ante ataques DOS.](#)

[Directiva Timeout](#)

[Directiva KeepAliveTimeOut](#)

[Directiva MaxRequestWorkers](#)

[Directiva RequestReadTimeout](#)

[Directiva LimitRequestBody](#)

[SSL en apache](#)

[Bibliografía](#)

1. Introducción

En la actualidad cada vez más aplicaciones y servicios se desarrollan para entornos web, siendo accesibles a través del navegador facilitando su acceso desde las redes internas de las empresas o a través de internet, esta facilidad de acceso también puede suponer una vulnerabilidad ante ataques externos hacia los servidores web que ofrecen los servicios y contenidos, por este motivo se debe poner especial énfasis en la seguridad de los servidores web.

Algunos de los problemas de seguridad más comunes son: caídas del servidor debidas a ataques de denegación de servicio, problemas de seguridad derivados de una mala configuración, robo de información confidencial, modificaciones no autorizadas de los documentos web (Defacement) o inyección de código malicioso.

En este trabajo se intenta aportar unas directrices y consejos para intentar en la medida de lo posible asegurar un servidor HTTP Apache, concretamente la versión 2.2.22.

Brevemente los contenidos que desarrollamos en este trabajo son:

- Cómo modificar la configuración de Apache y sus medidas de seguridad.
- Control de acceso a la información.
- Configuración del protocolo HTTPS, que proporciona cifrado de la comunicación y autenticación del servidor.
- Seguridad básica ante ataque DOS.
- Opciones de configuración de acceso a información importante del servidor.

Todas las instrucciones que se muestran en este documento han sido probadas en una máquina con Ubuntu 12.04.05 y Apache 2.2.22.

2. Primeros pasos

2.1. Usuarios

Normalmente Apache es iniciado por el usuario root, y después cambia a un usuario con privilegios limitados para servir las peticiones.

En primer lugar, debemos comprobar que el usuario y grupo que usa el servidor para responder las peticiones no tenga privilegios para acceder a los archivos que no están destinados a ser visible para el mundo exterior, ni ejecutar código que no sea para las solicitudes HTTP.

Comprobamos que usuario y grupo están designados para que el servidor responda las peticiones. Estos se definen mediante las directivas **User** y **Group**.

```
egrep "^User|^Group|^SuexecUserGroup"  
/etc/apache2/apache2.conf /etc/apache2/sites-available/*
```

```
adalsa@ubuntu-cliente:~$ egrep "^User|^Group|^SuexecUserGroup" /etc/apache2/apache2.conf /etc/apache  
2/sites-available/*  
/etc/apache2/apache2.conf:User ${APACHE_RUN_USER}  
/etc/apache2/apache2.conf:Group ${APACHE_RUN_GROUP}
```

Como se puede ver en la imagen apache está configurado por defecto para usar el usuario y grupo definidos en las variables de entorno **APACHE_RUN_USER** y **APACHE_RUN_GROUP**, el valor de estas está definido en **/etc/apache2/envvars**.

```
adalsa@ubuntu-cliente:~$ egrep "APACHE_RUN_USER|APACHE_RUN_GROUP" /etc/apache2/envvars  
export APACHE_RUN_USER=www-data  
export APACHE_RUN_GROUP=www-data
```

Tanto el usuario **www-data** como el grupo **www-data** son creados automáticamente en el proceso de instalación. De esta forma cualquier documento al que no pueda acceder este usuario consecuentemente será inaccesible para el cliente web.

2.2. Permisos del directorio raíz del servidor

Debemos asegurar los archivos contenidos en la carpeta raíz del servidor para que estos solo puedan ser modificados por el root. Esto no solo se aplica a los archivos, también a los directorios que los contienen y a los padres de todos los directorios.

Para ello ejecutamos los siguientes órdenes:

```
# chown --recursive 0:0 /etc/apache2
# chmod --recursive 750 /etc/apache2
```

Con la primera orden establecemos como usuario y grupo propietario a root, la segunda orden concede todos los privilegios al usuario root, privilegios de lectura y ejecución para el grupo root, y ningún permiso para el resto de usuarios.

2.3. Deshabilitar módulos

Como medida de seguridad es aconsejable deshabilitar los módulos que no se vayan utilizar, así evitamos posibles ataques a vulnerabilidades de módulos que no son necesarios y además ahorraremos recursos.

Dentro del directorio **/etc/apache2** podemos encontrar los directorios:

-*mods-available*: Donde se encuentran los módulos disponibles para ser habilitados.

-*mods-enabled*: Donde se encuentran enlaces simbólicos a los módulos de *mods-available* están actualmente habilitados.

Para ver los módulos cargados en apache podemos usar:

```
# apache2ctl -M
```

```
adalsa@ubuntu-cliente:~$ apache2ctl -M
Loaded Modules:
  core_module (static)
  log_config_module (static)
  logio_module (static)
  mpm_worker_module (static)
  http_module (static)
  so_module (static)
  alias_module (shared)
  auth_basic_module (shared)
  authn_file_module (shared)
  authz_default_module (shared)
  authz_groupfile_module (shared)
  authz_host_module (shared)
  authz_user_module (shared)
  autoindex_module (shared)
  cgid_module (shared)
  deflate_module (shared)
  dir_module (shared)
  env_module (shared)
  mime_module (shared)
  negotiation_module (shared)
  proxy_module (shared)
  reqtimeout_module (shared)
  setenvif_module (shared)
  status_module (shared)
Syntax OK
```

Para habilitar o deshabilitar un módulo disponemos de los siguientes comandos

```
# a2dismod proxy_ftp //Deshabilita el módulo proxy_ftp
```

```
# a2enmod proxy_ftp //Habilita el módulo proxy_ftp
```

Para activar la nueva configuración es necesario reiniciar Apache:

```
# service apache2 restart
```

3. Configuración mediante directivas

3.1. Archivos de configuración

La configuración de Apache se realiza a través de directivas en ficheros de configuración de texto plano. Históricamente el archivo principal de configuración ha sido `httpd.conf`, pero esto puede variar dependiendo de la distribución que usemos, en nuestro caso el fichero de configuración principal es `apache2.conf`. Podemos consultar la ubicación del fichero de configuración principal mediante el siguiente comando:

```
apahce2ctl -V
```

```
adalsa@ubuntu-cliente:~$ apachectl -V
Server version: Apache/2.2.22 (Ubuntu)
Server built:   Mar  5 2015 18:10:14
Server's Module Magic Number: 20051115:30
Server loaded:  APR 1.4.6, APR-Util 1.3.12
Compiled using: APR 1.4.6, APR-Util 1.3.12
Architecture:   64-bit
Server MPM:      Worker
  threaded:      yes (fixed thread count)
  forked:        yes (variable process count)
Server compiled with....
-D APACHE_MPM_DIR="server/mpm/worker"
-D APR_HAS_SENDFILE
-D APR_HAS_MMAP
-D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
-D APR_USE_SYSVSEM_SERIALIZE
-D APR_USE_PTHREAD_SERIALIZE
-D SINGLE_LISTEN_UNSERIALIZED_ACCEPT
-D APR_HAS_OTHER_CHILD
-D AP_HAVE_RELIABLE_PIPED_LOGS
-D DYNAMIC_MODULE_LIMIT=128
-D HTTPD_ROOT="/etc/apache2"
-D SUEXEC_BIN="/usr/lib/apache2/suexec"
-D DEFAULT_PIDLOG="/var/run/apache2.pid"
-D DEFAULT_SCOREBOARD="logs/apache_runtime_status"
-D DEFAULT_ERRORLOG="logs/error_log"
-D AP_TYPES_CONFIG_FILE="mime.types"
-D SERVER_CONFIG_FILE="apache2.conf"
adalsa@ubuntu-cliente:~$
```

Observando la salida, podemos ver la ruta del fichero de configuración principal.

```
SERVER_CONFIG_FILE="apache2.conf"
```

Generalmente los archivos que posee apache para su configuración son:

- `/etc/apache2/apache2.conf`, Es el archivo principal contiene la mayoría de directivas de configuración de apache, se aconseja no modificarlo directamente, e incluir en él los archivos de configuración, para evitar problemas al realizar actualizaciones de apache y tener un mayor control y organización.
- `/etc/apache2/mods-enabled`, en este directorio se encuentran los archivos que gestionan la carga y configuración de los distintos y posibles módulos de apache.
- `/etc/httpd/httpd.conf`, este archivo contiene información y directivas aplicables que controlan las funciones del servidor.
- `/etc/apache2/ports.conf`, contiene las directivas y reglas de control sobre los puertos que usará apache.
- `/etc/apache2/sites-enabled`: este directorio contiene los archivos de configuración de cada virtual host.

3.2 Ocultamiento de información

Una de las primeras fases de los ataques sobre los sistemas de información es la revelación de información. Cuanta más información sobre el sistema tenga el atacante, más fácil le resultará encontrar vulnerabilidades existentes en el sistema atacado.

3.2.1. Información del Servidor

Una de las medidas que se pueden tomar es ocultar toda información de la versión de Apache que se muestra en el pie de página de los documentos generados por el servidor (mensajes de error, listados de directorios `ftp mod_proxy`, `salida mod_info`, ...), para que el atacante no obtenga información específica.

Forbidden

You don't have permission to access / on this server.

Apache/2.2.22 (Ubuntu) Server at 192.168.1.3 Port 80

Para ello utilizamos la siguiente directiva:

```
ServerSignature Off
```

Y con esto podemos ver que ahora no nos muestra la información de Apache.

Forbidden

You don't have permission to access / on this server.

También podemos desactivar el envío de la versión de apache en las cabeceras de respuesta a los clientes mediante la siguiente directiva:

```
ServerTokens Prod
```

```
root@ubuntu-cliente:/etc/apache2# curl -i http://127.0.0.1/pagina_inexistente
HTTP/1.1 404 Not Found
Date: Sun, 31 May 2015 01:46:00 GMT
Server: Apache
Vary: Accept-Encoding
Content-Length: 216
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /pagina_inexistente was not found on this server.</p>
</body></html>
root@ubuntu-cliente:/etc/apache2# _
```

La opción *Prod* es la más restrictiva pero sigue mostrando el nombre del servidor como se puede apreciar en la imagen

3.2.2. Deshabilitar el listado de ficheros

Si solicitamos a Apache un directorio en vez de un archivo, éste nos devolverá un listado de todos los archivos del directorio. Podemos controlar este comportamiento y otras características para un directorio en particular mediante la directiva *Options* que puede tener los siguientes valores:

- **ExecCGI:** se permite la ejecución de scripts CGI.
- **FollowSymLinks:** el servidor seguirá los enlaces simbólicos.
- **Includes:** se permiten incluir Server-side.
- **IncludesNOEXEC:** se permiten incluir Server-side pero se deshabilitan las órdenes `#exec` y `#exec CGI`.
- **Indexes:** Si una URL solicita un directorio y no existe `DirectoryIndex`
- **MultiViews:** se permiten mostrar contenido negociado en función de diversos valores.
- **SymLinksIfOwnerMatch:** Se sigue un enlace simbólico sólo si los propietarios del enlace y del destino coinciden.
- **All:** todas las opciones salvo `MultiViews`.
- **None:** No permite ninguna opción.

Para desactivar la indexación de carpetas bastaría con usar la siguiente directiva:

```
Options -Indexes
```

En nuestro caso hemos usado el más restrictivo que es *none* para el directorio raíz de este modo todas las características incluidas el índice de carpetas estarán desactivadas.

```
GNU nano 2.2.6 File: sites-enabled/000-default
<VirtualHost *:80>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www
    <Directory />
        Order deny,allow
        Deny from all
        Options None
        AllowOverride None
    </Directory>
```

3.2.3. Limitar el acceso a archivos por su extensión

Nos puede interesar bloquear el acceso a determinados ficheros que necesariamente deben estar dentro del **DocumentRoot** pero no están destinados a ser accedidos por el público, como por ejemplo los ficheros **.htaccess** y **.htpasswd**, para denegar el acceso a estos ficheros utilizaremos la siguiente directiva que deniegue el acceso a todos los ficheros que comiencen por **.ht**:

```
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
```

Para denegar el acceso a un directorio se utiliza la directiva **DirectoryMatch**:

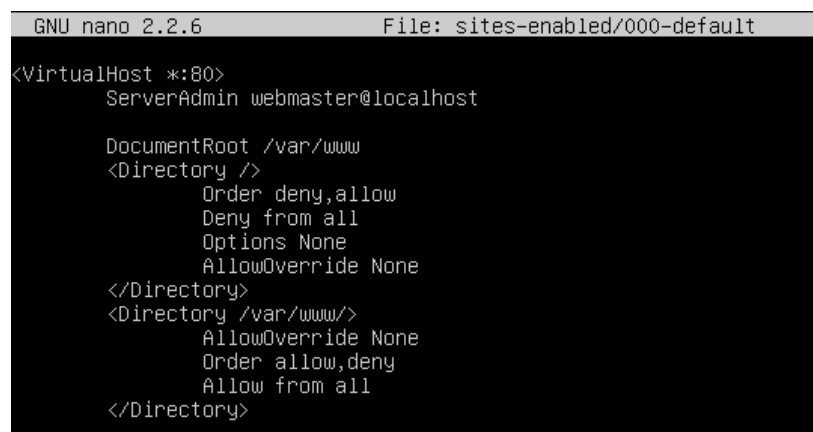
```
<DirectoryMatch /Backup/>
    Order Allow,Deny
    Deny from all
</DirectoryMatch>
```

3.3 Control de acceso a los archivos publicados

Unos de los problemas más importantes para Apache es que no se aplican las directivas adecuadas a los ficheros que se ubican en el directorio del sitio web, este directorio se especifica con la directiva **DocumentRoot**.

3.3.1. Denegar el acceso por defecto

Es recomendable denegar el acceso a todos los directorios por defecto y permitir el acceso sólo al directorio especificado en el **DocumentRoot** explícitamente.



```
GNU nano 2.2.6 File: sites-enabled/000-default
<VirtualHost *:80>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www
    <Directory />
        Order deny,allow
        Deny from all
        Options None
        AllowOverride None
    </Directory>
    <Directory /var/www/>
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>
```

La directiva **Directory** sirve para aplicar una serie de directivas sobre la carpeta especificada y a todas sus subcarpetas. Lo más común es que existan varias de estas directivas, y que como también se aplican a los subdirectorios, existan directivas contradictorias aplicadas a un directorio. En este caso, la directiva del **Directory** más específica es la que prevalece a menos que se especifique lo contrario mediante la directiva **AllowOverride All/None/directiva**.

Las directivas **Allow** y **Deny** son utilizadas para permitir o denegar respectivamente el acceso a un directorio y **Order** especifica el orden en el que serán evaluadas.

3.3.2. Evitar la resolución de enlaces simbólicos

Apache puede recibir una petición a un archivo que es, a nivel de sistema operativo del servidor, un enlace simbólico y devolver el archivo apuntado por él aunque se encuentre fuera del DocumentRoot.

Esta funcionalidad puede posibilitar que un atacante, que tenga permisos de escritura sobre el DocumentRoot, cree un enlace simbólico a archivos contenidos fuera del DocumentRoot para luego acceder a ellos a través del navegador.

Para deshabilitar esta funcionalidad se utiliza la directiva:

```
Options -FollowSymLinks
```

También podemos mantener la resolución de enlaces pero restringiendo que solo se puedan usar si el archivo destino pertenece al mismo usuario que el enlace simbólico, para ello usamos la siguiente directiva:

```
Options -FollowSymLinks +SymLinksIfOwnerMatch
```

3.4 Autenticación, Autorización y Control de Acceso

3.4.1. Autenticación

Podemos activar un proceso de autenticación en el que al usuario que intenta acceder a un recurso se le soliciten unas credenciales.

Para este método de autenticación usaremos los ficheros **.htaccess**, estos archivos nos permiten definir directivas de configuración para un directorio en concreto (incluidos subdirectorios), la carpeta a la que afectarán estas directivas será en la que está ubicado el archivo **.htaccess**.

Para que estos archivos puedan sobrescribir las directivas del fichero de configuración principal que atañen a la seguridad usaremos la siguiente directiva:

```
GNU nano 2.2.6 File: sites-enabled/000-default
<VirtualHost *:80>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www
    <Directory />
        Order deny,allow
        Deny from all
        Options None
        AllowOverride None
    </Directory>
    <Directory /var/www/>
        AllowOverride AuthConfig
        Order allow,deny
        Allow from all
    </Directory>
```

En primer lugar crearemos el fichero que contendrá las contraseñas añadiendo un usuario, para ello usamos el comando **htpasswd**.

```
root@ubuntu-cliente:/etc/apache2# htpasswd -c /etc/apache2/passwd adrian
New password:
Re-type new password:
Adding password for user adrian
```

Las contraseñas se almacenan como una combinación del hash MD5 de la contraseña más un salt aleatorio de 32bits.

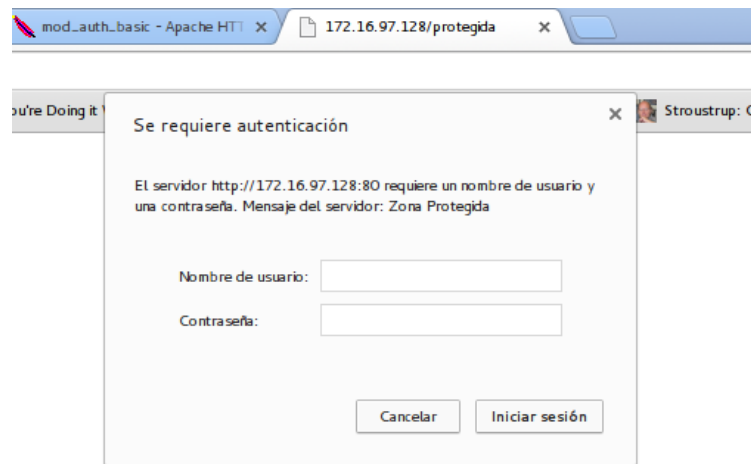
Una vez creado el fichero de contraseñas crearemos un fichero **.htaccess** en la directorio que deseamos proteger.

```
GNU nano 2.2.6      File: /var/www/protegida/.htaccess
AuthType Basic
AuthName "Zona Protegida"
AuthUserFile "/etc/apache2/passwd"
Require valid-user
```

Con estas directivas hemos especificado un método de autenticación basado en credenciales en texto plano, cuya ruta es **/etc/apache2/passwd** y con la directiva **Require valid-user** especificamos que puede acceder cualquier usuario que aparezca en el archivo de contraseñas,, si quisiéramos especificar qué usuarios pueden acceder usamos la directiva:

```
Require adrian
```

Podemos comprobar que al intentar acceder a través del navegador, cuando intentamos acceder al sitio web nos pedirá las credenciales de acceso.



3.4.2. Control de Acceso

Podemos controlar el acceso a cualquier recurso basándose en la dirección del host del visitante. Esto nos permitiría por ejemplo restringir el acceso a unos determinados recursos destinados solo a la red interna de la empresa.

Para este paso necesitaremos el módulo **mod_auth_host**, en caso de no tenerlo activado usamos el siguiente comando para activarlo:

```
a2enmod authz_host
```

Para controlar el acceso utilizaremos la directiva **Require** que admite dos usos:

```
Require host address  
Require ip ip.address
```

Se puede especificar más de una ip en una misma directiva.

También podemos usar la opción **not** delante de **ip** o **host** para negar el requisito. Por ejemplo para negar el acceso a una ip que ha estado atacando:

```
<RequireAll>  
    Require all granted  
    Require not ip 217.216.37.42
```

```
</RequireAll>
```

Incluso podemos bloquear el acceso desde un dominio:

```
Require not host ugr.es
```

3.5. Seguridad ante ataques DOS.

Un ataque de denegación de servicio, básicamente consiste en saturar los recursos del sistema impidiendo a consecuencia el acceso a los usuarios legítimos.

3.5.1. Directiva Timeout

Esta directiva define el tiempo que espera apache para ciertas circunstancias I/O, como por ejemplo:

- El tiempo máximo para esperar un paquete TCP del cliente, si el buffer de lectura está vacío.
- El tiempo máximo para esperar el acuse de recibo de un paquete, si el buffer de envío está lleno.

Este valor viene por defecto establecido en 300 seg, pero en caso de que nuestro servidor sea susceptible de ataques DOS podemos reducir a unos pocos segundos, por ejemplo:

```
Timeout 10
```

3.5.2. Directiva KeepAliveTimeOut

La directiva KeepAliveTimeout establece el número de segundos que el servidor esperará tras haber dado servicio a una petición, antes de cerrar la conexión. Establecemos un KeepAliveTimeout de 5 segundos, si tenemos un servidor muy susceptible a ataques podría fijarse en menos, o incluso desactivar la directiva KeepAlive.

```
KeepAliveTimeout 5
```


3.5.3. Directiva MaxRequestWorkers

Modificando el número máximo de peticiones que puede manejar simultáneamente el servidor podemos evitar que el servidor se quede sin recursos debido a un alto número de peticiones.

```
MaxRequestWorkers 256
```

3.5.4. Directiva RequestReadTimeout

Esta directiva permite limitar el tiempo que un cliente puede tardar para enviar la petición, permite modificar individualmente el tiempo de espera para la cabecera y el cuerpo.

```
RequestReadTimeout header=10 body=20
```

3.5.5. Directiva LimitRequestBody

Con esta directiva podemos especificar el número de bytes que se permite en el cuerpo de una petición.

```
LimitRequestBody 102400
```

4. SSL en apache

En primer lugar activaremos el módulo ssl ejecutando el siguiente comando:

```
#a2enmod ssl
```

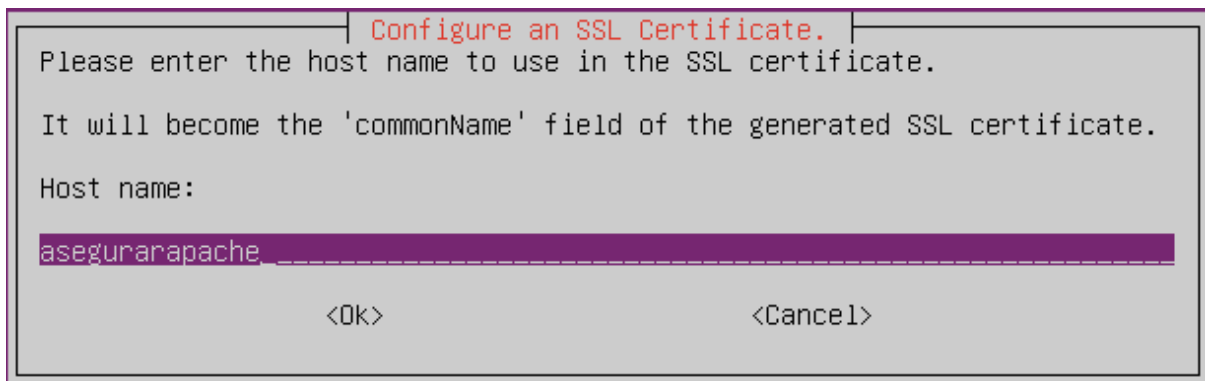
También debemos activar el sitio **default-ssl** que viene incluido por defecto en Apache, para ello usamos el comando:

```
#a2ensite default_ssl
```

Ahora generamos un certificado firmado por nosotros mismos, para ello necesitamos el paquete **ssl-cert** y mediante el siguiente comando creamos el certificado:

```
make-ssl-cert /usr/share/ssl-cert/ssleay.cnf  
/etc/apache2/ssl/certificado.crt
```

Nos mostrará un diálogo para introducir el nombre para el certificado SSL.



Hay que comprobar que el certificado solo tenga permisos de lectura para root.

```
root@ubuntu-cliente:/etc/apache2# ll /etc/apache2/ssl/certificado.crt  
-rw----- 1 root root 2693 May 31 20:04 /etc/apache2/ssl/certificado.crt
```

Por último tendremos que editar el fichero de configuración del virtual host ssl.

```
GNU nano 2.2.6      File: sites-enabled/default-ssl

#  SSL Engine Switch:
#  Enable/Disable SSL for this virtual host.
SSLEngine on

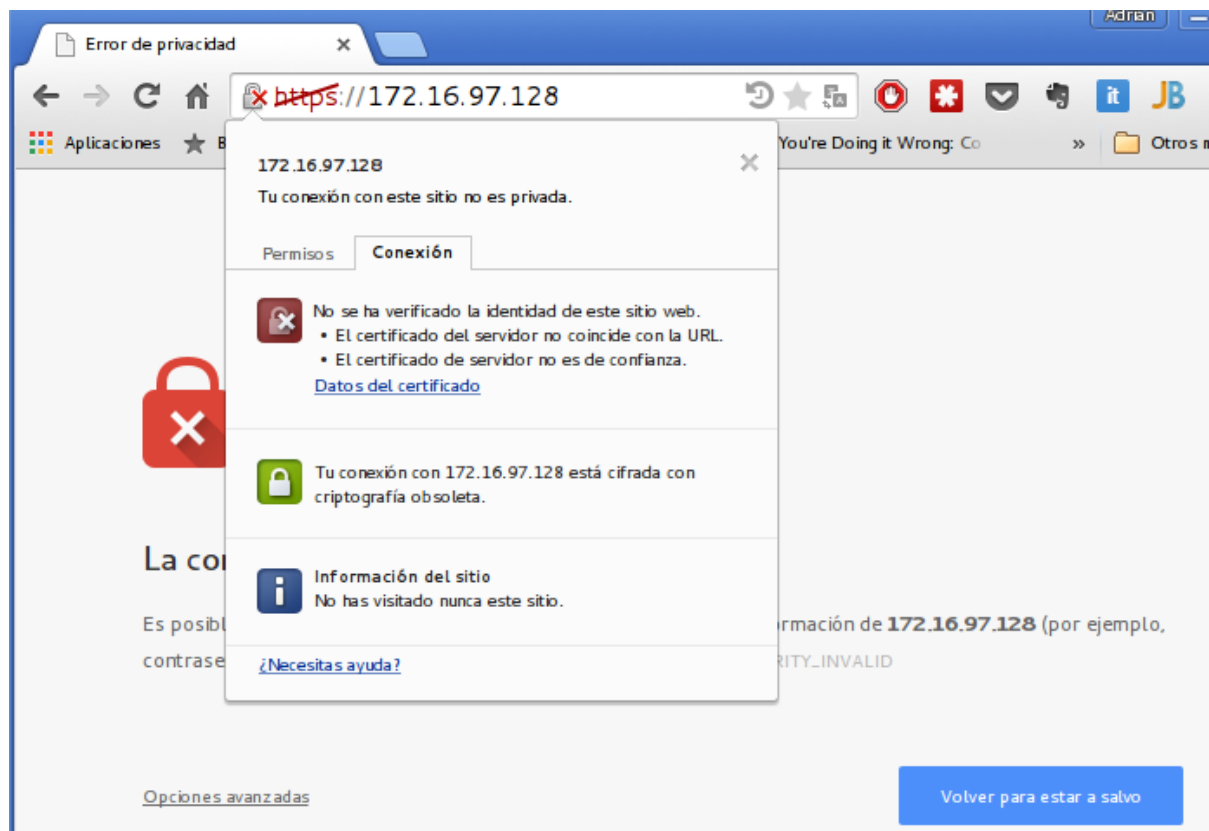
#  A self-signed (snakeoil) certificate can be created by installing
#  the ssl-cert package. See
#  /usr/share/doc/apache2.2-common/README.Debian.gz for more info.
#  If both key and certificate are stored in the same file, only the
#  SSLCertificateFile directive is needed.
SSLCertificateFile  /etc/apache2/ssl/certificado.crt
#SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
```

Añadimos la directiva ***SSLCertificateFile*** apuntando al certificado ***.crt*** que acabamos de crear, no es necesario usar la directiva ***SSLCertificateKeyFile*** ya que la key esta ya incluida en el archivo ***.crt***.

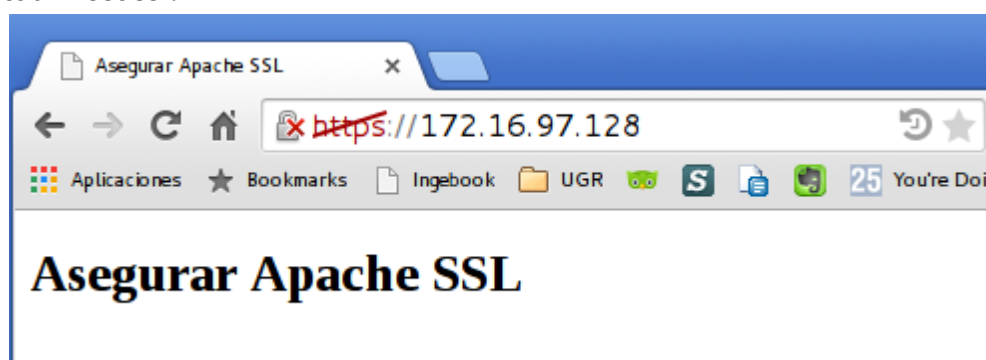
Reiniciamos el servidor para que los cambios tengan efecto.

```
#apache2ctl restart
```

Para probar que todo se ha configurado correctamente probamos a acceder desde un navegador especificando el protocolo ***https://***.



La advertencia que se ve en la imagen es debido a que el certificado que usamos está firmado por nosotros mismo y no por una autoridad reconocida por el explorador. Una vez que le indiquemos al explorador que queremos continuar se mostrará correctamente la página principal que se encuentre en el **DocumentRoot** del virtual host ssl.



Bibliografía

- Apache 2.2 Security Tips (http://httpd.apache.org/docs/2.2/misc/security_tips.html)
- Apache 2.2 Documentation (<http://httpd.apache.org/docs/2.2/en/>)
- Apache Web Server Hardening Security
(<http://geekflare.com/apache-web-server-hardening-security/>)