



Asegurar un servidor Apache



Índice



- Introducción
- Principales archivos que forman Apache
- Medidas de seguridad básicas
- Encriptar ficheros
- Repositorio del trabajo



Introducción

En este trabajo se presentarán una serie de medidas básicas que debemos tomar para hacer más seguro nuestro servidor Apache. El proceso será realizado en una máquina virtual que contiene Ubuntu Server.



Principales archivos que forman Apache

- **/etc/apache2/apache2.conf**: el archivo raíz es el que incluye a los demás. No se debe modificar este archivo.
- **mods-enabled/*.load y mods-enabled/*.conf**: la finalidad de estos archivos es la carga y configuración de los módulos de Apache.
- **httpd.conf**: directivas aplicables a todos los servidores web.
- **ports.conf**: define en qué puertos “escuchará” Apache.
- **conf.d/**: directorio que contiene archivos de configuración para cada funcionalidad de apache (charset, php, security, etc.)
- **sites-enabled/**: directorio que contiene los archivos de configuración de cada virtual host

Medidas de seguridad básicas



En este apartado iremos viendo una a una todas las medidas más básicas que podemos utilizar para asegurar nuestro servidor Apache. Cada medida irá explicada mediante una captura de pantalla con el proceso, para que se pueda ver bien como se hace.

Todas estas medidas se realizan añadiendo directivas al archivo **httpd.conf** .



¡ATENCIÓN!

En algunas distribuciones de Linux, como por ejemplo Ubuntu, el archivo **httpd.conf** no existe como tal, sino que su contenido está incluido en el archivo **apache2.conf** . En este caso, este será el archivo que tendremos que modificar. La zona concreta donde empezar a escribir es justo debajo de lo siguiente:

```
# Sets the default security model of the Apache2 HTTPD server. It does
# not allow access to the root filesystem outside of /usr/share and /var/www.
# The former is used by web applications packaged in Debian,
# the latter may be used for local directories served by the web server. If
# your system is serving content from a sub-directory in /srv you must allow
# access here, or in any related virtual host.
```

Esconder el número de versión de Apache y otra información relevante



Tenemos que añadir las sentencias:

- *ServerSignature Off*: Para ocultar la firma del servidor.
- *ServerTokens Prod*: Le decimos lo que escribirá Apache en la cabecera de respuesta HTTP del servidor.



Esconder el número de versión de Apache y otra información relevante

```
# Sets the default security model of the Apache2 HTTPD server. It does
# not allow access to the root filesystem outside of /usr/share and /var/www.
# The former is used by web applications packaged in Debian,
# the latter may be used for local directories served by the web server. If
# your system is serving content from a sub-directory in /srv you must allow
# access here, or in any related virtual host.
```

```
ServerSignature Off
ServerTokens Prod
```


Asegurarse que archivos fuera de la raíz web no sean accesibles



Tenemos que hacer que Apache no sea capaz de acceder a ningún archivo que no vaya a necesitar. En este caso vamos a evitar que Apache acceda a archivos que se encuentren fuera de su directorio

Añadimos al archivo de configuración las sentencias que vemos en la página siguiente.



Asegurarse que archivos fuera de la raíz web no sean accesibles



Como podemos ver, se deniegan todos los accesos a los directorios que no sean /var/www. Además podemos ver la orden *Options None* que más adelante veremos para qué sirve. *AllowOverride None* controla que otros usuarios no puedan poner archivos en el directorio.

```
<Directory />
    Order Deny,Allow
    Deny from all
    Options None
    AllowOverride None
</Directory>

<Directory /var/www/>
    Order Allow,Deny
    Allow from all
</Directory>_
```

Permisos sobre los ficheros



Ningún usuario debe tener acceso a los ficheros alojados en el servidor, ya que podría instalar código malicioso. Para ello ejecutamos las órdenes:

- *chown -R root:root /usr/share/apache*
- *chmod -R o-rwx /usr/share/apache*

Con este comando estamos cambiando el usuario propietario a root y quitando todos los permisos al resto de usuarios.



Permisos sobre los ficheros



```
root@UbuntuServer:/# chown -R root:root /usr/share/apache2
root@UbuntuServer:/# chmod -R o-rwx /usr/share/apache2/
root@UbuntuServer:/#
```

Ahora vamos a eliminar los permisos de lectura sobre archivos de configuración y logs de apache. Para ello ejecutamos los comandos:

```
chmod -R go-r /etc/apache2/
```

```
chmod -R go-r /var/log/apache2/
```

```
root@UbuntuServer:/# chmod -R go-r /etc/apache2/
root@UbuntuServer:/# chmod -R go-r /var/log/apache2/
root@UbuntuServer:/# _
```

Desactivar la exploración de directorios y las inclusiones del lado del servidor

En este apartado es donde toma importancia el comando *Options none* que habíamos puesto antes, recordemos:

```
<Directory />
    Order Deny,Allow
    Deny from all
    Options None
    AllowOverride None
</Directory>

<Directory /var/www/>
    Order Allow,Deny
    Allow from all
</Directory>_
```

No permitir a Apache utilizar enlaces simbólicos




Con esta medida evitamos que un atacante con permisos de escritura dentro del directorio pueda crear enlaces a archivos fuera del directorio. Tenemos que escribir la sentencia:

Options -FollowSymLinks

No obstante, como podemos observar, la medida *Options none* tomada anteriormente es más restrictiva que esta, por tanto ya estamos protegidos ante este tipo de amenaza si usamos el comando anterior




Desactivar los módulos no necesarios



En muchas ocasiones Apache viene instalado con módulos que no necesitaremos. Aquí aprenderemos a instalar o desinstalar los módulos que queramos.

Primero de todo, tener en cuenta que en las distribuciones que posean el archivo **httpd.conf**, estos módulos simplemente se activarán o desactivarán comentándolos con un # en la línea en la que aparezca *LoadModule <Nombre_del_modulo>*



Desactivar los módulos no necesarios

En nuestro caso, como estamos trabajando con una distribución Ubuntu Server, el proceso es diferente, ya que no existe dicho archivo de configuración.

Para empezar, podemos encontrar una lista con los módulos instalados en el directorio:

/etc/apache2/mods-available

Desactivar los módulos no necesarios

Aquí podemos ver la lista de módulos que tenemos instalados, simplemente haciendo un `ls` en el directorio mencionado

```
Ubuntu Server trabajo [Corriendo] - Oracle VM VirtualBox
Máquina Ver Dispositivos Ayuda
authn_socache.load lbmethod_bybusyness.load request.load
authnz_ldap.load lbmethod_byrequests.load rewrite.load
authz_core.load lbmethod_bytraffic.load sed.load
authz_dbd.load lbmethod_heartbeat.load session_cookie.load
authz_dbm.load ldap.conf session_crypto.load
authz_groupfile.load ldap.load session_dbd.load
authz_host.load log_debug.load session.load
authz_owner.load log_forensic.load setenvif.conf
authz_user.load lua.load setenvif.load
autoindex.conf macro.load slotmem_plain.load
autoindex.load mime.conf slotmem_shm.load
buffer.load mime.load socache_dbm.load
cache_disk.conf mime_magic.conf socache_memcache.load
cache_disk.load mime_magic.load socache_shmcb.load
cache.load mpm_event.conf spelling.load
cache_socache.load mpm_event.load ssl.conf
cgid.conf mpm_prefork.conf ssl.load
cgid.load mpm_prefork.load status.conf
cgi.load mpm_worker.conf status.load
charset_lite.load mpm_worker.load substitute.load
data.load negotiation.conf suexec.load
dav_fs.conf negotiation.load unique_id.load
dav_fs.load php5.conf userdir.conf
dav.load php5.load userdir.load
dav_lock.load proxy_ajp.load usertrack.load
dbd.load proxy_balancer.conf vhost_alias.load
deflate.conf proxy_balancer.load xml2enc.load
deflate.load proxy.conf
dialup.load proxy_connect.load
root@UbuntuServer:/etc/apache2/mods-available#
```

Desactivar los módulos no necesarios



Los módulos activos podemos encontrarlos en el directorio:

/etc/apache2/mods-enabled

Podemos instalar o eliminar módulos mediante el gestor *apt* y automáticamente se añaden o eliminan del directorio de módulos instalados.



Desactivar los módulos no necesarios



Para activarlos o desactivarlos usaremos los comandos *a2enmod* y *a2dismod* respectivamente.

En las diapositivas siguientes veremos un ejemplo de su uso.



Desactivar los módulos no necesarios

```
Ubuntu Server trabajo [Corriendo] - Oracle VM VirtualBox
Máquina Ver Dispositivos Ayuda
root@UbuntuServer:/etc/apache2/mods-available# sudo a2enmod buffer.load
Enabling module buffer.
To activate the new configuration, you need to run:
    service apache2 restart
root@UbuntuServer:/etc/apache2/mods-available# sudo service apache2 restart
* Restarting web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
[ OK ]
root@UbuntuServer:/etc/apache2/mods-available#
```

```
Ubuntu Server trabajo [Corriendo] - Oracle VM VirtualBox
Máquina Ver Dispositivos Ayuda
root@UbuntuServer:/etc/apache2/mods-enabled# ls
access_compat.load  authz_user.load  env.load          php5.conf
alias.conf          autoindex.conf  filter.load       php5.load
alias.load          autoindex.load  mime.conf         setenvif.conf
auth_basic.load     buffer.load     mime.load         setenvif.load
authn_core.load     deflate.conf    mpm_prefork.conf status.conf
authn_file.load     deflate.load    mpm_prefork.load status.load
authz_core.load     dir.conf       negotiation.conf
authz_host.load     dir.load       negotiation.load
```

Desactivar los módulos no necesarios

```
Ubuntu Server trabajo [Corriendo] - Oracle VM VirtualBox
Máquina Ver Dispositivos Ayuda
root@UbuntuServer:/etc/apache2/mods-enabled# sudo a2dismod buffer.load
Module buffer disabled.
To activate the new configuration, you need to run:
  service apache2 restart
root@UbuntuServer:/etc/apache2/mods-enabled# sudo service apache2 restart
* Restarting web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
[ OK ]
```

```
Ubuntu Server trabajo [Corriendo] - Oracle VM VirtualBox
Máquina Ver Dispositivos Ayuda
root@UbuntuServer:/etc/apache2/mods-enabled# ls
access_compat.load  authz_user.load  filter.load      php5.load
alias.conf          autoindex.conf  mime.conf        setenvif.conf
alias.load          autoindex.load  mime.load        setenvif.load
auth_basic.load     deflate.conf     mpm_prefork.conf status.conf
authn_core.load     deflate.load     mpm_prefork.load status.load
authn_file.load     dir.conf        negotiation.conf
authz_core.load     dir.load        negotiation.load
authz_host.load     env.load        php5.conf
```

Disminuir el valor de Timeout

Disminuyendo el valor del timeout dificultamos un ataque de denegación de servicios. Por defecto suele estar en 300 seg, podemos ponerlo más corto añadiendo:

Timeout tiempo_que_queramos

Nosotros vamos a ponerlo a 100 por ejemplo:

```
#  
# Timeout: The number of seconds before receives and sends time out.  
#  
Timeout 100
```

Ajustar el tamaño de los pedidos

Esta medida también ayuda a prevenir ataques de denegación de servicios. Tenemos que alterar el parámetro *LimitRequestBody*. Por defecto está configurado para no tener límite, nosotros podemos ajustar ese número dependiendo de nuestras necesidades. El número que le pongamos viene en bytes.

En nuestro caso le pondremos por ejemplo 2MB de límite, simplemente escribimos en el fichero **apache2.conf**:

LimitRequestBody 2048

Limitar el acceso a archivos por extensión



En nuestro caso, lo que queremos es evitar que terceros no autorizados visualicen los archivos que contienen las directivas de Apache, para ello tenemos que escribir Deny from all para los archivos que contienen esas directivas.

¿Cómo lo hacemos? Pues entramos en apache2.conf y vamos a escribir lo que aparece en la diapositiva siguiente.





Limitar el acceso a archivos por extensión

Por defecto viene configurado como *Require all denied*

Nosotros vamos a dejarlo como sigue:

```
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<FilesMatch "^\.ht">
    Order allow,deny
    Deny from all
</FilesMatch>
```





Limitar el acceso a archivos por extensión

Además vamos a evitar que se publiquen los archivos de copia de seguridad mediante las sentencias:

```
<FilesMatch "(\\.bak$|\\.BAK$)">  
    Order Allow,Deny  
    Deny from all  
</FilesMatch>_
```





Limitar el acceso a archivos por extensión

También vamos a denegar el acceso a todos los directorios CVS (utilizados en sistemas de control de versiones de código fuente) mediante las sentencias:

```
<DirectoryMatch /CVS/>  
    Order Allow,Deny  
    Deny from all  
</DirectoryMatch>_
```



Encriptar ficheros

Para encriptar ficheros, tenemos que ir al archivo **apache2.conf** y añadir las siguientes directivas:

```
<Directory /var/www/html/secret>  
    AuthUserFile /home/santiago/.htpasswd  
    AuthName "Introduzca el usuario y la clave de acceso"  
    AuthType Basic  
    require valid-user  
</Directory>
```

Encriptar ficheros



Una vez hecho esto, vamos a la ruta donde queramos guardar el archivo de claves y ejecutamos *sudo nano .htpasswd*

Después, generamos la clave con:
htpasswd .htpasswd santiago

Ya tenemos nuestro archivo de claves creado y el directorio está encriptado



Repositorio del trabajo



<https://github.com/santidediego/swap1415/tree/master/Trabajo%20de%20la%20asignatura>

¿Qué puedes encontrar?

- La memoria de configuración de seguridad básica de Apache
- Manual de instalación del módulo mod-evasive
- Prueba de evasión de un ataque DoS
- Documento completo sobre como encriptar ficheros

