Manual de Supervivencia del Administrador de Apache

En gnu/Linux, por supuesto

10 de noviembre de 2006

Autor: Miguel Jaque Barbero

Índice de Contenidos

Capítulo 1. Introducción

1.1. Contenido

Este documento no es un manual completo de Apache. Tampoco es una guía para programadores web ni, mucho menos, para quienes deseen colaborar en la programación del Servidor Web Apache. Simplemente es una Guía Rápida de Referencia para quienes tengan la suerte de administrar un Servidor Web Apache.

Naturalmente, este documento contiene errores (y supongo que muchos). Yo ya sé que no soy perfecto, y si tu creías que lo era, lamento defraudarte. Pero puedes ayudarme a mejorarlo enviándome un correo con las erratas, pifias y sugerencias (mjaque@ilkebenson.com). Te lo agradezco de antemano.

Vamos a lo técnico.

Este documento está basado en Apache 2. No hago ninguna referencia a las versiones anteriores de Apache. Para todos los puntos, utilizaré un sistema operativo gnu/Linux. En concreto, Debian 3.1. Tendrás que adaptar lo que hay a la distribución o el sistema operativo que utilices.

1.2. Licencia

1.3. Responsabilidad

No se asume ninguna responsabilidad por los contenidos de este documento. El lector asume el riesgo derivado del uso de los conceptos, ejemplos y cualquier otro contenido. Al tratarse de una nueva edición, este documento puede contener errores e imprecisiones.

1.4. Acerca del Autor

Miguel Jaque Barbero nació en Barcelona en 1968. Es Ingeniero Superior de Telecomunicación por la Universidad Politécnica de Madrid y Máster en Administración de Empresas por el Instituto de Empresa de Madrid.

Ha desarrollado toda su carrera profesional en el sector de la ingeniería de software y, desde 1999 centrado exclusivamente en tecnologías de software libre a través de Ilke Benson (www.ilkebenson.com).

Su actividad profesional se centra desde entonces en el desarrollo de proyectos, formación y consultoría, utilizando exclusivamente estas tecnologías.

Para contactar con el autor: mjaque@ilkebenson.com

Capítulo 2. Introducción

2.1. Tienes Suerte de Utilizar Apache

Sin duda Apache es uno de los mejores servidores web que existen hoy. Para muchos, el mejor.

Desde su creación[1] en 1995 ha dominado el mercado de servidores (ver www.netcraft.com).

Hoy, Apache es desarrollado por una comunidad de voluntarios que tiene su sede en www.apache.org. Allí podrás encontrar toda la documentación detallada.

2.2. Instalación

Con Debian, la instalación de Apache es muy sencilla. Basta con ejecutar el comando:

```
# apt-get install apache2
```

Y tendrás el servidor web instalado.

2.3. Configuración

La configuración de Debian, por defecto, incluye:

Los archivos de configuración de Apache (conf) en `/etc/apache2`

El directorio raíz para los documentos (htdocs) en `/var/www`

2.4. Arranque y Parada

Apache se ejecuta como un demonio. Para arrancarlo basta con ejecutar el comando:

```
# /etc/init.d/apache2 start
```

Y para detenerlo:

```
# /etc/init.d/apache2 stop
```

Echándole un vistazo al script (`/etc/init.d/apache2`) verás que el comando que realmente arranca y detiene Apache es apache2ctl. Consultando la página de manual (`man apache2ctl`) verás que tiene muchas más opciones que start y stop.

2.5. Procesos

Aunque Apache se ejecuta inicialmente como root, pues en un sistema Linux sólo root está autorizado a abrir sockets en puertos por debajo del 1000, y Apache escucha, por defecto, en el tradicional puerto 80 del protocolo HTTP.

Al arrancar Apache el proceso de root arrancará otros procesos hijos que ya se ejecutan con otro usuario y grupo menos peligroso que root. Por defecto estos procesos hijos se ejecutan con el usuario **www-data** y el grupo **www-data**. Asegúrate de que este usuario

1 Para conocer la historia de Apache puedes consultar http://en.wikipedia.org/wiki/Apache_web_server

tiene permiso para leer los documentos que debe servir Apache.

El proceso inicial no atenderá ninguna petición de cliente para evitar así problemas de seguridad. En su lugar, serán los procesos hijos quienes se encarguen de servir las páginas.

Puedes comprobar todo esto con el comando `ps -aux` para ver los procesos en ejecución en tu sistema.

Capítulo 3. Servidor Básico

El fichero de configuración de Apache que Debian instala por defecto (`/etc/apach2/apache2.conf`), aunque es muy bueno, es también muy complicado y no nos sirve para explicar los conceptos más elementales de Apache.

Así que vamos a cambiarlo (guardando antes una copia de seguridad) por este otro:

3.1. Servidor Sencillo

```
#Fichero de Configuración SENCILLO (/etc/apache2/apache2.conf)

#Nombre con el que el Servidor se conoce a sí mismo
ServerName "aym.juntaex.es"

#Directorio con los ficheros de configuración de Apache
ServerRoot "/etc/apache2"

#Directorio raíz de los documentos publicados
DocumentRoot "/var/www"

#Fichero en el que se guarda el número del proceso Apache
PidFile /var/run/apache2.pid

#Usuario y grupo con los que se ejecutará Apache
User www-data
Group www-data

#Fichero de log para los errores
ErrorLog /var/log/apache2/error.log

#Puerto en el que escuchará Apache
Listen 80

#Lista de ficheros que pueden servir como índices de directorio
DirectoryIndex index.html index.htm

#Fichero con la lista de tipos mime
TypesConfig /etc/mime.types

#Tipo de fichero por defecto
DefaultType text/plain
```

Vamos a explicarlo paso a paso.

1. El fichero es un fichero de texto plano.
2. Cada línea contiene una **directiva** de configuración.
3. Las líneas que empiezan por **#** son comentarios.

3.1.1. Directivas Básicas

ServerName

Establece el nombre con el que el servidor se conoce a sí mismo.

Esta directiva se utiliza para las redirecciones. Es decir, cuando Apache le tiene que indicar al cliente (el navegador) otra dirección a la que tiene que ir.

Es importante que ese nombre se pueda resuelver por DNS. De lo contrario el cliente no podrá acceder a la página redireccionada.

ServerRoot

Establece el directorio en el que se encuentran los ficheros de configuración de Apache.

Y si Apache no conoce el directorio de configuración, ¿cómo puede acceder a éste fichero de configuración?

La respuesta a esta paradoja es que Apache puede arrancarse pasándole como parámetro un fichero de configuración. Pero puede ocurrir que en el fichero de configuración hagamos referencia a otros ficheros que deben incluirse.

DocumentRoot

Establece el directorio en el que se encuentran los ficheros que Apache servirá a los clientes (páginas HTML, scripts PHP, CGIs, etc.).

PidFile

Establece el fichero en el que se guardará el número del proceso de Apache.

Este fichero es el que se lee cuando hay que parar/matar el proceso.

User

Establece el usuario con el que se ejecutará Apache.

Bueno, realmente el proceso Apache se ejecuta como root, porque normalmente tiene que abrir un socket de escucha para el puerto 80 y, en POSIX, sólo root puede abrir puertos por debajo del 1000.

Sin embargo, tras arrancar ese primer proceso como root, Apache crea varios procesos hijos que se ejecutan con el usuario establecido en esta directiva (www-data en Debian). Estos procesos serán quienes realmente atenderán las peticiones de los usuarios.

Group

Establece el grupo con el que se ejecutará Apache (sus procesos de escucha).

ErrorLog

Establece el fichero de log de errores de Apache.

En este fichero se registrarán los fallos de acceso, intentos de acceso a recursos sin autorización, páginas no encontradas, etc.

Listen

Establece la dirección IP y el puerto en el que escuchará Apache.

Por defecto, Apache escuchará en todas las direcciones IP habilitadas en la máquina.

DirectoryIndex

Establece los nombres de ficheros que servirán como índices al acceder a un directorio sin indicar ningún recurso concreto.

Así, si un usuario solicita www.miweb.org/dir1/ Apache buscará en ese directorio ficheros con el nombre indicado en esta directiva para entregárselos.

TypesConfig

Establece el fichero con la lista de tipos Mime.

Los tipos Mime constituyen un estándar que relaciona tipos de ficheros con sus extensiones y le permiten a Apache informar al navegador del tipo de fichero que le está entregando. Así, el navegador decide como presentarlo (mostrando una página web, ejecutando un plugin, guardándolo en disco...)

DefaultType

Establece el tipo Mime por defecto para aquellos ficheros cuya extensión no figure en la lista de tipos Mime.

3.2. Servidor Configurado por Directorios

En el servidor sencillo que acabamos de ver, la configuración es la misma para todos los documentos que estemos publicando.

En algunos casos puede interesarnos tener configuraciones diferentes para distintos directorios e incluso para distintos ficheros.

Para eso, podemos utilizar un fichero de configuración como el siguiente:

```
#Fichero de Configuración POR BLOQUES

ServerName "aym.juntaex.es"
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain

DirectoryIndex index.html index.htm

#Establecemos la configuración de cada directorio
<Directory "/var/www">
      #No permitimos los índices automáticos en ningún sitio (salvo
      #los que permitamos explícitamente)
      Options -Indexes
</Directory>

<Directory "/var/www/descargas">
      #En este directorio, permitimos índices automáticos, pero no
      #permitimos enlaces simbólicos
      Options +Indexes -FollowSymLinks
      #No permitimos a nadie acceder a los ficheros .htaccess de este
      #directorio
      <Files .htaccess>
            order allow,deny
            deny from all
      </Files>
</Directory>

<Location /server-status>
      #En esta dirección, mostramos información sobre el estado del
      #servidor
      SetHandler server-status
</Location>
```

3.2.1. Directivas para la Configuración por Bloques

Al aplicar configuraciones diferentes por directorio, por fichero o por localización, decimos que estamos aplicando "Configuración por Bloques".

Hemos utilizado las siguientes directivas:

<Directory>

Indica que el bloque de configuración que abarca (entre <Directory> y </Directory>) se aplica al directorio indicado y a sus subdirectorios.

También hay una directiva <DirectoryMatch> que permite utilizar expresiones regulares. Así, un mismo bloque de configuración puede aplicarse a varios directorios.

Option

Modifica las opciones que se aplican a un directorio.

Las opciones se añaden (con +) o se quitan (con -) respecto a las que en ese momento se aplicasen sobre el directorio.

Mediante esta directiva pueden añadirse/quitarse las siguientes opciones: All, ExecCGI, FollowSymLinks, Includes, IncludesNOEXEC, Indexes, Multiviews y SymLinksIfOwnerMatch.

<File>

Indica que el bloque de configuración que abarca (entre <File> y </File>) se aplicará a los ficheros cuyo nombre coincida con el indicado.

También hay una directiva <FileMatch> que permite utilizar expresiones regulares. Así, un mismo bloque de configuración puede aplicarse a varios ficheros.

<Location>

Indica que el bloque de configuración que abarca (entre <File> y </File>) se aplicará a las localizaciones que coincidan con la indicada.

La "localización" es la dirección que solicita el cliente. Fíjate que no es lo mismo que el sistema de ficheros (sobre el que trabajan <Directory> y <File>). Utilizando redirecciones y alias es posible que un cliente solicite una determinada "localización" y que la página que se le entregue tenga un path completamente distinto al que él especificó.

También hay una directiva <LocationMatch> que permite utilizar expresiones regulares. Así, un mismo bloque de configuración puede aplicarse a varias localizaciones.

SetHandler

Establece el manejador que se utilizará para atender la peticiones a un directorio, a un tipo de ficheros o a una localización.

Un "handler" es una representación interna de Apache de una acción que se va a ejecutar cuando hay una llamada a un fichero. Generalmente, los ficheros tienen handlers implícitos, basados en el tipo de fichero de que se trata. Normalmente, todos los ficheros son simplemente servidos por el servidor, pero algunos tipos de ficheros se tratan de forma diferente.

Los posibles handlers son: default-handler, send-as-is, cgi-script, imap-file, server-info, server-status y type-map.

Capítulo 4. Hosts Virtuales

Tener todo un servidor Apache para atender sólo un sitio web es una pérdida de recursos.

Apache es capaz de atender, desde una sola máquina a todo un conjunto de sitios web. Es decir, podemos servir al mismo tiempo peticiones para aym.juntaex,es, bs.juntaex.es, www.ilkebenson.com, www.apache.org, www.kernel.org...

Esto se hace utilizando “hosts virtuales”.

4.1. Opciones de Arquitectura

Para atender varios sitios web primero debemos conseguir que las peticiones de los clientes para esas URLs lleguen hasta nuestro servidor. Este es un problema de la configuración de DNSs que, como administradores de Apache, no nos corresponde.

Pero, si lo que quieres es hacer pruebas, puedes modificar el fichero hosts de tus clientes (/etc/hosts en Debian) y mapear en él los sitios web con las direcciones IP que escuchen tu/s tarjetas de red.

Una vez hecho esto, tendremos varias opciones para configurar nuestros hosts virtuales.

En primer lugar, podemos establecer varias direcciones IP y asignar una a cada host virtual. Esto lo llamaremos “hosts virtuales por IP”.

En segundo lugar, podemos establecer distintos puertos de escucha para sitio web. Sí, esto es algo complicado porque implica decirle al cliente a qué puerto debe dirigirse. Pero puede ser útil para redes internas. Esto lo llamaremos “hosts virtuales por Puerto”.

Y, en tercer lugar, el protocolo HTTP 1.1 permite al cliente indicarnos, mediante una cabecera, el nombre del sitio web al que quiere acceder. Como ya apenas quedan navegadores que no soporten el protocolo HTTP 1.1, está es la opción más utilizada.

Y por último, podemos hacer una mezcla con todas estas opciones.

4.2. Hosts Virtuales por IP

Supongamos en primer lugar, que nuestro servidor atiende dos direcciones IP y que asignamos cada una de ellas a un sitio web.

El fichero de configuración sería el siguiente:

```
#Fichero de Configuración para HOSTS VIRTUALES POR IP

ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain

DirectoryIndex index.html index.htm

#Establecemos la configuración para cada Host Virtual
<VirtualHost 192.168.2.166>
      #Cada uno tiene su nombre
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"

      #Dividimos los ficheros de log
      ErrorLog /tmp/aym_ERROR.log
      TransferLog /tmp/aym_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.200>
      ServerName "idt.juntaex.es"
      DocumentRoot "/var/www/idt"
      ErrorLog /tmp/idt_ERROR.log
      TransferLog /tmp/idt_ACCESS.log
</VirtualHost>
```

4.3. Hosts Virtuales por Puerto

Hagamos lo mismo, pero en una de las direcciones IP, utilicemos dos puertos TCP/IP distintos para atender dos sitios diferentes.

El fichero de configuración sería el siguiente:

```
#Fichero de Configuración para HOSTS VIRTUALES POR IP Y PUERTO

ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain

DirectoryIndex index.html index.htm

#Indicamos el puerto de cada sitio web
<VirtualHost 192.168.2.166:80>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"
      ErrorLog /tmp/aym_ERROR.log
      TransferLog /tmp/aym_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.166:8080>
      ServerName "intranet.aym.juntaex.es"
      DocumentRoot "/var/www/aym/intranet"
      ErrorLog /tmp/aym_intranet_ERROR.log
      TransferLog /tmp/aym_intranet_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.200>
      ServerName "idt.juntaex.es"
      DocumentRoot "/var/www/idt"
      ErrorLog /tmp/idt_ERROR.log
      TransferLog /tmp/idt_ACCESS.log
</VirtualHost>
```

4.4. Hosts Virtuales por Nombre

Y ahora, configuremos el servidor utilizando los nombres de cada sitio web:

```
#Fichero de Configuración para HOSTS VIRTUALES POR NOMBRE

ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain

DirectoryIndex index.html index.htm

#La configuración de hosts virtuales por nombre se aplica a las
#peticiones recibidas en esta IP
NameVirtualHost 192.168.2.166

#Configuramos haciendo referencia a cada sitio por su nombre
<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"
      ErrorLog /tmp/aym_ERROR.log
      TransferLog /tmp/aym_ACCESS.log
</VirtualHost>

<VirtualHost bs.juntaex.es>
      ServerName "bs.juntaex.es"
      DocumentRoot "/var/www/bs"
      ErrorLog /tmp/bs_ERROR.log
      TransferLog /tmp/bs_ACCESS.log
</VirtualHost>
```

4.5. Hosts Virtuales por Nombre, IP y Puerto

Para completar el ejemplo, configuremos el servidor utilizando todas las opciones:

```
#Fichero de Configuración para Hosts Virtuales

ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain

DirectoryIndex index.html index.htm

NameVirtualHost 192.168.2.166

<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"
      ErrorLog /tmp/aym_ERROR.log
      TransferLog /tmp/aym_ACCESS.log
</VirtualHost>

<VirtualHost bs.juntaex.es>
      ServerName "bs.juntaex.es"
      DocumentRoot "/var/www/bs"
      ErrorLog /tmp/bs_ERROR.log
      TransferLog /tmp/bs_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.200:80>
      ServerName "bs.juntaex.es"
      DocumentRoot "/var/www/bs"
      ErrorLog /tmp/bs_ERROR.log
      TransferLog /tmp/bs_ACCESS.log
</VirtualHost>

<VirtualHost 192.168.2.200:8080>
      ServerName "intranet.bs.juntaex.es"
      DocumentRoot "/var/www/bs/intranet"
      ErrorLog /tmp/bs_intranet_ERROR.log
      TransferLog /tmp/bs_intranet_ACCESS.log
</VirtualHost>
```

4.6. Hosts Virtuales Dinámicos

Pero... ¿qué pasa si tenemos que configurar decenas, cientos o incluso miles de sitios web tal y como ocurre en un ISP?

Para eso tenemos la opción de Hosts Virtuales Dinámicos. Veamos un ejemplo:

```
#Fichero de Configuración para Hosts Virtuales Dinámicos

ServerName "sabio"
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain
ErrorLog /tmp/juntaex_ERROR.log

DirectoryIndex index.html index.htm

#Cargamos el módulo que nos permite hacer configuración dinámica de
#hosts virtuales
Include /etc/apache2/mods-available/vhost_alias.load

#Le indicamos a Apache que utilice como nombre (ServerName) el de la
#petición recibida
UseCanonicalName Off

#Establecemos el DocumentRoot para cada sitio web que atendemos.
VirtualDocumentRoot /var/www/%1.0
```

Pero, si tenemos muchos sitios web, también puede ser útil que cada uno de ellos tenga su propio fichero de configuración (ya le daremos permiso a sus administradores para que lo gestionen ellos mismos).

Esto podemos conseguirlo cargando todos los ficheros de configuración de los sitios virtuales desde un directorio.

Podríamos hacerlo utilizando la siguiente línea en el fichero de configuración de Apache:

```
Include /etc/apache2/sites-enabled/[^.#]*
```

Esta directiva incluye en el fichero de configuración todos los ficheros que encuentre en el directorio indicado.

4.7. Directivas para la Configuración de Hosts Virtuales

Veamos las directivas que hemos utilizado:

<VirtualHost>

Indica que el bloque de configuración que abarca (entre <VirtualHost> y </VirtualHost>) se aplica al sitio web indicado.

Cada host virtual se puede identificar por IP, IP:Puerto o por nombre.

NameVirtualHost

Establece la dirección IP sobre la que se configurarán hosts virtuales.

VirtualDocumentRoot

Establece dinámicamente la raíz de los documentos (DocumentRoot) para los hosts virtuales.

Include

Incluye uno o varios ficheros en el fichero de configuración.

UseCanonicalName

Establece de qué forma conocerá Apache su propio nombre (para las redirecciones).

TransferLog

Establece el fichero de log de acceso de Apache.

En este fichero se registrarán los accesos a las páginas servidas por Apache.

Capítulo 5. Autenticación

En muchas ocasiones querremos restringir el acceso a algunos recursos (páginas). Con Apache podemos establecer mecanismos de usuario y contraseña, para limitar el acceso. Además, los usuarios pueden incluirse en grupos y establecer permisos para estos últimos.

¡PERO CUIDADO! la transmisión de información tiene una encriptación muy débil. No montes ninguno de los mecanismos que veremos a continuación si no utilizas también SSL. Cualquier sniffer podrá fácilmente robarte las contraseñas.

5.1. ¿Cómo funciona?

Una vez que establecemos que un determinado recurso de nuestro servidor requiere autenticación, al intentar acceder a él Apache devuelve al cliente un mensaje 401 (Authentication Required). Normalmente, en los navegadores más actuales, si existe alguna pareja usuario/clave guardada para el recurso la enviarán al servidor de forma automática. Si no es así, preguntarán al usuario.

Desde el lado del servidor, hay dos mecanismos de autenticación.

5.2. Autenticación Básica

Con este mecanismo la contraseña es enviada en claro por la red, o como mucho, con una codificación en base 64 (fácil, fácil...). Veamos un ejemplo:

```
#Fichero de Configuración con Autenticación Básica

ServerName "sabio"
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain
DirectoryIndex index.html index.htm

NameVirtualHost 192.168.0.51
<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"
      ErrorLog /tmp/aym_ERROR.log
      TransferLog /tmp/aym_ACCESS.log
      <Directory /var/www/aym/pac>
            #Establecemos el tipo de Control de Acceso que se
            #utilizará para acceder a este directorio
            AuthType Basic

            #Le damos un nombre al entorno privado que protegemos
            AuthName "Información Privada de la PAC"

            #Fichero con la información de los usuarios autorizados y
            #sus contraseñas (encriptadas).
            AuthUserFile /usr/cursoapache/users

            #Fichero con la información de los grupos y sus usuarios
            #miembros.
            AuthGroupFile /usr/cursoapache/groups

            #Establecemos el nivel de seguridad para este directorio.
            require valid-user
            #Otros valores posibles son:
                  # require user usuario1 usuario2 ... lista con los
                  # usuarios autorizados
                  # require group grupo1 grupo2 ... lista con los
                  # grupos autorizados
                  # require user usu1 usu2 group gr1 gr2 ... una
                  # combinación de ambos
      </Directory>
</VirtualHost>
```

5.2.1. Ficheros de Usuarios y Grupos

La información con los usuarios y las contraseñas se guarda en el fichero señalado por la directiva AuthUserFile. Este fichero tendrá algo parecido a:

```
usu1:$apr1$Ym2fv...$kGAt2g1y6dOSt404Xfka2.
usu2:$apr1$0D0nG/..$cv8eAB0Fykj/PnoIP/0x6.
```

Y, naturalmente, debe estar donde ningún usuario del servidor web pueda llegar a leerlo.

Como poca gente es capaz de escribir directamente la contraseña encriptada, Apache incluye una aplicación para gestionar este fichero. Se trata de `htpasswd2`.

Para crear un fichero como el del ejemplo, tendremos que ejecutar los siguientes comandos:

```
# htpasswd2 -cm /usr/cursoapache/users usu1
# htpasswd2 -m /usr/cursoapache/users usu2
```

El primero crea el fichero y le añade el usuario usu1. La opción c indica que se debe crear un nuevo fichero y la opción m que la contraseña se encriptará con MD5.

El segundo comando simplemente añade el usuario us2 al fichero ya creado.

En ambos casos se nos preguntará por la contraseña.

El fichero de grupos contiene la información de estos, indicando qué usuarios pertenecen a cada grupo. No requiere ningún comando, pues se trata de un simple fichero de texto, con una estructura como la siguiente:

```
pac: usu1 usu2
desarrollo: prog1 prog2
```

Fíjate que los usuarios de cada grupo no se separan por comas (como se hace en /etc/group), sino por espacios en blanco.

5.3. Autenticación Básica con Permisos Especiales (Allow)

Pero en ocasiones necesitamos establecer permisos no solo por usuario, también por dirección IP de origen, subred, etc.

Para eso disponemos de la directiva Allow. Vemos un ejemplo:

```
#Fichero de Configuración con Autenticación Básica y Allow

ServerName "sabio"
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain
ErrorLog /tmp/juntaex_ERROR.log
DirectoryIndex index.html index.htm

NameVirtualHost 192.168.0.51

<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"
      ErrorLog /tmp/aym_ERROR.log
      TransferLog /tmp/aym_ACCESS.log
      <Directory /var/www/aym/pac>
            AuthType Basic
            AuthName "Información Privada de la PAC"
            AuthUserFile /usr/cursoapache/users
            AuthGroupFile /usr/cursoapache/groups
            require valid-user
      </Directory>
      <Directory /var/www/curso/aym/sistemas>
            #Indicamos que sólo puede accederse desde esta subred
            Allow from 192.168.0.0/255.255.0.0
            #También podemos especificar nombres de dominios,
            #direcciones IP, redes (especificación CIDR)...

            #Denegamos el acceso desde una dirección IP determinada
            Deny from 192.168.0.50
            #También podemos especificar nombres de dominios,
            #direcciones IP, redes (especificación CIDR)...

            #Especificamos el orden de aplicación de las directivas
            #anteriores. ¡La última prevalece!
            Order allow,deny
            #¡OJO! el argumento de Order NO LLEVA ESPACIOS.
            #Otros valores posibles son deny,allow y mutual-failure
      </Directory>
</VirtualHost>
```

5.4. Autenticación con Digest

Podemos mejorar algo la seguridad utilizando Digest. En lugar de enviar la contraseña en claro (o con codificación en base 64) por la red, con Digest las contraseñas no viajan por la red.

En lugar de eso, al pedir un recurso protegido, el servidor envía al cliente un número (noonce). Utilizando ese número "único", la URI del recurso solicitado y la contraseña, el navegador realiza un digest. Es decir, un cálculo basado en MD5 que le da un número muy raro y difícil de obtener sin la contraseña (muy difícil).

El servidor comprueba el digest utilizando para ello la información almacenada. Por peculiaridades matemáticas de este tipo de operaciones, el servidor no necesita guardar la contraseña (ni siquiera encriptada). Le basta con guardar otro digest basado en la misma contraseña para poder comprobarlo.

Ciertamente, este mecanismo reduce el riesgo de captura de contraseñas. Veamos un ejemplo:

```
#Fichero de Configuración con Autenticación con Digest

ServerName "sabio"
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain
ErrorLog /tmp/juntaex_ERROR.log
DirectoryIndex index.html index.htm

NameVirtualHost 192.168.0.51

#Cargamos el módulo que necesitamos
Include /etc/apache2/mods-available/auth_digest.load

<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      ServerAdmin "webmaster@aym.juntaex.es"
      DocumentRoot "/var/www/aym"
      ErrorLog /tmp/aym_ERROR.log
      TransferLog /tmp/aym_ACCESS.log
      <Directory /var/www/aym/pac>
            AuthType Digest
            AuthName "pac"
            #Fichero con los Digest de los usuarios autorizados
            AuthDigestFile /usr/cursoapache/digest
            require valid-user
      </Directory>
</VirtualHost>
```

5.4.1. Fichero de Digest

La gestión del fichero de digest en el servidor se realiza con la aplicación htdigest2. Para añadir los mismos usuarios que teníamos, debemos realizar los siguientes comandos:

```
# htdigest -c /usr/cursoxul/digest pac usu1
# htdigest /usr/cursoxul/digest pac usu2
```

De nuevo, la opción c en el primer comando creará el fichero.

Fíjate que es necesario pasarle al comando el nombre del entorno protegido ("pac").

5.5. Autenticación Personalizada

Pero en ocasiones necesitaremos que los propios administradores de los dominios que alberguemos establezcan sus permisos de acceso. Y, naturalmente no queremos que urgen en nuestro fichero de configuración.

Para ello, Apache permite especificar ficheros de configuración adicionales que se guardarán en cada directorio. En cada petición, Apache busca, lee y procesa estos fichero (con la consiguiente pérdida de tiempo).

Veamos un ejemplo reducido sólo a la configuración del servidor virtual:

```
... resto del fichero de configuración de Apache

<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"
      ErrorLog /tmp/aym_ERROR.log
      TransferLog /tmp/aym_ACCESS.log

      #Especificamos el fichero con la configuración particular de
      #cada directorio
      AccessFileName .htaccess

      <Directory /var/www/aym/desarrollo>
            AuthType Basic
            AuthName "desarrollo"
            AuthUserFile /usr/cursoapache/users
            AuthGroupFile /usr/cursoapache/groups
            require group desarrollo

            #Indicamos qué directivas de configuración están
            #autorizadas en el fichero .htaccess
            AllowOverride AuthConfig
            #Otras opciones son All, FileInfo, Indexes, Limit,
            # Options y None o combinación de ellas AllowOverride
            #AutConfig Indexes
      </Directory>
</VirtualHost>
```

En el servidor virtual aym.juntaex.es, en su directorio /desarrollo, podremos para cada uno de sus subdirectorios crear un fichero llamado .htaccess con directivas como Require, AuthType, AuthUserFile, AuthGroupFile, AuthName, etc.

5.6. Directivas para la Autenticación

Veamos las directivas que hemos utilizado:

AuthType

Establece el tipo de autenticación de usuario que se utilizará.

Los valores posibles son Basic y Digest.

AuthName

Establece el nombre para el espacio protegido.

AuthUserFile

Establece el fichero del servidor que guarda la información de usuarios y sus contraseñas encriptadas (para autenticación Básica).

AuthDigestFile

Establece el fichero del servidor que guarda la información de usuarios y sus digests (para autenticación Digest).

AuthGroupFile

Establece el fichero del servidor que guarda la información de grupos de usuarios (para autenticación Básica).

Require

Establece qué usuarios podrán acceder al recurso protegido.

Los valores posibles son: user, group, valid-user o cualquier combinación de ellos. Al utilizar user y/o group, debe añadirse una lista separada por espacios con los nombres de los usuarios y/o los grupos autorizados.

Allow

Controla qué clientes pueden acceder al recurso protegido.

Pueden establecerse controles por IP, subred y nombres de dominios.

Deny

Controla qué clientes no pueden acceder al recurso protegido.

Pueden establecerse controles por IP, subred y nombres de dominios.

Order

Establece en qué orden se aplican las directivas Allow y Deny.

Los valores posibles son “Deny,Allow”, “Allow,Deny” y “Mutual-Failure”. La última prevalece.

AccessFileName

Establece el nombre para los ficheros de configuración distribuidos.

AllowOverride

Establece qué directivas están permitidas en los ficheros de configuración distribuidos.

Los valores posibles son: AuthConfig, FileInfo, Indexes, Limit y Options.

5.7. Otras Opciones de Autenticación

Pero no todo es Basic y Digest. Al menos, en lo que al servidor respecta, éste puede guardar la información de autenticación en otros sistemas además de en simples ficheros de texto plano.

Apache permite guardar la contraseña en bases de datos (tipo DB y DBM), en un directorio LDAP e incluso en bases de datos relacionales como MySQL.

Capítulo 6. Negociación de Contenidos

Apache es capaz de entregar recursos diferentes en función de las preferencias del cliente. Es decir, que para un cliente italiano, Apache puede entregare la versión italiana de un documento y para un Coreano, entregarle la versión coreana (siempre que ambas existan en su directorio de recursos).

Esta adaptación a las preferencias del cliente no se limita al idioma. También puede aplicarse a:

- **Tipo Mime de documento**. Por ejemplo, entregando imágenes en formato jpg para navegadores antiguos y en png para los más modernos.
- **La conjunto de caracteres del recurso (charset)**. Variando entre iso-8859-15 para clientes del sur de Europa e iso-8859-5 para los clientes turcos.
- **La codificación (enconding)**. Variando entre un recurso comprimido como gzip y otro como rar.
- **Y el Idioma**, mencionado anteriormente.

Para conseguir todo esto, Apache dispone de dos mecanismos diferentes.

6.1. Negociación de Contenidos por MultiViews

Este es el sistema más fácil de implementar, pero también el menos potente. Veamos como funciona:

```
<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"

      #Asociamos cada extensión con su idioma
      AddLanguage es .es
      AddLanguage en .en

      #Establecemos las prioridades por si el navegador no nos dice
      #nada.
      LanguagePriority es en
      <Directory /var/www/aym/>
            #Cualquier index.* puede valer como índice
            DirectoryIndex index

            #Activamos la negociación de contenidos
            Options +MultiViews
      </Directory>
</VirtualHost>
```

Ahora, bastará con que en el directorio /var/www/aym pongamos un fichero index para cada idioma. Los ficheros deben llamarse index.html.es, index.html.en, etc.

6.2. Negociación de Contenidos con Type Maps

La segunda forma que tenemos para negociar contenidos es mediante mapas de tipos (type maps), también conocidos como ficheros .var. Vamos con el ejemplo:

```
#Fichero de Configuración para Negociación de Contenidos por Type
#Maps

ServerName "sabio"
ServerRoot "/etc/apache2"
DocumentRoot "/var/www"
PidFile /var/run/apache2.pid
User www-data
Group www-data
ErrorLog /var/log/apache2/error.log
Listen 80
TypesConfig /etc/mime.types
DefaultType text/plain
ErrorLog /tmp/juntaex_ERROR.log
DirectoryIndex index.html index.html

#Establecemos el manejador para los ficheros .var
AddHandler type-map var

NameVirtualHost 192.168.0.51
<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"
      <Directory /var/www/aym/>
            # En este directorio, el índice será un fichero .var
            DirectoryIndex index.var
      </Directory>
</VirtualHost>
```

Este mecanismo es mucho más potente y permitiría incluso negociar plugins.

Veamos ahora como escribimos el fichero **index.var** para que nos permita mostrar los índices en cada idioma.

```
# Establecemos el nombre del fichero y el tipo de variación
URI: index; vary="language"
# también podemos variar por type, charset y enconding

# Si Content-language es es, el fichero será index.es.html
URI: index.es.html
Content-type: text/html
Content-language: es

URI: index.en.html
Content-type: text/html
Content-language: en
```

Los ficheros .var también pueden incluir contenido en línea. Este es un ejemplo, de documento de error con variación:

```
URI: 404; vary="language"

Content-language: es
Content-type: text/html; charset=ISO-8859-1
Body:----------es--
El documento solicitado <b>¡NO ESTÁ!</b>
Por favor contacte con el
<a href="mailto:info@info.info">webmaster</a>
en caso de que usted crea que existe un error en el servidor.
----------es--

Content-language: en
Content-type: text/html; charset=ISO-8859-1
Body:----------en--
The requested document <b>ISN'T HERE!</b>
Please contact
<a href="mailto:info@info.info">webmaster</a>
if you presume there is an error in this server.
----------en--
```

Para activar este documento de error, nos basta con añadir la directiva:

```
ErrorDocument 404 /errores/404.html.var
```

6.3. Directivas para la Negociación de Contenidos

Veamos las directivas utilizadas:

AddLanguage

Asocia una extensión con un determinado idioma (para MultiViews).

LanguagePriority

Establece las prioridades de los idiomas por si el navegador no indica ninguna preferencia.

ErrorDocument

Asocia un documento a un mensaje de error.

Hay otras directivas, como Options y AddHandler que ya habíamos visto en capítulos anteriores.

Capítulo 7. Índices

En ocasiones puede resultar imposible mantener actualizada la página de índice de un directorio. Por ejemplo, si se trata de un directorio de descargas en el que varios usuarios pueden subir información para que otros accedan a ella.

Además, para qué vamos a trabajar haciendo índices en HTML si Apache puede hacerlos por nosotros.

Apache dispone de directivas no solo para realizar índices muy potentes, sino también para personalizarlos añadiendo o quitando opciones.

Veamos como configurarlos:

```
<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/curso/aym"
      <Directory /var/www/curso/aym/formularios>
            #Establecemos índices bonitos para este directorio
            IndexOptions FancyIndexing

            #Añadimos una descripción para algunos recursos
            AddDescription "Formulario de Contacto" contacto.html

            #Hacemos que algunos elementos no aparezcan en el índice
            IndexIgnore *.jpg
            IndexIgnore ..
      </Directory>
</VirtualHost>
```

7.1. Directivas para la Configuración de Índices

Hay muchas directivas que nos permitirán configurar los índices. Aquí hay algunas:

IndexOptions

Establece las opciones para los índices. Pueden ser, entre otras: DescriptionWidth, FancyIndexing, FoldersFirst, IconsAreLinks, IconHeight, IconWidth, IgnoreCase, ScanHTMLTitles ...

IndexIgnore

Establece que ficheros no aparecerán en el índice.

AddDescription

Añade una descripción para uno o varios ficheros del índice.

AddIcon

Asocia un icono a un fichero o tipo de ficheros

Capítulo 8. Redirecciones

En ocasiones es necesario mover las cosas de sitio. Hay que mover directorios de un sitio a otro, trasladar ficheros ... y queremos que nuestros clientes sigan accediendo como lo hacían antes.

Para conseguirlo tenemos tres mecanismos:

8.1. Redirecciones con Alias

Alias nos permite dar acceso a recursos que no están en la dirección indicada por la directiva DocumentRoot.

Por ejemplo,

```
<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"

      #Damos acceso a un directorio que NO está en DocumentRoot
      Alias /bienestarsocial /var/www/bs

      #También podemos crear accesos a directorios de DocumentRoot
      Alias /proy1 /var/www/aym/desarrollo/proy1/test

</VirtualHost>
```

8.2. Redirecciones con Redirect

Redirect asocia una URL antigua con una nueva. La nueva URL es notificada al cliente para que realice la conexión con ella.

Por ejemplo:

```
<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/curso/aym"

      #Redireccionamos a los clientes a la nueva web
      Redirect /pac http://www.agralia.es
</VirtualHost>
```

8.3. Rewrite

Todo lo anterior, y mucho más, podemos hacerlo con Rewrite.

Rewrite es un módulo muy extenso y completo que, básicamente, aplica un patrón a una URL y realiza sustituciones en ella.

Por ejemplo:

```
#Cargamos el módulo que necesitamos
Include /etc/apache2/mods-available/rewrite.load

NameVirtualHost 192.168.0.51
<VirtualHost aym.juntaex.es>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/curso/aym"

      #Activamos el motor de rewrite
      RewriteEngine on

      #Establecemos el fichero de log
      RewriteLog /tmp/log_rewrite

      #Establecemos el nivel de log
      RewriteLogLevel 9

      #La dirección aym.juntaex.es/info/ la redirigimos a agralia
      RewriteRule ^/info/$ http://www.agralia.es
</VirtualHost>
```

Pero hay reglas de rewrite mucho más complicadas y útiles. Por ejemplo, imaginemos que tenemos una página web cuyo contenido depende de un parámetro (típico de páginas PHP, Perl, etc.). El problema es que ni a los seres humano ni tampoco a los buscadores les gusta recordar una URL del tipo http://aym.juntaex.es/prog.php?opcion=1. Preferimos las URLS directas.

¿Qué podemos hacer?

Podemos reescribir las peticiones que nos lleguen a una dirección "fácil". Por ejemplo, si la opción 1 nos da información de la PAC y la opción 2 nos da información del Tabaco, podemos crear dos direcciones "ficticias" que sean http://aym.juntaex.es/pac/ y http://aym.juntaex.es/tabaco/ y poner en el fichero de configuración la siguientes directivas:

```
RewriteRule ^/pac/$ prog.php?opcion=1 [L]
RewriteRule ^/tabaco/$ prog.php?opcion=2 [L]
```

Fíjate en la opción [L] al final. Indica al proceso que es una regla final. Si se aplica, no se seguirán aplicando más reglas.

Las reglas de rewrite se ejecutan en cascada. La URL de entrada para la segunda regla es la salida de la primera. Esto nos permite utilizar reglas encadenadas.

Las opciones son casi ilimitadas: podemos pasar partes de la URL como parámetros, podemos establecer condiciones, cadenas de reglas... Cualquier problema de redirección, por complicado que sea, puede resolverse con este módulo.

8.4. Directivas para la Configuración de Redirecciones

Hemos utilizado las siguientes directivas:

Alias

Asocia URLs con ficheros del sistema.

Redirect

Envía una redirección externa indicándole al cliente que cargue otra URL.

RewriteEngine

Activa/Desactiva el motor de rewrite.

RewriteLog

Establece el fichero de log para el motor de rewrite.

RewriteLogLevel

Establece el nivel de depuración para el motor de rewrite.

RewriteRule

Establece una regla para el motor de rewrite.

Capítulo 9. Contenidos Dinámicos con CGI

Ciertamente, hoy no podemos limitar nuestro web a los contenidos estáticos de HTML. Necesitamos conectar con bases de datos, ofrecer contenidos personalizados a cada usuario, generar informes al momento, vender, tramitar...

Para todo esto necesitamos que nuestro servidor web genere los contenidos dinámicamente. Y, afortunadamente, existen muchas tecnologías que nos facilitan la vida.

9.1. ¿Qué es CGI?

CGI son las siglas de **Common Gateway Interface**. Se trata de una de las herramientas más antiguas para generar contenidos dinámicos en el web.

CGI proporciona un interfaz entre el servidor web y las aplicaciones que generarán dinámicamente los contenidos. Se trata de un interfaz antiguo y muy conocido, pero algo ineficiente. De todas formas, merece la pena que le echemos un vistazo.

9.2. Ejemplo de CGI

A través de CGI podemos ejecutar cualquier programa del sistema operativo del servidor (configurando los permisos adecuadamente). Será el programa llamado quien se encargará de generar el HTML (o lo que queramos) que el servidor web entregará al cliente.

Para nuestro ejemplo, vamos a crear un script de bash muy sencillito:

```
#!/bin/bash
echo "Content-Type: text/plain"
echo
echo "Hola Mundo. Soy un CGI"
```

¡Cuidado! En un cgi todo es importante. Hay que especificar la cabecera, hay que dejar una línea en blanco después de las cabeceras...

También tenemos que hacer el script ejecutable para el usuario de Apache. Para ello, crearemos un directorio cd cgi en /usr/cursoapache/cgi-bin. En él guardaremos el script anterior y le haremos ejecutable. Bastará con ejecutar el comando:

```
#> chmod +x /usr/cursoapache/cgi-bin/ejemplo.cgi
```

Es importante dejar los cgis en un directorio que no pertenezca al árbol de documentos de Apache (DocumentRoot). De esta forma evitamos que, por error, alguien pueda acceder al código y aprovechar las debilidades de nuestros programas.

Nos resta decirle a Apache que tiene que ejecutar los programas de ese directorio como si fueran cgis (que es lo que son).

Para ello, realizamos el siguiente fichero de configuración:

```
#Fichero de Configuración para CGIs

ServerName "localhost"
ServerRoot "/etc/apache2"
PidFile /var/run/apache2.pid
TypesConfig /etc/mime.types
DefaultType text/plain
User www-data
Group www-data
ErrorLog /tmp/apache_ERROR.log
TransferLog /tmp/apache_ACCESS.log
DocumentRoot "/var/www"
Listen 80
DirectoryIndex index.html index.html

# Incluimos el módulo que necesitamos para ejecutar CGIs
Include "/etc/apache2/mods-available/cgi.load"

# Establecemos el directorio que contendrá programas CGI
ScriptAlias /cgi-bin/ /usr/cursoapache/cgi-bin/
```

Ahora, si vamos a http://localhost/cgi-bin/ejemplo.cgi, podremos ver nuestro programa funcionando.

9.3. Configuración de CGIs para Usuarios

La configuración anterior se aplica a todo el servidor web. En ocasiones, en muy pocas ocasiones, necesitaremos configurar el servidor de modo que nuestros usuarios puedan subir y ejecutar programas CGI.

¡Esto es un enorme problema de seguridad!

Así que, lo mejor es no hacerlo. Pero, si no tienes más remedio, si confías en tus usuarios, en que no cometerán errores en sus programas, en que serán cuidadosos con su uso... entonces puedes configurar el servidor de la siguiente forma:

```
#Fichero de Configuración para CGIs II

ServerName "localhost"
ServerRoot "/etc/apache2"
PidFile /var/run/apache2.pid
TypesConfig /etc/mime.types
DefaultType text/plain
User www-data
Group www-data
ErrorLog /tmp/apache_ERROR.log
TransferLog /tmp/apache_ACCESS.log

Listen 80
DirectoryIndex index.html index.html

Include "/etc/apache2/mods-available/cgi.load"
DocumentRoot "/var/www"

<VirtualHost 192.168.0.51>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"

      #Permitimos ejecutar CGIs en esta localización
      <Location /cgi >
            AddHandler cgi-script cgi
            Options +ExecCGI
      </Location>
</VirtualHost>
```

Con esta configuración, el administrador del dominio virtual puede crear sus propios CGIs e instalarlos en su directorio /cgi (que correponde con /var/www/aym/cgi si no se utiliza ningún Alias).

Pero, antes de hacerlo... ¿Estás seguro de que quieres hacer algo así? ¿No hay otra alternativa?

9.4. Directivas para la Configuración de CGI

Hemos utilizado las siguientes directivas:

ScriptAlias

Asocia una localización con un directorio del sistema de ficheros y establece su contenido como CGIs.

Options +ExecCGI

Permite la ejecución de CGIs para el directorio.

Capítulo 10. Server Side Includes

Naturalmente hay otras opciones para generar contenidos dinámicos. Una de las menos peligrosas, y también de las menos potentes, es utilizar SSI (Server Side Includes).

Mediante esta tecnología, Apache parsea los documentos HTML antes de enviárselos al cliente. En este parseo, detecta unas marcas especiales y las sustituye por los valores que correspondan.

De esta forma, la página final mezcla su contenido estático con algunos campos variables. Estos pueden ser:

- Valores de Variables.
- Salidas de Comandos.
- Tamaño de un Fichero.
- Fecha de Última Modificación de un Fichero.

También es posible mediante SSI incluir unos ficheros en otros.

Sintaxis de SSI

Las directivas de SSI, esas que luego serán reescritas, tiene la sintaxis de un comentario de HTML. Es decir

```
<!--#nombre de la directiva parámetros -->
```

Por ejemplo, la siguiente directiva se traducirá en la fecha de modificación del fichero indicado

```
<!--#flastmod virtual="fichero.html"-->
```

Donde flastmod es el nombre de la directiva, virtual es el nombre del parámetro y "fichero.html" su valor.

La extensión de los ficheros SSI suele ser .shtml

10.1. Ejemplo de SSI

Vamos a escribir un pequeño ejemplo de SSI:

```
<!--#config errmsg=" Error de SSI "-->
<!--#config sizefmt="bytes"-->
El tamaño de este fichero es <!--#fsize file="ssi.shtml"-->bytes.
```

Para poder ejecutarlo, establecemos este fichero de configuración de Apache:

```
#Fichero de Configuración para SSI

ServerName "localhost"
ServerRoot "/etc/apache2"
PidFile /var/run/apache2.pid
TypesConfig /etc/mime.types
DefaultType text/plain
User www-data
Group www-data
ErrorLog /tmp/apache_ERROR.log
TransferLog /tmp/apache_ACCESS.log

Listen 80
DirectoryIndex index.html index.html

DocumentRoot "/var/www"

#Incluimos el módulo que necesitamos
Include "/etc/apache2/mods-available/include.load"

#Asociamos la extensión shtml al manejador de SSI
AddHandler server-parsed shtml

#Establecemos que se pueden ejecutar en todos los directorios
#(por defecto suele estar deshabilitado)
Options +Includes
```

Ahora, si reiniciamos Apache y pedimos http://localhost/ssi.shtml podremos ver la salida parseada.

10.2. Ejemplo Avanzado de SSI

SSI tiene mucho más. Nos permite incluir ficheros, definir variables e incluso utilizar estructuras de control como if, then, else.

Vamos a ver un ejemplo en el que utilizamos SSI para definir un mismo pie para todas las páginas de nuestro website.

En primer lugar, creamos un fichero SSI con la información que queremos mostrar:

```
<!--  Evitamos que el usuario vea los errores.
      Ya los veremos en el log-->
<!--#config errmsg="<!-- Error de SSI -->"-->

<!-- Configuramos el formato de la fecha según strfmt -->
<!--#config timefmt="%A, %e de %B de %Y a las %H:%M:%S"-->

<hr/>

<!-- Utilizando echo podemos ver el contenido de variables-->
<p>CopyRight Ilke Benson 2006 - <!--#echo var="DOCUMENT_URI"--> -
Modificado el <!--#echo var="LAST_MODIFIED"-->.</p>
<!--#echo var="DATE_LOCAL"-->
```

Este fichero lo guardaremos en /usr/cursoapache/ssi con el nombre pie.shtml. Ahora lo enlazaremos en el resto de páginas de nuestro website.

Para ello, cambiamos nuestro index.html por index.shtml:

```
<html>
      <head><title>Consejería de Agricultura y
Medioambiente</title></head>
      <body>
            <h1>Consejería de Agricultura y Medioambiente</h1>
            Bla, bla, bla...

            <!-- La directiva include nos permite incluir ficheros-->
            <!--#include virtual="/include/pie.shtml"-->
      </body>
</html>
```

Por último, sólo nos resta configurar adecuadamente nuestro servidor Apache:

```
#Fichero de Configuración para SSI General

ServerName "localhost"
ServerRoot "/etc/apache2"
PidFile /var/run/apache2.pid
TypesConfig /etc/mime.types
DefaultType text/plain
User www-data
Group www-data
ErrorLog /tmp/apache_ERROR.log
TransferLog /tmp/apache_ACCESS.log

Listen 80
DirectoryIndex index.shtml index.html index.html

DocumentRoot "/var/www"

Include "/etc/apache2/mods-available/include.load"
AddHandler server-parsed shtml

#Creamos un alias para encontrar los includes
Alias /include /usr/cursoapache/ssi

<VirtualHost 192.168.0.51>
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/aym"
</VirtualHost>
```

Si ahora vamos con nuestro navegador a http://aym.juntaex.es/ veremos el pie que hemos creado.

Capítulo 11. Logging

Los ficheros de log son un instrumento crítico para la seguridad de nuestro servidor. En este capítulo veremos cómo configurarlos para que Apache guarde la información que necesitamos, tanto para mejorar su rendimiento como para resolver problemas a los usuarios, conocer a qué secciones acceden más, intentos de acceso no autorizados, etc.

Pero no basta con configurar correctamente los logs para que guarden la información que queremos. También tenemos que consultarla periódicamente. Sino, cualquier información que pudiéramos registrar será inútil.

Existen muchos programas que te pueden ayudar a inspeccionar y gestionar los logs de Apache. Puedes encontrarlos en los paquetes apache2-utils, scanerrlog, visitors, vlogger, webdruid...

11.1. Ficheros de Log

Aunque ya los hemos utilizado, vamos a mencionarlos explícitamente.

Apache utiliza dos ficheros de log:

- **ErrorLog** – Para registrar los errores que se producen al acceder a los recursos de Apache.
- **TransferLog** – Para registrar los accesos que no producen error.

11.1.1. ErrorLog

El contenido de ErrorLog es algo parecido a esto:

```
[Tue Nov 21 11:20:14 2006] [notice] caught SIGTERM, shutting down
[Tue Nov 21 11:20:15 2006] [notice] Apache/2.0.55 (Debian) configured
      -- resuming normal operations
[Tue Nov 21 11:20:18 2006] [error] [client 127.0.0.1] unknown
     directive "fsze="ssi.shtml"" in parsed doc /var/www/ssi.shtml
[Tue Nov 21 12:12:36 2006] [error] [client 192.168.0.51] File does
not exist: /var/www/curso/aym/no_existo
```

En él podemos ver los mensajes de cierre de Apache, de reinicio, errores de directivas SSI, páginas no encontradas, etc.

11.1.2. TransferLog

El contenido de TransferLog será parecido a esto:

```
192.168.0.51 - - [21/Nov/2006:10:22:32 +0100] "GET /cgi/aym.cgi
                 HTTP/1.1" 200 51
127.0.0.1 - - [21/Nov/2006:11:07:30 +0100] "GET /ssi.shtml HTTP/1.1"
                 200 130
192.168.0.51 - - [21/Nov/2006:11:43:20 +0100] "GET /index.shtml
                 HTTP/1.1" 200 348
192.168.0.51 - - [21/Nov/2006:12:12:36 +0100] "GET /no_existo
           HTTP/1.1" 404 207
```

En él podemos ver desde qué dirección se ha accedido a nuestro servidor, en que momento, qué petición se recibió, resultado, etc.

11.2. Directivas para la Configuración de Logs

Podemos utilizar las siguientes directivas para la configuración de los ficheros de log:

ErrorLog

Establece el fichero que para el log de errores.

Opcionalmente puede ser un programa que, por ejemplo, registre los errores en una base de datos.

LogLevel

Establece el nivel de detalle para el log de errores.

Los valores posibles son: emerg, alert, crit, error, warn, notice, info y debug.

TransferLog

Establece el fichero que para el log de accesos.

Opcionalmente puede ser un programa que, por ejemplo, registre los accesos en una base de datos.

LogFormat

Define un formato de log para el log de accesos o para un log personalizado.

CustomLog

Establece el fichero y el formato para un log personalizado.

Capítulo 12. PHP

Vale, hay muchas formas de generar contenidos dinámicos en un servidor web. Pero la que hoy en día se lleva la palma es PHP.

PHP es un lenguaje de script, con orientación a objetos (especialmente desde la versión 5), con gran cantidad de librerías funcionales (conexión con bases de datos, tratamientos de XML, funciones matemáticas, generación de PDFs...) y una dilatada carrera como lenguaje de programación para aplicaciones web.

Así que, no podemos pasarlo por alto.

12.1. Instalación y Configuración

Para poder ejecutar programas PHP en tu servidor Apache primero debemos configurarlo correctamente. Necesitamos un paquete Debian que instalaremos con el siguiente comando:

```
#> apt-get install libapach2-mod-php5
```

Este comando instalará en nuestro ordenador, entre otros, los ficheros /etc/apache2/mods-available/php5.load y /etc/apache2/mods-available/php5.conf. Conviene echar un vistazo a su contenido para entender su funcionamiento.

Estos son ambos ficheros comentados:

```
#Contenido de php5.load

#Cargamos el módulo que interpretará los ficheros php
LoadModule php5_module /usr/lib/apache2/modules/libphp5.so
```

```
#Contenido de php5.conf

#Si el módulo está cargado...
<IfModule mod_php5.c>
  #Asociamos las extensiones con el tipo Mime de php
  AddType application/x-httpd-php .php .phtml .php3
  AddType application/x-httpd-php-source .phps
</IfModule>
```

Como ves, no son ficheros complicados, así que puedes incluirlos en cualquier otra configuración que tengas para tu servidor.

12.2. Comprobando que Funciona

Vamos ahora a crear una sencilla página php para comprobar que todo funciona.

Este es el código:

```
<?php
      echo “Hola Mundo”;
?>
```

Salvemos ahora la página como holamundo.php (dentro de nuestro árbol DocumentRoot) y, tras reiniciar Apache, podremos acceder con nuestro navegador a esa dirección. La página nos mostrará el mensaje “Hola Mundo”.

Fíjate que, si tienes algún error en la configuración, Apache no sabrá qué hacer con el fichero de extensión PHP. Si tienes el DefaultType a text/plain, verás en tu navegador el código de PHP. Si lo tienes a text/html, no verás nada (salvo que mires el código fuente de la página), las marcas de PHP son ignoradas por el navegador cuando se interpretan como PHP.

Naturalmente no podemos meter en este Manual de Supervivencia todo un curso de PHP. Puedes encontrar toda la información que necesitas sobre PHP en www.php.net.

Pero si podemos aprovechar la ocasión para aprender un poco.

12.3. Practicando con PHP

Vamos a preparar un ejemplo más complicado. Hagamos el típico ejercicio de presentar en la pantalla los números primos que hay entre el 1 y el 100.

En este ejercicio necesitaremos utilizar estructuras de control (bucles for e if then) y utilizaremos funciones que guardaremos en un fichero externo.

Este es el código de primos.php

```
<?php
      require_once("/var/www/lib.php");

      $html = "";
      $html .= "<html><body>";
      $html .= "<h1>Calculando N&uacute;meros Primos</h1>";

      for ($i=1; $i<100; $i++){
            if(es_primo($i))
                  $html .= $i.", ";
      }

      $html .= "</body></html>";
      echo $html;
?>
```

Como ves, este programa carga un fichero con otras funciones auxiliares. En este caso, el fichero lib.php contiene la función es_primo(). Fíjate que la instrucción require_once garantiza que el fichero se carga sólo una vez, aunque sea llamado en varios sitios (por ejemplo, porque incluimos ficheros que incluyen ficheros que incluyen ficheros...)

El código de lib.php es el siguiente:

```
<?php
      function es_primo($num){
            for ($i=2; $i<$num; $i++)
                  if ($num % $i == 0) return false;
            return true;
      }
?>
```

12.4. Manejo de Formularios

Pero el uso más habitual de PHP es para crear aplicaciones web. En ellas, necesitamos que el usuario nos envíe información, procesarla y devolverle un resultado.

Vamos a ver, con el siguiente ejemplo de login, como nos llegan las variables de usuario a php y como las leemos.

En primer lugar, el usuario introduce los valores que nos quiere enviar en un formulario HTML que nos envía al programa PHP (especificándolo en el atributo action del elemento form). Este es el código:

```
<html>
      <head>
            <title>Login del Ejercicio de PHP</title>
      </head>
      <body>
            <form action="login.php" method="POST">
                  Nombre: <input name="nombre" type="text"/><br/>
                  Clave: <input name="clave" type="password"/><br/>
                  <input type="submit" value="Aceptar"/>
            </form>
      </body>
</html>
```

Fíjate en los nombres que damos a los campos; nos servirán para acceder a las variables al recibir el formulario. Esto lo hacemos en el fichero login.php

```
<?php
      $html = “<html><body>”;
      $html .= 'Has hecho login<br/>';
      $html .= "El nombre es ".$_REQUEST['nombre']."<br/>";
      $html .= "La clave es ".$_REQUEST['clave']."<br/>";

      echo $html;
?>
```

Como ves, la información que nos envía el usuario se recibe en un array indexado (un array que, en lugar de utilizar un índice numérico, utiliza nombres). El array indexado es $_REQUEST.

Salva ambos ficheros en el mismo directorio y comprueba su funcionamiento.

12.5. Evitando el Caos

Al programar en PHP podemos encontrarnos fácilmente metidos en un buen lío. Si no somos cuidadosos, nos encontraremos con funcionalidad distribuida entre varios ficheros (como en el caso anterior), código de distintos lenguajes en un mismo fichero, etc.

Vamos a simplificar nuestro ejemplo anterior. Para ello, incluiremos el código del formulario en el mismo fichero que el código de la respuesta al formulario.

```
<?php
      poner_cabecera();
      if (!isset($_REQUEST['nombre'])){
            $html = '<form action="login.php" method="POST">';
            $html .= 'Nombre: <input name="nombre"
                        type="text"/><br/>';
            $html .= 'Clave: <input name="clave"
                        type="password"/><br/>';
            $html .= '<input type="submit" value="Aceptar"/>';
            $html .= '</form>';
      }
      else{
            $html = 'Has hecho \'login\'&amp;apos;<br/>';
            $html .= "El nombre es ".$_REQUEST['nombre']."<br/>";
            $html .= "La clave es ".$_REQUEST['clave']."<br/>";
      }
      echo $html;
      poner_pie();

      function poner_cabecera(){
            echo "<html><body>";
      }
      function poner_pie(){
            echo "</body></html>";
      }
?>
```

En este ejemplo, hemos creado dos funciones para poner el pie y la cabecera de las páginas. Normalmente estas funciones estarán en un fichero externo (que cargaremos con include o require_once) y serán compartidas por otras páginas.

Además, el programa tiene dos partes correspondientes a la estructura if then.

En la primera, se muestra el formulario de login. En la segunda, se muestra la respuesta al formulario.

Decidimos si ejecutar una u otra en función de que exista (isset) la variable $_REQUEST.

Capítulo 13. Aplicaciones Web con LAMP

LAMP son las siglas de Linux-Apache-MySQL-PHP y es, sin duda, la arquitectura más utilizada hoy para la programación de aplicaciones web.

Ya hemos visto como funcionan conjuntamente Linux, Apache y PHP. Así que sólo nos queda ver cómo integramos la Base de Datos y accedemos a ella.

13.1. Instalación y Configuración de MySQL

Hablamos de Debian, así que la instalación de MySQL tendrá que ser fácil. En concreto, tenemos que instalar dos paquetes. El primero es el propio servidor MySQL, y el segundo es la librería de funciones de PHP para acceder a MySQL.

El comando para hacer todo esto (en menos de 5 segundos) es el siguiente:

```
#> apt-get install mysql-server php5-mysql
```

No necesitamos ninguna configuración particular, así que vamos a crear un nueva base de datos llamada “curso” con un usuario para acceder a ella y una tabla con algunas columnas.

Para ello, escribimos el siguiente fichero SQL:

```
DROP DATABASE IF EXISTS curso;
CREATE DATABASE curso;
GRANT SELECT, INSERT, UPDATE, DELETE ON curso.* TO curso IDENTIFIED
BY "clave_curso";

USE curso;

DROP TABLE IF EXISTS usuario;
CREATE TABLE usuario (
      id INT AUTO_INCREMENT,
      usuario VARCHAR(20) UNIQUE,
      clave VARCHAR(100),
      categoria VARCHAR(50),
      correo VARCHAR(100),
      PRIMARY KEY(id)
);
```

Guardamos el fichero como curso.sql y lo ejecutamos con el siguiente comando:

```
$> mysql -u root < curso.sql
```

Tendremos creada nuestra base de datos.

De nuevo, no estamos en un curso de MySQL. Así que tendrás que consultar la documentación en www.mysql.com si necesitas actualizar tu SQL.

13.2. Conectando con la Base de Datos

Como primer ejercicio, conectemos nuestro usuario "curso" con la base de datos.

Para ello, escribimos el siguiente programa PHP:

```
<?php
      echo "Me voy a conectar a MySQL<br/>";

      $conexion = mysql_connect('localhost', 'root', '');

      if (!$conexion) {
            echo "Fall&oacute; la conexi&oacute;n a la base de
                  datos.".mysql_error();

            exit();
      }
?>
```

En este ejemplo realmente lo que estamos haciendo es "conectar" con el servidor de base de datos. Pero no basta con eso, necesitamos especificar cuál de todas las bases de datos del servidor es la que vamos a utilizar.

Eso lo hacemos con la instrucción: mysql_select_db(nombre_db).

13.3. Ejecutando Sentencias SQL

Ejecutar sentencias sencillas, como INSERT, DELETE o UPDATE que no devuelven ningún resultado, es fácil.

Basta con conectarse con el servidor de base de datos, seleccionar la base de datos con la que queremos operar y lanzar la sentencia SQL.

Este es un ejemplo:

```
<?php

      mysql_connect('localhost','curso','clave_curso');
      mysql_select_db('curso');
      $consulta = "DELETE FROM usuario WHERE id=".$_REQUEST['id'];

      if (mysql_query($consulta))
            echo "Se borr&oacute; el registro";
      else
            echo "Fall&oacute; el borrado.<br/>".mysql_error();
?>
```

13.4. Insertando Datos

Vamos a utilizar lo que ya sabemos de PHP para hacer un formulario que inserte datos en la tabla de la base de datos:

```
<?php
      poner_cabecera();
      if (!isset($_REQUEST['usuario'])){
            $html = '<form action="insertar.php" method="POST">';
            $html .= 'Id: <input name="id" type="text"/><br/>';
            $html .= 'Usuario: <input name="usuario" type="text"/><br/>';
            $html .= 'Clave: <input name="clave" type="password"/><br/>';
            $html .= 'Repita la Clave: <input name="clave2"
                  type="password"/><br/>';
            $html .= 'Nombre Completo: <input name="nombre"
                  type="text"/><br/>';
            $html .= '<input type="submit" value="Insertar"/>';
            $html .= '</form>';
      }
      else{
            $conexion = mysql_connect('localhost', 'root', '');
            if (!$conexion){
                  echo "Se fastidi&oacute;";
                  exit();
            }
            mysql_select_db('test');
            $consulta = 'INSERT INTO usuario ';
            $consulta .= 'SET id='.$_REQUEST['id'].',';
            $consulta .= ' usuario="'.$_REQUEST['usuario'].'",';
            $consulta .= ' clave=PASSWORD("'.$_REQUEST['clave'].'"),';
            $consulta .= ' nombre="'.$_REQUEST['nombre'].'"';


            if(!mysql_query($consulta)){
                  echo "La consulta ha fallado: ".mysql_error();
                  exit();
            }
            else
                  $html = "El registro se insert&oacute; con &eacute;xito";

      }
      echo $html;
      poner_pie();

      function poner_cabecera(){
            echo "<html><body>";
      }
      function poner_pie(){
            echo "</body></html>";
      }
?>
```

La instrucción mysql_query ejecuta la consulta SQL. Para sentencias INSERT, UPDATE o DELETE es fácil. Pero la cosa se complica cuando tenemos que ejecutar una sentencia SELECT y recorrer el conjunto de resultados que generan.

13.5. Consultas de SELECT

Veamos un ejemplo más completo:

```
<?php

	poner_cabecera();
	$conexion = mysql_connect('localhost', 'root', '');
	if (!$conexion){
		echo "Se fastidiÃ³";
		exit();
	}
	mysql_select_db('test');
	$consulta = 'SELECT * FROM usuario';

	$resultado = mysql_query($consulta);
	if(!$resultado){
		echo "La consulta ha fallado: ".mysql_error();
		exit();
	}

	$html = '<table border="1">';
	while($fila = mysql_fetch_array($resultado)){
		$html .= '<tr>';
		$html .= '<td>'.$fila['id'].'</td>';
		$html .= '<td>'.$fila['usuario'].'</td>';
		$html .= '<td>'.$fila['clave'].'</td>';
		$html .= '<td>'.$fila['nombre'].'</td>';
		$html .= '<td>';
		$html .= '<a
			href="http://localhost/borrar.php?id='.$fila['id'];
		$html .= '">Borrar Usuario</a></td>';
		$html .= '</tr>';
	}
	$html .= "</table>";

	echo $html;
	poner_pie();

	function poner_cabecera(){
		echo "<html><body>";
	}
	function poner_pie(){
		echo "</body></html>";
	}
?>
```

Capítulo 14. Secure Sockets Layer (SSL)

SSL es un protocolo criptográfico de comunicación utilizado para garantizar la identidad y la privacidad de las comunicaciones web.

Técnicamente se corresponde con la capa de transporte del Modelo ISO, por eso también es conocido como TLS (Transport Layer Security).

14.1. Como Funciona SSL

- Cuando el navegador solicita una página SSL al servidor le envía un mensaje "ClientHello" con información de "handshake". En él figuran los algoritmos de cifrado soportados, la máxima versión de SSL soportada y varios números aleatorios que serán utilizados en la comunicación.
- El navegador sabe que una página es segura porque su URL establece el protocolo https (P.ej. https://cajabadajoz.es).
- El servidor responde con un mensaje "ServerHello" en el que elige los parámetros de la comunicación entre los ofrecidos por el cliente.
- Ahora que ya están de acuerdo, el servidor su certificado al navegador, generalmente del tipo X.509.
- Existe la posibilidad de que el servidor solicite al navegador su propio certificado. Esto serviría para identificar al cliente. Pero no es lo habitual. En la comunicación SSL, solo está identificado el servidor.
- Por último, ambos negocian una clave secreta "master secret" que suele ser una clave elegida por el navegador y encriptada por éste con la clave pública del servidor. Esta será la clave que ambos utilicen en sus comunicaciones.

14.2. Instalación

La instalación de versiones anteriores a Apache 2 eran bastante complicadas. Ahora, con Apache 2 y especialmente con Debian, la instalación es realmente sencilla... ¡No hay que instalar nada!

Apache 2 ya incluye soporte para SSL.

14.2.1. Generando el Certificado

El único "problemilla" es que necesitamos un certificado para nuestro servidor.

Para eso, recurrimos al comando apache2-ssl-certificate. Al ejecutarlo, nos planteará algunas preguntas para la información del certificado y generará dos archivos en el directorio /etc/apache2/ssl

Ya tenemos nuestro certificado.

Lo que hemos hecho es generando nuestro propio certificado de servidor, lo

cual es cómodo para un curso como este pero difícilmente creíble si pretendemos utilizarlo en un web público.

Si quieres un certificado "serio" puedes acudir a cualquier empresa especializada (Thawte, CertiSign, VeriSign o la propia FNMT).

14.3. Configuración

La configuración es igual que todo lo que hemos visto anteriormente... salvo para las directivas propias de SSL.

Por cierto, si vas a utilizar SSL con HostsVirtuales, tendrás que asignar a cada host una IP (configuración de hosts virtuales por IP ver 4.2), no puedes hacerlo por nombres.

14.4. Ejemplo

Veamos un ejemplo de configuración:

```
#Fichero de Configuración para Hosts Virtuales con SSL

ServerName "sabio"
ServerRoot "/etc/apache2"
PidFile /var/run/apache2.pid
ErrorLog /tmp/apache2.log
TypesConfig /etc/mime.types
DefaultType text/plain
User www-data
Group www-data

Listen 80
Listen 443

DirectoryIndex index.html index.html

#Cargamos el módulo que necesitamos y su configuración
Include /etc/apache2/mods-available/ssl.load
Include /etc/apache2/mods-available/ssl.conf

#Indicamos donde tenemos nuestro certificado
SSLCertificateFile /etc/apache2/ssl/apache.pem

<VirtualHost 192.168.0.3>
      SSLEngine on
      ServerName "aym.juntaex.es"
      DocumentRoot "/var/www/curso/aym"
      ErrorLog /tmp/aym_ERROR.log
      TransferLog /tmp/aym_ACCESS.log
      <Location /seguro>
            SSLRequireSSL
      </Location>
</VirtualHost>
```

Capítulo 15. Consejos para Mejorar la Seguridad

Apache no tiene problemas especiales de seguridad. Más bien todo lo contrario. Se trata de un servidor muy seguro y con pocos puntos débiles (que son rápidamente corregidos cuando se detectan).

Sin embargo, una configuración descuidada puede poner en riesgo el servidor y su contenido. Y, como no todo el mundo está especializado en seguridad web, no viene mal repasar algunos conceptos.

15.1. Mantente al Día

Cada día se detectan nuevos tipos de ataques, problemas de configuración, agujeros de software... “Los Malos” están siempre al acecho y, desgraciadamente, siempre irán por delante. Es muy difícil prevenir nuevos tipos de ataque que todavía no se han inventado.

Lo que sí debe hacer un buen administrador es mantenerse informado de los nuevos problemas de seguridad que se detecten. Así puede valorarlos y tomar las medidas que considere oportunas (a veces es mejor no hacer nada que meternos en una compleja actualización para prevenir un ataque poco probable o escasamente dañino).

Apache dispone de una lista de distribución para mantener informados a los administradores. Puedes darte de alta en http://httpd.apache.org/lists.html#http-announce.

15.2. Protege los Ficheros de Configuración

Los ficheros de configuración de Apache, así como sus subdirectorios, sólo necesita leerlos root. No hay porque dar permisos, ni de lectura, a ningún otro usuario.

15.3. Vigila los Logs

“Ojos que no ven, tortazo que te pegas”

¿De qué te sirve tener unos logs magníficos, perfectamente configurados si nunca los consultas?

Pon los sistemas de análisis que te avisen de las situaciones de los logs y búscate un rato para echarles un vistazo de vez en cuando.

Lo ideal sería que te hicieras tus propios programas de análisis y añadieras reglas para detectar situaciones de riesgo.

15.4. Evita los CGI

Simplemente son un gran problema de seguridad.

Si realmente los necesitas, es mejor que los escribas tú mismo o alguien de quien te fíes mucho. Y activa la opción suexec para asegurarte de que se ejecutan con los permisos que tú decidas (los del propietario del fichero). ¡¡NO LOS GUARDES COMO ROOT!!

15.5. Evita SSI

Con la directiva exec cmd de SSI se puede hacer el mismo daño que con un CGI. Además, los SSI pueden suponer una pérdida de rendimiento.

Pero si realmente los necesitas, puedes activar la opción suexec, y al menos evitarás que se ejecuten con los permisos de Apache (no se te ocurra guardarlos con el usuario root).

15.6. Evita los Contenidos Dinámicos en General

Otros lenguajes como PHP, Perl o TCL se ejecutarán con los permisos de los propietarios de los ficheros. Eso significa que pueden hacer muchas cosas, quizás demasiadas.

Si no puedes evitar los programas de generación de contenidos en tu servidor, al menos revisa bien con qué usuario se están ejecutando y qué permisos tienen esos usuarios.

15.7. Vigila los Enlaces Simbólicos

Lo mejor es no permitirlos. Así evitas que un usuario acceda a contenido al que no debería acceder.

Pero si quieres habilitarlos, al menos asegúrate de que el destino del enlace es legal añadiendo la opción SymLinksIfOwnerMatch

15.8. Haz Copias de Seguridad

Por último, ten en cuenta que lo más valioso de tu servidor probablemente sea la información que alberga.

Un ataque por denegación de servicio puede dejarnos fuera de la red unas horas, pero no afectará a nuestros contenidos.

Sin embargo, un intruso puede dañar o eliminar la información que albergamos. Asegúrate que puedes reponerla.

Capítulo 16. Optimización de Rendimiento

Al igual que ocurre con la seguridad, Apache no tiene problemas de rendimiento. Más bien al contrario.

Es cierto que no es un servidor diseñado para batir records en bancos de pruebas. Pero en entornos reales de producción su rendimiento es más que aceptable.

De todas formas, si eres un histérico del rendimiento o tienes una máquina realmente pequeña (para la carga de tu servidor) aquí tienes algunos consejos:

16.1. Más RAM

Lo más importante para el rendimiento de un servidor web es no swapear. Y eso depende de la cantidad de memoria RAM que tengas disponible.

Para evitarlo, controla el parámetro MaxClients que determina el número máximo de procesos Apache que se ejecutarán en la máquina. Mide cuanto ocupa cada uno de ellos y reparte la memoria RAM disponible.

Es cierto que hay otros factores que hacen el cálculo inexacto... Hay otros procesos en el sistema que consumirán RAM al ejecutarse, pero también hay memoria compartida entre todos los procesos de Apache.

El resto de parámetros son importantes. Por ejemplo, una buena CPU, una buena tarjeta de Red, discos rápidos... pero sobretodo, ¡Más RAM!

Cuanta más RAM tengas, más procesos de Apache (si swapear) podrás tener arrancados para atender a tus clientes.

16.2. Usa Linux 2.4 o Superior

Apache funciona en otros entornos.

Pero si quieres que corra, ponle sobre una buena carretera. Las llamadas al Sistema Operativo que necesita Apache están mucho más optimizadas si puedes ejecutarlo sobre Linux 2.6.

16.3. Evita la Resolución de DNSs Inversos

Cada vez que Apache tiene que resolver el nombre de máquina o dominio que corresponde a una dirección IP, debe hacer una llamada a un DNS. Y eso consume mucho tiempo.

Apache necesita resolver DNSs inversos cuando utilizas HostNameLookUps o directivas Allow, Deny basadas en nombres de máquina o de dominio. Intenta evitarlas. Si lo configuras por IP será mucho más rápido (la dirección IP viene en el paquete TCP/IP del cliente).

16.4. Evita los Enlaces Simbólicos

Si permites enlaces simbólicos, cada vez que Apache entra en un directorio tendrá que buscarlos. Si además utilizas SymLinksIfOwnerMatch, también tendrá que comprobar a quién pertenece el enlace y a quién pertenece el destino del enlace... Mucho trabajo.

16.5. Evita los Ficheros .htaccess

Está muy bien tener configuración personalizada por los usuarios. Pero tiene un coste.

Cada vez que Apache tiene que buscar un recurso, debe comprobar si en ese directorio (y en todos los anteriores) hay ficheros de configuración particulares que sean aplicables... Si la estructura de directorios es muy profunda, puede requerir un montón de llamadas al sistema. Y si encima hay ficheros .htaccess, también tendrá que abrirlos y procesarlos.

Por lo tanto, evita la directiva AllowOverride.

16.6. La Negociación de Contenidos

Sí, es cierto que la negociación de contenidos requiere tiempo de proceso. Pero es tan escaso y al mismo tiempo resuelve tantos problemas, que son muy raras las ocasiones en que necesites evitarla.

Si realmente necesitas negociación de contenidos, Apache resuelve la situación con el menor coste de rendimiento posible.

Es preferible utilizar Type Maps que Multiviews. pues este último requiere mirar en todo el directorio para ver qué archivos hay mientras que con Type Maps la configuración está en un único fichero.

16.7. Evita SSI

Si activas SSI (Server Side Includes) Apache tendrá que parsear todos los ficheros susceptibles de contener SSI, lo tengan o no.

16.8. Compila Tu Apache

Como última medida, puedes reducir la imagen en memoria de los procesos de Apache si compilas a tu medida el código.

Compílalo incluyendo únicamente los módulos que necesitas. Cuanto más pequeño sea el ejecutable, mas procesos podrás albergar en la RAM del servidor sin necesidad de swapear. Y cuantos más procesos, más peticiones podrás atender.

Pon especial cuidado en elegir la arquitectura MPM más conveniente para tu servidor. Si, como hemos recomendado, utilizas Linux 2.6, compila tu Apache con soporte para worker.

Así obtendrás los mejores resultados.

Capítulo 17. Bibliografía

La mejor fuente de documentación para Apache está en www.apache.org. Ahí encontrarás la referencia a todas las directivas de configuración, documentos sobre SSL, Seguridad, MPM, Rendimiento, Instalación, Compilación... ¡Hasta el código! :-)

También hay libros muy buenos. Generalmente escritos por alguno de los programadores de Apache que participa en la Comunidad. Puedes consultar, por ejemplo, “Apache Reference Guide” de Ben y Peter Laurie de la editorial O'Reilly.