Administración y Gestión de un Servidor Web Apache

por SLaYeR y Lechon

Ante cualquier duda o sugerencia no dudes en ponerte en contacto con nosotros.

SLaYeR slayer.ma@gmail.com
Lechon jose_5sp@hotmail.com

Índice de contenidos

1. Introducción

El manual que a continuación vais a leer es el primero que publico, en colaboración con mi compañero Lechon, y esperamos que no sea el unico.
Hemos decidido realizarlo sobre el servidor web Apache, pues creemos que hoy en dia a cualquier aficionado a la informatica se le ha pasado la idea de montar un servidor web para su propio uso y experimentacion.
Lejos de ser un manual definitivo, tenéis entre vuestras manos una guía de iniciación a la gestión de un servidor web.

Los puntos a tratar durante el desarrollo de este manual son los siguientes:

- Descripción de la estructura de la instalación del Apache 2.0
- Configuración y aspectos básicos del servidor Apache
- Descripción de servicios básicos y avanzados

Esperamos que disfrutéis leyendo este manual tanto como nosotros escribiéndolo, ni que decir que ante cualquier duda o sugerencia referente al texto la podeis plantear por correo o postearla directamente en los foros de Wadalbertia a cualquiera de los dos, haremos todo cuanto este en nuestras manos por resolver vuestras dudas.

1.1 ¿Por qué Apache?

Hemos decidido usar Apache por varios aspectos que, de forma general, consideramos que todo servidor web debe satisfacer. Algunos de estos aspectos son:

- Es uno de los servidores web mas utilizados a nivel mundial
- Es un sistema multiplataforma
- Posee infinidad de paquetes y módulos que nos permiten trabajar con gran cantidad de lenguajes de programación web, así como intérpretes de SQL y otras funciones.
- Permite transacciones seguras mediante SSL (Secure Socket Layer)
- Contiene soporte para Hosts virtuales

Apache es, hoy por hoy, uno de los servidores web mas utilizados a escala mundial, en gran parte se debe a que es Freeware bajo licencia GNU, y en parte también se debe a su robustez y sus múltiples posibilidades. Son ya muchas las empresas que gestionan su propio host (ya sea para Intranets o Internet) mediante un servidor Apache, lo cual les permite ahorrar las tasas de hosting privado o público que ofrecen numerosas empresas dedicadas a este tipo de servicios. Con Apache cualquiera puede montar su propio servidor Web y hacer uso de él.

El hecho de ser multiplataforma, es decir, el hecho de que soporte infinidad de sistemas operativos distintos entre si, es también un gran aliciente para su enorme proliferación. Hay versiones de Apache para los sistemas operativos mas usados (Windows, Linux, Unix, Solaris, Mac…)

Otra de las características más atractivas de este servidor, es que esta continuamente añadiendo nuevas características y mejoras, lo cual nos garantiza un crecimiento futuro. Además es relativamente fácil de configurar, puesto que solo existe 1 fichero de configuración.

1.2 ¿Por qué Linux?

Otro de los aspectos que nos gustaría recalcar es el por qué de usar una distribución Linux para este proyecto.

Entre otros muchos aspectos hemos elegido usar Linux sobretodo por ser un sistema operativo robusto, basado en plataformas Unix, lo cual nos proporciona un extra de seguridad a nuestro servidor.
Este tipo de sistemas están también, al igual que Apache, en continua expansión, añade mejoras y funciones nuevas casi a diario, y cada vez más personas y corporaciones optan por sistemas Linux atraídas, entre otras características, por su coste nulo y su continua proliferación.

Cabe destacar que grandes empresas como Google utilizan servidores basados en distribuciones Linux con Apache.

En comparación con otros sistemas operativos, Linux nos ofrece ventajas indudables. En los casos de grandes servidores, la más alta versión de Windows enfocada a Servidores, la

Windows DataCenter Server, limita la gestión de memoria física a 64Gb, y la potencia de cómputo a 32 procesadores. Estas limitaciones, que no obstante ofrecen un rendimiento muy alto, desaparecen en Servidores Unix y basados en Unix.

1.3 El equipo de prácticas

La máquina que utilizaremos para las prácticas es un Pentium II a 350 MHz, con 166Mb de RAM y un disco duro de unos 4Gb.

Prepararemos la máquina instalando en ella el sistema operativo, una distribución de SuSE Linux, concretamente la 9.2, e instalándole una interfaz de red Ethernet 10/100. Junto con la distribución instalaremos también el Apache 2.0 que nos proporciona el mismo paquete.

Con este equipo realizaremos las prácticas convenientes para aprender el uso y manejo del servidor Apache.

Esto demostrará que para montar nuestro propio servidor no necesitamos realizar una inversión cuantiosa sino que, mediante un equipo en desuso y un poco de creatividad podemos montar nuestro propio servidor casero, aunque eso si, nunca podremos obtener por estos métodos la seguridad que nos proporciona una máquina pensada para estos propósitos.

2. Instalación y descripción de Apache 2.0 y Linux

Para empezar con el proyecto procederemos a la instalación de Linux. Como anteriormente hemos citado instalaremos la versión 9.2 de la distribución SuSE, como no vamos a instalar nada mas que el sistema operativo y el servidor, obviaremos realizar particiones de disco, aunque seria mas recomendable guardar la información en una partición distinta a la de la que esta instalado el sistema operativo, pues de este modo podemos recuperar la información en caso de que caiga el sistema operativo y nos veamos obligados a formatear.
Como el proceso de instalación suele ser muy orientativo y es bastante similar a cualquier otra instalación de otro sistema operativo, comentaremos la misma muy por encima, sin entrar en detalles.
Para proceder a la instalación de SuSE 9.2, antes que nada, configuraremos la BIOS de la máquina para que arranque en primer lugar desde el lector de CD/DVD, a continuación insertaremos el disco que contiene la distribución en el lector y comenzaremos la instalación.
Al contrario que en anteriores versiones, la actual instalación de Linux es relativamente simple en comparación con las anteriores. Durante el proceso de instalación se requiere la presencia del usuario únicamente para seleccionar el idioma de la instalación, la distribución del teclado, la zona horaria y, lo que es mas importante, los paquetes y configuraciones que deseamos agregar a la instalación. Cuando hablamos de paquetes nos referimos a ciertos componentes de la instalación que por defecto no se instalan; entre ellos se incluyen herramientas de todo tipo, retoque fotográfico, herramientas CASE, compiladores de lenguajes de programación, y un largo etcétera.
De todos estos paquetes seleccionaremos obligatoriamente el de "Herramientas y Servicios de Red", o, en su defecto, buscaremos el paquete de "Apache2" y "Apache2-docs", que corresponden al servidor Apache y la documentación del mismo.

Es muy probable que en algún momento de la instalación se pida configuración de red, para lo cual indicaremos la de la red a la que estamos conectados o, en caso de disponer de un router, elegiremos configuración por DHCP.
También es necesario indicar la contraseña del *root*, es decir, el usuario que actuará de administrador de la máquina, y es probable que se pida la creación de otro usuario.
La creación de otro usuario distinto al del root es importante, ya que solo debemos usar la cuenta de root cuando sea *imprescindible*, ya que en el caso de que suframos una ataque a nuestro servidor, el atacante correría en el sistema con privilegios de *root* lo cual puede desencadenar en un desastre.
La mejor opción es crear un usuario con ciertos permisos sobre el sistema y, cuando necesitemos privilegios de administrador para realizar alguna modificación o cambio de configuración, podemos abrir una shell de sistema en modo *root*.

A medida que transcurra el proyecto y detallemos la estructura del Apache iremos añadiendo algunos comentarios sobre prácticas aconsejables para mantener seguro nuestro servidor.

2.1 Arquitectura del Servidor Apache

El servidor Apache es un software que está estructurado en módulos. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Los módulos del Apache se pueden clasificar en tres categorías:

- **Módulos Base:** Módulo con las funciones básicas del Apache
- **Módulos Multiproceso:** son los responsables de la unión con los puertos de la máquina, acepando las peticiones y enviando a los hijos a atender a las peticiones
- **Módulos Adicionales:** Cualquier otro módulo que le añada una funcionalidad al servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código.
El resto de funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software.

Todas las funciones básicas de los módulos y sus configuraciones se pueden encontrar en el fichero de configuración *httpd.conf*. Esta característica nos permite obviar explicar todos y cada uno de los módulos de Apache, ya que sus directivas se encuentran en el fichero de configuración anteriormente mencionado, y que, de forma mas explicita, comentaremos mas adelante.

2.2 Estructura de la instalación de Apache

La estructura de la instalación siempre es un punto un tanto conflictivo, puesto que hay muchas distribuciones y versiones de Linux, puede cambiar la ubicación de las carpetas o los ficheros que mencionamos a continuación, no obstante no suele ser difícil encontrarlos si seguimos la documentación de Apache. A continuación detallamos la estructura que ha generado la instalación de Apache en nuestro servidor:

- **/bin/apache2:** Este directorio contiene los ficheros ejecutables de Apache 2.
- **/etc/init.d/apache2:** Aquí están los demonios que permiten iniciar, parar o reiniciar los servicios de Apache, algo que durante su configuración haremos mucho.
- **/var/logs/apache2:** En este directorio se almacenan lo logs de error, los registros de acceso etcétera. Este directorio conforma nuestra principal fuente de auditoria respecto al servidor, por lo que hay que tenerla muy en cuenta.

- **/srv/www/htdocs:** En este directorio situaremos los archivos y páginas que vamos a ofrecer. Siempre y cuando no se especifique lo contrario este será el directorio por defecto en el que guardaremos los archivos.

- **/bin/apache2/manual:** Contiene los ficheros de ayuda y los manuales de Apache.

- **/etc/httpd/httpd.conf:** Este es el fichero de configuración manual de Apache.

Es más que aconsejable que sobre estos directorios solo tenga permisos de escritura y ejecución el usuario *root*, ya que cualquier modificación de alguno de estos directorios puede desde obligarnos a dejar de servir páginas hasta dejar completamente inutilizable el servidor.

3. Configuración de Apache

En este capitulo vamos a desmembrar de forma detallada los aspectos más importantes del fichero de configuración *httpd.conf* el cual, sin duda, es el indiscutible protagonista de este apartado.

3.1 El fichero httpd.conf

El fichero *httpd.conf* es, como antes habíamos mencionado, la principal fuente de configuración del servidor Apache, en el se encuentran todas las directivas aplicables a la configuración, y conocer profundamente este fichero y su estructura es vital para el administrador del servidor.

El fichero de configuración se divide en cuatro secciones, a saber:

- Directivas de funcionamiento
- Parámetros de gestión de recursos
- Hosts Virtuales
- Parámetros de seguridad

Es necesario que sobre este fichero no tenga acceso ningún usuario que no sea *root*, ya que las modificaciones de este fichero pueden desencadenar en terribles agujeros de seguridad que nos dejarían indefensos frente a un ataque.

3.1.1 Directivas de funcionamiento

Las directivas de funcionamiento del fichero de configuración de Apache son, como su propio nombre indica, unas directrices que definen el funcionamiento interno y externo de Apache en aspectos tales como el nombre del servidor, la dirección de correo del webmaster, el número de subprocesos que se abrirán en cada sesión de Apache etc.
A continuación vamos a ver una relación de las directivas más importantes de este apartado:

ServerName: Especifica el nombre y el puerto que el servidor utiliza para identificarse, normalmente se determina automáticamente, pero es recomendable especificarlo explícitamente para que no haya problemas al iniciar el servidor. Si el servidor no tiene un nombre registrado en las DNS, se recomienda poner su número IP. No puede estar dentro de ninguna sección.
La sintaxis de uso es:
*ServerName direccionIP:Puerto p**.e. ServerName localhost:80***

ServerAdmin: especifica la dirección de correo electrónico del administrador, esta dirección aparece en los mensajes de error, para permitir al usuario notificar un error al administrador. No puede estar dentro de ninguna sección. Se encuentra disponible a través

del módulo Core. Es aconsejable que la dirección de correo no pertenezca al dominio administrado, pues, en caso de que falle, no podremos acceder al correo que probablemente pueda contener mensajes advirtiéndonos del fallo.

ServerRoot: especifica la ubicación del directorio raíz donde se encuentra instalado el Apache, a partir del cual se crea el árbol de directorios comentado anteriormente. Esta directiva no debería cambiar a no ser que se mueva la carpeta de instalación de apache a otro directorio. Se encuentra disponible a través del módulo Core.

KeepAlive: especifica si se utilizarán conexiones persistentes, es decir, que todas las peticiones de un usuario se atenderán con la misma conexión.

MaxKeepAliveRequests: número máximo de conexiones persistentes. (Número máximo de usuarios concurrentes si KeepAlive esta en ON). Para establecer este parámetro, hay que tener en cuenta el ancho de banda de salida de nuestro servidor, por el cual deberá ser enviada toda la información, si se establece un valor muy grande respecto al ancho de banda, el tiempo de respuesta se verá incrementado para cada usuario. Se encuentra disponible a través del módulo Core.

KeepAliveTimeout: tiempo que espera en segundos entre peticiones de un usuario, antes de considerar que este ha terminado, y cerrar su conexión. Esta directiva solo es valida si la directiva KeepAlive esta activada.

ServerType: Indica la forma de arrancar Apache. Puede tener como valores *inetd* o *standalone.*
Su sintaxis es la que a continuación se especifica:
ServerType [inetd | standalone]

El arranque mediante *inetd* indica que no existe un servidor httpd permanente arrancado, sino que el arranque se realiza a partir del servidor de red inetd o xinetd. Cuando inetd recibe una conexión de un servicio, genera un nuevo proceso hijo que ejecuta el programa que proporciona el servicio. En este caso el servidor ejecutado es Apache. El servidor inetd vuelve a su estado original (esperar conexiones) y el proceso recientemente generado sirve la hoja HTML al cliente que la solicitó.
La forma habitual de trabajar se realiza en standalone. En este caso es el propio servidor Apache el que esta "escuchando" las conexiones de la red. Cuando se recibe una petición HTTP, se genera un nuevo proceso (si es necesario) y se sirve la petición. En la mayor parte de las ocasiones se ahorra para cada petición la creación de un proceso hijo.
Las directivas Port, MinSpareServers, MaxSpareServers, StartServers, MaxClients y MaxRequestPerChild son solo validas para la configuración en standalone

Port y **Listen:** Se utilizan solo cuando se ejecuta el servidor en modo *standalone.* La directiva **Port** especifica el puerto por el que Apache acepta conexiones. Si el servidor debe atender a varios puertos, se deben usar tantas directivas **Listen** adicionales como sean necesarias.
La configuración de ejemplo que a continuación transcribimos le indica a Apache que atienda las peticiones web por los puertos 80 y 8080
Port 80
Listen 8080

MaxClients: Se utiliza para especificar el máximo número de clientes (navegadores) que se pueden conectar simultáneamente al servidor. Si se alcanza este límite, los clientes quedan bloqueados hasta que algunos de los servidores queda libre. Este valor no conviene que sea demasiado bajo (su valor por defecto es 150) y se utiliza como mecanismo para limitar el numero de procesos que se crean ante un numero de peticiones demasiado elevado. Este límite, que se impone el propio servidor Apache evita caídas indeseadas del servidor. Su sintaxis se especifica a continuación:
MaxClients <Numero Entero>

TimeOut: el valor se utiliza para configurar medido en segundos, tres parámetros:

1. El tiempo tal que puede tardar una petición en ser recibida entera
2. La cantidad de tiempo que espera entre recepción de paquetes TCP
3. La cantidad de tiempo entre ACK's en transmisiones TCP

Pasado este tiempo se produce un mensaje de error en el que se indica que se ha consumido el tiempo máximo de espera. Establecer un valor muy pequeño puede dar lugar a que los usuarios reciban este mensaje de error, y establecer un valor muy pequeño dará lugar a una sobrecarga de la máquina.

CacheNegotiateDocs: Se utiliza para trabajar con proxies. Si esta directiva no aparece, el servidor Apache envía una respuesta que indica a los servidores Proxy que no almacenen los documentos en su caché. Esto impide que se obtengan versiones antiguas de páginas HTML, pero tiene como desventaja que se desaprovecha el ancho de banda de la red. Si la directiva aparece, se permite que los proxies almacenen documentos proporcionados por este servidor en su caché. Su sintaxis es la que a continuación se especifica.
CacheNegotiateDocs

3.1.2 Parámetros de Gestión de Recursos

Siguiendo con la estructura propuesta por el fichero de configuración, dejamos atrás las Directivas de Funcionamiento y nos adentramos en el siguiente apartado. Este apartado, el de Parámetros de Gestión de Recursos, no es más que una colección de directivas (como anteriormente) dedicadas a indicar a Apache donde encontrar la información necesaria en cada momento o para definir como debe actuar de cara a sucesos como no existir la página solicitada o gestionar Hosts Virtuales. A continuación veremos los parámetros más interesantes de este apartado:

DocumentRoot: Esta directiva le indica a Apache la carpeta raíz que se ubica en el servidor, desde la que se servirán los documentos. Por defecto, todas las peticiones tendrán como raíz esta carpeta, a no ser que se utilicen alias (Directorios virtuales en IIS).
La carpeta raíz por defecto es la carpeta htdocs, la cual podemos encontrar en la ruta /srv/www/htdocs. Si se cambia este directorio por otro, practica muy recomendable, es muy importante que se ponga el nuevo valor, no solo en esta línea, sino también en la sección <Directory> en la que se establecen los parámetros de configuración de este directorio.

DirectoryIndex: Mediante este parámetro, indicamos a Apache qué fichero debe buscar, por defecto, en caso de que no se especifique ninguno. Este fichero, de forma

predeterminada es *index.html*, es decir, si desde el navegador tratamos de acceder a *www.upv.es* el servidor por defecto servirá *www.upv.es/index.html*.

Una de las partes atractivas de esta directiva es que podemos indicarle más de un fichero a buscar de la siguiente forma:

DirectoryIndex fichero1 fichero2 fichero3

El orden con el que se especifica el nombre del fichero determinara la prioridad a la hora de decidir que fichero es el que se muestra.

RedirectPermanent: Se utiliza para indicar que una página alojada en el servidor web ha pasado ha estar en otro URL. De esta forma, la respuesta que genera el servidor web permite al cliente iniciar una nueva petición a un nuevo servidor que aloja la página pedida.

DefaultType: Especifica el tipo mime que se servirá por defecto en caso de no conocer la extensión del fichero que se esta sirviendo. Por defecto, se indicará que se sirve texto plano, con el valor *text/plain*.

ErrorLog: Especifica la ubicación del fichero que contiene el registro de errores, por defecto en la carpeta logs (*/var/logs/apache2)*

LogLevel: Indica el tipo de mensajes que se guardaran en el fichero de registro de errores, dependiendo de los valores especificados, se guardaran unos u otros.
Los valores disponibles son: *debug, info, notice, warn, error, crit, alert, emerg.*

LogFormat: La directiva permite definir el formato que se utilizará para almacenar los registros. A cada formato se le puede asignar un nombre, utilizándolo luego para crear distintos tipos de ficheros de registro. Para saber más consultar el Apéndice A
Sintaxis: *LogFormat "configuracionError" nombre*

IndexOptions: Esta directiva controla la apariencia de la página que se mostrara a un usuario cuando se pide la lista de ficheros de un directorio.
Entre las opciones posibles, destaca:
FancyIndexing, que muestra los nombres de los ficheros con iconos etc…

FolderFirst: Hace que primero se muestren los directorios, esta opción solo se puede establecer en el caso de que FancyIndexing este activa.

3.1.3 Hosts Virtuales

Al entrar en este apartado, llegamos a una de las partes mas interesantes de Apache, el uso de Hosts Virtuales o *Alias*, aunque nos vemos obligados a recalcar que este apartado pertenece en parte a las Directivas de Funcionamiento que anteriormente describíamos.

Mediante la correcta configuración de esta directiva podemos alojar mas de un sitio web en el mismo servidor, totalmente independientes uno del otro. A continuación describiremos con un caso teórico el uso de esta directiva, después, claro está, de una precisa explicación del funcionamiento de la misma.

Directivas *VirtualHost* y *NameVirtualHost*

Esta es una directiva de bloque que se utiliza para especificar que un mismo servidor Apache sirva peticiones a varias direcciones web. De esta forma se podrán atender peticiones web para dos cabeceras distintas en un mismo servidor.
Hay dos alternativas básicas para aplicar esta técnica:

- ✓ **Basada en Nombres DNS**
  - o Se trata de que un computador con una única interfaz de red que solamente tiene asignada una dirección IP pueda servir páginas a dos nombre se servidores Web diferentes. Para que sea posible la utilización de esta técnica es necesario definir al menos un alias DNS

- ✓ **Basada en dirección IP**
  - o Se trata de que un computador con varias interfaces de red albergue dos dominios diferentes dependiendo de la IP que haya atendido la petición. Esta opción es útil cuando un computador esta conectado a dos redes diferentes (Intranet e Internet) y desea que el usuario tenga una visión diferente dependiendo de la interfaz de red por la que acceda al computador.

La opción mas comúnmente utilizada es la primera y para ello e necesario emplear las directivas *NameVirtualHost* y *VirtualHost* en el archivo de configuración de Apache.

Supongamos que deseamos dar de alta dos dominios, www.servicio1.es y www.servicio2.es, que se desea atender vía web. En este caso se deben definir dos directivas *VirtualHost* que contengan directivas ya explicadas, que particularizaran el servicio proporcionado por cada nombre.

Por otra parte, es necesario incluir la directiva *NameVirtualHost* que indica que dirección IP va a proporcionar el servicio a los hosts virtuales. Si el computador solamente tiene una dirección de red, basta con especificar la dirección IP del servidor que ejecuta Apache, o el carácter *. En este último caso se indicara que todas las direcciones asignadas al servidor que ejecuta Apache se utilizaran como direcciones asignadas a los *Hosts Virtuales*.

En el caso teórico que describimos a continuación, la máquina con una única tarjeta de red cuya dirección es 138.100.9.22 sirve a dos dominios. Dentro de cada sección *VirtualHost* se

especifican directivas que particularizan cada host virtual. Merece especial atención la directiva *DocumentRoot*, puesto que el sistema de archivos que se exporta vía web es diferente para cada host. De esta forma, aunque ambas máquinas se resuelvan a la misma dirección IP, el acceso a las dos URL proporciona páginas diferentes.

Configuración del ejemplo

```
NameVirtualHost *

<VirtualHost www.servicio1.es>
        ServerAdmin admin1@servicio1.es
        DocumentRoot "/srv/www/htdocs/uno"
        ServerName www.servicio1.es
</VirtualHost>

<VirtualHost www.servicio2.es>
        ServerAdmin admin2@servicior2.es
        DocumentRoot "/srv/www/htdocs/dos"
        ServerName www.servicio2.es
</VirtualHost>
```

La técnica de hosts virtuales basada en direcciones IP requiere que el computador que alberga los dos hosts tenga dos direcciones IP diferentes (dos interfaces de red). En este caso cada sección *VirtualHost* hace referencia a cada una de las direcciones.

Configuración del ejemplo

```
<VirtualHost 138.100.8.22>
        ServerAdmin admin1@servicio1.es
        DocumentRoot "/srv/www/htdocs/uno"
        ServerName www.servicio1.es
</VirtualHost>

<VirtualHost 138.100.9.22>
        ServerAdmin admin2@servicior2.es
        DocumentRoot "/srv/www/htdocs/dos"
        ServerName www.servicio2.es
</VirtualHost>
```

3.1.4 Directivas de Seguridad

A continuación, serán expuestas las directivas de configuración del servidor que especifican quien tiene permisos para acceder a determinados recursos. De esta forma se pueden restringir accesos no autorizados a determinados documentos. En esta sección se exponen las formas mas simples que ofrece Apache para proteger documentos:

- ✓ Autorización de acceso **basada en el usuario.**
- ✓ Autorización de acceso **basada en el nombre o dirección del cliente que realiza la petición.**

Estos dos tipos de autorizaciones se pueden especificar de dos formas:

- ✓ En un **archivo de configuración global del servidor:** access.conf o http.conf
- ✓ En un **archivo de configuración que puede haber en cada directorio del sistema de archivos exportado** (véase la directiva AccessFileName).

En esta sección se muestran las directivas más importantes que aparecen en los archivos de configuración de Apache, y que permiten establecer los mecanismos básicos de autorización por computador y por usuario. Estas directivas aparecerán en el archivo global *access.conf* o *httpd.conf* y en el que especifica la directiva AccessFileName.

Directivas

AccessFileName: Se utiliza para especificar el nombre de cada archivo que contiene directivas que especifican los parámetros de seguridad para cada directorio. Su sintaxis se especifica a continuación:
AccessFileName [archivo]

En cada directorio exportado puede haber un archivo con este nombre que contendrá directivas que especifican parámetros de seguridad para los archivos del directorio en el que esta almacenado el archivo y sus subdirectorios.

AllowOverride: Cuando el servidor encuentra un fichero .htaccess (como se explica en la directiva *AccessFileName*) es necesario saber que directivas presentes en ese fichero pueden prevalecer sobre las directivas de configuración previas. En ese momento es cuando entra en juego la directiva *AllowOverride*, porque especifica que tipos de directivas se pueden modificar por el archivo *.htaccess* de un directorio.

La directiva *AllowOverride* puede definirse como:
AllowOverride opcion opcion opcion….Cada una de estas opciones que se aplican como parámetro a esta directiva indica que directivas se pueden sobrescribir. Seguidamente se explican las más importantes:

- **AuthConfig:**
  - Permite el uso de directivas que permiten el control de acceso a usuarios a una zona Web: *AuthGroupFile, AuthName, AuthType, AuthUserFile, requiere.* Hay mas directivas como: *AuthDBMGroupFile, AuthDBMUserFile, etc.*

- **Indexes:**
  - Permite la aparición de directivas que permiten mostrar el contenido de un directorio que no contiene el archivo *index.htm: AddDescription, AddIcon, AddIconByEncoding, AddIconByType, DefaultIcon*…

- **Limit:**
  - Permite la aparición de la directiva *Limit* que permite acotar el acceso a una zona Web por nombre o dirección Ip del cliente. Esta directiva se explicara seguidamente.

- **Options:**
  - Permite la aparición de la directiva *Options* que permite modificar los valores de seguridad por defecto que haya establecido el administrador del servidor Web.

- **All:**
  - Si aparece se permiten todas las opciones anteriores.

- **None:**
  - Si aparece no se permite ninguna de las opciones anteriores.

Sintaxis: AllowOverride All|None|*directive-type* [*directive-type*] ...
***Valor por defecto*:** AllowOverride All

En este ejemplo que se muestra a continuación no se permite que los usuarios almacenen en su zona Web programas CGI que se puedan ejecutar ni que se pueda mostrar el contenido de un directorio que no contiene el archivo *index.html*.

```
<Directory /home/*/public_html/>
      Options –ExecCgi –Indexes
      AllowOverride Indexes Options
</Directory>
```

Por otra parte, se permite que sea el usuario el que, si lo desea, pueda cambiar estas restricciones modificando el archivo *.htaccess* para cambiar las autorizaciones por defecto, de tal forma que si se incluye la siguiente línea en el archivo *.htaccess* de un directorio Web de usuario, podrá mostrar el contenido de un directorio y ejecutar programas CGI.

```
Options +ExecCgi +Indexes
```

Limit: Se utiliza para limitar el acceso a la información de un directorio bajo un conjunto de métodos. Esta limitación se realiza incluyendo un conjunto de directivas dentro del bloque. Su sintaxis se especifica a continuación:
<Limit métodos>
Directivas
</Limit>

Dependiendo de las operaciones que se realicen sobre el subárbol de directorios que se desea limitar se debe especificar un método u otro. Supóngase que en un directorio no se necesita que ningún cliente "envíe" ningún tipo de información al servidor. En este caso habría que habilitar los métodos que permiten obtener información del servidor (GET y HEAD) y restringir los métodos POST, PUT y DELETE.
<Limit GET HEAD>
Directivas
</Limit>

<Limit DELETE POST PUT>
Directivas
</Limit>

Los métodos incluidos en la lista pueden ser uno o más de los siguientes: GET, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, y UNLOCK.
Los nombres de los métodos distinguen mayúsculas de minúsculas**.** Si usa GET también se restringirán las peticiones HEAD. El método TRACE no puede limitarse.

Consejo:
Es mejor usar una sección *LimitExcept* en lugar de una sección *Limit* cuando se quiere restringir el acceso, porque una sección *LimitExcept* protege contra métodos arbitrarios.

LimitExcept: Se usa para englobar un grupo de directivas de control de acceso que se aplicarán a cualquier método de acceso HTTP no especificado en los argumentos; es lo contrario a lo que hace una sección *Limit* y puede usarse para controlar tanto métodos estándar como no estándar o métodos no reconocidos. Sintaxis:

<LimitExcept *method* [*method*] ... >
</LimitExcept>

Autorización de acceso por dirección del cliente:
Directivas allow, deny y order

La autorización de acceso a un subárbol de directorios basado en la dirección del cliente autoriza o deniega el acceso dependiendo de la dirección IP del host que haya generado la petición sin necesidad de pedir información adicional al cliente. La especificación del PC se puede realizar por nombre de DNS o por la dirección IP. La especificación de acceso se realiza con las directivas:

- ✓ Order
- ✓ Allow from
- ✓ Deny from

Estas tres directivas deben estar incluidas en un bloque Limit (anteriormente explicado).
La directiva *order* define el orden en el que se deben procesar las directivas *allow* y *deny*. Las posibles opciones son:

- ✓ **Order allow, deny:** primero se evalúan las directivas *allow* y posteriormente las directivas *deny*.
- ✓ **Order deny, allow:** primero se evalúan las directivas *deny* y posteriormente las directivas *allow*.
- ✓ **Order mutual-failure:** las máquinas que aparecen asociadas a la directiva *allow* y no aparecen en la directiva *deny* son autorizadas

La directiva **allow from** especifica una lista de máquinas a las que se les autoriza el acceso a un directorio o árbol de directorios. La directiva **deny from** especifica una lista de computadores a los que se les deniega el acceso a un directorio o árbol de directorios. Los parámetros que contiene pueden ser los siguientes:

- ✓ **all:** se autoriza/deniega el acceso a todas las máquinas independientemente de su IP
- ✓ **dirección IP:** se autoriza el acceso a la máquina cuya IP sea la especificada.
- ✓ **dominio:** se autoriza el acceso a las máquinas que pertenezcan al dominio indicado.

Ejemplo:

<Limit GET>
order deny, allow
deny from all
allow from upv.es
allow from upc.es
</Limit>

Autorización de acceso por usuario

La autentificación es cualquier proceso mediante el cual se verifica que alguien es quien dice ser, es decir; un proceso por el cual a alguien se le permite acceder donde desea u obtener la información busca.

La autorización de acceso basada en autentificación de usuarios permite controlar el conjunto de usuarios que tienen acceso a un subárbol de directorios. Esta autentificación se basa en utilizar nombres de usuarios y contraseñas. Cuando el cliente accede a dicho subárbol, el servidor le pedirá que se identifique mediante un nombre de usuario y una contraseña. Si la identificación tiene éxito, se autoriza al usuario a entrar en la zona restringida sin volver a pedirle la información de identificación. Las directivas que permiten la autentificación de usuario son:

Directiva require: Especifica el usuario o conjunto de usuarios que están autorizados a entrar en la zona restringida.
Sintaxis:

Require *entity-name* [*entity-name*] ...

Require debe ser usada de forma conjunta con las directivas *AuthName*, *AuthType*, y directivas como *AuthUserFile* y *AuthGroupFile* (para definir usuarios y grupos) para funcionar correctamente.

Esta directiva está incluida en el bloque *Limit.* Si el último parámetro es *valid-user,* indica que se autoriza a un cliente a entrar en el espacio web siempre que proporcione un usuario válido, es decir, debe estar dado de alta en el archivo de usuarios y debe proporcionar su contraseña correcta.

Require user *userid* [*userid*] ...
: Solo los usuarios mencionados pueden acceder al recurso.

Require group *group-name* [*group-name*] ...
: Solo los usuarios pertenecientes a los grupos mencionados pueden acceder al recurso.

Require valid-user
: Todos los usarios pueden acceder al recurso.

Directiva AuthType: Especifica el tipo de autentificación que se requiere. Actualmente solamente están implementadas las opciones *Basic* y *Digest*. La primera utiliza el mismo algoritmo que utilizan los sistemas Unix para codificar las palabras clave; La otra se basa en MD5 y no todos los navegadores lo soportan.
Para que funcione correctamente, esta directiva tiene que ir acompañada por las directivas *AuthName* y *require*, y de directivas como *AuthUserFile* y *AuthGroupFile*.
Sintaxis:

AuthType Basic|Digest

Directiva AuthName: Esta directiva especifica el nombre de dominio que se muestra al solicitar autorización para acceder a un directorio. Este nombre de dominio se muestra al cliente para que el usuario sepa qué nombre de usuario y contraseña ha de introducir. AuthName toma solamente un argumento; si el nombre de dominio contiene algún espacio, debe escribirse entre comillas. Para que funcione correctamente, esta directiva debe usarse junto con las directivas *AuthType* y *require*,, y con directivas como *AuthUserFile* y *AuthGroupFile*.

Directiva AuthUserFile: Especifica el archivo que contiene los usuarios con las palabras clave. Cada línea del archivo de usuarios contiene un *username* seguido de un punto y coma, seguido de la contraseña cifrada. Si el mismo usuario ID es definido múltiples veces, mod_auth usarán la primera aparición para verificar la contraseña.
Sintaxis:

AuthUserFile *file-path*

Este archivo se compone de dos campos: el primero es el nombre de usuario y el segundo es la palabra clave que tiene asociada (cifrada). Para añadir un nuevo usuario a este archivo, Apache incluye un programa que permite dar de alta a un usuario o cambiar su clave. Este programa es htpasswd, y su sintaxis es la que a continuación se especifica:

htpasswd -c Filename username

La opción –c indica que se debe crear un nuevo archivo de usuarios y solo se debe incluir la primera vez que se crea un usuario. A este programa se le pasan dos parámetros: el primero es el archivo que contiene los usuarios y el segundo es el nombre de usuario que se desea añadir al archivo o al que se desea cambiar su contraseña.

Directiva AuthGroupFile: Especifica el archivo que contiene los grupos con sus palabras clave. El fichero *AuthGroupFile* se edita manualmente, usando tu editor de texto preferido, y su formato es el siguiente:
Sintaxis:

nombre_del_grupo: usuario1 usuario2 usuario3

Aspectos sobre seguridad
El administrador debe asegurarse de que el archivo *AuthUserFile* es almacenado fuera del árbol de documento del servidor web. No debe ponerse en el directorio que protege. En ese caso, los clientes podrían ser capaces de trasvasar el archivo.

Apéndice A: Elaborar Formatos de Log's

Elaborar nuestros propios formatos de log es vital cuando necesitamos saber exactamente una información concreta. Para ello, Apache nos brinda una directiva/herramienta que nos permite definir el formato de los archivos log y después asignar este formato a un fichero log.

Para crear un formato para los logs utilizamos la directiva/herramienta *LogFormat*, la cual, mediante unos parámetros define un formato de log reconocible por un nombre que hemos decidido nosotros. Por ejemplo:

```
LogFormat "%h %l %u %t \"%r \" %s %b" formato1
```

El formato que se ha definido incluye:

- ✓ %h (nombre o IP del computador que generó la petición)
- ✓ %l (identificación que proporciona el cliente)
- ✓ %u (nombre del usuario con el que se ha accedido a la página)
- ✓ %t (fecha en la que se ha atendido a la petición HTTP)
- ✓ %r (petición servida)
- ✓ %s (código de tres dígitos HTTP que indica el resultado de la petición)
- ✓ %b (numero de bytes transferidos)

Mediante el uso de estas variables podemos crear formatos a nuestra medida para controlar mas cómodamente nuestros logs, solamente debemos recordar el nombre que le asignamos a cada estilo de formato para después aplicarlo a nuestros ficheros de log mediante la directiva *CustomLog*.

La directiva CustomLog nos permite especificar el nombre de un formato especificado por la directiva LogFormat, de tal forma que se puede ajustar el tipo de información que se desea que aparezca en el archivo de log.
Ejemplo:

CustomLog logs/access_log formato1

En este caso se define el formato que se desea que contenga el archivo de transferencias.

Apéndice B: Practicas Seguras

La seguridad del servidor es uno de los aspectos mas importantes a tener en cuenta para su correcta administración, ya que al proporcionar varios servicios mantiene abiertos varios puertos que, en el peor de los casos, pueden ser victimas de un ataque. Para poderle dar mas seguridad deberíamos instalar un escáner de puertos para escanear nuestros propios puertos y, lo mas importante, debemos activar el firewall *IPTables*, una potente herramienta que viene incorporada en el kernel. Este cortafuegos lo podemos configurar para aceptar, denegar o ignorar los paquetes que pasen por nuestro servidor. IPTables tiene diferentes nombres según la versión del kernel que utilicemos:

- ✓ Kernel 2.0 → IPFwadm
- ✓ Kernel 2.2 → IPChains
- ✓ Kernel 2.4 → IPTables

Configuración y uso de IPTables

El programa nos ofrece la posibilidad de configurar tres tablas: INPUT, FORWARD y OUTPUT, para catalogar los paquetes:

- ✓ **INPUT:** Los paquetes que llegan al servidor desde el exterior o desde la mima maquina pasan por la tabla de INPUT. Es de donde vienen todos los paquetes.
- ✓ **FOREWARD:** Los paquetes que llegan al servidor para ser enrutados hacia otro sitio pasan por la regla FORWARD.
- ✓ **OUTPUT:** Los paquetes generados en el mismo servidor y listos para ser enviados hacia el exterior pasan por la tabla OUTPUT.

Cada tabla consta de una serie lógica de reglas (ordenadas) que deciden el destino del paquete. El paquete acabara básicamente de una de estas formas:

- ✓ **ACCEPT:** Se deja pasar el paquete; el firewall no actúa.
- ✓ **REJECT:** Se deniega el paquete. El firewall rechaza la conexión, el emisor del paquete recibe un rechazo.
- ✓ **DROP:** El paquete es ignorado. El emisor desconoce que ha pasado con el paquete, es como si el servidor estuviera fuera de servicio y el paquete se perdiera.

Para entender mejor estas reglas repasaremos algunos conceptos básicos de TCP/IP y el envío de paquetes, aunque para mas información recomiendo leer el "Taller de TCP/IP" de Vic_Thor, disponible en este enlace por cortesía de Yorkshire.

http://www.wadalbertia.org/docs/taller_tcpip.pdf

Envío de paquetes con TCP/IP

Para establecer una conexión a una maquina y un puerto determinados, hay que hacer el conocido como *"three way handshake"* (saludo de las tres vías). Consiste en lo siguiente:

1. Enviamos un paquete con el BIT SYN (*Synchronize*) activado.
2. Se nos contesta con un paquete con el BIT ACK (*Acknowledgement*) activado y hará lo mismo que nosotros: enviar un paquete SYN y esperar un ACK nuestro; entonces ya puede empezar a enviar y recibir datos.

El proceso suele simplificarse enviado en un solo paquete los bits SYN y ACK activados, por lo que finalmente quedaría así:

EMISOR	Flujo de datos	RECEPTOR	DESCRIPCIÓN
SYN	→		¿Me recibes?, Quiero conectar.
	←	SYN/ACK	Si, te recibo. ¿Empezamos?
ACK	→		Empecemos.

Esto, evidentemente, solo sucede si el puerto de conexión esta abierto. Si dicho puerto está cerrado, al enviar el SYN responde con un RST/ACK y finaliza la conexión. IPTables por defecto contesta de otro modo; mediante un mensaje "ICMP Port Unreachable", pero lo podemos especificar con el parámetro *--reject-with*.

En resumen, la regla ACCEPT enviara un SYN/ACK y establecerá la conexión pedida, la regla REJECT enviara un ICMP Port Unreachable o un RST/ACK, por tanto rechazaremos la conexión, y finalmente, con DROP no enviaremos nada, simplemente desecharemos el paquete, solo queda que el cliente acabe la conexión después de enviar numerosos SYN. Esto último es muy interesante, pues nos permite hacer ver que el servidor esta apagado o fuera de servicio.

Nuestro cometido es, si queremos crear un firewall, definir estas reglas para las tres tablas de modo que solo recibamos paquetes de los clientes que conocemos y tenemos interés en aceptar.

Bibliografía

- ✓ http://es.tldp.org (Proyecto LuCAs)
    - o Página dedicada a proyectos y FAQ's sobre sistemas Linux y programas de libre distribución con licencia GNU

- ✓ www.desarrolloweb.com
    - o Página dedicada a la recopilación de noticias y FAQ's sobre el desarrollo de webs o aplicaciones web.

- ✓ http://iptables-tutorial.frozentux.net/spanish/chunkyhtml/
    - o Traducción al español de un completo manual sobre el manejo y uso de IPTables en Linux

- ✓ Programación de Aplicaciones Web (S. Rodríguez. Editorial Thomson)
    - o Libro-Manual basado en el uso de distintos lenguajes de programación web y el manejo de comercio web y administración de servidores

- ✓ Documentación interna de Apache