Fortificando un Servidor Apache (I de IV)

Estos documentos han sido escritos y publicados por:

Chema Alonso, MVP de Windows Security y escribe diariamente en su blog de "Un Informático en el lado del mal".

Chema trabaja en Informática 64 y escriben en los blogs Un Informático en el lado del mal y vista-tecnica

Recopilación: Cristian Borghello, Director de www.segu-info.com.ar

V1.0 – 071104

Indice

Fortificando un Servidor Apache (I de IV)

****
Artículo publicado en:
http://elladodelmal.blogspot.com/2007/09/fortificando-un-servidor-apache-i-de-iv.html
****

Muchas son las veces que hemos hecho hincapié en definir la seguridad informática como tres factores complementados, los Procesos, las Personas y la Tecnología. De nada sirve tener la mejor tecnología sin unos buenos procesos de implantación, gestión y actualización que la mantengan segura o sin unos usuarios responsables. En este artículo vamos a ver algunas recomendaciones para fortificar un servidor Web con Apache. Las recomendaciones de seguridad suelen ser comunes en todos los productos y plataformas y lo que suelen cambiar son las herramientas para conseguir los mismos fines.

Las leyes de la fortificación

La fortificación de sistemas tiene tres principios básicos que dirigen todo el proceso y que debes tener siempre presente en cualquier fortificación que vayas a realizar, así que, para fortificar un servidor web con Apache deberemos seguirlas también:

- **Mínimo Punto de Exposición (MPE)**: Un servidor sólo debe exponerse en lo que sea estrictamente necesario para su rol, es decir, un servidor Web no debe tener cargado el software de impresión y mucho menos ejecutando demonios de servicio de impresión en red. Esta regla ha hecho que las instalaciones de los sistemas operativos hayan dejado de realizarse orientadas a componentes, es decir, instalando módulos, y se realicen orientadas a roles, es decir, instalando módulos absolutamente necesarios para el cumplimiento de un rol.

- **Mínimo Privilegio Posible (MPP)**: Todo componente dentro de un sistema debe ejecutarse con los privilegios necesarios para cumplir con su rol y nada más. Un sitio web nunca debe correr como root porque no es necesario para dar servicio y lo único que puede suceder es que un atacante que consiga vulnerar el sitio obtenga esos privilegios de root en el sistema.

- **Defensa en Profundidad (DP)**: Se deben implementar todas las medidas de seguridad que sean posibles teniendo en cuanta dos factores: Primero: Una medida de seguridad no debe anular a otra. El ejemplo más claro de esto es cifrar las comunicaciones e instalar un Sistema de Detección de Intrusiones de Red (NIDS), ya que el segundo no podría detectar ataques por red si se usan los canales cifrados. Y Segundo: Las medidas de protección no pueden anular la utilidad de un sistema. Si las medidas de protección hacen que el sistema deje de dar servicio en tiempo útil entonces no son medidas viables.

Las fuentes del servidor

Cuando instalas un servidor Apache tienes varias formas de obtener y mantener los ficheros del servidor. El primer camino es ir directamente al proyecto y descargarte la última versión compatible con tu sistema operativo y después realizar las comprobaciones oportunas de los ficheros mediante el hash y la firma, luego lo configuras, lo compilas y listo. Esto es importante sobre todo cuando te descargas ficheros desde otros Servidores Mirror. La otra opción es bajarte los binarios compilados para tu sistema operativo con lo que deberías realizar la comprobación del hash y la firma de los ficheros pero directamente de los binarios. Procura bajarte los ficheros compilados de los servidores que marcan los fabricantes del

sistema operativo para evitar problemas. Puede suceder, como le pasó a WordPress no hace mucho que vulneren su servidor y directamente el código que te descargas del propio fabricante venga troyanizado, pero es menor.

El elegir correctamente el lugar de la descarga del fichero evita situaciones que a priori parecen algo paranoicas pero existen formas de atacar sistemas desde las fuentes. Si te interesa estar informado sobre estas técnicas, hay documentos, como "Troyanizando Apache y sus Módulos", escrito por Sp4rK, disponibles en Internet. Para este ejemplo he elegido la última versión de Apache disponible de la rama 2.X, la 2.24, como dice en la propia web del proyecto es principalmente una versión bugfix, es decir, correctora de bugs, y es la mejor opción del proyecto Apache. Puedes acceder a los ficheros en la siguiente URL: http://httpd.apache.org/download.cgi y es importante que si estas realizando una actualización de una versión a otra revises el Changelog dónde se indican los principales cambios de cada versión para que no te encuentres con una situación de incompatibilidades con otro software de tu servidor.

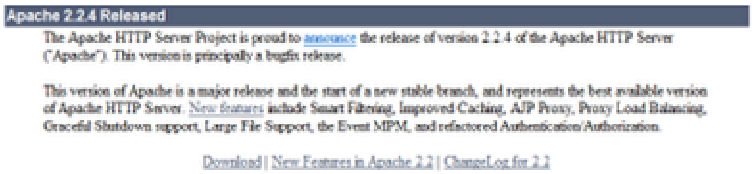

Imagen: Anuncio de versión Apache 2.2.4.

No solo hay que descargarse la última versión sino que además, es importantísimo estar suscrito a las actualizaciones de seguridad del producto pues mantener el servidor actualizado es una de las principales cosas de las que preocuparse para mantener segura una infraestructura con Apache.

Una vez que te hayas descargado puedes accede tanto a los fuentes del archivo, como al fichero de hash MD5 y a la firma pública de los códigos fuente y puedes comprobar que el fichero es el mismo que ellos publican realizando, en primer lugar una comprobación del hash y en segundo lugar una comprobación de la firma. Para que puedas realizar la comprobación de la firma es necesario que tengas las firmas de los distribuidores de los ficheros del proyecto que puedes descargar de esta URL: http://www.apache.org/dist/httpd/KEYS

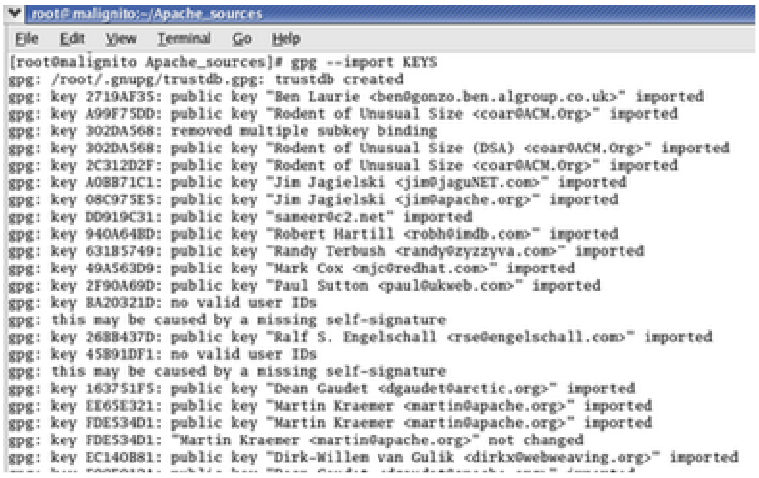

Imagen: Importación de claves con gpg --import KEYS

Y luego verificamos la firma con *gpg --verify http-2.2.4.tar.gz.asc* que es el fichero con la firma de las fuentes y el hash md5 generando un nuevo hash del fichero con el comando *md5 http-2.2.4.tar.gz* y comprobando si el valor es el mismo que nos ofrecen en el fichero *http-2.2.4.tar.gz.md5*.

Imagen: Verificación de firma y hash

Compilación de los archivos fuente

Antes de empezar a configurar los parámetros del servicio debemos configurarlo. En esta parte se deben realizar una serie de acciones que deben ser tenidas en cuenta.

Configuración del banner del servidor Web.

Una de las reglas de oro es dar la menor información posible sobre tu sistema, para reducir el número de atacantes. Muchos escáneres de vulnerabilidades realizan una comprobación del banner que devuelve el servidor web para poder aplicar unos u otros exploits. Una de las recomendaciones es cambiar dicho banner. Para cambiar el banner debemos tocar el archivo ap_release.h antes de configurar el servidor.

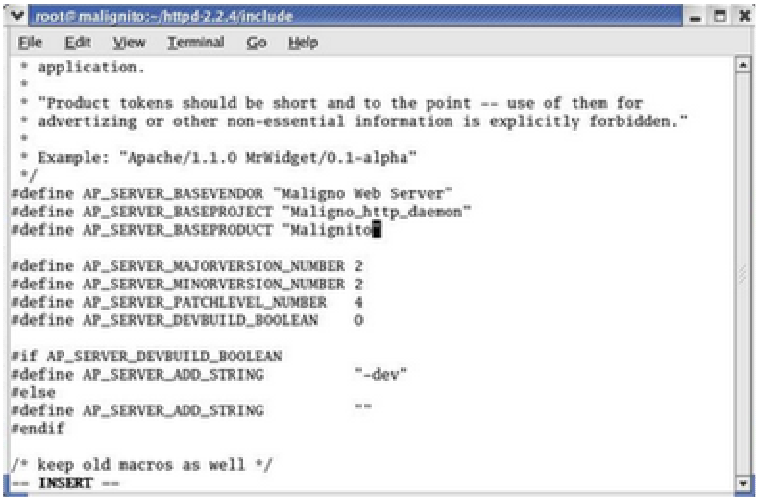

Imagen: include/ap_release.h

Como se puede ver en la captura se puede cambiar la versión de Apache, yo le he dejado la 2.2.4.0. Cambiar ese banner ayudaría a que Apache bajara en las estadísticas de Netcraft, tan llevadas y traídas en el mundo de la competición software, así que si no quieres dar ninguna información pero quieres que las

estadísticas sigan contando "un Apache más" te recomiendo que pongas un genérico Apache utilizando la clave ServerTokens Prod en el archivo httpd.conf. Yo por mi parte le he bautizado con el nombre de Malignito.

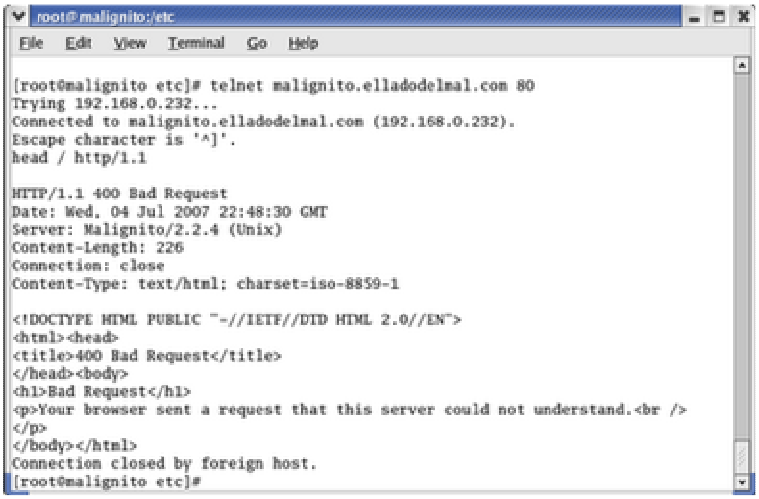

Imagen: Respuesta ante conexión por el puerto 80 una vez que el servidor ya está corriendo

Configuración e Instalación

Una vez que hayas cambiado el banner, podemos pasar a configurar los módulos que quieres habilitar de tu servidor Apache, para ello, antes de realizar la configuración de los fuentes te recomiendo que analices bien que necesitas tener cargado y que no necesitas tener cargado. Todos los módulos que hay para Apache están documentados en la siguiente URL: http://modules.apache.org/ y tienes una lista reducida en la siguiente URL: http://httpd.apache.org/docs/2.0/es/mod/. Actualmente hay más de 400 módulos disponibles para Apache, luego es importante que sepas los que quieres tener funcionando y los que no. Algunos módulos que se suelen cargar por defecto y no suelen ser necesarios pueden ser:

- *mod_imap*: Que ofrece servicio de mapeo automático de ficheros de índice del lado del servidor.
- *mod_include*: Habilita los includes de ficheros del lado del seridor. Los .shtml.
- *mod_info*: Da información sobre el servidor. Los escáneres rastrean la información que ofrece. Se suele habilitar en pruebas y desarrollo, pero no en producción.
- *mod_userdir*: Para mapear los directories personales de los usuarios. También está el *mod-ldap-userdir* para hacerlo vía árboles ldap.
- *mod_status*: Para tener estadísticas.
- *mod_cgi*: Ofrece soporte para ejecución de cgis. Si no tienes programas cgi en tu servidor deshabilítalo.
- *mod_autoindex*: Listados de directorio para cuando no hay archivo por defecto.

Si ya has realizado tu selección de módulos puedes proceder a su configuración con el siguiente comando:

Imagen: Configurando módulos en Apache.

Como se puede ver basta con usar el comando *--disable-modulo* para evitar que se cargue un determinado módulo en el sistema. Lógicamente solo podrás habilitar o deshabilitar aquellos módulos que tengas cargados, si deseas agregar nuevos módulos estos deben ser agregados mediante el comando config-status.

Compilación

Una vez que tengas configuradas las listas de módulos que necesitas puedes pasar a la ejecución del comando make y make install para acabar de instalar Apache y cuando lo tengas listo, podrás ver la lista de módulos que tienes cargados con el comando *httpd –l*.

Imagen: Lista de módulos cargados

El último comando que se debe revisar es httpd –V que nos permitirá ver cuáles son las opciones de compilación con que se ha instalado el servidor.

Fortificando un Servidor Apache (II de IV)

****
Artículo publicado en:
http://elladodelmal.blogspot.com/2007/09/fortificando-un-servidor-apache-ii-de.html
****

Configuración del httpd.conf

Cuando ya tengamos instalado de forma segura el servidor Apache, deberemos pasar a configurar el archivo httpd.conf de manera que nuestro sistema quede fortificado, para ello vamos a ver una serie de parámetros importantes. Como recomendación importante de seguridad, ante posibles fallos de otros servicios del sistema, es recomendable que sólo el administrador del sistema tenga acceso a los archivos de configuración del sistema para evitar que otro usuario pueda manipularlos.

Las cuentas del sistema

Para gestionar el servidor web se necesitan una serie de cuentas que hemos de configurar, en ellas vamos a aplicar los principios de Mínimo Privilegio Posible. Para ello en primer lugar vamos a crear un grupo para los usuarios que puedan gestionar el servidor, es decir, los que puedan administrarlo y detener y arrancar los servicios necesarios que llamaremos web_admins y otro grupo para los usuarios que van a representar a los servidores Web, que llamaremos web_servers.
Para correr el servidor Web, aplicando el MPP, vamos a utilizar un usuario que no tenga ni Shell ni se pueda logar, podemos utilizar el usuario nobody o crear uno para dar aun menos información a un posible atacante. Para ello creamos un usuario del grupo web_servers com home en el directorio dónde vayamos a publicar los documentos y por seguridad le configuraremos un Shell de comandos inexistente.

#useradd –d /opt/apache2/htdocs –g web_admins –s /bin/noshell web_user

Y luego le bloquearemos la cuenta para que no pueda tener login:

#passwd –l web_user

Y comprobamos en el archivo de configuración de cuentas /etc/passwd que el usuario está bien creado – con la shell falsa, el grupo correspondiente y el home correcto - y en el archivo de configuración de contraseñas /etc/shadow que la cuenta está bloqueada, es decir, que tiene el signo de cierre de admiración "!".

Por último comprueba que no puedes ejecutar login interactivo:

#login web_user

Para hacer que nuestros servicios http corran con estas credenciales deberemos modificar el archivo de configuración httpd.conf en las claves User y Group:

User web_user
Group web_users

Si todo ha ido bien, cuando ejecutemos el demonio httpd con httpd –k start, éste se ejecutará en el sistema con las cuentas no privilegiadas que hemos creado.

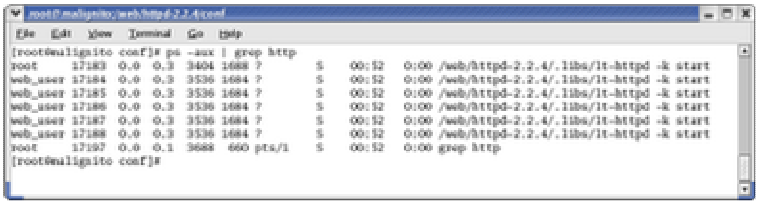

Imagen: Privilegios de los demonios httpd

Ocultar Información

Ya hemos hablado antes de la clave ServerTokens Prod para ocultar información en el banner, pero hay otras claves que ayudan a ocultar más información del sistema.

- ServerSignature Off: Evita que se envíe información sobre la versión del servidor en las páginas de error.

- ErrorDocument numError errores/paginaError.html: Esta directiva nos permite cambiar las páginas de respuesta de errores ante cada uno de los tipos de error caracterizados por su número.

Ajustando algunos Límites

Cuando tenemos un servidor web expuesto que puede ser atacado es recomendable ajustar algunos límites de opciones para evitar el daño que pueda producir un ataque.

- Valor de Timeout: Este es el valor que una conexión inactiva va a estar mantenida por el servidor antes de que sea cerrada. El valor por defecto es de 300 segundos pero en un eventual ataque de denegación de servicio a generado a baso de abrir conexiones el impacto será mucho menor con valores de timeout menores. Reduce el valor de la variable Timeout a 40 o 50 segundos.

- Límites del tamaño de las peticiones: A la hora de mantener la conversación entre el cliente y el servidor se pueden configurar límites en cada uno de los apartados tanto de la petición como de la respuesta. Una petición de envío de 4 megas no permitirá subir ficheros al servidor de más de ese tamaño pero si no quieres que se suban ficheros más grandes puedes reducir el impacto de ataques. Algunos valores que puedes ajustar para ajustar el funcionamiento de tu servidor son LimitRequestBody, LimitRequestFields, LimitRequestFieldSize, LimitRequestLine o LimitXMLRequesBody si estás trabajando con documentos XML.

- Concurrencia: Debes ajustar la concurrencia de usuarios a los límites que soporta tu infraestructura, para ello puedes ajustar los valores MaxClients para definir el número de procesos hijos que pueden ser creados en peticiones http. Por defecto el valor es de 256 luego si tu sistema no soporta estas usuarios concurrentemente tal vez sea recomendable limitar el tope. Esto puede evitar ataques de Denegación de Servicio que bloqueen la memoria de tu sistema. En entornos de multiproceso podemos regular, para optimizar y limitar el consume de recursos los valores de StartServers, MaxSpareServers, MaxRequestsPerChild, ThreadsPerChild, ServerLimit, y MaxSpareThreads. Estas directivas van a regular el número de procesos que se van a abrir, el número de procesos hijos en espera que se pueden tener, el número de threads por proceso, etc… Configura estos valores adecuadamente en entornos críticos y de alto rendimiento. Para realizar un buen ajuste de estos valores es necesario que realices un profile del rendimiento de tu sistema antes.

Permisos sobre Directorios

Al final el servicio httpd va a servir documentos que están almacenados en el sistema de ficheros por lo que es importante securizar en que partes se tiene permisos y en cuales no, para ello por defecto se pueden configurar las opciones más restrictivas sobre el sistema de ficheros y luego configurar en cada directorio las opciones particulares.

Para restringir el sistema de ficheros debemos configurar algo como:

[Directory /]
Order Deny, Allow
Deny from all
Options None
[/Directory]

En esta sección del archivo de configuración estamos restringiendo el acceso a todos los usuarios a todos los ficheros del directorio / y además, con Options None estamos prohibiendo el listado de directorios, la ejecución de programas CGI, la posibilidad de seguir links simbólicos y la de inclusión de documentos del lado del servidor.

La directiva Order indica el modo en que se van a procesar las peticiones, es interesante la manera de funcionar de la misma ya que los resultados de las directivas Deny o Allow dependerán de la configuración de Order.

	Order Allow, Deny	Order Deny, Allow
Concuerda con Allow	Permitido	Permitido
Concuerda con Deny	Denegado	Denegado
No Concuerda con ninguna	Denegado	Permitido
Concuerda con ámbas	Denegado	Permitido

Tabla Opciones Order

Esto quiere decir que al elegir Order Allow, Deny primero se ejecutarán todos los Allow y luego todos los Deny y si una situación no queda reflejado, es decir, no está explícitamente permitido o hay un conflicto, es decir, está explícitamente permitido y explícitamente denegado, entonces se Deniega el acceso. Por el contrario la configuración Order Deny, Allow funciona a la inversa. Para asegurar la restricción a todos los clientes se añade, en la configuración inicial la opción Deny from all.

A partir esta configuración podemos ir permitiendo o no opciones en cada uno de los directorios, por ejemplo, en el directorio principal dónde vamos a guardar los documentos, generalmente htdocs podremos permitir el acceso añadiendo:

[Directory /htdocs]
Order Allow, Deny
Allow from all
[/Directory]

Las opciones Allow y Deny nos van a permitir configurar reglas de concesión o denegación de acceso a los clientes. Estas reglas pueden aplicarse a todos "all" o a un dominio de origen, o a una dirección IPv4 o a una dirección IPv6 o incluso utilizando variables de entorno a aquellos que cumplan un determinado valor, como por ejemplo, que tengan un determinado cliente.

Podríamos permitir el acceso solo al dominio de nuestra organización:

Allow from elladodelmal.com

O denegar el acceso a todos lo que tuvieran un cliente Firefox:

```
SetEnvIf User-Agent ^Firefox/2\.0 NO_PASA
[Directory /docroot]
Order Allow, Deny
Allow from all
Deny from env=NO_PASA
[/Directory]
```

En este ejemplo hemos creado, con la directiva SetEnvif una variable de entorno para aquellos clientes con Firefox/2.0 cuyo valor es NO_PASA. Lógicamente estas opciones también pueden configurarse en los firewalls de acceso, pero, siguiendo la regla de Defensa en Profundidad también debería configurarse en los directorios. Además, es posible que, en accesos en la red local no se pase por algún firewall a nivel de aplicación que pueda inspeccionar rutas de acceso en el protocolo http por lo que es conveniente configurar estas opciones de seguridad en el servidor web.

Si queremos configurar alguna opción en concreta para un directorio, por ejemplo para el directorio de programas cgi, para ejecutables, o si queremos habilitar listados de directorios lo podemos hacer con la directiva Options. Situando un – antes de la directiva esta queda deshabilitada, de lo contrario se habilita.

- All: Representa a todas las opciones. Esto indicaría que todas quedan habilitadas, a excepción de Multiviews.

- None: No queda habilitada ninguna opción.

- ExecCGI: Con esta opción vamos a permitir o denegar la ejecución de scripts cgi de un directorio a través del modulo mod_cgi.

- Indexes: Esta opción permitirá el listado de archivos de un directorio a través del módulo mod_indexes si no se encuentra el archivo DirectoryIndex.

- Includes: Permite la inclusión de archivos del lado del servidor a través del modulo mod_include. Puede ser un riesgo frente a ataques Remote File Inclusion. Es conveniente deshabilitarlo si no se van a usar en ninguna aplicación.

- IncludesNOEXEC: Permite el uso de Includes del lado del servidor pero deshabilita las directivas exec cmd y exec cgi. Esta directiva sí que permite el uso de incluir de forma virtual scripts CGI desde directorios especificados con usando ScriptAlias.

- FollowSymLinks: Permite al usuario navegar por los directorios a través de enlaces simbólicos. Es recomendable deshabilitarlo.

- SymLinksIfOwnerMatch: Permite el seguimiento de links simbólicos solo si el fichero o directorio final es del mismo usuario que el enlace.

- MultiViews: Se permiten Multiviews de contenido, a través del uso del módulo mod_negotiation.

Unos ejemplos de esto serían:

- Options –Indexes ExecCGI –Includes: En este ejemplo se permite la ejecución de

scripts CGI pero no se permite el listado de directorios ni la inclusión de documentos del lado del servidor para evitar una eventual inclusión de una Shell.

- Options None: No se permite ninguna opción.

Fortificando un Servidor Apache (III de IV)

****
Artículo publicado en:
http://elladodelmal.blogspot.com/2007/11/fortificando-un-servidor-apache-iii-de.html
****

En estas dos últimas partes, vamos a tratar de varios aspectos importantes en la fortificación de un servidor Apache, a saber, en primer lugar vamos a ver como configurar los sistemas de cifrado de comunicaciones con SSL/TLS con nuestros sitios web; en segundo lugar la configuración de los registros de actividad para su posterior análisis, es decir, los logs; en tercer lugar vamos a ver los sistemas de autenticación de usuarios y por último como configurar *mod_security* para filtrar las URLs de acceso a nuestros sitios. En esta tercer parte Mod_SSL y la Gestión de Logs.

Mod_SSL

El protocolo de comunicaciones SSL/TLS se utiliza para cifrar la comunicación entre dos equipos y autenticar a los participantes de la misma. Este artículo no pretende mostrar el funcionamiento del sistema SSL, ampliamente descrito en numerosos artículos desde su aparición, sino simplemente como se debe configurar en un servidor Apache para que las conexiones a nuestro servidor estén cifradas y, en caso de que así se desee, también autenticadas.

Siempre que se utilice SSL/TLS estamos cifrando las comunicaciones extremo a extremo, sin embargo el proceso de autenticación requiere de una configuración con más cuidado. Si se desea autenticar al servidor, es decir, que los clientes tengan la certeza de que se están comunicando con el servidor que ellos desean, es necesario utilizar un certificado emitido por una Entidad Emisora de Certificados contrastable por los usuarios de nuestro sistema, o lo que es lo mismo, una Entidad en la que los clientes confíen y tengan la clave pública de esta instalada en su máquina. Si esto no se produce, el uso de SSL ayuda a cifrar las comunicaciones pero no ayudará a detectar un certificado falso (fake) emitido por un atacante en medio. Es por ello, a pesar de que en el presente artículo, y, como muestra didáctica únicamente, se utiliza un certificado emitido por nosotros mismos, se recomienda utilizar un certificado de servidor emitido por una CA de confianza para nuestros usuarios.

Con SSL, además de autenticar el servidor, se pueden autenticar a los clientes mediante certificados digitales a la hora de iniciar la conexión con SSL, aunque esta no es una práctica muy extendida debido a la complejidad en el despliegue y mantenimiento de los certificados de los clientes.

Para realizar un ejemplo de configuración, en este artículo hemos utilizado un Certificado Digital para servidores web creado por nosotros. Hemos utilizado para ello la utilizad OpenSSL [http://www.openssl.org] y hemos creado uno como se puede ver en la siguiente captura.

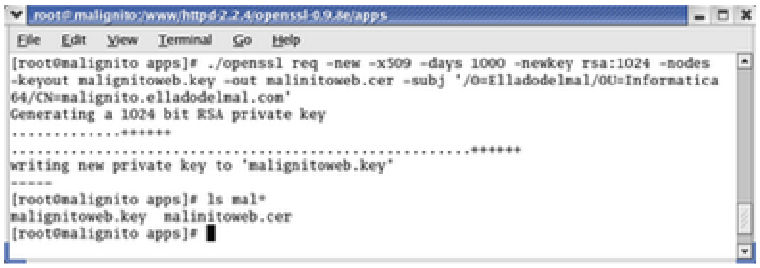

Imagen: Creación de un certificado digital con openssl

El proceso que hay que seguir para la creación del certificado es sencillo. Primero hay que instalar la herramienta openssl siguiendo los pasos de configuración (*./configure*), construcción (*make*) e instalación (*make install*). Una vez que tengamos funcionando la herramienta en nuestro sistema podremos crear el certificado con el comando que se ve en la imagen.

Con ese comando le estamos diciendo a la herramienta que nos cree un certificado X.509 con una caducidad de 1000 días, es decir, pasados estos 1000 días será necesario renovar el certificado. Este certificado va a ser creado con una clave de 1024 bytes y se van a generar dos ficheros, uno el del certificado y otro el de la petición y no van a estar protegidos por contraseña. Como se puede apreciar, hemos pedido que el certificado se emita para nuestro servidor al que hemos nombrado como "*malignito.elladodelmal.com*".

Con el certificado digital emitido ya tenemos lo necesario para configurar el soporte SSL en nuestro servidor web. Lo primero que hemos de preparar el que el servidor Apache cargue el módulo SSL, para ello, como vimos en el artículo del mes pasado, realizando una configuración inicial podemos utilizar la opción de preparar el servidor con soporte para el *mod_ssl* usando *configure –prefix=ruta –enable-ssl*. Algunas distribuciones de Linux, como Debian, ofrecen una versión de Apache acompañada de algunas herramientas que ayudan en la gestión de módulos como a2enmod que permite agregar módulos a Apache. Como vimos en las partes anteriores, con el comando *httpd –l* se puede comprobar la lista de módulos que están activos.

Imagen: Lista de módulos cargados en Apache. Mod_ssl cargado.

Ya está cargado el modulo SSL en el servidor Apache, ahora deberemos configurar una serie de parámetros para dar soporte a SSL a nivel de servidor o de Virtual Host. Para ello se deben configurar las opciones en el archivo httpd.conf. SSL viene acompañado de muchas opciones y es recomendable, para ajustes especiales, consultar la documentación de mod_ssl que está disponible en la siguiente URL http://httpd.apache.org/docs/2.2/en/mod/mod_ssl.html. No obstante, la siguiente tabla muestra las configuraciones que se deben realizar para que nuestro servidor esté funcionando correctamente:

- ***SSLEngine on/off***: Este parámetro activa el uso de SSL en nuestro servidor. Si, la configuración del servicio fuera errónea el soporte no se activaría y puede llegar a no levantar los demonios de Apache, luego es importante tener correctamente configurado el servicio antes de ponerlo activo.

- ***SSLProtocol***:Este parámetro se utiliza para determinar cuáles van a ser los protocolos de cifrado que se van a utilizar en nuestro servidor. Hay que tener en cuenta que cuando se produce el "handshake" o saludo inicial entre el cliente y el servidor, estos negocian el protocolo a utilizar. Si no deseamos que se utilice un protocolo antiguo o inseguro debemos deshabilitar el uso de todos a excepción de los protocolos seguros. Esta acción puede producir problemas de acceso en clientes antiguos. La lista de protocolos que vienen con SSL son: SSLv2, SSLv3, TLSv1.

- ***SSLCipherSuite***: Una vez elegido el protocolo SSL a utilizar, en mod_ssl podremos configurar las opciones de cifrado, para ello podemos elegir los algoritmos de generación de clave, decantándonos por el uso de RSA o de Diffie-Hellman con claves RSA o Diffie-Hellman con claves DSA, etc… Así mismo podremos elegir los algoritmos de firma, de codificación, y las longitudes de cifrado a usar. Es decir, podemos realizar un ajuste fino de la criptografía que nos va a

permitir securizar las comunicaciones hasta nuestro deseo. Hay que tener en cuenta que el deshabilitar ciertas opciones de cifrado puede generar conflictos con clientes que no tengan una suite criptográfica amplia y moderna.

- ***SSLOptions***: Este parámetro se va a utilizar para configurar diferentes comportamientos en diferentes situaciones. LA opción +StrictRequire se va a utilizar para deshabilitar el acceso por medio http a aquellas rutas en las que se exija SSL.

- ***SSLCertificateFile***: Ruta al archivo del certificado del sitio

- ***SSLCertificateKeyFile***: Ruta al archivo key del certificado.

- ***SSLCACertificateFile***: El certificado digital de la Entididad Certificadora.

- ***SSLCARevocationFile***: Archivo dónde se encuentra la CRL (Lista de Certificados Revocados).

- ***SSLRequire***: Este parámetro se utiliza para exigir un cumplimiento de opciones SSL a la hora de acceder a una determinada ruta del servidor. Se utilizan expresiones regulares para poder afinar las restricciones.
SSLRequireSSL Parámetro para forzar el uso de http-s en un determinado directorio. Si está configurada la opción +StricRequire se prohibirá el uso de http.

En la siguiente captura tenemos una configuración válida del archivo httpd.conf que levanta los demonios httpd con soporte SSL. Como se puede ver se ha habilitado la escucha por el puerto 443, que es el que por defecto se utiliza para el protocolo https-s. Se han configurado los parámetros SSLEngine on para levantar los servicios SSL, SSLCertificateFile y SSLCertificateKeyFile con las rutas a los ficheros creados con el comando openssl; se ha activado el soporte para TLSv1 y SSLv3 en exclusividad y se ha configurado la suite de cifrado con soporte Medio y Alto y el uso de protocolos de firma SHA1 y MD5.
Después, en la ruta de acceso a los documentos se ha exigido el uso de SSL con el parámetro SSLRequireSSL.

```
Listen 192.168.0.232:80
Listen 192.168.0.232:443
<IfModule !mpm_netware_module>
User web_user
Group web_users
</IfModule>
ServerAdmin maligno@elladodelmal.com
ServerName malignito.elladodelmal.com:443
SSLEngine on
SSLOptions +StrictRequire
SSLProtocol -all +TLSv1 +SSLv3
SSLCipherSuite HIGH:MEDIUM:!aNULL:+SHA1:+MD5:+HIGH:+MEDIUM
SSLSessionCacheTimeout 600
SSLCertificateFile /www/httpd-2.2.4/certificado/malinitoweb.cer
SSLCertificateKeyFile /www/httpd-2.2.4/certificado/malignitoweb.key
DocumentRoot "/web/elladodelmal"
<Directory />
    Options -Indexes FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>
<Directory "/web/elladodelmal">
    Options FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
    SSLRequireSSL
</Directory>
```

Imagen: Configuración httpd.conf para soprote SSL.

Con esta configuración, cuando se levanta el servicio http y se intenta acceder a la ruta por defecto de nuestro servidor se obtendrá un mensaje de error, que puede ser personalizado como se explicó en el artículo del mes pasado, en el que nos avisa de que es estrictamente necesario el uso de *httpd-s*.

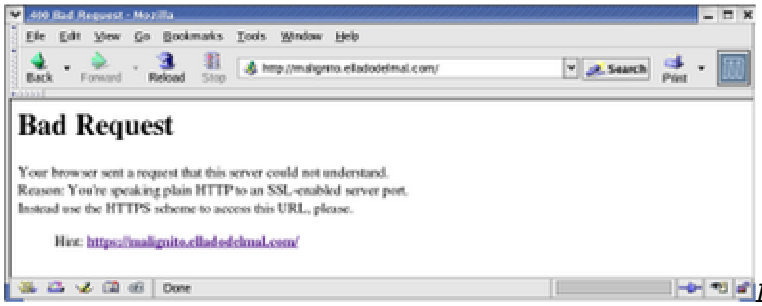

Imagen: Error, la conexión debe realizarse por medio de http-s.

Si se accede al servidor mediante el uso de una conexión http-s el acceso será concedido y se podrá consultar toda la información sobre el certificado digital que se está utilizando el servidor.

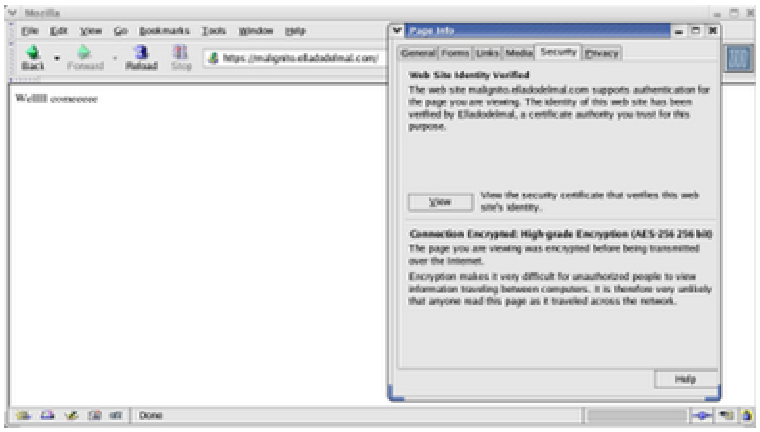

Imagen: Acceso mediante http-s

Gestión de Logs

La configuración de los registros de acceso y los eventos que ser producen en un servidor Web son de especial importancia pues pueden ayudar a detectar preventivamente un ataque al sistema, una debilidad que está siendo explotada o simplemente un fallo en la configuración o en el sistema. Para ello Apache dispone de herramientas flexibles que van a permitir una selección flexible de lo que se quiere guardar registro, dónde y en que formato se va a guardar esta información. Apache gestiona, por defecto, tres tipos de registros: El registro de errores, el registro de acceso y el registro del identificador del proceso (PID) del demonio del servicio. El registro de errores se guarda en un fichero que es marcado por la directiva errorlog. Esta directiva también puede utilizarse para, en lugar de guardar la información en ficheros, utilizar el servicio syslog para registrar los eventos vía red y de forma centralizada. El uso del servicio syslog será de especial utilidad cuando tenemos sistemas de análisis de log centralizados o de correlación de

eventos.

ErrorLog /ruta/ficheros/log/error_log ó *ErrorLog syslog:user*

El nivel de información que se va a almacenar en los ficheros de registro se configura con la directiva LogLevel en la que se pueden ajustar diferentes valores:

LogLevel

- ***Emerg***: Sólo se almacenan los mensajes que dejan al sistema incapaz de ser utilizado.

- ***Alert***: Cuando se produce un error en el sistema que requiere la ejecución inmediata de una acción para corregirlo.

- ***Crit***: Fallos críticos del sistema. No requieren acción inmediata pero pueden dejar el sistema no disponible.

- ***Error***: Condiciones de error en el uso del sistema. No tiene porque afectar al uso del sistema.

- ***Warn***: Avisos. Se producen cuando algo no está realizándose correctamente. Puede producirse por un script o un cliente que no realiza la negociación correctamente con el servidor.

- ***Notice***: Información significativa del funcionamiento del sistema.

- ***Info***: Información general del sistema.

- ***debug***: Información de debuging del sistema. Cuando abre o cierra conexiones o ficheros, etc…

Cuando se establece un valor de LogLevel, se van a registrar todos los eventos de mayor o igual importancia que la que se ha configurado, de tal manera, que si se establece el valor warn, se van a registrar los eventos de tipo warn, error, critical, alert y emergency.

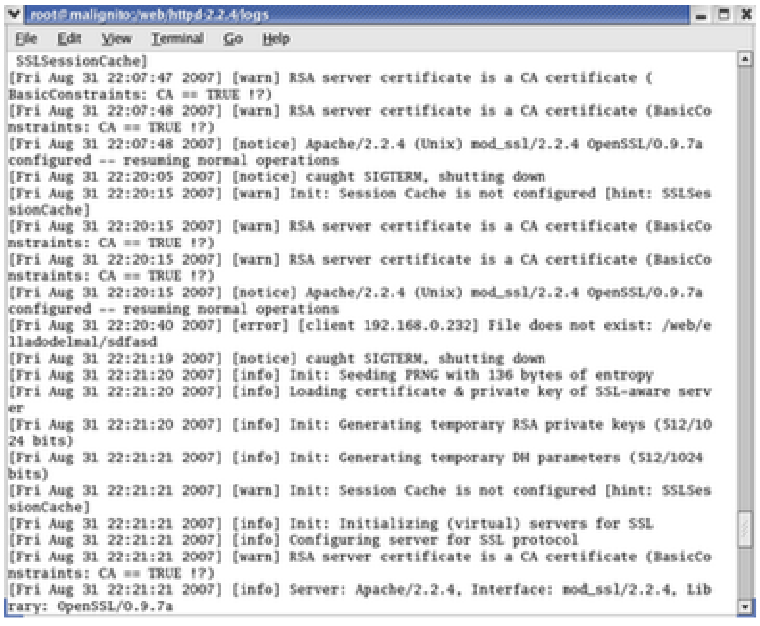

Imagen: Archivo error_log configurado con nivel notice

Como se puede ver en la imagen anterior, se ha configurado el nivel notice, y así, por ejemplo, se puede observar el registro de los eventos info en los que se muestra la información relativa a la negociación SSL entre clientes y servidores. En un servidor de mucho tráfico este nivel de detalle puede generar un colapso en los almacenes de log.

Los registros de acceso al servidor se almacenan aparte de los errores del servidor. Estos registros guardan información relativa a todos los accesos a documentos o intentos de acceso a los documentos. Para almacenar estar información Apache se apoya en el uso de dos módulos, que son *mod_log_confing* y *mod_setenvif*. El primero se utiliza para configurar el lugar y el formato de los ficheros de registro y el segundo para utilizar variables de entorno que permitan generar ficheros de registro condicionales, que serán muy útiles a la hora de detectar sucesos en entornos con mucho tráfico.

Para configurar los registros de acceso utilizamos las directivas *LogFormat* y *CustomLog*.

LogFormat determina que información se va a escribir y de qué forma, en el fichero de registro. La forma común es tiene esta estructura:

LogFormat "%h %l %u %t \"%r\" %>s %b" common

- %h: IP del cliente.
- %I: Identificación del cliente utilizando el servido indentd. Esta información solo será accesible si Apache se ha configurado con Identitycheck activado. Esto puede generar un cuello de botella en la creación de los ficheros log, por lo que sólo se debe utilizar en entornos de Intranets, redes privadas, etc…, controlados.
- %u: Usuario cliente.
- %t: Fecha y hora.
- %r: Petición realizada
- %s: Código de status del servidor.
- %b: Tamaño de la respuesta.

Con la directiva *LogFormat* te puedes crear el formato de fichero log que a ti te convenga utilizando estas variables o añadiendo información de cabeceras del cliente como:

LogFormat "%h %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"" miformato

Aquí hemos añadido % {Referer} y %{User-Agent} al formato de registro y hemos quitado %I y %u.

Una vez definido el formato que a utilizar hemos de marcar el lugar de registro con:

CustomLog /ruta/logs/access_log miformato

Podemos crear tantos ficheros de log como deseemos, así por ejemplo, en uno podemos almacenar la dirección IP y la hora y en otro La dirección IP, la petición y el referer.

Control de Logs

Los ficheros de log suelen tener un rápido crecimiento por lo que deben ser

controlados y tener un mantenimiento muy ajustado. Para ello se pueden utilizar los sistemas de log rotacionales, que permiten, utilizando el programa externo rotatelog crear archivos automáticos cada cierto tamaño o tiempo. Para ello definimos en ErrorLog el formato de los ficheros:

ErrorLog "|bin/rotatelogs /var/logs/errorlog.%Y-%m-%d-%H_%M_%S 50M"

Es importante el uso del pipe para la dirección de los registros de logs. En este ejemplo se utilizan las variables de fechar %Y (year), %m (month), %d (day),… para crear un nuevo archivo de log cada 50 Megabytes de tamaño.

O directamente con ***CustomLog***, que creará *logapache.nnnn* (nnnn el tiempo de creación)

CustomLog "|bin/rotatelogs /ruta/logs/logapache 50M" miformato

Otra opción para gestionar los logs en Apache es el uso de *cronolog*. Su funcionamiento es similar al del uso de logs rotacionales pero permite configuraciones más flexibles. No viene con Apache y hay que descargarlo desde la siguiente URL: http://cronolog.org/

Fortificando un Servidor Apache (IV de IV)

****

Artículo publicado en:
http://elladodelmal.blogspot.com/2007/11/fortificando-un-servidor-apache-iv-de.html

****

mod_chroot

Una de las características que se siguen por seguridad es "enjaular" los procesos de un sistema. El termino enjaular se utiliza para referirnos a diferentes grados de aislamiento del proceso dentro del sistema operativo. Se puede enjaular un proceso dentro del sistema de ficheros, dentro de su parcela de memoria o incluso en las funciones de red. Una de las características más comunes es aislar los demonios de los servidores web dentro un sistema de ficheros virtual mediante el uso de las funciones chroot.

Realizar un chroot a un proceso siempre ha sido una tarea ardua, ya que hay que proveerle a dicho proceso de todas las utilizadas y ficheros de configuración que pudiera necesitar dentro de una nueva estructura de ficheros. Esto es así para muchos servicios pero desde la aparición de mod_chroot no es necesario en Apache.
Este módulo se carga dentro del sistema y ya se encarga él de aislar el sistema de ficheros a los que se van a enlazar los demonios httpd y todos sus hijos. Esto se debe a la implementación a nivel de sistema operativo de la función chroot() en las principales distribuciones Linux.

Para ello descargamos el modulo *mod_chroot* [http://core.segfault.pl/~hobbit/mod_chroot/] y lo incluimos dentro de los módulos del servidor Apache. Podemos realizar una compilación estática, con *./configure –add-module modules/mod_chroot* o una compilación dinámica utilizando el DSO (*Dynamic System Object*) mediante el siguiente comando:

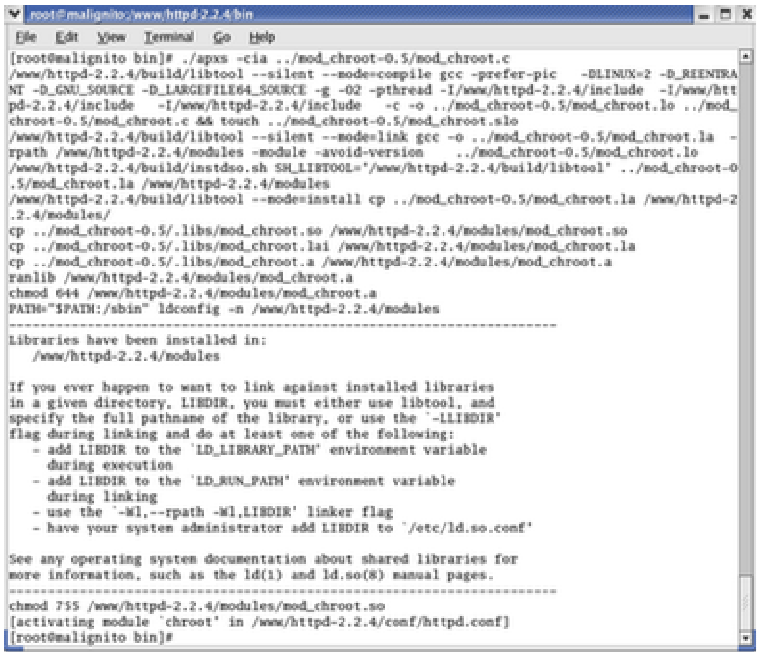

Imagen: Carga dinámica de mod_chroot

Una vez cargado el módulo, se procederá a configurar el archivo de configuración httpd.conf para decidir en qué directorio queremos enjaular los procesos httpd. Para ello se configura la directiva ChrootDir como se puede ver en la siguiente captura.

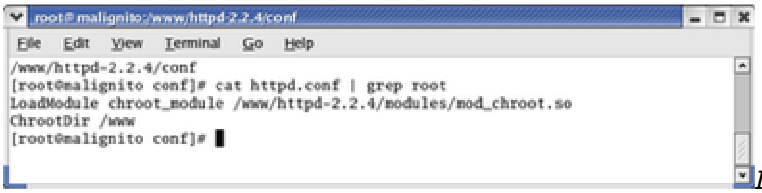

Imagen: Configuración mod_chroot en httpd.conf

Mod_chroot no realiza aislamiento de memoria, y en algunos casos puede requerir de la configuración de algún directorio especial dentro del sistema de ficheros "chrooteado" sobre todo para servicios de multiprocesamiento simétrico o para la inclusión dentro del servicio httpd del soporte de alguna función especial, pero como se puede comprobar, utilizando mod_chroot para Apache, es mucho más sencillo a como tradicionalmente había que configurar un soporte chroot para cualquier otro servicio.

Autenticación y Autorización

Autenticar es el mecanismo por el cual garantizamos que alguien es quien dice ser y autorizar es permitir que ese alguien pueda acceder a un determinado recurso. Para realizar estas dos funciones en Apache utilizamos en primer lugar un tipo de autenticación y un proveedor de autenticación. El tipo de autenticación es la forma en la que el cliente envía sus credenciales, esta puede ser en texto claro (Basic) utilizando el módulo *mod_auth_basic* o en codificación MD5 Digest utilizando *mod_auth_digest*. La elección de la autenticación se configura con la directiva *AuthType Basic* o *AuthType Digest*. Esto determina la forma en la que se transmiten las credenciales. Lógicamente en un entorno fortificado no se recomienda el uso de la autenticación Basic.

Una vez elegido el tipo de autenticación se debe configurar el proveedor de autenticación, o lo que es lo mismo, dónde está el repositorio de credenciales contra el que se deben comprobar las enviadas por el cliente. Estos repositorios pueden ser un árbol Ldad, un fichero del sistema operativo, una tabla en una base de datos, etc… Lógicamente, la configuración necesaria para cada entorno de autenticación será diferente.

Si se configura la autenticación utilizando un fichero, utilizando el módulo *mod_authn_file*, se debe crear dicho fichero previamente utilizando las herramientas *htpasswd* o *htdigest*. Estas herramientas nos crearán un fichero de usuarios, que deberemos situar fuera de la ruta pública del servidor Apache y se configurará en httpd.conf la directiva *AuthUserFile* como se ve en el siguiente ejemplo:

AuthType Basic
AuthName "Carpetas Privadas"
AuthUserFile /web/usuarios

Si se desea utilizar una tabla en una base de datos para autenticar a los usuarios,

utilizando el módulo *mod_authn_dbd*, se debe configurar el conector con la base de datos, y la tabla de usuarios que se va a utilizar. El siguiente ejemplo muestra una posible configuración de *httpd.conf*. En primer lugar hay que cargar el driver de la base de datos con la que deseemos conectar. Este driver debe estar previamente cargado en el sistema operativo.

DBDriver pgsql

En segundo lugar se deberá configurar la cadena de conexión a la base de datos utilizando la directiva DBDParams:

DBDParams "dbname=mibasededatos user=miusuario password=mipassword"

Y por ultimo configurar las opciones de autenticación del servidor.

[Directory /www/privado]
AuthType Basic
AuthName "My Server"
AuthBasicProvider dbd
AuthDBDUserPWQuery "select password from authn where username = %s"
[/Directory]

Quizá, la forma más interesante de autenticar un sistema de usuarios complejo sea el uso de árboles LDAP. Para ello, se utilizan los módulos *mod_authnz_ldap* y *mod_ldap*. El primero se utiliza para autenticar y autorizar el acceso a recursos mediante la comprobación en árboles LDAP mientras que el segundo optimiza la gestión de memoria e hilos de procesos en las conexiones LDAP.

LDAP es un sistema jerárquico de objetos organizado en forma de árbol. Cada objeto pertenece a su clase y añade propiedades de su clase y propiedades heredadas de las clases madre. En un sistema LDAP se pueden dar de alta, usuarios, recursos y cualquier tipo de objeto que se desee. Cada elemento es posible referenciarlo mediante una descripción de la ruta que ocupa en el árbol o mediante el cumplimiento de unas condiciones de un filtro de búsqueda.

Para configurar la autenticación LDAP se deben configurar la directiva *AuthLDAPURL* que es la cadena de conexión al árbol LDAP y tiene la siguiente estructura:

ldap://host:port/basedn?attribute?scope?filter

- *Host*: Servidor dónde reside el árbol LDAP.
- *Puerto*: Generalmente el 389.
- *Basedn*: Es la ruta a un punto del árbol a partir del cual se realizan las búsquedas. Puede ser la raíz del árbol o una rama del mismo. Tiene la estructura de una ruta LDAP, es decir, ou=unidad_organizativa, dc= mi_dominio, o=organización
- *Attribute*: El atributo que identifica al usuario, por defecto, en los objetos user de los árboles LDAP es uid, pero puede cambiarse.
- Scope: Las consultas LDAP pueden ser sólo en la rama del árbol elegida o recursivamente en profundidad.
- *Filter*: filtro sobre los objetos. Por ejemplo que sean del tipo usuarios_reales o cualquier otro filtrado para determinar que un objeto es válido para logarse. (tipo=usuarios_reales).

Una vez autenticado a los usuarios por cualquiera de los proveedores anteriores se podrán establecer las reglas de control de acceso a los recursos. Así por ejemplo se podrá:

- ***Require valid-user***: Se exige un usuario autenticado.
- ***Require user maligno***: Se exige el usuario maligno.
- ***Require group locos***: Se exige el grupo locos.
- ***Require ldap-user***: Se exige un determinado usuario LDAP.
- ***Require ldap-group***: Se exige un determinado grupo LDAP.
- ***Require ldap-dn***: Se exige la pertenencia a una determinada ruta DN en LDAP.
- ***Require ldap-attribute tipo=admins***: Se exige un determinado valor en un determinado atributo de un usuario ldap.
- ***Requite ldap-filter***: Se exige el cumplimiento de un determinado filtro LDAP.

Mod_security

La securización y fortificación de un servidor Apache tiene muchos aspectos que tocar, pero después de todo lo que se ha tratado no me gustaría irme sin hacer una mención especial *mod_security*.

Este módulo nos permite realizar filtrado a nivel de aplicación, es decir, nos permite configurar reglas de firewall para las peticiones web que realizan los clientes. Es común que muchos de los ataques hoy en día no se produzcan contra el software del servidor y sí contra aplicaciones web mal desarrolladas o desactualizadas. Con mod_security vamos a poder configurar que peticiones van a ser procesadas o no por nuestras aplicaciones web.

Supongamos que tenemos una aplicación web vulnerable a un ataque de SQL Injection corriendo sobre nuestro servidor Apache. Con mod_security podremos evitar que cualquier intento de explotación llegue hasta la aplicación ya que será este componente el que la detenga antes de ser redirigida a la aplicación. Este tipo de comportamiento es deseable cuando estamos administrando un servicio sobre el que están corriendo aplicaciones web que no tenemos controladas. Ésta no es la mejor manera de proteger una aplicación web, pero sí es una forma de fortificarla aplicando el principio de Defensa en Profundidad ya que si la aplicación web queda comprometida puede quedar comprometida toda la infraestructura de los servicios web.
Al estar corriendo a nivel de aplicación y directamente sobre el servidor http el sistema no se ve limitado por los sistemas de cifrado. Hay que tener en cuenta que si disponemos de un NIDS (Network Intrusion Detection System) o sistema de detección de intrusiones de red, este no podrá analizar todas las peticiones que vayan cifradas sobre canales SSL, como por ejemplo http-s. En este caso, *mod_security* filtra las peticiones una vez se ha eliminado la capa SSL.

Es una buena decisión de diseño situar este componente en el frontal que redirige todas las peticiones hacia nuestros servicios, es decir, si tenemos un servidor que está realizando las opciones de Proxy reverso, es ahí donde hay que situar este componente.

El proceso de instalación se realiza mediante el uso del comando make. Para ello, tras descargar la última versión del módulo de http://www.modsecurity.org y descomprimir el paquete, hay que configurar en el archivo *Makefile* el valor la variable *top_dir* con la ruta dónde está instalado el software del servidor de Apache. Una vez configurado procedemos a realizar la ejecución de los comandos *make* y *make install* para dejar el módulo compilado y cargado en *httdp.conf*. Es necesario cargar la librería de objetos *libxml2.so* mediante la configuración de *LoadFile /usr/lib/libxml2.so* en el fichero *httpd.conf*.

Mod_security se configura por reglas, estas vienen en archivos independientes dentro del directorio *rules*. Todos los archivos tienen un numerito que se usa para indicar el orden de preferencia, ya que estas se van a ejecutar de menor a mayor.

Es decir, si una petición de URL coincide con dos reglas se disparará la que primero se encuentre, ya sea de permitir o de denegar el acceso de esa petición.

Imagen: Directorio rules de mod_security

Así *modsecurity_crs_10_config.conf* será el primer conjunto de reglas a evaluar. Estas son las reglas "*core*" y siempre deben ser las primeras que se evalúen, así que, si queremos crear un fichero de reglas que evite un falso negativo o deseamos crearnos nuestra propia regla para evitar ataques de LDAP injection a nuestras aplicaciones podríamos crearnos un fichero de reglas que se llamara *modsecurity_crs_15_ldapinjection.conf*, y sería el segundo conjunto de reglas en ejecutarse.

Una vez instalado *mod_security* debemos configurar las directivas del módulo. Estas directivas se pueden configurar en diferentes ámbitos, es decir, de forma global, a nivel de servidor, de virtual host o de directorio.

```
[IfModule mod_security.c]
SecRuleEngine On
SecAuditEngine RelevantOnly
SecAuditLog logs/audit_log
SecDefaultAction "deny,log,status:500"
[/IfModule]
```

El segmento anterior es una configuración típica. Las directivas configuradas tienen las siguientes implicaciones:

- ***SecRuleEngine on/off***: Activa o desactiva *mod_security*. Esta configuración se puede poner a nivel de servidor, virtual host o directorio.
- ***SecAuditLog***: Ruta del fichero de log
- ***SecFilterSancPost on/off***: Aplicar *mod_security* a peticiones por Post on/off.
- ***SecDefaultAction***: Orden de actuación en caso de no estar definida una acción para la regla que hay que aplicar. En este caso denegar la petición, registrarlo y devolver un código de status 500. Las posibilidades que se pueden aplicar a una acción por defecto son más de 30 y van desde ejecutar un programar, dirigir la navegación a una redirección, devolver un mensaje, tirar la petición etc…

A partir de la configuración incial podemos personalizar las reglas tanto como deseemos. *Mod_security* viene acompañado de un conjunto básico de reglas preparadas para detectar violaciones de protocolo, ataques comunes de web, ocultar información ofrecida por el servidor, detección contra troyanos y contra recolectores de información. No obstante, es importante aplicar nuevas reglas frente a las nuevas amenazas por lo que la actualización debe ser constante.

La parte fuerte de *mod_security* es su motor de reglas. Está basado en *perl* y se tiene una estructura de definición de reglas tan compleja o tan sencilla como se necesite. Cada regla se define con el siguiente comando:

SecRule Varialbes Operadores [Acciones]

Por ejemplo, la siguiente regla bloquea el acceso a cualquier cliente que se identifique como uno de los *User-Agents* de la lista:

SecRule REQUEST_HEADERS:User-Agent "@pm WebZIP WebCopier Webster WebStripper SiteSnagger ProWebWalker CheeseBot" "deny,status:403

La configuración de mod_securty como un WAF (*Web Application Filter*) ajustado requiere conocer el lengunage de definición de las reglas, pero la documentación que se ofrece en sitio del proyecto es de muy buena calidad.

Otras acciones

Fortificar un servidor Apache implica fortificar toda la infraestructura, existen muchas más opciones para aumentar la seguridad en el servidor web pero lo más importante, además de todo lo visto en estos dos artículos, es la fortificación del sistema operativo dónde va a correr el servidor Apache, para ello puedes usar SELinux o herramientas como **Bastille Linux** si corre Apache sobre Linux [http://www.bastille-linux.org/] o **Security Configuration Wizard (SCW)** si corre sobre Windows Server [http://www.microsoft.com/windowsserver2003/technologies/security/configwiz/default.mspx] que te pueden ayudar.