UD 2. ADMINISTRACIÓN DE SERVIDORES WEB

1. MÓDULOS DE APACHE

Apache está compuesto por **módulos,** algunos se instalan por defecto en el servidor y otros los activa y configura el usuario cuando los necesita. Los módulos se utilizan para agrupar funcionalidades de Apache muy concretas.

http://httpd.apache.org/docs/2.4/mod/

Como ya sabemos en /etc/apache2/ hay 2 carpetas: mods-avaiIable (módulos disponibles) y mods-enabled (módulos activados). Cuando arranca apache, cargará los módulos que están activados (los del mods-enabled).

Cuando se activa un módulo, se crea un enlace simbólico (una especie de acceso directo) de los archivos correspondientes del directorio mods-avaiIable a los del mods-enabled.

Cada módulo está compuesto por 2 archivos: el de configuración (nombre\_modulo.conf) y el de carga (nombre\_modulo.load)



Para habilitarlos también podemos hacerlo creando enlaces simbólicos

para ver módulos instalados

apachectl –M

**Módulo UserDir. Directorios personales**

http://httpd.apache.org/docs/2.4/mod/mod\_userdir.html#userdir

En determinadas circunstancias (por ejemplo una institución educativa como un instituto o una universidad) es útil que cada usuario tenga un directorio en el que pueda crear su propio conjunto de páginas web. Si el número de usuarios es grande, esto puede cargar innecesariamente de trabajo al administrador del servidor. Apache ofrece una alternativa automatizada para esta situación mediante el módulo *mod\_userdir*. El módulo userdir permite que cada usuario tenga un directorio personal en el servidor, que suele ser   
/home/usuario/public\_html

Usu1

Pag1.html

Pag2.html

Usu2

Pag1.html

Pag2.html

Si activamos el módulo (por defecto no viene activo) cada usuario tendrá un espacio al que se accederá mediante la URL http://sitioejemplo.com/~nombreusuario donde nombreusuario será el nombre que tiene en el servidor Linux

Activamos el módulo userdir**.**

|  |
| --- |
| **Comandos** |
| * sudo a2enmod userdir (activar módulo) * sudo a2dismod userdir (desactivar módulo) * /etc/init.d/apache2 graceful (o restart) |

* Vamos a /etc/apache2/mods-enabled y vemos que están userdir.conf y userdir.load. En el fichero userdir.conf está configurada la directiva UserDir que es la que permite que cada usuario del sistema tenga su propia carpeta personal public\_html

<IfModule mod\_userdir.c>

UserDir public\_html

UserDir disabled root

<Directory /home/\*/public\_html>

AllowOverride FileInfo AuthConfig Limit Indexes

Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec

<Limit GET POST OPTIONS>

Require all granted

</Limit>

<LimitExcept GET POST OPTIONS>

Require all denied

</LimitExcept>

</Directory>

</IfModule>

* Simplemente vamos al HOME de algún usuario (/home/teresa) y creamos la carpeta public\_html.

mkdir public\_html

Nota: Si quisiésemos hacer que cada usuario nuevo creado en la máquina tuviese en su home la carpeta public\_html por defecto, habría que crear el directorio public\_html en `/etc/skel'

* Para acceder vamos a la dirección ip del servidor (o el dominio) y agregamos /~USUARIO.

<http://www.misitio.com/~teresa/>

* Si quisiésemos cambiar la configuración por defecto de UserDir podríamos por ejemplo modificarlo a nivel de conf global o a nivel de virtual host. Por ejemplo si quisiésemos habilitar a algunos usuarios y a los demás deshabilitarlos…

UserDir disabled

UserDir enabled user1 user2 user3

Si quisiésemos deshabilitar unos pocos y los demás habilitarlos…

UserDir disabled user4 user5 user6

EJERCICIO: Crea un nuevo usuario “departamentos” en tu máquina (sudo adduser departamentos). Activa el módulo "mod\_userdir" y configura el servidor para que solamente se tenga acceso al directorio personal de “departamentos” (prueba que a otro directorio privado de otro usuario no se tiene permiso)

**Los módulos mod\_auth/mod\_authn. Sitios web no anónimos (Autenticación)**

Hay distintos módulos involucrados en el proceso de autenticación. Nos van a permitir realizar autorización de una manera bastante básica estableciendo acceso mediante usuario y contraseña a secciones de nuestro sitio.

Los módulos involucrados son los siguientes:

 Authentication type

* [mod\_auth\_basic](http://httpd.apache.org/docs/2.4/mod/mod_auth_basic.html)
* [mod\_auth\_digest](http://httpd.apache.org/docs/2.4/mod/mod_auth_digest.html)

 Authentication provider

* [mod\_authn\_anon](http://httpd.apache.org/docs/2.4/mod/mod_authn_anon.html)
* [mod\_authn\_dbd](http://httpd.apache.org/docs/2.4/mod/mod_authn_dbd.html)
* [mod\_authn\_dbm](http://httpd.apache.org/docs/2.4/mod/mod_authn_dbm.html)
* [mod\_authn\_file](http://httpd.apache.org/docs/2.4/mod/mod_authn_file.html)
* [mod\_authnz\_ldap](http://httpd.apache.org/docs/2.4/mod/mod_authnz_ldap.html)
* [mod\_authn\_socache](http://httpd.apache.org/docs/2.4/mod/mod_authn_socache.html)

Además de estos módulos, existe el [*mod\_authn\_core*](http://httpd.apache.org/docs/2.4/mod/mod_authz_core.html) que implementa las directivas del core para el resto de módulos.

**DIRECTIVAS:**

**AuthType:** Indica el tipo de autentificación, vamos a usar el tipo Basic que transmite la autenticación sin cifrar. La otra opción que existe es Digest que a diferencia de la opción Basic no transmite los nombres de usuario y contraseña como texto plano (y por lo tanto es una opción de seguridad mejor) pero que no está disponible para todos los navegadores Web. El cifrado que usa la opción Digest es MD5 que es bastante débil y aunque su uso no es idéntico al de la opción Basic es bastante similar, por lo que no lo veremos. Para conseguir la máxima seguridad se utiliza ssl sobre este tipo de autorización, como veremos más adelante.

AuthType Basic

**AuthName:** Esta directiva especifica el nombre que se muestra al solicitar autorización para acceder a un directorio. La cadena de caracteres que se especifique como valor de AuthName será lo que aparecerá en el cuadro de diálogo de acceso de la mayoría de los navegadores.

AuthName Escriba su nombre y contraseña

**AuthUserFile**: indica la ruta donde se encuentra el fichero que contiene la lista de usuarios y contraseñas. Estos usuarios y contraseñas deberán haber sido creados anteriormente mediante el uso del comando htpasswd.

AuthUserFile /var/secreto/.miembros

**AuthGroupFile:** lo mismo que el anterior pero con el nombre de los grupos de cuentas de usuario que usará apache para la autentificación. Sirve para organizar los usuarios en grupos y controlar su acceso de una manera más cómoda.

AuthGroupFile /var/secreto/.grupos

**Require user:** indica cuales de los usuarios de AuthUserFile tienen acceso al directorio especificado en la directiva Directory. El usuario debe estar contenido en el fichero indicado en la directiva AuthUserFile.

Require user teresa

Si queremos que sean todos pondremos

Require valid-user.

**Require group:** lo mismo pero para los grupos. El grupo debe estar contenido en el fichero indicado a través de la directiva AuthGroupFile.

Require group profesores

Ejemplo de configuración para solicitar autenticación en el acceso al directorio “privado”

<Directory "/var/www/privado">

AuthName “Acceso privado: Introduzca su usuario y contraseña”

AuthType Basic

AuthUserFile /var/secreto/.miembros

Require valid-user

</Directory>

**GETTING IT WORKING**

**Ejemplo autenticación mediante usuarios:**

1. En /var/www/ creamos directorio “privado” donde meteremos las páginas web protegidas.

mkdir /var/www/privado

Creamos el fichero de usuarios y contraseñas con el comando htpasswd cuya sintaxis es:

htpasswd ruta\_fichero\_AuthUserFile nombre\_usuario

(apt-get install apache2-utils para utilizar comando htpasswd)

htpasswd -c /etc/apache2/seguridad/.usuarios david

(el -c solo es necesario la primera vez para que cree el archivo, así no es necesario crear el archivo de antemano).

Vemos que se ha creado el archivo (oculto) y que está encriptada la contraseña.

Si queremos meter más usuarios

htpasswd /etc/apache2/seguridad/.usuarios ana

1. En el host virtual que queramos configurar en /etc/apache2/sites-available, añadimos lo siguiente:

<VirtualHost…>

….

<Directory /var/www/privado> AuthType Basic AuthName "Escribe usuario y contraseña para entrar" AuthUserFile /etc/apache2/seguridad/.usuarios Require user david </Directory>

</VirtualHost>

Hay que tener cuidado con este tipo de autentificación porque al poner un usuario válido guarda los datos en la caché del navegador y aunque salgamos del directorio protegido se puede volver a acceder a él sin necesidad de que pida la contraseña hasta que expire la caché. Para hacer pruebas es mejor usar el comando wget

wget --user=”usuario” --password=”pass” url

**Ejemplo autenticación mediante grupos:**

1. Activar el módulo correspondiente

a2enmod authz\_groupfile

1. Creamos en /etc/apache2/seguridad el archivo .grupos con el siguiente formato nombre\_grupo: lista\_usuarios\_separados\_por\_espacios

Ej:

informatica: david anaimagen: teresa carlos

1. Configurar host

<Directory /var/www/privado>

AuthType Basic

AuthName "Escribe usuario y contraseña para entrar"

AuthUserFile /etc/apache2/seguridad/.usuarios

AuthGroupFile /etc/apache2/seguridad/.grupos

Require group informatica

</Directory>

Al acceder, si ponemos un nombre de usuario y contraseña de un usuario perteneciente al grupo informática, permitirá el acceso.

**EJERCICIO: Crea un directorio privado dentro de /var/www/departamentos/privado. Crea dos directorios dentro que se llamen “informatica” e “imagen” para guardar los archivos privados de esos departamentos respectivamente. Modifica la configuración del host virtual por defecto “departamentos” para que sólo puedan acceder a cada uno de ambos directorios los usuarios pertenecientes al grupo correspondiente. ¿Puede un usuario estar en más de un grupo?**

**Módulo mod\_authz. Control de acceso a máquinas. Autorización**

Al igual que con la autenticación en la autorización se van a ver involucrados distintos módulos.

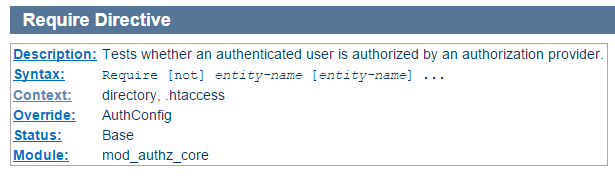
Authorization

* [mod\_authnz\_ldap](http://httpd.apache.org/docs/2.4/mod/mod_authnz_ldap.html)
* [mod\_authz\_dbd](http://httpd.apache.org/docs/2.4/mod/mod_authz_dbd.html)
* [mod\_authz\_dbm](http://httpd.apache.org/docs/2.4/mod/mod_authz_dbm.html)
* [mod\_authz\_groupfile](http://httpd.apache.org/docs/2.4/mod/mod_authz_groupfile.html)
* [mod\_authz\_host](http://httpd.apache.org/docs/2.4/mod/mod_authz_host.html)
* [mod\_authz\_owner](http://httpd.apache.org/docs/2.4/mod/mod_authz_owner.html)
* [mod\_authz\_user](http://httpd.apache.org/docs/2.4/mod/mod_authz_user.html)

Además de estos módulos, existe el [mod\_authz\_core](http://httpd.apache.org/docs/2.4/mod/mod_authz_core.html) que implementa las directivas del core para el resto de módulos.

¿Para qué vamos a usar estos módulos? Cuando un usuario hace una petición HTTP al servidor, este puede atenderla o no en función de la máquina desde la que se hace la solicitud. Esto nos puede servir para **limitar y controlar el acceso a determinadas máquinas** si no disponemos de un firewall (por ejemplo).

En Apache 2.2, el control de acceso se realizaba mediante las directivas *Order*, A*llow, Deny, and Satisfy*. En Apache 2.4, el control de acceso se realiza mediante el módulo mod\_authz\_core con la directiva *Require* como principal directiva para permitir o denegar acceso a los recursos. Gracias a esta directiva la forma en que la autorización se puede aplicar es ahora mucho más flexible que antes.Aunque el módulo mod\_access\_compat permite utilizar las directivas antiguas, esto no se recomienda ya que en una futura versión desaparecerán.



Como muestra la doc de apache, esta directiva se puede encontrar a nivel de contexto de directory o en el .htaccess.

**Control de acceso por host e ip([mod\_authz\_host](http://httpd.apache.org/docs/2.4/mod/mod_authz_host.html).)**

http://httpd.apache.org/docs/2.4/mod/mod\_authz\_host.html#requiredirectives

Directivas que se pueden utilizar con Require para este caso

* host: Se usa para comprobar el nombre de host de un cliente.
* ip: Se usa para comprobar la ip (completa o parcial) de un cliente.
* all: todo el tráfico. Se usa para poner valores por defecto.

Ejemplos

Require ip 10.1.2.3

Require ip 192.168.1.104 192.168.1.205

Require ip 10 172.20 192.168.2

Require ip 10.1.0.0/255.255.0.0

Require ip 10.1.0.0/16

Require host example.org

Require host .net example.edu

Require all granted

Require all denied

Por otro lado, se ha definido otras directivas, que permiten la composición de requisitos de acceso.

* RequireAll: Ningún requisito de autorización debe fallar y al menos uno debe ser éxito.
* RequireAny: Si se cumple uno de los requisitos de autorización en este bloque, este bloque está marcado como satisfecho.
* RequireNone: Ninguno de los requisitos debe ser éxito para que la directiva no falle.

OJO En los casos en los que las directivas Require no estén dentro de uno de los 3 bloques anteriores se considera que está dentro del bloque “RequireAny” es decir en el momento que la primera directiva Require se haga true, las demás directivas Require se ignorarán.

Para negar un elemento en particular se puede insertar “not”. Pero ojo porque las reglas con “not” nunca devolverán “true” (not false!= true). De esto se deduce que no se puedan meter dentro de los bloque “RequireAny” ni “RequireNone” ya que nunca tendrán efecto en ellos. Sólo se pueden meter dentro del bloque “RequireAll” pero teniendo en cuenta que no se puede usar por sí mismo para permitir o rechazar una petición, ya que “not true” no constituye false. Por lo tanto al usar un requerimiento con not en un bloque “RequireAll”, tenemos que tener un elemento adicional que evalúe a true o a false.

Ejemplo: Si denegamos la ip 10.252.46.165 de la siguiente forma:

Require not ip 10.252.46.165

Saldría el siguiente error “negative Require directive has no effect in <RequireAny> directive”. Por eso tendriamos que configurarlo:

<RequireAll>

Require all granted

Require not ip 10.252.46.165

</RequireAll>

En este ejemplo se aceptan todas las conexiones, excepto si vienen del usuario malicioso cuya ip es 10.252.46.165

En el siguiente ejemplo, extraído de la web de Apache, se permite el acceso a todo el mundo excepto si pertenecen a la red 192.168.205.0, o su host es "\*.phishers.example.com", o "\*.moreidiots.example" o "\*.ke":

<RequireAll>

Require all granted

<RequireNone>

Require ip 192.168.205

Require host phishers.example.com moreidiots.example

Require host ke

</RequireNone>

</RequireAll>

**EJERCICIO: Permitir a una ip**

**Configura un directorio /var/www/departamentos/autorizado en el sitio virtual departamentos el cual solo pueda ser accedido por tu máquina física y pruébalo.**

**EJERCICIO: Denegar a una ip**

**Configura un directorio /var/www/departamentos/autorizado en el sitio virtual departamentos el cual no pueda ser accedido por tu máquina física y pruébalo.**

**EJERCICIO: Denegar a todos**

**EJERCICIO: Permitir a todos**

**Módulo mod\_rewrite.**

El **módulo mod\_rewrite** es un módulo que permite crear direcciones URL alternativas a las dinámicas generadas por la programación de nuestros sitio web (blog, foro, portal…), de tal modo que sean más legibles y fáciles de recordar.

**EJERCICIO Investiga las siguientes directivas y prueba algún ejemplo****. RewriteBase,** **RewriteCond,** **RewriteEngine,** **RewriteRule, RewriteCond.**

**Módulo PageSpeed**

**EJERCICIO Instala este módulo para Apache, investiga para qué se puede usar y pruébalo.**

1. Archivos .HTACCESS

Cuando el administrador de sistemas no permite al programador tocar su configuración general del servidor, existe una solución para que el programador pueda meter mano a las configuraciones de sus aplicaciones y la solución es los ficheros .htaccess.

El fichero .htaccess permite definir diferentes directivas de configuración para un directorio (con sus respectivos subdirectorios) en concreto sin necesidad de editar el archivo de configuración principal de Apache.

Para permitir el uso de ficheros .htaccess en nuestro servidor o sitio virtual (la directiva se puede usar en ambos entornos) lo primero que debemos hacer es modificar la directiva

AllowOverride para que permita el contexto .htaccess.

<https://httpd.apache.org/docs/2.4/es/mod/core.html#allowoverride>

Por ejemplo

<Directory /var/www/privado> AllowOverride All </Directory>

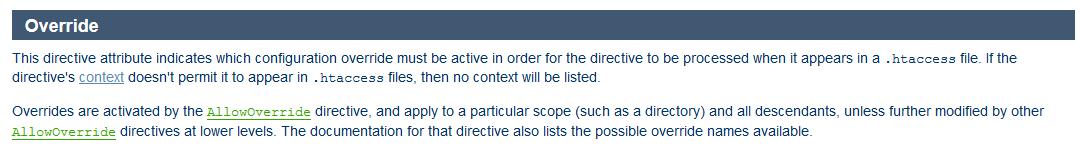
Es decir, debemos modificar el directorio que queremos que se pueda configurar mediante el fichero “.htaccess” sustituyendo la directiva.

AllowOverride None (No permite que se pueda usar fichero htaccess)

por

AllowOverride directive-type (Permite que se pueda configurar el fichero htaccess con el tipo de directivas “directive-type”)

OJO! Que directive-type será el campo Override de la directiva que queramos usar en el htaccess

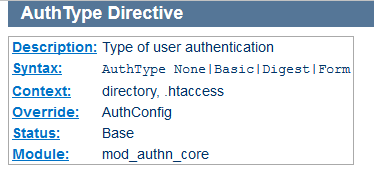
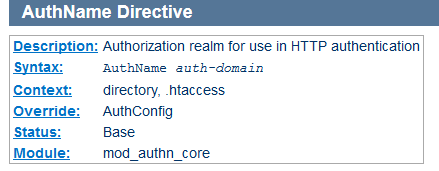


Por ejemplo si queremos permitir autenticación para una aplicación en concreto y el sysadmin no nos da acceso a los ficheros de configuración de apache, podemos hacer lo siguiente:

En el server, el sysadmin tiene que dejarnos usar htacces.

<Directory /var/www/html/miaplicacion> AllowOverride AuthConfig </Directory>

Si nos fijamos, todas las directivas que queremos configurar tienen el campo Override a AuthConfig de ahí que el tipo de directiva sea esa.



Accedemos al document root (/var/www/html/miaplicacion) de nuestra aplicación y creamos el fichero .htaccess (permisos 644) a continuación le damos el contenido que queramos similiar al siguiente:

AuthType Basic

AuthName "Escribe usuario y contraseña para entrar"

AuthUserFile /etc/apache2/seguridad/.usuarios

Require user david

AuthGroupFile /etc/apache2/seguridad/grupos

Require group informatica

Resumiendo, lo que hemos hecho ha sido sacar la configuración del control de acceso del archivo de configuración del servidor/sitio virtual a un archivo independiente. OJO: La creación o modificación de un fichero de este tipo no implica reiniciar el servidor.

El lugar en el que modificar la directiva depende de lo que necesitemos. Ten en cuenta que las directivas se heredan si no se encuentra otra más específica y /var/www está configurado por defecto con AllowOverride None

En muchos sitios indican que hay que permitir la sobreescritura de todas las directivas mediante AllowOverride All pero es evidente que es peor opción.

**EJERCICIO: Prueba a configurar el directorio “/var/www/departamentos/privado/informatica” con un fichero “.htaccess” en vez de por “virtualHost” con la misma configuración que el ejercicio del apartado *de autenticación.* Comprueba que se comporta de la misma forma.**