Wordpress

CMS para webs.

- Pasos para instalar un Wordpress seguro
- Fichero .htaccess en Wordpress

Pasos para instalar un Wordpress seguro

Todos soñamos con un WordPress que funcione como un reloj suizo: veloz, fiable y blindado contra las amenazas. Sin embargo, la realidad suele ser otra: sitios desactualizados, lentos y vulnerables a los piratas informáticos.

¿Te suena familiar? ¡No te preocupes! No es necesario caer en la paranoia ni convertirte en un experto en seguridad informática. La clave está en una instalación correcta y un mantenimiento periódico.

Olvídate de las complejidades:

- **Instalación**: Sigue una guía paso a paso para configurar tu WordPress de forma segura desde el principio.
- **Actualizaciones**: Mantén tu sitio actualizado con las últimas versiones de WordPress, plugins y temas. Es tan sencillo como activar las actualizaciones automáticas.
- **Mantenimiento**: Dedica un tiempo cada mes a realizar tareas básicas como revisar la seguridad, optimizar la velocidad y realizar copias de seguridad.

Con estos sencillos pasos, podrás disfrutar de un WordPress:

- **Seguro**: Protegido contra malware, hackers y otras amenazas.
- **Rápido**: Cargando a la veloci dad del rayo para ofrecer una experiencia óptima a tus visitantes.
- **Sin estrés**: Olvídate de las preocupaciones y disfruta de tu sitio web sin complicaciones.

¿Te animas a dar el paso?

En internet encontrarás una gran cantidad de recursos para ayudarte a conseguir un WordPress seguro, rápido y sin estrés. ¡Empieza hoy mismo a mejorar tu sitio web!

Recuerda: La seguridad y el rendimiento no son opciones, son necesidades para cualquier sitio web. Invierte un poco de tiempo y esfuerzo en cuidar tu WordPress y te lo agradecerá con creces.

Lo primero que tendrás que hacer será instalar el Wordpress, si quieres empezar bien puedes olvidarte del aburrido "wp_" para el prefijo de tu base de datos. ¡Es hora de darle rienda suelta a tu imaginación! ¿Qué tal "epsteuagi_vlodaor_" o una frase que solo tú entiendas? De esta manera ya no será obvio el nombre de tus tablas por si tuvieras algún incidente de inyección de SQL, aúnque a día de hoy esto debería de estar solventado no está de más tener este tipo de precauciones normalmente.

No os cuento como realizar una instalación básica de WordPress pues tenéis miles de tutoriales para hacer esto, símplemente recordad cambiar el prefijo de las tablas de la base de datos.

Una vez tengáis el Wordpress instalador es el momento de instalar el certificado SSL, que ya estamos en el siglo XXI, para ello tenéis un plugin maravilloso que es el **Really Simple SSL** y que os puede ayudar muchísimo para conseguir esta meta.

Y antes de meternos en harina con la seguridad de Wordpress vamos a repetir un mantra: "*Todos los ficheros con permisos 644 y todos los directorios con permisos 755*", al menos es lo que Wordpress recomienda, así que vamos a seguir sus indicaciones.

Una vez instalado podéis comprobar que sean como queremos las URLs, es decir, los enlaces permanentes, esto podemos hacerlo en "**Ajustes -> Enlaces Permanentes**". Y aprovechado que estáis ya por aquí podemos ir a "**Ajustes -> Generales**" y asegurarnos que la casilla "**Cualquiera puede registrarse**" no esté marcada.

Fichero .htaccess en Wordpress

Strict Transport

Ahora si os parece bien vamos a retocar el fichero .htaccess. Lo primero va a ser configurar el Strict Transport y unas cuantas políticas de seguridad básicas:

```
#STRICT TRANSPORT Y PERMISION POLICY
<IfModule mod_headers.c>
Header set Strict-Transport-Security "max-age=31536000; preload" env=HTTPS
Header always set Content-Security-Policy "upgrade-insecure-requests"
Header always set X-Content-Type-Options "nosniff"
Header always set X-XSS-Protection "1; mode=block"
Header always set Expect-CT "max-age=7776000, enforce"
Header always set Referrer-Policy: "no-referrer-when-downgrade"
Header always set Permissions-Policy "geolocation=(); midi=();notifications=();push=();sync-xhr=();accelerometer=();
       gyroscope=(); magnetometer=(); payment=(); camera=(); microphone=();usb=(); xr=();speaker=(self);vibrate=();
       fullscreen=(self);"
Header always append X-Frame-Options SAMEORIGIN
</IfModule>
```

Si me permitís os voy a explicar qué hace cada una de estas políticas para que no tengáis que hacer un acto de fe en vuestros Wordpress:

- **Strict-Transport-Security (HSTS)**: Strict-Transport-Security "max-age=31536000; preload" env=HTTPS asegura que los navegadores solo se comuniquen con el servidor a través de HTTPS, no HTTP, durante el tiempo especificado en max-age (en este caso, un año expresado en segundos). La opción preload indica que el sitio desea ser incluido en la lista de sitios precargados de HSTS que los navegadores mantienen. Esto se hace para evitar el primer acceso inseguro antes de que el navegador sepa que el sitio debe ser accedido solo a través de HTTPS.
- **Content-Security-Policy (CSP)**: Content-Security-Policy "upgrade-insecure-requests" instruye a los navegadores a tratar todos los recursos que se cargarían mediante HTTP como si se hubieran solicitado a través de HTTPS. Esto ayuda a mitigar ciertos ataques, como el "hombre en el medio".
- **X-Content-Type-Options**: X-Content-Type-Options "nosniff" prohíbe al navegador intentar deducir el MIME type de los recursos descargados, lo que obliga a usar el tipo proporcionado por el servidor. Esto reduce el riesgo de ataques basados en la confusión del tipo MIME.
- **X-XSS-Protection**: X-XSS-Protection "1; mode=block" estaba destinado a activar un filtro de protección contra XSS (Cross-Site Scripting) en navegadores que lo soportan,

bloqueando la carga de la página si se detecta un ataque. Sin embargo, esta cabecera está quedando obsoleta debido a las mejoras en los navegadores modernos.
- **Expect-CT**: Expect-CT "max-age=7776000, enforce" asegura que el navegador solo cargue el sitio si este presenta certificados válidos de Transparency Certificate (CT). Esto ayuda a prevenir ciertos tipos de ataques MITM (hombre en el medio) utilizando certificados SSL falsificados.
- **Referrer-Policy**: Referrer-Policy: "no-referrer-when-downgrade" controla la información que se incluye como referente cuando se navega de un sitio a otro. En este caso, impide enviar el referente si se pasa de HTTPS a HTTP, protegiendo la privacidad del usuario.
- **Permissions-Policy** (antes Feature-Policy): Esta política permite a los desarrolladores habilitar, deshabilitar o modificar el comportamiento de ciertas API y características web en el navegador. Por ejemplo, geolocation=(); deshabilita la geolocalización para todos los sitios excepto los especificados (en este caso, ninguno).
- **X-Frame-Options**: X-Frame-Options SAMEORIGIN previene ataques de clickjacking al asegurar que el contenido del sitio no pueda ser embutido en iframes de otros dominios, permitiendo solo los originados en el mismo dominio.

El texto para copiar es el siguiente:

```
#STRICT TRANSPORT Y PERMISION POLICY
<IfModule mod_headers.c>
Header set Strict-Transport-Security "max-age=31536000; preload" env=HTTPS
Header always set Content-Security-Policy "upgrade-insecure-requests"
Header always set X-Content-Type-Options "nosniff"
Header always set X-XSS-Protection "1; mode=block"
Header always set Expect-CT "max-age=7776000, enforce"
Header always set Referrer-Policy: "no-referrer-when-downgrade"
Header always set Permissions-Policy "geolocation=(); midi=();notifications=();push=();sync-
xhr=();accelerometer=(); gyroscope=(); magnetometer=(); payment=(); camera=();
microphone=();usb=(); xr=();speaker=(self);vibrate=();fullscreen=(self);"
Header always append X-Frame-Options SAMEORIGIN
</IfModule>
```

Forzar a www con https

El siguiente pasó estará más enfocado a forzar que nuestra página funcione siempre con https y con www:

```
#FORZAR A www con HTTPS
RewriteEngine On
RewriteCond %{HTTP_HOST} ^loseduardos.com [NC]
RewriteRule ^(.*)$ https://www.loseduardos.com/$1 [L,R=301]
```

El significado de estas líneas es el siguiente:

RewriteEngine On: Esta línea activa el motor de reescritura de URLs de Apache. Es necesario para poder utilizar las directivas de reescritura (RewriteRule, RewriteCond, etc.) en el archivo .htaccess.

- **RewriteCond %{HTTP_HOST} ^loseduardos.com [NC]:** Esta es una condición (RewriteCond) que se debe cumplir para que se ejecute la regla de reescritura (RewriteRule) que sigue. Esta línea específica verifica el host (el dominio) de la solicitud HTTP. %{HTTP_HOST} es una variable que contiene el dominio desde el que se realiza la solicitud. La expresión regular ^loseduardos.com coincide con cualquier solicitud hecha exactamente a loseduardos.com (sin www. al principio). [NC] significa "No Case", lo que indica que la comparación se debe hacer sin distinguir entre mayúsculas y minúsculas, haciendo que tanto loseduardos.com como LOSEDUARDOS.COM cumplan la condición.
- **RewriteEngine On**: Esta línea activa el motor de reescritura de URLs de Apache. Es necesario para poder utilizar las directivas de reescritura (RewriteRule, RewriteCond, etc.) en el archivo .htaccess.
- **RewriteCond %{HTTP_HOST} ^loseduardos.com [NC]**: Esta es una condición (RewriteCond) que se debe cumplir para que se ejecute la regla de reescritura (RewriteRule) que sigue. Esta línea específica verifica el host (el dominio) de la solicitud HTTP. %{HTTP_HOST} es una variable que contiene el dominio desde el que se realiza la solicitud. La expresión regular ^loseduardos.com coincide con cualquier solicitud hecha exactamente a loseduardos.com (sin www. al principio). [NC] significa "No Case", lo que indica que la comparación se debe hacer sin distinguir entre mayúsculas y minúsculas, haciendo que tanto loseduardos.com como LOSEDUARDOS.COM cumplan la condición.
- **RewriteRule ^(.)$ https://www.loseduardos.com/$1 [L,R=301]**: Esta es la regla de reescritura que se aplica si se cumple la condición anterior. La regla se descompone de la siguiente manera:
  - **^(.)$** es una expresión regular que coincide con cualquier solicitud. ^ indica el inicio de la solicitud, (.) captura cualquier carácter (y cualquier cantidad de ellos), y $ indica el final de la solicitud. Esto básicamente captura toda la URL después del dominio.
  - **https://www.loseduardos.com/$1** es la URL de destino a la que se redirige la solicitud. $1 se refiere a la primera captura realizada con (.) en la expresión regular, es decir, todo el path de la URL solicitada originalmente.
    - **[L,R=301]** son flags que modifican el comportamiento de la regla:

- **L** significa "Last", indicando que esta será la última regla que se aplicará si se cumple la condición. Es decir, no se evaluarán más reglas de reescritura después de esta.
- **R=301** significa "Redirect" con un código de estado HTTP 301, que es el código para "Moved Permanently". Esto informa al navegador (o a cualquier cliente HTTP) que la página ha sido movida permanentemente a la nueva URL especificada. Este tipo de redirección también ayuda a mantener la fuerza del SEO al pasar la relevancia de la URL original a la nueva.

Si queréis copiar el código, aquí os dejo:

```
RewriteEngine On
RewriteCond %{HTTP_HOST} ^loseduardos.com [NC]
RewriteRule ^(.*)$ https://www.loseduardos.com/$
```

Bloqueo de wp-config.php

Es importante que bajo ningún motivo el fichero wp-config.php sea accesible desde fuera del servidor. En ese fichero es donde tenemos la configuración de nuestro Wordpress y es crítico que esté protegido.

El significado de estas líneas es el siguiente:

- **<files wp-config.php>**: Esta directiva inicia una sección que se aplicará específicamente al archivo wp-config.php. La etiqueta <files> es utilizada para envolver las reglas que se quieren aplicar a uno o más archivos específicos. En este caso, se está apuntando únicamente al archivo wp-config.php.
- **order allow,deny**: Esta línea establece el orden en que se evalúan las reglas de permiso. Primero se evalúan las reglas de "allow" (permitir), seguido de las reglas de "deny" (denegar). En el contexto de este bloque, esta directiva prepara el escenario para que cualquier regla de acceso "permitir" se procese antes que las de "denegar", pero como solo habrá una regla de "denegar" y ninguna de "permitir", efectivamente se está

asegurando de que se deniegue el acceso a todos.

- **deny from all**: Esta línea niega el acceso al archivo **wp-config.php** desde cualquier dirección IP. Es decir, **nadie puede acceder directamente a este archivo a través del navegador o de cualquier solicitud HTTP externa**. Esta es una medida de seguridad crucial, ya que previene que actores maliciosos puedan leer o descargar el archivo que contiene información sensible.
- **</files>**: Esta directiva cierra la sección que especifica las reglas aplicables al archivo wp-config.php. Todo lo que esté entre las etiquetas <files wp-config.php> y </files> se aplica exclusivamente a ese archivo.

Para que podáis copiar el texto:

```
#BLOQUEO ACCESO AL wp-config.php
<files wp-config.php>
order allow,deny
deny from all
</files>
```

Proteger ficheros sueltos

En Wordpress, como en cualquier otro CMS es muy importante que sólo se sirvan los ficheros que tienen que servirse y no otros.

Es muy común en administradores poco experimentados que dejen copias en el servidor de backups. Estos backups pueden incluir ficheros sql con copias de la base de datos, ficheros comprimidos con los ficheros o cualquier otra combinación, así que es importante a ese tipo de ficheros no darle acceso desde fuera.

```
# SECURE LOOSE FILES
# http://m0n.co/04
<IfModule mod_alias.c>
RedirectMatch 403 (?i)(^#.*#|~)$
RedirectMatch 403 (?i)/readme\.(html|txt)
RedirectMatch 403 (?i)\.(ds_store|well-known)
RedirectMatch 403 (?i)/wp-config-sample\.php
RedirectMatch 403 (?i)\.(7z|bak|bz2|com|conf|dist|fla|git|inc|ini|log|old|psd|rar|tar|tgz|save|sh|sql|svn|swo|swp)$
</IfModule>
```

Esta configuración de un archivo .htaccess está diseñada para mejorar la seguridad de un sitio web al restringir el acceso a ciertos archivos que podrían ser sensibles o no destinados a ser accesibles públicamente. La configuración usa mod_alias para redireccionar ciertas solicitudes a un error 403, lo que significa "Prohibido" o "Acceso Denegado".

- **<IfModule mod_alias.c>**: Esta directiva verifica si el módulo mod_alias está presente y activo en el servidor Apache. mod_alias proporciona varias directivas para redireccionar y remapear URLs, y las reglas dentro de este bloque solo se aplican si dicho módulo está disponible. Esto es útil para evitar errores en caso de que el módulo no esté habilitado.
- **RedirectMatch 403 (?i)(^#.*#|~)$**: Esta regla utiliza RedirectMatch para enviar un estado HTTP 403 a cualquier solicitud que coincida con el patrón regular dado. El patrón **(?i)(^#.*#|~)$** busca archivos que empiecen y terminen con # o aquellos que terminen con ~, con (?i) haciendo la expresión insensible a mayúsculas o minúsculas. Estos archivos suelen ser copias de seguridad o temporales que algunos editores de texto dejan atrás.
- **RedirectMatch 403 (?i)/readme.(html|txt)**: Deniega el acceso a archivos readme.html o readme.txt, los cuales pueden revelar información sobre el software del sitio que podría ser útil para un atacante.
- **RedirectMatch 403 (?i).(ds_store|well-known)**: Bloquea el acceso a archivos .ds_store (usados por macOS para almacenar atributos personalizados de una carpeta) y a cualquier ruta que contenga /well-known, que suele usarse para la configuración de servicios web, pero puede ser sensible.
- **RedirectMatch 403 (?i)/wp-config-sample.php**: Específicamente prohíbe el acceso al archivo wp-config-sample.php de WordPress, que contiene el formato de configuración predeterminado y podría ser usado para adivinar configuraciones de seguridad.
- **RedirectMatch 403 (?i).(7z|bak|bz2|com|conf|dist|fla|git|inc|ini|log|old|psd|rar|tar|tgz|save|sh|sql|svn|swo|swp)$**: Esta regla bloquea una amplia gama de tipos de archivo que normalmente no deberían ser accesibles desde el web, incluyendo copias de seguridad, archivos de configuración, registros, y varios formatos de archivo comprimido. La presencia de (?i) al principio hace que la coincidencia sea insensible a mayúsculas y minúsculas.
- **</IfModule>**: Cierra el bloque que verifica la presencia de mod_alias.c, asegurando que las reglas solo se apliquen si este módulo está activo.

Por supuesto haz uso de tu imaginación y revisa tus necesidades para adaptar estas líneas a las necesidades que tu tengas.

Si necesitáis el texto os lo dejo aquí para copiar y pegar:

```
# SECURE LOOSE FILES
# http://m0n.co/04
<IfModule mod_alias.c>
RedirectMatch 403 (?i)(^#.*#|~)$
RedirectMatch 403 (?i)/readme\.(html|txt)
RedirectMatch 403 (?i)\.(ds_store|well-known)
RedirectMatch 403 (?i)/wp-config-sample\.php
RedirectMatch 403
(?i)\.(7z|bak|bz2|com|conf|dist|fla|git|inc|ini|log|old|psd|rar|tar|tgz|save|sh|sql|svn|swo|sw
p)$
```

```
</IfModule>
```

Bloqueo al wp-login.php y al xmlrpc.php

Hay dos ficheros en Wordpress que son el interfaz por el que se introducen los datos, el primero es el wp-login.php. Aquí es donde los autenticaremos para poder subir contenido al Wordpress. En mi caso concreto uso una VPN para conectarme a la administración del Wordpress y tampoco permito que se pueda acceder al xmlrpc desde fuera.

El fichero xmlrpc.php sirve para habilitar la interoperabilidad entre WordPress y otros sistemas mediante el protocolo XML-RPC. XML-RPC es un protocolo que permite que sistemas en diferentes entornos se comuniquen entre sí, enviando llamadas RPC (Remote Procedure Calls, o Llamadas a Procedimientos Remotos) codificadas en XML a través de HTTP.

Con XML-RPC se pueden realizar publicaciones remotas, gestionar contenidos, comentarios, en fin, se pueden hacer muchas cosas y desde el punto de vista de seguridad pues es una preocupación, aunque es cierto que en las versiones más modernas de Wordpress, por ejemplo la introducción de la API REST en WordPress 4.7 fue un gran avance.

No obstante nosotros vamos a bloquearlo todo.

```
# BLOQUEO A FICHEROS QUE NO TIENEN QUE SER ACCESIBLES
<Files xmlrpc.php>
order deny,allow
deny from all
allow from 2001:0DB8::12AB
allow from 2001:0DB8::/32
allow from 192.0.2.14
allow from 192.0.2.0/24
allow from 192.0.64.0/18
</Files>
<Files wp-login.php>
order deny,allow
deny from all
allow from 2001:0DB8::12AB
allow from 2001:0DB8::/32
allow from 192.0.2.14
allow from 192.0.2.0/24
</Files>
```

En este ejemplo os muestro como bloquear ambos ficheros, en el caso de xmlrpc.php fijaté que permito el acceso desde la 192.0.64.0/18.

```
edu@andromeda:~$ whois 192.0.64.0
NetRange:       192.0.64.0 - 192.0.127.255
CIDR:           192.0.64.0/18
NetName:        AUTOMATTIC
NetHandle:      NET-192-0-64-0-1
Parent:         NET192 (NET-192-0-0-0-0)
NetType:        Direct Allocation
OriginAS:       AS2635
Organization:   Automattic, Inc (AUTOM-93)
RegDate:        2012-11-20
Updated:        2021-12-14
Ref:            https://rdap.arin.net/registry/ip/192.0.64.0
```

Este rango es el de Automattic y se utiliza para plugins como por ejemplo el JetPack.

Esta configuración tiene la siguiente explicación:

- **<Files xmlrpc.php>**: Esta línea inicia una directiva <Files> para aplicar reglas específicamente al archivo xmlrpc.php. La directiva <Files> se utiliza para encerrar un grupo de directivas que se aplicarán solo al archivo nombrado, en este caso, xmlrpc.php.
- **order deny,allow**: Esta línea establece el orden en que se evaluarán las reglas de acceso. Primero se evalúan las reglas de deny (negar), seguidas por las reglas de allow (permitir). Esto significa que, por defecto, se denegará el acceso a menos que se cumpla una regla de permitir que coincida específicamente.
- **deny from all**: Esta directiva niega el acceso a todos los usuarios, sin importar su dirección IP. Es la configuración predeterminada antes de permitir accesos específicos.
- **allow from 192.0.64.0/18**: Esta línea permite el acceso al archivo únicamente desde direcciones IP dentro del rango 192.0.64.0 a 192.0.127.255 (lo que se indica con la notación /18). Esto significa que solo las solicitudes originadas desde esta gama de direcciones IP tendrán permiso para acceder al archivo xmlrpc.php.

Bloqueo del Spam no referido

Los malos son gente muy ingeniosa y tenemos que protegernos de muchas cosas, como por ejemplo que un comentario sólo pueda ser introducido si se viene desde el propio post, algo que parece muy obvio, pero que no lo es tanto, esto también lo puedes bloquear en el .htaccess:

```
# BLOQUEO SPAM NO REFERIDO
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteCond %{REQUEST_METHOD} POST
RewriteCond %{HTTP_USER_AGENT} ^$ [OR]
RewriteCond %{HTTP_REFERER} !^http(s)?://([^.]+\.)?loseduardos\.com [NC]
RewriteCond %{REQUEST_URI} /wp-comments-post\.php [NC]
RewriteRule .* - [F,L]
</IfModule>
```

Seguro que quieres saber qué hace cada línea, así que vamos allá:

- **<IfModule mod_rewrite.c>**: Esta línea verifica si el módulo mod_rewrite está disponible en el servidor Apache. mod_rewrite es un módulo muy potente que permite reescribir las URLs en tiempo real. Si el módulo no está habilitado, las instrucciones dentro de este bloque serán ignoradas.
- **RewriteEngine On**: Activa el motor de reescritura de URLs para Apache. Es necesario para que las siguientes reglas de reescritura funcionen.
- **RewriteCond %{REQUEST_METHOD} POST**: Esta condición se aplica solo a las solicitudes HTTP que utilizan el método POST. Los comentarios en WordPress se envían utilizando este método, por lo tanto, esta regla se asegura de que solo se filtren estas solicitudes.
- **RewriteCond %{HTTP_USER_AGENT} ^$ [OR]**: Esta condición verifica si el agente de usuario (User-Agent) de la solicitud HTTP está vacío. Un User-Agent vacío puede ser indicativo de un bot o una solicitud automatizada. La bandera [OR] significa que esta condición o la siguiente pueden ser verdaderas para que se aplique la regla.
- **RewriteCond %{HTTP_REFERER} !^http(s)?://([^.]+\.)?loseduardos\.com [NC]**: Verifica el referente (Referer) de la solicitud. Si la solicitud no proviene de loseduardos.com (o cualquier subdominio del mismo), se cumple la condición. ! al inicio invierte la condición (es decir, se cumple si la URL del referente NO coincide con el patrón). La expresión regular permite tanto http como https. [NC] hace que la comparación sea insensible a mayúsculas/minúsculas.
- **RewriteCond %{REQUEST_URI} /wp-comments-post\.php [NC]**: Esta es la regla de reescritura que se aplica si todas las condiciones anteriores son verdaderas. .* significa que la regla se aplica a cualquier solicitud que cumpla con las condiciones anteriores. - indica que la URL no se reescribirá. **[F,L]** son banderas: **F** manda un código de estado HTTP 403 Forbidden (Prohibido), efectivamente bloqueando la solicitud, y **L** indica que esta será la última regla que se aplicará si se cumple la condición.

Si quieres copiar el código, aquí te lo dejo:

```
# BLOQUEO SPAM NO REFERIDO
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteCond %{REQUEST_METHOD} POST
RewriteCond %{HTTP_USER_AGENT} ^$ [OR]
RewriteCond %{HTTP_REFERER} !^http(s)?://([^.]+\.)?loseduardos\.com [NC]
RewriteCond %{REQUEST_URI} /wp-comments-post\.php [NC]
RewriteRule .* - [F,L]
</IfModule>
```

Bloquear los crawlers

Los rawlers supuestamente deberíamos de poder bloquearlos con el robots.txt, pero muchas veces no hacen mucho casí, así que una de las formas que tenemos es bloquearlos utilizando el user-agent:

De esta manera bloqueamos todos los user-agents que queramos y al final pues podemos bloquear los crawlers por extensión.

Aquí dejo el código para copiar y pegar.

```
#BLOQUEAR CRAWLERS INDESEADOS
RewriteCond %{HTTP_USER_AGENT} (VelenPublicWebCrawler|Baiduspider|magpie-
crawler|CCBot|okhttp|GPTBot) [NC]
RewriteRule .* - [R=403,L]
```

Configurar la cache

También podemos ayudar con la caché utilizando el .htaccess, para ello podemos realizar la siguiente configuración:

```
# START Browser cache
<IfModule mod_expires.c>
ExpiresDefault                          "access plus 1 month"
ExpiresByType text/html                 "access plus 1 seconds"
ExpiresByType text/xml                  "access plus 1 seconds"
ExpiresByType text/plain                "access plus 1 seconds"
ExpiresByType application/xml           "access plus 1 seconds"
ExpiresByType application/json          "access plus 1 seconds"
ExpiresByType application/rss+xml       "access plus 1 hour"
ExpiresByType text/css                  "access plus 1 month"
ExpiresByType text/javascript           "access plus 1 month"
ExpiresByType application/font-woff     "access plus 1 month"
ExpiresByType application/font-woff2    "access plus 1 month"
ExpiresByType application/font-sfnt     "access plus 1 month"
ExpiresByType application/vnd.ms-fontobject "access plus 1 month"
ExpiresByType application/javascript    "access plus 1 month"
ExpiresByType application/x-javascript  "access plus 1 month"
ExpiresByType image/x-ico               "access plus 1 month"
ExpiresByType image/x-icon              "access plus 1 month"
ExpiresByType image/gif                 "access plus 1 month"
ExpiresByType image/png                 "access plus 1 month"
ExpiresByType image/jpe                 "access plus 1 month"
ExpiresByType image/jpg                 "access plus 1 month"
ExpiresByType image/jpeg                "access plus 1 month"
ExpiresByType font/truetype             "access plus 1 month"
ExpiresByType font/opentype             "access plus 1 month"
ExpiresByType font/ttf                  "access plus 1 month"
ExpiresByType font/woff                 "access plus 1 month"
ExpiresByType font/woff2                "access plus 1 month"
ExpiresByType application/x-font-woff    "access plus 1 month"
ExpiresByType video/ogg                 "access plus 1 month"
ExpiresByType audio/ogg                 "access plus 1 month"
ExpiresByType video/mp4                 "access plus 1 month"
ExpiresByType video/webm                "access plus 1 month"
ExpiresByType image/svg                 "access plus 1 month"
ExpiresByType image/svg+xml             "access plus 1 month"
ExpiresByType application/pdf           "access plus 1 month"
ExpiresByType application/vnd.ms-fontobject "access plus 1 month"
</IfModule>
# END Browser cache
```

Este código es sobretodo importante para mejorar el rendimiento de nuestro wordpress, vamos a ver línea a línea qué hace esto:

- **<IfModule mod_expires.c>**: Esta línea verifica si el módulo mod_expires está habilitado en el servidor Apache. Este módulo se utiliza para controlar la caducidad de los contenidos en la caché del navegador. El contenido dentro de este bloque <IfModule> solo se ejecuta si mod_expires está disponible.
- **ExpiresDefault "access plus 1 month"**: Esta directiva establece el tiempo de expiración por defecto para todos los tipos de contenido que no están explícitamente mencionados por una directiva ExpiresByType. En este caso, indica que el contenido debe ser considerado fresco y guardado en la caché del navegador durante 1 mes desde el momento de acceso. Esto significa que, a menos que se especifique de otra manera para tipos de contenido específicos, los recursos serán almacenados en la caché del navegador y no serán solicitados nuevamente al servidor hasta que pase un mes.
- **ExpiresByType text/html "access plus 1 seconds"**: Esta directiva especifica el tiempo de expiración para los documentos text/html, es decir, las páginas HTML. Aquí se establece que las páginas HTML deben ser almacenadas en la caché del navegador y consideradas frescas durante 1 segundo después de ser accedidas. Esto efectivamente significa que casi cada vez que el usuario solicite una página HTML, el navegador comprobará si hay una versión más reciente en el servidor, ya que el contenido solo se almacena en caché por un segundo.

El código para copiar y pegar es el siguiente:

```
# START Browser cache
<IfModule mod_expires.c>
ExpiresDefault                                "access plus 1 month"
ExpiresByType text/html                       "access plus 1 seconds"
ExpiresByType text/xml                        "access plus 1 seconds"
ExpiresByType text/plain                      "access plus 1 seconds"
ExpiresByType application/xml                 "access plus 1 seconds"
ExpiresByType application/json                "access plus 1 seconds"
ExpiresByType application/rss+xml             "access plus 1 hour"
ExpiresByType text/css                        "access plus 1 month"
ExpiresByType text/javascript                 "access plus 1 month"
ExpiresByType application/font-woff           "access plus 1 month"
ExpiresByType application/font-woff2          "access plus 1 month"
ExpiresByType application/font-sfnt           "access plus 1 month"
ExpiresByType application/vnd.ms-fontobject   "access plus 1 month"
ExpiresByType application/javascript          "access plus 1 month"
ExpiresByType application/x-javascript        "access plus 1 month"
ExpiresByType image/x-ico                     "access plus 1 month"
ExpiresByType image/x-icon                    "access plus 1 month"
```

```
ExpiresByType image/gif                    "access plus 1 month"
ExpiresByType image/png                    "access plus 1 month"
ExpiresByType image/jpe                    "access plus 1 month"
ExpiresByType image/jpg                    "access plus 1 month"
ExpiresByType image/jpeg                   "access plus 1 month"
ExpiresByType font/truetype                "access plus 1 month"
ExpiresByType font/opentype                "access plus 1 month"
ExpiresByType font/ttf                     "access plus 1 month"
ExpiresByType font/woff                    "access plus 1 month"
ExpiresByType font/woff2                   "access plus 1 month"
ExpiresByType application/x-font-woff      "access plus 1 month"
ExpiresByType video/ogg                    "access plus 1 month"
ExpiresByType audio/ogg                    "access plus 1 month"
ExpiresByType video/mp4                    "access plus 1 month"
ExpiresByType video/webm                   "access plus 1 month"
ExpiresByType image/svg                    "access plus 1 month"
ExpiresByType image/svg+xml                "access plus 1 month"
ExpiresByType application/pdf              "access plus 1 month"
ExpiresByType application/vnd.ms-fontobject "access plus 1 month"
</IfModule>
# END Browser cache
```

Configurar la compresión

Otro factor importante a tener en cuenta para mejorar el rendimiento de nuestro Wordpress es la compresión:

```
# BEGIN HttpHeadersCompression
<IfModule mod_deflate.c>
AddOutputFilterByType DEFLATE application/javascript
AddOutputFilterByType DEFLATE application/x-javascript
AddOutputFilterByType DEFLATE application/json
AddOutputFilterByType DEFLATE application/ld+json
AddOutputFilterByType DEFLATE application/manifest+json
AddOutputFilterByType DEFLATE application/rdf+xml
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/schema+json
AddOutputFilterByType DEFLATE application/vnd.geo+json
AddOutputFilterByType DEFLATE application/x-web-app-manifest+json
AddOutputFilterByType DEFLATE application/vnd.ms-fontobject
AddOutputFilterByType DEFLATE application/x-font-ttf
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE font/opentype
AddOutputFilterByType DEFLATE font/eot
AddOutputFilterByType DEFLATE image/bmp
AddOutputFilterByType DEFLATE image/svg+xml
AddOutputFilterByType DEFLATE image/x-icon
AddOutputFilterByType DEFLATE image/vnd.microsoft.icon
AddOutputFilterByType DEFLATE text/javascript
AddOutputFilterByType DEFLATE text/css
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/x-component
AddOutputFilterByType DEFLATE text/xml
</IfModule>
```

El contenido se comprimirá utilizando mod_deflate que permite al servidor comprimir los archivos antes de enviarlos al navegador del cliente, esto puede reducir significativamente el tiempo de carga de las páginas web al disminuir el tamaño de los datos transmitidos y por tanto ofrecer al cliente una sensación de mayor velocidad.,

La explicación del código es la siguiente:

- **<IfModule mod_deflate.c>**: Esta línea verifica si el módulo mod_deflate está disponible y habilitado en el servidor Apache. El módulo mod_deflate es el responsable de la compresión de los contenidos antes de que sean enviados al cliente. Utilizar <IfModule> asegura que las directivas dentro de este bloque solo se apliquen si mod_deflate está

activo, evitando errores en caso de que el módulo no esté presente o habilitado.
- **AddOutputFilterByType DEFLATE application/javascript**: Esta directiva le dice a Apache que aplique la compresión DEFLATE a los archivos de tipo application/javascript. DEFLATE es un algoritmo de compresión que reduce efectivamente el tamaño de los archivos de texto, como los archivos JavaScript, CSS, HTML, etc. AddOutputFilterByType especifica que la compresión debe aplicarse solamente a los contenidos que tengan el MIME type indicado, en este caso, application/javascript. Esto significa que todos los archivos JavaScript servidos por el servidor serán comprimidos automáticamente, reduciendo el tiempo que tardan en ser descargados por el navegador del usuario.

El código para copiar y pegar es el siguiente:

```
# BEGIN HttpHeadersCompression
<IfModule mod_deflate.c>
AddOutputFilterByType DEFLATE application/javascript
AddOutputFilterByType DEFLATE application/x-javascript
AddOutputFilterByType DEFLATE application/json
AddOutputFilterByType DEFLATE application/ld+json
AddOutputFilterByType DEFLATE application/manifest+json
AddOutputFilterByType DEFLATE application/rdf+xml
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/schema+json
AddOutputFilterByType DEFLATE application/vnd.geo+json
AddOutputFilterByType DEFLATE application/x-web-app-manifest+json
AddOutputFilterByType DEFLATE application/vnd.ms-fontobject
AddOutputFilterByType DEFLATE application/x-font-ttf
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE font/opentype
AddOutputFilterByType DEFLATE font/eot
AddOutputFilterByType DEFLATE image/bmp
AddOutputFilterByType DEFLATE image/svg+xml
AddOutputFilterByType DEFLATE image/x-icon
AddOutputFilterByType DEFLATE image/vnd.microsoft.icon
AddOutputFilterByType DEFLATE text/javascript
AddOutputFilterByType DEFLATE text/css
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/x-component
AddOutputFilterByType DEFLATE text/xml
</IfModule>
```

```
# END HttpHeadersCompression
```

Redirecciones

Las redirecciones las dejo aquí porque resulta sorprendente la cantidad de Wordpress que he visto con plugins para meter redirecciones cuando es algo tan sumamente sencillo, así que aquí dejo un par de ejemplos

```
RewriteRule ^blog/profesional.*$ https://www.loseduardos.com/profesional/ [R=301,L]
Redirect 302 /2020/12/19/hablando-de-redirtecciones/
https://www.loseduardos.com/2021/12/19/podcast-14-hablando-de-redirtecciones/
```