Administración de módulos en Apache

Antonio Espín Herranz

Contenidos

- Módulos más importantes de Apache.
- Introducción a mod_rewrite, mod_ssl, mod_security.
- Gestión de módulos dinámicos.
- Optimización de carga y rendimiento.
- Instalación y configuración de módulos

Módulos

- Los **módulos de Apache** son componentes que amplían las funcionalidades del servidor web.
- Permiten añadir soporte a cosas como autenticación, compresión, reescritura de URLs, SSL, estadísticas, y mucho más.
- Se **activan y configuran** en Apache, especialmente en Windows.
- Sólo activar los módulos que necesitemos → activar módulos innecesarios penaliza en el rendimiento.

Módulos más importantes de Apache

- Saber que módulos tenemos instalados:
- httpd –M

- Por funcionalidad tenemos:
  - Módulos de seguridad.
  - Módulos de rendimiento
  - Módulos de reescritura y redirección
  - Módulos de monitoreo y administración
  - Otros módulos últiles

Módulos de Seguridad

- **mod_security**: Firewall de aplicaciones web (WAF). Bloquea ataques como inyecciones SQL, XSS, etc.
- **mod_evasive**: Protege contra ataques DoS y escaneos agresivos.
- **mod_auth_basic**: Implementa autenticación básica con usuario y contraseña.
- **mod_ssl**: Habilita HTTPS y cifrado TLS.

Módulos de rendimiento

- **mod_deflate**: Comprime contenido con Gzip para reducir el tamaño de las respuestas.
- **mod_pagespeed**: Optimiza automáticamente HTML, CSS, JS e imágenes.
- **mod_cache**: Almacena contenido en caché para acelerar la entrega.
- **mod_expires**: Controla la expiración de recursos estáticos (ideal para navegadores).

Módulos de reescritura y redirección

- **mod_rewrite**: Reescribe URLs dinámicamente. Muy usado para SEO y redirecciones.

- **mod_alias**: Redirecciona rutas o mapea URLs a carpetas específicas.

- **mod_speling**: Corrige errores ortográficos en URLs (útil en sistemas sensibles a mayúsculas/minúsculas).

Módulos de monitoreo y administración

- **mod_status**: Muestra el estado del servidor en tiempo real (procesos, carga, etc.).
- **mod_info**: Expone la configuración activa del servidor (útil para auditorías).
- **mod_log_config**: Controla el formato y destino de los logs de acceso.

Otros módulos útiles

- **mod_userdir**: Permite acceder a carpetas personales vía http://localhost/~usuario.
- **mod_autoindex**: Genera listados de directorios si no hay index.html.
- **mod_cgi**: Ejecuta scripts CGI (como Perl o Python).

Módulos: mod_rewrite, mod_ssl, mod_security

- **mod_rewrite**
  - **Función:** Permite reescribir URLs de forma dinámica mediante reglas basadas en expresiones regulares.
  - **Usos comunes:** Crear URLs amigables, redirecciones 301/302, forzar HTTPS, control de acceso por IP o cabeceras.
  - **Ejemplo:** Convertir /producto/123 en producto.php?id=123.

- **mod_ssl**
  - **Función:** Habilita el soporte para conexiones HTTPS mediante el protocolo SSL/TLS.
  - **Usos comunes:** Cifrado de datos entre el cliente y el servidor, configuración de certificados digitales, cumplimiento de normativas de seguridad.
  - **Ejemplo:** Configurar un VirtualHost en el puerto 443 con un certificado SSL.

- **mod_security**
  - **Función:** Actúa como un firewall de aplicaciones web (WAF) para proteger contra ataques como inyecciones SQL, XSS, etc.
  - **Usos comunes:** Filtrado de peticiones maliciosas, reglas de seguridad personalizadas, cumplimiento de OWASP.
  - **Ejemplo:** Bloquear peticiones con patrones sospechosos en los parámetros de la URL.

Expresiones Regulares

- Dentro de los módulos de **Apache** vamos a tener que utilizar expresiones regulares, por ejemplo: en el módulo de **mod_rewrite**:

- En las **expresiones regulares** tendremos:
  - **Metacaracteres básicos**
  - **Cuantificadores**
  - **Caracteres especiales**

Metacaracteres básicos

Símbolo	Significado	Ejemplo		
`.`	Cualquier carácter excepto salto de línea	`a.c` → coincide con `abc`, `axc`		
`^`	Inicio de línea o cadena	`^hola` → debe comenzar con "hola"		
`$`	Fin de línea o cadena	`fin$` → debe terminar con "fin"		
`*`	0 o más repeticiones del carácter anterior	`a*` → `""`, `a`, `aa`, `aaa`		
`+`	1 o más repeticiones	`a+` → `a`, `aa`, `aaa`		
`?`	0 o 1 repetición (opcional)	`colou?r` → `color` o `colour`		
`` ` ``	`` ` ``	Alternativa (OR)	`` `rojo ``	azul` → coincide con uno u otro
`()`	Agrupa patrones y captura	`(abc)+` → `abc`, `abcabc`		
`[]`	Clase de caracteres	`[aeiou]` → cualquier vocal		
`[^]`	Negación de clase	`[^0-9]` → cualquier no dígito		

Cuantificadores

Símbolo	Significado	Ejemplo
`{n}`	Exactamente n repeticiones	`\d{3}` → exactamente 3 dígitos
`{n,}`	Al menos n repeticiones	`\d{2,}` → 2 o más dígitos
`{n,m}`	Entre n y m repeticiones	`\d{2,4}` → 2 a 4 dígitos

Caracteres especiales

Secuencia	Significado	Equivalente
`\d`	Dígito	`[0-9]`
`\D`	No dígito	`[^0-9]`
`\w`	Carácter alfanumérico o guion bajo	`[A-Za-z0-9_]`
`\W`	No alfanumérico	`[^A-Za-z0-9_]`
`\s`	Espacio en blanco	espacio, tab, salto de línea
`\S`	No espacio en blanco	cualquier carácter visible

mod_rewrite

mod_rewrite

- ***mod_rewrite es un módulo de Apache que permite reescribir URLs de forma dinámica usando reglas basadas en expresiones regulares.***

- Es extremadamente potente y se usa para:
  - Redireccionamientos SEO-friendly
  - Reestructuración de URLs
  - Control de acceso
  - Forzado de HTTPS
  - Proxy inverso y balanceo de carga

mod_rewrite

- Para utilizar el módulo: **mod_rewrite** hay que **activarlo** en el archivo de configuración.

- En **httpd.conf**
  - **LoadModule rewrite_module modules/mod_rewrite.so**

- En el archivo **.htaccess** o en la configuración de Apache:
  - **RewriteEngine On**

mod_rewrite

- Dispone de las directivas:

- **RewriteEngine** On:
  - Activa el motor de reescritura

- **RewriteCond**:
  - Condición que debe cumplirse para aplicar la regla

- **RewriteRule**:
  - Regla que define cómo se transforma la URL

- Estas tres forman el núcleo de mod_rewrite.

mod_rewrite

- **Sintaxis general:**

RewriteEngine On

 # (Opcional) Una o más condiciones
RewriteCond <condición 1>
RewriteCond <condición 2>
…

 # Regla que se aplica si TODAS las condiciones anteriores se cumplen
RewriteRule <patrón> <sustitución> [flags]

mod_rewrite

- No hay limite de RewriteCond antes de RewriteRule
- Se evalúan en orden y tienen que cumplirse todas para aplicar la regla.
- Se puede aplicar un flag [OR] y se aplica la condición con OR, la condición que contiene el flag y la siguiente.

- **Ejemplo**:
  - **Bloquea el acceso a /admin si la IP es 123.45.67.89 OR 111.22.33.44**
  - RewriteCond %{REMOTE_ADDR} ^123\.45\.67\.89$ [**OR**]
  - RewriteCond %{REMOTE_ADDR} ^111\.22\.33\.44$
  - RewriteRule ^admin - [F,L]

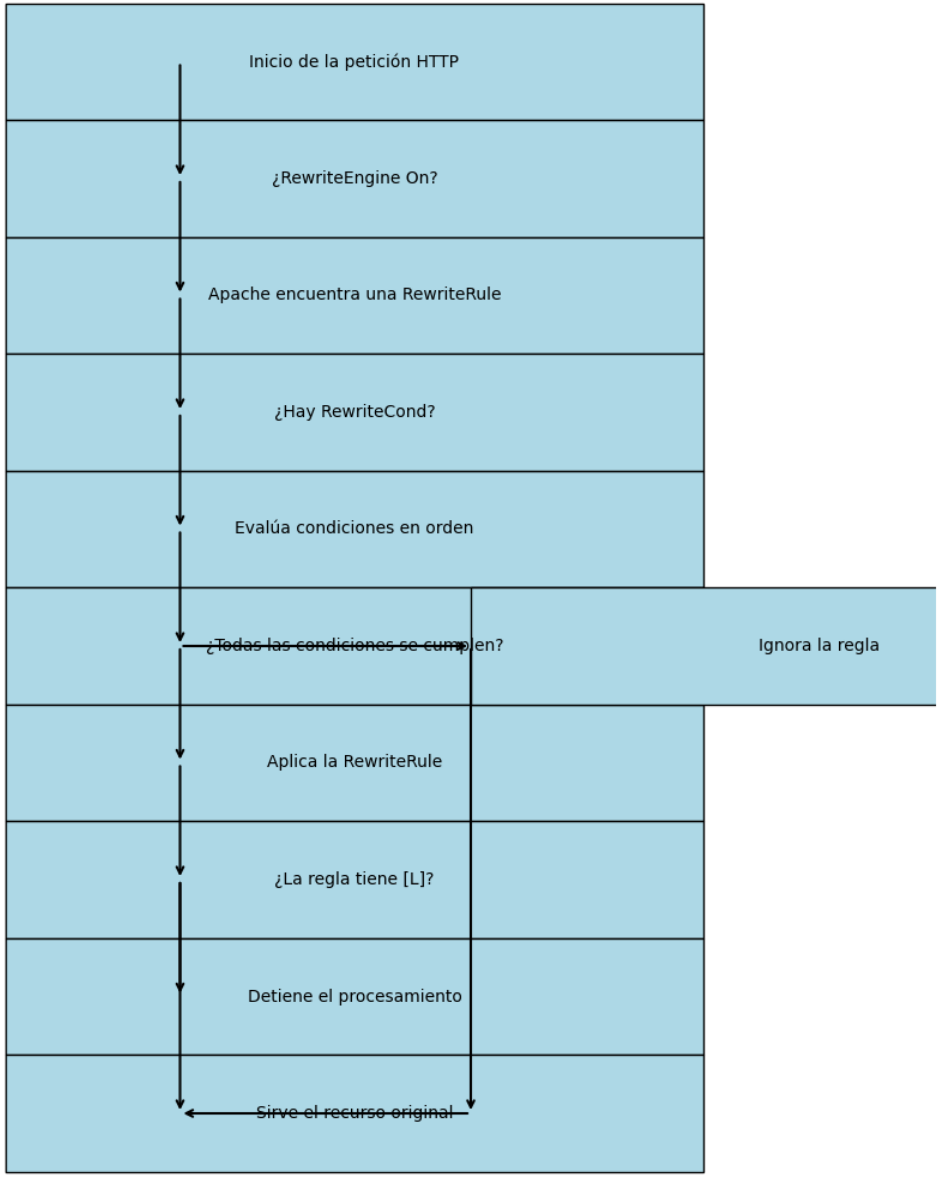
mod_rewrite

- **¿Qué pasa si no se cumplen las condiciones?**
  - Si **no se cumplen las condiciones**, **la** RewriteRule **no se ejecuta**.
  - En ese caso, Apache **sigue buscando otras reglas** (a menos que haya un **[L]** en una regla anterior que ya se aplicó).
  - Si **ninguna regla se aplica**, Apache simplemente **sirve el recurso original** como si no hubiera reglas.

mod_rewrite

- **Flujo de evaluación:**
  - Apache encuentra una RewriteRule.
  - Si hay condiciones (RewriteCond), las evalúa:
  - Si todas se cumplen, aplica la RewriteRule.
  - Si alguna falla, ignora esa regla y pasa a la siguiente.
  - Si no hay condiciones, la RewriteRule se evalúa directamente.
  - Si una regla se aplica y tiene **[L]**, se detiene el procesamiento.

  - **[L]** → Significa: Last Rule, detiene el procesamiento de reglas si esta se cumple.



Flujo de ejecución

mod_rewrite

- **Variables del Servidor**, se usan dentro de **RewriteCond**

Variable	Descripción
`%{REQUEST_URI}`	URI solicitada, incluyendo el path y la query string.
`%{REMOTE_ADDR}`	Dirección IP del cliente.
`%{HTTP_HOST}`	Host solicitado (dominio).
`%{QUERY_STRING}`	Cadena de consulta (lo que va después de `?`).
`%{REQUEST_METHOD}`	Método HTTP (`GET`, `POST`, etc.).
`%{SERVER_PORT}`	Puerto del servidor (`80`, `443`, etc.).
`%{HTTPS}`	Si la conexión es segura (`on` o vacío).
`%{TIME_YEAR}`	Año actual.
`%{ENV:VAR}`	Variable de entorno personalizada.
`%{HTTP:User-Agent}`	Cabecera del navegador del cliente.

mod_rewrite

- **Flags** que nos podemos encontrar en las reglas:

Flag	Significado
`[L]`	**Last rule**: detiene el procesamiento de reglas si esta se cumple.
`[F]`	**Forbidden**: devuelve un error 403 (acceso prohibido).
`[R=301]`	**Redirect permanente**: redirige con código HTTP 301 (útil para SEO).

mod_rewrite

- **Otros flags**:

Flag	Descripción
`[R]`	Redirección temporal (302 por defecto).
`[G]`	Gone: devuelve error 410 (contenido eliminado).
`[NC]`	No Case: ignora mayúsculas/minúsculas en la expresión regular.
`[QSA]`	Query String Append: conserva los parámetros de la URL original.
`[NE]`	No Escape: evita que caracteres especiales se escapen automáticamente.
`[PT]`	Pass Through: pasa la URL reescrita al siguiente handler (útil con mod_alias).
`[NS]`	No Subrequest: solo aplica en peticiones principales, no en subpeticiones.
`[OR]`	Or: permite combinar condiciones con lógica OR.

mod_rewrite

- **Grupo captura: $1, $2, etc.**
  - Hacen referencia al **grupo captura** dentro de la **expresión regular**
  - Es la parte de la expresión regular que va dentro de los **paréntesis**
  - Se toman de izquierda a derecha: **$1**, **$2**, etc.
  - A parte del $ los podemos ver con **%**

  - Los vamos a ver en **RewriteRule** y también en **RewriteCond**
  - RewriteRule ^producto/**([0-9]+)**$ producto.php?id=**$1** [L]

mod_rewrite

- **Redirección simple:**
  - RewriteEngine On
  - RewriteRule ^oldpage\.html$ newpage.html [R=301,L]

  - Expr. Reg:
    - Si empieza por oldpage
    - \. Escapar el “.”, no utilizar como comodín.
    - html$ termina en extensión html
    - 301 → código http: movido permanentemente.

mod_rewrite

- Forzar **HTTPS**
  - Redirigir todo el tráfico de http a https

  - RewriteCond %{HTTPS} off
  - RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

mod_rewrite

- **URLs limpias:**
  - Sobretodo se utiliza en servicios REST, en vez de utilizar la clásica queryString se utilizan URLs bonitas.
  - Una petición entrante del estilo: producto/123 se traduce internamente a producto/?id=123
  - RewriteRule ^producto/([0-9]+)$ producto.php?id=$1 [L]

mod_rewrite

- **Bloquear direcciones IP específicas:**
  - RewriteCond %{REMOTE_ADDR} ^123.\456\.789\.000$
  - RewriteRule ^.*$ - [F]

- https://www.elarraydejota.com/ejemplos-de-redirecciones-utiles-y-comunes-con-mod_rewrite-para-apache/

Ejemplo

- RewriteCond %{REQUEST_URI} ^/admin
- RewriteCond %{REMOTE_ADDR} !^192\.168\.1\.
- RewriteRule ^.*$ - [F,L]

- Bloquea el acceso a /admin si la IP no es local.
- Devuelve error 403.
- Detiene el procesamiento de reglas.

mod_rewrite

- Con este módulo también se incluyen:
  - RewriteMap: para redirecciones basadas en archivos externos
  - Redirecciones dependientes del tiempo
  - Balanceo de carga por URL
  - Regeneración de contenido al vuelo
  - Reescritura condicional por encabezados HTTP o variables de entorno

.htaccess vs httpd.conf

- Existen **diferencias** entre colocar la configuración de **mod_rewrite** en el archivo .**htaccess** (por cada sitio) y **httpd.conf** (global).

- **Ventajas en httpd.conf:**
  - **Mejor rendimiento**: Apache no tiene que buscar y leer **.htaccess** en cada carpeta.
  - **Mayor seguridad**: Evita que usuarios sin privilegios modifiquen reglas críticas.
  - **Configuración más limpia y centralizada**.

- **Tener en cuenta:**
  - Requiere **acceso al sistema** (**root** o **sudo**).
  - Cambios requieren **reiniciar Apache** (sudo systemctl restart apache2).
  - **No es práctico en entornos de hosting compartido**.

Ejemplo

- Dentro **de httpd.conf**

<**Directory** "/var/www/html/misitio">
  AllowOverride None
  RewriteEngine On
  RewriteRule ^producto/([0-9]+)$ producto.php?id=$1 [L]
</Directory>

- Para Linux: /var/www/html/misitio
- Para Windows: C:/Apache24/htdocs/misitio

Ubicación

- Dentro del fichero: **httpd.conf**
- Las reglas se encontrarán dentro de Directory o VirtualHost, pero si lo ponemos fuera (el contexto global), puede afectar a todo el servidor.

Comparativa

Característica	`.htaccess`	`httpd.conf` (o `apache2.conf`)
📍 Ubicación	Dentro del directorio del sitio web	Archivo global de configuración del servidor
🔒 Acceso requerido	Usuario con acceso al sitio (no root)	Requiere permisos de administrador (root/sudo)
⚡ Rendimiento	Menor: Apache revisa `.htaccess` en cada petición	Mayor: se carga una vez al iniciar Apache
🔄 Flexibilidad	Alta: permite cambios sin reiniciar Apache	Baja: requiere reinicio para aplicar cambios
🔧 Activación de módulos	No se puede activar módulos	Se activan con `LoadModule`
🧩 Uso de `mod_rewrite`	Requiere `AllowOverride FileInfo` o `All`	Se configura directamente sin restricciones
🛡️ Seguridad	Menor: usuarios pueden modificar reglas sensibles	Mayor: configuración centralizada y controlada
📦 Estructura	Solo admite directivas permitidas por `AllowOverride`	Permite bloques `<Directory>`, `<VirtualHost>`, etc.
🔁 Aplicación por carpeta	Sí, afecta solo al directorio donde está ubicado	Sí, pero requiere definir rutas absolutas
🧠 Uso recomendado	Hosting compartido, sitios con acceso limitado	Servidores dedicados, producción, entornos controlados

mod_ssl

mod_ssl

- **mod_ssl** es el módulo de Apache que habilita el soporte para **SSL/TLS**, permitiendo conexiones seguras (**HTTPS**).

- Se basa en **OpenSSL** y permite cifrar el tráfico entre el cliente y el servidor.

- Activar el módulo en httpd.conf:
  - **LoadModule ssl_module modules/mod_ssl.so**

- Y el archivo de configuración SSL:
  - **Include conf/extra/httpd-ssl.conf**

mod_ssl

- Crear los certificados:
  - Podemos usar certificados autofirmados para pruebas o certificados reales para producción.
  - **Certificado público:** C:/Apache24/conf/ssl/ejemplo.crt
  - **Clave privada:** C:/Apache24/conf/ssl/ejemplo.key

- Utilizar el programa OpenSSL, disponible en la carpeta bin de Apache.

  openssl req -x509 -nodes -days 365 -newkey rsa:2048 ^
  -keyout C:\Apache24\conf**ssl**\ejemplo.**key** ^
  -out C:\Apache24\conf**ssl**\ejemplo.**crt**

- **^** se utiliza para indicar que el comando continua (en Windows).

mod_ssl: open_ssl

Fragmento	Significado
`openssl`	Llama al programa OpenSSL
`req`	Inicia el proceso de creación de un certificado X.509
`-x509`	Indica que se generará un certificado autofirmado (no se crea CSR)
`-nodes`	No cifra la clave privada con contraseña (útil para Apache)
`-days 365`	El certificado será válido por 365 días
`-newkey rsa:2048`	Crea una nueva clave RSA de 2048 bits junto con el certificado
`^`	En Windows, indica que el comando continúa en la siguiente línea
`-keyout C:\Apache24\conf\ssl\ejemplo.key`	Ruta donde se guardará la clave privada
`-out C:\Apache24\conf\ssl\ejemplo.crt`	Ruta donde se guardará el certificado público

mod_ssl

- **Configuración en httpd-ssl.conf en Windows**

```
<VirtualHost _default_:443>
  ServerName localhost
  DocumentRoot "C:/Apache24/htdocs"

  SSLEngine on
  SSLCertificateFile "C:/Apache24/conf/ssl/ejemplo.crt"
  SSLCertificateKeyFile "C:/Apache24/conf/ssl/ejemplo.key“
  # SSLCertificateChainFile "C:/Apache24/conf/ssl/chain.crt"  # Descomenta si usas certificados intermedios

  <Directory "C:/Apache24/htdocs">
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
  </Directory>

</VirtualHost>
```

- Reiniciar Apache: **httpd –k restart**

Options Indexes FollowSymLinks

Indexes: permite mostrar listado de archivos si no hay index.html

FollowSymLinks: permite seguir enlaces simbólicos

AllowOverride All

Permite que .**htaccess sobrescriba** cualquier configuración

Require all granted

Permite acceso a **todos** los **usuarios** (sin restricciones)

mod_ssl

- Puedes duplicar o adaptar este archivo si quieres configurar **múltiples sitios HTTPS** con diferentes dominios o certificados.

- Solo asegúrate de usar **VirtualHost *:443** y cambiar el **ServerName** y **DocumentRoot** según cada caso.

mod_ssl

- **Directivas**

Directiva	Función
`SSLEngine`	Activa o desactiva SSL (`on` o `off`)
`SSLCertificateFile`	Ruta al certificado público (`.crt`)
`SSLCertificateKeyFile`	Ruta a la clave privada (`.key`)
`SSLCertificateChainFile`	Certificados intermedios (si aplica)
`SSLProtocol`	Protocolos permitidos (`all -SSLv3 -TLSv1`)
`SSLCipherSuite`	Cifrado permitido (puede restringirse por seguridad)
`SSLHonorCipherOrder`	Prioriza el orden del servidor en el handshake

mod_ssl

- Más directivas

`SSLHonorCipherOrder`	Prioriza el orden de cifrado del servidor en el handshake	`SSLHonorCipherOrder on`
`SSLVerifyClient`	Solicita verificación del cliente (para certificados cliente)	`SSLVerifyClient optional`
`SSLVerifyDepth`	Profundidad máxima de verificación en la cadena de certificados	`SSLVerifyDepth 2`
`SSLOptions`	Opciones adicionales como exportación de claves o compatibilidad	`SSLOptions +StdEnvVars`
`SSLPassPhraseDialog`	Método para ingresar la frase de contraseña de la clave privada	`SSLPassPhraseDialog builtin`
`SSLLog` / `SSLLogLevel`	(Obsoleto) Usado en versiones antiguas para registrar eventos SSL	—

mod_ssl

- **Configuración en httpd-ssl.conf en linux**

```
<VirtualHost *:443>
  ServerName www.ejemplo.com
  DocumentRoot "/var/www/html"

  SSLEngine on
  SSLCertificateFile /etc/ssl/certs/ejemplo.crt
  SSLCertificateKeyFile /etc/ssl/private/ejemplo.key
  SSLCertificateChainFile /etc/ssl/certs/chain.crt

  SSLProtocol all -SSLv3 -TLSv1
  SSLCipherSuite HIGH:!aNULL:!MD5
  SSLHonorCipherOrder on
</VirtualHost>
```

Prueba con dos dominios diferentes
Uno por http y otro https

mod_ssl

- Documentación: https://httpd.apache.org/docs/trunk/es/mod/mod_ssl.html

- El módulo expone variables de entorno que se pueden utilizar en los logs.
  - **CustomLog logs/ssl_request_log "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"**
    - Fecha y hora
    - IP del cliente
    - Protocolo SSL (ej. TLSv1.2)
    - Cifrado usado (ej. ECDHE-RSA-AES256-GCM-SHA384)
    - Petición HTTP
    - Tamaño de respuesta

    - Tiene que estar dentro de un bloque HTTPS si no, las variables no las va a reconocer.

Ejemplo (***prueba, comprobar el log***)

```
<VirtualHost *:443>
   ServerName seguro.midominio.local
   DocumentRoot "C:/Apache24/htdocs/seguro"

   SSLEngine on
   SSLCertificateFile "C:/Apache24/conf/ssl/server.crt"
   SSLCertificateKeyFile "C:/Apache24/conf/ssl/server.key"

   CustomLog "logs/ssl_request_log" "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

   <Directory "C:/Apache24/htdocs/seguro">
      Require all granted
   </Directory>
</VirtualHost>
```

Variables SSL para CustomLog

- Para utilizarlas ${NOMBRE_VAR]**x**
  - **La x tiene que estar en la cadena formato.**

Variable	Uso en `CustomLog`	Descripción
`SSL_PROTOCOL`	`%{SSL_PROTOCOL}x`	Protocolo negociado (ej. TLSv1.2)
`SSL_CIPHER`	`%{SSL_CIPHER}x`	Cifrado negociado (ej. AES256-GCM-SHA384)
`SSL_CLIENT_S_DN`	`%{SSL_CLIENT_S_DN}x`	DN completo del certificado del cliente
`SSL_CLIENT_S_DN_CN`	`%{SSL_CLIENT_S_DN_CN}x`	Nombre común del cliente
`SSL_CLIENT_VERIFY`	`%{SSL_CLIENT_VERIFY}x`	Resultado de la verificación del certificado
`SSL_SERVER_S_DN_CN`	`%{SSL_SERVER_S_DN_CN}x`	Nombre común del servidor

Variables NO disponibles en CustomLog

- SSL_SESSION_ID
- SSL_COMPRESS_METHOD
- SSL_CIPHER_USEKEYSIZE

Otro Ejemplo de CustomLog

- CustomLog logs/ssl_access.log "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x %{SSL_CLIENT_S_DN_CN}x \"%r\" %b“

- El **CN**: Common Name (cuando estamos creando el certificado)

  - El **Common Name (CN)** es parte del **Distinguished Name (DN)** del certificado X.509. Es un campo que normalmente identifica al propietario del certificado.

  - Puede ser:
    - El **nombre completo** de una persona (ej. Juan Pérez)
    - El **nombre de una empresa** (ej. Acme Corp)
    - Un **nombre de usuario** o **ID** (ej. usuario123)
    - Un **dominio** (en certificados de servidor)

Activar autenticación por certificado de cliente

- Crear una CA local y firmar certificados
- **1.Crear la CA (Autoridad Certificadora)**
  - A. Generar clave privada:
    - **openssl genrsa -out ca.key 4096**

  - B. Crear el certificado autofirmado:
    - **openssl req -x509 -new -nodes -key ca.key -sha256 -days 3650 -out ca.crt**

- **2.Crear certificado de servidor firmado por la CA**
  - A. Generar clave privada del servidor
    - **openssl genrsa -out server.key 2048**

  - B. Crear CSR (solicitud de firma)
    - openssl req -new -key **server.key** -out **server.csr**

  - C. Firmar el certificado con la CA
  - openssl x509 -req -in **server.csr** -CA ca.crt -CAkey **ca.key** -CAcreateserial -out **server.crt** -days 365 -sha256

Activar autenticación por certificado de cliente

- **3.Crear certificado de cliente firmado por la CA**
  - **A. Generar clave privada del cliente**
    - openssl genrsa -out **client.key** 2048
  - **B. Crear CSR del cliente (CSR: Solicitud de firma de certificado)**
    - openssl req -new -key **client.key** -out **client.csr**
  - **C. Firmar el certificado del cliente con la CA**
    - openssl x509 -req -in client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out **client.crt** -days 365 -sha256

Activar autenticación por certificado de cliente **(en el Servidor)**

- Bloque <VirtualHost *:443>

- SSLEngine on
- SSLCertificateFile "C:/Apache24/conf/ssl/**server.crt**"
- SSLCertificateKeyFile "C:/Apache24/conf/ssl/**server.key**"
- SSLCACertificateFile "C:/Apache24/conf/ssl/**ca.crt**"

Activar autenticación por certificado de cliente **(en el Cliente)**

- **Para autenticar clientes:**
- SSLVerifyClient require
- SSLCACertificateFile "C:/Apache24/conf/ssl/**ca.crt**"

Bloque virtualhost 443 (final)

```
<VirtualHost *:443>
    ServerName seguro.midominio.local
    DocumentRoot "C:/Apache24/htdocs/seguro"

    # Activar SSL
    SSLEngine on

    # Certificado del servidor firmado por tu CA local
    SSLCertificateFile "C:/Apache24/conf/ssl/server.crt"
    SSLCertificateKeyFile "C:/Apache24/conf/ssl/server.key"

    # Certificado de la CA que firmó los certificados de cliente
    SSLCACertificateFile "C:/Apache24/conf/ssl/ca.crt"

    # Autenticación de cliente
    SSLVerifyClient require
    SSLVerifyDepth 2

    # Log personalizado con CN del cliente
    CustomLog "logs/ssl_access.log" "%t %h %{SSL_CLIENT_S_DN_CN}x \"%r\" %>s %b"

    # Opcional: restringir acceso por CN específico
    <Location />
        SSLRequire %{SSL_CLIENT_S_DN_CN} eq "Juan Pérez"
    </Location>

    # Permisos de directorio
    <Directory "C:/Apache24/htdocs/seguro">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

OJO el certificado tiene que ser generado para Juan Perez en el CN

Prueba

- **Instala client.crt en el navegador del cliente.**
- Necesitamos:
  - El archivo del certificado de cliente: **client.crt**
  - La clave privada: **client.key** (solo si el navegador lo solicita)
  - Lo ideal es tener ambos combinados en un archivo .p12 o .pfx (PKCS#12), que incluye certificado + clave

  - ***openssl pkcs12 -export -out client.p12 -inkey client.key -in client.crt -name "Juan Pérez"***

- ***Instalar en el Navegador…***

- Accede al sitio HTTPS.
- Apache pedirá el certificado, lo validará contra ca.crt, y si es válido, permitirá el acceso.

Instalar en el Navegador

- **Firefox**
  - Ve a **Configuración** → **Privacidad y seguridad** → **Certificados** → **Ver certificados**
  - Pestaña **Sus certificados** → **Importar**
  - Selecciona client.p12 y escribe la contraseña
  - El certificado estará disponible para autenticación SSL

- **Chrome / Edge (Windows)**
  - Chrome y Edge usan el almacén de certificados de Windows:
  - Abre el archivo client.p12 (doble clic)
  - Se abrirá el asistente de importación de certificados
  - Elige **Almacén personal** o **Usuarios actuales**
  - Introduce la contraseña
  - Finaliza la importación

mod_security

mod_security

- **mod_security** es un módulo que actúa como un **WAF (Web Application Firewall)**. Analiza las peticiones HTTP y bloquea ataques comunes como:
  - Inyección SQL
  - Cross-site scripting (XSS)
  - Inclusión de archivos (LFI/RFI)
  - Escaneos automatizados

mod_security

- **Directivas:**

Directiva	Función
`SecRuleEngine`	Activa el motor de reglas (`On` , `DetectionOnly` , `Off`)
`SecRequestBodyAccess`	Permite inspeccionar el cuerpo de la petición
`SecRule`	Define una regla personalizada
`Include`	Carga reglas externas (como OWASP CRS)
`SecAuditLog`	Ruta del log de auditoría
`SecDefaultAction`	Acción por defecto para las reglas (`deny` , `pass` , `log` , etc.)

Gestión de módulos dinámicos

- **Módulos dinámicos vs estáticos**
  - **Estáticos**: compilados directamente en Apache (no necesitan **LoadModule**).
  - **Dinámicos**: se cargan con **LoadModule** desde la carpeta modules/.
  - En Windows, la mayoría son dinámicos y están en C:\Apache24\modules\

- Los módulos se activan / desactivan descomentando las líneas, borrando la #
  - LoadModule rewrite_module modules/mod_rewrite.so
  - #LoadModule rewrite_module modules/mod_rewrite.so

- Una vez activado, podemos usar sus directivas en **httpd.conf**, **.htaccess** o dentro de bloques <**Directory**> o <**VirtualHost**>.

Carga y rendimiento

- **1. Cada módulo consume recursos**
  - Al activarse, un módulo se carga en memoria y puede ejecutar procesos en cada solicitud.
  - Módulos como mod_security, mod_rewrite o mod_deflate analizan o transforman el tráfico, lo que puede aumentar la carga si no están optimizados.

- **2. Módulos críticos vs. opcionales**
  - **Críticos**: necesarios para funciones básicas (SSL, autenticación, redirección).
  - **Opcionales**: como mod_status, mod_info, mod_autoindex, pueden desactivarse si no se usan.

Carga y rendimiento

- **3. Desactivar lo innecesario mejora el rendimiento**
  - Apache permite desactivar módulos no usados para liberar RAM y CPU.
  - En Linux: a2dismod nombre_modulo
  - En Windows: comenta la línea LoadModule en httpd.conf

Carga y Rendimiento

- Mejoran el rendimiento

Módulo	Beneficio principal
`mod_deflate`	Compresión Gzip para reducir tamaño de respuesta
`mod_cache`	Almacenamiento en caché de contenido estático
`mod_expires`	Control de caché en navegador del cliente
`mod_pagespeed`	Optimización automática de contenido web

Carga y Rendimiento

- Módulos que pueden ralentizar si no se configuran bien

Módulo	Riesgo de carga excesiva
`mod_security`	Inspección profunda de cada solicitud
`mod_rewrite`	Reescritura compleja puede ralentizar
`mod_proxy`	Redirección de tráfico externo