

Hardening del servidor

¿Qué es?

En este apartado vamos a securizar todavía más el servidor. Esto lo hacemos junto a la securización o endurecimiento conseguido a través del firewall (iptables) y la configuración lógica, es decir, la creación de usuarios y grupos necesarios para cumplir con el requisito de **mínimo privilegio y mínima exposición**.

1 Configurar apache

Para conocer qué módulos están activos (aquellos que están en la carpeta `mods_enabled`). Podemos hacer un listado mediante el siguiente comando

```
sudo apache2ctl -t -D DUMP_MODULES
```

o mediante

```
kali@kali:~$ sudo a2dismod
Your choices are: access_compat alias auth_basic authn_core authn_file authz_core authz_host
Which module(s) do you want to disable (wildcards ok)?
```

Por ejemplo vamos a eliminar el módulo `mod_autoindex`. Si no sabes qué función tiene este módulo, puedes visitar esta página de apache. En pocas palabras, impide la creación automática de una página `index` cuando no encuentra el archivo (`index.html` o `index.php`) en la carpeta que se visita. Esta página generada automáticamente muestra todos aquellos archivos y directorios de la carpeta en cuestión.

Un ejemplo se ve al utilizar un Google Dork: `inurl: indexof` que devuelve páginas que están alojados en sitios donde se han dejado configuraciones por defecto en el servidor.

Otra opción que debemos suprimir es que **apache** no devuelva el tipo de servidor (o al menos la versión). De esta forma no damos pistas al atacante de la versión y/o tipo de servidor.

Index of /wp-content/uploads








Name	Last modified	Size	Description
 Parent Directory		-	
 2018/	2018-12-01 00:53	-	
 2019/	2019-12-01 00:04	-	
 2020/	2020-12-01 00:12	-	
 2021/	2021-03-01 00:09	-	
 GeoLite2-Country.mmdb	2018-08-23 17:18	3.3M	
 et_temp/	2018-01-31 16:05	-	

Figure 1: image-20210323194227850

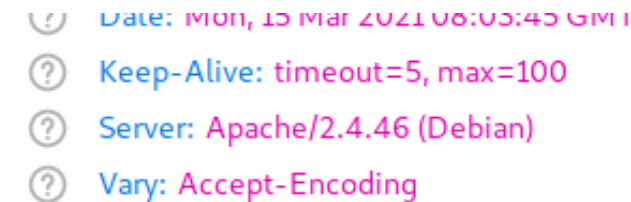
Por ejemplo, si lanzamos el comando:

```
curl --head localhost
```

que devuelve

```
HTTP/1.1 200 OK
Date: Mon, 15 Mar 2021 08:00:08 GMT
Server: Apache/2.4.46 (Debian)
Content-Type: text/html; charset=UTF-8
```

También se puede comprobar en las cabeceras de respuesta mediante el navegador



```
? Date: Mon, 15 Mar 2021 08:03:43 GMT
? Keep-Alive: timeout=5, max=100
? Server: Apache/2.4.46 (Debian)
? Vary: Accept-Encoding
```

Figure 2: image-20210315090439867

Para evitar que muestre la signature debemos modificar la configuración de apache en el archivo `/etc/apache2/apache2.conf` y añadir `ServerTokens ProductOnly`, de esta forma sólo pueden consultar la versión los módulos de apache y con `ServerSignature Off`, eliminamos completamente la signature

1.1 HSTS

Según la Wikipedia

HTTP Strict Transport Security o HTTP con Seguridad de Transporte Estricta (HSTS), es una política de seguridad web establecida para evitar ataques que puedan interceptar comunicaciones, cookies, etc. Según este mecanismo un servidor web declara que los agentes de usuario compatibles (es decir, los navegadores), solamente pueden interactuar con ellos mediante conexiones HTTP seguras (es decir, en HTTP sobre TLS/SSL1). HSTS es un estándar del IETF y se especifica en el RFC 6797.

La política HSTS es comunicada por el servidor al navegador a través de un campo de la cabecera HTTP de respuesta denominado “**Strict Transport-Security**”. La política HSTS especifica un período de tiempo durante el cual el agente de usuario deberá acceder al servidor sólo en forma segura.

En pocas palabras, le indica al navegador que, durante un tiempo definido en la cabecera, **sólo** puede actuar mediante HTTPS en el servidor.

Para configurarlo en Apache se debe añadir en el archivo de configuración del host virtual:

```
<VirtualHost *:443>
...
Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains"
...
</VirtualHost>
```

Que le indica al navegador que debe recordar durante más o menos 2 años (2 años*365 días*24 horas *60 minutos *60 segundos) que sólo debe acceder a la versión segura del sitio.

Más información en Developer Mozilla

1.2 CSP

Según Developer Mozilla

Política de Seguridad del Contenido o (CSP) - del inglés ***Content Security Policy*** - es una capa de seguridad adicional que ayuda a prevenir y mitigar algunos tipos de ataque, incluyendo Cross Site Scripting (XSS) y ataques de inyección de datos. Estos ataques son usados con diversos propósitos, desde robar información hasta desfiguración de sitios o distribución de malware .

Esto lo consigue mediante el envío de una cabecera de respuesta donde se indica de qué orígenes puede cargar el navegador contenido. Por ejemplo, podemos fijar que sólo se carguen scripts desde la propia página y así prevenir algunos ataques de XSS

Una cabecera de ejemplo sería

```
Content-Security-Policy: default-src 'self'
```

que indica todo el contenido provenga del mismo origen que el del sitio (esto excluye subdominios).

El administrador de un sitio web desea permitir que los usuarios de una aplicación web incluyan imágenes de cualquier origen en su propio contenido, pero restringen los medios de audio o video a proveedores de confianza, y todas las secuencias de comandos solo a un servidor específico que aloja un código de confianza.

```
Content-Security-Policy: default-src 'self'; img-src *; media-src media1.com media2.com; script-src
```

Aquí, de forma predeterminada, el contenido solo se permite desde el origen del documento, con las siguientes excepciones:

- Las imágenes pueden cargarse desde cualquier lugar (tenga en cuenta el comodín "*").
- Los archivos de medios solo están permitidos desde media1.com y media2.com (y no desde los subdominios de esos sitios).
- El script ejecutable solo está permitido desde userscripts.example.com.

Más ejemplos en la página de Mozilla

Para definirla en apache

```
Header set Content-Security-Policy \
    default-src 'self'; \
    img-src *; \
    media-src media1.com media2.com; \
    script-src userscripts.example.com
```

Práctica 1

Configura tu instalación de apache para:

- Deshabilitar el módulo autoindex

-
- Configurar la cabecera HSTS. Hay que habilitar el módulo **headers**. Recuerda que hay que instalar el certificado digital para el sitio seguro tal y como hicimos en la práctica de apache
 - Configurar la cabecera CSP con alguno de los ejemplos
 - Una vez configurado de esta manera crea un Dockerfile con toda esta configuración.

NOTA Para que docker pueda escuchar en dos puertos (80 y 403) hemos de ejecutar el comando run con estos puertos mapeados, por ejemplo:

```
docker run \  
  --detach \  
  --rm \  
  -p 8080:80 \  
  -p 8081:443 \  
  --name="hardenowasp" \  
  hardenowasp`
```

2 Web Application Firewall (WAF)

Según la Wikipedia

Un firewall de aplicaciones web (WAF) es un tipo de firewall que supervisa, filtra o bloquea el tráfico HTTP hacia y desde una aplicación web. Se diferencia de un firewall normal en que puede filtrar el contenido de aplicaciones web específicas, mientras que un firewall de red protege el tráfico entre los servidores. Al inspeccionar el tráfico HTTP un WAF protege a las aplicaciones web contra ataques como los de inyección SQL, XSS y falsificación de petición de sitios cruzados (CSRF).

En 2002 se creó el proyecto de código abierto ModSecurity para hacer la tecnología WAF más accesible y resolver los obstáculos dentro de la industria, como casos de negocios, barreras de costos y los conjuntos de reglas particulares de cada empresa. ModSecurity creó un conjunto de reglas básicas para proteger las aplicaciones web, basado en las vulnerabilidades detectadas por el OASIS Web Application Security Technical Committee's (WAS TC). En 2003, este trabajo fue ampliado y estandarizado con la creación de la Lista Top 10 del Open Web Application Security Project's (OWASP). OWASP publica con cierta regularidad una lista con los 10 riesgos de seguridad más críticos de las aplicaciones web. Esta lista se convertiría en la referencia de la industria para muchos temas de seguridad en la web.

Configurarlo **es bastante complicado** ya que funciona por reglas por las que aceptamos o rechazamos peticiones, pero podemos instalarlo **con**

las reglas que trae ya definidas el paquete. Simplemente, hemos de copiar el archivo `/etc/modsecurity/modsecurity.conf-recommended` en `/etc/modsecurity/modsecurity.conf`

Se reinicia Apache y ya funciona!

Para comprobar que está en funcionamiento, copia el archivo `post.php` realizado en el documento de validación en el `document_root` de Apache.

Si introducimos una entrada en el formulario que está bloqueada nos saltará un `status code 403`

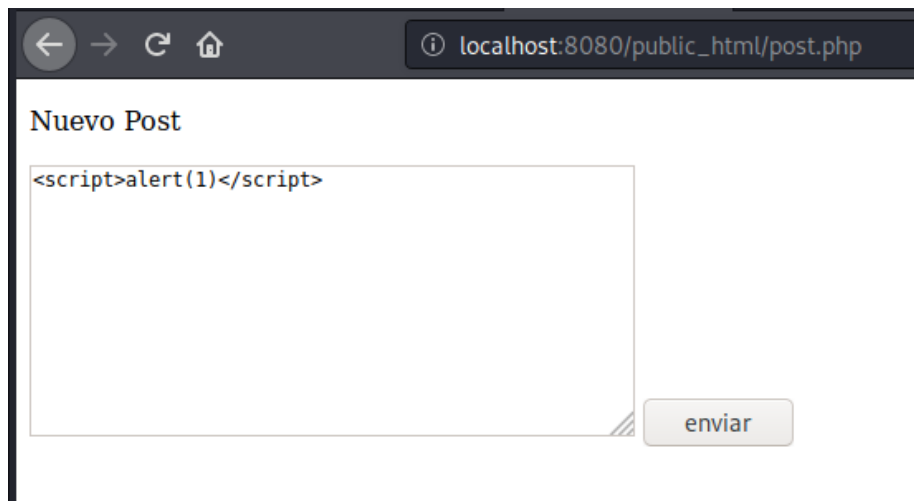


Figure 3: image-20210322185726779

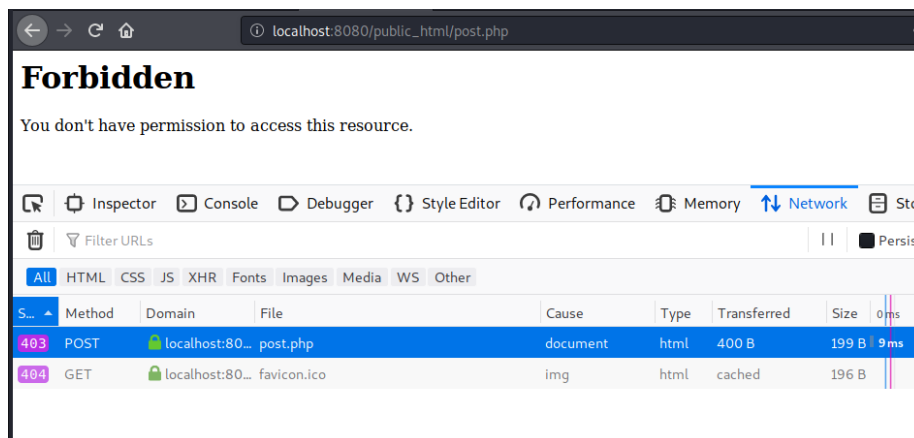


Figure 4: image-20210322185933273

Práctica 2

- Configura tu instalación de apache para que se atenga a las reglas de `mod_security`.
- Una vez configurado, crea un imagen de docker que configure una instalación de apache con `mod_security`.

3 Instalar reglas OWASP

Pero la OWASP provee una configuración por defecto que incluye una protección para las reglas más comunes. Así que lo mejor es empezar por este conjunto de reglas y luego ir añadiendo las propias.

Una solución de compromiso para no dar todas las reglas, se muestra una configuración que tiene OWASP. Para instalarlo en nuestro servidor:

1. Instalar `libapache2-mod-security2`
2. Hacemos un clon del repositorio

```
git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git
```

3. Entramos en la carpeta `owasp-modsecurity-crs` `cd owasp-modsecurity-crs` y movemos el archivo `crs-setup.conf.example`

```
sudo mv crs-setup.conf.example /etc/modsecurity/crs-setup.conf
```

4. Ahora movemos las reglas

```
sudo mv rules/ /etc/modsecurity
```

Si encuentras algún error, introduce:

```
sudo mkdir /etc/modsecurity/rules
cd rules
sudo cp *.* /etc/modsecurity/rules
```

5. Comprueba que en el archivo `security2.conf` se cargan las reglas

```
sudo nano /etc/apache2/mods-enabled/security2.conf
```

Comprueba que están las siguientes dos líneas:

```
IncludeOptional /etc/modsecurity/*.conf
Include /etc/modsecurity/rules/*.conf
```

Si no están, añádelas

6. Vamos a comprobar que funciona. Para ello edita el archivo de configuración del Host Virtual

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

y añade

```
SecRuleEngine On
SecRule ARGS:testparam "@contains test" "id:1234,deny,status:403,msg:'Cazado por Ciber"
```

y reinicia Apache

```
sudo systemctl restart apache2
```

7. Ahora prueba este comando `curl localhost/index.html?testparam=test` Será *cazado* por la regla que hemos definido en el archivo de configuración.

Y esta será la respuesta:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
```

8. También puedes probar a introducir las siguientes URLs

```
localhost/index.html?exec=/bin/bash
```

que pararía un ataque de command injection.

Y

```
localhost/index.html?exec=../../
```

que pararía un ataque de path traversal

Si compruebas el log de Apache

```
sudo tail /var/log/apache2/error.log
```

verás que las últimas líneas reflejan cómo han sido bloqueadas por `modsecurity`

```
[Tue Mar 23 14:05:41.485707 2021] [:error] [pid 2128] [client ::1:43736] [client ::1] M
```

Práctica 3

- Realiza una instalación de apache en la que se incluyan las reglas OWASP para `mod_security`.
- Una vez comprobada la instalación de estas reglas, crea un imagen de docker que realice la instalación y configuración de estas reglas

4 apache extra

Si queremos que sólo sirva tráfico a una IP, en `000-default.conf`

```
<Location />  
Require ip 192.168.1.24  
</Location>
```

De esta forma sólo le damos acceso a nuestro reverse proxy

Otra configuración por defecto que es recomendable es prohibir el uso del archivo `.htaccess` en la instalación de apache ya que este uso obliga a que en cada petición de página se deba *parsear* este archivo.

5 MySql

Restringir sólo el puerto local y nunca a la máquina remota

Dentro de `/etc/mysql` está el archivo `my.cnf`

dentro de `[mysqld]` poner

```
bind-address = 127.0.0.1
```

Reiniciar y ya sólo deja acceder desde localhost

Otra configuración interesante es eliminar la posibilidad de que mysql pueda leer cualquier archivo del sistema y eso es muy inseguro. Esta inclusión se realiza mediante `load_file('/etc/apt/sources.list')`

Para eliminar este permiso para listar archivos.

en `mariadb.conf.d` en `[mysqld]`

```
local-infile = 0
```

```
secure-file-priv = /dev/null
```

Otra medida es renombrar el usuario root. Para ello,

```
update mysql.user set user="ciberseguridad" where user="root"
flush privileges
```

6 Database Firewall (DBFW)

Al igual que los WAF, los DBFW se sitúan entre el agente de usuario y el servidor de base de datos para intentar *parar* las inyecciones de SQL (que veremos más adelante)

Se pueden configurar mediante:

- **listas blancas:** La Lista Blanca contiene secuencias de instrucciones SQL que se utilizan habitualmente en un entorno de base de datos determinado (por lo que se considera seguro). El firewall de la base de datos compara todas las consultas entrantes con las declaraciones de la Lista Blanca para definir si debe ignorarlas.
- **listas negras:** Esta lista contiene la descripción de amenazas potenciales. Si alguna declaración SQL detectada por un firewall está presente en la Lista Negra, esa consulta se bloqueará de inmediato.

Hay una implementación Open Source es GreenSQL que puede funcionar con los SGDB más comunes aunque la versión comunidad sólo protege contra ataques de inyección de SQL. La versión PRO protege además contra desbordamientos de buffer, escalada de privilegios, denegaciones de servicio...

7 Privilegios de los usuarios

De forma homóloga a los que ocurre en el sistema linux, en MySQL debemos tener una correcta gestión de los usuarios, ya sean para personas o cuentas de servicio para dar acceso a las aplicaciones, otorgando solo los permisos necesarios de los datos necesarios para cumplir con el requisito de mínimo privilegio y mínima exposición.

8 nginx y modsecurity

RETO

Instala nginx y realiza las mismas configuraciones que hemos llevado a cabo en Apache

- Instalación de PHP con un archivo index.php con el siguiente contenido

```
<?php  
phpinfo();
```

- directorio protegido mediante autenticación
- Certificado digital y configuración de un servidor seguro
- headers HSTS y CSP
- Reglas OWASP de mod_security
- La entrega estará formada por la imagen docker y una explicación detallada del proceso de instalación y configuración

Mas info en

<https://phoenixnap.com/kb/setup-configure-modsecurity-on-apache>

<https://www.securityartwork.es/2013/05/23/database-firewalls-introduccion/>

<https://developer.mozilla.org/es/docs/Web/HTTP/CSP>

<https://developer.mozilla.org/es/docs/Web/HTTP/Headers/Strict-Transport-Security>