

TALLER DE HARDENING DEL SERVIDOR WEB

Mario Lasso Mesa. IES Alixar

ÍNDICE

| | |
|---|-----------|
| 1. CONSTRUCCIÓN DEL CONTENEDOR APACHE | 2 |
| 1.1. Instalación de Docker y Docker Compose | 2 |
| 1.2. Ejecutar el contenedor con un usuario no root | 2 |
| 1.3. Deshabilitar el autoindexado en Apache | 3 |
| 1.4. No mostrar información del servidor | 5 |
| 1.5. Habilitar la cabecera CSP | 6 |
| 1.6. ModSecurity Web Application Firewall | 9 |
| 1.7. Protección de ataques DDoS con Mod_Evasive | 12 |
| 2. HTTPS CON LET'S ENCRYPT | 14 |
| 2.1. Registro de un nombre de dominio | 14 |
| 2.2. Generación del certificado con certbot | 16 |
| 2.3. Configuración de apache para usar el certificado | 20 |
| 2.4. Comprobación de la validez del certificado en el cliente | 24 |
| 2.5. Cabecera HTTP Strict Transport Security | 25 |
| 2.6. Renovación automática del certificado Let's Encrypt | 27 |
| 3. NGINX COMO PROXY WEB INVERSO | 28 |
| 4. HARDENING DEL CMS WORDPRESS | 32 |
| 4.1. Instalación de Wordpress y Proxy inverso nginx | 32 |
| 4.2. Fichero .htaccess | 36 |
| 4.3. Plugins de seguridad | 38 |
| 5. SEGURIDAD DE LA RED | 43 |
| 5.1. Reglas del firewall | 43 |
| 5.2. pfBlockerNG | 45 |
| 5.3. Suricata IPS | 47 |
| 6. ENLACES DE REFERENCIA | 49 |
| 7. REPOSITORIOS DEL TALLER | 49 |

1. CONSTRUCCIÓN DEL CONTENEDOR APACHE

1.1. Instalación de Docker y Docker Compose

Esta práctica se ha realizado en un sistema Ubuntu 24.04. Instala Docker Engine y Docker Compose en un sistema Ubuntu siguiendo las instrucciones de la [página web de Docker](#). Se recomienda añadir el repositorio de Docker, en lugar de usar los repositorios de Ubuntu, para disponer así de la última versión.

El demonio de Docker sólo puede ser lanzado por root (sudo) o por los usuarios pertenecientes al grupo docker. Añade un usuario sin privilegios al [grupo docker](#).

```
sudo usermod -aG docker $USER
newgrp docker
```

1.2. Ejecutar el contenedor con un usuario no root

En primer lugar, seleccionamos la imagen base que usaremos para instalar el servidor web. Vamos a comparar las vulnerabilidades de las imágenes debian, ubuntu y alpine escaneándolas mediante herramienta [Trivy](#). Para reutilizar la BD de vulnerabilidades se monta el directorio caché en el host mediante un bind mount. También requiere montar el socket de Docker, por lo que no debería crearse este contenedor en un host en producción accesible, ya que se podría comprometer y dar acceso privilegiado a un atacante.

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -v
$PWD:/tmp/.cache/ aquasec/trivy image debian
```

```
debian (debian 12.8)
=====
Total: 73 (UNKNOWN: 0, LOW: 57, MEDIUM: 14, HIGH: 1, CRITICAL: 1)
```

```
ubuntu (ubuntu 24.04)
=====
Total: 14 (UNKNOWN: 0, LOW: 6, MEDIUM: 8, HIGH: 0, CRITICAL: 0)
```

```
alpine (alpine 3.21.0)
=====
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```

Aunque la imagen alpine 3.21.0 presenta menos vulnerabilidades, como imagen base para el contenedor apache elegiremos ubuntu 24.04, el cual incluye php8.3 en su repositorio de software. En un apartado posterior se usará una imagen basada en alpine para construir un contenedor con el servidor nginx.

Construimos la imagen a partir de una imagen base de ubuntu con una instalación básica de apache más el módulo de PHP.

Al final del fichero Dockerfile se cambia al usuario www-data para evitar que el [contenedor se ejecute como root](#), lo cual supondría un [riesgo](#). Será preciso cambiar el propietario a www-data de los directorios en los que apache necesita escribir.

```
FROM ubuntu:latest
RUN apt update && \
apt -y upgrade && \
apt -qq -y install apache2 \
php libapache2-mod-php
RUN mkdir -p /var/run/apache2 /var/lock/apache2 /var/log/apache2
RUN chown -R www-data:www-data \
/var/run/apache2 /var/lock/apache2 /var/log/apache2
WORKDIR /var/www/iesalixar
USER www-data
EXPOSE 80
CMD ["apachectl", "-D", "FOREGROUND"]
```

Ejecutamos el contenedor y abrimos una shell para verificar (id, whoami) que el usuario con el que se ejecuta es www-data.

```
docker build -t hardening_apache:1.0 .
mkdir test
ls -l
touch test/fichero.txt
docker run --rm --name webserver -d -v ./test:/var/www/html/test \
-p 80:80 hardening_apache:1.0
```

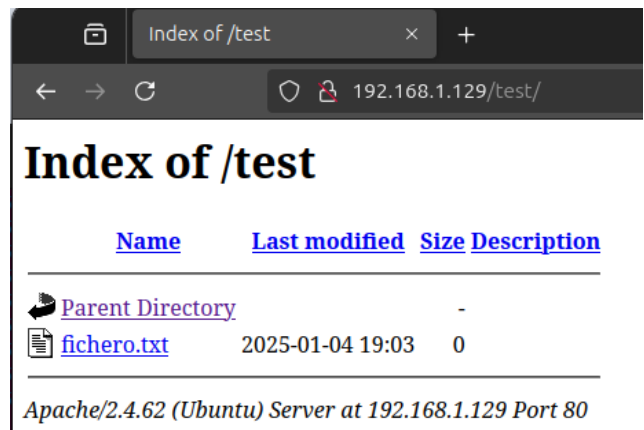
```
mario@UbuntuLXC-Mario:~/hardening_apache$ docker exec -it webserver bash
www-data@d2b46f442f35:~/iesalixar$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@d2b46f442f35:~/iesalixar$ pwd
/var/www/iesalixar
www-data@d2b46f442f35:~/iesalixar$
```

1.3. Deshabilitar el autoindexado en Apache

En este apartado vamos a deshabilitar el listado que por defecto hace Apache del contenido de aquellos directorios que no tienen una página index. Se trata de directorios en los que el usuario de apache www-data tiene permiso de lectura, de modo que es capaz de listar el contenido. Se deberían asignar los mínimos permisos de Linux, de modo que Apache solo tenga permiso de acceso (711) al directorio.

Ejecutamos el contenedor y comprobamos que por defecto es posible listar el contenido de un directorio web sin página de index.

```
docker build -t hardening_apache:1.1 .
mkdir test
ls -l
touch test/fichero.txt
docker run --rm --name webserver -d \
-v ./test:/var/www/html/test \
-p 80:80 hardening_apache:1.1
```



- Solución 1: Añadimos la directiva Options -Indexes en el fichero de configuración global de Apache /etc/apache2/apache2.conf.

```
docker cp webserver:/etc/apache2/apache2.conf ./config
sudo nano ./config/apache2.conf
<Directory /var/www/>
    Options -Indexes
    AllowOverride None
    Require all granted
</Directory>
```

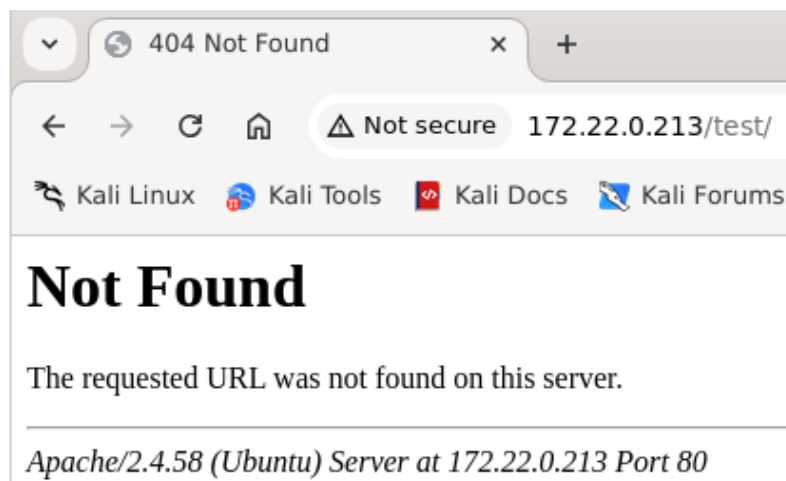
- Solución 2: Alternativamente, se puede desactivar el módulo autoindex. En general, si un módulo no se usa, es conveniente desactivarlo para reducir la superficie de ataque. Por tanto, ejecutamos también el comando a2dismod autoindex. Podemos ver los módulos activos de apache mediante el comando apachectl -M.

El Dockerfile con ambas soluciones implementadas quedaría así:

```
FROM ubuntu:latest
RUN apt update && \
apt -y upgrade && \
apt -qq -y install apache2 \
php libapache2-mod-php
COPY ./config/apache2.conf /etc/apache2/apache2.conf
RUN a2dismod -f autoindex
RUN mkdir -p /var/run/apache2 /var/lock/apache2 /var/log/apache2
RUN chown -R www-data:www-data \
/var/run/apache2 /var/lock/apache2 /var/log/apache2
WORKDIR /var/www/
USER www-data
EXPOSE 80
CMD ["apachectl", "-D", "FOREGROUND"]
```

Volvemos a construir la imagen con el cambio y comprobamos que ya nos se lista el contenido del directorio web:

```
docker stop webserver
docker build -t hardening_apache:1.1 .
docker run --rm --name webserver -d \
-v ./test:/var/www/html/test -p 80:80 hardening_apache:1.1
```



1.4. No mostrar información del servidor

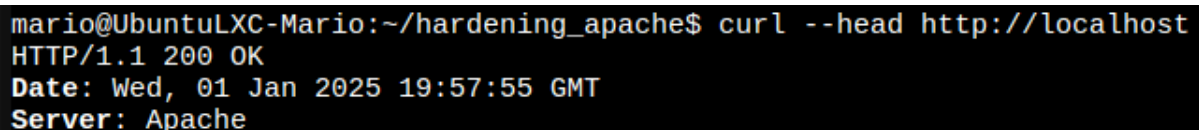
Obtenemos el fichero `security.conf` desde el contenedor y lo modificamos para cambiar las siguientes directivas `ServerTokens Prody` y `ServerSignature Off`.

```
docker cp webserver:/etc/apache2/conf-available/security.conf ./config
docker stop webserver
```

Y añadimos el fichero modificado en la imagen docker:

```
FROM ubuntu:latest
RUN apt update && \
apt -y upgrade && \
apt -qq -y install apache2 \
    php libapache2-mod-php
COPY ./config/apache2.conf /etc/apache2/apache2.conf
COPY ./config/security.conf /etc/apache2/conf-available/security.conf
RUN a2dismod -f autoindex
RUN mkdir -p /var/run/apache2 /var/lock/apache2 /var/log/apache2
RUN chown -R www-data:www-data \
    /var/run/apache2 /var/lock/apache2 /var/log/apache2
WORKDIR /var/www/
USER www-data
EXPOSE 80
CMD ["apachectl", "-D", "FOREGROUND"]
```

```
docker build -t hardening_apache:1.2 .
docker run --rm --name webserver -d -p 80:80 hardening_apache:1.2
```



```
mario@UbuntuLXC-Mario:~/hardening_apache$ curl --head http://localhost
HTTP/1.1 200 OK
Date: Wed, 01 Jan 2025 19:57:55 GMT
Server: Apache
```

Adicionalmente, sin necesidad de modificar el código fuente de Apache, el módulo `mod_security` permite, entre otras cosas que veremos, cambiar el string “Apache” por cualquier otro mediante la directiva `SecServerSignature`. Podríamos confundir al atacante y responder con otro tipo de servidor en las cabeceras o no mostrar ningún texto (espacio en blanco).

Añadimos al Dockerfile la instalación y activación del módulo `mod_security`

```
RUN apt update && \
apt -y upgrade && \
apt -qq -y install apache2 \
    php libapache2-mod-php \
    libapache2-mod-security2
RUN a2enmod security2
COPY ./config/apache2.conf /etc/apache2/apache2.conf
COPY ./config/security.conf /etc/apache2/conf-available/security.conf
RUN a2dismod -f autoindex
RUN mkdir -p /var/run/apache2 /var/lock/apache2 /var/log/apache2
RUN chown -R www-data:www-data \
    /var/run/apache2 /var/lock/apache2 /var/log/apache2
WORKDIR /var/www
USER www-data
```

```
EXPOSE 80
CMD ["apachectl", "-D", "FOREGROUND"]
```

Obtenemos desde el contenedor el fichero de configuración de mod_security

```
docker stop webserver
docker build -t hardening_apache:1.3 .
docker run --rm --name webserver -d -p 80:80 hardening_apache:1.3
docker cp webserver:/etc/modsecurity/modsecurity.conf-recommended
./config/modsecurity.conf
```

Y añadimos las siguientes directivas:

```
ServerTokens Min
SecServerSignature ``
```

Copiamos el fichero modsecurity.conf a la imagen añadiendo la siguiente línea al Dockerfile:

```
COPY ./config/modsecurity.conf /etc/modsecurity/modsecurity.conf
```

Y comprobamos que ha tenido efecto viendo que tampoco devuelve el tipo de servidor:

```
docker stop webserver
docker build -t hardening_apache:1.3 .
docker run --rm --name webserver -d -p 80:80 hardening_apache:1.3
```

```
mario@UbuntuLXC-Mario:~/hardening_apache$ curl --head http://localhost
HTTP/1.1 200 OK
Date: Wed, 01 Jan 2025 20:49:49 GMT
Server:
```

1.5. Habilitar la cabecera CSP

La cabecera de respuesta **Content Security Policy** permite al servidor web indicar al navegador desde qué orígenes tiene permitido descargar activos (imágenes, multimedia, estilos, scripts Javascript, etc), lo que sirve para mitigar los ataques **XSS** (Cross Site Scripting). Por defecto, sólo vamos a permitir que se descarguen activos desde el dominio origen de la propia página web (Content-Security-Policy: default-src 'self'). Para definir la cabecera con esta política en la respuesta, en Apache se utiliza la directiva Header set a nivel del virtualhost, la cual requiere activar el módulo headers.

Modificamos el fichero de configuración del virtualhost añadiendo la cabecera CSP:

```
docker cp webserver:/etc/apache2/sites-available/000-default.conf
./config/iesalixar.conf
docker cp webserver:/etc/apache2/sites-available/default-ssl.conf
./config/iesalixar-ssl.conf
docker stop webserver
cat ./config/iesalixar.conf
```

```
<VirtualHost *:80>
ServerName www.iesalixar.freeddns.org
ServerAlias iesalixar.freeddns.org
DocumentRoot /var/www/iesalixar
```

```

        ErrorLog ${APACHE_LOG_DIR}/iesalixar_error.log
        CustomLog      ${APACHE_LOG_DIR}/iesalixar_access.log
        combined
        Header set Content-Security-Policy "default-src 'self';"
    </VirtualHost>

```

En el Dockerfile, copiamos el fichero de configuración del virtualhost a la imagen, activamos el módulo headers y el sitio web.

```

COPY ./config/iesalixar.conf /etc/apache2/sites-available/iesalixar.conf

RUN a2enmod headers

RUN rm -rf /var/www/html && mkdir /var/www/iesalixar && chmod 711 /var/www/iesalixar

RUN a2dissite 000-default && a2ensite iesalixar
--
docker build -t hardening_apache:1.4 .
docker run --rm --name webserver -d -p 80:80 -v ./test:/var/www/iesalixar hardening_apache:1.4

```

```

mario@UbuntuLXC-Mario:~/hardening_apache$ curl --head http://iesalixar.freedomdns.org/prueba-csp.php
HTTP/1.1 200 OK
Date: Thu, 02 Jan 2025 12:50:06 GMT
Server:
Content-Security-Policy: default-src 'self';
Content-Type: text/html; charset=UTF-8

```

En el cliente, accedemos al sitio indicando el nombre en la URL. Para ello, añadimos el nombre y su alias al fichero /etc/hosts de resolución local de nombres del equipo cliente.

Hacemos una prueba pasando en petición GET un parámetro con JS:

```

http://iesalixar.freedomdns.org/prueba-csp.php?jsp=<script>alert(document.domain);</script>

```

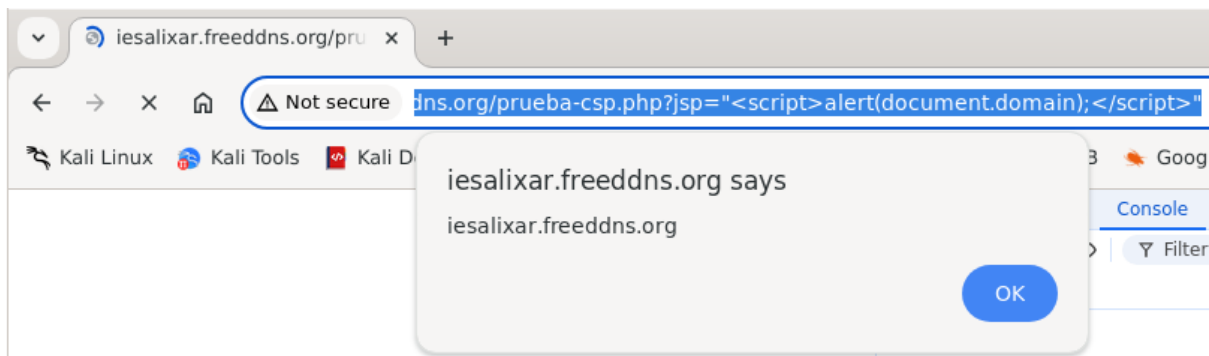
Y en el servidor añadimos dicha entrada en la respuesta sin sanitizarla:

```

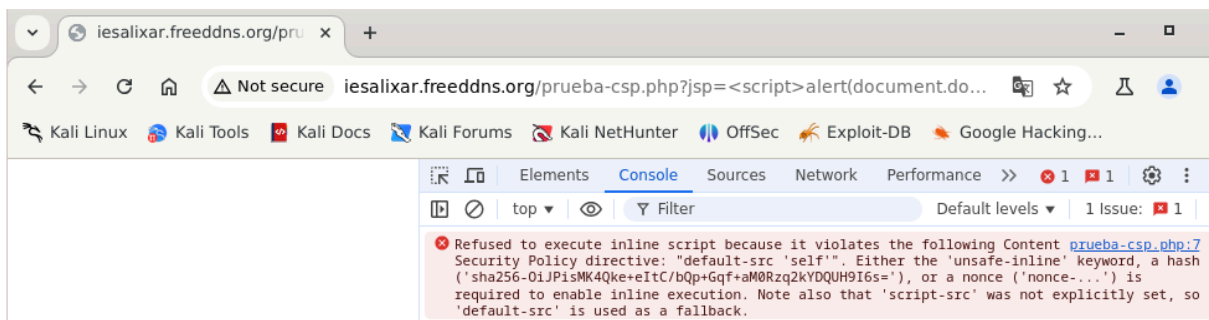
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
</head>
<body>
<?php
    if(isset($_REQUEST['jsp'])){
        echo $_REQUEST['jsp'];
    }
?>
</body>
</html>

```

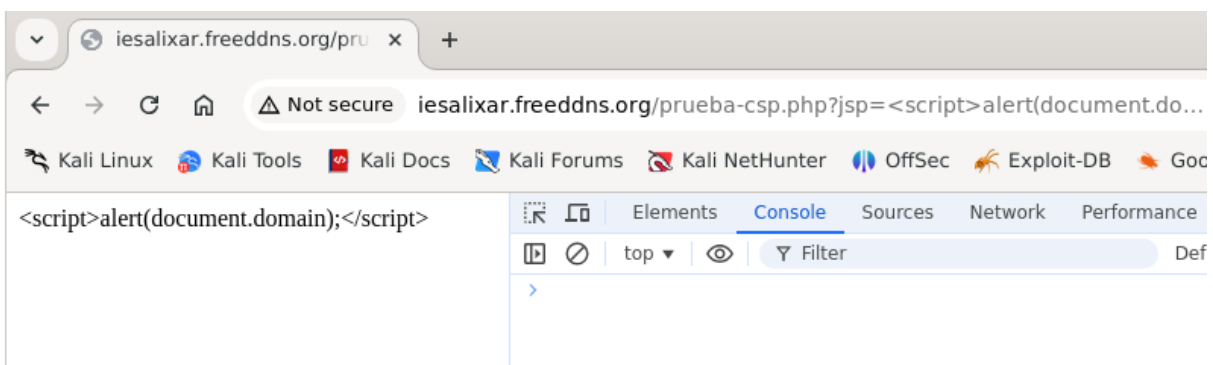
Sin el uso de la cabecera CSP con la política, el navegador ejecutaría el código JS.



Pero con la cabecera CSP, se evita la ejecución del JS inyectado en la página.



En este caso concreto, el XSS reflejado también se hubiera evitado si la entrada hubiera sido parseada antes en el backend.



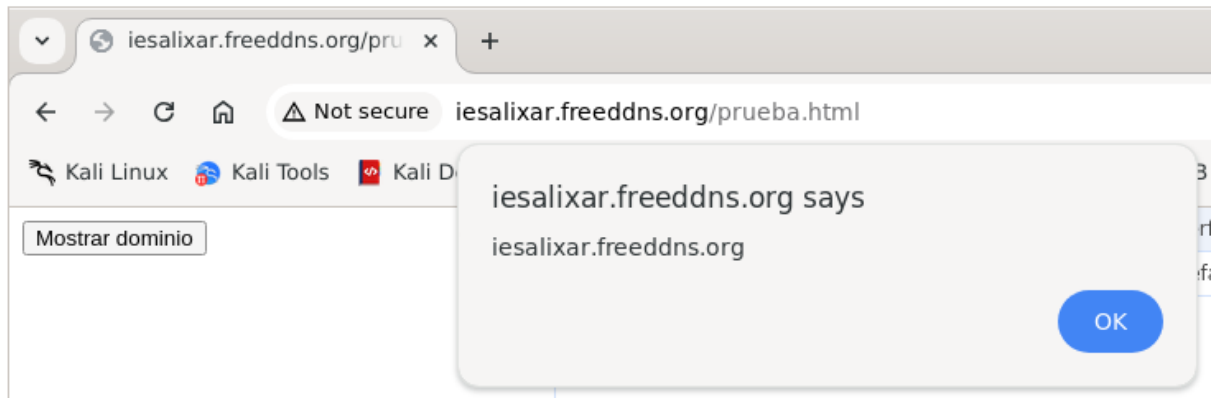
Otro ejemplo sencillo en el que puede verse que la política sí permite ejecutar código en el navegador que provenga de un fichero .js del mismo servidor web.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
</head>
<body>
  <button id="button">Mostrar dominio</button>
  <script src="script.js"></script>
</body>
</html>
```

```
cat test/script.js
document.getElementById("button").addEventListener("click",
mostrar_dominio);
```

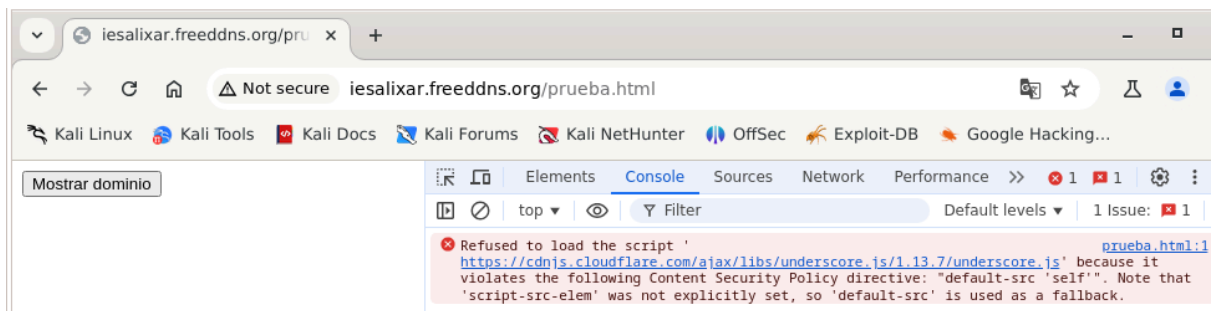


```
function mostrar_dominio() {
    alert(document.domain);
}
```



Pero si añadimos un script de un origen externo, el navegador lo bloquea.

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/underscore.js/1.13.7/underscore.js"></script>
```



1.6. ModSecurity Web Application Firewall

En nuestro Dockerfile ya habíamos instalado y activado el módulo `mod_security` en un apartado anterior. Copiamos el archivo de configuración por defecto de ModSecurity (`modsecurity.conf-recommended`) que habíamos configurado para usarlo.

```
RUN apt update && \
apt -y upgrade && \
apt -qq -y install apache2 \
    php libapache2-mod-php \
libapache2-mod-security2 git
RUN a2enmod security2
COPY ./config/modsecurity.conf
/etc/modsecurity/modsecurity.conf
```

En el fichero de configuración `/etc/modsecurity/modsecurity.conf` cambiamos el valor de la directiva `SecRuleEngine` de `DetectionOnly` a `On`.

```
cat config/modsecurity.conf
```

```
ServerTokens Min
SecServerSignature " "
SecRuleEngine On
```

Construimos la imagen y probamos el funcionamiento de ModSecurity con las reglas que tiene precargadas. Los eventos quedarán registrados en el fichero `/var/log/apache2/modsec_audit.log` y en el log de errores del virtualhost en `/var/log/apache2/iesalixar_error.log`.

```
docker build -t hardening_apache:1.5 .
docker run --rm --name webserver -d -p 80:80 hardening_apache:1.5
```

```
mario@UbuntuLXC-Mario:~/hardening_apache$ curl http://iesalixar.freedomdns.org/prueba.html?exec=/bin/bash
<!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>
mario@UbuntuLXC-Mario:~/hardening_apache$
```

```
mario@UbuntuLXC-Mario:~/hardening_apache$ docker exec -it webserver tail /var/log/apache2/iesalixar_error.log
[Sat Jan 04 12:51:23.882851 2025] [security2:error] [pid 9] [client 172.17.0.1:51888] [client 172.17.0.1] ModSecurity: Warning. Matched phrase "bin/bash" at ARGS:exec. [file "/usr/share/modsecurity-crs/rules/REQUEST-932-APPLICATION-ATTACK-RCE.conf"] [line "501"] [id "932160"] [msg "Remote Command Execution: Unix Shell Code Found"] [data "Matched Data: bin/bash found within ARGS:exec: /bin/bash"] [severity "CRITICAL"] [ver "OWASP CRS/3.3.5"] [tag "application-multi"] [tag "language-shell"] [tag "platform-unix"] [tag "attack-rce"] [tag "paranoia-level/1"] [tag "OWASP CRS"] [tag "capec/1000/152/248/88"] [tag "PCI/6.5.2"] [hostname "iesalixar.freedomdns.org"] [uri "/prueba.html"] [unique_id "Z3kuy_XIpwIUM2N3XJk_oQAAAAA"]
[Sat Jan 04 12:51:23.883337 2025] [security2:error] [pid 9] [client 172.17.0.1:51888] [client 172.17.0.1] ModSecurity: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. [file "/usr/share/modsecurity-crs/rules/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "94"] [id "949110"] [msg "Inbound Anomaly Score Exceeded (Total Score: 5)"] [severity "CRITICAL"] [ver "OWASP CRS/3.3.5"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"] [hostname "iesalixar.freedomdns.org"] [uri "/prueba.html"] [unique_id "Z3kuy_XIpwIUM2N3XJk_oQAAAAA"]
[Sat Jan 04 12:51:23.883598 2025] [security2:error] [pid 9] [client 172.17.0.1:51888] [client 172.17.0.1] ModSecurity: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "/usr/share/modsecurity-crs/rules/RESPONSE-980-CORRELATION.conf"] [line "92"] [id "980130"] [msg "Inbound Anomaly Score Exceeded (Total Inbound Score: 5 - SQLI=0,XSS=0,RFI=0,LFI=0,RCE=5,PHPI=0,HTTP=0,SESS=0): individual paranoia level scores: 5, 0, 0, 0"] [ver "OWASP CRS/3.3.5"] [tag "event-correlation"] [hostname "iesalixar.freedomdns.org"] [uri "/prueba.html"] [unique_id "Z3kuy_XIpwIUM2N3XJk_oQAAAAA"]
mario@UbuntuLXC-Mario:~/hardening_apache$
```

Se pueden descargar del [repositorio](#) oficial del proyecto [OWASP CRS](#) las últimas reglas disponibles del OWASP para ModSecurity. Clonamos en el directorio `/usr/share/modsecurity-crs` el repositorio, eliminando antes las reglas que vienen precargadas en `/usr/share/modsecurity-crs`. Renombramos el fichero `crs-setup.conf.example` a `crs-setup.conf` para configurarlas.

```
RUN rm -rf /usr/share/modsecurity-crs && \
git clone https://github.com/coreruleset/coreruleset
/usr/share/modsecurity-crs && \
mv /usr/share/modsecurity-crs/crs-setup.conf.example
/usr/share/modsecurity-crs/crs-setup.conf
```

Nos aseguramos que se cargan las reglas de OWASP en Apache. Para ello, modificamos el fichero `/etc/apache2/mods-available/security2.conf` :

```
docker cp webserver:/etc/apache2/mods-available/security2.conf config/security2.conf
cat config/security2.conf
```

```
<IfModule security2_module>
# Default Debian dir for modsecurity's persistent data
SecDataDir /var/cache/modsecurity
# Include all the *.conf files in /etc/modsecurity.
# Keeping your local configuration in that directory
# will allow for an easy upgrade of THIS file and
# make your life easier
```


1.7. Protección de ataques DDoS con Mod_Evasive

En el Dockerfile, instalamos también el módulo mod_evasive y lo activamos.

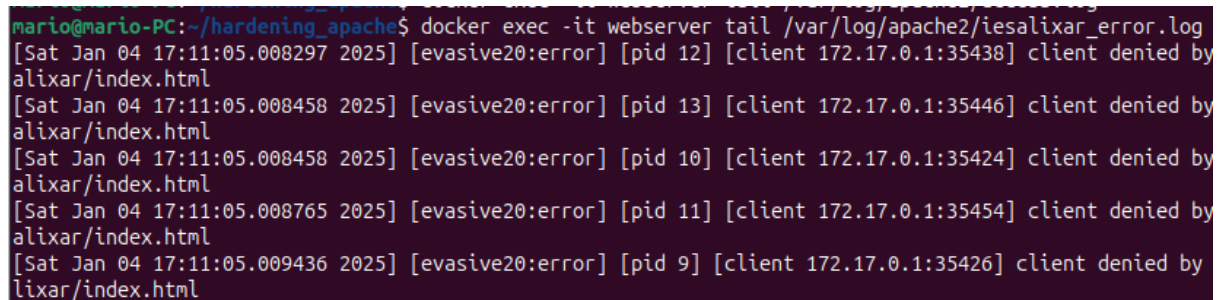
```
RUN apt update && \
    apt -y upgrade && \
    apt -qq -y install apache2 \
    php libapache2-mod-php \
    libapache2-mod-security2 git \
    libapache2-mod-evasive
```

Ajustamos su configuración con el fin de evitar falsos positivos, lo cual dependerá de la cantidad y tipo de tráfico que debería recibir el sitio web.

```
docker stop webserver
docker build -t hardening_apache:1.7 .
docker run --rm --name webserver -d -p 80:80 hardening_apache:1.7
docker cp webserver:/etc/apache2/mods-available/evasive.conf
config/evasive.conf
cat config/evasive.conf
DOSHashTableSize    2048
DOSPageCount        5
DOSSiteCount        100
DOSPageInterval     1
DOSSiteInterval     2
DOSBlockingPeriod   10
#DOSWhitelist
#DOSEmailNotify      you@yourdomain.com
#DOSSystemCommand    "su - someuser -c '/sbin/... %s ...'"
DOSLogDir            "/var/log/mod_evasive"
```

Y añadimos el fichero de configuración de mod_evasive a la imagen, además de crear el directorio con Apache como propietario para volcar los logs:

```
COPY config/evasive.conf /etc/apache2/mods-enabled/evasive.conf
RUN mkdir -p /var/log/mod_evasive && chown -R www-data:www-data
/var/log/mod_evasive
docker stop webserver
docker build -t hardening_apache:1.7 .
docker run --rm --name webserver -d -p 80:80 -v ./test:/var/www/iesalixar
hardening_apache:1.7
```



```
mario@mario-PC:~/hardening_apache$ docker exec -it webserver tail /var/log/apache2/iesalixar_error.log
[Sat Jan 04 17:11:05.008297 2025] [evasive20:error] [pid 12] [client 172.17.0.1:35438] client denied by
alixar/index.html
[Sat Jan 04 17:11:05.008458 2025] [evasive20:error] [pid 13] [client 172.17.0.1:35446] client denied by
alixar/index.html
[Sat Jan 04 17:11:05.008458 2025] [evasive20:error] [pid 10] [client 172.17.0.1:35424] client denied by
alixar/index.html
[Sat Jan 04 17:11:05.008765 2025] [evasive20:error] [pid 11] [client 172.17.0.1:35454] client denied by
alixar/index.html
[Sat Jan 04 17:11:05.009436 2025] [evasive20:error] [pid 9] [client 172.17.0.1:35426] client denied by
lixar/index.html
```

```

mario@mario-PC:~/hardening_apache$ docker exec -it webserver tail /var/log/apache2/iesalixar_access.log
172.17.0.1 - - [04/Jan/2025:17:11:05 +0000] "GET /index.html HTTP/1.0" 403 358 "-" "ApacheBench/2.3"
172.17.0.1 - - [04/Jan/2025:17:11:05 +0000] "GET /index.html HTTP/1.0" 403 358 "-" "ApacheBench/2.3"
172.17.0.1 - - [04/Jan/2025:17:11:05 +0000] "GET /index.html HTTP/1.0" 403 358 "-" "ApacheBench/2.3"
172.17.0.1 - - [04/Jan/2025:17:11:05 +0000] "GET /index.html HTTP/1.0" 403 358 "-" "ApacheBench/2.3"
172.17.0.1 - - [04/Jan/2025:17:11:05 +0000] "GET /index.html HTTP/1.0" 403 358 "-" "ApacheBench/2.3"
172.17.0.1 - - [04/Jan/2025:17:11:05 +0000] "GET /index.html HTTP/1.0" 403 358 "-" "ApacheBench/2.3"
172.17.0.1 - - [04/Jan/2025:17:11:05 +0000] "GET /index.html HTTP/1.0" 403 358 "-" "ApacheBench/2.3"
172.17.0.1 - - [04/Jan/2025:17:11:05 +0000] "GET /index.html HTTP/1.0" 403 358 "-" "ApacheBench/2.3"
172.17.0.1 - - [04/Jan/2025:17:11:05 +0000] "GET /index.html HTTP/1.0" 403 358 "-" "ApacheBench/2.3"
172.17.0.1 - - [04/Jan/2025:17:11:05 +0000] "GET /index.html HTTP/1.0" 403 358 "-" "ApacheBench/2.3"
mario@mario-PC:~/hardening_apache$

```

Probamos mod_evasive con ApacheBench:

```

mario@mario-PC:~/hardening_apache$ ab -n100 -c5 http://iesalixar.freedomdns.org/index.html
This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking iesalixar.freedomdns.org (be patient).....done


Server Software:
Server Hostname:      iesalixar.freedomdns.org
Server Port:          80

Document Path:        /index.html
Document Length:      10672 bytes

Concurrency Level:    5
Time taken for tests:  0.091 seconds
Complete requests:    100
Failed requests:       70
  (Connect: 0, Receive: 0, Length: 70, Exceptions: 0)
Non-2xx responses:    70
Total transferred:    353350 bytes
HTML transferred:     334090 bytes
Requests per second:  1094.92 [#/sec] (mean)
Time per request:     4.567 [ms] (mean)
Time per request:     0.913 [ms] (mean, across all concurrent requests)
Transfer rate:        3778.22 [Kbytes/sec] received


Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0     1
Processing:      2    4   2.4      3    15
Waiting:         1    4   2.4      3    15
Total:          2    4   2.4      3    15


Percentage of the requests served within a certain time (ms)
 50%    3
 66%    4
 75%    5
 80%    5
 90%    7
 95%    9
 98%   13
 99%   15
100%   15 (longest request)

```

2. HTTPS CON LET'S ENCRYPT

Resulta fundamental que las comunicaciones con nuestro servidor web vayan cifradas, ya que es preciso garantizar la confidencialidad y la integridad de la información. En suma a esto, los clientes web deben poder autenticar a nuestro servidor web, de modo que tengan la garantía de que la comunicación no se realiza con un sitio web que no sea el nuestro. Esto se consigue mediante el uso de un certificado digital firmado por parte de una autoridad certificadora en la que el cliente web confía. El cliente web valida la firma del certificado recibido mediante el certificado de la CA que tiene disponible en su almacén de certificados, comprobando así que el certificado recibido del servidor ha sido emitido para el nombre del sitio al que quiere acceder.

Como necesitamos un certificado que sea válido a nivel público para cualquier navegador, vamos a solicitarlo de forma gratuita a la CA Let's Encrypt. Para comprobar que el servidor que el agente que solicita el certificado controla realmente el dominio, Let's Encrypt proporciona dos posibles [tipos de retos](#):

- **HTTP-01 challenge:** Coloca temporalmente un recurso HTTP con una determinada información bajo una URL específica del servidor web `http://<YOUR_DOMAIN>/.well-known/acme-challenge/<TOKEN>`. Let's Encrypt puede verificarlo haciendo una HTTP al recurso generado en esa URL. Aunque es más sencillo, este reto resulta menos seguro (malware en ese directorio oculto y con acceso permitido puede pasar desapercibido). Además requiere que el servidor web esté arriba y que estén abiertos los puertos TCP 80 y 443. Por esta razón, no lo usaremos.
- **DNS-01 challenge:** Consiste en crear un registro TXT temporal bajo el dominio en el DNS con una determinada información. Para ello, certbot usa la API y el token de autenticación correspondiente a la cuenta que administra el dominio.

2.1. Registro de un nombre de dominio

Antes de poder solicitar un certificado, necesitaremos un nombre DNS para nuestro sitio web que en cada momento se resuelva por la IP pública dinámica de acceso a nuestro servidor. Para esta práctica se ha elegido el registrador de dominio acreditado [Dynu](#), ya que proporciona un servicio gratuito con una API compatible con el DNS challenge de Let's Encrypt. De forma gratuita se puede (se debe) activar el doble factor de autenticación (2FA) para el acceso a la cuenta.

En el apartado Dynamic DNS Service añadimos un subdominio gratuito a partir de uno de los dominios existentes. Para la práctica hemos creado el nombre `iesalixar.freeddns.org`.

Tras esto, una vez se propaguen los cambios en la zona DNS, se podrá resolver el nombre por la IP pública correspondiente mediante el comando nslookup:

```
mario@mario-PC:~$ nslookup iesalixar.freedomdns.org
Server:          127.0.0.53
Address:         127.0.0.53#53

Non-authoritative answer:
Name:   iesalixar.freedomdns.org
Address: 81.36.184.144

mario@mario-PC:~$ nslookup www.iesalixar.freedomdns.org
Server:          127.0.0.53
Address:         127.0.0.53#53

Non-authoritative answer:
Name:   www.iesalixar.freedomdns.org
Address: 81.36.184.144
```

Dynu proporciona un agente [Dynu Update Client](#) compatible con Linux que instalado en un equipo de la red se encarga de actualizar la IP pública asociada al nombre (DNS dinámico). Alternativamente, es posible usar otros métodos (curl, wget, nsupdate, GnuDIP) compatibles con la mayoría de routers y firewalls.

Los pasos indicados en la [documentación](#) para configurar el DNS dinámico mediante el Dynu Update Client son los siguientes:

- Descargar el paquete e instalar el servicio:
`sudo dpkg -i dynu-ip-update-client_0.1.0-1_amd64.deb`
- Configurar las credenciales de la cuenta Dynu en el fichero `/etc/dynuiuc/dynuiuc.conf`.

```
username XXXXXXXX
password XXXXXXXX
location
ipv4 true
ipv6 true
pollinterval 120
debug false
quiet false
```

Se debe usar para el DNS dinámico una contraseña distinta a la que da acceso a la web de administración. Para ello, se puede definir una contraseña alternativa en el campo New IP Update Password en Control Panel > My Account.

- Reiniciar el servicio y habilitarlo para que arranque en el inicio:

```
sudo systemctl restart dynuiuc.service
sudo systemctl enable dynuiuc.service
sudo systemctl status dynuiuc.service
```

- Revisar el log para ver que está comprobando si hay cambio de IP:

```
tail -f /var/log/dynuiuc.log
Fri Jan 10 23:23:51 2025 [ENGINE] Checking if IP update is required.
Fri Jan 10 23:23:54 2025 [ENGINE] IP address update initiated.
Fri Jan 10 23:23:54 2025 [ENGINE] Status Code: Good
Fri Jan 10 23:23:54 2025 [ENGINE] Current IPv4 Address:
Fri Jan 10 23:23:54 2025 [ENGINE] Current IPv6 Address:
Fri Jan 10 23:25:54 2025 [ENGINE] Checking if IP update is required.
Fri Jan 10 23:26:00 2025 [ENGINE] IP address update initiated.
Fri Jan 10 23:26:00 2025 [ENGINE] Status Code: No Change
```

2.2. Generación del certificado con certbot

El agente que proporciona Let's Encrypt para automatizar el proceso de solicitud del certificado y del par de claves pública/privada se llama [certbot](#). Durante el proceso, el programa solicita el nombre del host con el dominio (FQDN) para el que se emitirá el certificado, así como una dirección de correo electrónico válida. Esta dirección se utiliza para las notificaciones de renovación y los avisos de seguridad.

```
Dockerfile certbot + plugin certbot-dns-dynu
FROM python:3.9-slim
# Install Certbot and certbot-dns-dynu
RUN pip install certbot certbot-dns-dynu
```

Definimos el fichero .env con las variables de entorno:

```
EMAIL=prof.mario.lasso-mesa@iesalixar.org
DOMAIN=iesalixar.freeddns.org
ALTDOMAIN=www.iesalixar.freeddns.org
```

Y el fichero dynu-credentials.ini con la [API Key](#) obtenida en nuestra cuenta de Dynu, al que se deben asignar permisos solo para el propietario (chmod 600):

```
dns_dynu_auth_token = XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```


Definimos un fichero docker-compose.yml que va a servir para ejecutar la imagen Docker que hemos creado con certbot y el plugin [certbot-dns-dynu](#). Indicamos los bind mounts y las variables de entorno necesarias para los argumentos del programa certbot. Este fichero docker-compose.yml inicial lo iremos ampliando para poder lanzar el resto de contenedores del servicio web de forma conjunta:

```
services:
  certbot:
    container_name: certbot_temp
    build: .
    environment:
      - EMAIL
      - DOMAIN
      - ALTDOMAIN
    command: >-
      certbot certonly --dry-run --reinstall
      --authenticator dns-dynu
      --dns-dynu-credentials /root/dynu-credentials.ini
      --email ${EMAIL} --agree-tos
      --preferred-challenges dns
      --post-hook 'chown root:${GRUPO} /etc/letsencrypt/live \
        etc/letsencrypt/archive \
        && chmod 710 /etc/letsencrypt/live /etc/letsencrypt/archive \
        && chown root:${GRUPO} /etc/letsencrypt/archive/${DOMAIN}/priv \
        && chmod 640 /etc/letsencrypt/archive/${DOMAIN}/privkey*.pem'
      -d ${DOMAIN} -d ${ALTDOMAIN}
  volumes:
    - ./certs:/etc/letsencrypt
    - ./lib:/var/lib/letsencrypt
    - ./dynu-credentials.ini:/root/dynu-credentials.ini
```

Ha sido necesario añadir la opción `--post-hook` para añadir a `www-data` (con GID 33 en la imagen) como grupo propietario y cambiar los permisos de los directorios `live` y `archive` que crea certbot, así como del fichero de la clave privada, para que el grupo `www-data` del contenedor apache pueda leerlos en el bind mount.

Se definirá una variable de entorno `GRUPO` en el fichero `.env` con el valor 33 correspondiente al GID de apache `www-data`, según lo mostrado en el fichero `/etc/passwd` de la imagen. Se cambia el grupo propietario al GID 33 del fichero de la clave y de los directorios `live` (enlaces simbólicos) y `archive` (certificado y clave) dentro de `/etc/letsencrypt`. Al asignarle permiso de ejecución al directorio (`g=x`), se permite acceso al usuario al grupo `www-data`, manteniendo a 0 los permisos para otros. Del mismo modo, con el permiso de lectura (`g=r`) se permite a `www-data` leer el fichero de la clave privada.

Esta opción `--post-hook` se ejecuta solo cuando se genera con éxito un nuevo certificado, no lanzándose si no es necesario renovarlo. Si se ejecutara el contenedor de apache como root (sin la orden `USER www-data` en el Dockerfile), el servidor web podría acceder al certificado y a la clave sin tener que cambiar los permisos, pero esto no es recomendable. Se obtenía el siguiente error en los logs de apache:

```
mario@UbuntuLXC-Mario:~/hardening_apache_ssl$ docker logs webserver
AH00526: Syntax error on line 32 of /etc/apache2/sites-enabled/iesalixar-ssl.conf:
SSLCertificateFile: file '/etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem' does not exist or is empty
AH00526: Syntax error on line 32 of /etc/apache2/sites-enabled/iesalixar-ssl.conf:
SSLCertificateFile: file '/etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem' does not exist or is empty
AH00526: Syntax error on line 32 of /etc/apache2/sites-enabled/iesalixar-ssl.conf:
SSLCertificateFile: file '/etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem' does not exist or is empty
AH00526: Syntax error on line 32 of /etc/apache2/sites-enabled/iesalixar-ssl.conf:
SSLCertificateFile: file '/etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem' does not exist or is empty
AH00526: Syntax error on line 32 of /etc/apache2/sites-enabled/iesalixar-ssl.conf:
SSLCertificateFile: file '/etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem' does not exist or is empty
AH00526: Syntax error on line 32 of /etc/apache2/sites-enabled/iesalixar-ssl.conf:
SSLCertificateFile: file '/etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem' does not exist or is empty
```

En principio ejecutamos certbot con la opción dry-run para simular la generación del certificado en el entorno de pruebas de Let's Encrypt, evitando así un posible bloqueo temporal por parte de los servidores de Let's Encrypt si llegamos a un límite de generación de certificados ([hasta 5](#)) de forma continuada para el mismo dominio.

```
mario@UbuntuLXC-Mario:~/hardening_apache2/dockerfiles/certbot$ docker compose up
[+] Running 1/0
  ✓ Container certbot_temp Created
Attaching to certbot_temp
certbot_temp | Saving debug log to /var/log/letsencrypt/letsencrypt.log
certbot_temp | Simulating a certificate request for iesalixar.freedomdns.org and
www.iesalixar.freedomdns.org
certbot_temp | Waiting 60 seconds for DNS changes to propagate
certbot_temp | The dry run was successful.
certbot_temp exited with code 0
```

Una vez que el resultado del challenge DNS sea exitoso, volvemos a ejecutarlo sin la opción `--dry-run` para obtener el certificado X509v3.

```
mario@UbuntuLXC-Mario:~/hardening_apache2/dockerfiles/certbot$ docker compose up
[+] Running 1/1
  ✓ Container certbot_temp Recreated
Attaching to certbot_temp
certbot_temp | Saving debug log to /var/log/letsencrypt/letsencrypt.log
certbot_temp | Account registered.
certbot_temp | Requesting a certificate for iesalixar.freedomdns.org and www.iesalixar.freedomdns.org
certbot_temp | Waiting 60 seconds for DNS changes to propagate
certbot_temp | Successfully received certificate.
certbot_temp | Certificate is saved at: /etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem
certbot_temp | Key is saved at: /etc/letsencrypt/live/iesalixar.freedomdns.org/privatekey.pem
certbot_temp | This certificate expires on 2025-04-10.
certbot_temp | These files will be updated when the certificate renews.
certbot_temp | NEXT STEPS:
certbot_temp | - The certificate will need to be renewed before it expires. Certbot can
  automatically renew the certificate in the background, but you may need to take steps to
  enable that functionality. See https://certbot.org/renewal-setup for instructions.
certbot_temp | - - - - -
certbot_temp | If you like Certbot, please consider supporting our work by:
certbot_temp | * Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
certbot_temp | * Donating to EFF: https://eff.org/donate-le
certbot_temp | - - - - -
certbot_temp exited with code 0
mario@UbuntuLXC-Mario:~/hardening_apache2/dockerfiles/certbot$
```

Vamos a comprobar que se han añadido los SANs (Subject Alternative Name) de nuestro dominio y dominio alternativo (con www). Este es el campo que va a comprobar el navegador, verificando que coincida con el nombre indicado en la URL. También debe aparecer como Issuer la CA Let's Encrypt, que es una CA de confianza para los navegadores web. Podemos ver el contenido del certificado mediante el comando openssl, indicando la ruta del bind mount en el host donde se ha guardado el resultado.

```
sudo openssl x509 --text --noout -in
certs/live/iesalixar.freedomdns.org/fullchain.pem | more
```

```
mario@UbuntuLXC-Mario:~/hardening_apache2/dockerfiles/certbot$ sudo openssl x509 --text
--noout -in certs/live/iesalixar.freedomdns.org/fullchain.pem | more
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      04:03:7d:dc:4b:9e:5f:11:4d:9f:f9:65:45:0f:96:d0:8c:17
    Signature Algorithm: ecdsa-with-SHA384
    Issuer: C = US, O = Let's Encrypt, CN = E5
    Validity
      Not Before: Jan 10 17:18:21 2025 GMT
      Not After : Apr 10 17:18:20 2025 GMT
    Subject: CN = iesalixar.freedomdns.org
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
        04:eb:ad:03:1a:bf:51:dc:82:09:a6:6d:cb:5c:db:
        b6:30:57:82:30:71:23:9e:fc:62:bd:6d:eb:94:63:
        87:8a:78:ed:dd:28:5d:b7:f9:6b:28:70:ff:6e:04:
        25:18:e2:c8:66:fd:46:72:b5:d6:fa:32:22:b1:5a:
        9d:d6:e5:fb:70
      ASN1 OID: prime256v1
      NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Key Usage: critical
        Digital Signature
      X509v3 Extended Key Usage:
        TLS Web Server Authentication, TLS Web Client Authentication
      X509v3 Basic Constraints: critical
        CA:FALSE
      X509v3 Subject Key Identifier:
        9B:EB:75:66:7D:B0:DE:15:31:EF:05:81:54:F4:40:B3:D8:4C:8E:DB
      X509v3 Authority Key Identifier:
        9F:2B:5F:CF:3C:21:4F:9D:04:B7:ED:2B:2C:C4:C6:70:8B:D2:D7:0D
      Authority Information Access:
        OCSP - URI:http://e5.o.lencr.org
        CA Issuers - URI:http://e5.i.lencr.org/
      X509v3 Subject Alternative Name:
        DNS:iesalixar.freedomdns.org, DNS:www.iesalixar.freedomdns.org
      X509v3 Certificate Policies:
        Policy: 2.23.140.1.2.1
      CT Precertificate SCTs:
        Signed Certificate Timestamp:
          Version : v1 (0x0)
          Log ID : CC:FB:0F:6A:85:71:09:65:FE:95:9B:53:CE:E9:B2:7C:
                    22:E9:85:5C:0D:97:8D:B6:A9:7E:54:C0:FE:4C:0D:B0
          Timestamp : Jan 10 18:16:51.753 2025 GMT
          Extensions: none
```

En el repositorio de DockerHub de la imagen certbot/certbot aparecen una serie de imágenes Docker de certbot ya preparadas que incluyen el [plugin](#) correspondiente para hacer el challenge DNS con la API de diferentes servicios de DNS. Adicionalmente, en la [documentación de Certbot](#) se listan otras imágenes Docker de terceros compatibles con la

API de otros servicios de DNS, pero tampoco aparece entre estos ninguna imagen preparada que incluya el plugin para el servicio Dynu que estamos usando. Es por ello que ha hecho falta construir una imagen personalizada instalando certbot y el plugin certbot-dns-dynu en una imagen base ligera que permite ejecutar programas Python.

En el caso de un dominio contratado en OVH como nuestro dominio alixarblue.team, sería inmediato obtener el certificado mediante el uso de la imagen docker [certbot/dns-ovh](https://hub.docker.com/r/certbot/dns-ovh).

```
mario@mario-PC:~$ nslookup alixarblue.team
Server:          127.0.0.53
Address:         127.0.0.53#53

Non-authoritative answer:
Name:   alixarblue.team
Address: 81.36.184.144
```

De forma similar, primero hay que generar una API Key haciendo login con la cuenta que administra el dominio: <https://api.ovh.com/createToken/>

Creamos un fichero dns-ovh-credentials.ini con las credenciales obtenidas (chmod 600):

```
dns_ovh_endpoint = ovh-eu
dns_ovh_application_key = XXXXXXXXXXXX
dns_ovh_application_secret = XXXXXXXXXXXXXXXXXXXXXXXXXXXX
dns_ovh_consumer_key = XXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Y lanzamos el contenedor con la imagen oficial para hacer el challenge DNS:

```
docker run -it --rm -v ./certs:/etc/letsencrypt -v
./lib:/var/lib/letsencrypt -v
./dns-ovh-credentials.ini:/root/dns-ovh-credentials.ini certbot/dns-ovh
certonly --dry-run --preferred-challenges dns-01 --authenticator dns-ovh
--dns-ovh-credentials /root/dns-ovh-credentials.ini --email
prof.mario.lasso-mesa@iesalixar.org --agree-tos -d "alixarblue.team" -d
"www.alixarblue.team"
```

```
mario@UbuntuLXC-Mario:~/prueba_certbot-dns-ovh$ docker run -it --rm -v ./certs:/etc/letsencrypt
-v ./lib:/var/lib/letsencrypt -v ./dns-ovh-credentials.ini:/root/dns-ovh-credentials.ini certbot
/dns-ovh certonly --dry-run --preferred-challenges dns-01 --authenticator dns-ovh --dns-ovh-cred
entials /root/dns-ovh-credentials.ini --email prof.mario.lasso-mesa@iesalixar.org --agree-tos -d
"alixarblue.team" -d "www.alixarblue.team"
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Simulating a certificate request for alixarblue.team and www.alixarblue.team
Waiting 120 seconds for DNS changes to propagate
The dry run was successful.
mario@UbuntuLXC-Mario:~/prueba_certbot-dns-ovh$
```

2.3. Configuración de apache para usar el certificado

El siguiente paso será configurar el virtualhost de Apache para que use el certificado Let's Encrypt y la clave privada generados. Para ello, modificamos el fichero ./config/iesalixar-ssl.conf del siguiente modo:

```
<VirtualHost *:443>
    ServerName www.iesalixar.freeddns.org
```

```

ServerAlias iesalixar.freedomdns.org
ServerAdmin webmaster@localhost
DocumentRoot /var/www/iesalixar
ErrorLog ${APACHE_LOG_DIR}/iesalixar_error.log
CustomLog ${APACHE_LOG_DIR}/iesalixar_access.log combined
SSLEngine on
SSLCertificateFile /etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/iesalixar.freedomdns.org/privkey.pem

<FilesMatch "\.(?:cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
    SSLOptions +StdEnvVars
</Directory>

Header set Content-Security-Policy "default-src 'self';"
SecRuleEngine On

</VirtualHost>

<VirtualHost *:80>
    ServerName www.iesalixar.freedomdns.org
    ServerAlias iesalixar.freedomdns.org
    Redirect permanent / https://www.iesalixar.freedomdns.org
</VirtualHost>

```

En el Dockerfile copiamos y habilitamos el fichero de configuración del virtualhost HTTPS en lugar del virtualhost HTTP. A su vez, hay que activar el módulo ssl de Apache y exponer el puerto 443. El Dockerfile de apache quedaría como sigue:

```

# Partimos de la imagen base de Ubuntu
FROM ubuntu:latest

# Actualizamos los paquetes e instalamos apache2 php mod_security git mod_evasive
RUN apt update && \
    apt -y upgrade && \
    apt -qq -y install apache2 \
    php libapache2-mod-php \
    libapache2-mod-security2 git libapache2-mod-evasive

# Activamos el modulo mod_security
RUN a2enmod security2

# Copiamos a la imagen el fichero de configuracion de mod_security
COPY ./config/modsecurity.conf /etc/modsecurity/modsecurity.conf

# Actualizamos las reglas del OWASP para ModSecurity y las habilitamos en Apache
RUN rm -rf /usr/share/modsecurity-crs && \
    git clone https://github.com/coreruleset/coreruleset /usr/share/modsecurity-crs && \
    mv /usr/share/modsecurity-crs/crs-setup.conf.example \
    /usr/share/modsecurity-crs/crs-setup.conf
COPY ./config/security2.conf /etc/apache2/mods-available/security2.conf

# Copiamos a la imagen el fichero de configuracion global y security.conf
COPY ./config/apache2.conf /etc/apache2/apache2.conf
COPY ./config/security.conf /etc/apache2/conf-available/security.conf

```

```

# Copiamos a la imagen el fichero de configuracion del virtualhost
COPY ./config/iesalixar-ssl.conf /etc/apache2/sites-available/iesalixar-ssl.conf

# Activamos el modulo headers y ssl
RUN a2enmod headers
RUN a2enmod ssl

# Creamos el directorio raiz del sitio y lo activamos
RUN rm -rf /var/www/html
WORKDIR /var/www
RUN git clone https://github.com/iesalixar/alixar-blue-team-webpage.git iesalixar
&& \
    chmod 711 /var/www/iesalixar
RUN a2dissite 000-default && a2ensite iesalixar-ssl

#Desactivamos el modulo autoindex
RUN a2dismod -f autoindex

# Cambio de propietario en directorios para que apache pueda escribir
RUN mkdir -p /var/run/apache2 /var/lock/apache2 /var/log/apache2
RUN chown -R www-data:www-data \
    /var/run/apache2 /var/lock/apache2 /var/log/apache2
WORKDIR /var/www/iesalixar
USER www-data

# El contenedor escucha en los puertos TCP 80 y 443
EXPOSE 80 443

```

Configuramos el fichero docker-compose.yml para el manejo de ambos contenedores, de manera que el certificado generado por certbot pueda ser usado por el virtualhost de apache, para lo cual usamos dos bind mounts en el host (en directorio live se tienen enlaces simbólicos al certificado y la clave privada en el directorio archive).

```

services:
  certbot:
    container_name: certbot
    build: ./dockerfiles/certbot
    environment:
      - EMAIL
      - DOMAIN
      - ALTDOMAIN
    command: >-
      certbot certonly --reinstall
      --authenticator dns-dynu
      --dns-dynu-credentials /root/dynu-credentials.ini
      --email ${EMAIL} --agree-tos --no-eff-email
      --preferred-challenges dns
      --post-hook 'chown root:www-data /etc/letsencrypt/live \
        /etc/letsencrypt/archive \
        && chmod 710 /etc/letsencrypt/live /etc/letsencrypt/archive \
        && chown root:www-data /etc/letsencrypt/archive/${DOMAIN}/privkey*.pem \
        && chmod 640 /etc/letsencrypt/archive/${DOMAIN}/privkey*.pem'
      -d ${DOMAIN} -d ${ALTDOMAIN}

  volumes:
    - ./certs:/etc/letsencrypt
    - ./dynu-credentials.ini:/root/dynu-credentials.ini
  networks:
    - web_network

```

```

webserver:
  container_name: webserver
  build: ./dockerfiles/apache
  volumes:
    - ./certs:/etc/letsencrypt:ro
  restart: unless-stopped
  ports:
    - "80:80"
    - "443:443"
  networks:
    - web_network

networks:
  web_network:
    driver: bridge

```

Por defecto, certbot usa el servidor de producción de Let's Encrypt (<https://acme-v02.api.letsencrypt.org/directory>). Sería conveniente indicar primero la opción `--dry-run` para probar la generación del certificado con certbot, ya que cualquier error en las credenciales configuradas puede hacer que falle el challenge DNS. Se debe evitar hacer pruebas directamente con el [entorno de producción de Let's Encrypt](#), ya que tras varios intentos se bloquean temporalmente las solicitudes para un nombre de dominio.

Al arrancar los contenedores con el comando `docker compose up -d`, la primera vez se hace el build de las imágenes a partir de los dockerfiles. Tras esto, certbot ejecuta el challenge DNS y espera un tiempo para finalizar. Por su parte, Apache falla la primera vez que arranca debido a que no encuentra ningún certificado y la clave privada para el virtualhost HTTPS, de forma que permanece en estado `restarting` reintentando el arranque. Ejecutamos `docker compose ps` para comprobarlo:

```

mario@UbuntuLXC-Mario:~/hardening_apache_ssl$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
a0818a051b1b   hardening_apache_ssl-certbot       "certbot certonly --..." 23 seconds ago Up 22 seconds
eebde2df909    hardening_apache_ssl-webserver     "apachectl -D FOREGR..." 23 seconds ago Restarting (1)
3 seconds ago   webserver

```

Una vez que termina de ejecutarse el contenedor certbot, el certificado y la clave privada se guardan en su ubicación predefinida, de forma que el contenedor apache consigue levantar el servicio con el sitio web HTTPS activado.

```

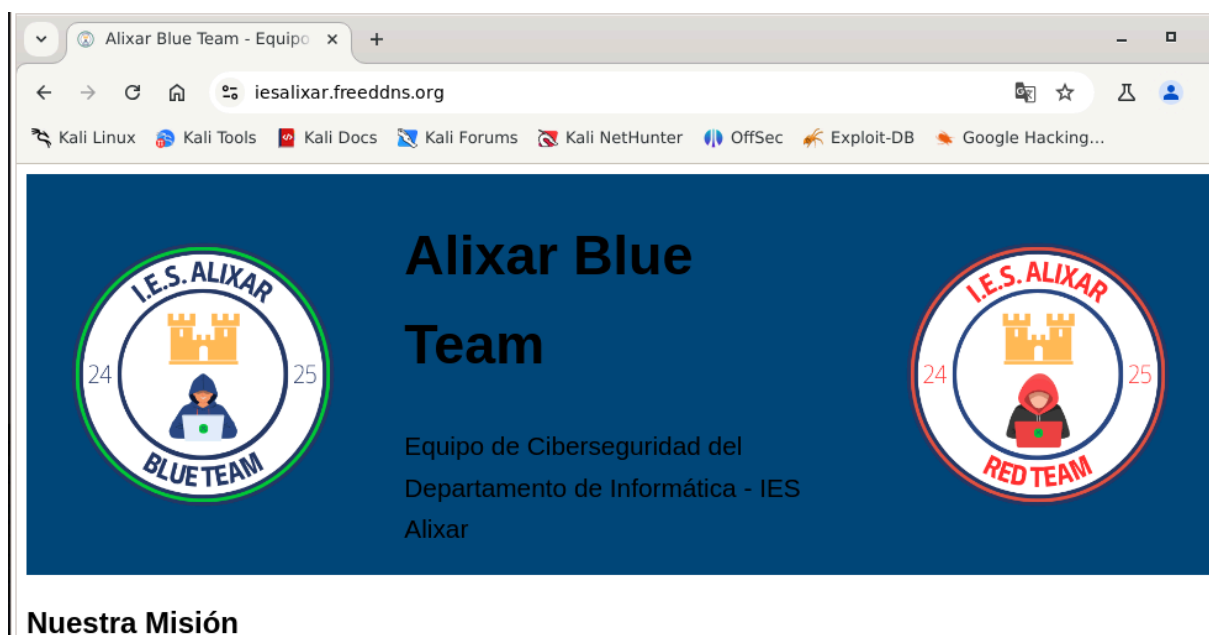
mario@UbuntuLXC-Mario:~/hardening_apache_ssl$ ls certs/live/
ls: cannot open directory 'certs/live/': Permission denied
mario@UbuntuLXC-Mario:~/hardening_apache_ssl$ sudo ls certs/live/
[sudo] password for mario:
README iesalixar.freedomdns.org
mario@UbuntuLXC-Mario:~/hardening_apache_ssl$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
eebde2df909    hardening_apache_ssl-webserver     "apachectl -D FOREGR..." About a minute ago Up 5 seconds
0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp webserver

```


2.4. Comprobación de la validez del certificado en el cliente

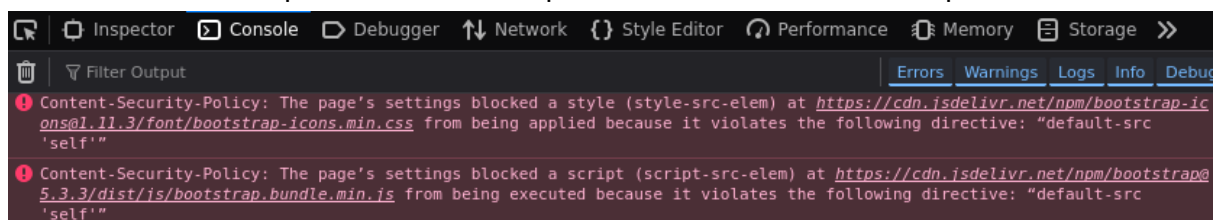
Desde un equipo cliente editamos el fichero `/etc/hosts` para añadir una línea, de forma que el cliente pueda resolver el nombre `www.iesalixar.freedomdns.org` (y también `iesalixar.freedomdns.org`) por la IP del host en el que se ejecuta el contenedor apache.

```
(usuario@kali)-[~]  
$ cat /etc/hosts | grep iesalixar  
172.22.0.220 iesalixar.freedomdns.org  
172.22.0.220 www.iesalixar.freedomdns.org
```



Si nuestro contenedor estuviese accesible desde el exterior de nuestra red, se podría usar igualmente el mismo nombre en la URL, ya que dicho nombre se resuelve por la IP pública correspondiente por el servicio DNS dinámico que hemos configurado previamente.

En la página web recibida como respuesta no se han podido cargar estilos y scripts externos debido a la política establecida por la cabecera CSP enviada por el servidor web.

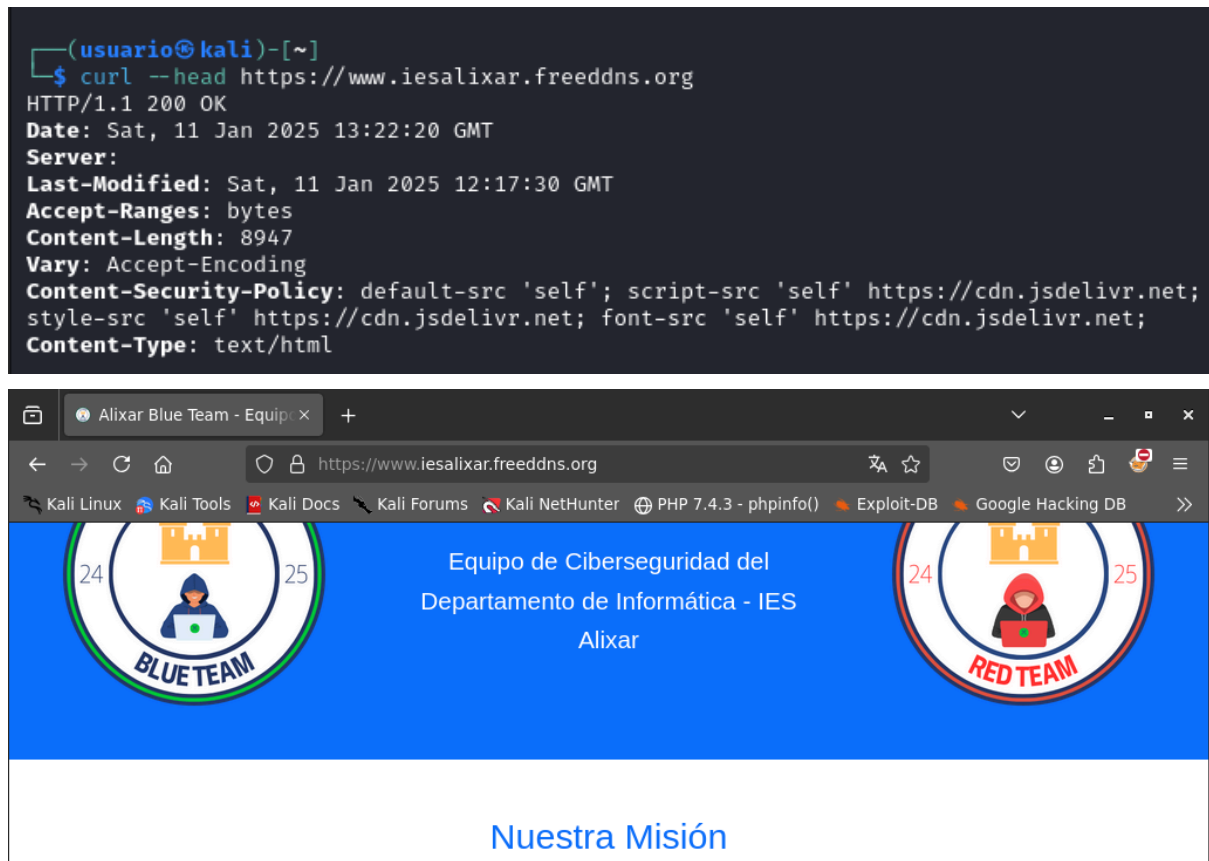


Modificamos la directiva CSP en el fichero de configuración del virtualhost para añadir a la política el CDN JSDelivr como origen permitido para estilos y scripts.


```
Header set Content-Security-Policy "default-src 'self'; script-src 'self' https://cdn.jsdelivr.net; style-src 'self' https://cdn.jsdelivr.net; font-src 'self' https://cdn.jsdelivr.net;"
```

Construimos de nuevo la imagen para que incluya este cambio:

```
docker-compose stop
docker-compose build
docker-compose up -d
```



2.5. Cabecera HTTP Strict Transport Security

Por otro lado, se ha añadido en el fichero de configuración del virtualhost la directiva para añadir a la respuesta la cabecera [HSTS](#) (HTTP Strict Transport Security), de modo que el servidor fuerce a los navegadores a que las conexiones al sitio web se hagan siempre a través de un canal seguro TLS. Si un usuario introduce en la URL el nombre del sitio sin especificar o incluso si indica al comienzo http en lugar de https, entonces el navegador automáticamente forzará la conexión a HTTPS. De este modo, con el uso de HTTPS no hará falta que el sitio añada una redirección en su configuración. Por su parte, con la opción `includeSubDomains` todos los subdominios deberán soportar HTTPS y no habrá que configurar el uso de la cabecera HSTS en cada uno de ellos.

```
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
```

Construimos de nuevo la imagen para que incluya este cambio:

```
docker-compose stop
docker-compose build
docker-compose up -d
```

Hacemos una petición al sitio y comprobamos que se devuelve la cabecera en la respuesta:

```
(usuario@kali)-[~]
$ curl --head https://www.iesalixar.freedomdns.org
HTTP/1.1 200 OK
Date: Sat, 11 Jan 2025 13:30:46 GMT
Server:
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Last-Modified: Sat, 11 Jan 2025 13:30:25 GMT
Accept-Ranges: bytes
Content-Length: 8947
Vary: Accept-Encoding
Content-Security-Policy: default-src 'self'; script-src 'self' https://cdn.jsdelivr.net; style-src 'self' https://cdn.jsdelivr.net; font-src 'self' https://cdn.jsdelivr.net;
Content-Type: text/html
```

Comprobamos también que el WAF ModSecurity sigue funcionando en el sitio HTTPS:

```
(usuario@kali)-[~]
$ curl -X POST https://iesalixar.freedomdns.org -d "username=usuario&password=' OR 1=1--"
<!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>
```

Una vez recibida la cabecera HSTS para un sitio, el navegador no permitirá acceder a ese sitio por HTTP. Del mismo modo, el navegador será igualmente estricto si se intenta acceder al sitio web con otro nombre diferente al del certificado:

NET::ERR_CERT_COMMON_NAME_INVALID

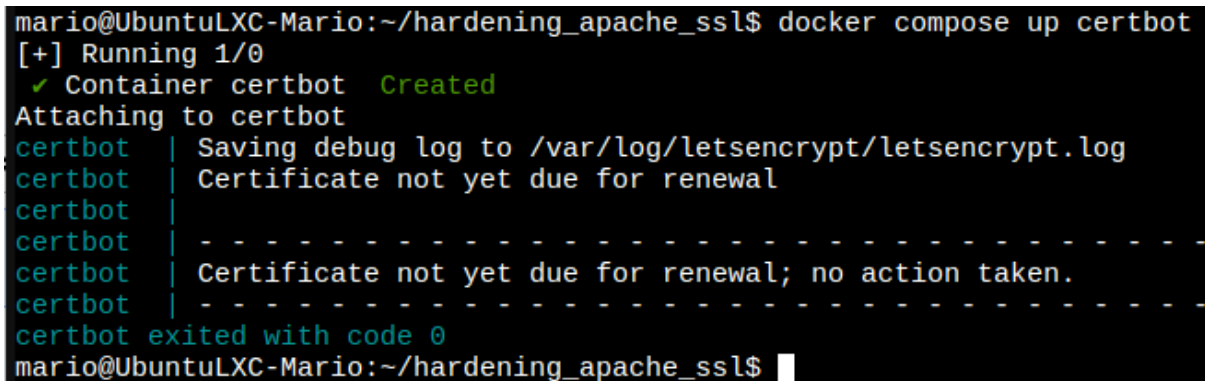
blog.iesalixar.freedomdns.org normally uses encryption to protect your information. When Chromium tried to connect to blog.iesalixar.freedomdns.org this time, the website sent back unusual and incorrect credentials. This may happen when an attacker is trying to pretend to be blog.iesalixar.freedomdns.org, or a Wi-Fi sign-in screen has interrupted the connection. Your information is still secure because Chromium stopped the connection before any data was exchanged.

You cannot visit blog.iesalixar.freedomdns.org right now because the website uses HSTS. Network errors and attacks are usually temporary, so this page will probably work later.

2.6. Renovación automática del certificado Let's Encrypt

El parámetro `--reinstall` en la ejecución del contenedor `certbot` sólo intentará crear o renovar el certificado si éste no existe o si está próximo a expirar (30 días antes de que se cumplan los 90 días de duración). Para la renovación del certificado será preciso crear una tarea programada en el `crontab` que ejecute un script `compose up` de `certbot` al menos una vez al día. El contenedor `certbot` terminará tras intentar renovar el certificado y no se volverá a ejecutar hasta la siguiente ejecución de la tarea programada.

```
docker compose -f /home/usuario/hardening_apache_ssl/docker-compose.yml up certbot
```



```
mario@UbuntuLXC-Mario:~/hardening_apache_ssl$ docker compose up certbot
[+] Running 1/0
 ✓ Container certbot Created
Attaching to certbot
certbot | Saving debug log to /var/log/letsencrypt/letsencrypt.log
certbot | Certificate not yet due for renewal
certbot |
certbot | - - - - -
certbot | Certificate not yet due for renewal; no action taken.
certbot | - - - - -
certbot exited with code 0
mario@UbuntuLXC-Mario:~/hardening_apache_ssl$
```

Si `certbot` obtiene un nuevo certificado y la clave privada asociada, lo almacenará en el directorio `/etc/letsencrypt/live` del `bind mount` compartido con el contenedor de `apache`. En este caso, será preciso hacer un `reload` el servidor web para que el cambio del certificado tenga efecto. Esto supone un reinicio de los procesos de tipo `graceful`, es decir, sin interrumpir las conexiones existentes, esperando a que se terminen de completar.

```
crontab -e
```

```
chmod u+x cron_job.sh
```

```
cat /etc/crontab
```

```
0 0 * * * $HOME/hardening_apache_ssl/scripts/cron_job.sh
```

```
/home/scripts/cron_job.sh
```

```
docker compose -f /home/usuario/hardening_apache_ssl/docker-compose.yml up certbot
```

```
docker compose -f /home/usuario/hardening_apache_ssl/docker-compose.yml exec \
webserver apachectl -k graceful
```

La solución propuesta en esta práctica para renovar el certificado Let's Encrypt del servidor web requiere poder editar el `crontab` del `host`, lo cual no siempre es posible. Además, implica una parada momentánea del servicio web todas las noches, sin interrumpir las conexiones existentes, por lo que podría mejorarse para que se haga un `reload` del servidor web sólo si el certificado efectivamente ha sido renovado (cada tres meses en lugar de a diario). Como alternativa, se podría haber instado `certbot` en la misma imagen `docker` que el servidor web, aprovechando así la posibilidad de usar un `hook` en `certbot` que permitiría recargar el servidor web sólo en el caso en que se renueve el certificado.

Por otro lado, existen soluciones más avanzadas que quedan fuera del alcance de esta práctica, como la imagen `Docker` desarrollada por la comunidad [linuxserver/swag](https://github.com/linuxserver/swag)

(Secure Web Application Gateway). Esta imagen libre implementa un proxy inverso nginx junto con la gestión de certificados con la mayoría de plugins dns disponibles para certbot, incluyendo el plugin [certbot-dns-dynu](#). Adicionalmente, la imagen Docker SWAG proporciona una capa extra de protección mediante la utilidad fail2ban para bloquear temporalmente en el firewall iptables aquellas direcciones IP que realicen accesos incorrectos al servidor, evitando así los accesos de bots que a día de hoy acaparan casi el 50% del tráfico de Internet.

Finalmente, paramos los contenedores y los eliminamos para realizar el siguiente apartado:

```
docker compose down
```

3. NGINX COMO PROXY WEB INVERSO

Nginx es una de las soluciones de proxy inverso y balanceo de carga más utilizadas. Mejora el rendimiento (proxy_cache), la seguridad (WAF, certificados SSL, DMZ), la fiabilidad y la escalabilidad de la web (balanceo de peticiones). Un proxy inverso web (o reverse proxy) es un servidor que actúa como intermediario entre los clientes web y los servidores web internos, reenviando las peticiones, de manera que un cliente web pueda acceder al contenido de un servidor web sin conocer su dirección IP o puerto.

Un proxy inverso se encarga de establecer la conexión HTTPS con los clientes y de gestionar los certificados de los sitios web. Suelen estar instalados en una red DMZ (zona desmilitarizada), especialmente protegidos por firewalls dada su mayor exposición al exterior.

Vamos a añadir un contenedor nginx que reciba las peticiones de nuestro sitio [www.iesalixar.freedomdns.org](https://iesalixar.freedomdns.org) y las reenvíe al contenedor apache capaz de ejecutar aplicaciones PHP que ya hemos implementado. Posteriormente, añadiremos un contenedor adicional independiente con la imagen oficial del CMS Wordpress, de tal manera que el proxy inverso nginx le reenvíe las peticiones hechas a la URL de nuestro sitio <https://iesalixar.freedomdns.org/blog>.

En el fichero docker-compose.yml resultante los puertos 80 y 443 del host dejan de estar mapeados con el contenedor de apache para mapearse con el contenedor de nginx. Respecto al certificado, el contenedor de nginx será el encargado de entregarlo a los clientes web, por lo que montará el bind mount compartido con el contenedor de certbot, el cual cambiará los permisos y el grupo propietario de nginx mediante un post-hook.

```
nginx: [emerg] cannot load certificate "/etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem": BIO_new_file() failed (SSL: error:80000000:system library::Permission denied:calling fopen(/etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem, r) error:10080002: BIO routines::system lib)
```

Se definirá una variable de entorno GRUPO en el fichero .env con el valor 101 correspondiente al GID de nginx, según lo mostrado en el fichero /etc/passwd de la imagen. Se cambia el grupo propietario al GID 101 del fichero de la clave y de los directorios live (enlaces simbólicos) y archive (certificado y clave) dentro de /etc/letsencrypt. Al asignarle permiso de ejecución al directorio (g=x), se permite acceso al usuario al grupo nginx,

manteniendo a 0 los permisos para otros. Del mismo modo, con el permiso de lectura (g=r) se permite a nginx leer el fichero de la clave privada.

```
services:
  certbot:
    container_name: certbot
    build: ./dockerfiles/certbot
    environment:
      - EMAIL
      - DOMAIN
      - ALTDOMAIN
    command: >-
      certbot certonly --reinstall
      --authenticator dns-dynu
      --dns-dynu-credentials /root/dynu-credentials.ini
      --email ${EMAIL} --agree-tos --no-eff-email
      --preferred-challenges dns
      --post-hook 'chown root:${GRUPO} /etc/letsencrypt/live \
        /etc/letsencrypt/archive && chmod 710 /etc/letsencrypt/live \
        /etc/letsencrypt/archive && chown root:${GRUPO} \
        /etc/letsencrypt/archive/${DOMAIN}/privkey*.pem \
        && chmod 640 /etc/letsencrypt/archive/${DOMAIN}/privkey*.pem'
      -d ${DOMAIN} -d ${ALTDOMAIN}
    volumes:
      - ./certs:/etc/letsencrypt
      - ./dynu-credentials.ini:/root/dynu-credentials.ini
    networks:
      - web_network

  nginx:
    container_name: nginx
    build: ./dockerfiles/nginx
    depends_on:
      - webserver
    ports:
      - "80:80"
      - "443:443"
    environment:
      - PROXY = 1
    volumes:
      - ./certs:/etc/letsencrypt:ro
    restart: unless-stopped
    networks:
      - web_network

  webserver:
    container_name: webserver
    build: ./dockerfiles/apache
    restart: unless-stopped
    networks:
      - web_network

networks:
  web_network:
    driver: bridge
```

Usaremos una [imagen docker oficial del OWASP](#) que incluye un servidor nginx junto con el WAF ModSecurity y las últimas reglas recomendadas. En el Dockerfile se copia el

fichero de configuración del sitio iesalixar-ssl.conf al directorio /etc/nginx/conf.d, el cual es el equivalente en Alpine para el fichero /etc/nginx/sites-available/ en Debian.

El contenido del Dockerfile quedaría así:

```
FROM owasp/modsecurity-crs:nginx-alpine
COPY ./config/iesalixar-ssl.conf /etc/nginx/conf.d/iesalixar-ssl.conf
```

El fichero de configuración del sitio del proxy inverso sería el siguiente:

```
server {
    server_name iesalixar.freedomdns.org www.iesalixar.freedomdns.org;
    location / {
        proxy_pass https://webserver;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_connect_timeout 300s;
        proxy_send_timeout 300s;
        proxy_read_timeout 300s;
        send_timeout 300s;
        proxy_buffer_size 16k;
        proxy_buffers 4 32k;
        proxy_busy_buffers_size 64k;
    }
}

add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
listen [::]:443 ssl ipv6only=on;
listen 443 ssl;
ssl_certificate /etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/iesalixar.freedomdns.org/privkey.pem;
}

server {
    if ($host = www.iesalixar.freedomdns.org) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
    if ($host = iesalixar.freedomdns.org) {
        return 301 https://$host$request_uri;
    }
    listen 80;
    listen [::]:80;
    server_name iesalixar.freedomdns.org www.iesalixar.freedomdns.org;
    return 404;
}
```

Tras construir la imagen (docker compose build) y ejecutar los contenedores (docker compose up -d), desde una shell dentro del contenedor de nginx podemos comprobar que llega la respuesta correcta al hacer la petición a la URL del virtualhost de apache:

```
mario@UbuntuLXC-Mario:~/hardening_apache_nginx$ docker compose ps
NAME                IMAGE                                COMMAND                                SERVICE
nginx               hardening_apache_nginx-nginx       "/docker-entrypoint..."            nginx
5 minutes ago      Up 5 minutes (healthy)              0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp, 8080/tcp
webserver           hardening_apache_nginx-iesalixar.freedomdns.org "apachectl -D FOREGR..."          iesalixar.freedomdns.org
58 minutes ago     Up 5 minutes                        80/tcp, 443/tcp
```

```
mario@UbuntuLXC-Mario:~/hardening_apache_nginx$ docker exec -it nginx sh
/usr/share/nginx/html # curl https://iesalixar.freedomdns.org
<!DOCTYPE html>
<html lang="es" >
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Alixar Blue Team - Equipo de Ciberseguridad del Departamento de Informática - IES Alixar</title>

    <!-- Metaetiquetas para SEO -->
```

A su vez, accediendo desde el cliente por https se comprueba que el proxy inverso reenvía al servidor web apache las peticiones que recibe al nombre del sitio, a la vez que devuelve al cliente la página web que obtiene de respuesta.

```
mario@UbuntuLXC-Mario:~/hardening_apache_nginx$ curl https://iesalixar.freedomdns.org
<!DOCTYPE html>
<html lang="es" >
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Alixar Blue Team - Equipo de Ciberseguridad del Departamento de Informática - IES Alixar</title>

    <!-- Metaetiquetas para SEO -->
    <meta name="title" content="Alixar Blue Team - Equipo de Ciberseguridad del Departamento de Informática - IES Alixar">
    <meta name="description" content="El Alixar Blue Team tiene como objetivo proporcionar una infraestructura de ciberseguridad simulando un entorno seguro que sirva como referencia para la docencia en el IES Alixar.">
    <meta name="image" content="https://alixarblue.team/assets/images/logo-blue-team.png">
```

Comprobamos que ModSecurity está funcionando haciendo una petición con una URL que haga saltar alguna regla. Tras esto, verificamos los errores en el log de nginx:

```
mario@UbuntuLXC-Mario:~/hardening_apache_nginx$ tail logs/nginx/error.log
2025/01/12 00:28:18 [warn] 1#1: "ssl_stapling" ignored, issuer certificate not found for certificate "/etc/nginx/conf/server.crt"
2025/01/12 00:28:18 [notice] 1#1: ModSecurity-nginx v1.0.3 (rules loaded inline/local/remote: 0/809/0)
2025/01/12 00:28:18 [notice] 1#1: libmodsecurity3 version 3.0.13
2025/01/12 00:28:21 [error] 575#575: *1 [client 172.20.0.1] ModSecurity: Access denied with code 403 (phase 2). Matched "Operator 'Ge' with parameter '5' against variable 'TX:BLOCKING_INBOUND_ANOMALY_SCORE' (Value: '20') [file "/etc/modsecurity.d/owasp-crs/rules/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "222"] [id "949110"] [rev "1"] [msg "Inbound Anomaly Score Exceeded (Total Score: 20)"] [data ""] [severity "0"] [ver "OWASP_CRS/4.10.0"] [maturity "0"] [accuracy "0"] [tag "modsecurity"] [tag "anomaly-evaluation"] [tag "OWASP_CRS"] [hostname "172.20.0.4"] [uri "/"] [unique_id "173664170168.364065"] [ref ""]], client: 172.20.0.1, server: iesalixar.freedomdns.org, request: "GET /?exec=../../../../ HTTP/1.1", host: "iesalixar.freedomdns.org"
mario@UbuntuLXC-Mario:~/hardening_apache_nginx$
```

Por último, añadimos la orden para hacer el reload del servicio nginx del contenedor al script que se ejecuta a diario con el fin de renovar y cargar el certificado si fuera necesario.

```
cat cron_job.sh
```

```
docker compose -f $HOME/hardening_apache_nginx/docker-compose.yml up certbot
```

```
docker compose -f $HOME/docker-compose.yml exec webserver apachectl -k graceful
```

```
docker compose -f $HOME/docker-compose.yml exec nginx nginx -s reload
```

4. HARDENING DEL CMS WORDPRESS

4.1. Instalación de Wordpress y Proxy inverso nginx

Añadimos dos contenedores adicionales al fichero docker-compose.yml anterior, uno para el contenedor de Wordpress y otro para el servidor de BD MySQL.

```
blog:
  container_name: blog
  depends_on:
    - db
  image: wordpress:latest
  restart: always
  environment:
    WORDPRESS_DB_HOST: db:3306
    WORDPRESS_DB_USER: ${MYSQL_USER}
    WORDPRESS_DB_PASSWORD: ${MYSQL_PASSWORD}
    WORDPRESS_DB_NAME: ${MYSQL_DATABASE}
  volumes:
    - ./wp-content:/var/www/html/wp-content
  networks:
    - web_network
db:
  container_name: db
  image: mysql:latest
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
  volumes:
    - mysql:/var/lib/mysql
  networks:
    - web_network
networks:
  web_network:
    driver: bridge
volumes:
  mysql: {}
```

Y establecemos el valor de las variables de entorno para la conexión del CMS a la BD en el fichero .env:

```
EMAIL=prof.mario.lasso-mesa@iesalixar.org
DOMAIN=iesalixar.freeddns.org
ALTDOMAIN=www.iesalixar.freeddns.org
MYSQL_ROOT_PASSWORD= WWWW
MYSQL_DATABASE= XXXX
MYSQL_USER= YYYY
MYSQL_PASSWORD= ZZZZ
```

El fichero de configuración del sitio en el proxy inverso sería el siguiente:

```
server {
    server_name blog.iesalixar.freeddns.org
    www.blog.iesalixar.freeddns.org;
    location / {
```



```

        proxy_pass http://blog;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_connect_timeout 300s;
        proxy_send_timeout 300s;
        proxy_read_timeout 300s;
        send_timeout 300s;
        proxy_buffer_size 16k;
        proxy_buffers 4 32k;
        proxy_busy_buffers_size 64k;
    }
    listen [::]:443 ssl ipv6only=on;
    listen 443 ssl;
    ssl_certificate
/etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/iesalixar.freedomdns.org/privkey.pem;
}
server {
    if ($host = www.blog.iesalixar.freedomdns.org) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
    if ($host = blog.iesalixar.freedomdns.org) {
        return 301 https://$host$request_uri;
    }
    listen 80;
    listen [::]:80;
    server_name blog.iesalixar.freedomdns.org www.blog.iesalixar.freedomdns.org;
    return 404;
}

```

El proxy inverso podrá recibir peticiones a dos URLs distintas:

- <https://iesalixar.freedomdns.org>, que reenviará al contenedor apache.
- <https://blog.iesalixar.freedomdns.org>, que reenviará al contenedor de Wordpress.

En este caso será preciso solicitar un certificado wildcard, de manera que sea válido para todos los FQDN que compartan el mismo dominio *.iesalixar.freedomdns.org.

Para solucionarlo, añadimos en el docker-compose.yml las siguientes opciones para el comando que ejecuta el contenedor certbot: -d \${DOMAIN} -d *.\${DOMAIN}

En el panel de control de Dynu (en Manage DNS Records dentro de Dynamic DNS Services), podemos comprobar que existe un registro de tipo A apuntando a nuestra IP pública.

| Hostname ? | Type ? | Data ? | TTL ? |
|----------------------------|--------|-----------------|-------|
| *.iesalixar.freedomdns.org | A | 81.36.184.144 | 120 |
| *.iesalixar.freedomdns.org | AAAA | | 120 |
| iesalixar.freedomdns.org | AAAA | | 120 |
| iesalixar.freedomdns.org | A | (81.36.184.144) | 120 |

Comprobamos que resolvemos el nombre blog.iesalixar.freedomdns.org

```
mario@mario-PC:~/hardening_apache_nginx_wp$ nslookup blog.iesalixar.freedomdns.org
Server:                127.0.0.53
Address:                127.0.0.53#53

Non-authoritative answer:
Name:   blog.iesalixar.freedomdns.org
Address: 81.36.184.144
```

Si clonamos el proyecto con git clone en otro PC donde tengamos Docker Engine instalado, podremos probar a generar el certificado y levantar los contenedores. Previamente, es preciso crear el fichero .env con los valores asignados a las variables de entorno. También creamos el fichero dynu-credential.ini con el API Key proporcionado por Dynu y le asignamos permisos (chmod 600) para protegerlo.

Tras esto, lanzamos los servicios con docker compose up -d.

Comprobamos con docker logs certbot cómo se genera el certificado wildcard:

```
mario@mario-PC:~/hardening_apache_nginx_wp$ docker logs certbot
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for iesalixar.freedomdns.org and *.iesalixar.freedomdns.org
Waiting 60 seconds for DNS changes to propagate

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/iesalixar.freedomdns.org/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/iesalixar.freedomdns.org/privkey.pem
This certificate expires on 2025-04-13.
These files will be updated when the certificate renews.
NEXT STEPS:
- The certificate will need to be renewed before it expires. Certbot can automatically
  renew the certificate in the background, but you may need to take steps to enable that function.
  See https://certbot.org/renewal-setup for instructions.
```

Tras esto se ejecuta el post-hook automáticamente, cambiando el grupo propietario y los permisos de los directorios live y archive (g=x) y del fichero de la clave privada (g=r).

```
mario@mario-PC:~/hardening_apache_nginx_wp$ ls -l certs/
total 20
drwx----- 4 root root      4096 ene 13 10:32 accounts
drwx--x--- 3 root messagebus 4096 ene 13 14:46 archive
drwx--x--- 3 root messagebus 4096 ene 13 14:46 live
drwxr-xr-x 2 root root      4096 ene 13 14:46 renewal
drwxr-xr-x 5 root root      4096 ene 13 09:56 renewal-hooks
mario@mario-PC:~/hardening_apache_nginx_wp$ sudo ls -l certs/archive/iesalixar.freedomdns.org
total 16
-rw-r--r-- 1 root root      1326 ene 13 14:46 cert1.pem
-rw-r--r-- 1 root root      1566 ene 13 14:46 chain1.pem
-rw-r--r-- 1 root root      2892 ene 13 14:46 fullchain1.pem
-rw-r----- 1 root messagebus 227 ene 13 14:46 privkey1.pem
mario@mario-PC:~/hardening_apache_nginx_wp$
```

Y vemos el contenido del certificado generado:

```
sudo openssl x509 --noout --text -in certs/archive/iesalixar.freedomdns.org/fullchain1.pem
```

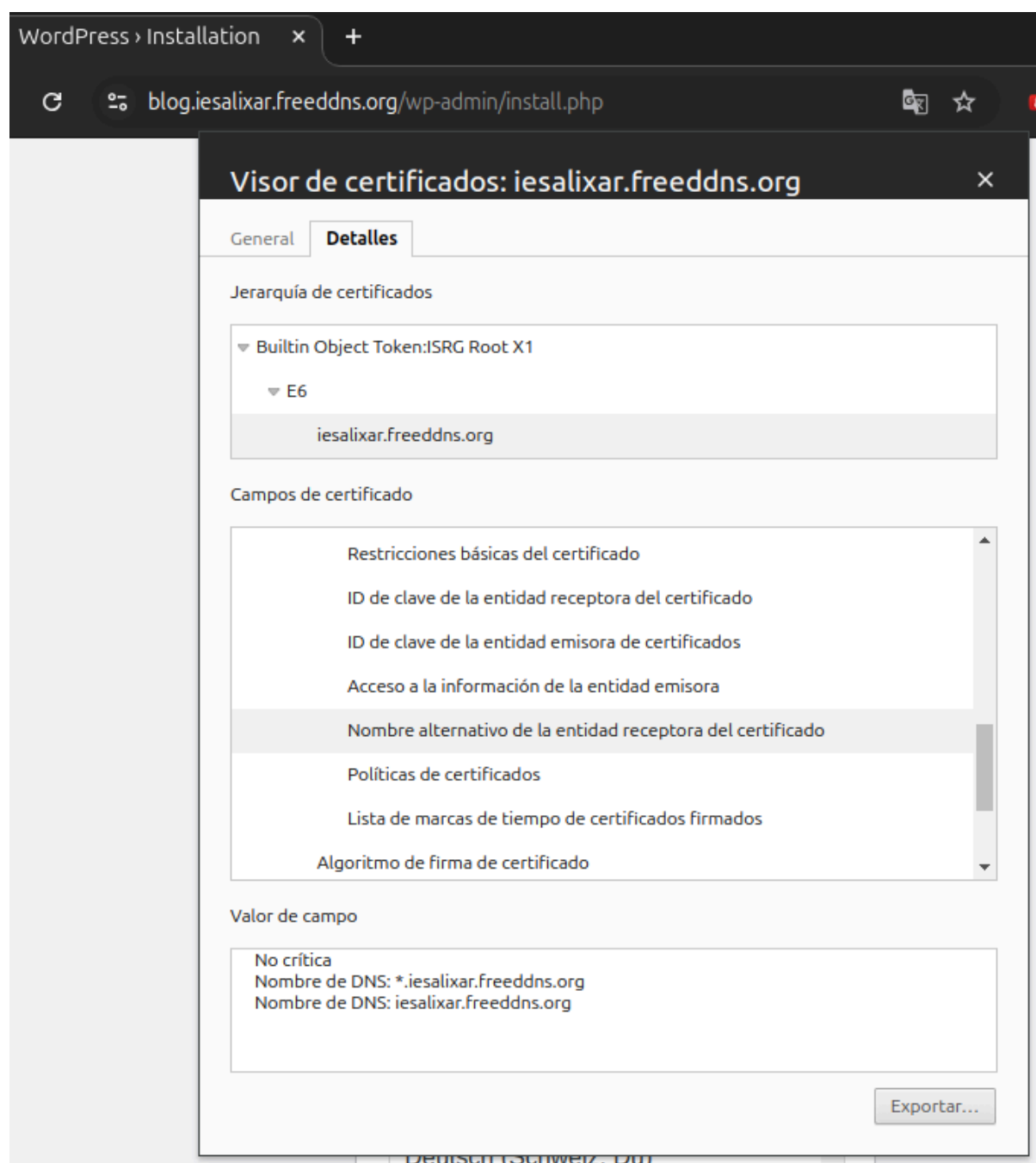
Certificate:

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
    04:25:e2:e8:2e:8d:0a:be:1f:7f:d6:7e:cf:cc:c5:4f:6d:4f
  Signature Algorithm: ecdsa-with-SHA384
  Issuer: C=US, O=Let's Encrypt, CN=E6
  Validity
    Not Before: Jan 13 12:47:57 2025 GMT
    Not After : Apr 13 12:47:56 2025 GMT
  Subject: CN=iesalixar.freedomdns.org
  Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
    Public-Key: (256 bit)
    pub:
      04:9f:68:fd:85:c8:88:bd:ba:07:aa:c2:1e:4f:78:
      f5:20:a4:1b:2a:d2:92:63:f5:1f:e1:b0:5e:ea:15:
      d3:73:8d:8d:74:e3:bd:61:f6:e7:2e:f7:57:10:5a:
      a1:81:cb:0a:0b:10:0c:46:f3:cc:16:6e:6b:f2:5c:
      d1:c2:da:38:42
    ASN1 OID: prime256v1
    NIST CURVE: P-256
  X509v3 extensions:
    X509v3 Key Usage: critical
      Digital Signature
    X509v3 Extended Key Usage:
      TLS Web Server Authentication, TLS Web Client Authentication
    X509v3 Basic Constraints: critical
      CA:FALSE
    X509v3 Subject Key Identifier:
      F4:22:B5:2F:5F:33:C5:9F:79:FD:B0:1C:39:27:53:4B:6F:BD:98:59
    X509v3 Authority Key Identifier:
      93:27:46:98:03:A9:51:68:8E:98:D6:C4:42:48:DB:23:BF:58:94:D2
    Authority Information Access:
      OCSP - URI:http://e6.o.lencr.org
      CA Issuers - URI:http://e6.i.lencr.org/
    X509v3 Subject Alternative Name:
      DNS:*.iesalixar.freedomdns.org, DNS:iesalixar.freedomdns.org
    X509v3 Certificate Policies:
      Policy: 2.23.140.1.2.1
    CT Precertificate SCTs:
      Signed Certificate Timestamp:
```

Para acceder con un navegador desde nuestro propio PC, añadimos las siguientes líneas al fichero /etc/hosts

```
127.0.0.1 iesalixar.freedomdns.org
127.0.0.1 www.iesalixar.freedomdns.org
127.0.0.1 blogiesalixar.freedomdns.org
127.0.0.1 www.blog.iesalixar.freedomdns.org
```

Comprobamos que nos devuelve un certificado válido con el nombre *.iesalixar.freedomdns.org como SAN (Subject Alternative Name).



Gracias al servidor proxy inverso nginx, la seguridad que proporciona la imagen docker del OWASP (owasp/modsecurity-crs:nginx-alpine) se aplica a todos los accesos que se hagan a los servidores web de nuestro sitio.

4.2. Fichero .htaccess

La primera medida a tomar será denegar el acceso al fichero wp-config.php. Este fichero contiene información sensible sobre la conexión a la base de datos, las claves de

seguridad, versiones y la configuración de la URL del sitio. Para ello hay que añadir la siguiente línea al fichero .htaccess situado en el directorio raíz del virtualhost:

```
# BEGIN WordPress
...
# END WordPress

<Files wp-config.php>
    order allow,deny
    deny from all
</files>
```

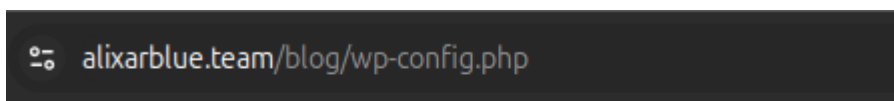
Crea el fichero .htaccess con el contenido anterior abriendo una shell en el contenedor:

```
docker exec -it blog bash
apt update
apt install nano
```

Habría que añadir dicho fichero en la imagen usando un Dockerfile:

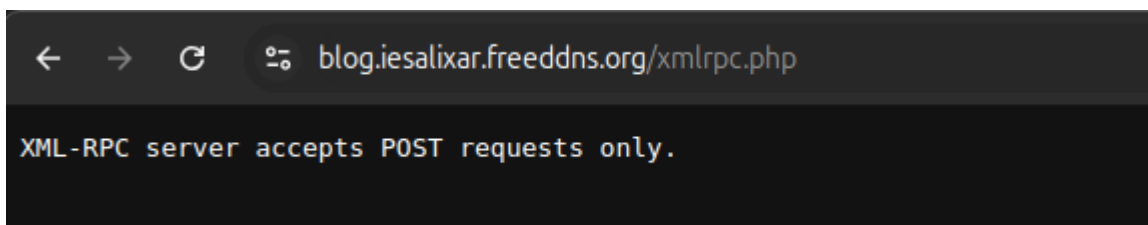
```
FROM WORDPRESS
COPY ./dockerfiles/blog/config/.htaccess
```

Comprueba que el servidor web no permite el acceso al archivo:
<https://blog.iesalixar.org/wp-config.php>



403 Forbidden

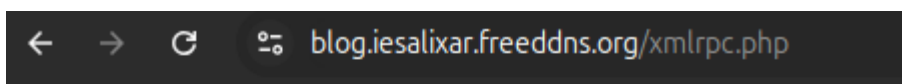
La ejecución de la herramienta WPScan será objeto del taller de Hacking web. La salida que nos proporciona un primer escaneo básico indica que XML-RPC está accesible en <https://blog.iesalixar.org/wp-config.php/xmlrpc.php>. Comprobamos que efectivamente está accesible:



XML-RPC se utiliza para conectar remotamente con WordPress desde aplicaciones de escritorio, móviles, etc. Se mantiene activo por defecto por compatibilidad, lo cual es inseguro ya que puede provocar ataques de denegación de servicio, de fuerza bruta, etc. Para evitarlo, se puede instalar un plugin (disable XML-RPC-API) o evitar acceso al recurso en la configuración del servidor web Apache mediante el fichero .htaccess.

Añade las siguientes líneas en el fichero .htaccess y comprueba que ya no es posible acceder:

```
<Files xmlrpc.php>
  Order Deny,Allow
  Deny from all
</Files>
```



Forbidden

You don't have permission to access this resource.

Apache/2.4.62 (Debian) Server at blog.iesalixar.freedomdns.org Port 80

4.3. Plugins de seguridad

En este apartado vamos a instalar plugins que proporcionan medidas adicionales de seguridad. Dichos plugins suelen tener funcionalidades más avanzadas que no son gratuitas. No obstante, con la funcionalidad básica que aportan conseguimos mejorar la seguridad de nuestro blog.

El siguiente paso consiste en limitar el número de intentos de login en el panel de administración. Mediante la herramienta WPScan disponible en Kali Linux se pueden listar los usuarios de WordPress e incluso obtener sus contraseñas. Para evitarlo, se propone instalar el plugin *Limit Login Attempts Reloaded*.

Accedemos al Dashboard de WordPress, apartado Plugins, Añadir Plugins, lo buscamos y pulsamos el botón Instalar y Activar. Hay que activarlo y a continuación puede modificar los Ajustes de la aplicación. A continuación, modificamos los ajustes de la aplicación: el número de reintentos permitidos (Limit Login Attempts Reloaded) y los minutos por bloqueo.

Bloqueo ?

4 reintentos permitidos ?

20 minutos por bloqueo ?

4 bloqueos incrementan el tiempo de bloqueo a 24 horas ?

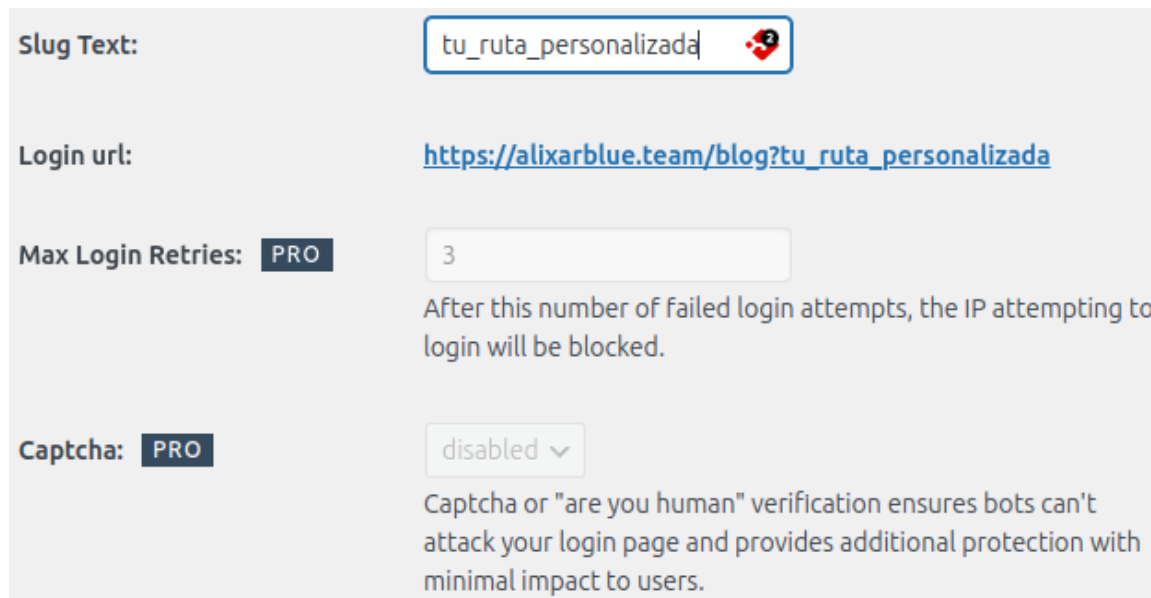
24 horas hasta reiniciar los reintentos ?

Después de que una dirección IP específica no consiga acceder 4 veces, se activa un bloqueo que dura 20 minutos. Si se producen más intentos fallidos en el plazo de 24 horas y provocan otro bloqueo, una vez que su total combinado alcance 4, la duración de 20 minutos se ampliará a 24 horas. El bloqueo se levantará cuando hayan pasado 24 horas desde el último incidente de bloqueo.

Orígenes IP de confianza ?

REMOTE_ADDR

Por otro lado, se recomienda modificar la ruta por defecto de acceso al Dashboard (/wp-admin). Una opción fácil para hacerlo es instalando un plugin, como por ejemplo *Easy Hide Login*. Utiliza este plugin para modificar la ruta, cierra sesión y prueba a hacer login accediendo con la nueva ruta. Comprueba que wp-admin ya no está disponible.



The screenshot shows the configuration page for the 'Easy Hide Login' plugin. It features several settings: 'Slug Text' is set to 'tu_ruta_personalizada'; 'Login url' is a custom URL 'https://alixarblue.team/blog?tu_ruta_personalizada'; 'Max Login Retries' is set to 3 with a 'PRO' label; and 'Captcha' is set to 'disabled' with a 'PRO' label. Each setting has a brief description of its function.

| | |
|------------------------|--|
| Slug Text: | tu_ruta_personalizada |
| Login url: | https://alixarblue.team/blog?tu_ruta_personalizada |
| Max Login Retries: PRO | 3 After this number of failed login attempts, the IP attempting to login will be blocked. |
| Captcha: PRO | disabled Captcha or "are you human" verification ensures bots can't attack your login page and provides additional protection with minimal impact to users. |

Otro plugin que hemos probado es [WordFence security Firewall & Malware Scan](#), para el cual se puede solicitar una licencia gratuita que permite el uso de una parte de sus funciones. Se recomienda activar inicialmente el modo aprendizaje, durante el cual no protegerá, para evitar que dé falsos positivos y sea más efectivo en el bloqueo de accesos indebidos.

Al instalarlo, ofrece una configuración por defecto para el fichero .htaccess, añadiendo nuevas líneas al activarlo.

```
# Wordfence WAF
<IfModule mod_php5.c>
php_value auto_prepend_file '/var/www/html/wordfence-waf.php'
</IfModule>
<IfModule mod_php7.c>
php_value auto_prepend_file '/var/www/html/wordfence-waf.php'
</IfModule>
<IfModule mod_php.c>
php_value auto_prepend_file '/var/www/html/wordfence-waf.php'
</IfModule>
<Files ".user.ini">
<IfModule mod_authz_core.c>
Require all denied
</IfModule>
<IfModule !mod_authz_core.c>
Order deny,allow
Deny from all
# END Wordfence WAF
```

Tiene muchas opciones, entre las que cabe destacar:

- WAF (Web Application Firewall) con un listado de reglas base en la versión Community para ataques SQLi, XSS, etc. La versión premium añade reglas de cortafuegos y la lista de bloqueo de IPs en tiempo real de Wordfence.

Cortafuegos

[Más información sobre el cortafuegos](#)



El cortafuegos de Wordfence está activado

La protección Premium está desactivada

Como usuario gratuito de Wordfence, actualmente estás utilizando la versión de la comunidad del feed de defensa contra amenazas. Los usuarios premium están protegidos por reglas de cortafuegos y firmas de malware adicionales. Actualiza hoy a Premium para mejorar tu protección.

[ACTUALIZAR A LA VERSIÓN PREMIUM](#)

[APRENDE MÁS](#)












Cortafuegos de la aplicación web
Detiene los ataques complejos
[Gestionar WAF](#)



Reglas del cortafuegos: Comunidad
La regla se actualiza con un retraso de 30 días
[Actualizar a la versión premium](#)

Reglas

| | Categoría | Descripción |
|---|-------------|--|
|  | whitelist | Whitelisted URL |
|  | lfi | Slider Revolution <= 4.1.4 - Directory Traversal |
|  | sqli | SQL Injection |
|  | xss | XSS: Cross Site Scripting |
|  | file_upload | Malicious File Upload |
|  | traversal | Directory Traversal |
|  | lfi | LFI: Local File Inclusion |
|  | xxe | XXE: External Entity Expansion |
|  | xss | DZS Video Gallery <= 8.60 - Reflected Cross-Site Scripting |

- Bloqueo tras varios intentos fallidos de login, reportando las direcciones IP que han intentado acceder. Añade opciones para forzar contraseñas seguras, para evitar contraseñas filtradas.

Protección contra ataques de fuerza bruta

Activa la protección de fuerza bruta ?

Esta opción activa todas las opciones de «protección contra fuerza bruta», incluida la aplicación estricta de contraseñas y la limitación de acceso no válido. Puedes modificar las opciones individuales a continuación.

OFF

ON

Bloquear después de cuántos fallos de acceso ?

20

Bloquear después de cuántos intentos de contraseña olvidada ?

20

Cuenta los errores durante qué período de tiempo ?

4 horas

Cantidad de tiempo que un usuario está bloqueado ?

4 horas

☐

Bloquear inmediatamente los nombres de usuario no válidos ?

Bloquear inmediatamente la IP de los usuarios que intentan acceder con estos nombres de usuario ?

Presiona enter para añadir un nombre de usuario

Requerir 2FA para la identificación de llamadas XML-RPC

Si está activado, las llamadas XML-RPC que requieren identificación también requerirán que se añada un código 2FA válido a la contraseña. Debes elegir la opción «omitido» si usas la aplicación de WordPress, el plugin Jetpack u otros servicios que requieren XML-RPC.

OMITIDO

OBLIGATORIO

Desactivar la identificación XML-RPC

☒

Si está desactivado las peticiones XML-RPC que traten de identificarse serán rechazadas, tenga el usuario activado 2FA o no.

- Doble factor de autenticación en el acceso.

Identificación de dos factores

[Aprende más sobre autenticación en dos factores](#)

La identificación de dos factores, o 2FA, mejora significativamente la seguridad en el acceso a tu web. La 2FA de Wordfence funciona con aplicaciones de primer nivel como Google Authenticator, FreeOTP y Authy. Para una lista de aplicaciones de primer orden probadas [haz clic aquí](#).

Editando usuario: mario (tú)

Wordfence 2FA activo

La identificación de dos factores de Wordfence está actualmente activa en tu cuenta. Puedes desactivarlo haciendo clic en el botón de abajo.


DESACTIVAR


Recuperación de códigos

Quedan 5 códigos de recuperación sin usar. Puedes generar un nuevo conjunto haciendo clic a continuación.

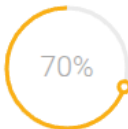
GENERAR CÓDIGOS NUEVOS

- Escaneos para buscar vulnerabilidades y malware. La versión premium añade firmas de malware adicionales.

 **Opciones de exploración y planificación** [Aprende más sobre la exploración](#)




Tipo de exploración: Estándar
Capacidad de detección estándar



Firmas de malware: Comunidad
Actualizaciones de firmas demoradas 30 días

Programando exploración

Programar exploraciones de Wordfence 

DESACTIVADO **ACTIVADO**

☒ Permite a Wordfence elegir cuándo explorar mi sitio (recomendado)

☐ Programar las exploraciones manualmente **Característica premium**

Opciones del tipo de exploración básica

☐ Exploración limitada

☒ Exploración estándar

☐ Alta sensibilidad

☐ Análisis personalizado

5. SEGURIDAD DE LA RED

5.1. Reglas del firewall

A nivel de firewall, en la interfaz WAN sólo permitimos las peticiones entrantes por los puertos TCP 80 (HTTP) y TCP 443 (HTTPS).

ante

WAN

LAN

CIBER

1SMR

DMZ

SERVIDORES

Reglas (arrastrar para cambiar el orden)

| | Departamento | Protocolo | Fuente | Puerto | Destino | Puerto | Puerta de enlace | Cola | Horario | Descripción |
|---|--------------|-----------|------------------------------------|--------|---------|-------------|------------------|---------|---------|--|
| ✗ | 0/16,94 MiB | * | Reservados No asignado por IANA | * | * | * | * | * | | Bloquear redes bogon (IP FALSAS) |
| ✓ | 1/139,81 MiB | IPv4 TCP | * | * | Nginx | 80 (HTTP) | * | ninguno | | NAT Port forward HTTP a proxy web inverso |
| ✓ | 1/1,31 GiB | IPv4 TCP | * | * | Nginx | 443 (HTTPS) | * | ninguno | | NAT Port forward HTTPS a proxy web inverso |

El cortafuegos pfSense se encarga de hacer NAT y reenviar al proxy web inverso las peticiones recibidas en la interfaz WAN (IP pública).

| Reglas / NAT / Redirección de puerto | | | | | | | | | | |
|--------------------------------------|-----------|---------------------|-----------------------|-----------------|---------------|--------|-----------------|--|--|--|
| de puerto | 1:1 | Saliente | NPT 27 | | | | | | | |
| Interfaz | Protocolo | Dirección de Origen | Los puertos de origen | Dest. Dirección | Dest. puertos | NAT IP | Los puertos NAT | Descripción | | |
| WAN | TCP | * | * | WAN address | 443 (HTTPS) | Nginx | 443 (HTTPS) | Port forward HTTPS a proxy web inverso | | |
| WAN | TCP | * | * | WAN address | 80 (HTTP) | Nginx | 80 (HTTP) | Port forward HTTP a proxy web inverso | | |

El servidor Nginx se debe encontrar en una DMZ separado del servidor web. Sólo permitimos pasar el tráfico HTTPS desde Nginx al servidor Wordpress. En pfSense se utilizan nombres de alias en las reglas para hacer referencia a IPs, de modo que se pueda cambiar la IP del alias sin tener que modificar la regla.

ante

WAN

LAN

CIBER

1SMR

DMZ

SERVIDORES

reglas (arrastrar para cambiar el orden)

| | Departamento | Protocolo | Fuente | Puerto | Destino | Puerto | Puerta de enlace | Cola | Horario | Descripción |
|--|--------------|---------------|-------------|--------|-------------|-------------|------------------|---------|---------|---|
| | 0/0 B | IPv4 * | * | * | pfB_PRI1_v4 | * | * | ninguno | | pfB_PRI1_v4 |
| | 0/13 KiB | IPv4 TCP | Nginx | * | Wordpress | 443 (HTTPS) | * | ninguno | | Reenvío peticiones HTTPS a Wordpress en red LAN |
| | 0/840 B | IPv4 ICMP any | DMZ subnets | * | DMZ address | * | * | ninguno | | Ping al firewall desde red DMZ |
| | 0/548 KiB | IPv4 TCP/UDP | DMZ subnets | * | DMZ address | 53 (DNS) | * | ninguno | | Peticiones a DNS server en Pfsense |

De la misma manera, el servidor web se encuentra en una red distinta al servidor de base de datos, el cual está en una red más aislada.

ante

WAN

LAN

CIBER

1SMR

DMZ

SERVIDORES

reglas (arrastrar para cambiar el orden)

| | Departamento | Protocolo | Fuente | Puerto | Destino | Puerto | Puerta de enlace | Cola | Horario | Descripción |
|---|--------------|-----------|-----------|--------|-------------|-----------|------------------|---------|---------|---|
| ✓ | 0/29 KiB | * | * | * | LAN Address | 443 80 22 | * | * | | Regla anti-bloqueo |
| 👉 | 0/0 B | IPv4 * | * | * | pfB_PRI1_v4 | * | * | ninguno | | pfB_PRI1_v4 |
| ✓ | 0/1,97 GiB | IPv4 TCP | Wordpress | * | MariaDB | 3306 | * | ninguno | | Consultas SQL desde Wordpress a MariaDB |





Se debe usar un puerto distinto al de por defecto para la conexión al servidor de BD. En el siguiente ejemplo se ha configurado el puerto TCP 3310 para la escucha de conexiones entrantes al servidor MariaDB:

| | | | | | | | | | |
|---|--------------|----------|-----------------|---|-----------|------|---|---------|--|
| ✓ | 0/206,89 MiB | IPv4 TCP | dpladia | * | dpladiaBD | 3310 | * | ninguno | Apache hacia SERVIDORES (Base Datos) |
| ✓ | 0/118,29 MiB | IPv4 TCP | acasmrWebServer | * | acasmrBD | 3310 | * | ninguno | Conexión a BBDD desde Server Web acasmr |
| ✓ | 0/33,60 MiB | IPv4 TCP | jgonalb | * | jgonalbBD | 3310 | * | ninguno | Conexión a BD desde servidor web jgonalb |










5.2. pfBlockerNG

Es un complemento de pfSense de código abierto que permite bloquear tráfico de red no deseado tanto de entrada como de salida, además de la resolución nombres de dominio para una navegación segura. Se instala en la opción System > Package Manager > Available Packages.





















- Filtra direcciones IP específicas en base a listas de bloqueo públicas de direcciones que han sido reportadas como spammers o bots maliciosos, tales como Spamhaus, Emerging Threats, AbuseIP. También es capaz de bloquear o restringir tráfico de países específicos mediante filtros de geolocalización (GeoIP).

| Categoría | | Alias/Group | Feed/Website | Header/URL |
|---------------|---|-------------|---|----------------|
| IPv4 Category |   | PRI1 | Abuse Feodo Tracker | Abuse_Feodo_C2 |
| IPv4 | | PRI1 | Abuse SSL Blacklist | Abuse_SSLBL |
| IPv4 | | PRI1 | CINS Army | CINS_army |
| IPv4 | | PRI1 | Emerging Threats | ET_Block |
| IPv4 | | PRI1 | Emerging Threats | ET_Comp |
| IPv4 | | PRI1 | Internet Storm Center | ISC_Block |
| IPv4 | | PRI1 | Pulsedive   | Pulsedive |
| IPv4 | | PRI1 | Spamhaus | Spamhaus_Drop |
| IPv4 | | PRI1 | Spamhaus | Spamhaus_eDrop |
| IPv4 | | PRI1 | Talos-Snort | Talos_BL |







- DNSBL: Si se configura el firewall como servidor DNS, se puede bloquear la navegación a dominios reportados como maliciosos, de spam o de phishing utilizando filtros basados en DNS.

| | | | | |
|-------|---|-----------|--|---------------|
| DNSBL |   | Malicious | Dan Pollock | SWC |
| DNSBL | | Malicious | Disconnect.Me | D_Me_Malv |
| DNSBL | | Malicious | Disconnect.Me | D_Me_Malw |
| DNSBL | | Malicious | Maltrail | Maltrail_BD |
| DNSBL | | Malicious | MVPS Hosts | MVPS |
| DNSBL | | Malicious | Spam404 | Spam404 |
| DNSBL | | Malicious | Stop Forum Spam | SFS_Toxic_BD |
| DNSBL |   | Phishing | Abuse URLhaus | Abuse_urlhaus |
| DNSBL | | Phishing | Bambenek Consulting   | BBC_DC2 |
| DNSBL | | Phishing | Internet Storm Center   | ISC_SDH |
| DNSBL | | Phishing | Malware Patrol  | MPatrol |
| DNSBL | | Phishing | OpenPhish | OpenPhish |
| DNSBL | | Phishing | PhishTank | PhishTank |

➤ Muestra un reporte de los dominios que se han bloqueado a nivel de DNS:























| Top Blocked Domain | | | |
|---|--------|-------|--------------------------------|
| | Cuenta | Tipo | Found [87] Blocked Domain(s) |
|   | 17654 | DNSBL | incoming.telemetry.mozilla.org |
|   | 565 | DNSBL | browser-intake-datadoghq.com |
|   | 292 | DNSBL | googleads.g.doubleclick.net |
|   | 212 | DNSBL | www.googletagmanager.com |
|   | 204 | DNSBL | ogads-pa.googleapis.com |
|   | 108 | DNSBL | static.doubleclick.net |
|   | 88 | DNSBL | beacons.gcp.gvt2.com |
|   | 68 | DNSBL | static.cloudflareinsights.com |
|   | 59 | DNSBL | beacons.gvt2.com |
|   | 53 | DNSBL | beacons2.gvt2.com |

➤ Y de las IPs que se han bloqueado

| Bloquear - Last 25 Alert Entries | | | | | | | |
|----------------------------------|-----|-----------------------------|--------|---------------------------|---|--------|---|
| Fecha | SI | Regla | Proto | Fuente | Destino | GeolIP | |
| Dec 10 17:35:03 | DMZ | pfB_PRI1_v4 (1770009636) | TCP-SA | 192.168.2.2:80 Unknown |  103.91.209.215:6180 Unknown | Unk |  |
| Dec 10 16:18:23 | DMZ | pfB_PRI1_v4 (1770009636) | TCP-SA | 192.168.2.2:80 Unknown |  103.91.209.182:6180 Unknown | Unk | |
| Dec 10 15:38:56 | DMZ | pfB_PRI1_v4 (1770009636) | TCP-SA | 192.168.2.2:80 Unknown |  103.91.211.56:6180 Unknown | Unk | |
| Dec 10 11:02:19 | DMZ | pfB_PRI1_v4 (1770009636) | TCP-SA | 192.168.2.2:80 Unknown |  103.91.209.215:6180 Unknown | Unk | |
| Dec 10 09:11:30 | DMZ | pfB_PRI1_v4 (1770009636) | TCP-SA | 192.168.2.2:80 Unknown |  111.180.204.170:6180 Unknown | Unk | |

5.3. Suricata IPS

Al integrarse con pfSense, Suricata puede configurarse como un IPS (Intrusion Prevention System), lo que le permite no solo detectar, sino también bloquear ataques en tiempo real. Es capaz de analizar el tráfico a nivel de HTTP, HTTPS, DNS, SMB, etc,

| | | | | | | | |
|---|---|--|---|---|-------|--|---|
| TCP | Generic Protocol Command Decode | 192.168.1.77   | 80 | 51.15.19.173     | 50476 | 1:2221010   | SURICATA HTTP unable to match response to request |
| TCP | Generic Protocol Command Decode | 167.94.138.53     | 56552 | 192.168.1.77   | 80 | 1:2290006   | SURICATA HTTP2 too long frame data |
| 162.142.125.94 |   | 01/15/2025 02:25:49 | SURICATA IKE invalid proposal | | | | 1:2224009 |
| 152.32.128.204 |   | 01/15/2025 03:56:32 | SURICATA Applayer Mismatch protocol both directions | | | | 1:2260000 |
| 147.185.132.100 |   | 01/15/2025 04:02:32 | SURICATA Applayer Mismatch protocol both directions | | | | 1:2260000 |
| 14 host IP addresses are currently being blocked. | | | | | | | |

Y bloquear tráfico no deseado o sospechoso según reglas públicas que se cargan como Emerging Threats y Snort VRT.

INSTALLED RULE SET MD5 SIGNATURES

| Rule Set Name/Publisher | MD5 Signature Hash | MD5 Signature Date |
|----------------------------------|----------------------------------|-----------------------------------|
| Emerging Threats Open Rules | 76019016871bb0f85282aedfc487365e | Wednesday, 15-Jan-25 00:15:28 CET |
| Snort Subscriber Rules | Not Downloaded | Not Downloaded |
| Snort GPLv2 Community Rules | 59111fa59c373415af35abd9968740cf | Wednesday, 15-Jan-25 00:15:28 CET |
| Feodo Tracker Botnet C2 IP Rules | f866e69bc63e87a3c013738068052bd7 | Wednesday, 15-Jan-25 00:15:27 CET |
| ABUSE.ch SSL Blacklist Rules | 02cd63325b5b18b817ef3589bb202314 | Wednesday, 15-Jan-25 00:15:27 CET |

La funcionalidad de IDS y registro de alertas se estudiará en el taller de Incidentes junto con la integración de Suricata con herramientas externas para su análisis avanzado como el SIM ELK Stack (ElasticSearch, Logstash, Kibana).

Se pueden producir falsos positivos cuando una actividad legítima es identificada erróneamente como maliciosa. Por tanto, puede ocurrir que el IPS bloquee tráfico importante y causar interrupciones del servicio. En este caso será preciso analizar las alertas y desactivar alguna de las reglas concretas. Adicionalmente, se pueden definir whitelists para excluir del bloqueo a máquinas internas o a sitios legítimos para descarga de software.

Algunos ejemplos de falsas alarmas que hemos tenido en nuestro entorno:

- Bloqueo de apt update y apt install de las VMs Ubuntu. No se podían actualizar los repositorios de Ubuntu ni instalar paquetes mediante apt porque había bloqueado IPs públicas de destino relativas a la instalación de software de Ubuntu.

| | | | | | | | | | | |
|------------------------|--|---|-----|------------------------|------------------|-------|------------------|----|---------------|--|
| 11/24/2024 16:11:51 | | 3 | TCP | Not Suspicious Traffic | 192.168.1.77 | 19515 | 91.189.91.81 | 80 | 1:2013504 | ET INFO GNU/Linux APT User-Agent Outbound likely related to package management |
| 11/24/2024 16:11:51 | | 3 | TCP | Not Suspicious Traffic | 192.168.1.77 | 19515 | 91.189.91.81 | 80 | 1:2013504 | ET INFO GNU/Linux APT User-Agent Outbound likely related to package management |
| 11/24/2024 16:11:51 | | 3 | TCP | Not Suspicious Traffic | 192.168.1.77 | 25673 | 91.189.91.81 | 80 | 1:2013504 | ET INFO GNU/Linux APT User-Agent Outbound likely related to package management |

- Bloqueo de pip install: ET INFO Observed File Hosting Service Domain (files.pythonhosted.org in TLS SNI) This site hosts packages and documentation uploaded by authors of packages on the Python Package Index

| | | | | | | | | | | |
|------------------------|--|---|-----|---------------|------------------|-------|---------------------|-----|---------------|--|
| 11/25/2024 18:03:47 | | 3 | UDP | Misc activity | 192.168.1.77 | 14370 | 1.1.1.1 | 53 | 1:2049201 | ET FILE_SHARIN File Hosting Service Domain Domain in DNS Lookup (files.pythonhosted.org) |
| 11/25/2024 18:03:46 | | 3 | TCP | Misc activity | 192.168.1.77 | 43034 | 151.101.128.223 | 443 | 1:2049202 | ET INFO Observ File Hosting Service Domain (files.pythonhosted.org in TLS SNI) |

6. ENLACES DE REFERENCIA

Docker Seguro:

[Best Practices for Running Docker Containers with Non-Root Users](#)
[Running Docker application containers more securely](#)

Hardening de Apache:

[INCIBE: Hardening básico de Apache](#)
[Apache Web Server and Security Guide](#)
[18 Apache Web Server Security and Hardening Tips](#)
[Apache Web Server Hardening](#)
[Securing apache2 with ModSecurity](#)
[Instalación y configuración de Mod_Security en Apache](#)
[Instalación y configuración de Mod_Evasive en Apache](#)
[How to Install and Configure ModEvasive with Apache on Ubuntu](#)

Let's Encrypt:

[Dynu DNS: How to generate Let's Encrypt certificates](#)
[Certbot docker](#)
[Certbot DNS plugins](#)

Wordpress:

[How to install WordPress on Ubuntu using Docker](#)

Pfsense + Suricata + pgBlockerNG:

[Implementando Pfsense con Suricata y pfBlockerNG](#)

7. REPOSITORIOS DEL TALLER

En el taller se ejecutarán servidores web en contenedores a partir de imágenes con las medidas de seguridad ya configuradas y se describirán las pruebas necesarias para comprobar que esas medidas funcionan. Se podrán realizar tres niveles de taller según su complejidad:

Nivel 1: Hardening de un servidor web Apache con ModSecurity junto con certbot para la obtención de un certificado Let's Encrypt.

https://github.com/marioalfonsolasso/hardening_apache_ssl.git

Nivel 2: Hardening del servidor web anterior añadiendo un proxy inverso nginx con el WAF ModSecurity.

https://github.com/marioalfonsolasso/hardening_apache_nginx.git

Nivel 3: Hardening de un CMS Wordpress accesible a través del proxy inverso nginx.

https://github.com/marioalfonsolasso/hardening_apache_nginx_wp.git