

IES El Caminás

Ciclo de Especialización en Ciberseguridad

Módulo: Puesta en Producción Segura

Título del trabajo: ACT_RA3_1_JAVLLUAPA_CIBERSEGURIDAD

Autor(es): Javier Lluesma Aparici

Profesor: Miguel López Novella

Fecha: 14/01/2025

Versión: 1.0

Índice de Contenidos

Índice de Contenidos	1
1. Introducción	2
2. Desarrollo de los contenidos.....	3
Práctica 1	3
Dockerfile - Práctica 1.....	6
Práctica 2 Integrar un WAF (ModSecurity)	8
Dockerfile - Práctica 2.....	10
Dockerfile - Práctica 3.....	12
Dockerfile - Práctica 4.....	16
Práctica Certificados (Dockerfile).....	20
Práctica Gold Image (Dockerfile).....	23
Estrategia de Capas (Dockerfile Versión Final de Cada Práctica).....	27
3. Bibliografía y otras fuentes consultadas.....	30
4. Conclusión	30

1. Introducción

El presente proyecto documenta el proceso de diseño, implementación y despliegue de una infraestructura web endurecida (Hardened Web Server) basada en el servidor Apache. La metodología empleada se fundamenta en el concepto de seguridad en capas (Defense in Depth), utilizando la tecnología de contenedores Docker para garantizar la portabilidad y la inmutabilidad de la configuración.

El objetivo principal ha sido transformar una instalación estándar de Apache en una Golden Image de alta seguridad. Para ello, se ha seguido una evolución incremental en la que cada etapa añadía una nueva barrera de defensa: desde la gestión de certificados SSL/TLS y la configuración de cabeceras de seguridad (HSTS, CSP), hasta la implementación de un Firewall de Aplicación Web (WAF) con reglas OWASP y sistemas de mitigación de ataques de Denegación de Servicio (DoS). El resultado final es un entorno robusto capaz de mitigar las vulnerabilidades más críticas identificadas en el sector.

2. Desarrollo de los contenidos

Práctica 1

Antes de proceder con la creación del dockerfile, se muestra cómo se realizaría la configuración de esta práctica en un servidor apache virtual corriente.

Lo primero que se debe hacer es deshabilitar mod_autoindex para ello debemos ejecutar a2dismod que permite deshabilitar módulos y el módulo a desactivar, seguido a esto reiniciaremos apache para aplicar los cambios.

```
vmate@vera-umate:~$ sudo a2dismod autoindex
WARNING: The following essential module will be disabled.
This might result in unexpected behavior and should NOT be done
unless you know exactly what you are doing!
    autoindex

To continue type in the phrase 'Yes, do as I say!' or retry by passing '-f': Yes, do as I say!
Module autoindex disabled.
To activate the new configuration, you need to run:
    systemctl restart apache2
vmate@vera-umate:~$ sudo systemctl restart apache2
vmate@vera-umate:~$
```

Imagen 1. Deshabilitar mod_autoindex.

A continuación, se listan los módulos cargados en apache para ver que el anterior ha sido deshabilitado correctamente.

AUDITORÍA TI NOVATECH

```
vmate@vera-umate:~$ sudo apache2ctl -t -D DUMP_MODULES
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
Loaded Modules:
 core_module (static)
 so_module (static)
 watchdog_module (static)
 http_module (static)
 log_config_module (static)
 logio_module (static)
 version_module (static)
 unixd_module (static)
 access_compat_module (shared)
 alias_module (shared)
 auth_basic_module (shared)
 authn_core_module (shared)
 authn_file_module (shared)
 authz_core_module (shared)
 authz_host_module (shared)
 authz_user_module (shared)
 deflate_module (shared)
 dir_module (shared)
 env_module (shared)
 filter_module (shared)
 mime_module (shared)
 mpm_event_module (shared)
 negotiation_module (shared)
 reqtimeout_module (shared)
 setenvif_module (shared)
 status_module (shared)
```

Imagen 2. Módulos cargados en servidor apache.

Ahora procederemos a ocultar la versión de apache para ello, accederemos al archivo de configuración y añadiremos las líneas que se visualizan en la siguiente imagen:

```
vmate@vera-umate:~$ sudo nano /etc/apache2/apache2.conf |
```

Imagen 3. Acceder al fichero de configuración.

```
ServerTokens ProductOnly
ServerSignature Off
```

Imagen 4. Ocultar versión de apache.

Tras esto reiniciaremos el servicio para aplicar los cambios efectuados.

```
vmate@vera-umate:~$ sudo systemctl restart apache2.service
vmate@vera-umate:~$
```

Imagen 5. Reiniciar servicio de apache.

Ahora vamos a habilitar el módulo headers (obligatorio para HSTS y CSP).

```
vmate@vera-umate:~$ sudo a2enmod headers
Enabling module headers.
To activate the new configuration, you need to run:
  systemctl restart apache2
vmate@vera-umate:~$ sudo systemctl restart apache2
```

Imagen 6. Habilitar módulo y aplicar cambios.

AUDITORÍA TI NOVATECH

Tras esto accederemos al fichero VirtualHost SSL predefinido, y dentro de <VirtualHost *:443>, añadiremos la línea que se visualiza en las siguientes imágenes y aplicaremos cambios.

```
vmate@vera-umate:~$ sudo nano /etc/apache2/sites-available/default-ssl.conf
```

Imagen 7. Acceder al fichero de configuración.

```
Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains"
```

Imagen 8. Configuración de HSTS.

```
vmate@vera-umate:~$ sudo systemctl restart apache2
```

Imagen 9. Aplicar cambios.

Tras esto habilitamos el módulo ssl ya que son necesarios los certificados SSL para la práctica y también la configuración realizada para que se aplique en el sitio web. Además se reinicia apache para aplicar los cambios.

```
vmate@vera-umate:~$ sudo a2enmod ssl
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
vmate@vera-umate:~$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
    systemctl reload apache2
vmate@vera-umate:~$ sudo systemctl restart apache2.service
vmate@vera-umate:~$
```

Imagen 10. Habilitar configuración.

Finalmente, volvemos al VirtualHost y configuramos CSP añadiendo la siguiente línea:

```
Header set Content-Security-Policy "default-src 'self'; img-src *; script-src 'self'"
```

Imagen 11. Configurar CSP.

Tras esto, solo queda comprobar que el resultado es el esperado, como se puede ver en la siguiente imagen al comprobar el trabajo realizado, no aparece la versión de apache que ha sido ocultada y se ven las líneas Strict-Transport-Security y Content-Security-Policy que hacen referencia a la configuración de HSTS y CSP.

```
vmate@vera-umate:~$ curl -k -I https://localhost
HTTP/1.1 200 OK
Date: Wed, 14 Jan 2026 21:04:56 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubDomains
Last-Modified: Wed, 14 Jan 2026 20:25:00 GMT
ETag: "29af-6485ee7081c64"
Accept-Ranges: bytes
Content-Length: 10671
Vary: Accept-Encoding
Content-Security-Policy: default-src 'self'; img-src *; script-src 'self'
Content-Type: text/html
```

Imagen 12. Verificación de funcionalidad de configuración.

Dockerfile - Práctica 1

Procedemos ahora con la creación del Dockerfile y otros archivos necesarios, ejecución del contenedor con la imagen creada y pruebas para verificar el funcionamiento.

En primer lugar, vamos a crear un archivo que llamaremos csp_hsts.conf y que servirá para configurar las cabeceras de seguridad necesarias, sin ensuciar el dockerfile, haciéndolo más claro y entendible.

Forzaremos además la desactivación explícita de opciones de directorio.



```
root@vera-umate:/home/vmate/RA3/p1
GNU nano 7.2                                     csp_hsts.conf
# Ocultar versión de servidor (Forzado)
ServerTokens ProductOnly
ServerSignature Off

<IfModule mod_headers.c>
    Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains"
    Header set Content-Security-Policy "default-src 'self'; img-src *; media-src media1.com media2.com; script-src userscripts.example"
</IfModule>

# Bloqueo extra para el listado de directorios
<Directory /var/www/html>
    Options -Indexes
</Directory>
```

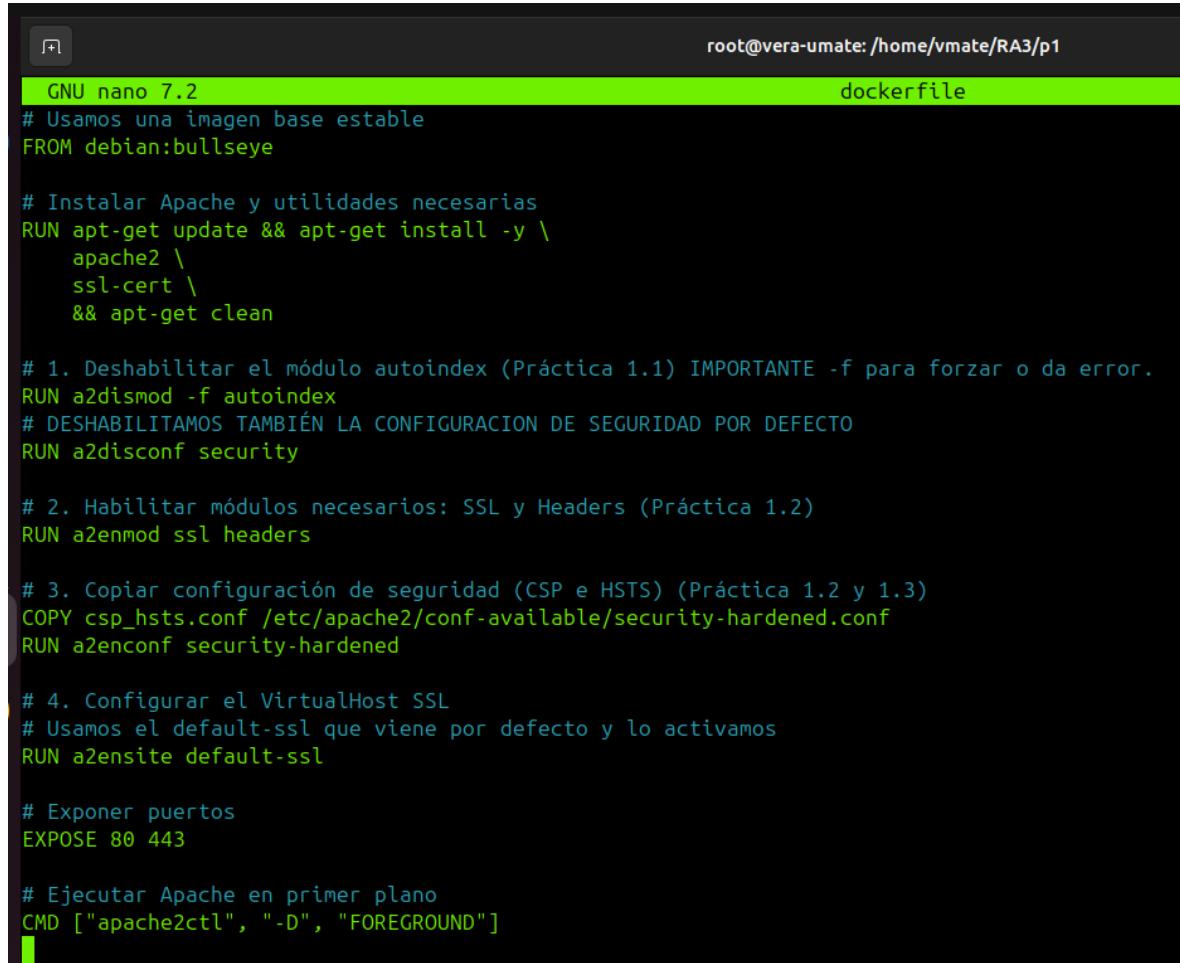
Imagen 13. Contenido del fichero csp_hsts.conf.

A continuación, crearemos el dockerfile, en el que sencillamente debemos ejecutar paso a paso los comandos vistos en la primera fase utilizando la sintaxis docker.

Los únicos respectivos cambios son:

- Forzaremos el deshabilitado de autoindex para que no nos de error ya que necesita doble verificación.
- Deshabilitamos también la configuración de seguridad por defecto para aplicar nuestro fichero de configuración personalizado.
- Copiaremos la configuración de seguridad creada al directorio de configuraciones disponibles de apache y habilitaremos el nuevo de fichero de configuración.
- Expondremos los puertos 80 y 443.

- Ejecutaremos Apache en primer plano.



The screenshot shows a terminal window with a dark background and light-colored text. At the top, it says "root@vera-umate: /home/vmate/RA3/p1". Below that, the title bar reads "GNU nano 7.2" and "dockerfile". The main area contains the Dockerfile code:

```
GNU nano 7.2
root@vera-umate: /home/vmate/RA3/p1
dockerfile

# Usamos una imagen base estable
FROM debian:bullseye

# Instalar Apache y utilidades necesarias
RUN apt-get update && apt-get install -y \
    apache2 \
    ssl-cert \
    && apt-get clean

# 1. Deshabilitar el módulo autoindex (Práctica 1.1) IMPORTANTE -f para forzar o da error.
RUN a2dismod -f autoindex
# DESHABILITAMOS TAMBIÉN LA CONFIGURACION DE SEGURIDAD POR DEFECTO
RUN a2disconf security

# 2. Habilitar módulos necesarios: SSL y Headers (Práctica 1.2)
RUN a2enmod ssl headers

# 3. Copiar configuración de seguridad (CSP e HSTS) (Práctica 1.2 y 1.3)
COPY csp_hsts.conf /etc/apache2/conf-available/security-hardened.conf
RUN a2enconf security-hardened

# 4. Configurar el VirtualHost SSL
# Usamos el default-ssl que viene por defecto y lo activamos
RUN a2ensite default-ssl

# Exponer puertos
EXPOSE 80 443

# Ejecutar Apache en primer plano
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Imagen 13. Contenido del dockerfile.

Tras esto procedemos a la construcción de la imagen a partir del fichero dockerfile creado y los ficheros adjuntos.

AUDITORÍA TI NOVATECH

```
root@vera-umate:/home/vmate/RA3/p1# docker build -t apache-p1 .
[+] Building 3.3s (13/13) FINISHED                                            docker:default
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 937B
=> [internal] load metadata for docker.io/library/debian:bullseye
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/8] FROM docker.io/library/debian:bullseye@sha256:8a08b2875ed17adc464ae03cf5f8652a41821fb8d3d2b17923f11e7489b146da 0.0s
=> => resolve docker.io/library/debian:bullseye@sha256:8a08b2875ed17adc464ae03cf5f8652a41821fb8d3d2b17923f11e7489b146da 0.0s
=> [internal] load build context
=> => transferring context: 500B
=> = CACHED [2/8] RUN apt-get update && apt-get install -y      apache2      ssl-cert      && apt-get clean 0.0s
=> = CACHED [3/8] RUN a2dismod -f autoindex 0.0s
=> [4/8] RUN a2disconf security 0.4s
=> [5/8] RUN a2enmod ssl headers 0.3s
=> [6/8] COPY csp_hsts.conf /etc/apache2/conf-available/security-hardened.conf 0.1s
=> [7/8] RUN a2enconf security-hardened 0.3s
=> [8/8] RUN a2ensite default-ssl 0.3s
=> exporting to image 0.7s
=> => exporting layers 0.4s
=> => exporting manifest sha256:9ee315c3e14111870ce11d577c9cba3eb315fcc80e0f1740bd861811e53887ca 0.0s
=> => exporting config sha256:c997ab65bc5844d0071669f28ad9629226d94e6222290459e9be1d7994ab0bbe 0.0s
=> => exporting attestation manifest sha256:c19adac52bf1c218778c32a4ab91b103320079d2e6dcc810618d8c1cc239e31 0.0s
=> => exporting manifest (list sha256:336e4c3ab6c61033bd0191e16468ed9c/b5f754b55/a0ec5ec741680352ee8b6c 0.0s
=> => naming to docker.io/library/apache-p1:latest 0.0s
=> => unpacking to docker.io/library/apache-p1:latest 0.1s
```

Imagen 14. Construcción de la imagen.

Una vez la imagen ha sido creada, arrancamos un contenedor con la imagen y redirigiendo el tráfico de los puertos 8080 y 8081 de la máquina real a los puertos 80 y 443 respectivamente.

Además, se comprueba que el contenedor creado, se ha levantado correctamente y se encuentra operativo.

```
root@vera-umate:/home/vmate/RA3/p1# docker run -d -p 8080:80 -p 8081:443 --name apache-test apache-p1
c98cf378cb3a8d16f30f348b6b9d442bf0b35c0fd687fd31318fcfa9aa9793e5
root@vera-umate:/home/vmate/RA3/p1# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
c98cf378cb3a apache-p1 "apache2ctl -d FOREG..." 6 seconds ago Up 5 seconds 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp, 0.0.0.0:8081->443/tcp, [::]:8081->443/tcp apache-test
```

Imagen 15. Creación de un nuevo contenedor.

Finalmente, como en la primera fase verificamos funcionamiento y visualizamos que todo funciona según lo planeado. La versión no se encuentra disponible y se visualizan las directivas CSP y HTTS configuradas.

```
00076969ad7c7e802acc8c5011e0e20e54799ed044052a5c0ed049722e640d4
root@vera-umate:/home/vmate/RA3/p1# curl -I http://localhost:8080
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2026 18:23:06 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubDomains
Last-Modified: Thu, 15 Jan 2026 18:05:35 GMT
ETag: "29cd-648711240b9c0"
Accept-Ranges: bytes
Content-Length: 10701
Vary: Accept-Encoding
Content-Security-Policy: default-src 'self'; img-src *; media-src media1.com media2.com; script-src userscripts.example.com
Content-Type: text/html
```

Imagen 16. Verificación de funcionamiento.

Práctica 2 Integrar un WAF (ModSecurity)

El primer paso para esta segunda práctica consistirá en instalar la librería perteneciente a mod-security tal como a continuación se visualiza:

```
vmate@vera-umate:~$ sudo apt install libapache2-mod-security2
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
El paquete indicado a continuación se instaló de forma automática y ya no es necesario.
  liblvm19
Utilice "sudo apt autoremove" para eliminarlo.
```

Imagen 17. Instalar librería necesaria.

Tras esto, vamos a hacer uso de la configuración recomendada proporcionada por modsecurity por lo que copiaremos el archivo de configuración recomendado y lo nombraremos como .conf sin añadidos.

```
Procesando dispositores para liblvm19 (2.3.9-ubuntu0.8) ...
vmate@vera-umate:~$ sudo cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf
vmate@vera-umate:~$
```

Imagen 18. Crear fichero de configuración.

Tras crear el archivo, lo editaremos y modificaremos la línea SecRuleEngine que viene por defecto con la opción **DetectionOnly** dejándola como **On** tal como se ve en las siguientes imágenes:

```
vmate@vera-umate:~$ sudo nano /etc/modsecurity/modsecurity.conf
```

Imagen 19. Editar fichero de configuración.

```
# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine DetectionOnly
```

Imagen 20. Línea por defecto.

```
# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine On
```

Imagen 21. Línea modificada.

Ahora, aunque no es necesario ya que ModSecurity bloquea intentos de inyección en la URL por defecto, vamos a crear una regla de prueba.

En ella indicaremos que si testparam es =test devuelva un mensaje de error 403, indicando que no tenemos permiso a este recurso, con el mensaje Cazado por Ciberseguridad.

```
vmate@vera-umate:~$ sudo service apache2 restart
vmate@vera-umate:~$ sudo nano /etc/apache2/sites-available/default-ssl.conf
vmate@vera-umate:~$
```

Imagen 22. Acceder a la configuración del Virtual Host.

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

Imagen 23. Creación de la regla de prueba.

Tras la creación de la regla, reiniciaremos apache para aplicar los cambios realizados.

```
vmate@vera-umate:~$ sudo service apache2 restart  
vmate@vera-umate:~$
```

Imagen 24. Reiniciar apache.

Ahora procedemos a realizar la prueba, siguiendo los parámetros de la regla creada y como se puede visualizar en la siguiente imagen, la configuración aplicada es verídica:

```
vmate@vera-umate:~$ curl "http://localhost/index.html?testparam=test"  
<!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>
```

Imagen 25. Verificar configuración.

Dockerfile - Práctica 2

Para la configuración de este segundo Dockerfile, vamos a utilizar la base construida en el primero y únicamente vamos a añadir el bloque de ModSecurity a lo que ya se encuentra establecido.

La línea de sed que se ve en la siguiente imagen es crítica, ya que por defecto ModSecurity solo anota los ataques en el log, pero deja pasar el tráfico, al cambiarlo a On como en la fase previa conseguimos que el WAF corte la conexión y devuelva el Status Code 403 que se pide en la práctica al detectar un comportamiento sospechoso.

De forma que el Dockerfile quede como se visualiza a continuación:

AUDITORÍA TI NOVATECH

```
root@vera-umate: /home/vmate/RA3/p1
GNU nano 7.2                                            dockerfile
FROM debian:bullseye

# Instalar Apache, SSL y el módulo de ModSecurity
RUN apt-get update && apt-get install -y \
    apache2 \
    ssl-cert \
    libapache2-mod-security2 \
    && apt-get clean

# 1. Hardening de la Práctica 1
RUN a2dismod -f autoindex
RUN a2disconf security
RUN a2enmod ssl headers

# 2. Configuración de ModSecurity (Práctica 2)
# Copiamos el archivo recomendado a la ubicación activa
RUN cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf

# IMPORTANTE: Por defecto ModSecurity viene en modo "DetectionOnly".
# Vamos a cambiarlo a "On" para que realmente bloquee ataques (403 Forbidden).
RUN sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/' /etc/modsecurity/modsecurity.conf

# 3. Aplicar las cabeceras personalizadas (HSTS, CSP)
COPY csp_hsts.conf /etc/apache2/conf-available/security-hardened.conf
RUN a2enconf security-hardened
RUN a2ensite default-ssl

EXPOSE 80 443

CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Imagen 26. Dockerfile modificado.

Tras esto vamos a crear una nueva imagen para esta segunda práctica con el nuevo dockerfile modificado, tal y como se muestra a continuación:

```
root@vera-umate:/home/vmate/RA3/p1# docker build -t apache-p2 .
[+] Building 23.1s (15/15) FINISHED
   digest: sha256:2eddd7cc91860d7d7af2706a813cd3b60757d1611566cde75fceab54fa856712
   docker:default
      0.0s
=> [internal] load build definition from dockerfile
      0.0s
=> => transferring dockerfile: 1.02KB
      0.0s
=> [internal] load metadata for docker.io/library/debian:bullseye
      0.9s
=> [internal] load .dockerrignore
      0.0s
=> => transferring context: 2B
      0.0s
=> CACHED [ 1/10] FROM docker.io/library/debian:bullseye@sha256:8a08b2875ed17adc464ae03cf5f8652a41821fb8d3d2b17923f11e7489b146d
      0.0s
=> => resolve docker.io/library/debian:bullseye@sha256:8a08b2875ed17adc464ae03cf5f8652a41821fb8d3d2b17923f11e7489b146da
      0.0s
=> [internal] load build context
      0.0s
=> => transferring context: 35B
      0.0s
=> [ 2/10] RUN apt-get update && apt-get install -y      apache2      ssl-cert      libapache2-mod-security2      && apt-get clean  13.8s
      0.3s
=> [ 3/10] RUN a2dismod -f autoindex
      0.3s
=> [ 4/10] RUN a2disconf security
      0.3s
=> [ 5/10] RUN a2enmod ssl headers
      0.3s
=> [ 6/10] RUN cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf
      0.2s
=> [ 7/10] RUN sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/' /etc/modsecurity/modsecurity.conf
      0.3s
=> [ 8/10] COPY csp_hsts.conf /etc/apache2/conf-available/security-hardened.conf
      0.1s
=> [ 9/10] RUN a2enconf security-hardened
      0.3s
=> [10/10] RUN a2ensite default-ssl
      0.3s
=> exporting to image
      5.9s
=> => exporting layers
      4.6s
=> => exporting manifest sha256:2eddd7cc91860d7d7af2706a813cd3b60757d1611566cde75fceab54fa856712
      0.0s
=> => exporting config sha256:f240b2dbd09fbdf002a9a0eacdd4fd393410da3c5628a5d1be20f3cd3b01084
      0.0s
=> => exporting attestation manifest sha256:e1bdb70e5658914e1c972a16f6c38d630e8ae75ffa33f70abf2e6b496b2a2732
      0.0s
=> => exporting manifest list sha256:ab2560b72ce0f6988d7294f94e7ae5070a44ca4abfd59ea7913b5b42c65577a2
      0.0s
=> => naming to docker.io/library/apache-p2:latest
      0.0s
=> => unpacking to docker.io/library/apache-p2:latest
      1.2s
```

Imagen 27. Crear nueva imagen.

Ahora lanzamos un nuevo contenedor con dicha imagen, es importante parar y eliminar previamente el anterior para evitar confusiones. Tras esto de nuevo comprobamos también que esté arrancado y operativo:

AUDITORÍA TI NOVATECH

```
root@vera-umate:/home/vmate/RA3/p1# docker run -d -p 8080:80 -p 8081:443 --name apache-test apache-p2
6b56566adb10c2673fea9eac79107d01b46fc24311b3dfa5aa19cffcaac4bc9b
root@vera-umate:/home/vmate/RA3/p1# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
6b56566adb10 apache-p2 "apache2ctl -D FORE..." 3 minutes ago Up 3 minutes 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp, 0.0.0.0:8081->443/tcp, [::]:8081->443/tcp apache-test
```

Imagen 28. Lanzar contenedor con nueva imagen.

Y finalmente comprobamos que el WAF funciona, para ello se realiza una prueba rápida en la que se intenta realizar un ataque de Path Traversal (es decir leer los archivos del sistema):

```
root@vera-umate:/home/vmate/RA3/p1# curl -I "http://localhost:8080/index.php?exec=/bin/bash"
HTTP/1.1 403 Forbidden
Date: Thu, 15 Jan 2026 20:26:52 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubDomains
Content-Type: text/html; charset=iso-8859-1
```

Imagen 29. Intento de ataque para verificar configuración.

Como se visualiza en la anterior imagen, el resultado es el esperado, ModSecurity intercepta la petición sospechosa y la bloquea antes de que llegue a la “aplicación”.

Dockerfile - Práctica 3

Para que Docker pueda cargar las reglas de forma correcta, es necesario que el archivo de configuración del módulo incluya la carpeta de reglas. Para ello vamos a crear un archivo local llamado security2.conf con el siguiente contenido:

```
root@vera-umate:/home/vmate/RA3/p1
security2.conf
GNU nano 7.2
<IfModule security2_module>
    SecDataDir /var/cache/modsecurity
    # Activamos el motor de reglas
    SecRuleEngine On

    # Cargamos primero la configuración base
    IncludeOptional /etc/modsecurity/*.conf
    # Cargamos las reglas de OWASP que descargaremos
    Include /etc/modsecurity/rules/*.conf
</IfModule>
```

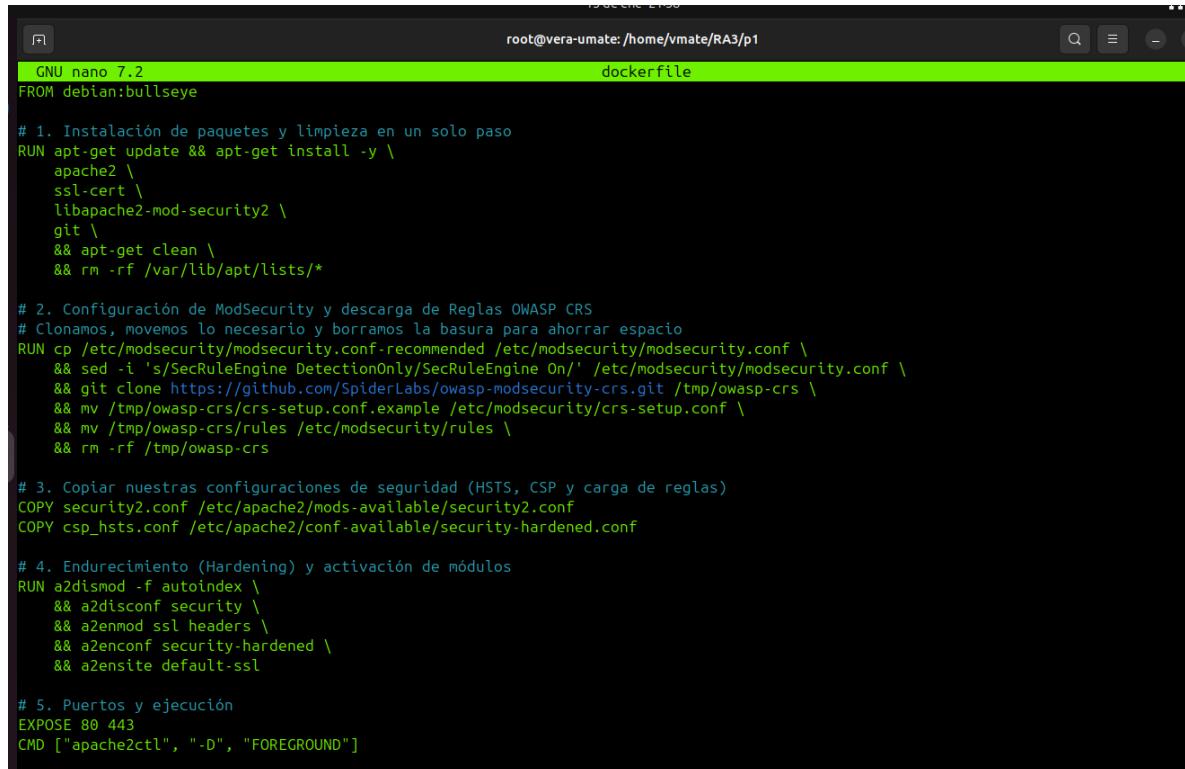
Imagen 30. Archivo security2.conf.

A continuación, se muestra cómo debe quedar el contenido del fichero Dockerfile, de nuevo hacemos uso de la base del anterior archivo con las pertinentes modificaciones, donde vamos a automatizar la descarga de las reglas desde GitHub y a organizar los directorios tal y como hemos hecho de forma manual en la fase previa.

Además, se ha reestructurado y se han quitado cosas innecesarias para dejar todo lo más limpio posible:

AUDITORÍA TI NOVATECH

- Regla testparam: No es necesaria, ahora el WAF hará uso de las reglas estándar de la industria (OWASP).
- Archivos temporales: Minimizamos el tamaño de la imagen con el comando rm -rf /tmp/owasp-crs y la limpieza de apt.
- Redundancia de comandos: Reestructuración para que el archivo sea más sencillo de leer y mantener.



```
GNU nano 7.2 dockerfile
FROM debian:bullseye

# 1. Instalación de paquetes y limpieza en un solo paso
RUN apt-get update && apt-get install -y \
    apache2 \
    ssl-cert \
    libapache2-mod-security2 \
    git \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# 2. Configuración de ModSecurity y descarga de Reglas OWASP CRS
# Clonamos, movemos lo necesario y borramos la basura para ahorrar espacio
RUN cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf \
    && sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/' /etc/modsecurity/modsecurity.conf \
    && git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git /tmp/owasp-crs \
    && mv /tmp/owasp-crs/crs-setup.conf.example /etc/modsecurity/crs-setup.conf \
    && mv /tmp/owasp-crs/rules /etc/modsecurity/rules \
    && rm -rf /tmp/owasp-crs

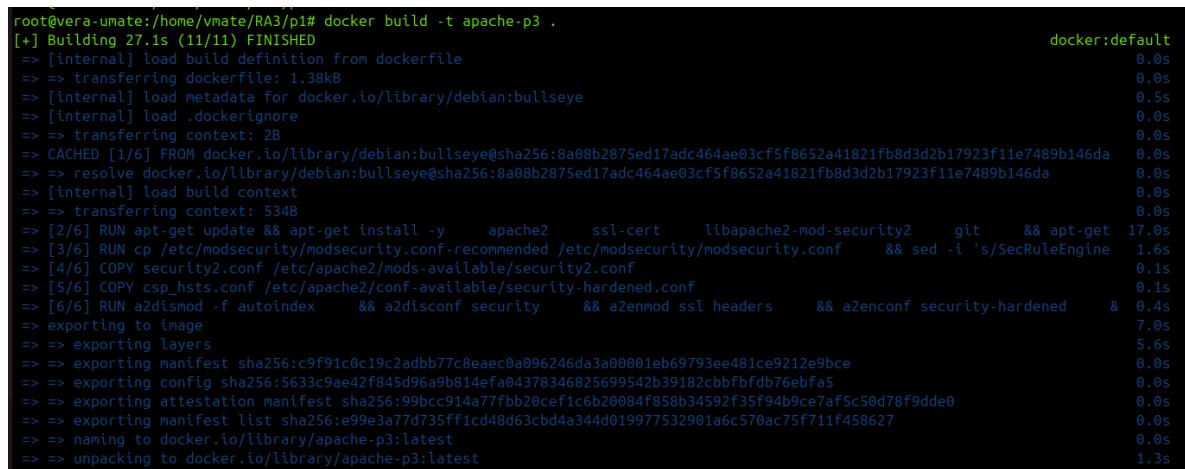
# 3. Copiar nuestras configuraciones de seguridad (HSTS, CSP y carga de reglas)
COPY security2.conf /etc/apache2/mods-available/security2.conf
COPY csp_hsts.conf /etc/apache2/conf-available/security-hardened.conf

# 4. Endurecimiento (Hardening) y activación de módulos
RUN a2dismod -f autoindex \
    && a2disconf security \
    && a2enmod ssl headers \
    && a2enconf security-hardened \
    && a2ensite default-ssl

# 5. Puertos y ejecución
EXPOSE 80 443
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Imagen 31. Dockerfile modificado.

De nuevo creamos la nueva imagen, lanzamos nuevo contendor tras eliminar el anterior y comprobamos su operatividad.



```
root@vera-umate:/home/vmate/RA3/p1# docker build -t apache-p3 .
[+] Building 27.1s (11/11) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 1.38kB
=> [internal] load metadata for docker.io/library/debian:bullseye
=> [internal] load .dockerrignore
=> => transferring context: 2B
=> CACHED [1/6] FROM docker.io/library/debian:bullseye@sha256:8a08b2875ed17adc464ae03cf5f8652a41821fb8d3d2b17923f11e7489b146da
=> => resolve docker.io/library/debian:bullseye@sha256:8a08b2875ed17adc464ae03cf5f8652a41821fb8d3d2b17923f11e7489b146da
=> [internal] load build context
=> => transferring context: 534B
=> [2/6] RUN apt-get update && apt-get install -y apache2 ssl-cert libapache2-mod-security2 git && apt-get 17.0s
=> [3/6] RUN cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf && sed -i 's/SecRuleEngine 1.6s
=> [4/6] COPY security2.conf /etc/apache2/mods-available/security2.conf 0.1s
=> [5/6] COPY csp_hsts.conf /etc/apache2/conf-available/security-hardened.conf 0.1s
=> [6/6] RUN a2dismod -f autoindex && a2disconf security && a2enmod ssl headers && a2enconf security-hardened & 0.4s
=> exporting to image 7.0s
=> => exporting layers 5.6s
=> => exporting manifest sha256:c9f91c0c19c2adb77c8eaec0a096246da3a00001eb69793ee481ce9212e9bce 0.0s
=> => exporting config sha256:5633c9aa42f845d96a9b814ef0a4378346825699542b39182cbfffd076ebfa5 0.0s
=> => exporting attestation manifest sha256:99bcc914a77fb20cef1c6b20084f858b34592f35f94b9ce7af5c50d78f9dde0 0.0s
=> => exporting manifest list sha256:e99e3a77d735ff1cd48d63cbd4a344d019977532901a6c570ac75f711f458627 0.0s
=> => naming to docker.io/library/apache-p3:latest 0.0s
=> => unpacking to docker.io/library/apache-p3:latest 1.3s
```

Imagen 32. Creación de nueva imagen.

AUDITORÍA TI NOVATECH

```
root@vera-umate:/home/vmate/RA3/p1# docker run -d -p 8080:80 -p 8081:443 --name apache-test apache-p3
448baea38c9a7a16378a97b0e8df6949cd9e514219310c2fd0837e7f2541f878
root@vera-umate:/home/vmate/RA3/p1# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
448baea38c9a apache-p3 "apache2ctl -D FORE..." 4 seconds ago Up 4 seconds 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp, 0.0.0.0:8081->443/tcp, [::]:8081->443/tcp apache-test
```

Imagen 33. Arrancar nuevo contenedor.

Finalmente, para la verificación de que las reglas OWASP están en funcionamiento trabajando en segundo plano vamos a lanzar los siguientes ataques:

- Inyección de comandos.
- Path Transversal.

Ambos ataques son detectados por las reglas OWASP, como se visualiza en la siguiente imagen:

```
root@vera-umate:/home/vmate/RA3/p1# curl -I "http://localhost:8080/index.html?exec=/bin/bash"
HTTP/1.1 403 Forbidden
Date: Thu, 15 Jan 2026 21:01:45 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubDomains
Content-Type: text/html; charset=iso-8859-1

root@vera-umate:/home/vmate/RA3/p1# curl -I "http://localhost:8080/index.html?exec=../../../../"
HTTP/1.1 403 Forbidden
Date: Thu, 15 Jan 2026 21:02:11 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubDomains
Content-Type: text/html; charset=iso-8859-1
```

Imagen 34. Pruebas para verificar funcionamiento de las reglas OWASP.

Además, si visualizamos los logs, veremos que los ataques han sido detectados por el servidor y hemos podido dar caza a ataques reales haciendo uso de las reglas OWASP CRS.

AUDITORÍA TI NOVATECH

Imagen 35. Logs generados.

A continuación, se muestra la detección de un ataque desglosado:

En primer lugar vemos como la regla OWASP **930110** (especializada en ataques de inclusión de archivos locales) detectó que en el parámetro exec alguien intentó poner `../../../../`. ModSecurity entiende que esto es un intento de saltar directorios para leer archivos sensibles del sistema (como `/etc/passwd`).

```
(?:[\\|\\"|/])\\(.\\|(?:[\\\\\\]|$))" at ARGS:exec. [file "/etc/modsecurity/rules/REQUEST-930-APPLICATION-ATTACK-LFI"] [id "930110"] [msg "Path Traversal Attack (/..)"] [data "Matched Data: /.. found within ARGS:exec: /..../"] [ver "OWASP CRS/3.2.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-lfi"]
```

Imagen 36. Detección del ataque.

ModSecurity con OWASP no bloquea siempre a la primera (depende de la configuración), sino que va sumando puntos de "malidad".

- Como el ataque era **CRITICAL**, le asignó **30 puntos** de golpe.
 - El límite por defecto suele ser 5. Al llegar a 30, el WAF dice: "Esto es definitivamente un ataque".

```
[msg "Inbound Anomaly Score Exceeded (Total Score: 30)"] [language_multi] [tag "platform_multi"] [tag "attack_cen
```

Imagen 37. Sistema de puntuación.

Aquí es donde se ejecuta la acción: el servidor le cierra la puerta al usuario y le devuelve el error **403 Forbidden**.

```
[J] ModSecurity: Access denied with code 403 (phase 2). Opened file "/etc/modsecurity/modsec2.conf" [File "Unknown Access"] Source: Fingerprint (T
```

Imagen 38. Bloqueo final.

Dockerfile - Práctica 4

Para esta práctica se nos pide instalar el módulo mod_evasive, herramienta fundamental para proteger la disponibilidad de nuestro servidor frente a ataques de fuerza bruta o inundaciones de peticiones.

Por ello vamos a crear un archivo independiente evasive.conf, que va a contener los parámetros de configuración a establecer por mod_evasive y que controlará los umbrales de baneo.

```
root@vera-umate:/home/vmate/RA3/p1# sudo nano evasive.conf
root@vera-umate:/home/vmate/RA3/p1# █
```

Imagen 39. Creación del documento evasive.conf.

```
GNU nano 7.2                                     evasive.conf
<IfModule mod_evasive20.c>
    DOSHashTableSize    2048
    DOSPageCount        5
    DOSSiteCount        100
    DOSPageInterval     1
    DOSSiteInterval     2
    DOSBlockingPeriod   10
    DOSLogDir           "/var/log/mod_evasive"
</IfModule>█
```

Imagen 40. Contenido de evasive.conf.

De nuevo, vamos a realizar algunas modificaciones en nuestro dockerfile añadiendo la instalación de libapache2-mod-evasive y la configuración de los directorios.

Debe crearse el directorio de logs y darle permisos ya que, por defecto, no se crea todo esto está documentado en el manual oficial y posteriormente copiar nuestro archivo de configuración creado.

Además, ya que en imágenes Debian, los arhchivos /usr/share/doc vienen comprimidos o no son instalados para ahorrar espacio, vamos a añadir una línea que asegure tener el test a mano para poder ser utilizado.

AUDITORÍA TI NOVATECH

```
GNU nano 7.2                                            dockerfile
FROM debian:bullseye

# 1. Instalación de paquetes y limpieza en un solo paso
RUN apt-get update && apt-get install -y \
    apache2 \
    ssl-cert \
    libapache2-mod-security2 \
    libapache2-mod-evasive \
    git \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# 2. Configuración de ModSecurity y descarga de Reglas OWASP CRS
# Clonamos, movemos lo necesario y borramos la basura para ahorrar espacio
RUN cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf \
    && sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/' /etc/modsecurity/modsecurity.conf \
    && git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git /tmp/owasp-crs \
    && mv /tmp/owasp-crs/crs-setup.conf.example /etc/modsecurity/crs-setup.conf \
    && mv /tmp/owasp-crs/rules /etc/modsecurity/rules \
    && rm -rf /tmp/owasp-crs

# 3. Configuración de mod_evasive
RUN mkdir -p /var/log/mod_evasive && chown -R www-data:www-data /var/log/mod_evasive
#COPIAMOS EL ARCHIVO DE CONFIGURACIÓN
COPY evasive.conf /etc/apache2/mods-available/evasive.conf
#TENER EL TEST DISPONIBLE
RUN cp /usr/share/doc/libapache2-mod-evasive/examples/test.pl /root/test.pl

# 4. Copiar nuestras configuraciones de seguridad (HSTS, CSP y carga de reglas)
COPY security2.conf /etc/apache2/mods-available/security2.conf
COPY csp_hsts.conf /etc/apache2/conf-available/security-hardened.conf

# 4. Endurecimiento (Hardening) y activación de módulos
RUN a2dismod -f autoindex \
    && a2disconf security \
    && a2enmod ssl headers evasive \
    && a2enconf security-hardened \
    && a2ensite default-ssl

# 5. Puertos y ejecución
EXPOSE 80 443
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Imagen 41. Contenido del dockerfile.

Creamos la nueva imagen, arrancamos un nuevo contenedor y comprobamos su operatividad.

```
root@vera-umate:/home/vmate/RA3/p1# docker build -t apache-p4 .
[+] Building 40.2s (13/13) FINISHED
  => [internal] load build definition from dockerfile
  => => transferring dockerfile: 1.63kB
  => [internal] load metadata for docker.io/library/debian:bullseye
  => [internal] load .dockignore
  => => transferring context: 2B
  => CACHED [1/8] FROM docker.io/library/debian@sha256:8a08b2875ed17adc464ae03cf5f8652a41821fb8d3d2b17923f11e748 0.1s
  => => resolve docker.io/library/debian@sha256:8a08b2875ed17adc464ae03cf5f8652a41821fb8d3d2b17923f11e7489b146da 0.0s
  => [internal] load build context
  => => transferring context: 360B
  => [2/8] RUN apt-get update && apt-get install -y apache2 ssl-cert libapache2-mod-security2 libapache2 28.5s
  => [3/8] RUN cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf && sed -i 's/SecRul 0.0s
  => [4/8] RUN mkdir -p /var/log/mod_evasive && chown -R www-data:www-data /var/log/mod_evasive 0.0s
  => [5/8] COPY evasive.conf /etc/apache2/mods-available/evasive.conf 0.1s
  => [6/8] COPY security2.conf /etc/apache2/mods-available/security2.conf 0.1s
  => [7/8] COPY csp_hsts.conf /etc/apache2/conf-available/security-hardened.conf 0.1s
  => [8/8] RUN a2dismod -f autoindex && a2disconf security && a2enmod ssl headers evasive && a2enconf securit 0.4s
  => exporting to image
  => exporting layers
  => => exporting manifest sha256:2291bf1799259ee3793172df849c96b748efd2948017d0237f8c0da42da10d7 0.0s
  => => exporting config sha256:80dc74a24ed0946e4e06761a6ecb495829d939094468068b0ca5e4e81a5fe1e6 0.0s
  => => exporting attestation manifest sha256:484cef5ef4841bee73c865b74ab671fd344def8de79f3a466972519dd268c102 0.0s
  => => exporting manifest list sha256:d3338643313aa3f1024462959f59070466fce8d34274e470de42a1660d8 0.0s
  => => naming to docker.io/library/apache-p4:latest
  => => unpacking to docker.io/library/apache-p4:latest 1.3s
root@vera-umate:/home/vmate/RA3/p1#
```

Imagen 42. Creación de la nueva imagen.

AUDITORÍA TI NOVATECH

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
root@vera-umate:/home/vmate/RA3/p1# docker run -d -p 8080:80 -p 8081:443 --name apache-test2 apache-p4
c733b9e0f0f89461cf9ab297cc110a5dc8d196a488a435e88eeef01f8079fce2
root@vera-umate:/home/vmate/RA3/p1# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
c733b9e0f0f8 apache-p4 "apache2ctl -D FOREG..." 4 seconds ago Up 4 seconds 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp, 0.0
.0.0:8081->443/tcp, [::]:8081->443/tcp apache-test2
root@vera-umate:/home/vmate/RA3/p1#
```

Imagen 43. Desplegar nuevo contenedor con la imagen creada.

Tras esto procedemos a la ejecución de la primera prueba, ejecutamos el script test.pl proporcionado por mod_evasive el cuál lanza peiticones en bucle muy rápido para verificar que el servidor empieza a devolver errores 403 (Forbidden).

Tras ejecutarlo, se visualiza una lista de respuestas, al principio nos indica HTTP/1.1 200 OK indicando que la conexión se establece correctamente, pero después de las primeras 8 peticiones, empieza a salir el mensaje 403 Forbidden, esto se debe a que se está aplicando el bloqueo configurado en los parámetros de mod_evasive.

Imagen 44. Ejecución de test.pl.

Tras verificar con el script de Perl que el módulo bloquea, vamos a ejecutar una segunda prueba de funcionamiento con Apache Bench, desde la máquina anfitrión. Se van a lanzar más de 100 peticiones muy rápido para superar el DOSPageCount 5 establecido.

AUDITORÍA TI NOVATECH

Como se puede visualizar en el informe, 88 de las 100 peticiones fallan, indicando de nuevo el bloqueo del ataque de denegación de servicio por parte de mod_evasive.

```
root@vera-umate:/home/vmate/RA3/p1# ab -n 100 -c 5 http://localhost:8080/index.html
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient).....done

Server Software:        Apache
Server Hostname:        localhost
Server Port:            8080

Document Path:          /index.html
Document Length:        10701 bytes

Concurrency Level:      5
Time taken for tests:  0.097 seconds
Complete requests:     100
Failed requests:        88
    (Connect: 0, Receive: 88, Length: 88, Exceptions: 0)
Non-2xx responses:     88
Total transferred:     171352 bytes
HTML transferred:      145924 bytes
Requests per second:   1030.74 [#/sec] (mean)
Time per request:      4.851 [ms] (mean)
Time per request:      0.970 [ms] (mean, across all concurrent requests)
Transfer rate:         1724.79 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       0    0   0.1     0     1
Processing:    1    4   3.9     3    16
Waiting:       0    3   3.6     1    15
Total:         1    5   3.9     3    16

Percentage of the requests served within a certain time (ms)
  50%   3
  66%   5
  75%   6
  80%   8
  90%  10
  95%  15
  98%  16
  99%  16
100% 16 (longest request)
```

Imagen 45. Prueba de funcionamiento con Apache Bench.

Ahora para poder ver que hemos sido bloqueados, podemos listar los logs generados en el directorio configurado previamente. En el encontramos archivos llamados dos-127.0.0.1, mostrando como mod_evasive ha identificado correctamente a un atacante.

```
root@vera-umate:/home/vmate/RA3/p1# docker exec apache-test2 ls -la /var/log/mod_evasive
total 16
drwxr-xr-x 1 www-data www-data 4096 Jan 16 18:53 .
drwxr-xr-x 1 root      root    4096 Jan 16 18:44 ..
-rw-r--r-- 1 www-data www-data    3 Jan 16 18:53 dos-172.17.0.1
root@vera-umate:/home/vmate/RA3/p1#
```

Imagen 46. Visualizar directorio de logs.

Finalmente, volvemos a ejecutar Apache Bench y guardamos los resultados en un fichero.txt que servirá de informe.

```
root@vera-umate:/home/vmate/RA3/p1# ab -n 100 -c 5 http://localhost:8080/index.html > informe_evasive.txt
root@vera-umate:/home/vmate/RA3/p1# ls
csp_hsts.conf  dockerfile  evasive.conf  informe_evasive.txt  security2.conf
```

Imagen 47. Creación del informe_evasive.txt.

Práctica Certificados (Dockerfile)

Para esta práctica se nos pide crear una imagen que levante un contenedor con un servidor Apache con certificado digital instalado.

A continuación, se muestra el dockerfile, necesario para poder completar esta actividad.

Siguiendo la información de la que se dispone, es bastante sencillo su montaje únicamente se destacaría la importancia de la opción -subj a la hora de generar el certificado para responder de forma automática a las preguntas realizadas durante la construcción.

Se destaca también el uso del archivo default-ssl.conf con los parámetros previamente configurados para tener el dockerfile lo más limpio y optimizado posible.

```
root@vera-umate:~/certificados
GNU nano 7.2
dockerfile
FROM debian:bullseye

# 1. Instalación de Apache y OpenSSL
RUN apt-get update && apt-get install -y \
    apache2 \
    ssl-cert \
&& apt-get clean

# 2. Habilitar modulo SSL
RUN a2enmod ssl

# 3.Creacion del directorio donde se almacenaran los certificados
RUN mkdir /etc/apache2/ssl

# 4. Generar certificado Auto-firmado de forma automatica
# El parametro -subj nos permite evitar que OpenSSL realice preguntas durante la construccion
RUN openssl req -x509 -nodes -days 365 \
    -newkey rsa:2048 \
    -keyout /etc/apache2/ssl/apache.key \
    -out /etc/apache2/ssl/apache.crt \
    -subj "/C=ES/ST=Castellon/L=Castellon/O=Caminas Web/OU=Server Dev/CN=www.midominioseguro.com"

# 5. Copiamos nuestra configuracion personalizada de VirtualHost
COPY default-ssl.conf /etc/apache2/sites-available/default-ssl.conf

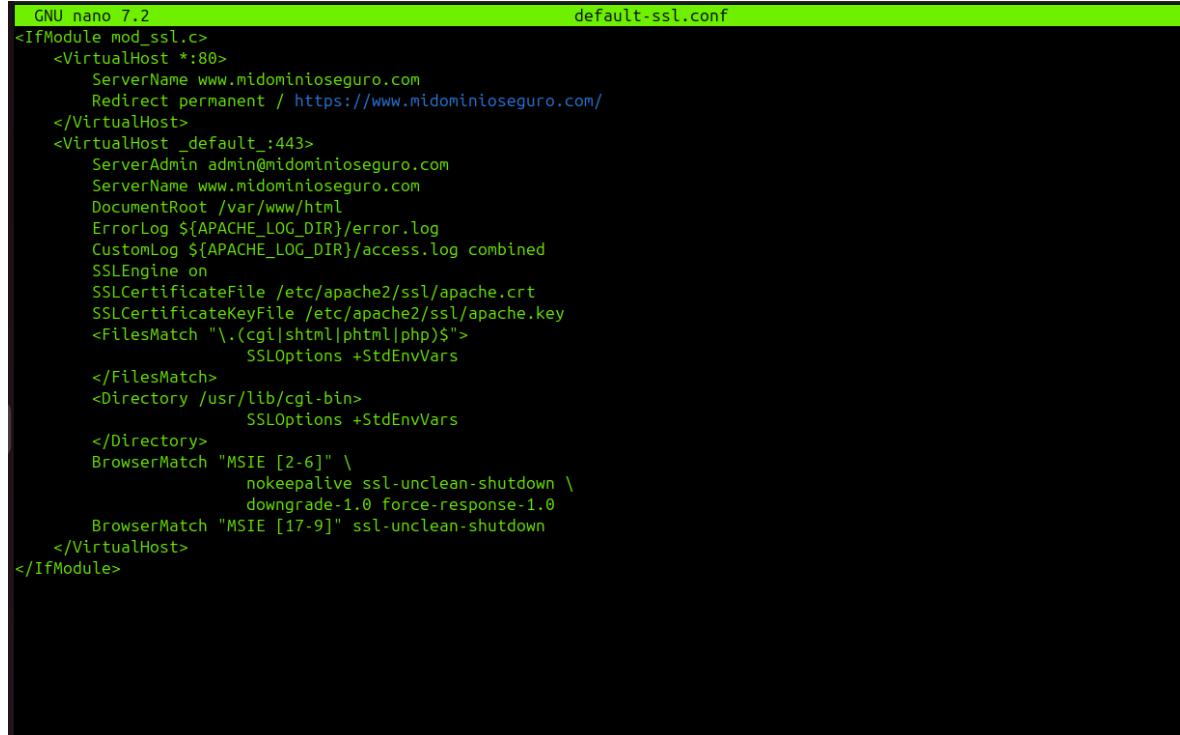
# 6. Activamos el sitio y desactivamos el sitio por defecto, de forma que se eviten posibles conflictos.
RUN a2ensite default-ssl.conf && a2dissite 000-default.conf

# Exponer puertos
EXPOSE 80 443

CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Imagen 48. Contenido del dockerfile.

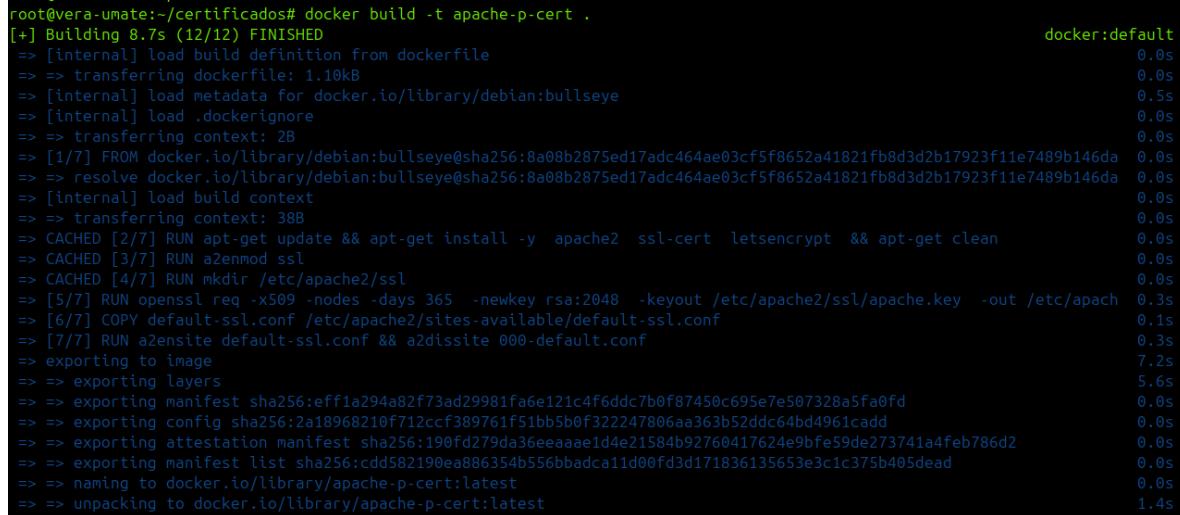
AUDITORÍA TI NOVATECH



```
GNU nano 7.2                               default-ssl.conf
<IfModule mod_ssl.c>
  <VirtualHost *:80>
    ServerName www.midominioseguro.com
    Redirect permanent / https://www.midominioseguro.com/
  </VirtualHost>
  <VirtualHost _default_:443>
    ServerAdmin admin@midominioseguro.com
    ServerName www.midominioseguro.com
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/apache.crt
    SSLCertificateKeyFile /etc/apache2/ssl/apache.key
    <FilesMatch "\.(cgi|shtml|phtml|php)$">
      SSLOptions +StdEnvVars
    </FilesMatch>
    <Directory /usr/lib/cgi-bin>
      SSLOptions +StdEnvVars
    </Directory>
    BrowserMatch "MSIE [2-6]" \
      nokeepalive ssl-unclean-shutdown \
      downgrade-1.0 force-response-1.0
    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
  </VirtualHost>
</IfModule>
```

Imagen 49. Contenido del fichero default-ssl.conf.

Ahora únicamente quedará el montaje de la imagen y tras esto levantar el contenedor con la misma y comprobar funcionamiento.



```
root@vera-umate:~/certificados# docker build -t apache-p-cert .
[+] Building 8.7s (12/12) FINISHED
   => [internal] load build definition from dockerfile
   => => transferring dockerfile: 1.10kB
   => [internal] load metadata for docker.io/library/debian:bullseye
   => [internal] load .dockerignore
   => => transferring context: 2B
   => [1/7] FROM docker.io/library/debian:bullseye@sha256:8a08b2875ed17adc464ae03cf5f8652a41821fb8d3d2b17923f11e7489b146da
   => => resolving docker.io/library/debian:bullseye@sha256:8a08b2875ed17adc464ae03cf5f8652a41821fb8d3d2b17923f11e7489b146da
   => [internal] load build context
   => => transferring context: 38B
   => CACHED [2/7] RUN apt-get update && apt-get install -y apache2 ssl-cert letsencrypt && apt-get clean
   => CACHED [3/7] RUN a2enmod ssl
   => CACHED [4/7] RUN mkdir /etc/apache2/ssl
   => [5/7] RUN openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.pem
   => [6/7] COPY default-ssl.conf /etc/apache2/sites-available/default-ssl.conf
   => [7/7] RUN a2ensite default-ssl.conf && a2dissite 000-default.conf
   => exporting to image
   => => exporting layers
   => => exporting manifest sha256:eff1a294a82f73ad29981fa6e121c4f6ddc7b0f87450c695e7e507328a5fa0fd
   => => exporting config sha256:a18968210f712ccf389761f51bb5b0f322247806aa363b52ddc64bd4961cadd
   => => exporting attestation manifest sha256:190fd279da36eeaaa1d4e21584b92760417624e9bfe59de273741a4feb786d2
   => => exporting manifest list sha256:cdd582190ea886354b556bbadca11d00fd3d171836135653e3c1c375b405dead
   => => naming to docker.io/library/apache-p-cert:latest
   => => unpacking to docker.io/library/apache-p-cert:latest
   => => 1.4s
```

Imagen 50. Imagen creada.

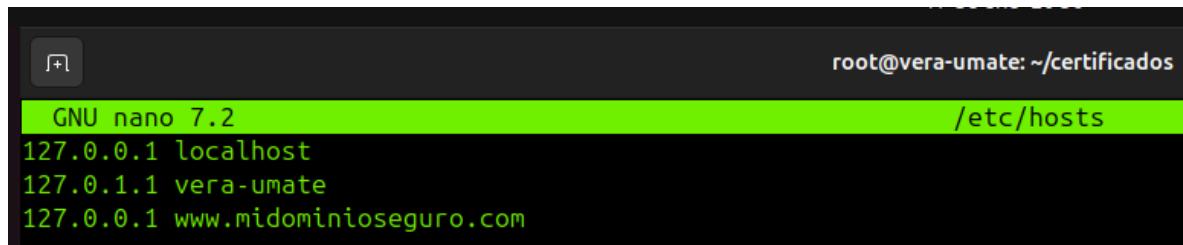


```
root@vera-umate:~/certificados# docker run -d -p 8080:80 -p 8081:443 --name apache-cert apache-p-cert
c8b2254be8c22b0027754d817144684e5b22338a41c46887278fc1ca9613baf
root@vera-umate:~/certificados# docker ps
CONTAINER ID   IMAGE       COMMAND           CREATED          STATUS          PORTS
NAMES
c8b2254be8c2   apache-p-cert   "apache2ctl -D FOREG..."   4 seconds ago   Up 3 seconds   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp,
0.0.0.0:8081->443/tcp, [::]:8081->443/tcp   apache-cert
root@vera-umate:~/certificados#
```

Imagen 51. Contenedor levantado.

AUDITORÍA TI NOVATECH

Finalmente configuraremos el fichero /etc/hosts con el nombre del dominio correspondiente para que lo reconozca.



```
root@vera-umate: ~/certificados
GNU nano 7.2
/etc/hosts
127.0.0.1 localhost
127.0.1.1 vera-umate
127.0.0.1 www.midominioseguro.com
```

Imagen 52. Configuración de /etc/hosts.

Tras esto, si accedemos a la URL indicada en nuestro navegador, podremos visualizar la correcta configuración del servidor y el certificado.

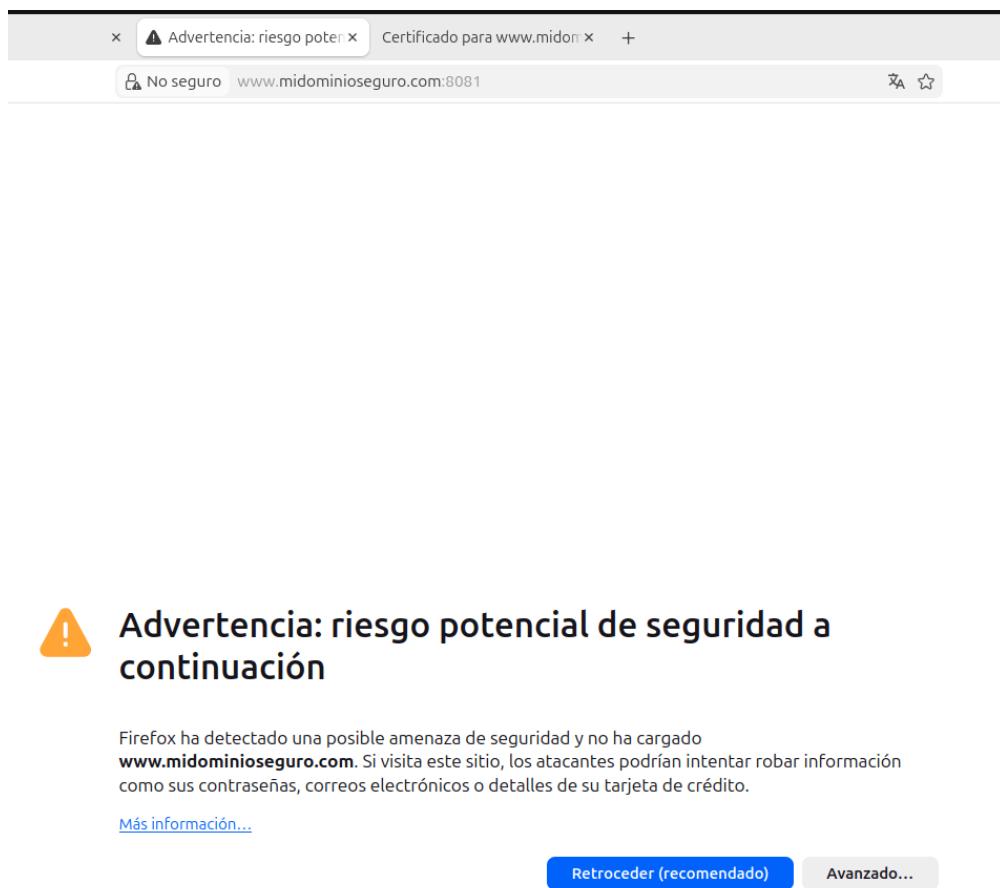


Imagen 53. Comprobación del dominio y funcionamiento del certificado.

AUDITORÍA TI NOVATECH

The screenshot shows a Firefox browser window with the address bar displaying 'Firefox about:certificate?cert=MIID5zCCAs%2BgAwIBAgIUZEEnUmCmf6x2hPxaxCtyb00EigXxwwDQYJKoZ'. The main content area is titled 'Certificado' and displays the following information:

www.midominioseguro.com	
Nombre del asunto	
País	ES
Estado/Provincia	Castellon
Localidad	Castellon
Organización	Caminas Web
Unidad organizativa	Server Dev
Nombre común	www.midominioseguro.com
Nombre del emisor	
País	ES
Estado/Provincia	Castellon
Localidad	Castellon
Organización	Caminas Web
Unidad organizativa	Server Dev
Nombre común	www.midominioseguro.com
Validez	
No antes	Sat, 17 Jan 2026 19:35:58 GMT
No después	Sun, 17 Jan 2027 19:35:58 GMT
Información de clave pública	
Algoritmo	RSA
Tamaño de la clave	2048
Exponente	65537
Módulo	B6:C5:14:10:8E:67:86:2E:B3:D9:FB:78:79:4C:A5:06:65:35:6D:54:FC:CE:B1:91:...

Imagen 54. Certificado.

Práctica Gold Image (Dockerfile)

Para esta última práctica se ha diseñado una golden image, haciendo referencia a la consolidación de todas las capas de seguridad configuradas en las prácticas anteriores, sumando protecciones críticas a nivel del sistema operativo, usuario y protocolo.

Se ha seguido la estrategia de herencia en cascada, garantizando que el contenedor final sea el más robusto de toda la serie realizada.

En el próximo apartado se desarrolla la estrategia de herencia en cascada ejecutada para todos los contenedores que ha permitido llegar a esta última versión.

A continuación, se muestra el dockerfile utilizado para la creación de este contenedor final:

AUDITORÍA TI NOVATECH

```
GNU nano 7.2                                            dockerfile
FROM javi2332/pps_p_certificados_javlluapa:latest

# 1. Crear usuario y grupo apache
RUN groupadd apache && useradd -g apache apache

# 2. Forzar activación de módulos (por si acaso)
RUN a2enmod headers rewrite

# 3. Aplicar el hardening del artículo directamente en el archivo principal
# Esto evita el error de "Invalid command Header" al cargar después del módulo
RUN echo "ServerTokens Prod" >> /etc/apache2/apache2.conf && \
    echo "ServerSignature Off" >> /etc/apache2/apache2.conf && \
    echo "TraceEnable Off" >> /etc/apache2/apache2.conf && \
    echo "Header always set Strict-Transport-Security \"max-age=63072000; includeSubDomains\"" >> /etc/apache2/apache2.conf &>
    echo "Header set Content-Security-Policy \"default-src 'self'; img-src *; media-src media1.com media2.com; script-src 'se>
    echo "Header always append X-Frame-Options SAMEORIGIN" >> /etc/apache2/apache2.conf && \
    echo "Header set X-XSS-Protection \"1; mode=block\"" >> /etc/apache2/apache2.conf

# 4. Arreglar permisos para el nuevo usuario 'apache'
# IMPORTANTE: Necesita escribir en logs de ModSec y Evasive heredados
RUN mkdir -p /var/log/mod_evasive /var/cache/modsecurity && \
    chown -R apache:apache /var/log/apache2 /var/log/mod_evasive /var/cache/modsecurity /var/www/html && \
    chmod -R 750 /etc/apache2/ /usr/sbin/apache2

# 5. Cambiar el usuario de ejecución en el entorno de Apache
RUN sed -i 's/export APACHE_RUN_USER=.*$/export APACHE_RUN_USER=apache/' /etc/apache2/envvars && \
    sed -i 's/export APACHE_RUN_GROUP=.*$/export APACHE_RUN_GROUP=apache/' /etc/apache2/envvars

EXPOSE 80 443
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Imagen 55. Dockerfile definitivo.

A continuación, se explican los parámetros del dockerfile creado:

- Se crea el usuario y grupo apache y configura el servidor para que use dicho usuario en envvars, aplicando el principio de menor privilegio. De esta forma si el servidor es comprometido, el atacante queda “atrapado” en un usuario sin permisos de administración.
- Tras recibir errores de “Invalid command Header”, al utilizar un archivo externo de configuración, se decide escribir las reglas al final del archivo maestro de Apache. De forma que al escribirlo al final de apache2.conf, nos aseguramos de que todos los módulos (headers, rewrite) ya estén cargados antes de que Apache intente leer las órdenes de seguridad.
- Se repara la herencia de las carpetas ModSecurity y Evasive creadas en las capas anteriores, nos aseguramos de que el nuevo usuario apache tiene permiso para poder escribir ahí sus logs. Gracias al blindaje extra aplicado con los permisos impedimos que cualquier otro usuario (que no sea root o apache) pueda leer los archivos de configuración, donde a veces hay información sensible.
- Se fuerza a que todas las peticiones usen HTTP 1.1 o superior, de forma que si alguien intenta usar herramientas de hacking antiguas o mal configuradas que usen HTTP 1.0, el servidor le responderá con un 403 Forbidden.

Esta imagen ha permitido no solo consolidar los módulos de las prácticas anteriores, sino securizar el entorno de ejecución.

Tras esto únicamente queda de nuevo, crear la nueva imagen, lanzar el contenedor con dicha imagen y comprobar su funcionamiento.

AUDITORÍA TI NOVATECH

```
javier@PC-LINUX-JAVIER:~/PPS/RA3/RA3_1/PRACTICA_GOLD_IMAGE$ docker build -t javi2332/pps_p_gold_image_javlluapa:latest .
[+] Building 1.0s (10/10) FINISHED
=> [internal] load build definition from dockerfile                               0.0s
=> => transferring dockerfile: 1.43kB                                         0.0s
=> [internal] load metadata for docker.io/javi2332/pps_p_certificados_javlluapa:latest      0.0s
=> [internal] load .dockignore                                              0.0s
=> => transferring context: 2B                                              0.0s
=> [1/6] FROM docker.io/javi2332/pps_p_certificados_javlluapa:latest@sha256:819973ebacd4db736bdd371743dd7eb8f4517279f9 0.0s
=> => resolve docker.io/javi2332/pps_p_certificados_javlluapa:latest@sha256:819973ebacd4db736bdd371743dd7eb8f4517279f9 0.0s
=> => CACHED [2/6] RUN groupadd apache && useradd -g apache apache           0.0s
=> => CACHED [3/6] RUN a2enmod headers rewrite                                0.0s
=> => CACHED [4/6] RUN echo "ServerTokens Prod" >> /etc/apache2/apache2.conf &&     echo "ServerSignature Off" >> /etc/ap 0.0s
=> => [5/6] RUN mkdir -p /var/log/mod_evasive /var/cache/modsecurity &&          chown -R apache:apache /var/log/apache2 /var/ 0.4s
=> => [6/6] RUN sed -i 's/export APACHE_RUN_USER=.*;/export APACHE_RUN_USER=apache/' /etc/apache2/envvars &&    sed -i 's 0.2s
=> => exporting to image                                                 0.3s
=> => exporting layers                                                 0.1s
=> => exporting manifest sha256:992cfb22fe5723cd0ce32903f0102a2859c026efc4fdcfacf44c25d6d14fcffc 0.0s
=> => exporting config sha256:32c314009dfcc74d937177a26c98b8e0bf64db67e2f3188a1d4fc12025941f 0.0s
=> => exporting attestation manifest sha256:18822b3c1eb5cce3920e52efe127ff9f534ff4905f54001ecd96f37fe3be10e371 0.0s
=> => exporting manifest list sha256:27c947c3d1bf40e365a266d1ce8aa3ac43648a00c0990cba44aba6b64de6635 0.0s
=> => naming to docker.io/javi2332/pps_p_gold_image_javlluapa:latest        0.0s
=> => unpacking to docker.io/javi2332/pps_p_gold_image_javlluapa:latest       0.1s
```

Imagen 56. Creación imagen definitiva.

```
javier@PC-LINUX-JAVIER:~/PPS/RA3/RA3_1/PRACTICA_GOLD_IMAGE$ docker run -d -p 8080:80 -p 8081:443 --name golden javi2332/pps_p_gold_image_javlluapa:latest
d2a9209c1f45f049fde805064733c43006f47717f1a0c79c8bd2e61ea5843fe6
javier@PC-LINUX-JAVIER:~/PPS/RA3/RA3_1/PRACTICA_GOLD_IMAGE$ docker ps
CONTAINER ID IMAGE COMMAND NAMES CREATED STATUS PORTS
d2a9209c1f45 javi2332/pps_p_gold_image_javlluapa:latest "apache2ctl -D FOREG..." 2 seconds ago Up 2 seconds 0.0.0.0:80->80/tcp, [::]:8080->80/tcp, 0.0.0.0:8081->443/tcp, [::]:8081->443/tcp golden
```

Imagen 57. Arrancar contenedor y comprobar funcionamiento.

Finalmente, se procede con la validación de seguridad del servidor para ello se han realizado diferentes pruebas que permiten confirmar la robustez del servidor:

- En primer lugar, se verifica el banner personalizado ocultando la versión y sistema operativo, también se muestra la aplicación frente a ataques xss y otros parámetros configurados.
 - Tras esto se verifica que el servidor rechaza o ignora la petición según las reglas de reescritura configuradas.
 - Se comprueba también que los procesos hijos se ejecutan bajo el usuario apache.
 - Finalmente se comprueban los certificados.

AUDITORÍA TI NOVATECH

```
javier@PC-LINUX-JAVIER:~/PPS/RA3/RA3_1/PRACTICA_GOLD_IMAGE$ curl -I -k https://www.midominioseguro.com:8081
HTTP/1.1 200 OK
Date: Sun, 18 Jan 2026 11:58:57 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubDomains
X-Frame-Options: SAMEORIGIN
Last-Modified: Sat, 17 Jan 2026 20:03:22 GMT
ETag: "29cd-6489af3295680"
Accept-Ranges: bytes
Content-Length: 10701
Vary: Accept-Encoding
Content-Security-Policy: default-src 'self'; img-src *; media-src media1.com media2.com; script-src 'self'
X-XSS-Protection: 1; mode=block
Content-Type: text/html

javier@PC-LINUX-JAVIER:~/PPS/RA3/RA3_1/PRACTICA_GOLD_IMAGE$ curl -I --http1.0 http://www.midominioseguro.com:8080
HTTP/1.1 301 Moved Permanently
Date: Sun, 18 Jan 2026 11:59:02 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubDomains
X-Frame-Options: SAMEORIGIN
Location: https://www.midominioseguro.com/
Connection: close
Content-Type: text/html; charset=iso-8859-1

javier@PC-LINUX-JAVIER:~/PPS/RA3/RA3_1/PRACTICA_GOLD_IMAGE$ docker exec golden ps -ef | grep apache
root      1      0  0 11:55 ?        00:00:00 /bin/sh /usr/sbin/apache2ctl -D FOREGROUND
root     15      1  0 11:55 ?        00:00:00 /usr/sbin/apache2 -D FOREGROUND
apache    16     15  0 11:55 ?        00:00:00 /usr/sbin/apache2 -D FOREGROUND
apache    17     15  0 11:55 ?        00:00:00 /usr/sbin/apache2 -D FOREGROUND
```

Imagen 58. Verificación en www.midominioseguro.com.

```
javier@PC-LINUX-JAVIER:~/PPS/RA3/RA3_1/PRACTICA_GOLD_IMAGE$ curl -I -k https://localhost:8081
HTTP/1.1 200 OK
Date: Sun, 18 Jan 2026 11:55:34 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubDomains
X-Frame-Options: SAMEORIGIN
Last-Modified: Sat, 17 Jan 2026 20:03:22 GMT
ETag: "29cd-6489af3295680"
Accept-Ranges: bytes
Content-Length: 10701
Vary: Accept-Encoding
Content-Security-Policy: default-src 'self'; img-src *; media-src media1.com media2.com; script-src 'self'
X-XSS-Protection: 1; mode=block
Content-Type: text/html

javier@PC-LINUX-JAVIER:~/PPS/RA3/RA3_1/PRACTICA_GOLD_IMAGE$ curl -I --http1.0 http://localhost:8080
HTTP/1.1 301 Moved Permanently
Date: Sun, 18 Jan 2026 11:56:27 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubDomains
X-Frame-Options: SAMEORIGIN
Location: https://www.midominioseguro.com/
Connection: close
Content-Type: text/html; charset=iso-8859-1

javier@PC-LINUX-JAVIER:~/PPS/RA3/RA3_1/PRACTICA_GOLD_IMAGE$ docker exec golden ps -ef | grep apache
root      1      0  0 11:55 ?        00:00:00 /bin/sh /usr/sbin/apache2ctl -D FOREGROUND
root     15      1  0 11:55 ?        00:00:00 /usr/sbin/apache2 -D FOREGROUND
apache    16     15  0 11:55 ?        00:00:00 /usr/sbin/apache2 -D FOREGROUND
apache    17     15  0 11:55 ?        00:00:00 /usr/sbin/apache2 -D FOREGROUND
```

Imagen 59. Verificación en localhost.

AUDITORÍA TI NOVATECH

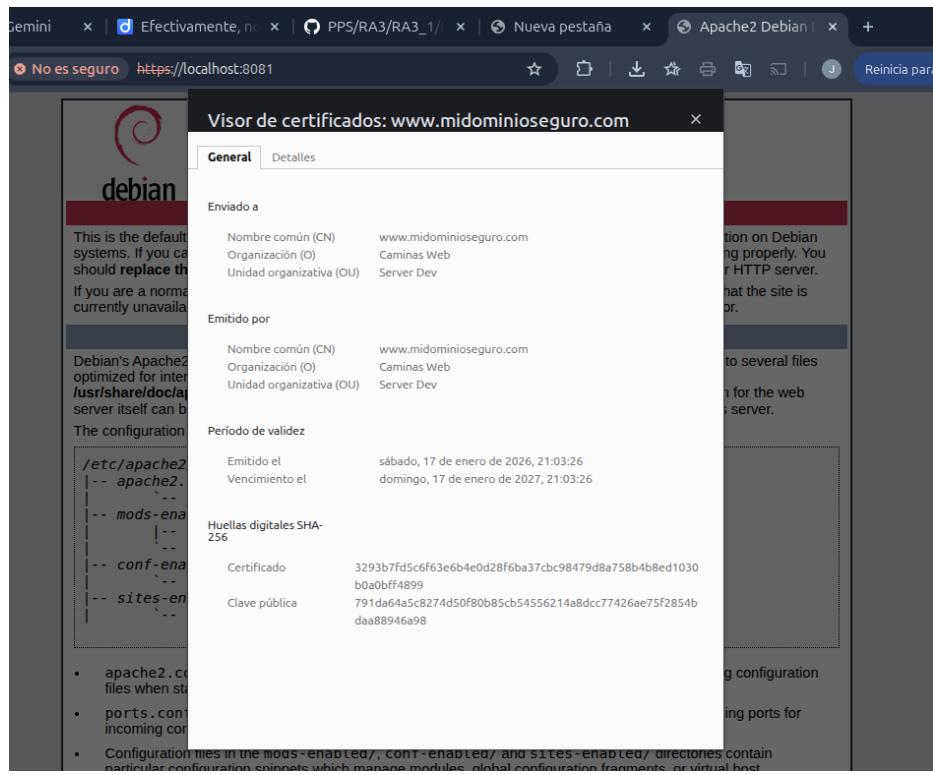


Imagen 60. Certificado aplicado en localhost.

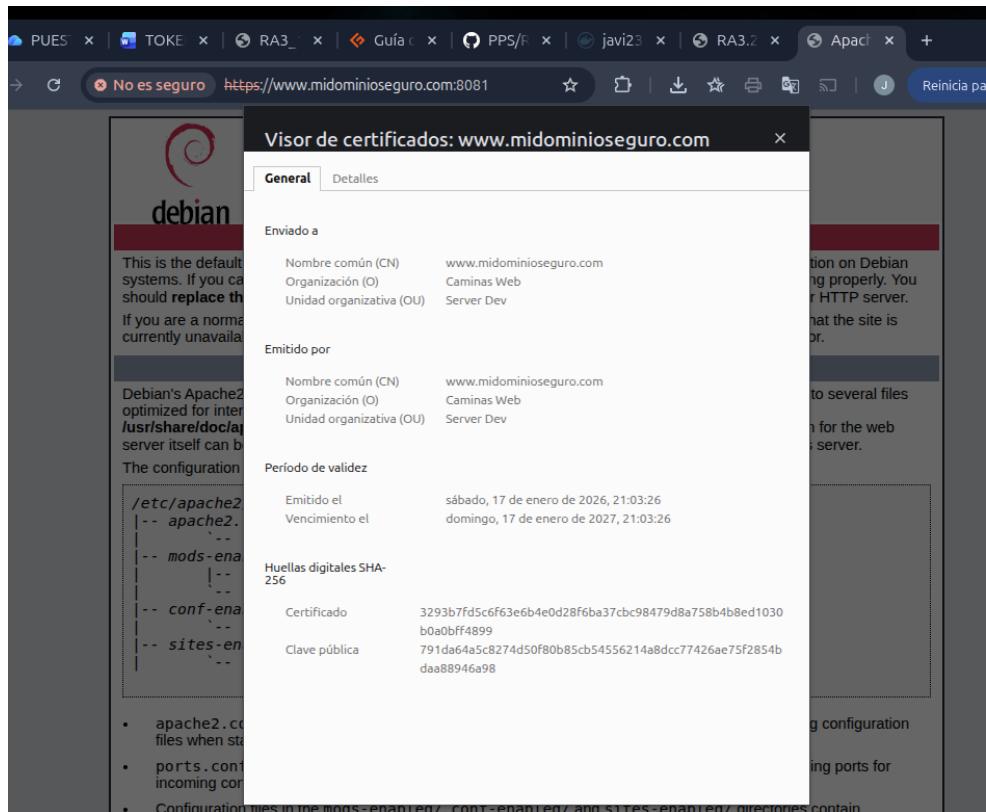


Imagen 61. Certificado aplicado en www.midominioseguro.com.

Estrategia de Capas (Dockerfile Versión Final de Cada Práctica)

Aquí finalmente se muestra como han sido modificados todos los dockerfile para seguir la estrategia de capas en cascada.

A partir del segundo dockerfile, se recoge la primera imagen creada y únicamente se instala el módulo de seguridad y se configura modsecurity.

```
GNU nano 7.2                                            dockerfile
# Heredamos de tu imagen ya subida
FROM javi2332/pps_p1_javlluapa:latest

# Solo instalamos el módulo de seguridad
RUN apt-get update && apt-get install -y \
    libapache2-mod-security2 \
    && apt-get clean

# Configuración de ModSecurity
RUN cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf \
    && sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/' /etc/modsecurity/modsecurity.conf

# No hace falta copiar csp_hsts.conf ni activar SSL, ya está integrado en la P1
EXPOSE 80 443

CMD ["apache2ctl", "-D", "FOREGROUND"]
[]
```

Imagen 61. Segundo dockerfile.

En la tercera únicamente se instala git para poder clonar el repositorio de reglas OWASP CRS y se descargan y configuran.

```
GNU nano 7.2                                            dockerfile
# 1. Heredamos de la imagen de la práctica 2
FROM javi2332/pps_p2_javlluapa:latest

# 2. Solo instalamos GIT (lo único que nos falta en esta capa para bajar OWASP)
RUN apt-get update && apt-get install -y \
    git \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# 3. Descargamos las Reglas OWASP CRS
# No hace falta configurar ModSecurity de nuevo, ya viene activo de la P2.
RUN git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git /tmp/owasp-crs \
    && mv /tmp/owasp-crs/crs-setup.conf.example /etc/modsecurity/crs-setup.conf \
    && mv /tmp/owasp-crs/rules /etc/modsecurity/rules \
    && rm -rf /tmp/owasp-crs

# 4. Copiamos la configuración específica de esta práctica
# IMPORTANTE: El archivo security2.conf activa las reglas del punto anterior
COPY security2.conf /etc/apache2/mods-available/security2.conf

# NOTA: No hace falta copiar csp_hsts.conf ni hacer hardening,
# ya lo heredamos de la P1 que está dentro de la P2.

EXPOSE 80 443
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Imagen 63. Tercer dockerfile.

AUDITORÍA TI NOVATECH

Para el cuarto, únicamente se instala el módulo mod-evasive y se configura, tras esto se activa el módulo y listo.

```
GNU nano 7.2                                            dockerfile
# 1. Heredamos de la Práctica 3 (que ya tiene Hardening + ModSec + OWASP)
FROM javi2332/pps_p3_javlluapa:latest

# 2. Solo instalamos el módulo que nos falta: mod-evasive
RUN apt-get update && apt-get install -y \
    libapache2-mod-evasive \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# 3. Configuración de mod_evasive
# Creamos el directorio de logs (necesario para que funcione)
RUN mkdir -p /var/log/mod_evasive && chown -R www-data:www-data /var/log/mod_evasive

# Copiamos el archivo de configuración específico
COPY evasive.conf /etc/apache2/mods-available/evasive.conf

# Copiamos el script de prueba a /root/ para tenerlo a mano
RUN cp /usr/share/doc/libapache2-mod-evasive/examples/test.pl /root/test.pl

# 4. Activamos el módulo
RUN a2enmod evasive

# No hace falta copiar csp_hsts.conf, ni security2.conf, ni hacer hardening, ya viene dentro de la imagen de la Práctica3

EXPOSE 80 443
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Imagen 64. Cuarto dockerfile.

En el siguiente en referencia a los certificados, únicamente se añade la creación de estos.

```
GNU nano 7.2                                            dockerfile
# Ahora hereda de la P4, por lo que ya tiene ModSecurity, OWASP y Evasive
FROM javi2332/pps_p4_javlluapa:latest

# Solo creamos el certificado, el resto ya se hereda
RUN mkdir -p /etc/apache2/ssl && \
    openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout /etc/apache2/ssl/apache.key \
    -out /etc/apache2/ssl/apache.crt \
    -subj "/C=ES/ST=Castellon/L=Castellon/O=Seguridad/CN=www.midominioseguro.com"

COPY default-ssl.conf /etc/apache2/sites-available/default-ssl.conf

RUN a2enmod ssl && \
    a2ensite default-ssl.conf && \
    a2dissite 000-default.conf

EXPOSE 80 443
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Imagen 65. Certificado dockerfile.

En la última imagen, se crea nuevo usuario apache y configuran permisos, se fuerza la activación de módulos y se aplica el hardening faltante visto en el artículo que no es aplicado con lo ya hecho en anteriores imágenes. Se aplican además algunos parches debido a errores producidos en la cascada al arrastrar tantas configuraciones.

```
GNU nano 7.2                                            dockerfile
FROM javi2332/pps_p_certificados_javlluapa:latest

# 1. Crear usuario y grupo apache
RUN groupadd apache && useradd -g apache apache

# 2. Forzar activación de módulos (por si acaso)
RUN a2enmod headers rewrite

# 3. Aplicar el hardening del artículo directamente en el archivo principal
# Esto evita el error de "Invalid command Header" al cargar después del módulo
RUN echo "ServerTokens Prod" >> /etc/apache2/apache2.conf && \
    echo "ServerSignature Off" >> /etc/apache2/apache2.conf && \
    echo "TraceEnable Off" >> /etc/apache2/apache2.conf && \
    echo "Header always set Strict-Transport-Security \"max-age=63072000; includeSubDomains\"" >> /etc/apache2/apache2.conf && \
    echo "Header set Content-Security-Policy \"default-src 'self'; img-src *; media-src media1.com media2.com; script-src 'self'" >> /etc/apache2/apache2.conf && \
    echo "Header always append X-Frame-Options SAMEORIGIN" >> /etc/apache2/apache2.conf && \
    echo "Header set X-XSS-Protection \"1; mode=block\"" >> /etc/apache2/apache2.conf

# 4. Arreglar permisos para el nuevo usuario 'apache'
# IMPORTANTE: Necesita escribir en logs de ModSec y Evasive heredados
RUN mkdir -p /var/log/mod_evasive /var/cache/modsecurity && \
    chown -R apache:apache /var/log/apache2 /var/log/mod_evasive /var/cache/modsecurity /var/www/html && \
    chmod -R 750 /etc/apache2/ /usr/sbin/apache2

# 5. Cambiar el usuario de ejecución en el entorno de Apache
RUN sed -i 's/export APACHE_RUN_USER=.*$/export APACHE_RUN_USER=apache/' /etc/apache2/envvars && \
    sed -i 's/export APACHE_RUN_GROUP=.*$/export APACHE_RUN_GROUP=apache/' /etc/apache2/envvars

EXPOSE 80 443
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

Imagen 66. Dockerfile gold image.

3. Bibliografía y otras fuentes consultadas

Para el desarrollo de este proyecto se han consultado estándares de la industria y documentación técnica oficial de las herramientas utilizadas:

- **OWASP (Open Web Application Security Project):** Guías de mitigación para el *Top 10* de vulnerabilidades web y documentación del *Core Rule Set (CRS)* para ModSecurity.
 - o <https://owasp.org>
- **Documentación Oficial de Apache HTTP Server (v2.4):** Manuales sobre módulos de seguridad (mod_ssl, mod_headers, mod_rewrite).
 - o <https://httpd.apache.org/docs/2.4/>
- **Docker Documentation:** Mejores prácticas para la creación de imágenes seguras y gestión de usuarios no privilegiados en contenedores.
 - o <https://docs.docker.com>
- **Mozilla Observatory:** Guías de implementación para cabeceras de seguridad (HSTS, CSP, X-Frame-Options).
 - o <https://infosec.mozilla.org/>
- **ModSecurity v3 Reference Manual:** Configuración del motor de inspección de tráfico y filtrado de peticiones.
 - o <https://github.com/SpiderLabs/ModSecurity>

4. Conclusión

La realización de este proyecto demuestra que la seguridad de un servidor web no depende de una única herramienta, sino de la correcta integración de múltiples controles técnicos. La implementación de la Golden Image ha permitido validar que:

- La herencia es clave: El uso de Docker permite construir una cadena de confianza donde cada capa añade valor sin comprometer la estabilidad del sistema.
- El endurecimiento (Hardening) es necesario: Ocultar banners de versión y ejecutar procesos bajo usuarios no privilegiados reduce drásticamente el impacto de posibles exploits.
- La visibilidad es protección: Las cabeceras de seguridad y el WAF no solo bloquean ataques, sino que informan al cliente (navegador) y al administrador sobre el estado de seguridad de la conexión.

En conclusión, este entorno proporciona una base sólida y escalable para desplegar aplicaciones web en entornos de producción, cumpliendo con los estándares actuales de protección de datos y resiliencia ante ataques cibernéticos.