GENERALITAT VALENCIANA
Conselleria d'Educació, Investigació, Cultura i Esport

Institut Educació Secundària El Caminàs

Unió Europea
Fons Social Europeu
El FSE inverteix en el teu futur

Ciclo de Especialización en Ciberseguridad

Puesta en Producción Segura

RA3.1: Server Hardening

Raúl Pastor Clemente

30 de enero de 2026

Objetivos

- Resolución de las actividades propuestas en el enunciado.

Resumen

Autor: Raúl Pastor Clemente

Profesor/a: Pau Conejero

Curso: Especialización FP en Ciberseguridad

Asignatura: Puesta en Producción Segura

Versión: 1.0

Enero-2026

IES El Caminás

Versiones

Versión	Fecha	Cambios
1.0	30/1/2026	Versión inicial del documento

Índice de Contenidos

1. Introducción

En el contexto de la Puesta en Producción Segura, el **hardening** de servidores es un proceso crítico que consiste en **reducir la superficie** de ataque de un sistema mediante la eliminación de vulnerabilidades y la configuración de medidas de seguridad proactivas.

El objetivo de esta práctica es implementar una estrategia de **defensa en profundidad** sobre un servidor web. A lo largo de las actividades, partimos de una instalación básica para ir añadiendo capas de seguridad progresivas: desde la configuración de medidas de seguridad básicas como la configuración de cabeceras de seguridad o la introducción de un cortafuegos de aplicación (WAF), pasando por la generación de certificados digitales personalizados, hasta llegar a la implementación de medidas de seguridad y ofuscación avanzadas.

2. Práctica 1

El objetivo principal de esta tarea ha sido transformar una instalación por defecto de un servidor web en un entorno **robusto y seguro**. Para ello, se ha seguido una estrategia de **defensa en profundidad**, aplicando capas de seguridad progresivas que van desde la configuración básica del servicio hasta la implementación de cortafuegos de aplicación y protección contra ataques masivos.

2.1. Securización en Apache

Esta fase agrupa todas las acciones de *hardening* realizadas sobre el servidor Apache HTTP Server. El proceso se ha llevado a cabo de manera incremental:

1. **Configuración de seguridad básica:** Se configuro la seguridad básica, se habilitó el sitio HTTPS y se ocultó la versión de Apache en las respuestas.
2. **Despliegue de WAF:** Se instaló el módulo **ModSecurity** para proteger la aplicación a nivel de capa 7 y se incluyó una regla de filtrado personalizada.
3. **Instalación del Core Rule Set:** Se instalaron las reglas CRS directamente desde los repositorios oficiales y se configuró su uso en Apache.
4. **Protección Anti-DDoS:** Finalmente, se incorporó el módulo **mod_evasive** para mitigar ataques de Denegación de Servicio.

El resultado es un contenedor Apache que no solo cifra las comunicaciones, sino que rechaza activamente tráfico malicioso y bloquea intentos de saturación.

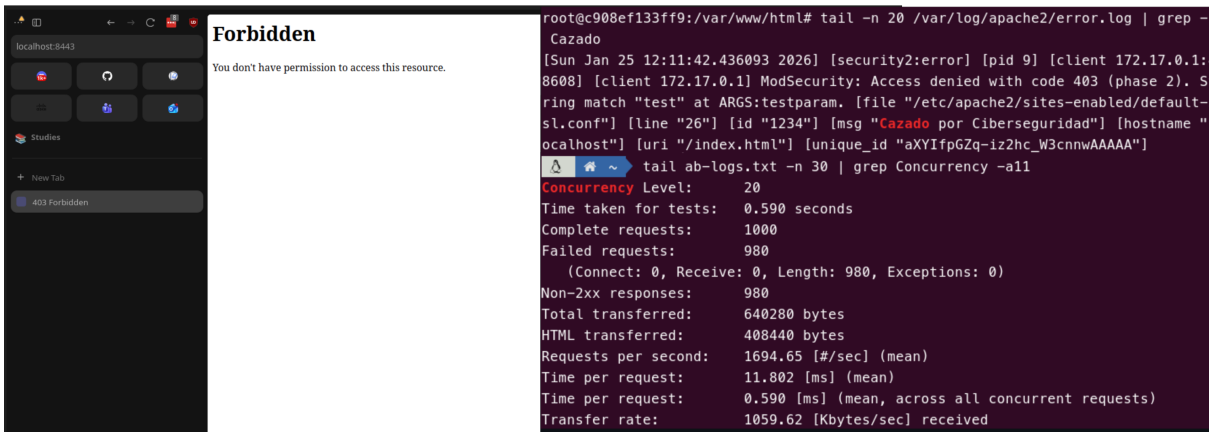

Ilustración 1: *Estado final de actividad 1 (Apache)*

2.2. Securización en Nginx

Para finalizar el primer bloque de ejercicios, se migró la arquitectura de seguridad al servidor **Nginx**. El objetivo fue replicar las funcionalidades logradas en Apache utilizando las herramientas nativas o compatibles con este servidor.

Se instaló Nginx junto con PHP-FPM y se adaptaron las reglas OWASP CRS y las configuraciones de seguridad para funcionar en este entorno. Asimismo, se implementaron medidas nativas de protección DOS en la configuración de Nginx.

Finalmente, se obtuvo un servidor Nginx funcional con un nivel de seguridad equivalente al de Apache, validando que las estrategias de *hardening* son aplicables independientemente de la tecnología del servidor web.

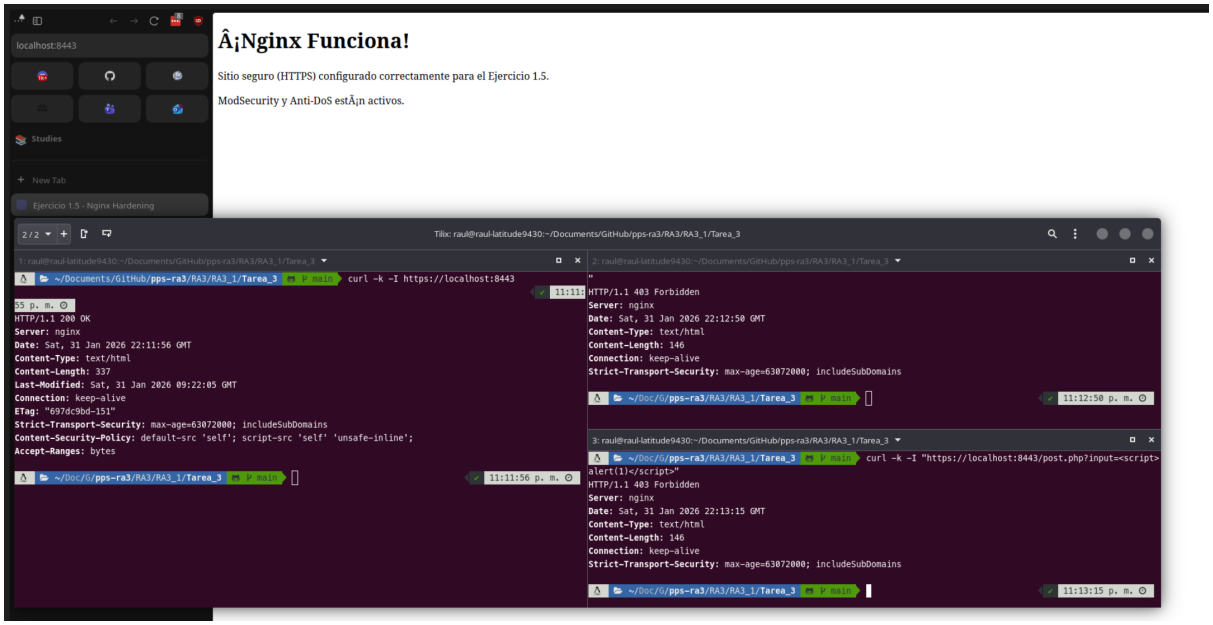

Ilustración 2: *Resultado de práctica 1(Nginx)*

3. Práctica 2

En esta tarea se ha profundizado en la seguridad de la capa de transporte (TLS/SSL), sustituyendo los certificados genéricos por una identidad propia y garantizando que todo el tráfico sea cifrado. Se han realizado dos cambios en la configuración del servidor Apache:

1. **Generación de Certificados Propios:** Se ha utilizado `openssl` para generar una **clave privada** RSA de 2048 bits y un certificado autofirmado (`apache.crt`) definiendo campos personalizados, para sustituir a los certificados "snakeoil" que Apache trae por defecto.
2. **Redirección Forzada a HTTPS:** Se ha modificado el `VirtualHost` del puerto 80 (`000-default.conf`) para incluir una directiva de redirección permanente, obligando a que cualquier petición HTTP sea redirigida automáticamente a HTTPS.

El **resultado final** es que, al acceder al servidor por el puerto estándar, el navegador es redirigido automáticamente a la versión segura. Al inspeccionar el certificado SSL, éste muestra los datos de la organización personalizados.

Ilustración 3: Resultado de la práctica 2

4. Práctica 3

Siguiendo a Geekflare, se aplicó un *hardening* estricto en Apache: deshabilitación de módulos y métodos inseguros (WebDAV, PUT/DELETE), restricción a TLS 1.2+ y bloqueo de HTTP 1.0. Se añadieron cabeceras de seguridad críticas, se ofuscó la firma del servidor y se redujeron los *timeouts*. Además, se blindaron los permisos de configuración y se creó el usuario `apache` para operar con mínimos privilegios.

Para adaptar la solución a Docker, se omitió fijar la IP en la directiva **`Listen`** para mantener la portabilidad del contenedor. Asimismo, el proceso inicia como **`root`** únicamente para vincular los puertos privilegiados (80/443), delegando inmediatamente la ejecución de los procesos de trabajo (*workers*) al usuario restringido **`apache`**.

Ilustración 4: *Resultado final de la práctica 3*

5. Conclusión

Realizar estas prácticas me ha servido para darme cuenta de que la seguridad "por defecto" en los servidores web es prácticamente nula y que es imprescindible dedicar tiempo a configurarlos correctamente antes de salir a producción. Me ha resultado muy útil ver en funcionamiento cómo capas de seguridad que hemos implementado bloquean ataques reales al instante. Además, he aprendido que nada es nunca seguro por completo, sino que la robustez del sistema depende de sumar muchas pequeñas barreras (certificados, permisos, cabeceras y límites) que, en conjunto, se lo ponen mucho más difícil a los atacantes.

6. Bibliografía

1. **Ponz, V.** (2021, 1 de marzo). *Hardening del servidor*. Ciberseguridad PePS. https://psegarrac.github.io/Ciberseguridad-PePS/tema3/seguridad/web/2021/03/01/Hardening-Servidor.html
2. **Ponz, V.** (2020, 8 de noviembre). *P1-SSL*. Ciberseguridad PePS. https://psegarrac.github.io/Ciberseguridad-PePS/tema1/practicas/2020/11/08/P1-SSL.html
3. **Kumar, C.** (2025, 5 de marzo). *Apache Web Server Hardening and Security Guide*. Geekflare. https://geekflare.com/cybersecurity/apache-web-server-hardening-security/
4. **Kumar, C.** (2024, 15 de mayo). *Guía de seguridad y endurecimiento del servidor web Nginx*. Geekflare. https://geekflare.com/es/nginx-webserver-security-hardening-guide/