OWASP Juice Shop

Web Application Penetration Test Report

Author: Chandu reddy
Date: (2025-08-19)
Target: OWASP Juice Shop (local/test instance)

Table of Contents

1. Executive Summary

This engagement performed a focused web application penetration test of an OWASP Juice Shop instance. The assessment identified five (5) security issues of varying severity that could be exploited by unauthorized users to access data, execute arbitrary scripts in victim browsers, or bypass authorization. These findings are actionable and include recommended remediations.

---

2. Scope & Test Environment

- **Target:** OWASP Juice Shop (local VM).
- **Base URL:** (e.g. http://localhost:3000)
- **Tester:** Chandu reddy
- **Date(s) of testing:** (2025-09-21)
- **Authorization:** Test instance / lab — no production systems targeted.

---

3. Methodology & Tools

Methodology: Manual testing backed by standard web-application pentest techniques: input fuzzing, parameter tampering, authentication checks, session/token analysis, and DOM inspection.

Tools used (examples): - Burp Suite (Proxy, Intruder, Repeater) - Chrome/Firefox Developer Tools - sqlmap (for confirmation, optional) - curl / https - Browser (to demonstrate XSS)

4. Findings (detailed)

4.1 Finding 1 — Stored Cross- Site Scripting (XSS)

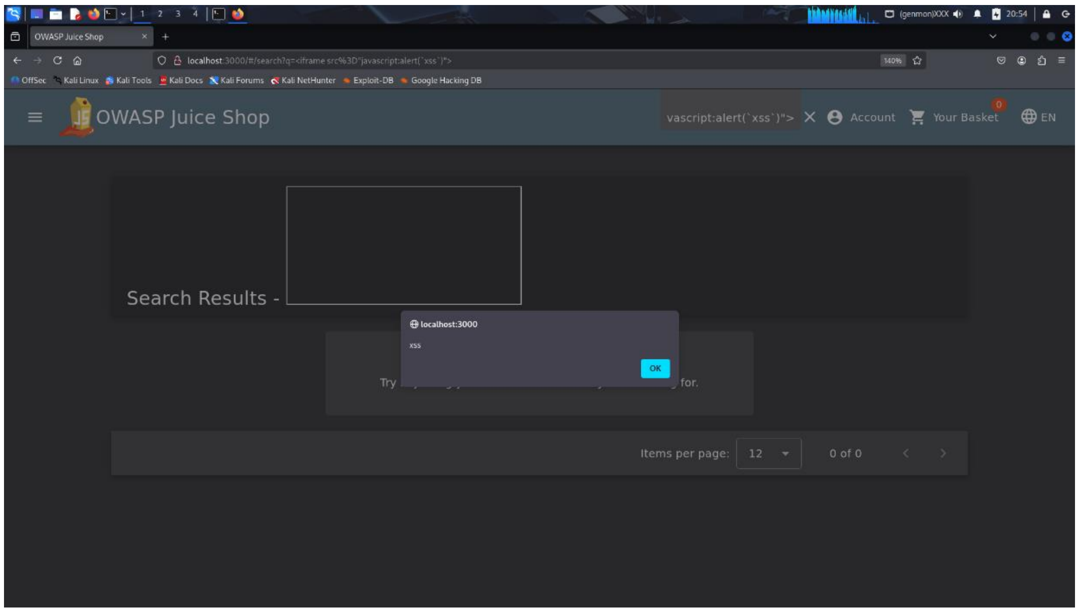
- **Severity:** High
- **Affected area:** Product reviews / customer feedback input fields
- **Description:** User-supplied input is stored and rendered without sufficient output encoding, allowing JavaScript payloads to execute in the context of other users who view the content.

Reproduction Steps: 1. Log in or use a test account. 2. Navigate to a product page and add a review/comment. 3. Submit a payload such as `<script>alert(document.cookie)</script>` (example payload — do not use in production). 4. Visit the product page while logged in as another user or refresh the page; the script executes.

Impact: An attacker can run arbitrary JavaScript in victims' browsers, leading to cookie theft, session hijacking, UI redress, or performing actions on behalf of the user.

Remediation: - Perform proper output encoding/escaping on all user-supplied data before rendering in HTML contexts. - Use a well-tested templating engine or a proven XSS protection library. - Implement a Content Security Policy (CSP) as defense-in-depth.

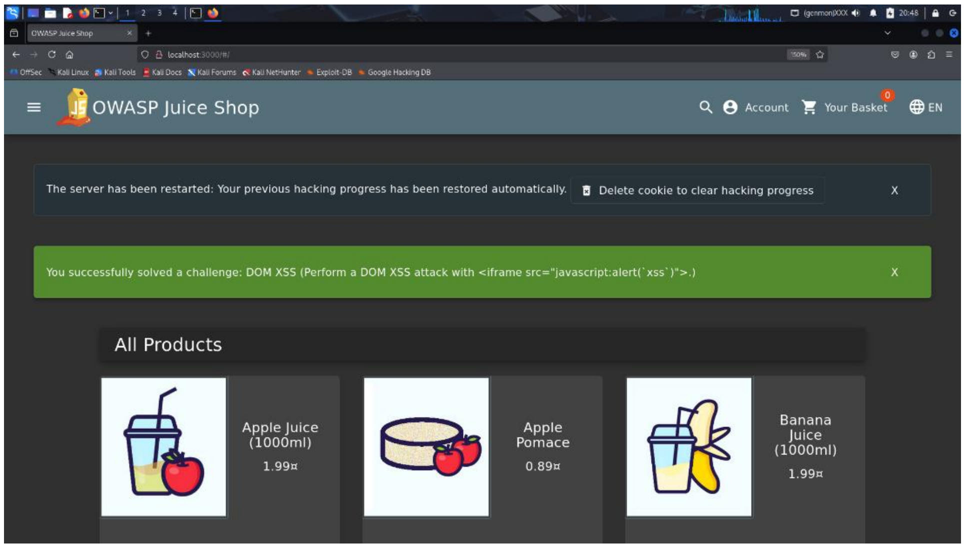
Suggested Priority: Patch input handling and implement CSP — **High**.



```
localhost:3000/#/search?q=<iframe src%3D"javascript:alert(`xss`)">
```

```
vascript:alert(`xss`)">
```

```
localhost:3000
xss
```



The server has been restarted: Your previous hacking progress has been restored automatically.

You successfully solved a challenge: DOM XSS (Perform a DOM XSS attack with <iframe src="javascript:alert(`xss`)">.)

4.2 Finding 2 — SQL Injection (Search / Login)

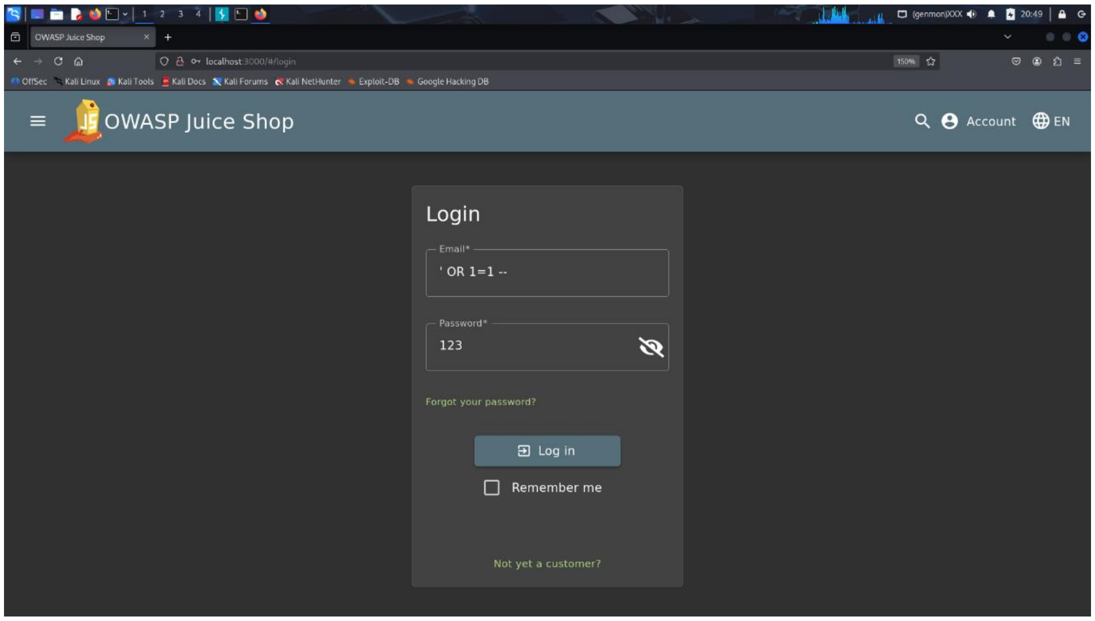
- **Severity:** High
- **Affected area:** Search box and/or authentication endpoints where inputs are concatenated into SQL queries.
- **Description:** Unsanitized input is used within SQL queries, allowing boolean/logical or UNION-based injection to alter query behavior and extract data.

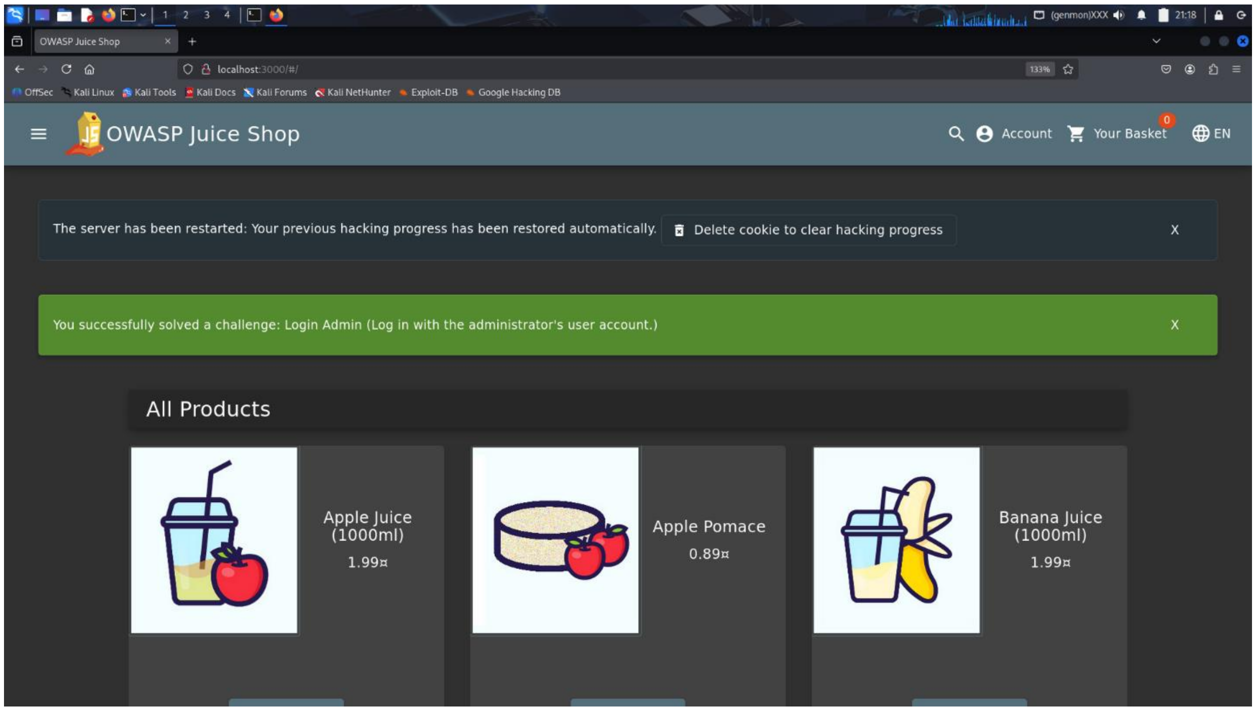
Reproduction Steps: 1. In the search field, inject a test payload like `' OR 1=1 --` (example) and observe results that differ from expected search behavior. 2. For login forms, attempt classic payloads in username/password fields and observe authentication bypass or error messages. 3. Confirm by sending crafted requests via Burp Repeater.

Impact: Full database exposure, account enumeration, authentication bypass, or data exfiltration.

Remediation: - Use parameterized queries / prepared statements for all DB access. - Implement input validation and least-privilege DB accounts. - Enable query logging and anomaly detection for suspicious queries.

Suggested Priority: Immediate — **High**.





The server has been restarted: Your previous hacking progress has been restored automatically. Delete cookie to clear hacking progress

You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)

4.3 Finding 3 — Insecure Direct Object Reference (IDOR) / Broken Access Control

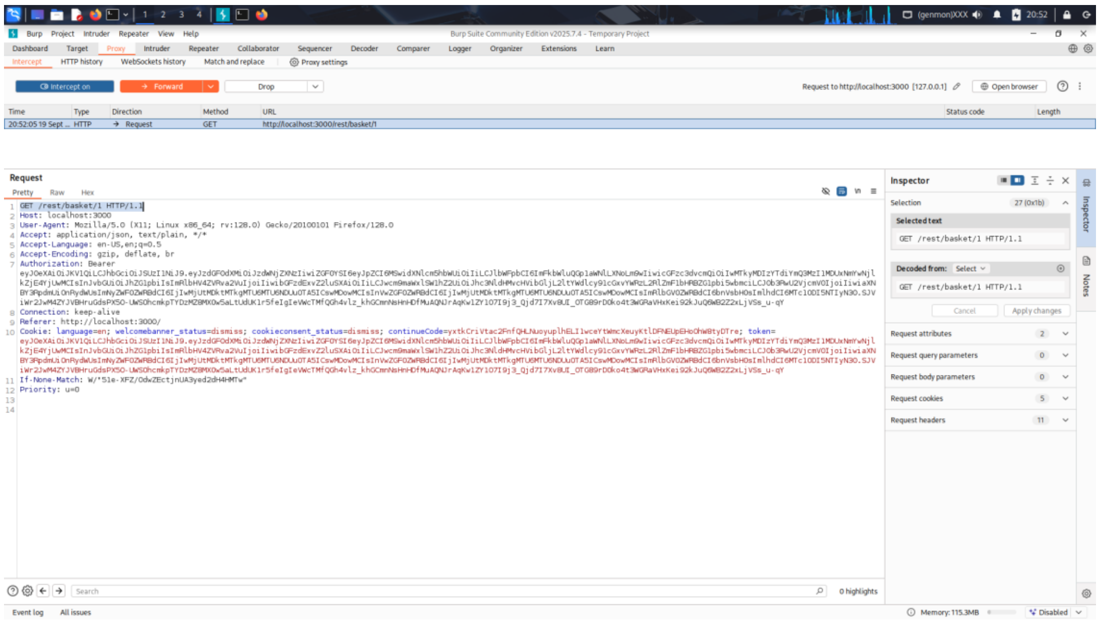
- **Severity:** High
- **Affected area:** Order/basket endpoints, user profile resources
- **Description:** Direct object identifiers (IDs) in URLs or API parameters are not properly validated against the authenticated user, allowing one user to access or modify another user's resources.

Reproduction Steps: 1. Log in as user A and create an order or save a wishlist. 2. Observe the resource ID in the request/URL (e.g., `/api/orders/1234`). 3. While authenticated as user B (or unauthenticated if possible), request `/api/orders/1234`. 4. If the server returns the resource or allows modification, the server lacks proper authorization checks.

Impact: Disclosure or modification of other users' orders, personal data, or performing actions on their behalf.

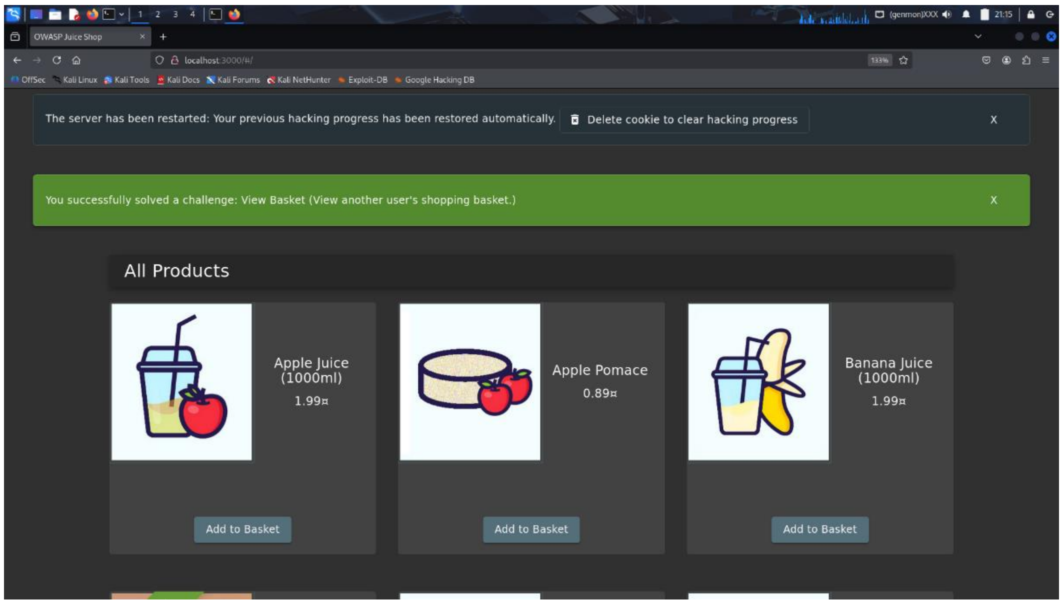
Remediation: - Enforce server-side authorization checks: verify resource ownership for every request. - Avoid exposing sensitive sequential IDs; consider using unpredictable UUIDs or mapping to user-scoped resources. - Implement centralized access control logic.

Suggested Priority: High.



Time	Type	Direction	Method	URL	Status code	Length
20:52:05 19 Sept ...	HTTP	→ Request	GET	http://localhost:3000/rest/basket/1		

```
GET /rest/basket/1 HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjl
kZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXN
BY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjUtMDktMTkgMTU6MTU6NDUuOTA5ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUtMDktMTkgMTU6MTU6NDUuOTA5ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTc1ODI5NTIyN30.SJV
iWr2JwM4ZYJVBHruGdsPX5O-UWS0hcmkpTYDzMZ8MX0w5aLtUdUK1r5feIgIeVWcTMfQGh4vlz_khGCmnNsHnHDfMuAQNJrAqKw1ZY107I9j3_Qjd7I7Xv8UI_0TG89rD0ko4t3WGRaVHxKei92kJuQ6WB2Z2xLjVSs_u-qY
Connection: keep-alive
Referer: http://localhost:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=yxtkCriVtac2FnfQHLNuoyuplhELI1wceYtWmcXeuyKtlDFNEUpEHoOhW8tyDTre; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjl
kZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXN
BY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjUtMDktMTkgMTU6MTU6NDUuOTA5ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUtMDktMTkgMTU6MTU6NDUuOTA5ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTc1ODI5NTIyN30.SJV
iWr2JwM4ZYJVBHruGdsPX5O-UWS0hcmkpTYDzMZ8MX0w5aLtUdUK1r5feIgIeVWcTMfQGh4vlz_khGCmnNsHnHDfMuAQNJrAqKw1ZY107I9j3_Qjd7I7Xv8UI_0TG89rD0ko4t3WGRaVHxKei92kJuQ6WB2Z2xLjVSs_u-qY
If-None-Match: W/"51e-XFZ/OdwZEctjnUA3yed2dH4HMTw"
Priority: u=0
```



The server has been restarted: Your previous hacking progress has been restored automatically.

You successfully solved a challenge: View Basket (View another user's shopping basket.)

4.4 Finding 4 — Broken Authentication (Weak Password Reset / Account Takeover)

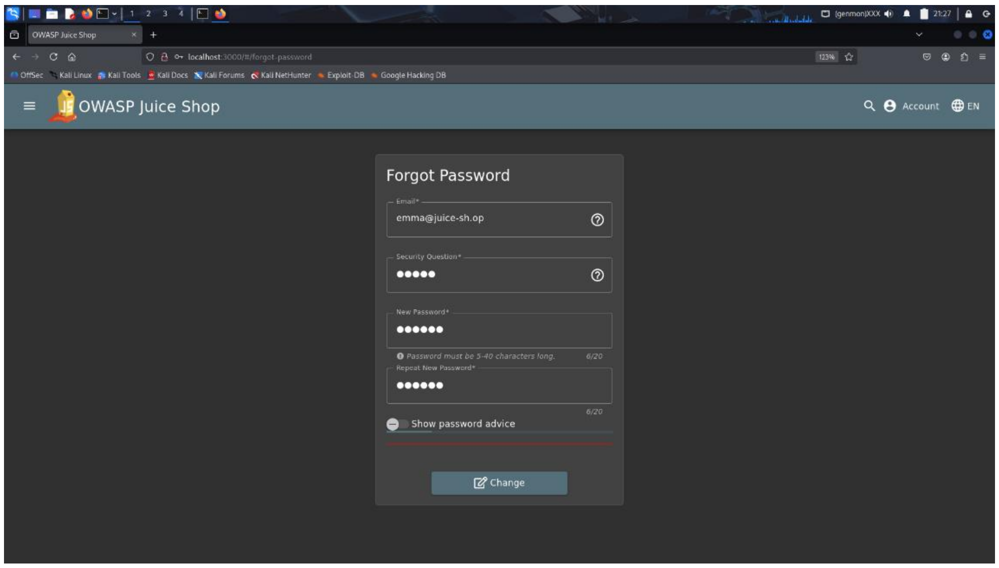
- **Severity:** Medium
- **Affected area:** Password reset workflow, session management
- **Description:** The password reset token or recovery workflow is predictable, not time-limited, or tokens are not invalidated after use. Session fixation or weak session invalidation may also be present.

Reproduction Steps: 1. Trigger the password reset flow for a test account and inspect the token mechanism (email, URL parameter, local logs). 2. Try reusing tokens after reset or guessing token values if short/predictable. 3. Check whether sessions are invalidated after password change.

Impact: An attacker may reset user passwords or hijack accounts, particularly if user emails are accessible or tokens predictable.

Remediation: - Use cryptographically secure, long, single-use tokens with short expiry. - Invalidate tokens after use and rotate sessions after password changes. - Rate-limit reset requests and monitor for abuse.

Suggested Priority: Medium — raise to high if tokens are predictable or not single-use.


OWASP Juice Shop

localhost:3000/#/forgot-password

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

OWASP Juice Shop Account EN

Forgot Password

Email*
emma@juice-sh.op

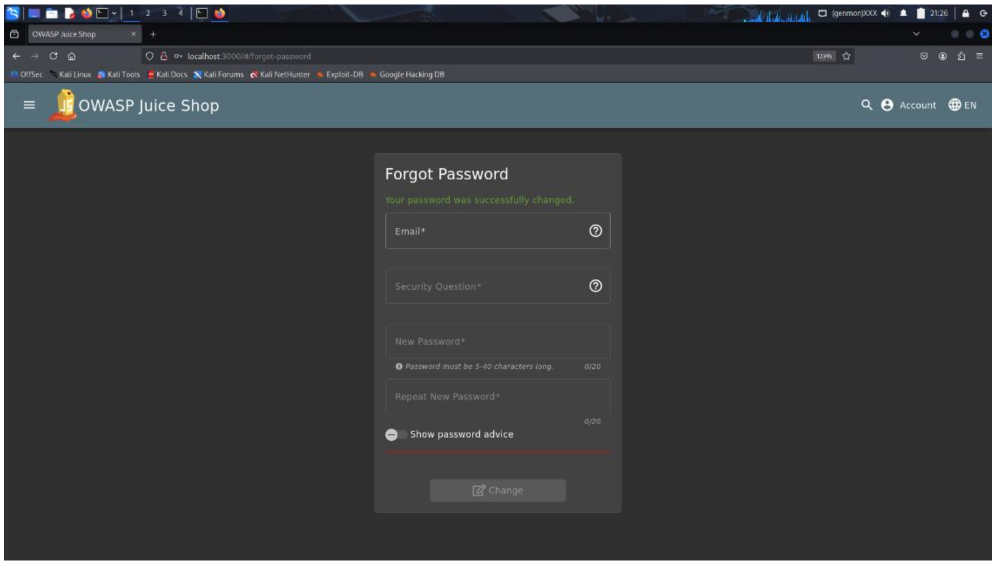
Security Question*
●●●●●

New Password*
●●●●●●
Password must be 5-40 characters long. 6/20

Repeat New Password*
●●●●●●
6/20

Show password advice

Change


OWASP Juice Shop

localhost:3000/#/forgot-password

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

OWASP Juice Shop Account EN

Forgot Password

Your password was successfully changed.

Email*

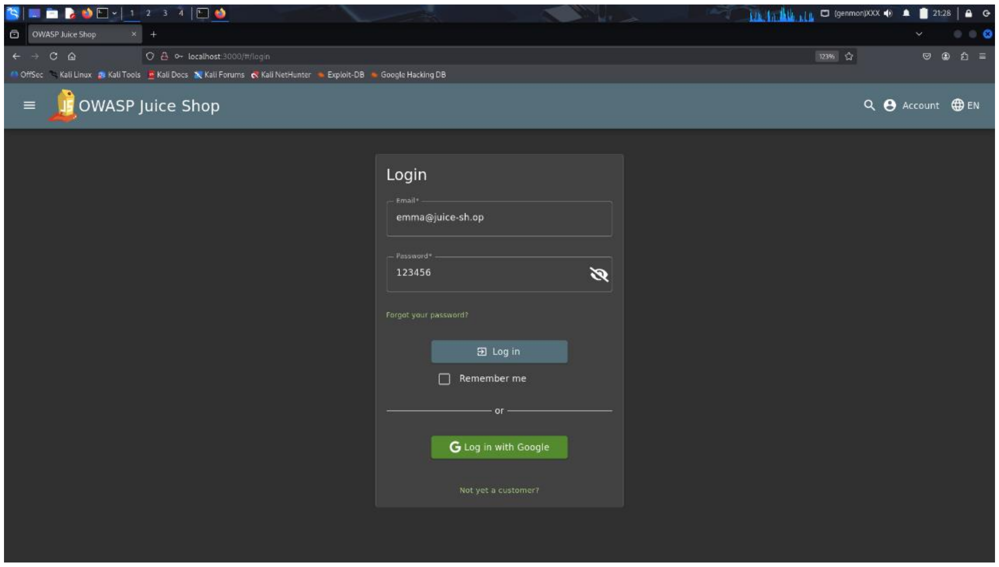
Security Question*

New Password*
Password must be 5-40 characters long. 0/20

Repeat New Password*
0/20

Show password advice

Change


OWASP Juice Shop

localhost:3000/#/login

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

OWASP Juice Shop Account EN

Login

Email*
emma@juice-sh.op

Password*
123456

Forgot your password?

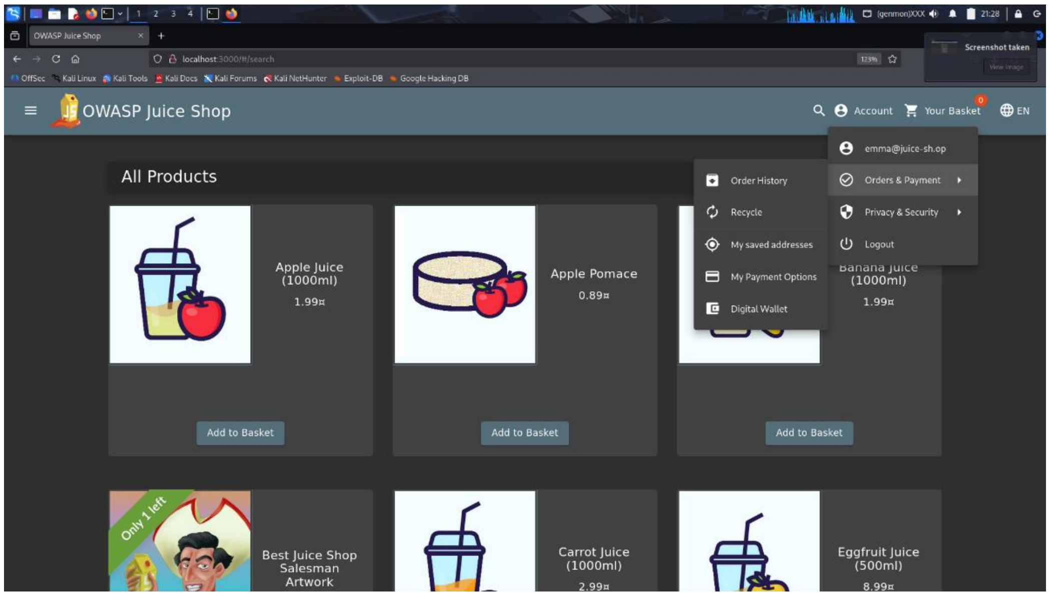
Log in

Remember me

or

Log in with Google

Not yet a customer?



4.5 Finding 5 — Sensitive Data Exposure (Tokens in localStorage, secrets stored client-side)

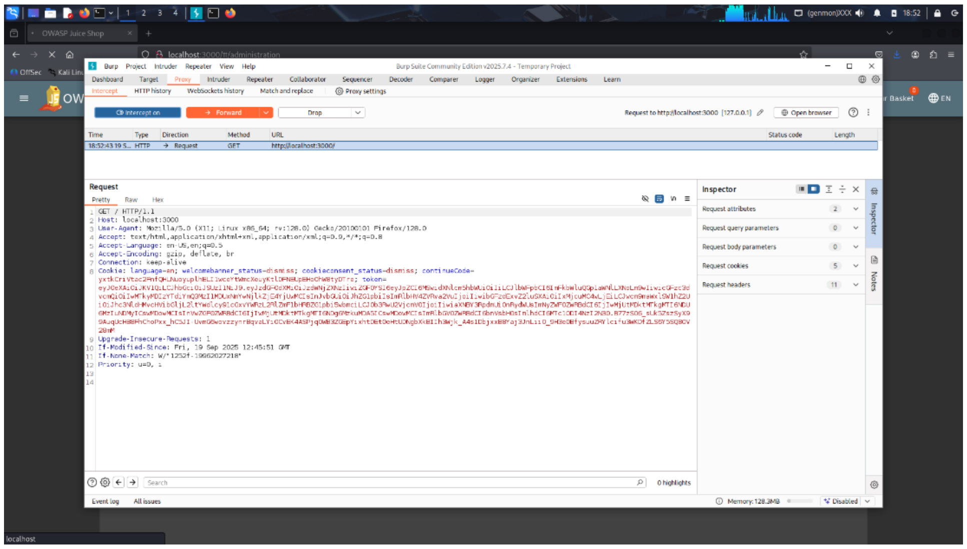
- **Severity:** Medium
- **Affected area:** Client storage (localStorage/sessionStorage) and logs
- **Description:** Authentication tokens, secrets, or sensitive user data are stored in client-side storage where they may be accessed by XSS or other attack vectors.

Reproduction Steps: 1. Log in and inspect browser storage (localStorage/sessionStorage) and cookies. 2. Observe any JWTs, session tokens, or plaintext secrets stored client-side.

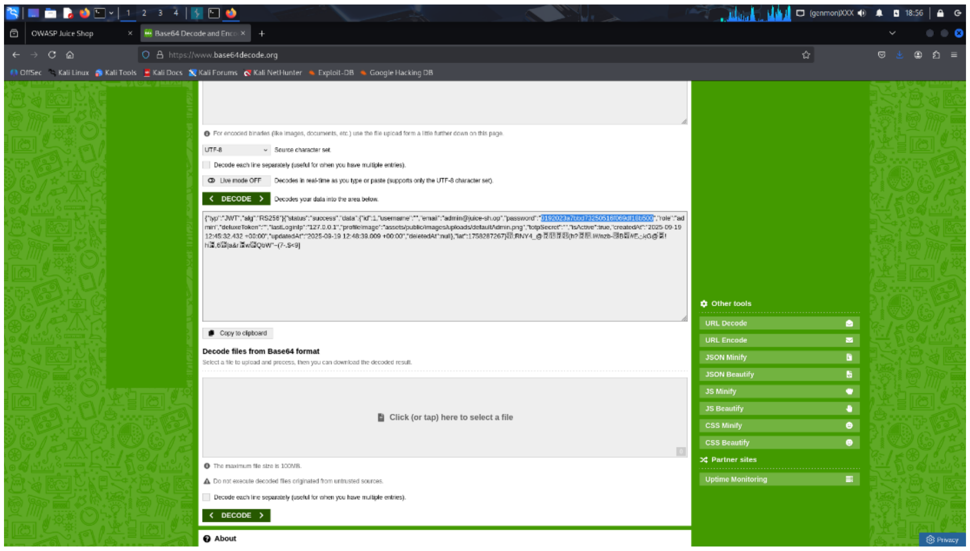
Impact: Combined with an XSS vulnerability, an attacker can steal tokens and impersonate the user. Persistent storage of tokens increases the attack surface.

Remediation: - Avoid storing sensitive tokens in localStorage. Use httpOnly, Secure cookies with sameSite attributes for session tokens. - Minimize sensitive data stored on the client. - Implement short token lifetimes and refresh tokens stored securely.

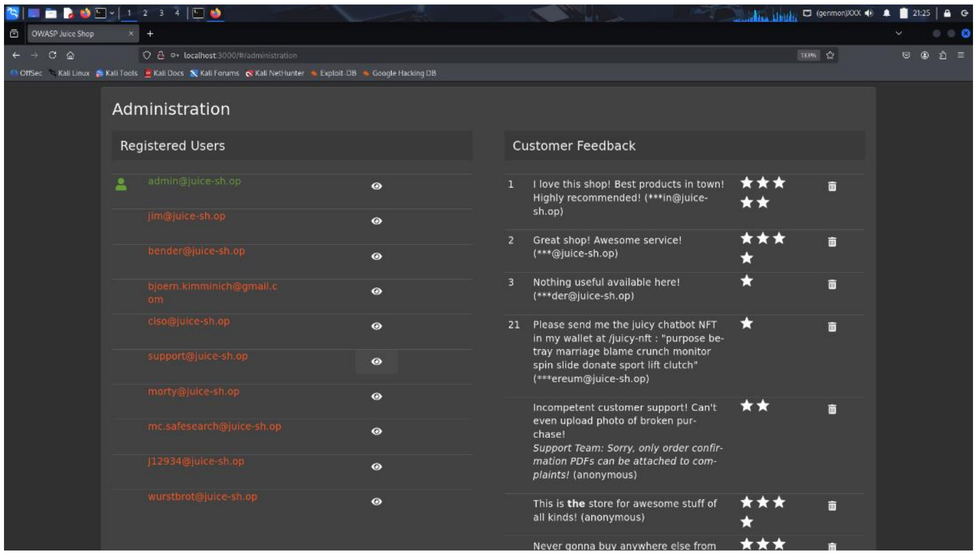
Suggested Priority: Medium.



```
GET / HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=
yxtkCriVtac2FnfQHLNuoyupLhELI1wceYtWmcXeuyKtlDFNEUpEHoChW8tyDTrs; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3d
vcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxMjcuMC4wLjEiLCJwcm9maWxlSW1hZ2U
iOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjUtMDktMTkgMTI6NDU
6MzIuNDMyICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUtMDktMTkgMTI6NDg6MzkuMDA5ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTc1ODI4NzI2N30.B77zS06_sUkSZszSyX9
9AuqUcHBBFhChoPxx_hC5JI-UvmG6wovzzynrBqvaLYiGCvEK4ASPjqOWB3ZGEpYixhtDEtDeHtUDNgbXkBIIh3Wjk_A4s1DbjxxBBYaj3JnLii0_SH3e0BfysuuZRYlcifu3WKDfZLS6Y5SQBOV
28mM
Upgrade-Insecure-Requests: 1
If-Modified-Since: Fri, 19 Sep 2025 12:45:51 GMT
If-None-Match: W/"1252f-19962027218"
Priority: u=0, i
```



```
{"typ":"JWT","alg":"RS256"}{"status":"success","data":{"id":1,"username":"","email":"admin@juice-sh.op","password":"0192023a7bbd73250516f069df18b500","role":"admin","deluxeToken":"","lastLoginIp":"127.0.0.1","profileImage":"assets/public/images/uploads/defaultAdmin.png","totpSecret":"","isActive":true,"createdAt":"2025-09-19 12:45:32.432 +00:00","updatedAt":"2025-09-19 12:48:39.009 +00:00","deletedAt":null},"iat":1758287267}[illegible]
```



Administration

Registered Users

- admin@juice-sh.op
- jim@juice-sh.op
- bender@juice-sh.op
- bjoern.kimminich@gmail.com
- ciso@juice-sh.op
- support@juice-sh.op
- morty@juice-sh.op
- mc.safesearch@juice-sh.op
- J12934@juice-sh.op
- wurstbrot@juice-sh.op

Customer Feedback

1	I love this shop! Best products in town! Highly recommended! (***in@juice-sh.op)	★★★★★
2	Great shop! Awesome service! (***@juice-sh.op)	★★★★
3	Nothing useful available here! (***der@juice-sh.op)	★
21	Please send me the juicy chatbot NFT in my wallet at /juicy-nft : "purpose betray marriage blame crunch monitor spin slide donate sport lift clutch" (***ereum@juice-sh.op)	★
	Incompetent customer support! Can't even upload photo of broken purchase! *Support Team: Sorry, only order confirmation PDFs can be attached to complaints!* (anonymous)	★★
	This is **the** store for awesome stuff of all kinds! (anonymous)	★★★★
	Never gonna buy anywhere else from	★★★

5. Overall Risk Summary & Recommendations

- **Highest priority:** Fix SQL Injection, Stored XSS, and IDOR immediately.
- **Medium priority:** Harden authentication flows and move sensitive tokens out of client-side storage.
- Implement a secure SDLC practice: input validation, output encoding, parameterized queries, centralized authorization checks, secure session management, and CSP.

---

6. Remediation Checklist (Quick wins)

- Parameterize all DB queries.
- Sanitize and encode output before rendering.
- Enforce server-side authorization for all resource access.
- Use secure, httpOnly cookies for session tokens.
- Implement CSP and set secure cookie attributes.
- Rotate and invalidate tokens correctly; implement rate-limiting.

---

7. Retest Guidance

- Provide fixed endpoints and sample requests that were vulnerable for verification.
- Re-run automated scans and manual verification for each finding.
- For XSS: test with non-trivial payloads and check CSP effectiveness.
- For SQLi: confirm with prepared statements and error suppression.

---

8. Appendix: Evidence, Tools, References

Evidence: Attach screenshots, HTTP request/response logs, and Burp Repeater history demonstrating the exploit.

Tools: Burp Suite, Browser DevTools, sqlmap (if used), curl.

References / Further reading: OWASP XSS Prevention Cheat Sheet, OWASP SQL Injection Prevention Cheat Sheet, OWASP Top 10.

End of report.