

Security Audit Report – Internee.pk

Scope

This penetration test was conducted on the Internee.pk platform to evaluate the security of critical areas including login pages, user profiles, and APIs. The assessment focused on detecting SQL Injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF) vulnerabilities.

Methodology

Testing was performed using OWASP ZAP with both automated and manual techniques:

- **Automated Scanning:** Active Scan on login endpoints (GET /sign-in, POST /sign-in/factor-one) and other captured routes.
- **Manual Fuzzing:** SQLi and XSS payloads injected into login-related requests and cookies.
- **CSRF Testing:** Login requests were replayed without cookies to verify CSRF token enforcement.
- **Security Headers Review:** HTTP responses checked for headers like Strict-Transport-Security and X-Frame-Options.

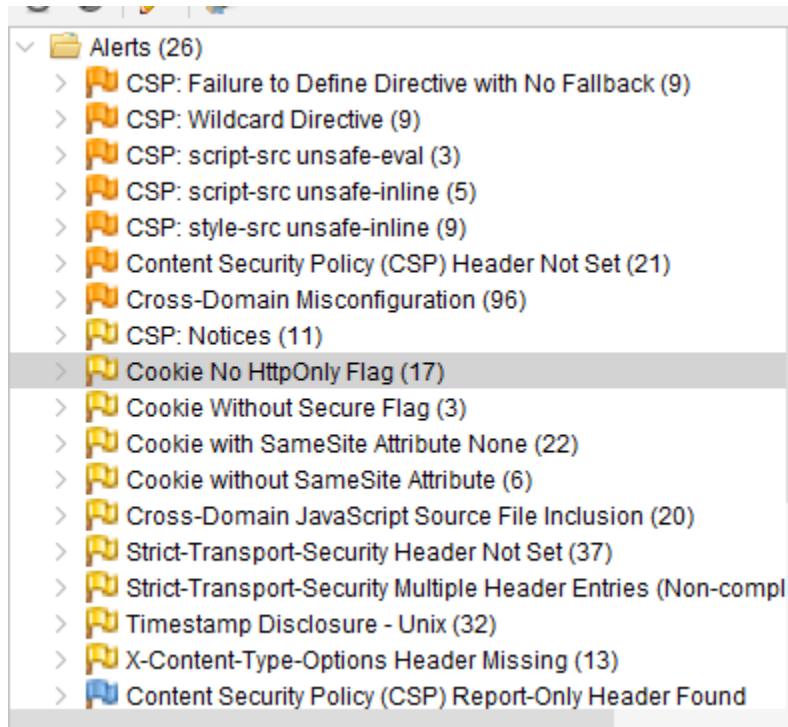
Findings

1. SQL Injection

- **Tested:** By fuzzing login and API parameters.
- **Result:** No evidence of SQL Injection vulnerabilities detected.

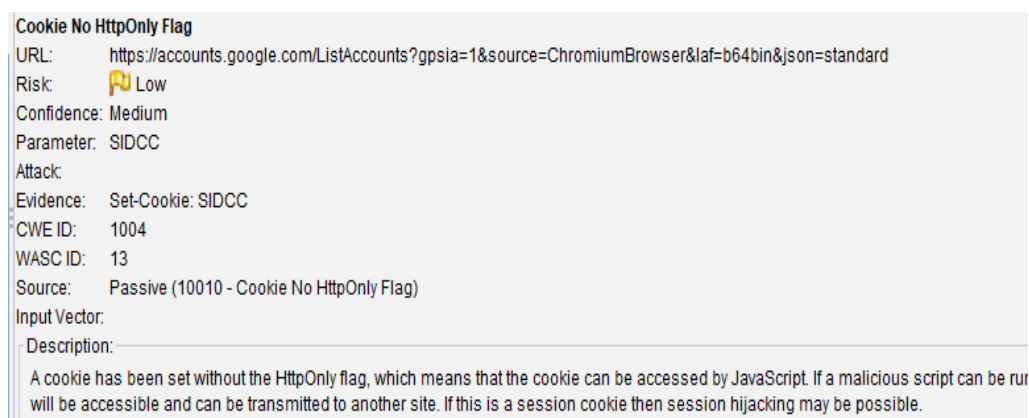
2. Cross-Site Scripting (XSS)

- **Tested:** By injecting script payloads into login and profile fields.
- **Result:** No reflected or stored XSS vulnerabilities detected.



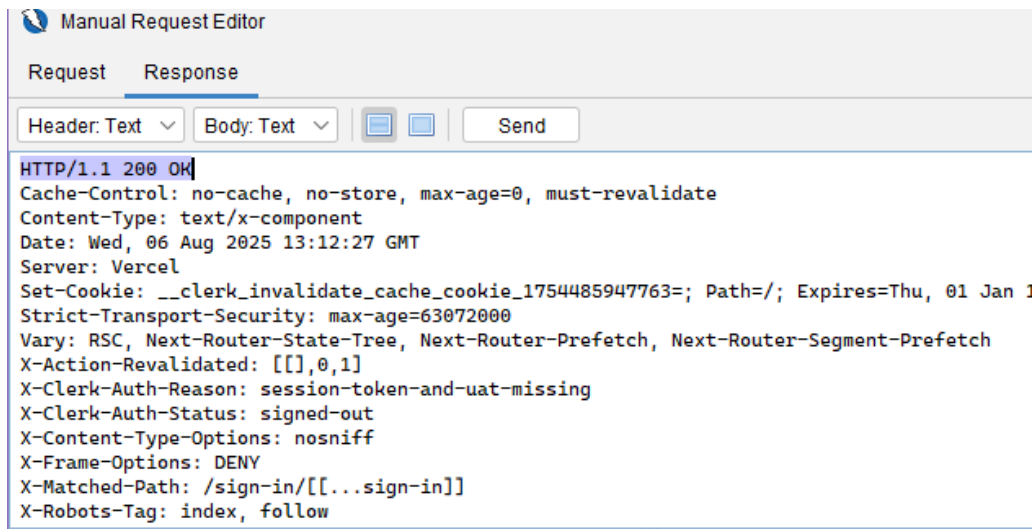
3. Cookies

- **Tested:** By Active scanning of Dashboard
- **Result:** Cookies without secure flag, No http flag



3. Cross-Site Request Forgery (CSRF)

- **Endpoint Tested:** POST /sign-in/factor-one
- **Observation:** Request was re-sent without cookies and still returned HTTP/1.1 200 OK.
- **Risk:** Indicates missing or insufficient CSRF protection.
- **Impact:** An attacker could trick a logged-in user into unknowingly executing authentication-related actions.



The screenshot shows a web application security tool's 'Manual Request Editor' interface. It has two tabs: 'Request' and 'Response', with 'Response' currently selected. Below the tabs are two dropdown menus: 'Header: Text' and 'Body: Text', followed by a 'Send' button. The main area displays the following HTTP response text:

```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Type: text/x-component
Date: Wed, 06 Aug 2025 13:12:27 GMT
Server: Vercel
Set-Cookie: __clerk_invalidate_cache_cookie_1754485947763=; Path=/; Expires=Thu, 01 Jan 1
Strict-Transport-Security: max-age=63072000
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Router-Segment-Prefetch
X-Action-Revalidated: [[],0,1]
X-Clerk-Auth-Reason: session-token-and-uat-missing
X-Clerk-Auth-Status: signed-out
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-Matched-Path: /sign-in/[[...sign-in]]
X-Robots-Tag: index, follow
```

Recommendations

- **Implement CSRF Tokens:** Validate unique tokens on all sensitive POST requests.
- **Set SameSite Cookies:** Use SameSite=Strict or Lax to reduce CSRF risk.
- **Enforce Session Validation:** Reject requests missing valid cookies/tokens.
- **Continue Regular Testing:** Run monthly OWASP ZAP scans.
- **Maintain Secure Coding Practices:** Ensure developers follow OWASP Top 10 guidelines.

Conclusion

The penetration test revealed **no SQL Injection or XSS vulnerabilities**, but identified a **potential CSRF risk** in the login flow (POST /sign-in/factor-one).

Implementing CSRF protection measures is strongly recommended to enhance application security.