TASK 1
WEB APPLICATION SECURITY TESTING

Target Application: OWASP Juice Shop

Prepared By: Aarya Kurhade

Date: 10/12/2025

1. Introduction

This report presents a detailed Web Application Security Testing assessment conducted on the OWASP Juice Shop application. The objective of this task was to identify critical security vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), authentication flaws, and security MISCONFGURATIONS using both manual testing and automated scanning tools.

2. Scope of Assessment

- Score Board
- DOM XSS
- Bonus Payload,
- Cross-Site Scripting
- SQL Injection
- Reflected Cross-Site Scripting (XSS)
- Missing Security Headers
- OWASP ZAP Automated Scan Results

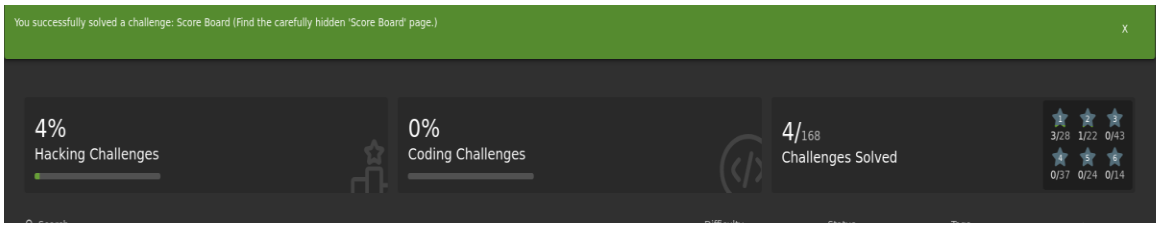
3. Testing Methodology

The security assessment was performed using a combination of manual testing techniques and automated security scanning. Manual testing was carried out by interacting with the application and manipulating inputs. Automated scanning was performed using OWASP ZAP.

4. Identified Vulnerabilities and Analysis

Score Board Exposure

Description
The application exposes a Score Board page that lists all challenges (vulnerabilities). In a real-world system, such information disclosure would provide attackers with a roadmap of weaknesses.


You successfully solved a challenge: Score Board (Find the carefully hidden 'Score Board' page.)

4% Hacking Challenges

0% Coding Challenges

4/168 Challenges Solved

Impact

- Reveals system weaknesses
- Helps attackers identify possible attack paths

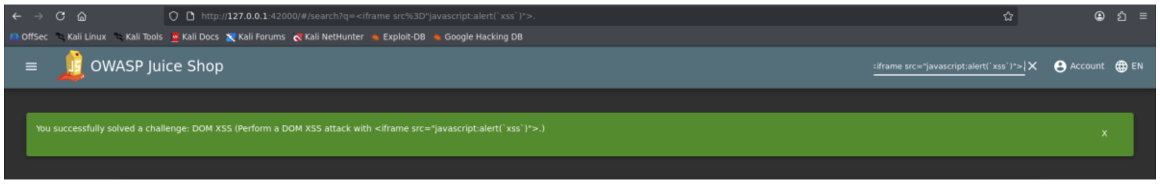
Mitigation

- Restrict access to internal/debug pages
- Implement role-based access control
- Disable such pages in production

DOM-Based XSS

Description

Unvalidated user input is processed on the client side, allowing JavaScript execution directly in the browser.


http://127.0.0.1:42000/#/search?q=<iframe src%3D"javascript:alert(`xss`)">

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

Impact

- Session hijacking
- Unauthorized user actions
- Data theft

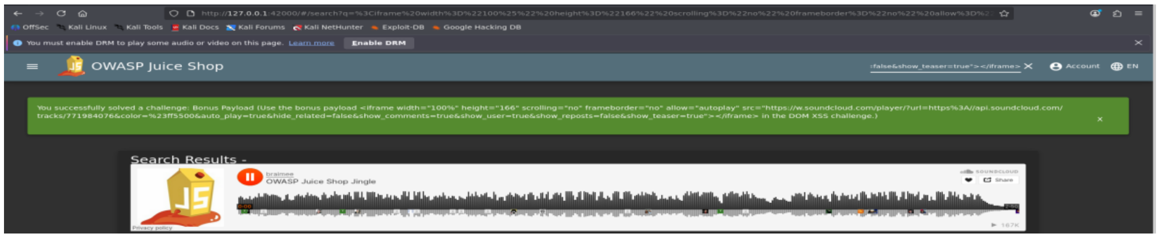
Mitigation

- Sanitize client-side input
- Avoid unsafe DOM functions (innerHTML, document.write)
- Implement Content Security Policy (CSP)

Bonus Payload Handling

Description

Certain pages accept unexpected inputs and reveal internal debug responses indicating insecure handling of user-controlled data.


http://127.0.0.1:42000/#/search?q=%3Ciframe%20width%3D%22100%25%22%20height%3D%22166%22%20scrolling%3D%22no%22%20frameborder%3D%22no%22%20allow%3D%2

You must enable DRM to play some audio or video on this page. Learn more Enable DRM

You successfully solved a challenge: Bonus Payload (Use the bonus payload <iframe width="100%" height="166" scrolling="no" frameborder="no" allow="autoplay" src="https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=true&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser=true"></iframe> in the DOM XSS challenge.)

Search Results -

Impact

- Internal logic exposure
- Increased attack surface

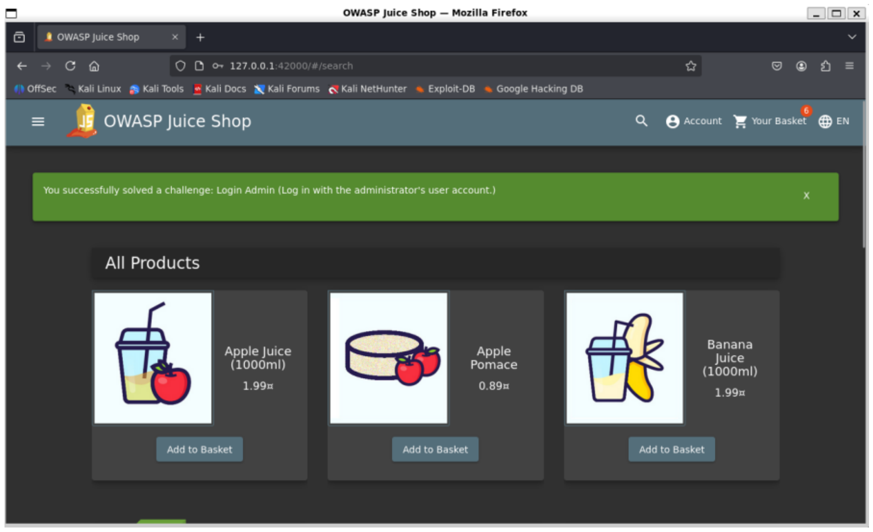
Mitigation

- Remove debug/test features
- Validate and sanitize all inputs
- Disable developer tools in production

SQL Injection – Admin Login Bypass

Description

Authentication bypass was achieved using SQL Injection.



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

Impact

- Attackers can exploit this to steal data
- Hijack sessions
- Gain admin access.

Mitigation

- Apply input validation
- Output encoding
- Prepared statements
- Security headers.

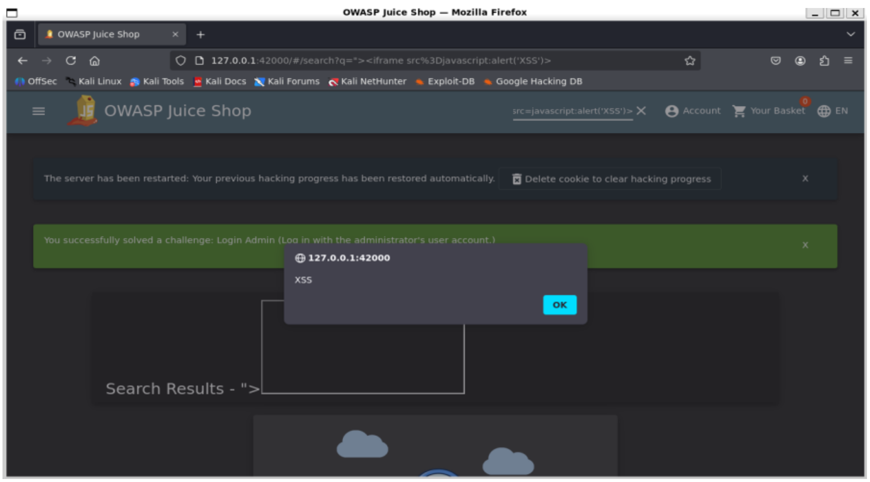
Reflected Cross-Site Scripting (XSS)

Description

Input reflected without sanitization allowing JavaScript execution.

Impact

- Unintended JavaScript execution
- Redirects and phishing attacks



```
127.0.0.1:42000/#/search?q="><iframe src%3Djavascript:alert('XSS')>
```

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.)

127.0.0.1:42000

XSS

Search Results - ">

Mitigation

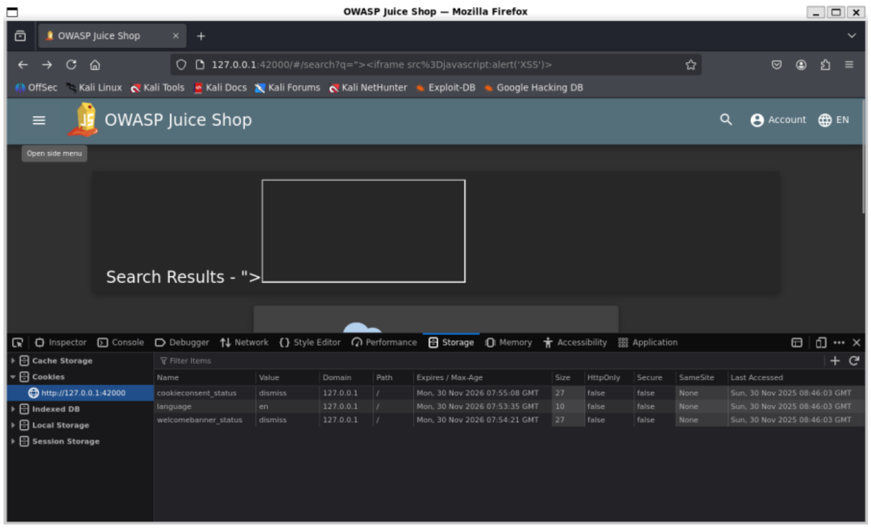
- Encode user input before rendering
- Implement server-side input validation
- Apply CSP headers

Stored Cross-Site Scripting (XSS)

Description: Malicious script stored in the database.

Impact: Attackers can exploit this to steal data, hijack sessions, or gain admin access.

Mitigation: Apply input validation, output encoding, prepared statements, and security headers.



```
127.0.0.1:42000/#/search?q="><iframe src%3Djavascript:alert('XSS')>
```

Search Results - ">

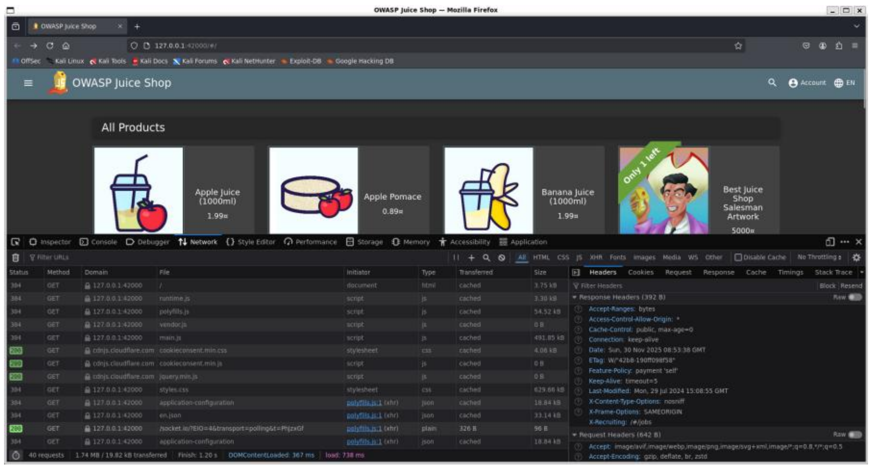
Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
cookieconsent_status	dismiss	127.0.0.1	/	Mon, 30 Nov 2026 07:55:08 GMT	27	false	false	None	Sun, 30 Nov 2025 08:46:03 GMT
language	en	127.0.0.1	/	Mon, 30 Nov 2026 07:53:35 GMT	10	false	false	None	Sun, 30 Nov 2025 08:46:03 GMT
welcomebanner_status	dismiss	127.0.0.1	/	Mon, 30 Nov 2026 07:54:21 GMT	27	false	false	None	Sun, 30 Nov 2025 08:46:03 GMT

Missing Security Headers

Description: CSP and X-Frame-Options headers were missing.

Impact: Attackers can exploit this to steal data, hijack sessions, or gain admin access.

Mitigation: Apply input validation, output encoding, prepared statements, and security headers.



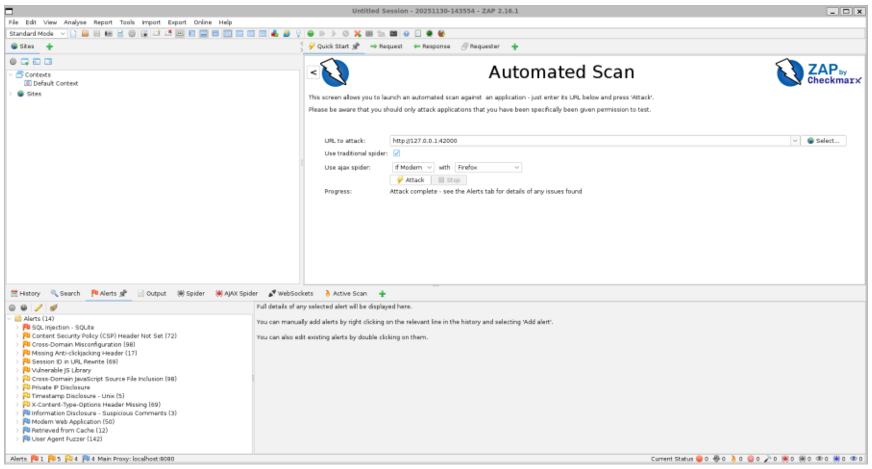
Status	Method	Domain	File	Initiator	Type	Transferred	Size
304	GET	127.0.0.1:42000	/	document	html	cached	3.75 kB
304	GET	127.0.0.1:42000	runtime.js	script	js	cached	3.30 kB
304	GET	127.0.0.1:42000	polyfills.js	script	js	cached	54.52 kB
304	GET	127.0.0.1:42000	vendor.js	script	js	cached	0 B
304	GET	127.0.0.1:42000	main.js	script	js	cached	491.85 kB
200	GET	cdnjs.cloudflare.com	cookieconsent.min.css	stylesheet	css	cached	4.06 kB
200	GET	cdnjs.cloudflare.com	cookieconsent.min.js	script	js	cached	0 B
200	GET	cdnjs.cloudflare.com	jquery.min.js	script	js	cached	0 B
304	GET	127.0.0.1:42000	styles.css	stylesheet	css	cached	629.66 kB
304	GET	127.0.0.1:42000	application-configuration	polyfills.js:1 (xhr)	json	cached	18.84 kB
304	GET	127.0.0.1:42000	en.json	polyfills.js:1 (xhr)	json	cached	33.14 kB
200	GET	127.0.0.1:42000	/socket.io/?EIO=4&transport=polling&t=PhjzxGf	polyfills.js:1 (xhr)	plain	326 B	96 B
304	GET	127.0.0.1:42000	application-configuration	polyfills.js:1 (xhr)	json	cached	18.84 kB

```
40 requests | 1.74 MB / 19.82 kB transferred | Finish: 1.20 s | DOMContentLoaded: 367 ms | load: 738 ms
```

```
Response Headers (392 B)
Accept-Ranges: bytes
Access-Control-Allow-Origin: *
Cache-Control: public, max-age=0
Connection: keep-alive
Date: Sun, 30 Nov 2025 08:53:38 GMT
ETag: W/"42b8-190ff098f58"
Feature-Policy: payment 'self'
Keep-Alive: timeout=5
Last-Modified: Mon, 29 Jul 2024 15:08:55 GMT
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Recruiting: /#/jobs
Request Headers (642 B)
Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
```

5. OWASP ZAP Automated Scan Results

An automated vulnerability scan was conducted using OWASP ZAP. The scan detected SQL Injection, security misconfigurations, missing headers, and information disclosure issues.



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://127.0.0.1:42000

Use traditional spider: ✓

Use ajax spider: If Modern with Firefox

Progress: Attack complete - see the Alerts tab for details of any issues found

```
Alerts (14)
  SQL Injection - SQLite
  Content Security Policy (CSP) Header Not Set (72)
  Cross-Domain Misconfiguration (98)
  Missing Anti-clickjacking Header (17)
  Session ID in URL Rewrite (69)
  Vulnerable JS Library
  Cross-Domain JavaScript Source File Inclusion (98)
  Private IP Disclosure
  Timestamp Disclosure - Unix (5)
  X-Content-Type-Options Header Missing (69)
  Information Disclosure - Suspicious Comments (3)
  Modern Web Application (50)
  Retrieved from Cache (12)
  User Agent Fuzzer (142)
```

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

Alerts 1 5 4 4 Main Proxy: localhost:8080

6. Conclusion

The OWASP Juice Shop application contains multiple critical vulnerabilities that could be exploited in real-world scenarios. Secure coding practices, regular testing, and proper configuration are essential to prevent attacks.

7. Tools Used

- OWASP Juice Shop
- OWASP ZAP Proxy
- Mozilla Firefox
- Kali Linux
- Browser Developer Tools