



WEB APPLICATION SECURITY TESTING

NAME: SMRIDHI GERA

TASK 1: WEB APPLICATION SECURITY TESTING

**PROGRAM: FUTURE INTERNS- CYBERSECURITY
INTERNSHIP**

DATE: OCTOBER 2025

TARGET:APPLICATION :OWASP JUICE SHOP

Task Summary

This assessment focused on identifying various loopholes and vulnerabilities in the **OWASP Juice Shop** application through **Burp Suite** and manual evaluation techniques. Several vulnerabilities were successfully detected, including **SQL Injection** and **User Information Leakage**, which could lead to potential attacks and account compromises. The findings were mapped to the **OWASP Top 10 (2021)** categories based on associated risk levels.

Tools Used

- Docker
- OWASP Juice Shop
- Burp Suite

Steps & Procedure

1. The **OWASP Juice Shop** application was deployed locally using **Docker**, ensuring a lightweight and isolated testing environment with all dependencies pre-configured.
2. **Burp Suite** was utilized for intercepting, scanning, and analyzing web requests and responses to identify vulnerabilities within the application.
3. All discovered issues were manually validated and mapped to the **OWASP Top 10 (2021)** categories based on their severity and potential impact.
4. Corresponding **mitigation strategies** were proposed for each identified vulnerability to enhance overall application security.

Identified Vulnerabilities

1. SQL Injection

Description:

The login form was vulnerable to SQL Injection, as it accepted unsanitized input that was directly concatenated into SQL queries. Attackers could inject payloads like '`' OR '1'='1`' to bypass authentication and access restricted areas.

Security Impact:

- Complete bypass of authentication.
- Unauthorized access to database records.
- Potential full compromise of database contents.

Risk Level: High

Evidence:

The screenshot shows the OWASP Juice Shop login interface. At the top, there is a message: "The server has been restarted: Your previous hacking progress has been restored automatically." Below this is a checkbox labeled "Delete cookie to clear hacking progress". The main area is a "Login" form. In the "Email*" field, the value "' OR 1=1--" is entered. In the "Password*" field, the value "test" is entered. Below the form, there is a link "Forgot your password?". At the bottom of the form, there is a "Log in" button with a key icon, a "Remember me" checkbox, and a "Log in with Google" button. A horizontal line with the word "or" is between the "Log in" button and the Google button. At the very bottom of the page, there is a link "Not yet a customer?".

Figure 1: SQLi Payload in Login Form



89	http://localhost:3000	POST	/rest/user/login	✓	127.0.0.1	
21	http://localhost:3000	PUT	/rest/continue-code/apply/kvQWD...	434 200	JSON	127.0.0.1

Request

Pretty Raw Hex

```
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 40
4 sec-ch-ua-platform: "Windows"
5 Accept-Language: en-US,en;q=0.9
6 Accept: application/json, text/plain, */*
7 sec-ch-ua: "Chromium";v="141", "Not?A_Brand";v="8"
8 Content-Type: application/json
9 sec-ch-ua-mobile: ?0
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=kvvQWDpj6xnrNSVlbRgCeyZL0eetKfg6uJX060EkzMPYs8amJXBKv4173qZ8
18 Connection: keep-alive
19
20 {
    "email": "" OR l=1--",
    "password": "test"
}
```

?

⟳

⟳

Search

0 highlights

Figure 2: burp suite intercepted request containing malicious payload

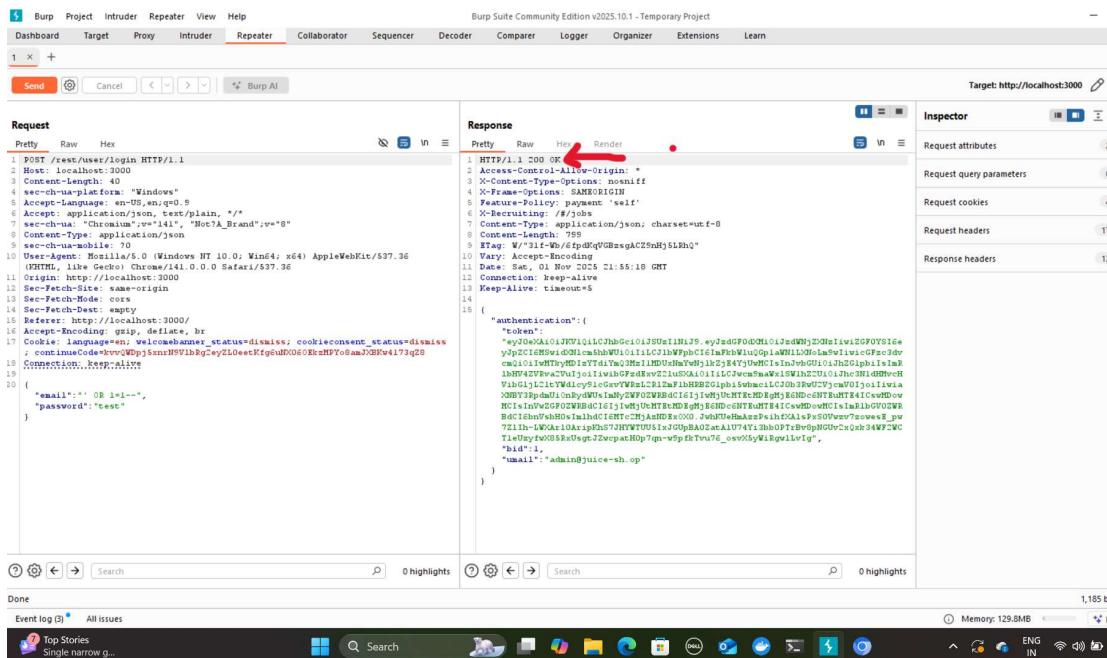


Figure 3: burp suite response showing http status 200 ok ,comforming successful login.

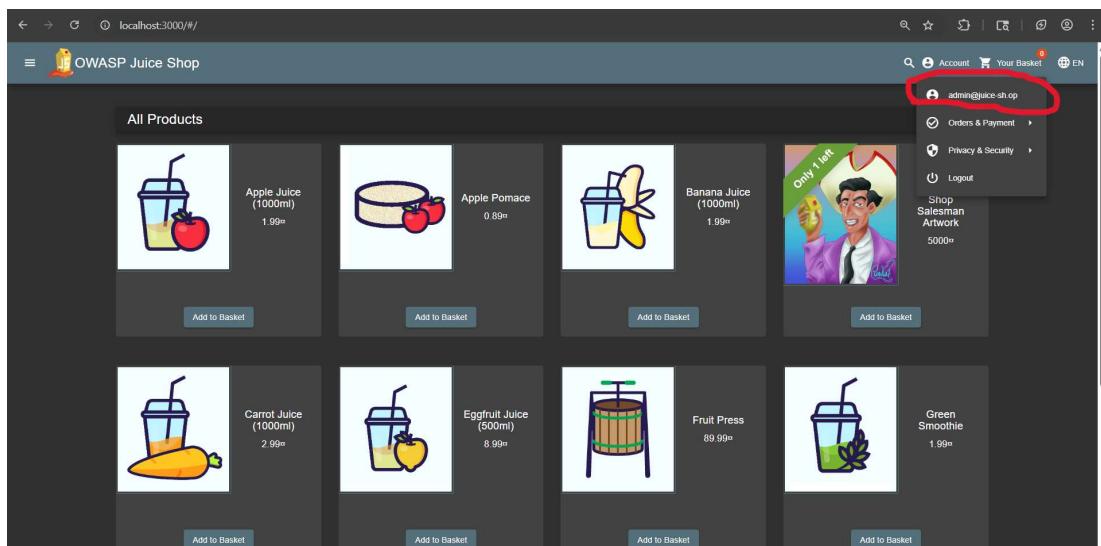


Figure 4: post login view showing access to admin account.

Mitigation Strategies:

- Use parameterized queries or prepared statements.
- Sanitize user inputs with strict whitelisting.
- Disable detailed SQL error messages in production.
- Implement least privilege access for the database user.

2. Cross-Site Scripting (XSS)

Description:

Certain input fields (such as feedback or product search) failed to properly sanitize user input, allowing JavaScript injection that executes in the victim's browser.

Security Impact:

- Theft of session cookies or tokens.
- Execution of unauthorized actions in user accounts.
- Phishing or defacement attacks on legitimate users.

Risk Level: High

EVIDENCE:

The screenshot shows a web application interface for "Customer Feedback". At the top, there is a navigation bar with the OWASP Juice Shop logo, a search icon, a shopping cart icon with '0' items, and a globe icon. The main form is titled "Customer Feedback". It contains the following fields:

- Author:** A text input field containing the value "****you+1@gmail.com".
- Comment*:** A text area input field containing the value "<script>alert('XSS')</script>".
- Rating:** A slider input field set to a value of 5.
- CAPTCHA:** A question "What is 1-3+1 ?" followed by a text input field containing the value "-3".
- Submit:** A blue button with a right-pointing arrow icon labeled "Submit".

The form has validation messages: "Max. 160 characters" above the comment area and "29/160" indicating the current character count. There is also a small note "Result*" next to the CAPTCHA input field.

Figure1: feedback form with payload

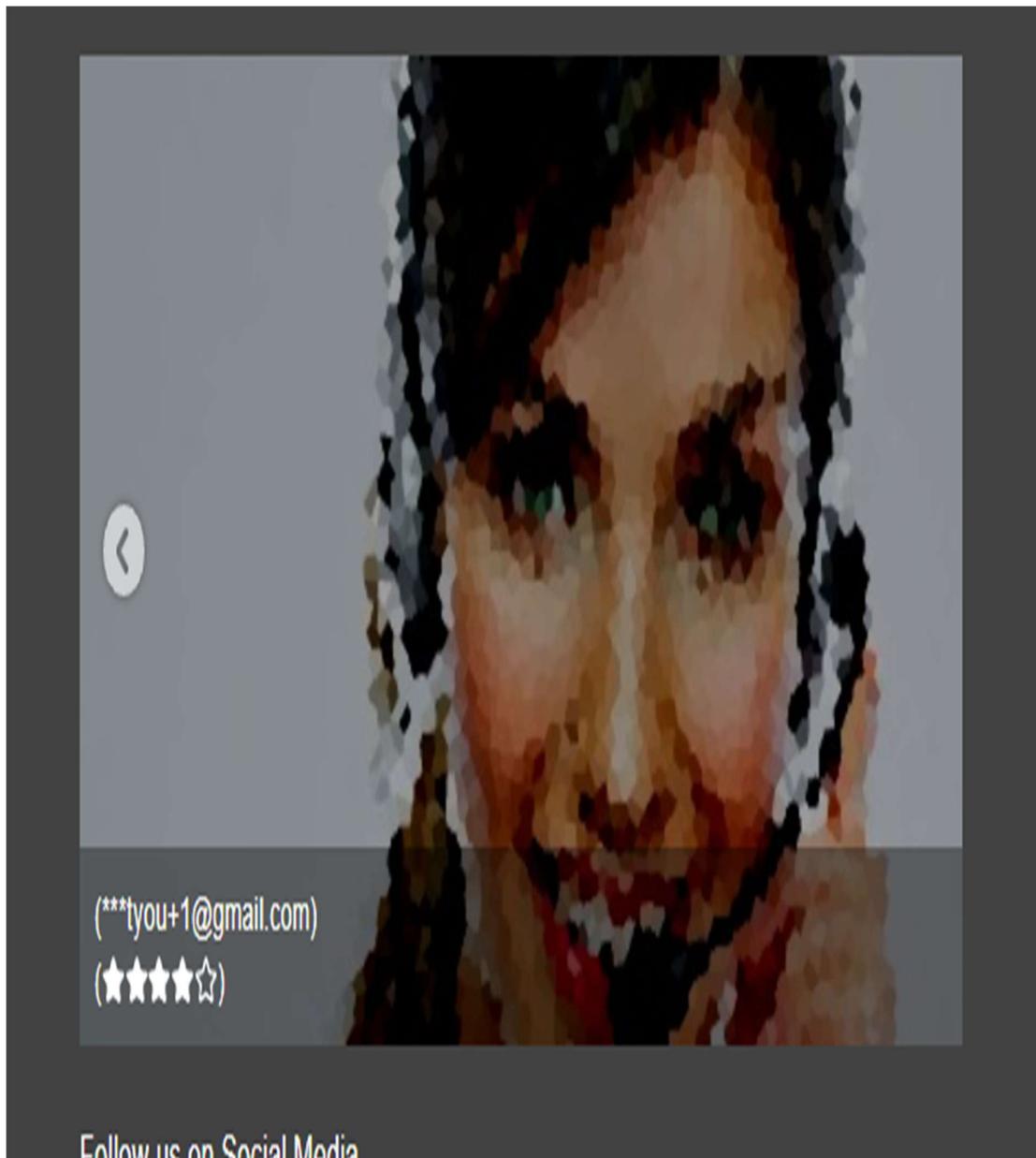


Figure2: alert box triggered after visiting abou

Mitigation Strategies:

- Implement input validation and output encoding.
- Use frameworks or libraries with built-in XSS protection.
- Apply a strong Content Security Policy (CSP).

3. Replay and Impersonation

Description:

Session tokens and authenticated requests could be replayed without verification, meaning previously captured requests (like feedback submission) could be resent to perform repeated actions.

Security Impact:

- Impersonation of legitimate users.
- Replay of actions such as duplicate submissions or transactions.
- Potential exploitation of long-lived tokens.

Risk Level:

Medium

The screenshot shows a "Customer Feedback" form on a dark-themed website. The form fields are as follows:

- Author:** ***tyou+1@gmail.com
- Comment*:** testing you
- Rating:** A slider bar with a green circle at the midpoint, indicating a rating of approximately 5 out of 10.
- CAPTCHA:** What is 10 - 6 + 8 ?
Result*: 12

A large blue "Submit" button is located at the bottom of the form.

Figure 1: original feedback request submitted through the customer feedback form.

The screenshot shows the Burp Suite Repeater interface. On the left, the 'Request' pane displays a replayed HTTP request with line numbers and color-coded status codes. The 'Pretty' tab is selected. The request includes various headers (User-Agent, Accept, Content-Type, Origin, Sec-Fetch-Site, Sec-Fetch-Mode, Sec-Fetch-Dest, Referer, Accept-Encoding, Cookie) and a JSON payload. The payload contains a 'message' field with the value 'testing you' and an 'author' field with the value 'testyout1@gmail.com'. On the right, the 'Inspector' pane is visible, showing sections for Request attributes, Request cookie, and Request header.

```
1 Chrome/141.0.0.0 Safari/537.36
2 Accept: application/json, text/plain, */*
3 Content-Type: application/json
4 Origin: http://localhost:3000
5 Sec-Fetch-Site: same-origin
6 Sec-Fetch-Mode: cors
7 Sec-Fetch-Dest: empty
8 Referer: http://localhost:3000/
9 Accept-Encoding: gzip, deflate, br
10 Cookie: language=en; welcomebanner_status=dismiss;
11 cookieconsent_status=dismiss; continueCode=
12 W2PaW3mD5oBa7Mp6PLlyrQKw2zd5btQfJKG0Rx1Nreb49VvJZq8gjn
13 EXYZr3; token=
14 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzd
15 WNjZXNzIiwicGFOYSI6eyJpZCI6MjMsInVzZXJuYWl1IjoiiIwiZWl
16 haWwiOiJOZXNOeW91KzFAZ2lhaWwuY29tIiwicGFzc3dvcmQiOiIxM
17 mJjZTM3NGU3YmUxNTE0MmU4MTcyZjY20GRhMDBkOCIsInJvbGUIoIJ
18 jDXNOb2llciIsImRlbHV4ZVRva2VuIjoiiIwiibGFzdExvZ2luSXAiO
19 iIwljAuMC4wiwicHJvZmlsZUltyWd1Ijoil2Fzc2V0cy9wdWJsaWM
20 vaWhZ2VzL3VwbG9hZHMvZGVmYXVsdC5zdmciLCJ0b3RwU2VjcmVOI
21 joiIiwiAxNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjUtMTE
22 tMDEgMTk6NDk6NDcu0DkzICswMDowMCIsInVzGF0ZWRBdCI6IjIwM
23 jUtMTEtMDEgMTk6NDk6NDcu0DkzICswMDowMCIsImRlbGV0ZWRBdCI
24 6bnVsbHOsImlhhdCI6MTc2MjAyNjU5OH0.scsJvhDoJWPyn_Wbxscot
25 lNaJYHKUjBkWCzACaYwArt2flzNhh20uXSoIyjvvQtgmSIZq2pU0g3
26 Js5j_oeQQfpDgajmLpznJXR1Koc5wUnBpMPqzke4eZfw1yy_Cp3oQ1
27 J8oZSMKt7NpMhX7M6SLjFtMsLmIVzTnJeLAnYixPs
28
29 Connection: keep-alive
30
31 {
32     "message": "testing you",
33     "author": "testyout1@gmail.com"
34 }
```

Figure 2: replayed request sent via burp suite repeater and accepted multiple times.

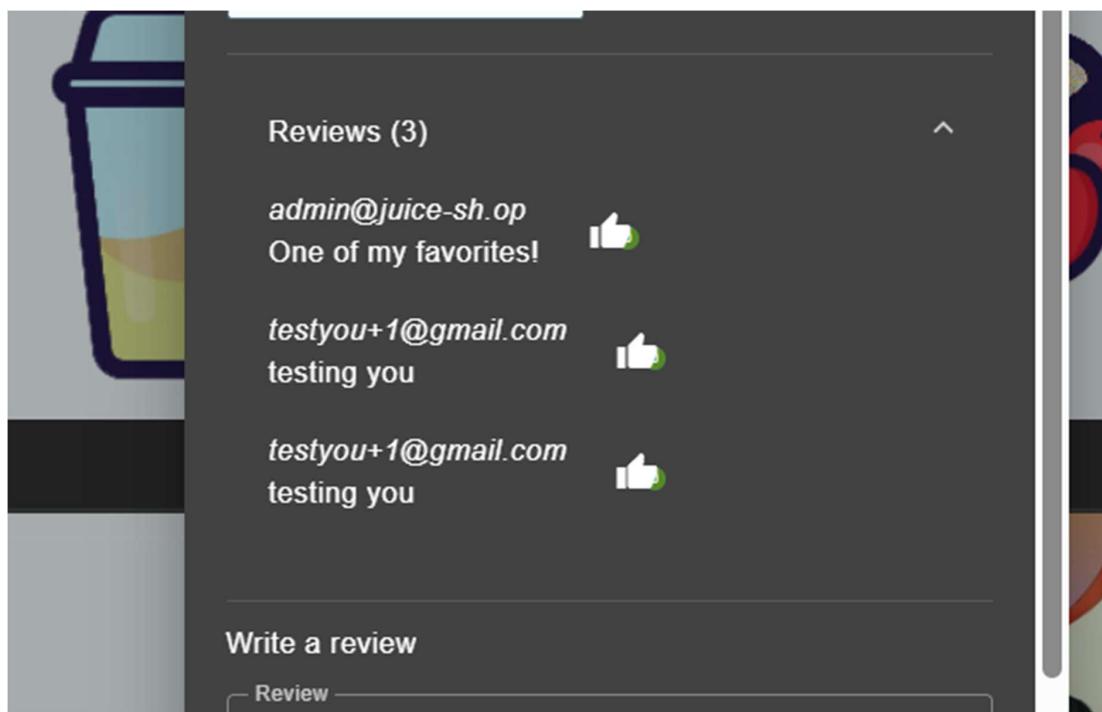


Figure 3: execution of replayed request multiple times

Mitigation Strategies:

- Implement nonce or timestamp validation to prevent replay.
- Use short-lived session tokens and refresh them periodically.
- Invalidate tokens immediately after logout or session timeout.

4. Insecure Token Design

Description:

The application's JSON Web Tokens (JWTs) used weak algorithms such as MD5 for signing, which are susceptible to collision attacks. Tokens also contained hashed user credentials that can be cracked.

Security Impact:

- Forged or modified tokens may be accepted by the server.
- Attackers can gain unauthorized access or persist sessions indefinitely.
- High risk of account takeover and privilege escalation.

Risk Level: High

Evidence:

Request	Response
<pre>✓ curl -X POST http://localhost:3000/test/user/login 415 {"error": "Unsupported Media Type", "message": "Content-Type must be application/json", "status": 415}</pre>	<pre>✓ curl -X POST http://localhost:3000/test/user/login 1190 {"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJsb2dpbiIsInR5cGUiOiJsb2dpbiJ9", "user": {"id": 1, "name": "John Doe", "email": "john.doe@example.com", "password": "\$2b\$10\$uXfZuLkQHqBvVgCQDwvPQ", "role": "user"}, "exp": 1628531200}</pre>
<pre>✓ curl -X POST http://localhost:3000/test/login 428 {"error": "Bad Request", "message": "Missing required parameter: email or password", "status": 428}</pre>	<pre>✓ curl -X POST http://localhost:3000/test/login 215 {"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJsb2dpbiIsInR5cGUiOiJsb2dpbiJ9", "user": {"id": 1, "name": "John Doe", "email": "john.doe@example.com", "password": "\$2b\$10\$uXfZuLkQHqBvVgCQDwvPQ", "role": "user"}, "exp": 1628531200}</pre>
<pre>✓ curl -X PUT http://localhost:3000/test/products/1/reviews 107 {"error": "Method Not Allowed", "message": "PUT method not allowed for this resource", "status": 405}</pre>	<pre>✓ curl -X PUT http://localhost:3000/test/products/1/reviews 409 {"error": "Conflict", "message": "The product already has a review from the user", "status": 409}</pre>

Mitigation Strategies:

- Use strong signing algorithms (e.g., HS256 or RS256).
- Include token expiration and implement key rotation.
- Avoid storing sensitive information inside JWTs.

5. Insecure Direct Object Reference (IDOR)

Description:

Internal references like user IDs were exposed in URLs and API endpoints without verifying ownership. Modifying these IDs (e.g., userId=101 → userId=102) allowed access to other users' data.

Security Impact:

- Unauthorized access to other users' personal data.
- Data manipulation and privacy violations.
- Breach of confidentiality and data integrity.

Risk Level: High

Request	Response
<pre>Pretty Raw Hex 1 GET /rest/products/2/reviews HTTP/1.1 2 Host: localhost:3000 3 sec-ch-ua-platform: "Windows" 4 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWJ2QHsiIiwicGF0YSI6eyJpZC i6MjMsInVzZXJuW11joiIiwiZW1haWviOj0ZDQ0eW51KzFAZ2lhaWwUY19tIiwicGFzc3dvcmQj0 iXhMjZTMHNUGUYm0xNT0Rhm4NtCv2jyZOGRAHMUBR0CIsInJvbGU10i3jdQ0ObCl1c1sImRlbH4/ ZVRva2VuIjoiIiwibGFzdEx2lNSXAx0iIw1jAuMC4vIiwiHjZmIsZ2lItY2d1IjoiLzfscZV0cy9 wdWjsaMWhraIbhZZVsL3VubGShEHNvZGVnYXVadCsmdc1LCJ0J3BwUWjcmU0joiIiwiw0QHRY3Pdm U1OnRydWUsIamlyZWF0ZWRBdc16i1jMjUcMTkMd5ghThENDcu0dkcICsvMDowHc1sInVz2GFOZ WBdAC16i1jMjUcMTkMd5ghThENDcu0dkcICsvMDowHc1sImRlbGV0ZWBdc16bnVsbsH0s1mlh dC16MTC2MjAyNyU50SHO.scsJvhDoJWPm_Whsxcf1NaYHRUjBwVCmAcyAArtCfisNh0uXs0Iyj vvQtgmS1ZqCpu0Ug3s5j_oeQQtpDgajmlpnzJKRIko5wUnBpMpqke4eZf2Iiy_Cp3oQ18oZ5MHC7 NpMrhV7HESjPch5mlVsnTnjeAxtYixF 5 Accept-Language: en-US,en;q=0.9 6 Accept: application/json, text/plain, /* 7 sec-ch-ua: "Chromium";v="141", "Not?A_Brand";v="8" 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36 9 sec-ch-ua-mobile: 20 10 Sec-Fetch-Site: same-origin 11 Sec-Fetch-Mode: cors 12 Sec-Fetch-Dest: empty 13 Referer: http://localhost:3000/ 14 Accept-Encoding: gzip, deflate, br 15 Cookie: language=en, welcomebanner_status=dismiss, cookieconsent_status=dismiss ; continueCode=WPAw3Ab5oBa7Mp6PLlyrQWk2dshbQfJNKGOxlnRheb49VvJZq8qjnEXYzr3; token= eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWJ2QHsiIiwicGF0YSI6eyJpZC i6MjMsInVzZXJuW11joiIiwiZW1haWviOj0ZDQ0eW51KzFAZ2lhaWwUY19tIiwicGFzc3dvcmQj0 iXhMjZTMHNUGUYm0xNT0Rhm4NtCv2jyZOGRAHMUBR0CIsInJvbGU10i3jdQ0ObCl1c1sImRlbH4/ ZVRva2VuIjoiIiwibGFzdEx2lNSXAx0iIw1jAuMC4vIiwiHjZmIsZ2lItY2d1IjoiLzfscZV0cy9 wdWjsaMWhraIbhZZVsL3VubGShEHNvZGVnYXVadCsmdc1LCJ0J3BwUWjcmU0joiIiwiw0QHRY3Pdm U1OnRydWUsIamlyZWF0ZWRBdc16i1jMjUcMTkMd5ghThENDcu0dkcICsvMDowHc1sInVz2GFOZ WBdAC16i1jMjUcMTkMd5ghThENDcu0dkcICsvMDowHc1sImRlbGV0ZWBdc16bnVsbsH0s1mlh dC16MTC2MjAyNyU50SHO.scsJvhDoJWPm_Whsxcf1NaYHRUjBwVCmAcyAArtCfisNh0uXs0Iyj vvQtgmS1ZqCpu0Ug3s5j_oeQQtpDgajmlpnzJKRIko5wUnBpMpqke4eZf2Iiy_Cp3oQ18oZ5MHC7</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 Access-Control-Allow-Origin: * 3 X-Content-Type-Options: nosniff 4 X-Frame-Options: SAMEORIGIN 5 Feature-Policy: payment 'self' 6 X-Reputing: /#jobs 7 Content-Type: application/json; charset=utf-8 8 Content-Length: 171 9 Etag: W/"ab-CHqkZEqcfcQHdZn4Uv7apCXqHOM" 10 Vary: Accept-Encoding 11 Date: Sat, 01 Nov 2025 20:34:13 GMT 12 Connection: keep-alive 13 Keep-Alive: timeout=5 14 15 {"status": "success", "data": [{"message": "Your firewall needs m0r3 muscl3", "author": "uvgoin@juice-sh.op", "product": "2", "likesCount": 0, "likedBy": [], "_id": "AFgxSWAcPf5cHF"}, {"status": "success", "data": [{"message": "Your firewall needs m0r3 muscl3", "author": "uvgoin@juice-sh.op", "product": "2", "likesCount": 0, "likedBy": [], "_id": "AFgxSWAcPf5cHF"}]}</pre>

Mitigation Strategies:

- Validate resource ownership server-side before processing requests.
- Apply “deny by default” access policies.
- Enforce authorization checks on every endpoint.

OWASP Top 10 (2021) Mapping

Detected Vulnerability	OWASP Category	Risk Rating
SQL Injection	A03 - Injection	High
Cross-Site Scripting (XSS)	A03 - Injection	High
Replay & Impersonation	A07 - Identification & Authentication Failures	Medium
Insecure Token Design	A02 - Cryptographic Failures	High
Insecure Direct Object Reference (IDOR)	A01 - Broken Access Control	High

Conclusion

The vulnerability assessment of OWASP Juice Shop using Docker and Burp Suite revealed multiple security weaknesses such as **SQL Injection, Cross-Site Scripting, Insecure Token Design, Replay & Impersonation, and Broken Access Control (IDOR)**. These vulnerabilities can lead to **data breaches, account hijacking, and unauthorized access** if left unresolved.