OWASP JUICE SHOP

VAPT REPORT

WHAT YOU NEED TO KNOW

This report presents the results of a security test conducted on the OWASP Juice Shop, a vulnerable web application used for learning purposes. The test revealed several critical and high-severity issues, including SQL injection, token leakage, weak password hashing, and DOM-based XSS.

The assessment simulated real-world attacks and demonstrated a full compromise of the application through chained vulnerabilities. These findings highlight the importance of secure coding practices, proper session handling, and input validation.

Tools Used

Tool	Purpose
Burp Suite	HTTP request interception & tampering
JWT.io	JSON Web Token decoding
Hash-Identifier	Hash algorithm detection
Online MD5 Cracker	Password recovery from hash
Developer Tools	Inspect browser storage

Exploitation Summary

Vulnerabilities Already Covered:

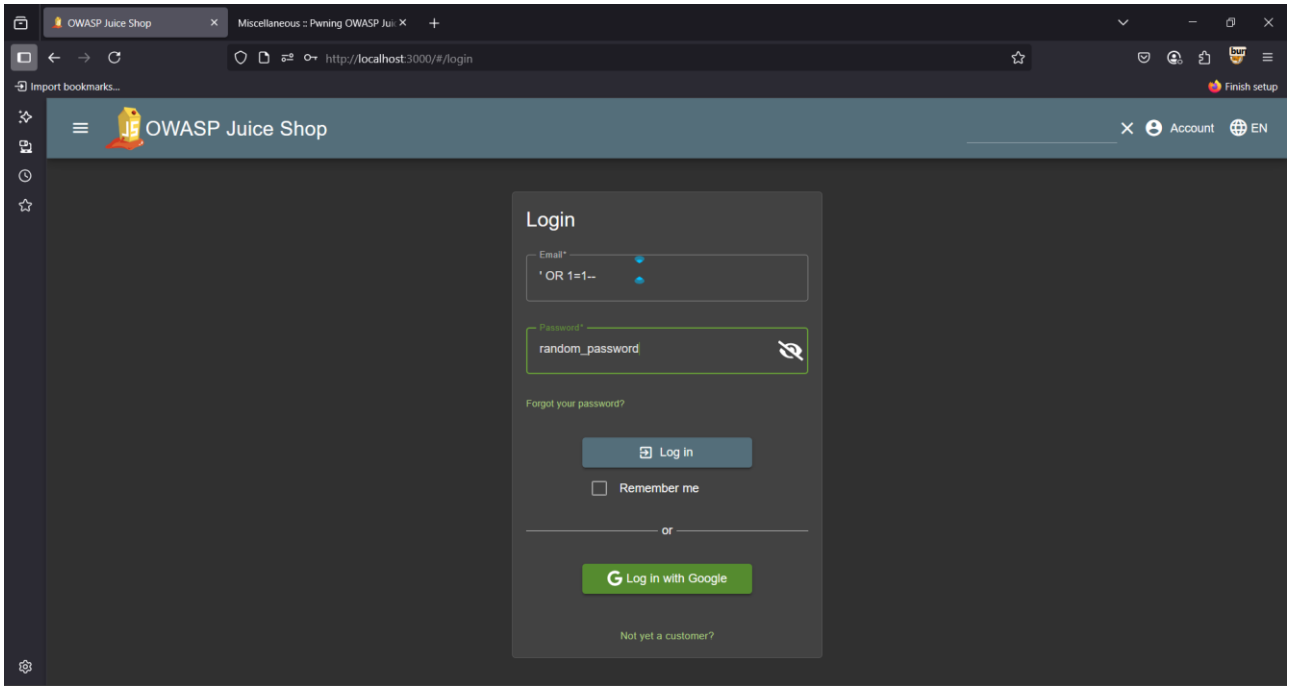
1. SQL Injection (SQLi) – Login Bypass → Critical
2. JWT Token Disclosure via DevTools – Sensitive token in localStorage
3. Weak Hashing (MD5) – Password can be cracked easily
4. DOM-based XSS – Executed via injected iframe in search

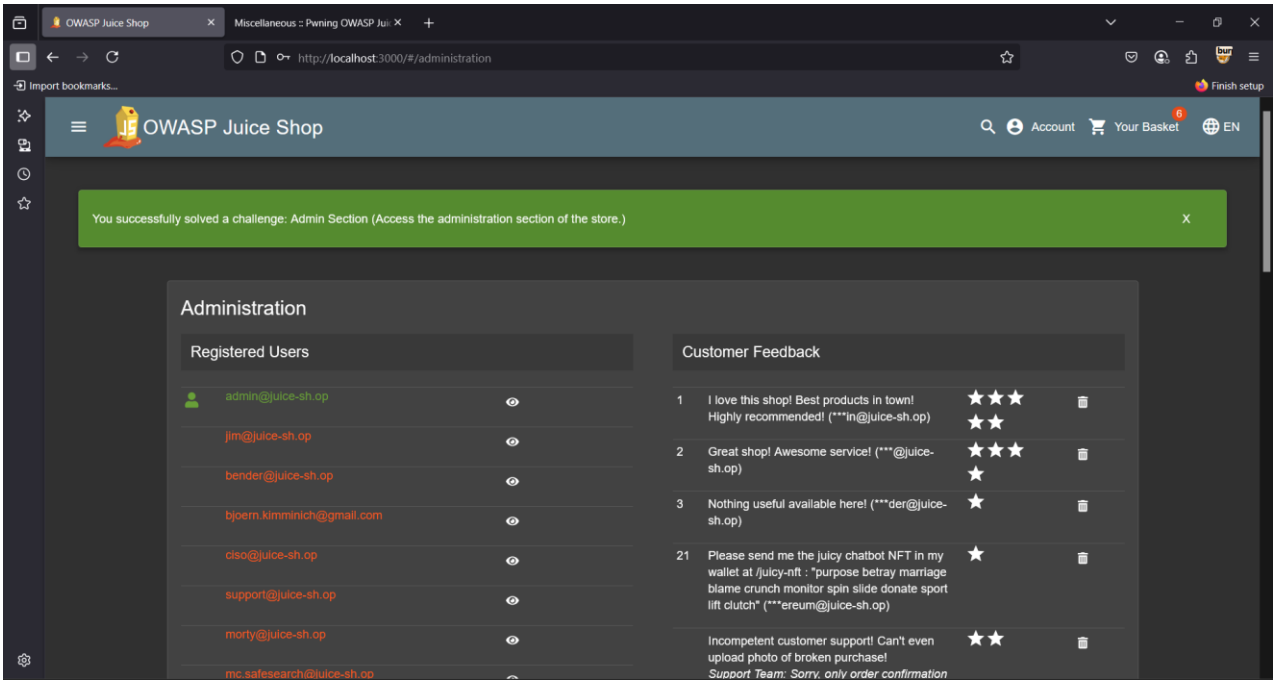
A. SQLi to Admin Access

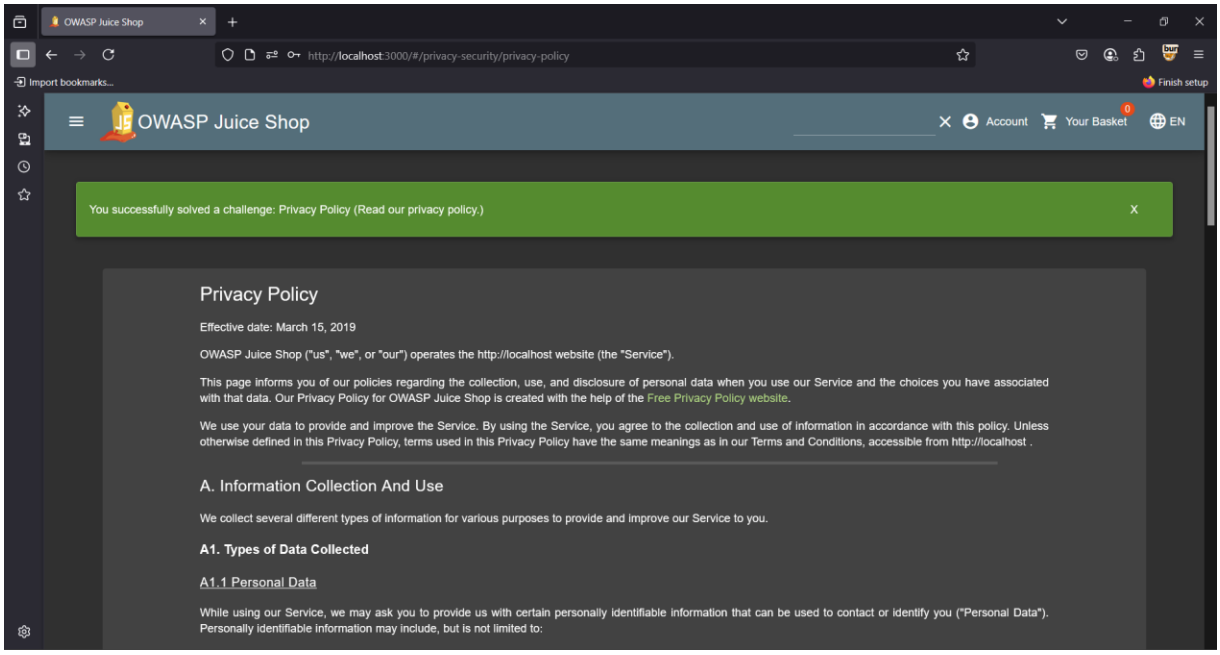
Target Website: http://localhost:3000 (OWASP Juice Shop)

1. Login Bypass using SQL Injection

Description: Used ' OR 1=1 -- in the login field to bypass authentication.
Impact: Unauthorized access to the application as a normal user.


OWASP Juice Shop
Miscellaneous :: Pwning OWASP Juice
http://localhost:3000/#/login
Import bookmarks...
Finish setup
OWASP Juice Shop
Account
EN
Login
Email*
' OR 1=1--
Password*
random_password
Forgot your password?
Log in
Remember me
or
Log in with Google
Not yet a customer?

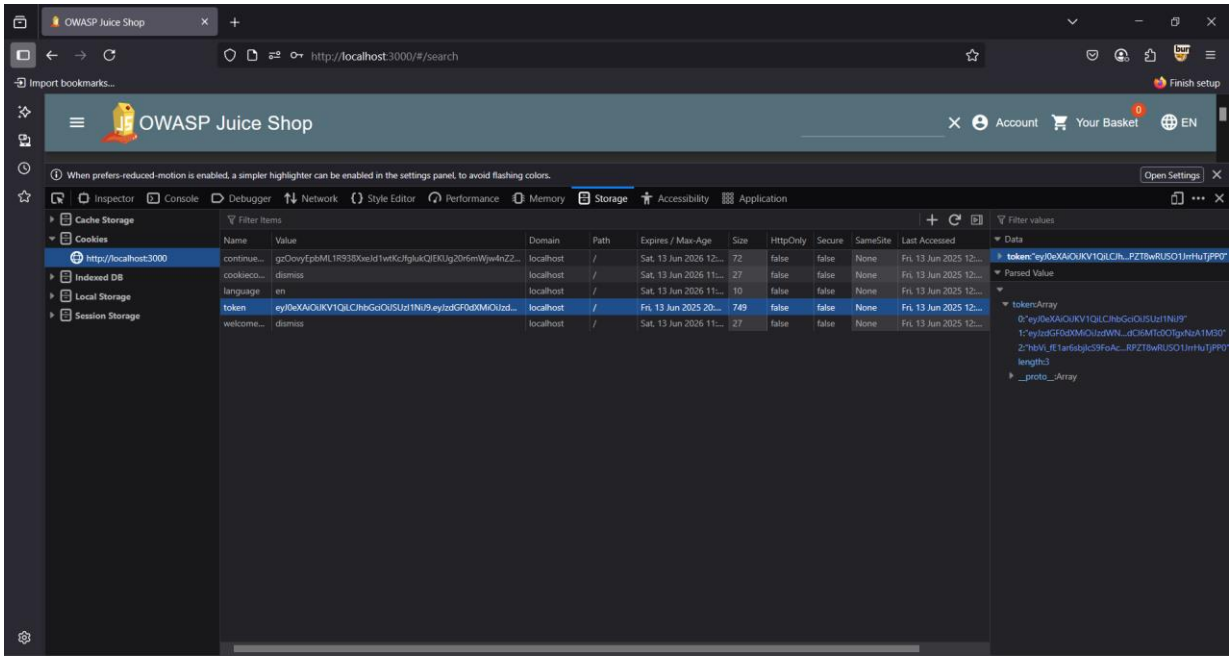

OWASP Juice Shop
Miscellaneous :: Pwning OWASP Juice
http://localhost:3000/#/administration
Import bookmarks...
Finish setup
OWASP Juice Shop
Account
Your Basket
6
EN
You successfully solved a challenge: Admin Section (Access the administration section of the store.)
Administration
Registered Users
admin@juice-sh.op
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
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



2. JWT Token Discovery via DevTools

Description: Inspected browser → Developer Tools → Application → Local Storage to retrieve the JWT token.

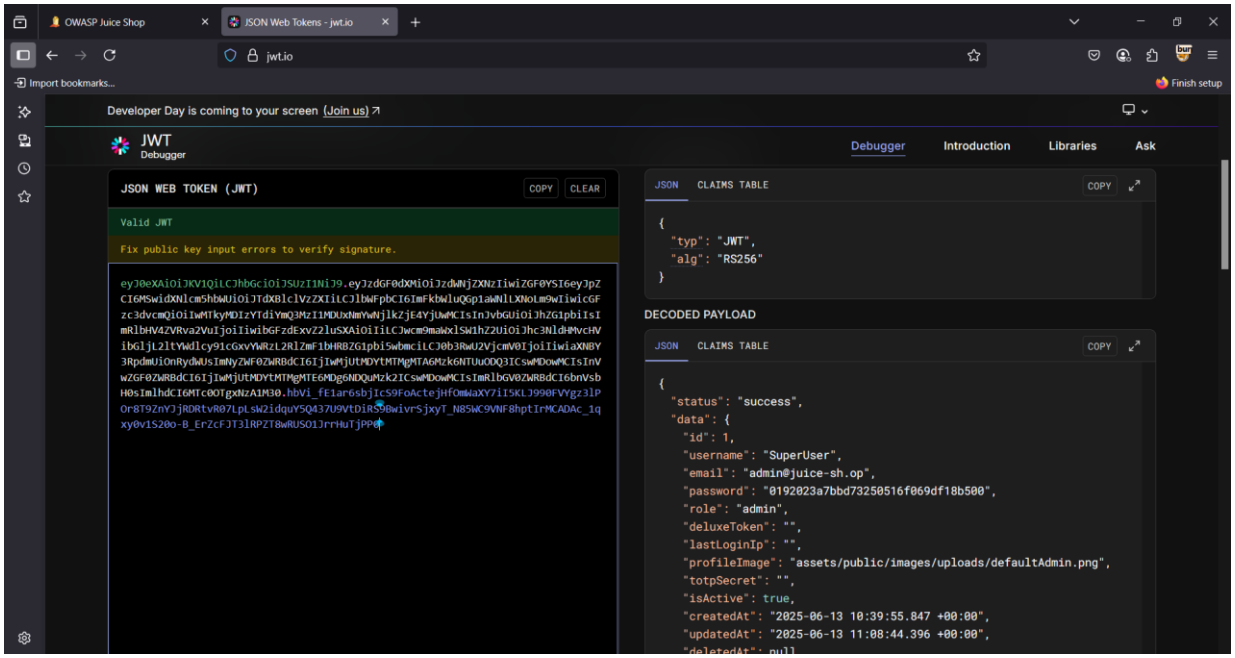
Impact: Exposed token could allow user impersonation or token analysis.



3. Token Decoding

Tool Used: JWT.io (or similar online tool)

Result: Decoded the token to reveal user data including a hashed password.



4. Hash Identification using Hash-Identifier

Tool Used: hash-identifier (Linux tool)

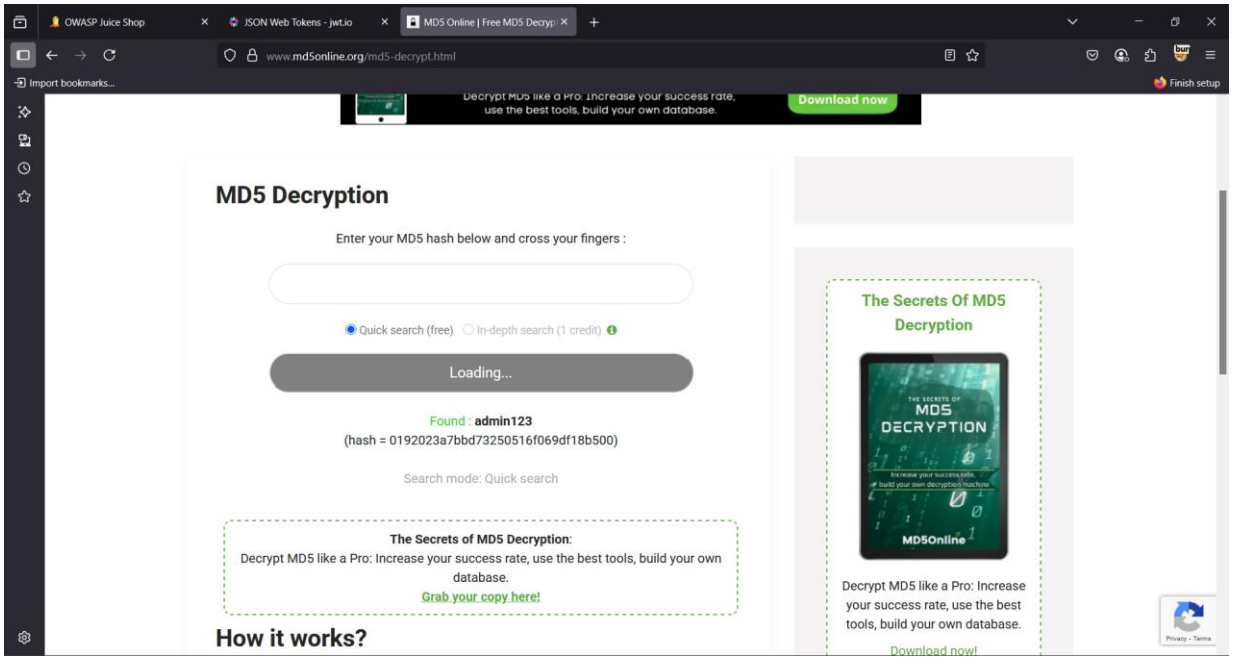
Hash Analyzed: 0192023a7bbd73250516f069df18b500

Result: Identified as MD5 hash.

5. MD5 Hash Cracking

Tool Used: Online MD5 hash cracker.

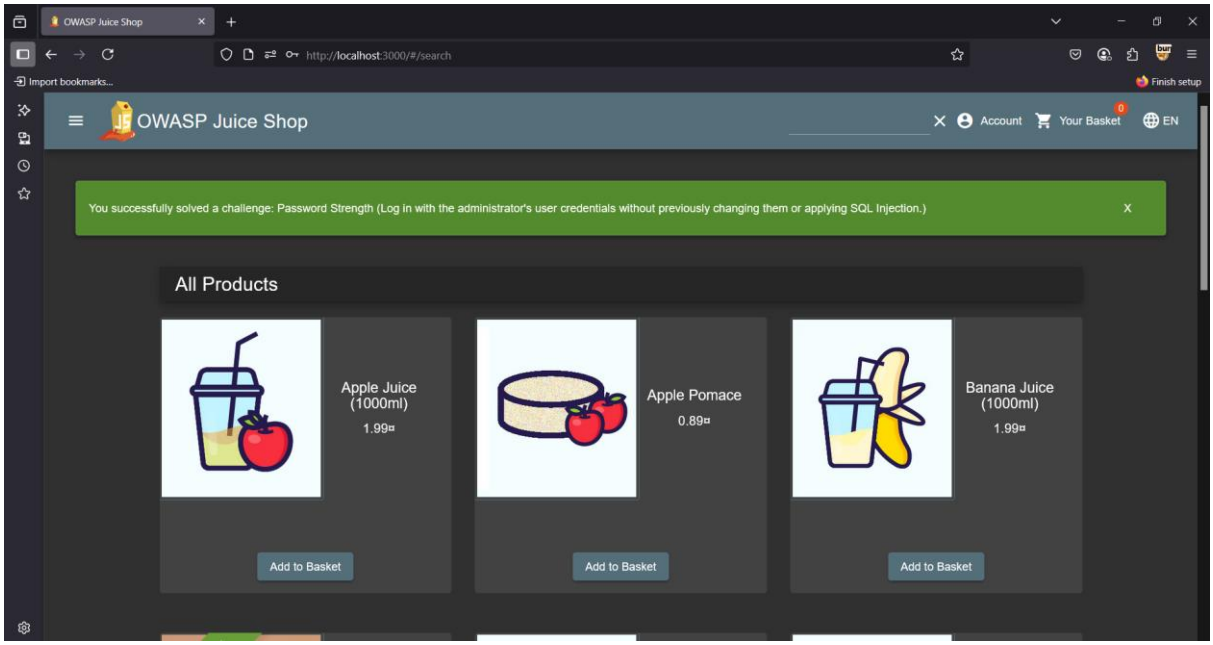
Result: MD5 hash matched to password “admin123”

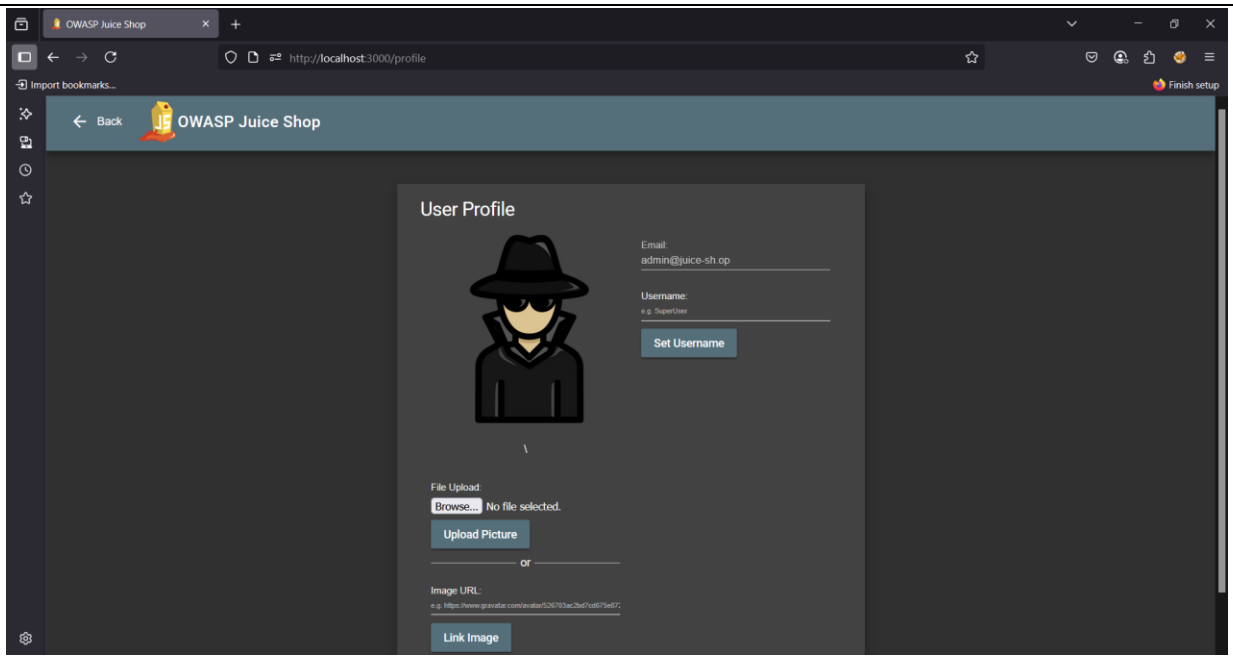


6. Admin Account Compromise

Action: Used recovered credentials to log in as admin.

Impact: Full application access with administrative privileges.





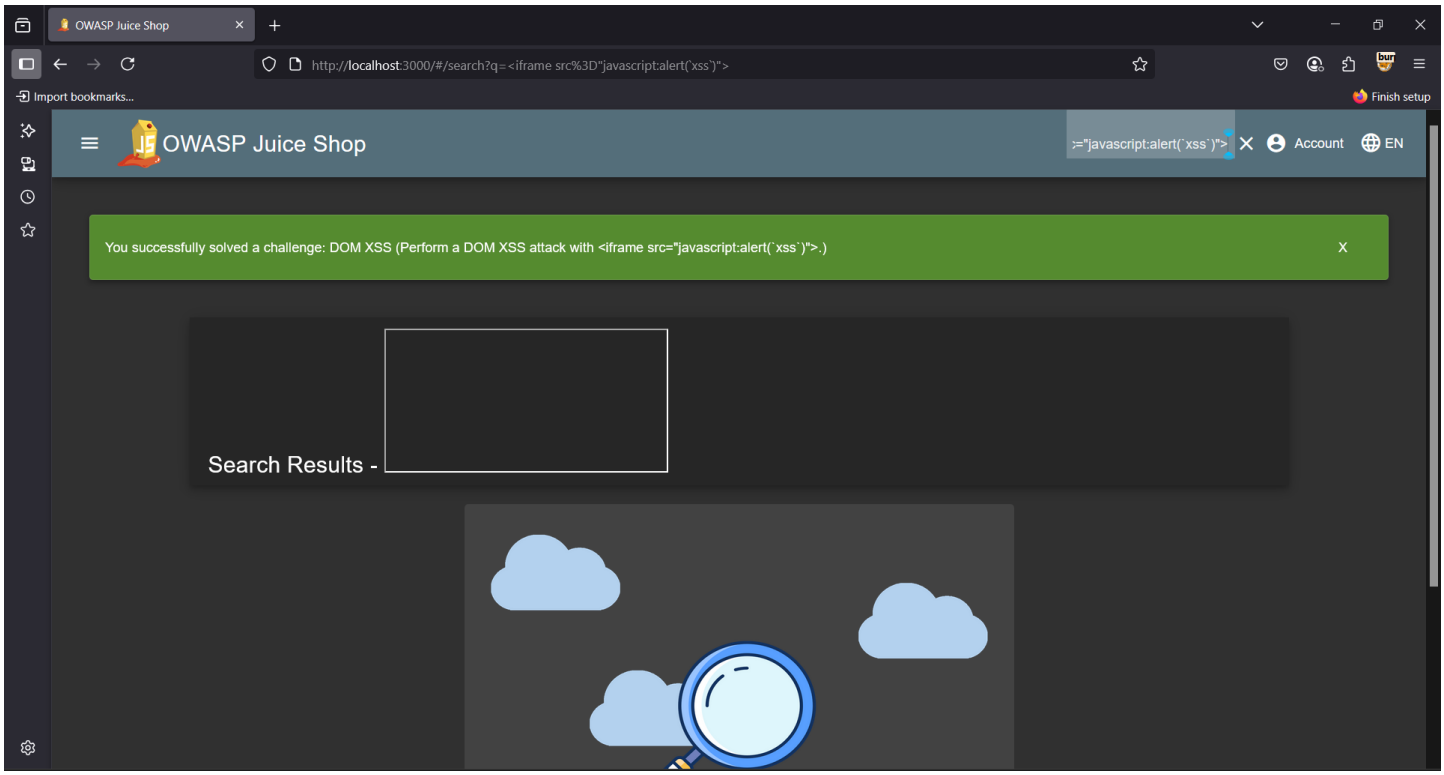
Conclusion: This chain of vulnerabilities following (SQLi → Token Leakage → Weak Hashing) demonstrates a critical security flaw that can lead to full compromise of the web application.

Mitigation

- Implement parameterized SQL queries.
- Sanitize all user input before processing.
- Limit access to debug tools and tokens in production.
- Use strong hashing algorithms such as bcrypt or Argon2.

B. DOM XSS Exploitation

- **Vulnerability:** DOM-based XSS occurs when the JavaScript code on the
- client side processes user input in an unsafe way, allowing script injection.
- **Payload Used:** <iframe src="javascript:alert('xss')">
- **URL:** http://localhost:3000/#/search?q=<iframe src="javascript:alert('xss')">
- **Steps Taken:**
  - Navigated to the search page in Juice Shop.
  - Injected the payload into the search query.
  - Observed a JavaScript alert() box triggered via the iframe tag.
- **Impact:**
  - Confirms the presence of DOM XSS, which can be weaponized for session hijacking, phishing, or redirect attacks.
  - No sanitization or encoding of input before rendering it in the DOM.



Mitigation

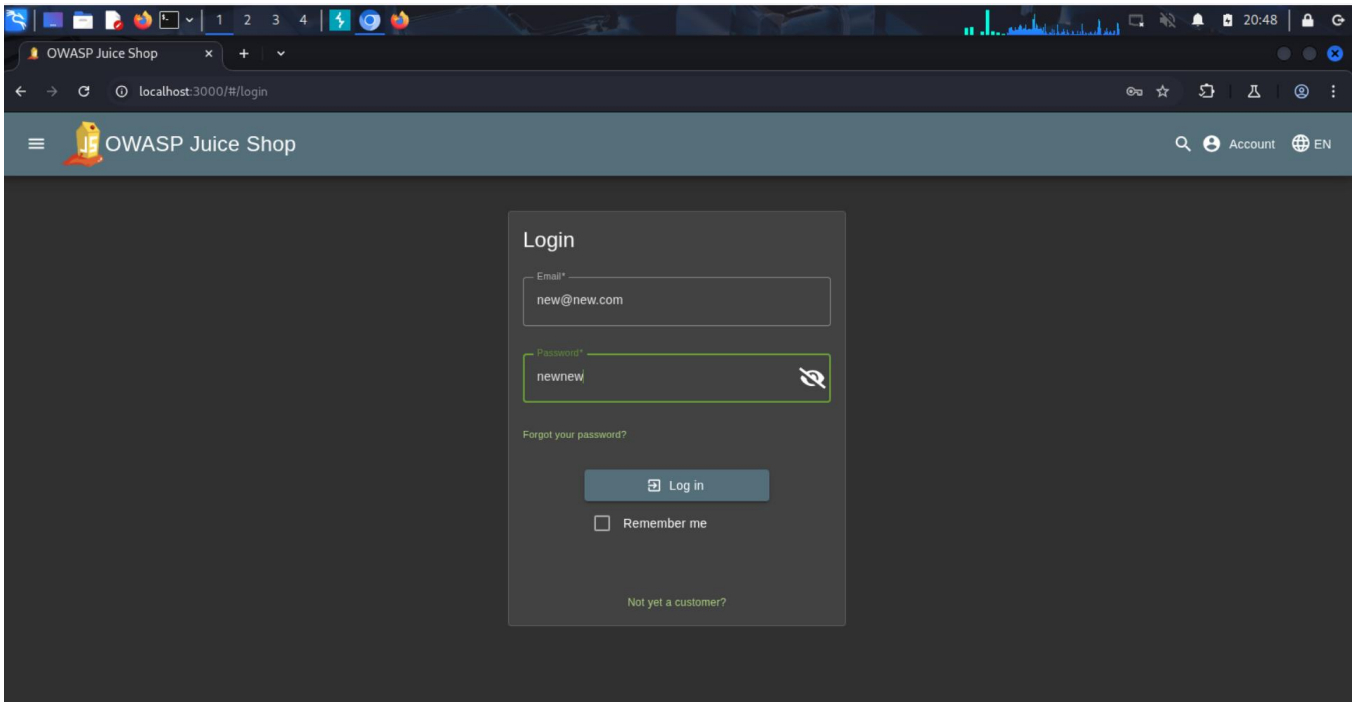
- Apply input sanitization and output encoding.
- Avoid unsafe JavaScript methods (e.g., `innerHTML`) for user-generated content.
- Implement a Content Security Policy (CSP).

C. Password Change Without Current Password Verification

Vulnerability: Password Change Without Current Password Verification
Target: http://localhost:3000/#/account/security
Steps Taken:

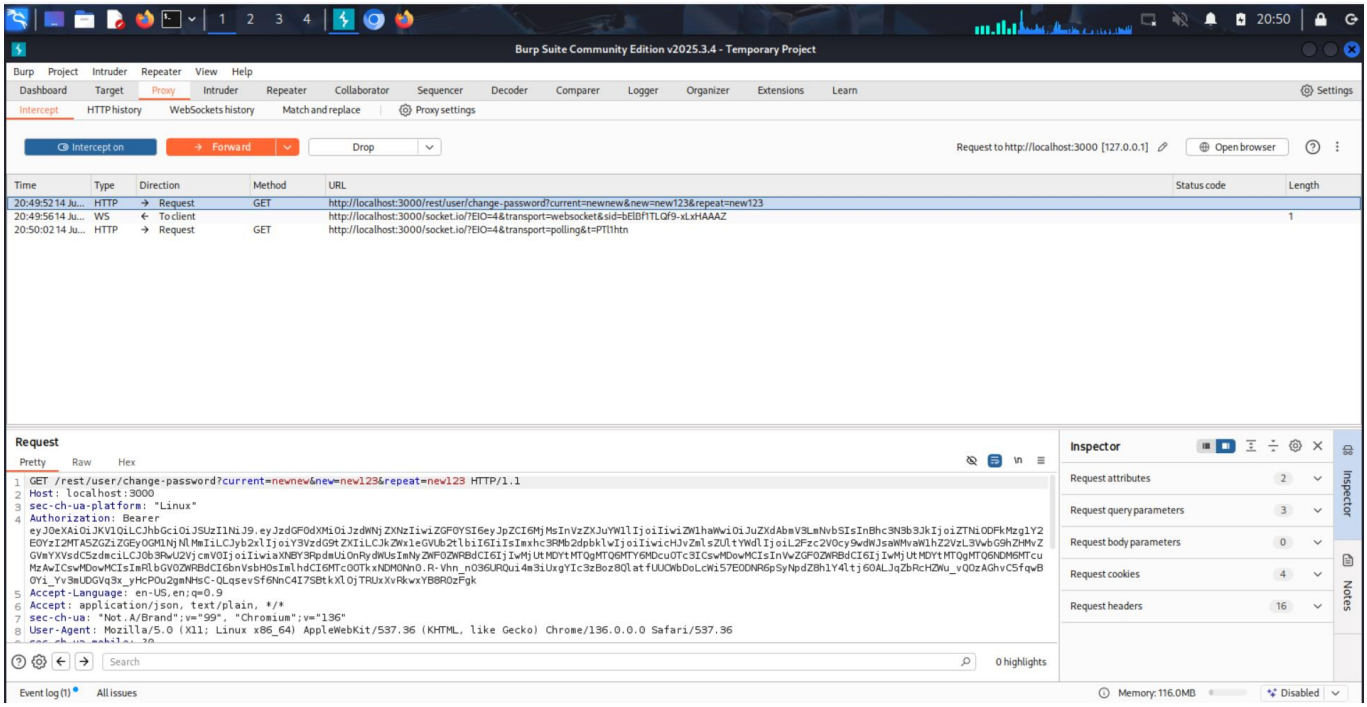
1. User Creation & Login

Created a new account:
Email: new@new.com
Password: newnew
Logged in successfully.



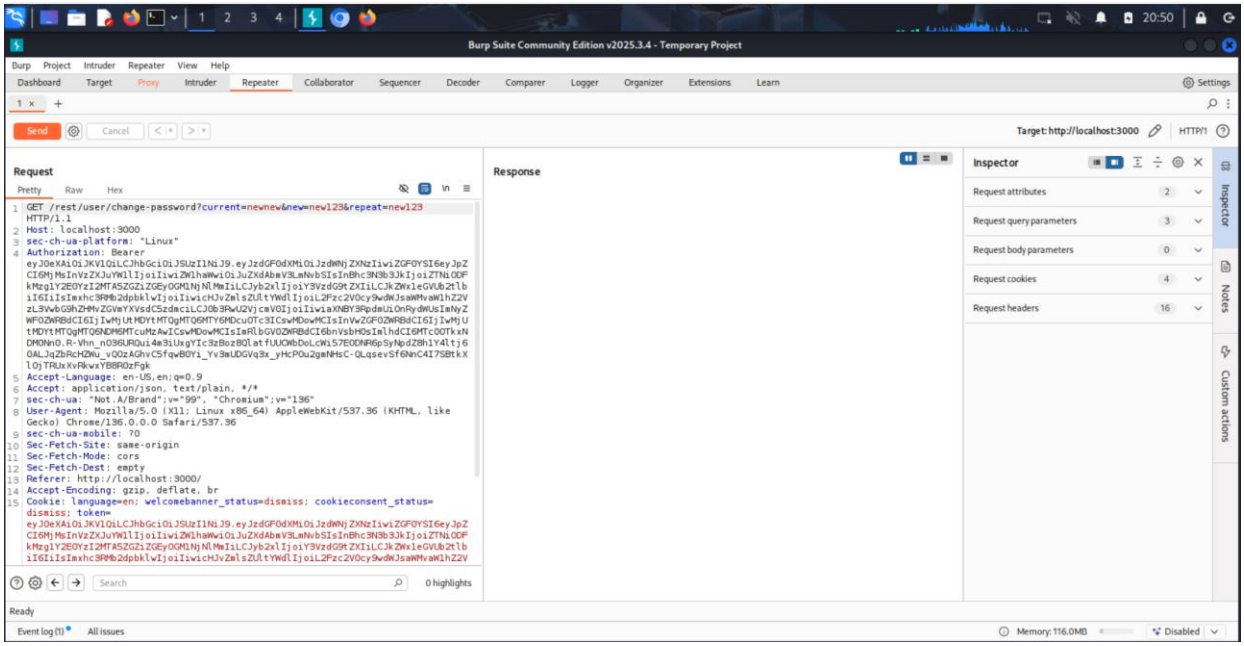
2. Burp Suite Setup

Started Burp Suite and intercepted requests.



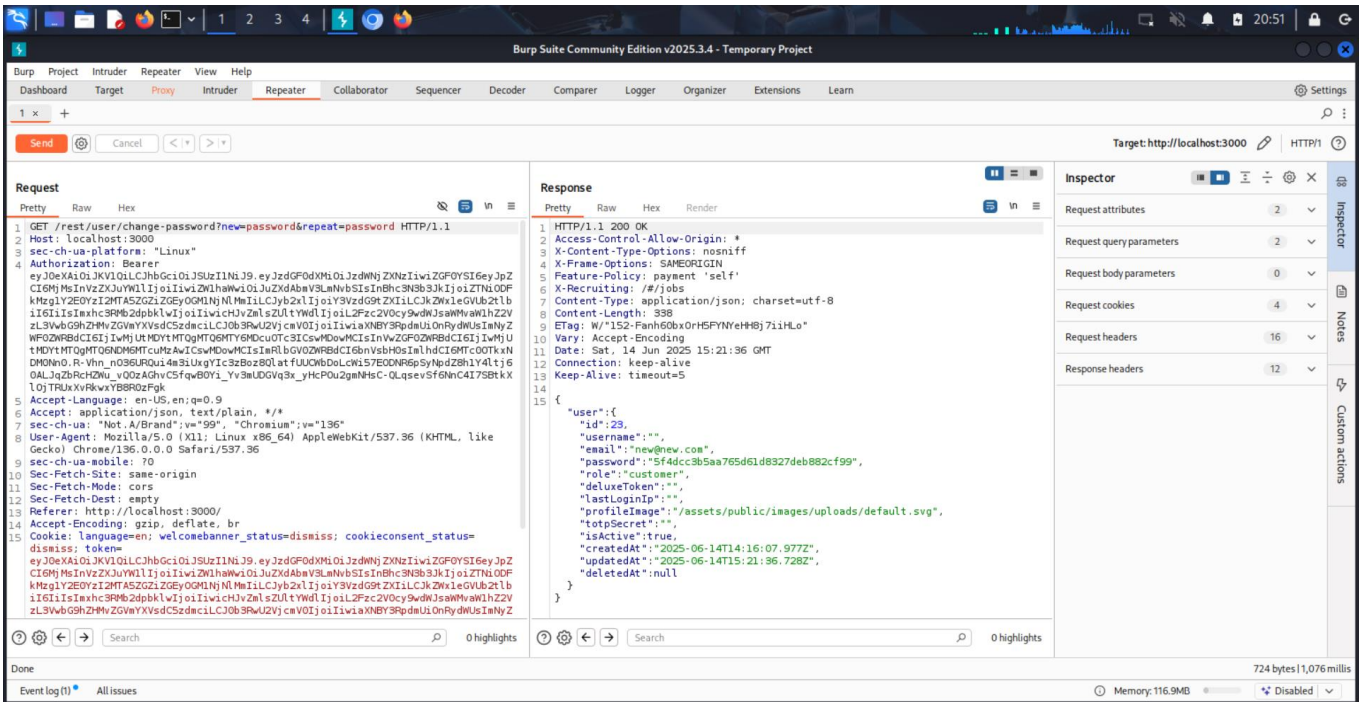
3. Captured Password Change Request

Navigated to the "Change Password" feature and submitted the form.
Intercepted the HTTP PUT request in Burp containing this JSON payload:



4.Sent Request to Repeater

Forwarded it to Burp Repeater for tampering.

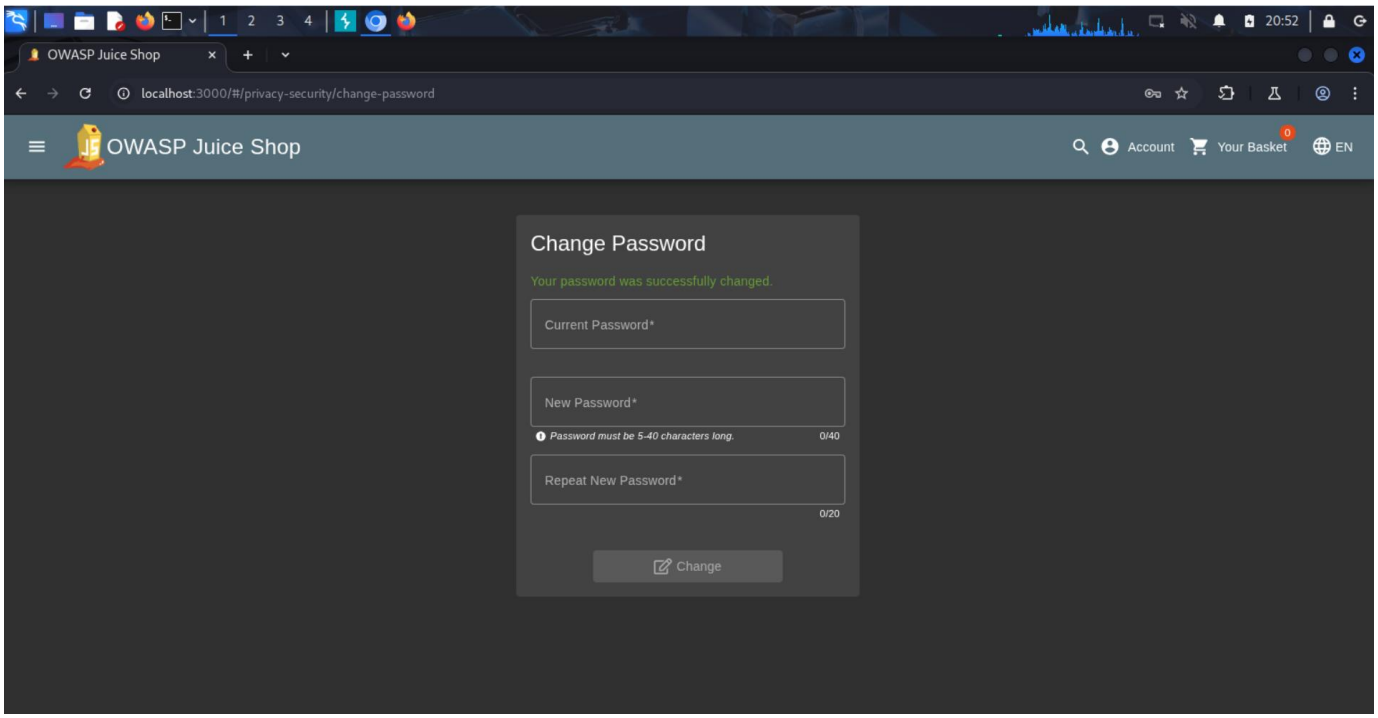


5. Bypassed Current Password Check

Removed the "current" field and sent:

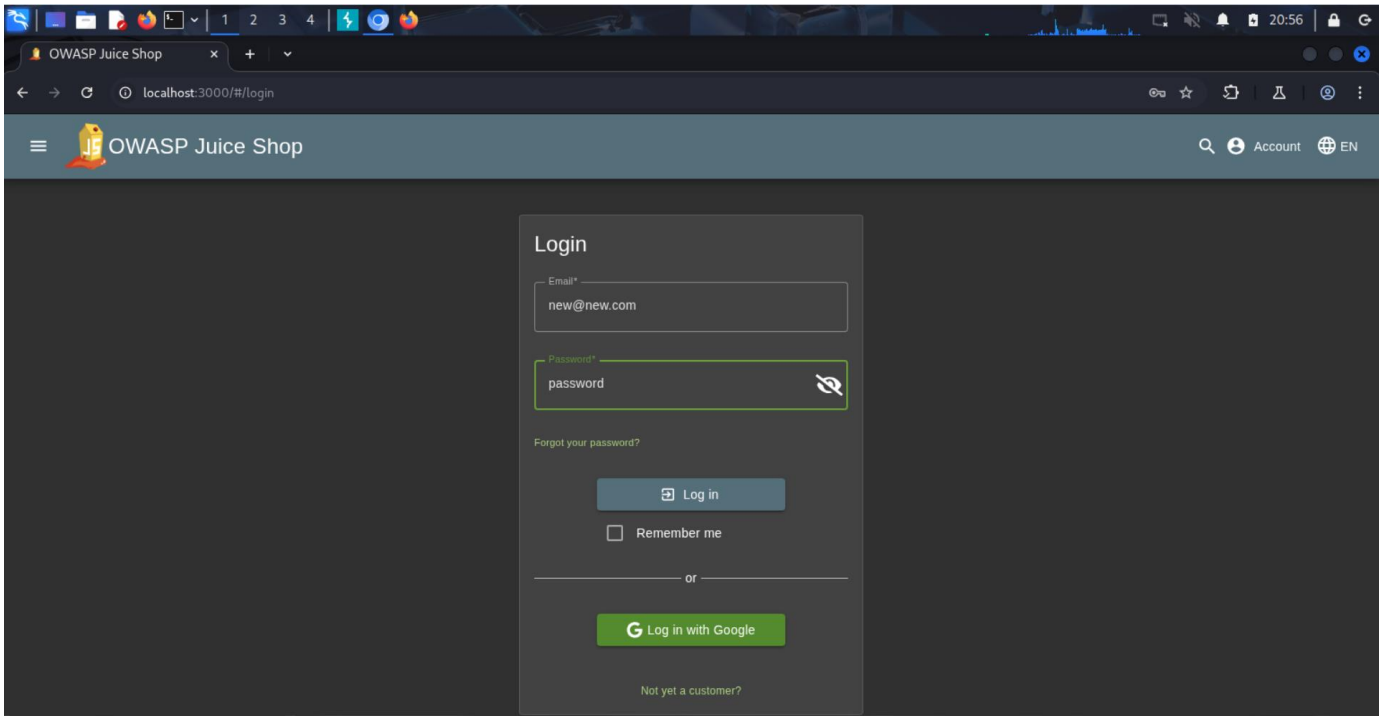
"new": "password",

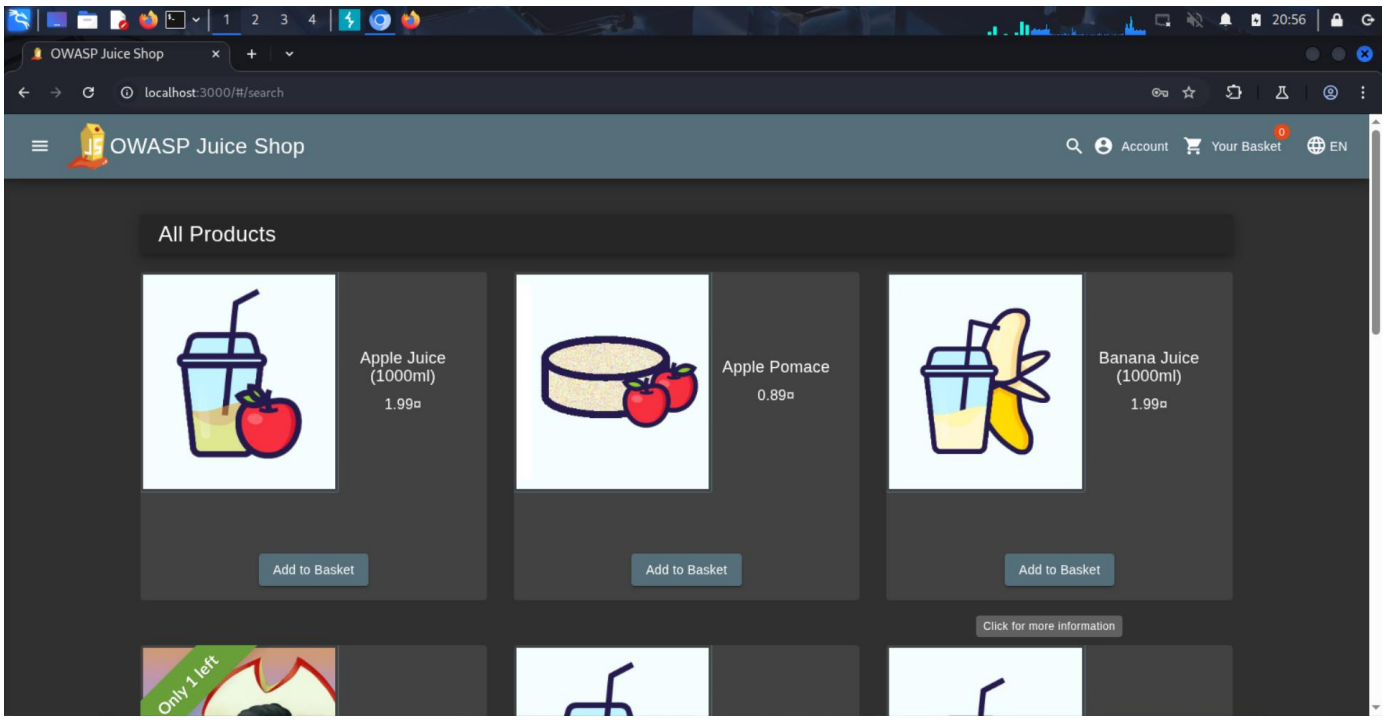
"repeat": "password"



6. Confirmed Exploitation

Logged out and successfully logged back in using:
Email: new@new.com
Password: password





Impact:

No current password verification allows attackers with session access (e.g., via XSS or CSRF) to change a user's password without knowing it.
Leads to account takeover, resulting in loss of user control, data breach, and escalated exploitation.

Mitigation

- Always require the current password before allowing password changes.
- Enforce session re-authentication for sensitive actions.

Risk Summary

Vulnerability	Severity	Exploitability	Business Impact
SQL Injection (Login Bypass)	Critical	Easy	Full user access & privilege escalation
JWT Token Leakage	High	Easy	Session impersonation
MD5 Weak Hashing	High	Easy	Password recovery
DOM-based XSS	High	Moderate	Session hijack / Phishing
Password Change without Verification	High	Easy	Account takeover