Date: May 2025
Prepared By: Sahithi Gudigopuram

Assessment Methodology:

S.No	Test Methodology	Scope
01	Penetration testing execution standards	It is a structured approach to identifying, exploiting, and reporting security vulnerabilities in systems, networks, or applications.
02	OWASP Web Security Testing Guide v4.2	Testing Juice shop vulnerabilities across the login user profile and APIs using tools like Burp Suite

Table of Contents

1. Executive summary

1.1 Summary

A comprehensive penetration test was conducted on the OWASP Juice Shop application, using manual techniques and industry-standard tools. The assessment focused on identifying web vulnerabilities outlined in the OWASP framework. Several high-impact issues were uncovered, including authentication flaws, exposed APIs, and input validation weaknesses, revealing exploitable vectors that threaten application confidentiality and integrity.

1.2. Strength

1. **Basic Security Headers Implemented:** Headers like X-Content-Type-Options: nosniff and X-Frame-Options: SAMEORIGIN are in place, helping defend against MIME-type sniffing and clickjacking.
2. **Access Control Enforcement on Admin Routes:** Attempts to access privileged sections like the admin panel were met with proper HTTP 403 responses, showing baseline role-based access control.
3. **HTTPS Ready & No Sensitive Data in Transit:** No indications of sensitive data (passwords, tokens) being transmitted insecurely via GET parameters or exposed in URLs.
4. **No Critical Customer Data Exposure:** During API testing, no personal financial data, credit card numbers, or identity information were leaked.
5. **Modern Tech Stack & Feature Flags:** The app leverages Angular, Node.js, and includes modern headers like feature-policy, demonstrating up-to-date technology use.
6. **Stable Session Management Logic:** While session sharing was discovered, the core login/logout flow was responsive and stable across sessions.
7. **Rich Developer Functionality:** Structured, RESTful API endpoints make the app test-friendly and modular, simplifying debugging and future hardening.

1.3. Weakness

1. **Unprotected API Endpoint**: /api/Challenges exposed sensitive challenge data without authentication.
2. **Weak Password Reset Logic:** Tokens weren't clearly validated same reset flow was shown to all users.
3. **IDOR (Insecure Direct Object Reference):** Manipulating object IDs (like basket IDs) gave access to other users' data.
4. **Cross-Site Scripting (XSS):** Reflected, stored, and DOM-based XSS all successfully executed.
5. **SQL Injection**: Auth bypass via ' OR 1=1 -- and error-based injection vectors confirmed.
6. **CSRF Vulnerability**: Lack of CSRF protection allowed silent password changes via crafted HTML.
7. **Missing Security Headers**: Absence of key headers like Content-Security-Policy, Strict-Transport-Security, etc.

1.4. Business Impact

1. **Unauthorised Access**: Exploitable flaws like IDOR and SQL Injection can allow attackers to bypass authentication and access other users' data.
2. **Data Leakage & Tampering**: Sensitive information and session tokens can be exposed or manipulated, compromising data integrity and confidentiality.
3. **Account Takeover Risks**: CSRF and weak session management may lead to full user account compromise without their awareness.
4. **Brand & Reputation Damage**: Persistent client-side attacks like XSS can be leveraged for phishing, reducing user trust and confidence.
5. **Regulatory Non-Compliance**: Failure to implement security best practices may lead to violations of data protection standards (e.g., GDPR, PCI-DSS)

2. Technical Summary

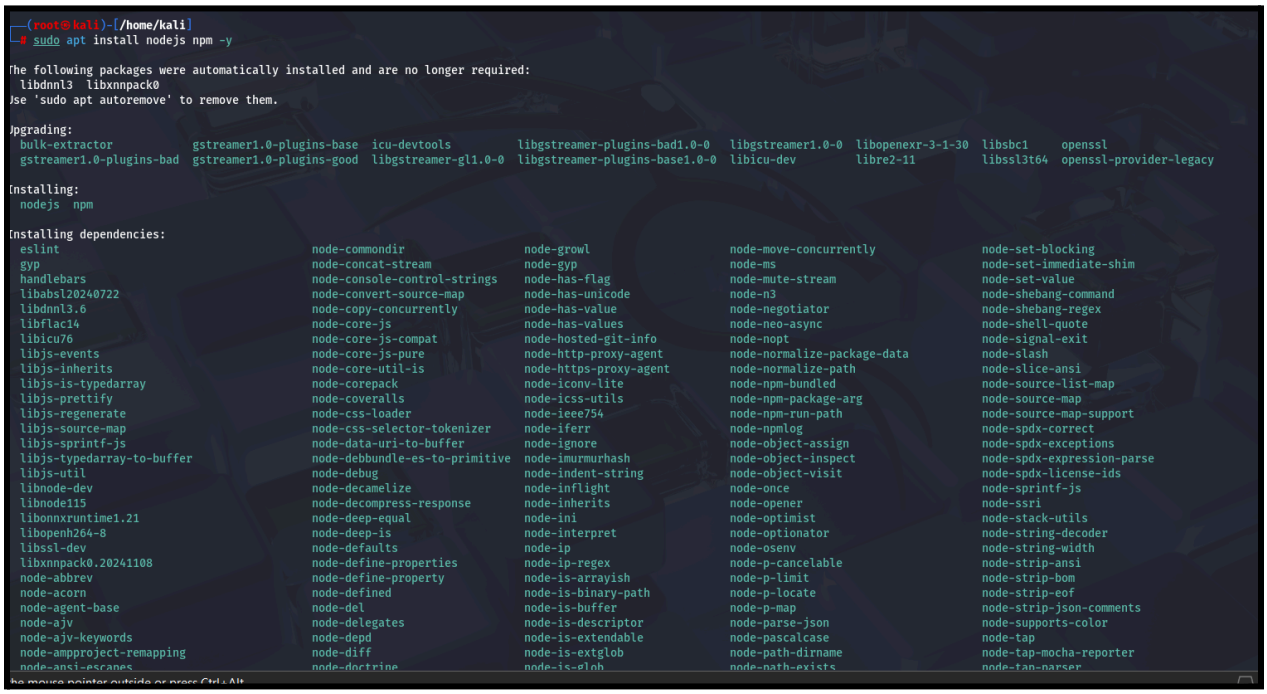
2.1. Scope & Objective

The assessment focused on evaluating the security posture of the OWASP Juice Shop application hosted at **http://10.0.0.212:3000**. Testing included critical web components such as login, search, feedback forms, and exposed API endpoints. The internal infrastructure and systems beyond this application were not included in scope. The objective was to identify vulnerabilities through simulated attack techniques, with emphasis on input handling, session management, access control, and authentication mechanisms. Findings were used to recommend actionable security improvements.

2.2. Testing Environment

Setting up OWASP Juice shop(Kali linux)

- Step 1: Open the terminal and update package repositories “sudo apt update”
- Step 2: Install Node.js (JavaScript runtime) “sudo apt install nodejs npm -y”



- Step 3: npm (Node package manager)“sudo apt install npm -y”

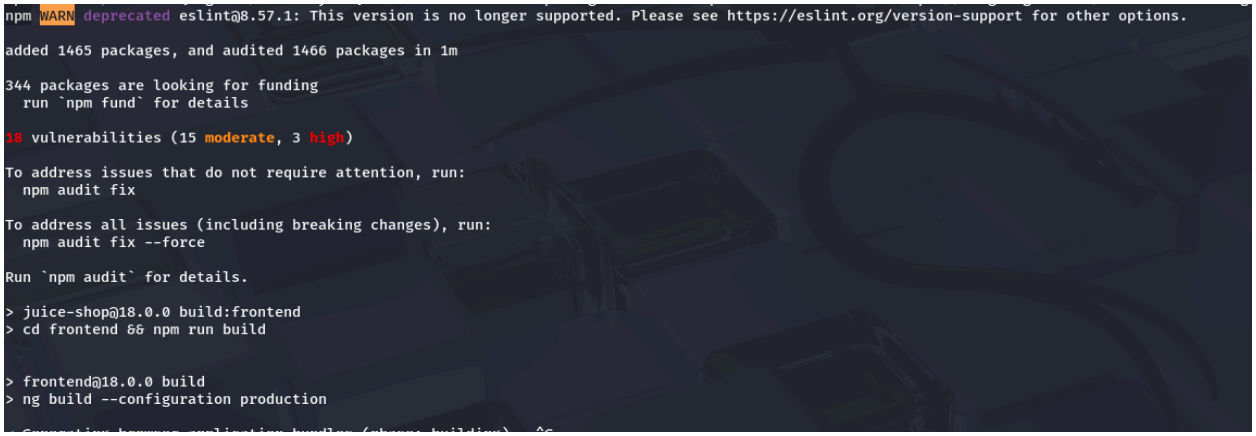
```
┌──(root㉿kali)-[/home/kali]
└─# sudo apt install npm -y

npm is already the newest version (9.2.0~ds1-3).
The following packages were automatically installed and are no longer required:
  libdnnl3  libxnnpack0
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 1973

┌──(root㉿kali)-[/home/kali]
└─# npm -v

9.2.0
```

- Step 4: Copy the official repository URL from GitHub and clone it
- Now OWASP JuiceShop is installed

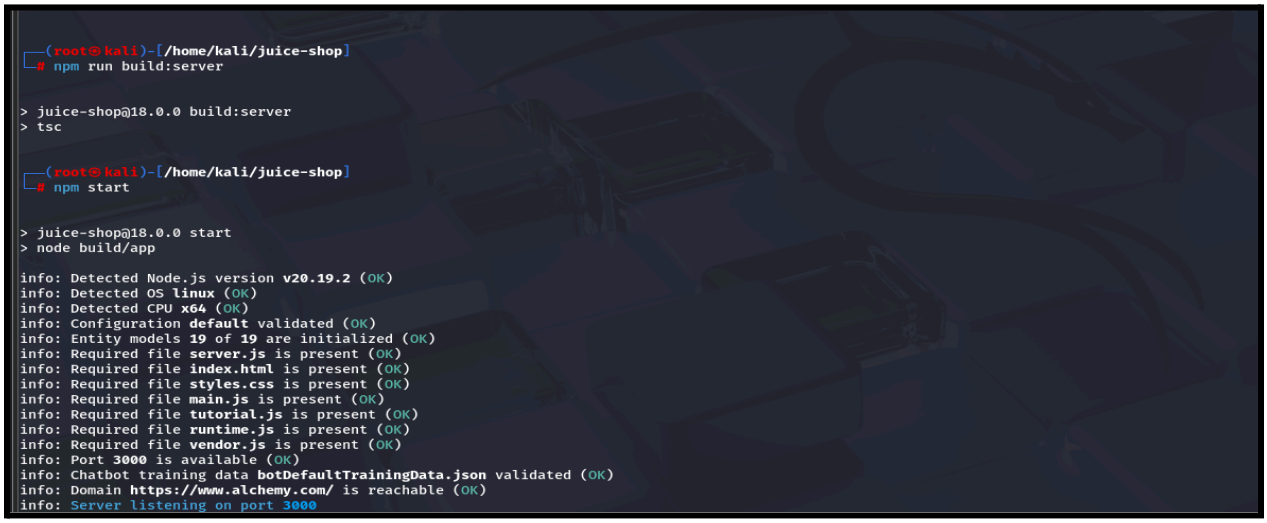
```
┌──(root㉿kali)-[/home/kali]
└─# git clone https://github.com/juice-shop/juice-shop.git
Cloning into 'juice-shop'...
remote: Enumerating objects: 138115, done.
remote: Total 138115 (delta 0), reused 0 (delta 0), pack-reused 138115 (from 1)
Receiving objects: 100% (138115/138115), 246.54 MiB | 26.56 MiB/s, done.
Resolving deltas: 100% (107977/107977), done.

┌──(root㉿kali)-[/home/kali]
└─# ls
create_jwt.py  Desktop    Downloads  juice-shop     Music  Pictures  requesttxt                                      splunk-9.2.1.deb  Templates  testfile.txt
csrf           Documents  DVWA       jwt-demo.html  myenv  Public    splunk-9.2.1-ae6821b2f2ec-linux-2.6-amd64.deb  splunk-9.4.2.deb  testdir    Videos

┌──(root㉿kali)-[/home/kali]
└─#
```

- Step 5: Change to the Juice Shop directory “cd juice-shop”
- Step 6: Now, install all project dependencies “npm install”

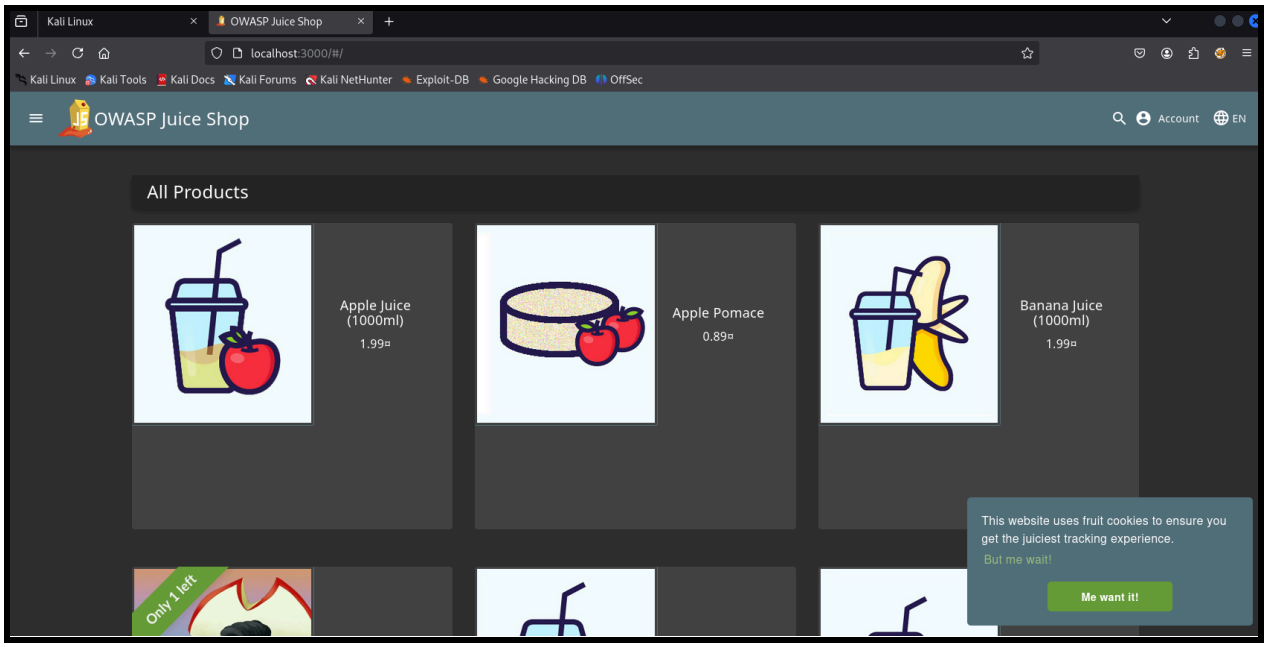
```
┌──(root㉿kali)-[/home/kali/juice-shop]
└─# npm install
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'libxmljs2@0.37.0',
npm WARN EBADENGINE   required: { node: '>=22' },
npm WARN EBADENGINE   current: { node: 'v20.19.2', npm: '9.2.0' }
npm WARN EBADENGINE }
npm WARN deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is
 more comprehensive and powerful.
npm WARN deprecated @npmcli/move-file@1.1.2: This functionality has been moved to @npmcli/fs
npm WARN deprecated formatio@1.1.1: This package is unmaintained. Use @sinonjs/formatio instead
npm WARN deprecated eivindfjeldstad-dot@0.0.1: Use @eivifj/dot instead
npm WARN deprecated samsam@1.1.2: This package has been deprecated in favour of @sinonjs/samsam
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated lodash.get@4.4.2: This package is deprecated. Use the optional chaining (?.) operator instead.
npm WARN deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated fstream@1.0.12: This package is no longer supported.
npm WARN deprecated are-we-there-yet@1.1.7: This package is no longer supported.
npm WARN deprecated gauge@2.7.4: This package is no longer supported.
npm WARN deprecated jws@3.2.6: Security update: Versions below 3.0.0 are deprecated.
npm WARN deprecated npmlog@4.1.2: This package is no longer supported.
npm WARN deprecated vm2@3.9.17: The library contains critical security issues and should not be used for production! The maintenance of the project has been discontinued. Consider migrating your code to i
ed-vm.
npm WARN deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm WARN deprecated messageformat@2.3.0: Package renamed as '@messageformat/core', see messageformat.github.io for more details. 'messageformat@4' will eventually provide a polyfill for Intl.MessageFormat
e it's been defined by Unicode & ECMA.
npm WARN deprecated @humanwhocodes/config-array@0.13.0: Use @eslint/config-array instead
npm WARN deprecated @humanwhocodes/object-schema@2.0.3: Use @eslint/object-schema instead
npm WARN deprecated ecstatic@3.3.2: This package is unmaintained and deprecated. See the GH Issue 259.
npm WARN deprecated osenv@0.1.5: This package is no longer supported.
npm WARN deprecated @types/socket.io-parser@3.0.0: This is a stub types definition. socket.io-parser provides its own type definitions, so you do not need this installed.
npm WARN deprecated @types/express-unless@2.0.3: This is a stub types definition. express-unless provides its own type definitions, so you do not need this installed.
npm WARN deprecated eslint-config-standard-with-typescript@43.0.1: Please use eslint-config-love instead.
```



- Compile the frontend and backend (TypeScript → JavaScript) “npm run build: frontend npm run build: server”
- Launch the application “npm start”



- Open browser “http://localhost:3000”



2.3. Testing Environment Overview

Component	Details
Target Application	The OWASP Juice Shop host is located on Kali Linux IP 10.0.0.212
Attacker Machine	An Ubuntu-based system used to perform reconnaissance and exploitation
Network setup	Internal lab using Bridged Adapter
Browsers & tools	Firefox, Burp suite, Nmap, Nikto, wappalzer..etc
Isolation	The lab was segmented from the external network to simulate a secure testing scope.

2.4. Technical Impact

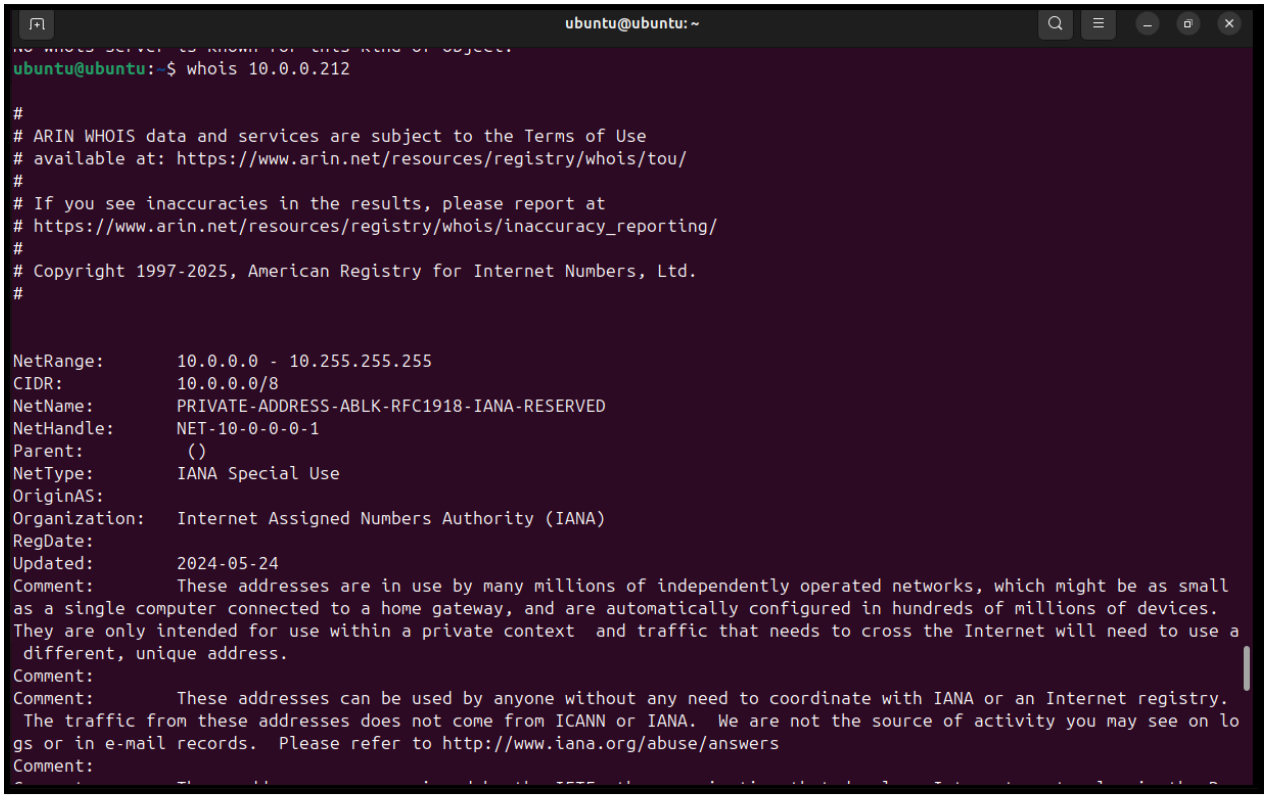
The vulnerabilities identified during the assessment pose significant technical risks to the confidentiality, integrity, and availability of the application. Exploits such as SQL Injection can allow unauthorised database access or manipulation, while XSS and CSRF attacks enable remote attackers to hijack sessions and perform unauthorised actions on behalf of users. IDOR and broken access controls further permit exposure or modification of user-specific resources. Together, these flaws could undermine secure session management, leak sensitive data, and facilitate attacker persistence within the application.

2.5. Table of Findings

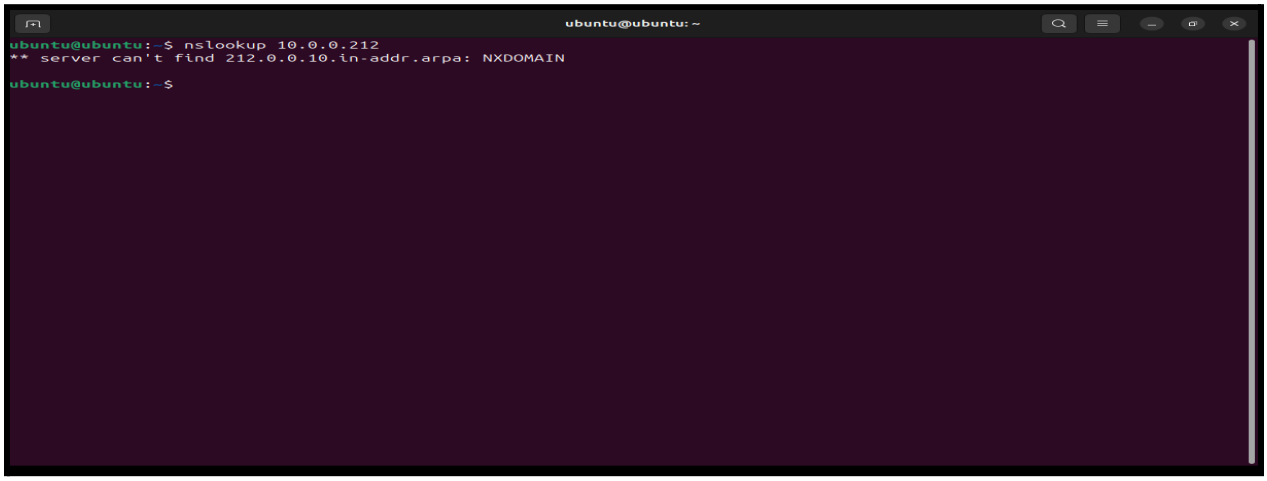
S.No	Category	Vulnerability	Risk level
1	Cross-site scripting(XSS)	Reflected, stored, DOM-based XSS	High
2	Injection	SQL Injection	High
3	Cross-site request forgery	CSRF on password change	High
4	Access Control	IDOR(Insecure direct object references)	Hight
5	Security Misconfiguration	Missing Security Headers	Medium
6	Unprotected API	/api/challenges exposed	Medium
7	Web Server Profiling	Nikto scan	Medium
8	Technology Fingerprinting	Wapplyzer	Low
9	Technology Fingerprinting	Whatweb	Low
10	Network mapping	Nmap scan	Info

3. Passive Reconnaissance

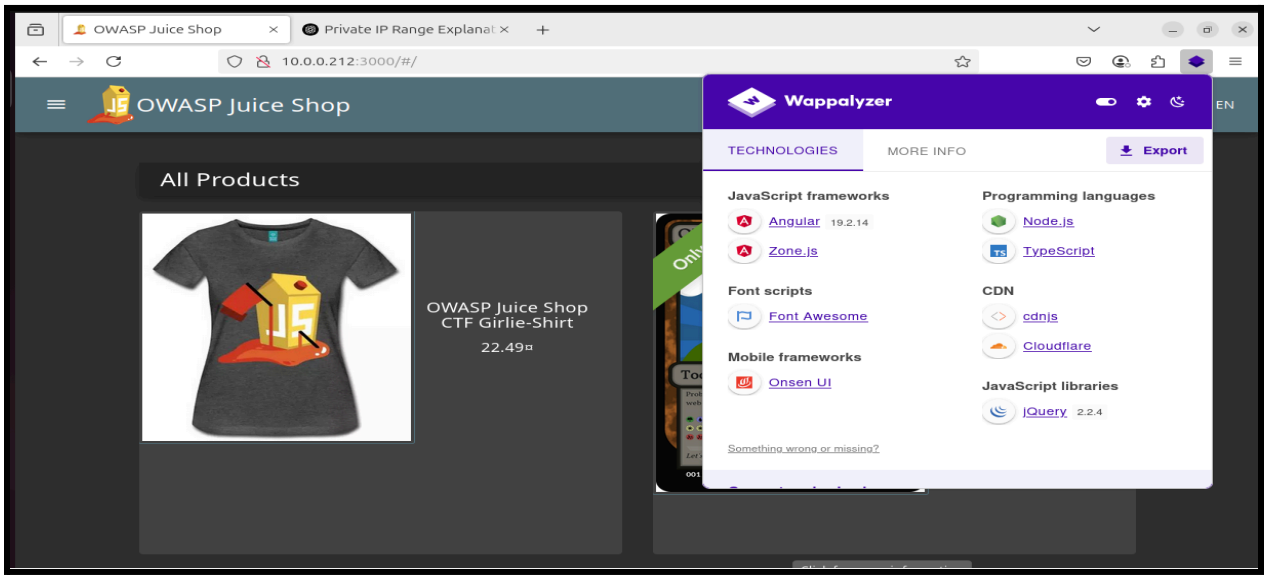
3.1. Whois: The whois result shows that IP 10.0.0.212 belongs to the private IP range 10.0.0.0/8, reserved by IANA for internal network use. This means it's not assigned to any public organisation and is used in local environments.



3.2. nslookup: This lookup attempted a reverse DNS resolution and still returned `NXDOMAIN`, which is also expected in most internal or lab setups unless you've configured reverse DNS.



3.3. Wappalyzer: The target web application is **OWASP Juice Shop**, built with **Angular 19.2.14**, **Node.js**, and **TypeScript**, and includes outdated components such as **jQuery 2.2.4**, presenting potential vectors for exploitation and security testing.



3.4. WhatWeb:

It is a passive reconnaissance tool used to identify technologies powering a web application by analysing HTTP headers and page content. It detected components such as HTML5, jQuery 2.2.4, and headers like X-Frame-Options and Feature-Policy. This non-intrusive scan provided insights into the application's framework and potential outdated libraries. It helped establish the technology stack and informed later vulnerability assessments.

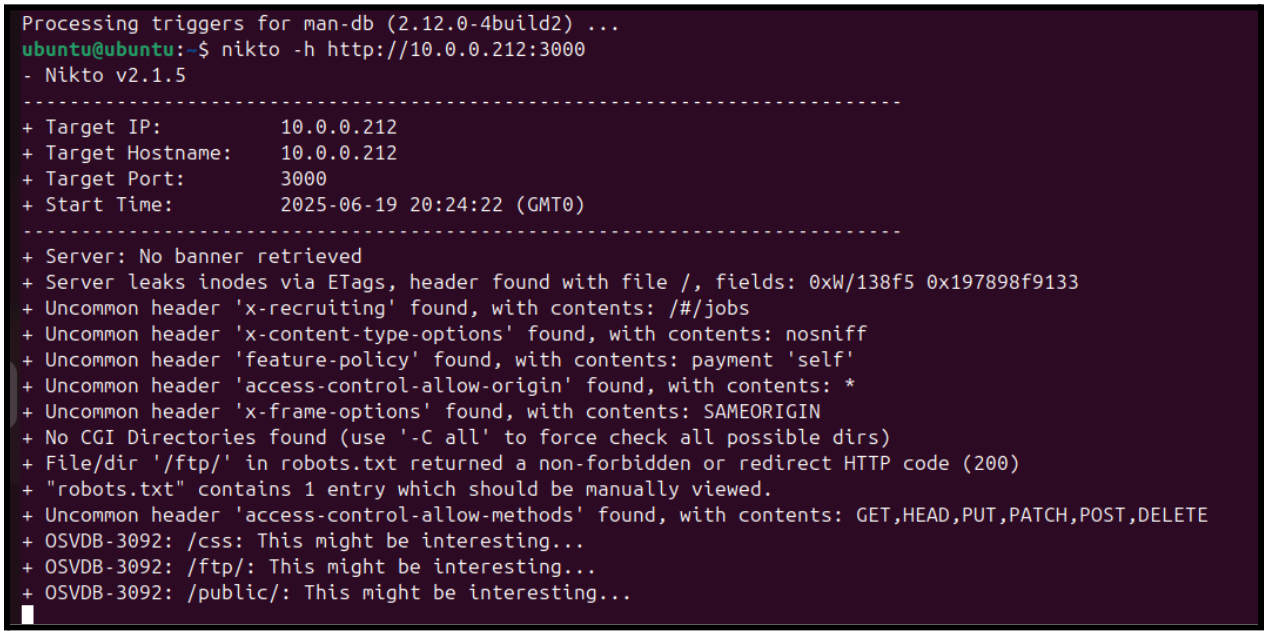
```
Processing triggers for libc-bin (2.39-0ubuntu8.4) ...
root@sahithi-VMware-Virtual-Platform:/home/sahithi# whatweb http://10.0.0.212:3000
http://10.0.0.212:3000 [200 OK] Country[RESERVED][ZZ], HTML5, IP[10.0.0.212], JQuery[2.2.4], Script[module], Title[OWASP Juice Shop], UncommonHeaders[access-cont
rol-allow-origin,x-content-type-options,feature-policy,x-recruiting], X-Frame-Options[SAMEORIGIN]
root@sahithi-VMware-Virtual-Platform:/home/sahithi# whatweb -v http://10.0.0.212:3000
WhatWeb report for http://10.0.0.212:3000
Status    : 200 OK
Title     : OWASP Juice Shop
IP        : 10.0.0.212
Country   : RESERVED, ZZ

Summary   : HTML5, JQuery[2.2.4], Script[module], UncommonHeaders[access-control-allow-origin,x-content-type-options,feature-policy,x-recruiting], X-Frame-Option
s[SAMEORIGIN]

Detected Plugins:
[ HTML5 ]
        HTML version 5, detected by the doctype declaration


[ JQuery ]
        A fast, concise, JavaScript that simplifies how to traverse
        HTML documents, handle events, perform animations, and add
        AJAX.

        Version      : 2.2.4
        Website      : http://jquery.com/

[ Script ]
        This plugin detects instances of script HTML elements and
        returns the script language/type.

        String       : module

[ UncommonHeaders ]
        Uncommon HTTP server headers. The blacklist includes all
```

```
        the standard headers and many non standard but common ones.
        Interesting but fairly common headers should have their own
        plugins, eg. x-powered-by, server and x-aspnet-version.
        Info about headers can be found at www.http-stats.com

        String       : access-control-allow-origin,x-content-type-options,feature-policy,x-recruiting (from headers)

[ X-Frame-Options ]
        This plugin retrieves the X-Frame-Options value from the
        HTTP header. - More Info:
        http://msdn.microsoft.com/en-us/library/cc288472%28VS.85%29.
        aspx

        String       : SAMEORIGIN

HTTP Headers:
        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
        Accept-Ranges: bytes
        Cache-Control: public, max-age=0
        Last-Modified: Tue, 24 Jun 2025 01:53:33 GMT
        ETag: W/"138f5-1979fa3fb0f"
        Content-Type: text/html; charset=UTF-8
        Vary: Accept-Encoding
        Content-Encoding: gzip
        Date: Tue, 24 Jun 2025 02:13:28 GMT
        Connection: close
        Transfer-Encoding: chunked

root@sahithi-VMware-Virtual-Platform:/home/sahithi# whatweb -i urls.txt
No targets selected
root@sahithi-VMware-Virtual-Platform:/home/sahithi#
```

4. Active Reconnaissance

4.1. Nikto

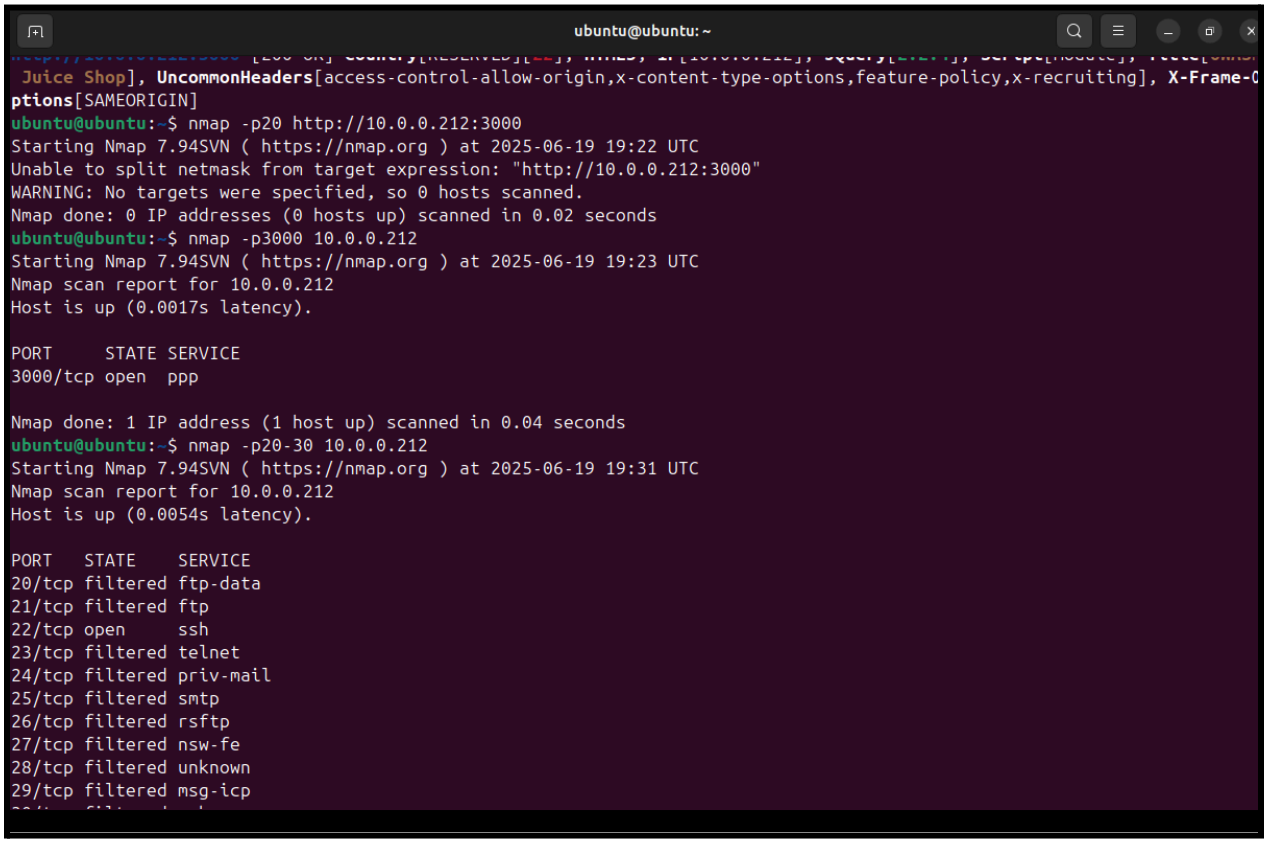
This scan on http://10.0.0.212:3000 uncovered multiple signs of insecure configuration and information exposure. Publicly accessible paths like **/ftp/, /css/,** and **/public/**—some even indexed in **robots.txt—**may unintentionally leak sensitive content or support files. Headers such as **x-recruiting** can expose internal tooling or company tech stacks if linked to job platforms or developer resources, while others like feature-policy suggest modern browser behaviour controls are in place. The server discloses inode information through weakly configured ETag headers, aiding potential fingerprinting attempts. Additionally, the permissive **CORS** setting (Access-Control-Allow-Origin: *) and absence of a server banner reflect a mix of basic security awareness alongside overlooked risks, warranting further manual inspection of exposed endpoints.



- **Remediation:** Remove or restrict access to sensitive directories like **/ftp/ and /public/,** and implement missing headers such as **Content-Security-Policy** and **Strict-Transport-Security** to reduce exposure.

4.2. Nmap

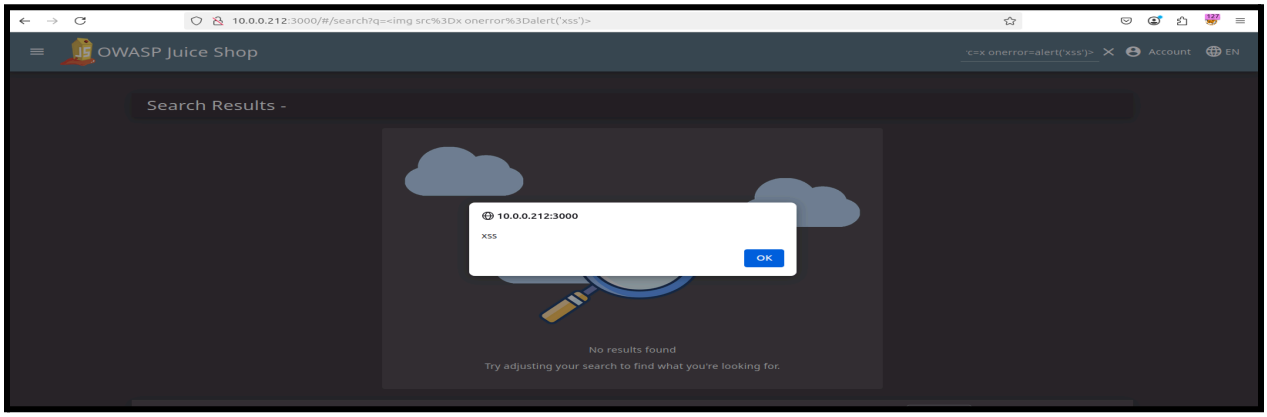
I discovered two open ports on the target system (10.0.0212), port 22 running SSH, which could allow secure remote access if valid credentials are found, and port 3000, hosting an unknown service labelled "ppp" that requires further investigation. These findings will guide the next steps in enumeration and potential exploitation.

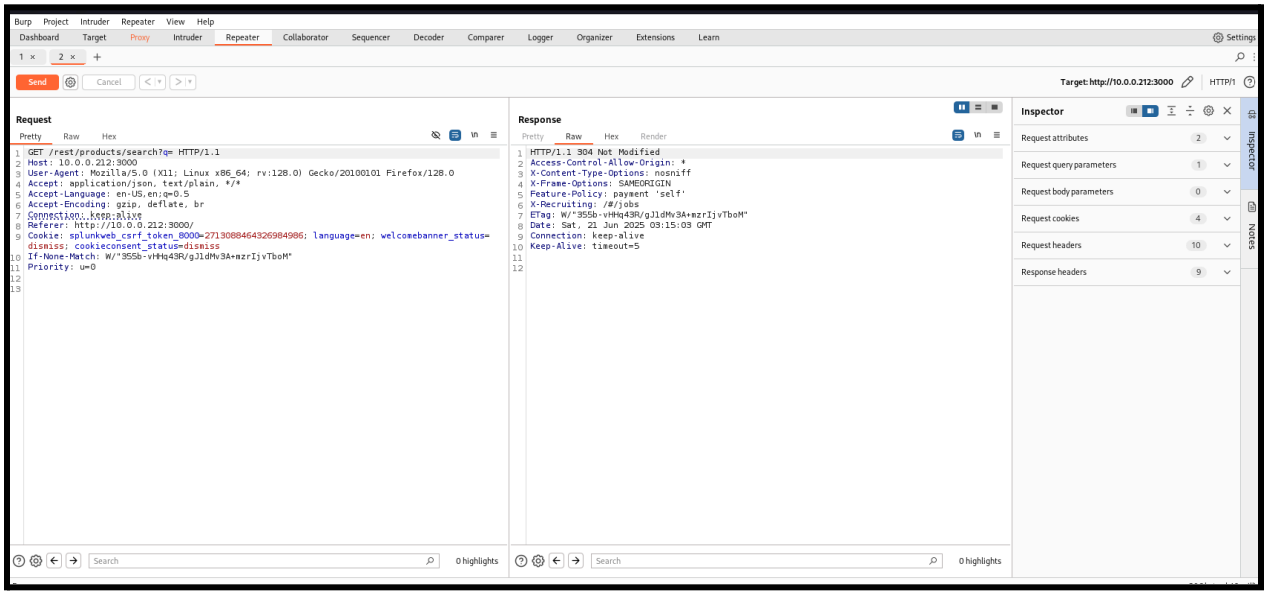


- **Remediation:** Restrict unnecessary open ports (e.g., port 22 if not needed) using firewall rules. Ensure only required services are exposed to trusted networks

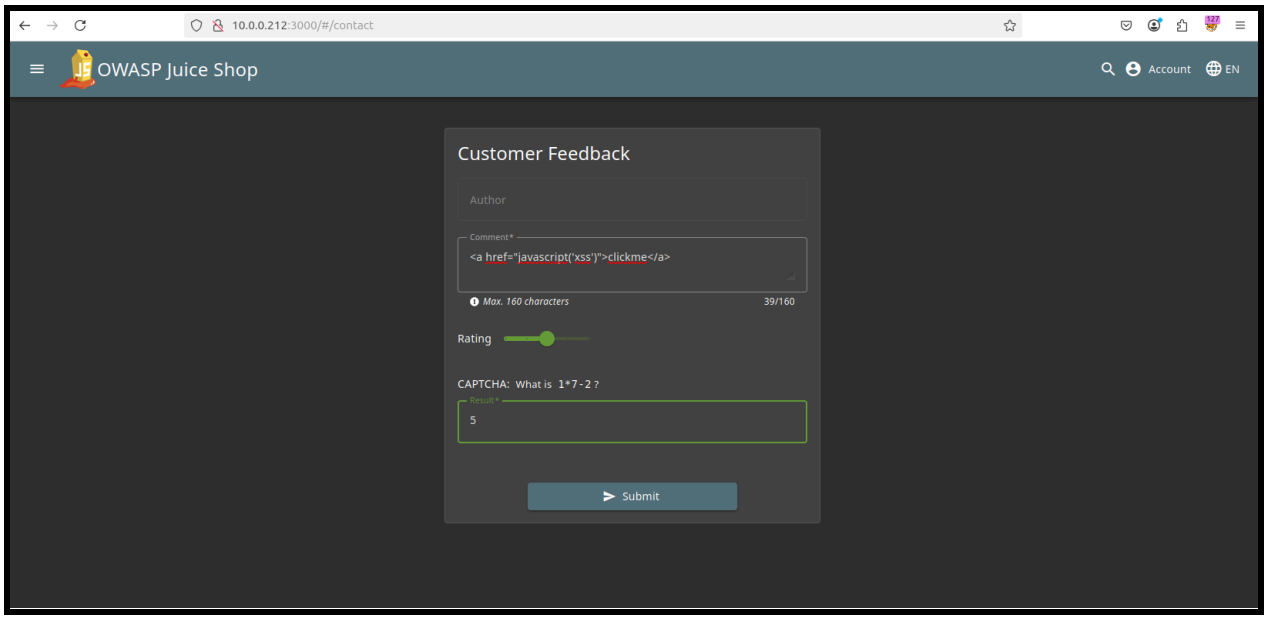
4.3. Cross-site scripting(xss)

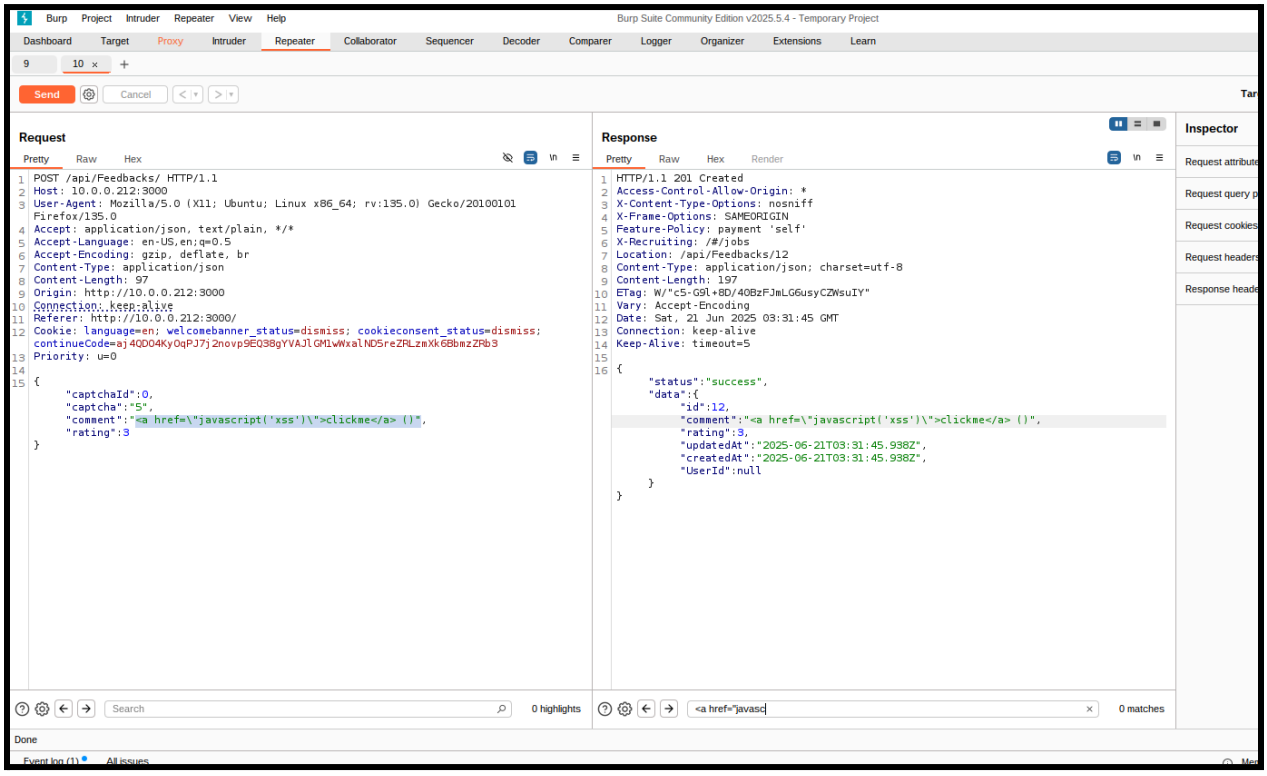
- Reflected XSS in search functionality using **<img src=x onerror=alert('xss')>** payload. Impact: Arbitrary JavaScript execution in the user's browser, allowing session hijacking or phishing attacks.



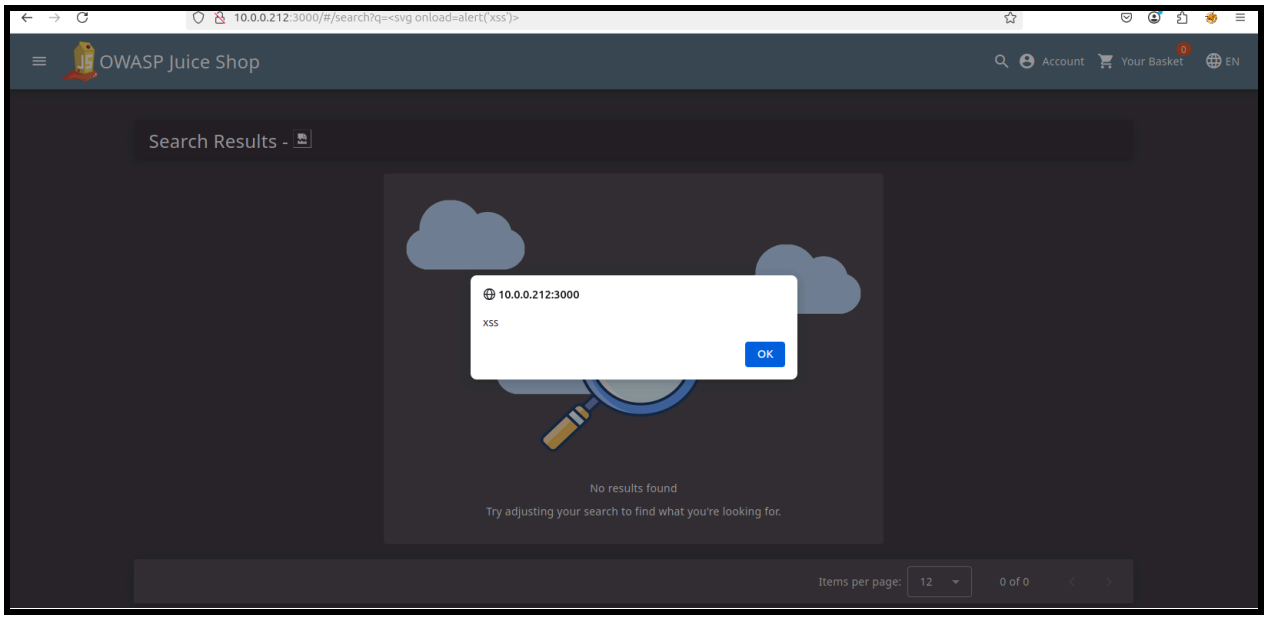


- Stored XSS via feedback input using clickme payload. **Impact:** Malicious JavaScript executes when the victim clicks the injected link, leading to session hijacking





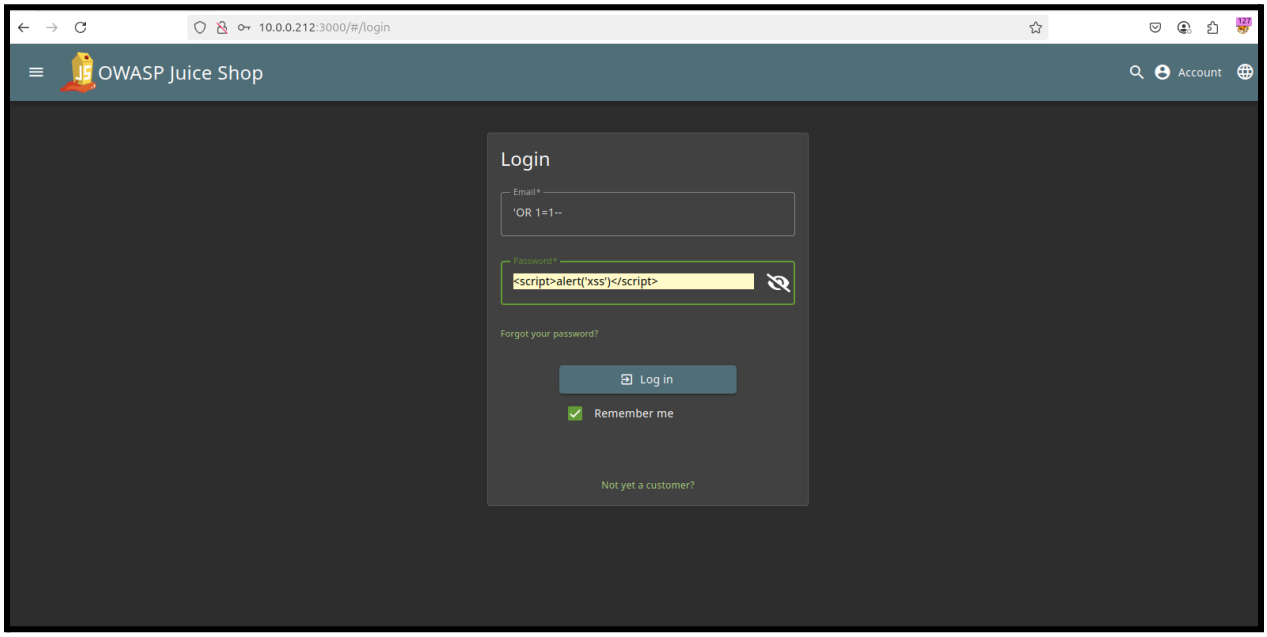
- DOM-based XSS in search feature via q parameter using <svg onload=alert('xss')>. **Impact:** Payload executes in the browser through client-side injection, allowing arbitrary JavaScript execution.

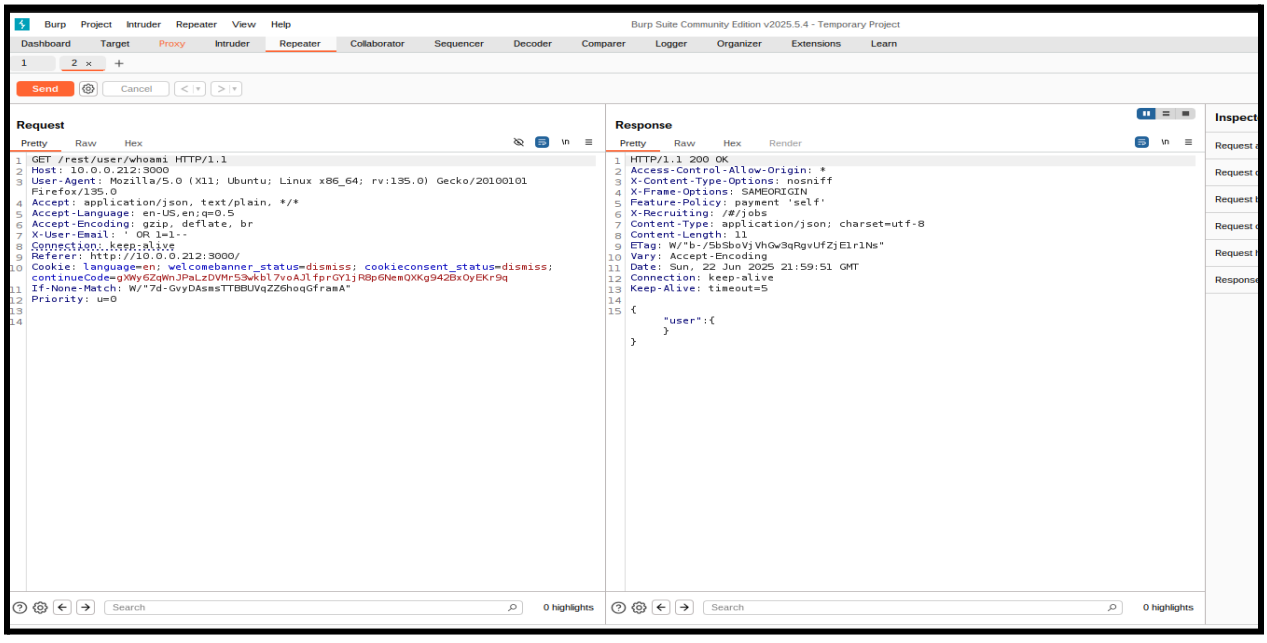


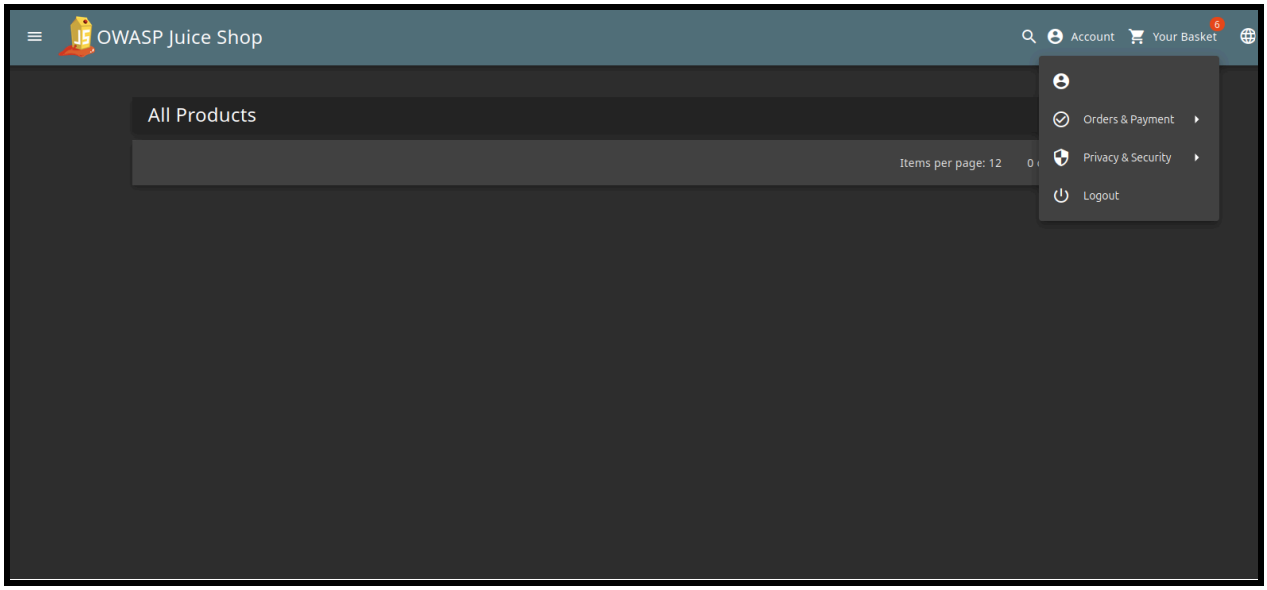
- **Remediation:**
    1. **Sanitise and validate all user inputs** on both client and server sides—reject or escape HTML, JavaScript, and script-related characters.
    2. **Implement context-aware output encoding** (e.g., HTML, URL, or JavaScript encoding) and deploy a strong Content Security Policy (CSP) to reduce script execution risk.

4.4. SQL injection

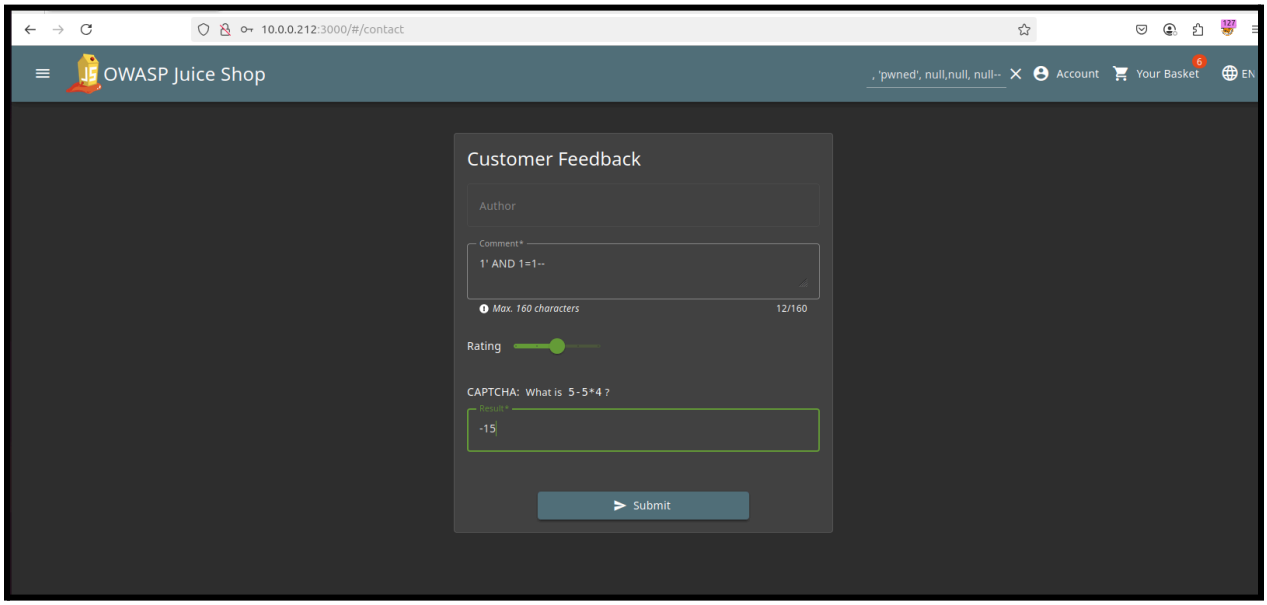
SQL Injection in login form using ' OR 1=1 -- payload to bypass authentication. **Impact:** User gained unauthorised access without valid credentials, demonstrating critical backend query manipulation.

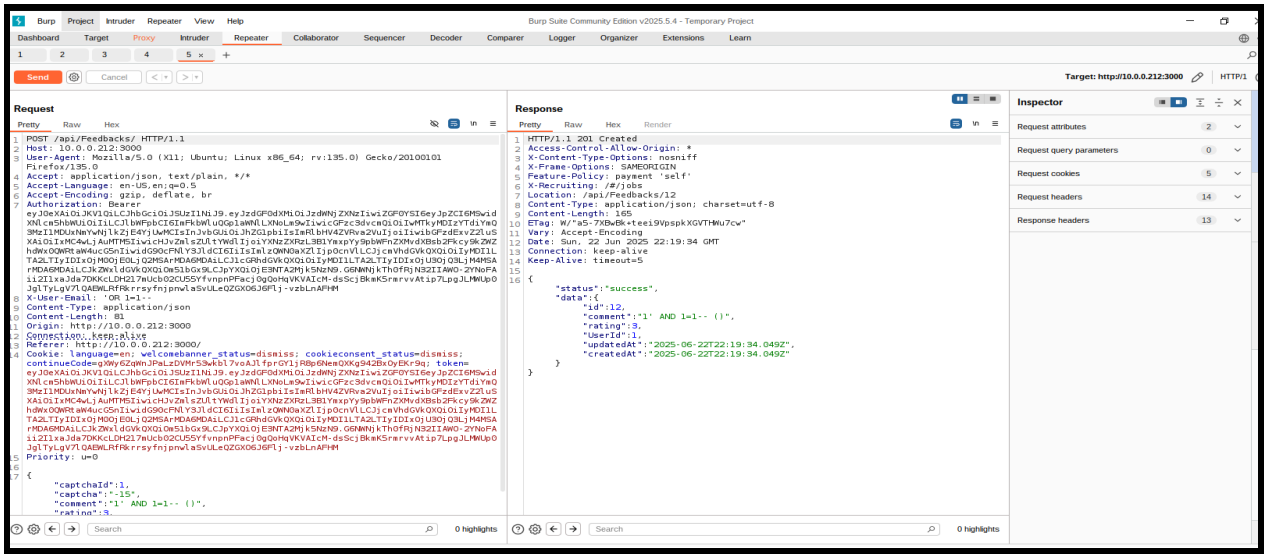


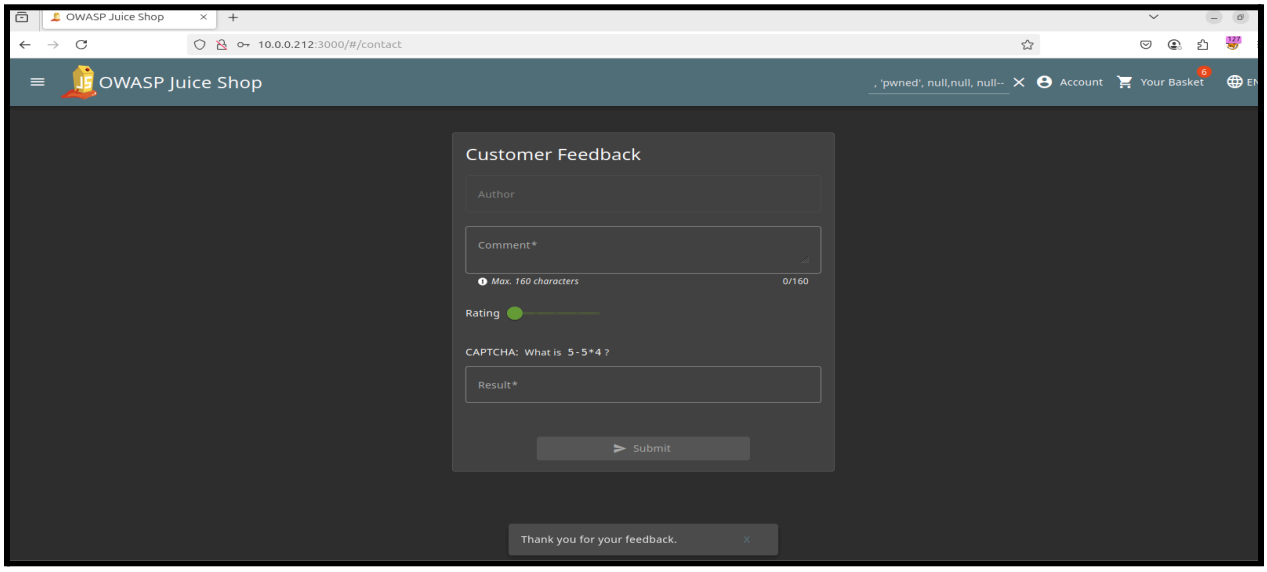




- Error-Based SQL Injection Attempt in Feedback API **Payload Used:** ‘1 AND 1=1-- () **Injection Point:** Comment field in Feedback form **Request Method:** POST to /api/Feedbacks/ **Observation:** Payload accepted (HTTP 201), echoed back without triggering SQL errors or altering behaviour.



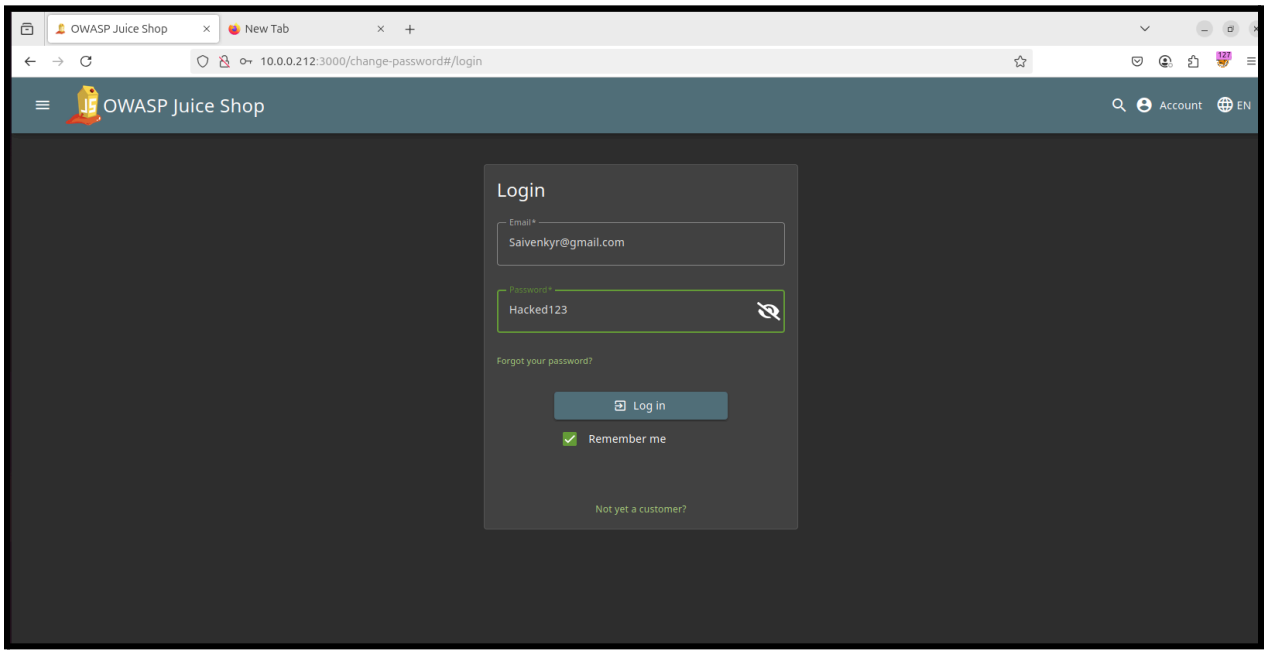


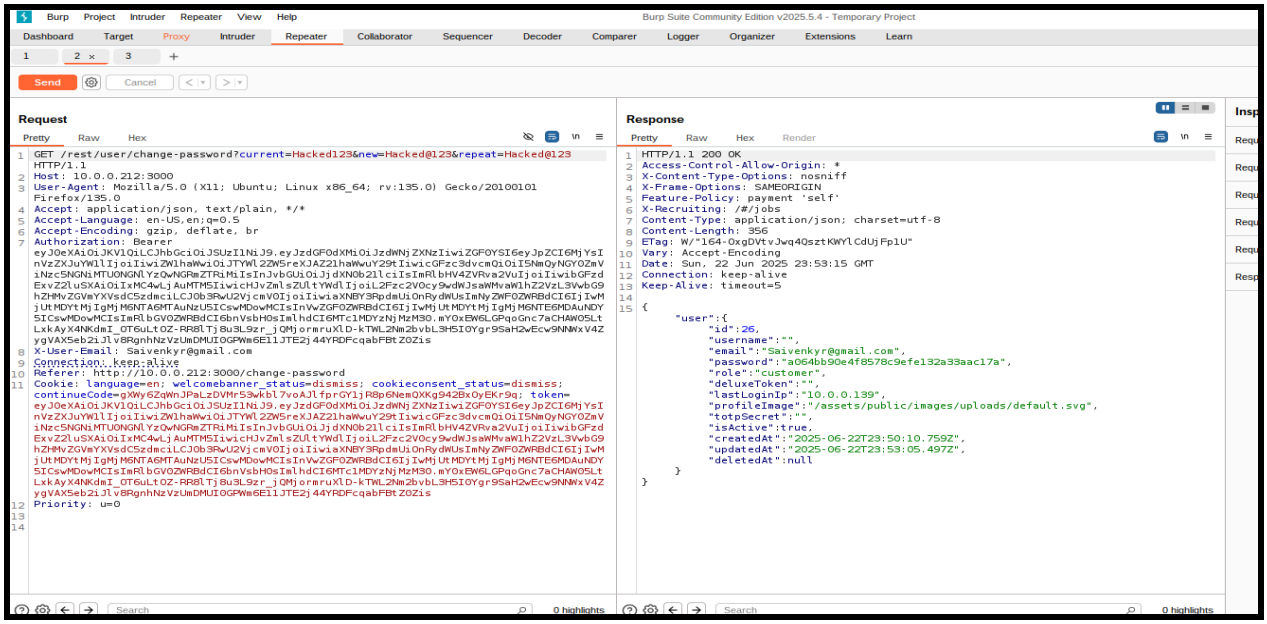


- **Remediation:**
    1. Use parameterised queries or prepared statements across all database interactions to prevent injection of malicious SQL code. Avoid dynamic query construction with direct user input.
    2. Implement input validation and least privilege principles—ensure only expected input is accepted, and the database account used by the application has limited permissions.

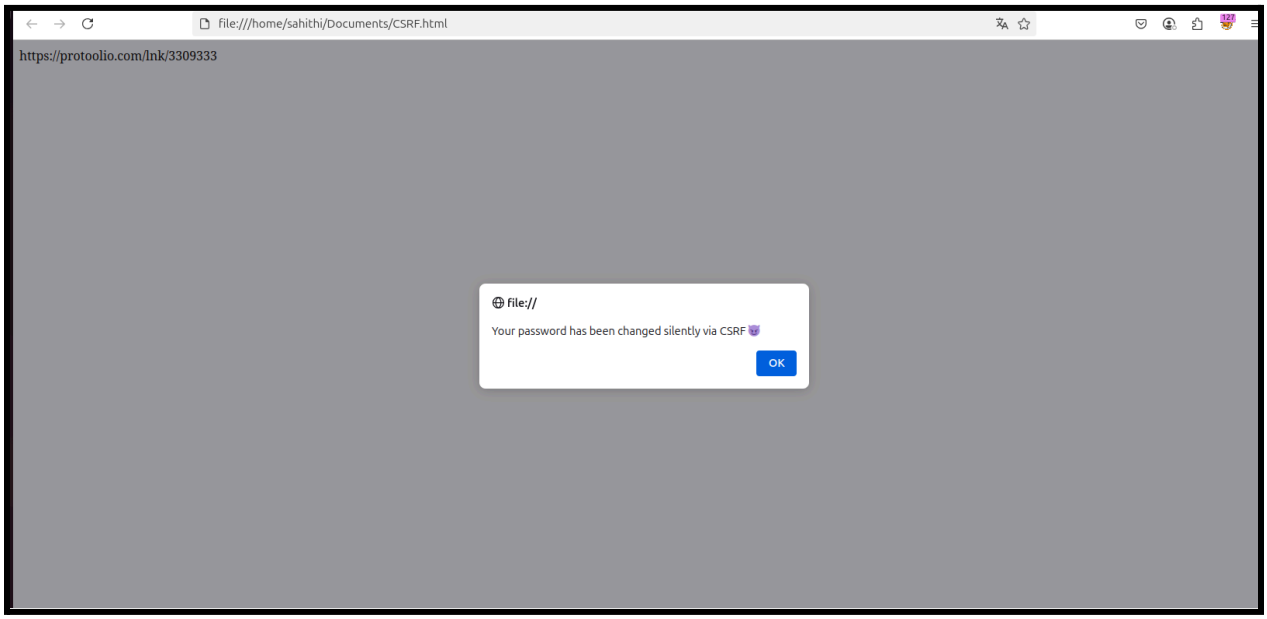
4.5. CSRF

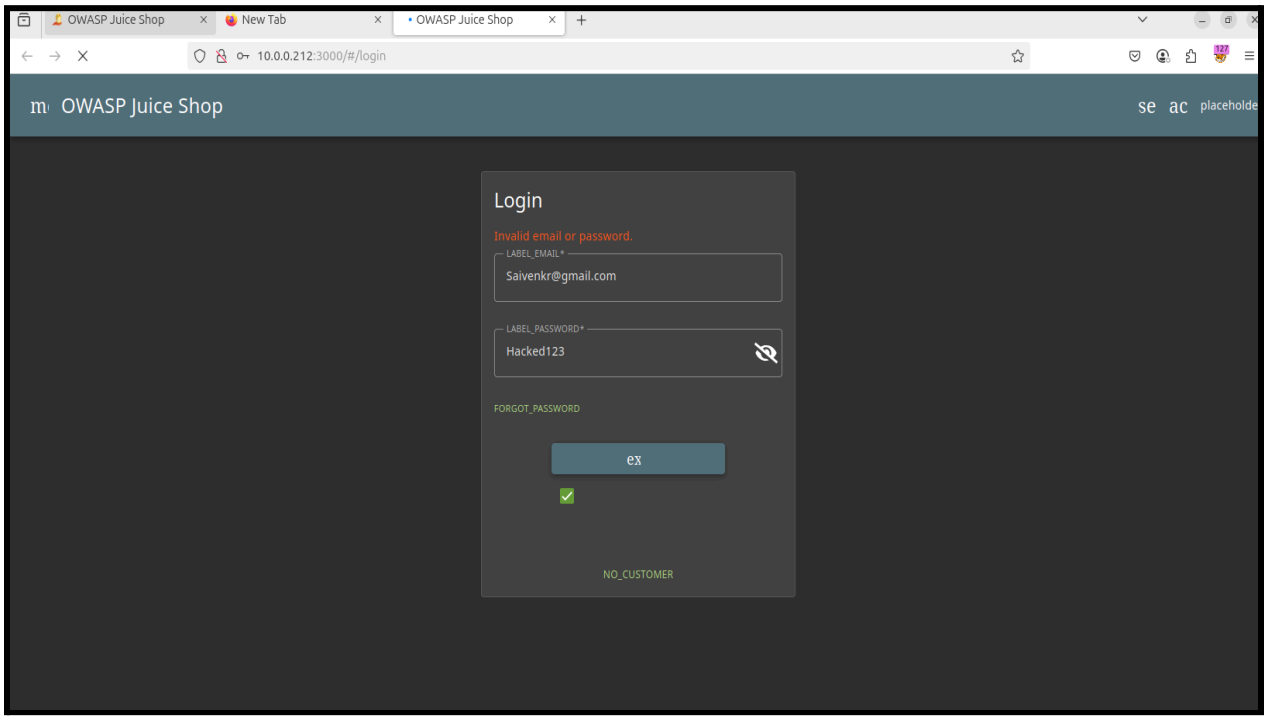
The password change endpoint in Juice Shop is vulnerable to CSRF, allowing unauthorised modification of a user's credentials without their knowledge. A crafted local HTML file (CSRF.html) silently triggers the exploit when opened during an active session, confirming a lack of CSRF protection on sensitive functionality.





If I clicked on this path, it is showing me “file:/home/sahithi/documents/CSRF.html”

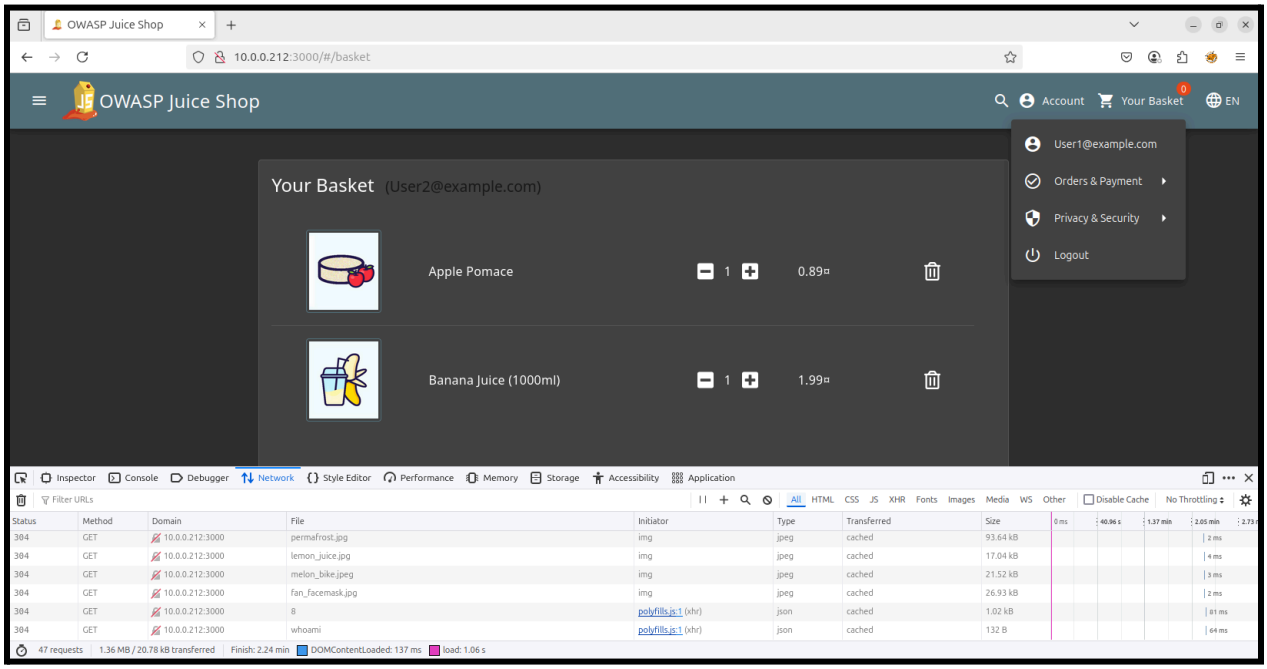


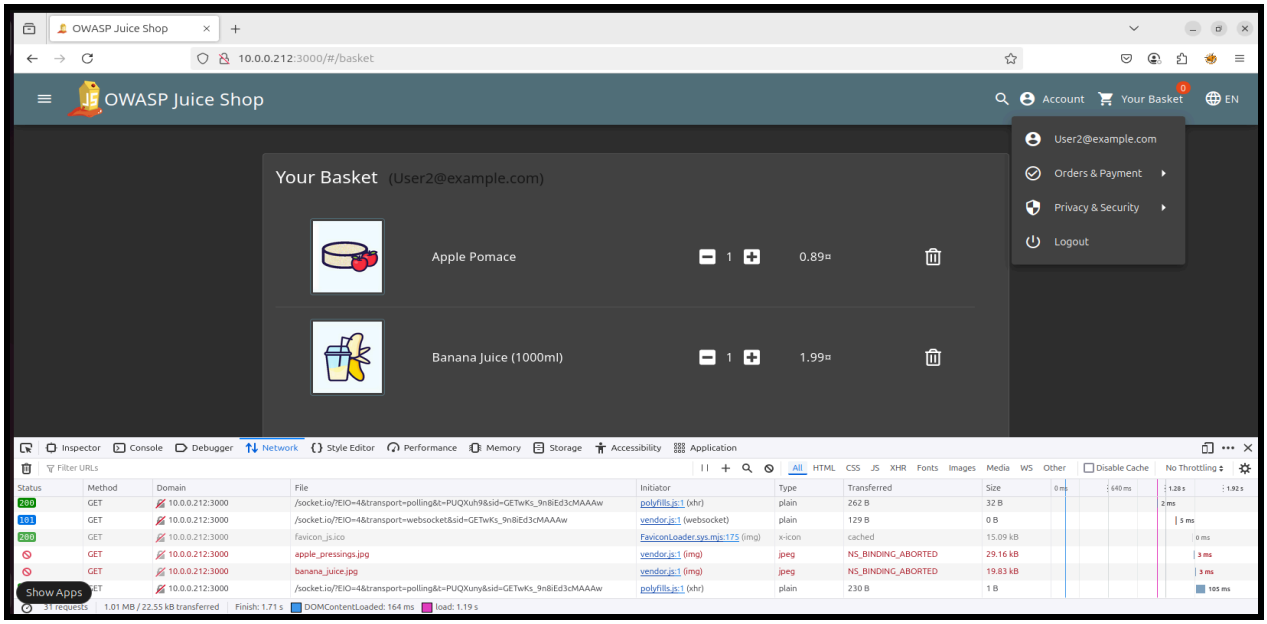


- **Remediation:**
    1. Implement anti-CSRF tokens in all state-changing requests (e.g., password updates, profile changes), and ensure tokens are unique per session and validated on the server side.
    2. Enforce SameSite cookie attributes (e.g., `SameSite=Strict` or `Lax`) and require user re-authentication for sensitive actions to reduce the chance of unauthorised cross-origin requests.

4.6. Broken Access Control

During the security testing of Juice Shop, I created two user accounts to evaluate access control. While logged in as User 1, I was able to access User 2's basket data by manipulating the `basketId` in the API request. The application responded with a `200 OK`, confirming unauthorised access to another user's private information. This reveals a Broken Access Control vulnerability, specifically an Insecure Direct Object Reference (IDOR). It highlights the absence of proper backend validation tied to resource ownership.

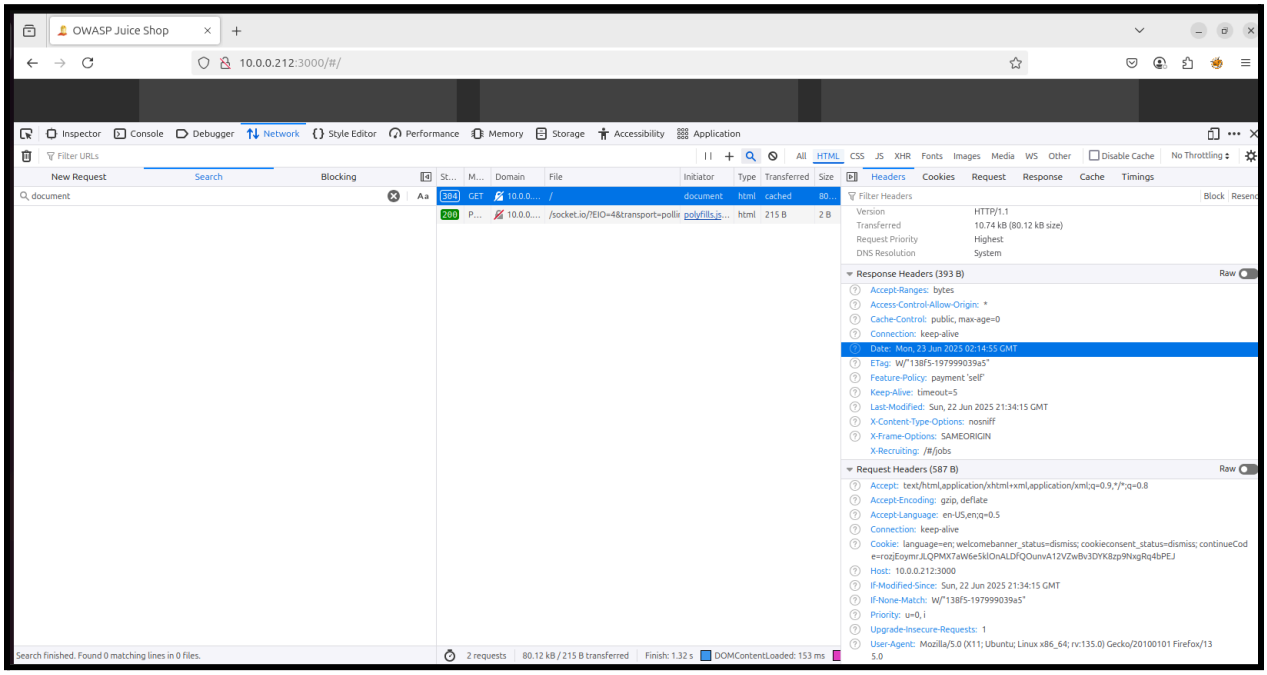




- **Remediation:**
    1. Implement robust access control checks on the server side—every request should be verified to ensure the authenticated user is authorised to access or modify the resource (e.g., Basket ID).
    2. Avoid relying on client-side identifiers alone—use session-bound tokens or role-based access enforcement to prevent unauthorised resource manipulation.

4.7. Security Misconfiguration

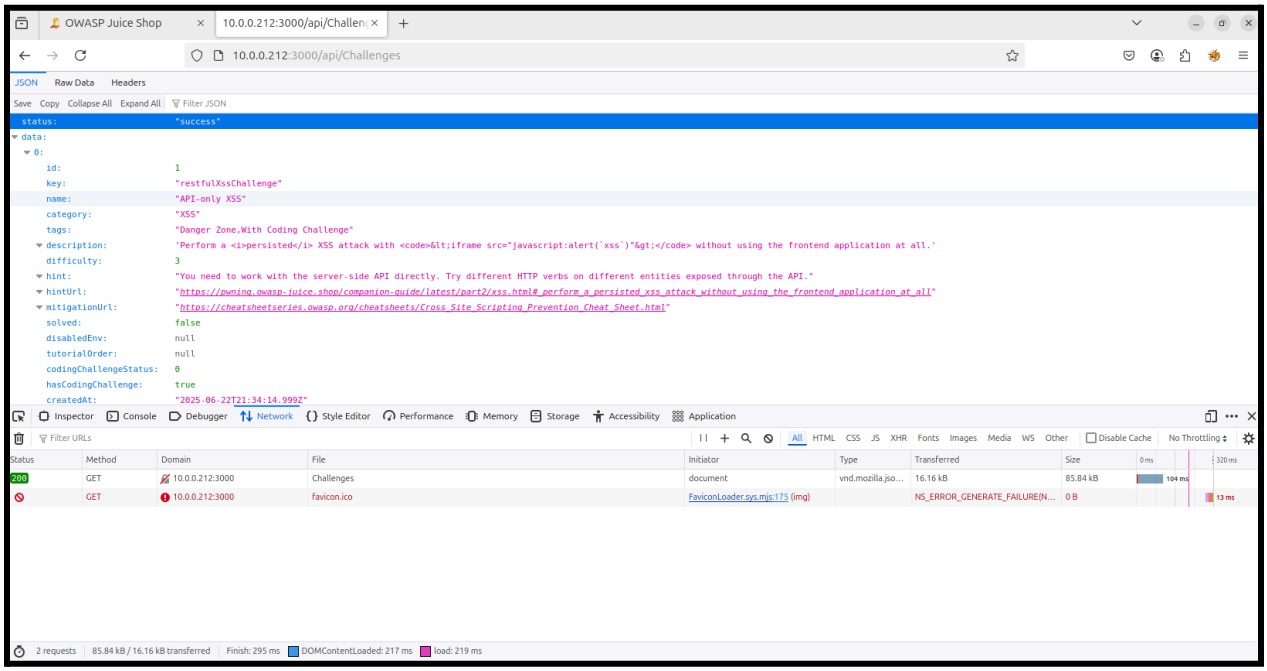
HTTP Headers: The application properly sets **X-Content-Type-Options:** nosniff and **X-Frame-Options**: SAMEORIGIN, offering basic protection. However, essential headers like Content-Security-Policy, Strict-Transport-Security, Referrer-Policy, and Permissions-Policy are missing. These headers defend against XSS, clickjacking, insecure transport, and data leakage. Their absence indicates a server-side security misconfiguration. Strengthening the header configuration is necessary to improve client-side security.



- **Remediation:** Configure the web server to include essential HTTP security headers such as **Content-Security-Policy**, **Strict-Transport-Security**, **X-Content-Type-Options,** and **Referrer-Policy**. These headers help mitigate XSS, clickjacking, and man-in-the-middle attacks

4.8. Unprotected API

The endpoint /api/Challenges is accessible without authentication, allowing public exposure of internal challenge data. This represents an **unprotected API** and a **security misconfiguration**. Sensitive metadata is returned with a 200 OK, violating access control principles.



- **Remediation:** Implement authentication and authorisation checks on all API endpoints. Ensure that sensitive endpoints (e.g., /api/Challenges) are not publicly accessible without valid user credentials, and follow the principle of least privilege for API access.

5. Appendix

S.no	Tool	Purpose
1	Whois	Gather domain and IP ownership details
2	nslookup	Perform DNS and reverse DNS lookup
3	Wapplazer	Identifying front-end and back-end technologies
4	Whatweb	Analysing HTTP responses and detecting tech stack
5	Nmap	Discovering open ports and services on the target system
6	Nikto	Enumerating web server misconfigurations and exposures
7	Burpsuite	Intercept test and manipulate web traffic and payloads

6. Key Observations

---

1. The application revealed multiple OWASP Top 10 vulnerabilities, including SQL Injection, Cross-Site Scripting (XSS), IDOR, CSRF, and Security Misconfigurations.
2. Lack of security headers and exposed API endpoints indicated potential for client-side exploitation and unauthorised access.
3. Technologies like jQuery 2.2.4 and unprotected directories suggested outdated components and misconfigured web servers.
4. Authentication and session management were implemented securely, and no token reuse or session fixation was observed.
5. Enumeration using Nmap, Nikto, and WhatWeb aided in mapping out the infrastructure and understanding the attack surface.

7. Conclusion

The assessment confirmed that while the application is functionally sound, it contains critical security flaws that could be exploited by malicious users. Remediation of identified issues—especially those involving input validation, access control, and secure server configurations—is essential. Implementing secure development practices and regular vulnerability assessments will greatly reduce the attack surface and improve overall resilience.