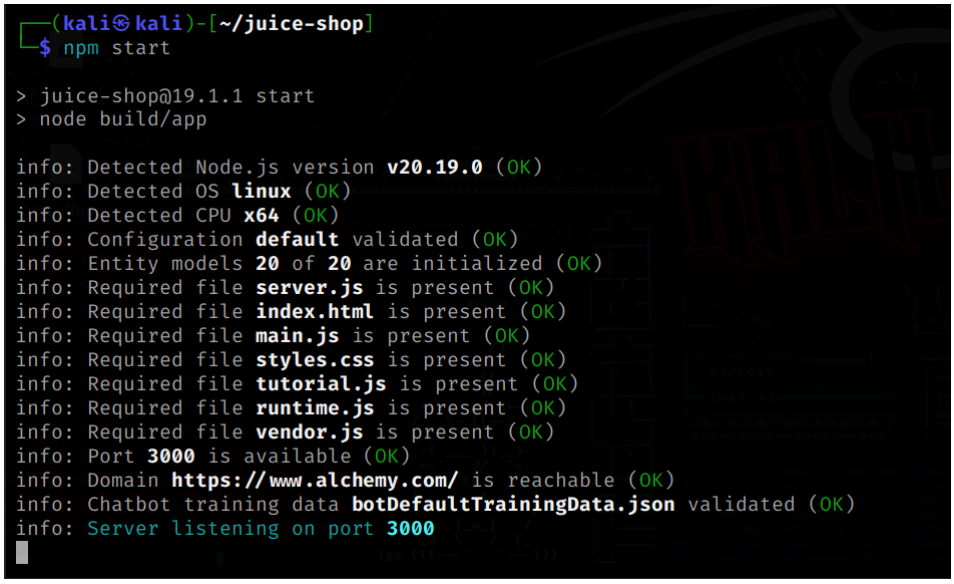
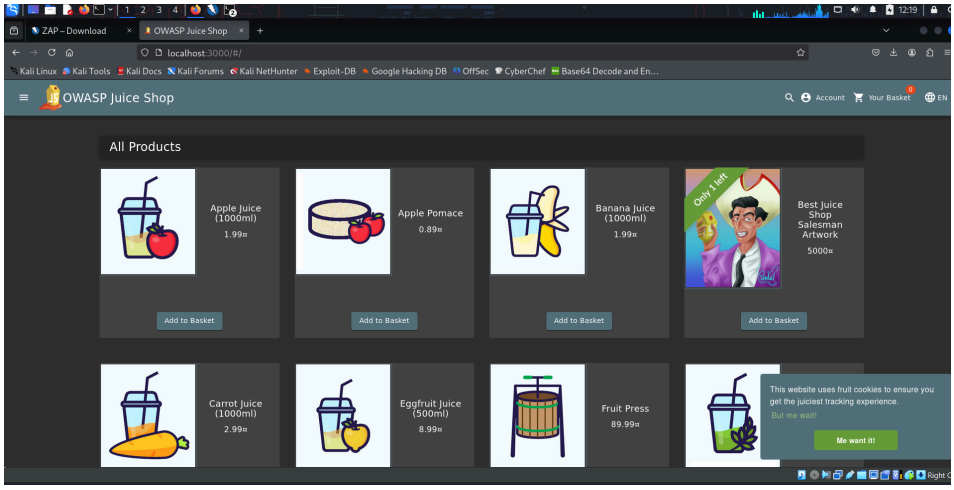
Week 1: Security Assessment

1. Understand the Application



Web application was running locally

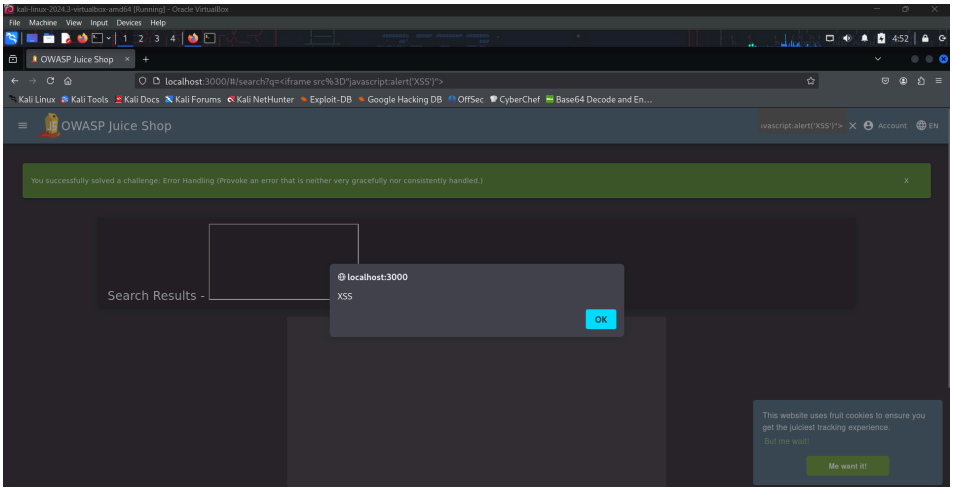


Juice Shop *vulnerable web applications*

2. Perform Basic Vulnerability Assessment

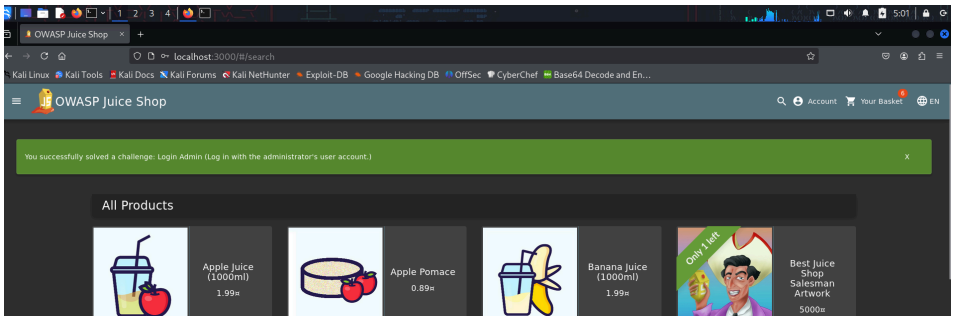
Vulnerabilities Found

- Reflected Cross-Site Scripting (XSS)
    - Location: Search Bar
    - Evidence: Executed <iframe src="javascript:alert('XSS')"> to trigger a popup.



Note on Payload: *The standard* `<script>` *payload was ineffective because the application inserts user input via* `innerHTML`*, which browsers do not execute. I successfully exploited the vulnerability using an* `<iframe>` *payload (*`<iframe src="javascript:alert('XSS')">`*), which leverages the* `src` *attribute to force JavaScript execution*

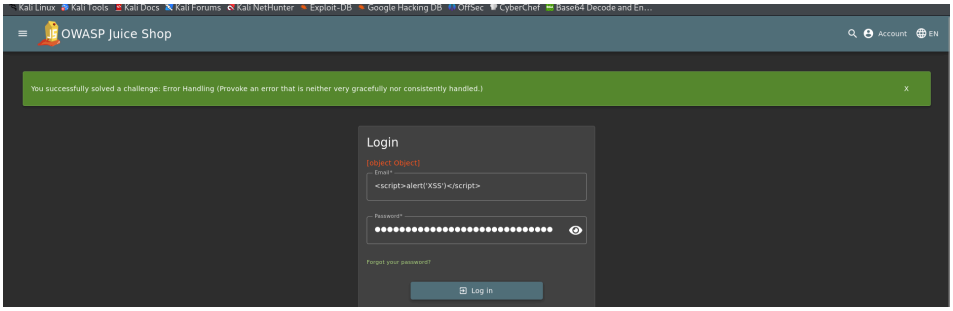
- SQL Injection (Authentication Bypass)
    - Location: Login Page (Email Field)
    - Evidence: Logged in as Admin using payload ' OR 1=1 -- .



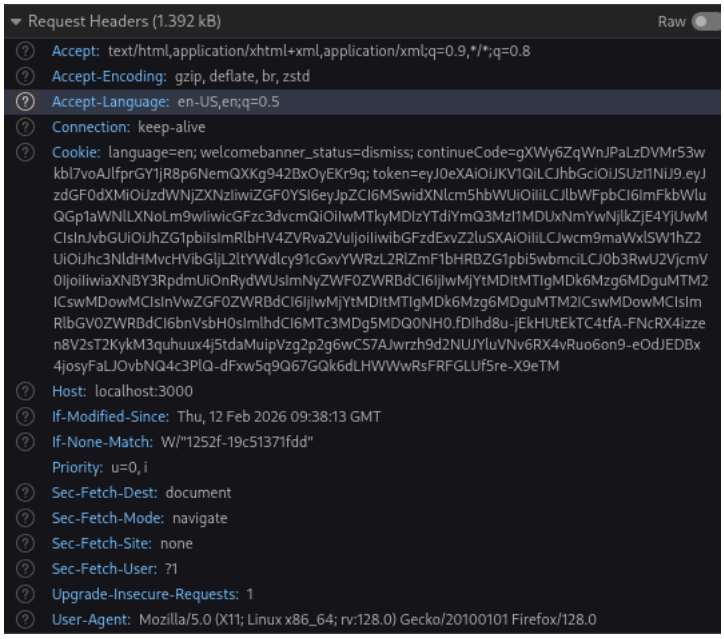
- Weak Password Hashing
    - Location: User Database
    - Evidence: Passwords are stored using MD5 (identified by 32-char hex strings). This algorithm is obsolete and vulnerable to cracking.

```
sqlite> SELECT email, password FROM Users;
admin@juice-sh.op|0192023a7bbd73250516f069df18b500
jim@juice-sh.op|e541ca7ecf72b8d1286474fc613e5e45
bender@juice-sh.op|0c36e517e3fa95aabf1bbffc6744a4ef
bjoern.kimminich@gmail.com|6edd9d726cbdc873c539e41ae8757b8c
ciso@juice-sh.op|861917d5fa5f1172f931dc700d81a8fb
support@juice-sh.op|3869433d74e3d0c86fd25562f836bc82
morty@juice-sh.op|f2f933d0bb0ba057bc8e33b8ebd6d9e8
mc.safesearch@juice-sh.op|b03f4b0ba8b458fa0acdc02cdb953bc8
J12934@juice-sh.op|3c2abc04e4a6ea8f1327d0aae3714b7d
wurstbrot@juice-sh.op|9ad5b0492bbe528583e128d2a8941de4
amy@juice-sh.op|030f05e45e30710c3ad3c32f00de0473
bjoern@juice-sh.op|7f311911af16fa8f418dd1a3051d6810
bjoern@owasp.org|9283f1b2e9669749081963be0462e466
chris.pike@juice-sh.op|10a783b9ed19ea1c67c3a27699f0095b
accountant@juice-sh.op|963e10f92a70b4b463220cb4c5d636dc
uvogin@juice-sh.op|05f92148b4b60f7dacd04cceebb8f1af
demo|fe01ce2a7fbac8fafaed7c982a04e229
john@juice-sh.op|00479e957b6b42c459ee5746478e4d45
emma@juice-sh.op|402f1c4a75e316afec5a6ea63147f739
stan@juice-sh.op|e9048a3f43dd5e094ef733f3bd88ea64
ethereum@juice-sh.op|2c17c6393771ee3048ae34d6b380c5ec
testing@juice-sh.op|b616a64605a07941fbd31868aea3b54b
sqlite>
```

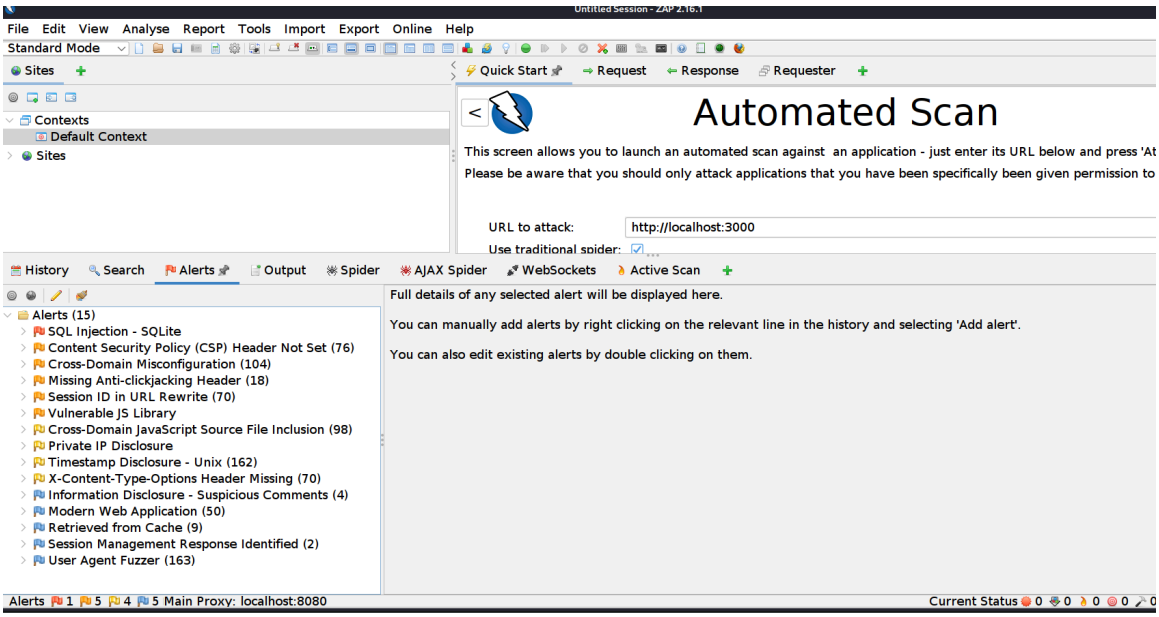
- Improper Error Handling
    - Location: Login Page
    - Evidence: Application returns raw [object Object] errors instead of user-friendly messages.



- Security Misconfiguration
    - Location: HTTP Response Headers (Global/All Pages)
    - Evidence: The application response is missing critical security headers such as `X-Frame-Options`, `X-Content-Type-Options`, and `Content-Security-Policy`



OWASP ZAP: Automated scanner for web app vulnerabilities



Areas of Improvement

- Input Sanitization: Implement strict validation using libraries like `validator` to prevent XSS and SQL Injection attacks .
- Secure Authentication: Upgrade password storage from weak MD5 to strong hashing algorithms like `bcrypt` and implement token-based authentication .
- Server Hardening: Configure secure HTTP headers using `helmet.js` and implement generic error messages to prevent information leakage .