🛡️ Web Application Security Report

Internship Task 1 – Future Interns | Track Code: CS

Intern Name: Jaikanth R

Platform: OWASP Juice Shop

Tools Used: Kali Linux, Node.js, Burp Suite, Browser, Git, NPM

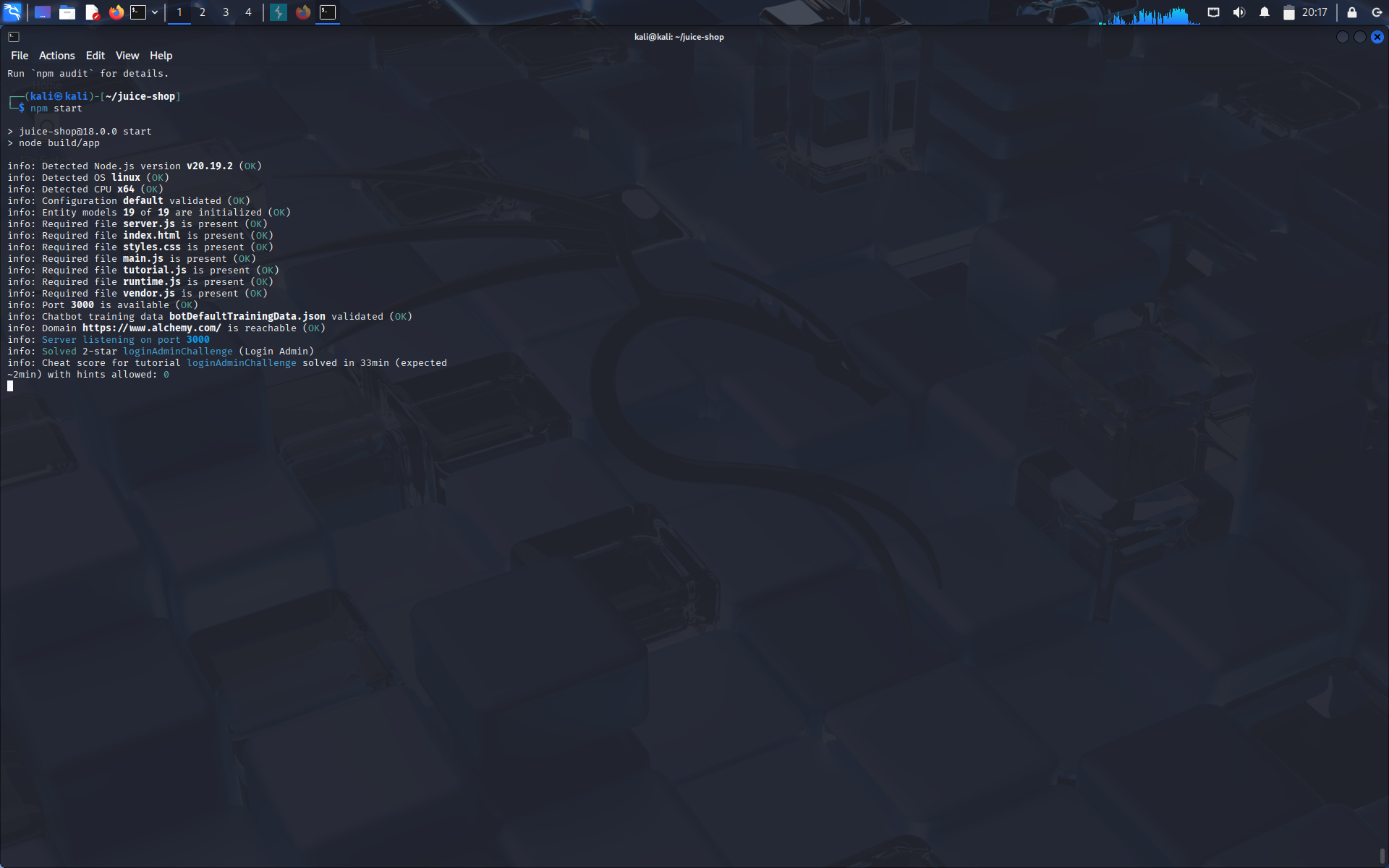
# 1. Environment Setup

To complete Task 1, the vulnerable application OWASP Juice Shop was deployed locally on Kali Linux (Virtual Machine) using Node.js and NPM.

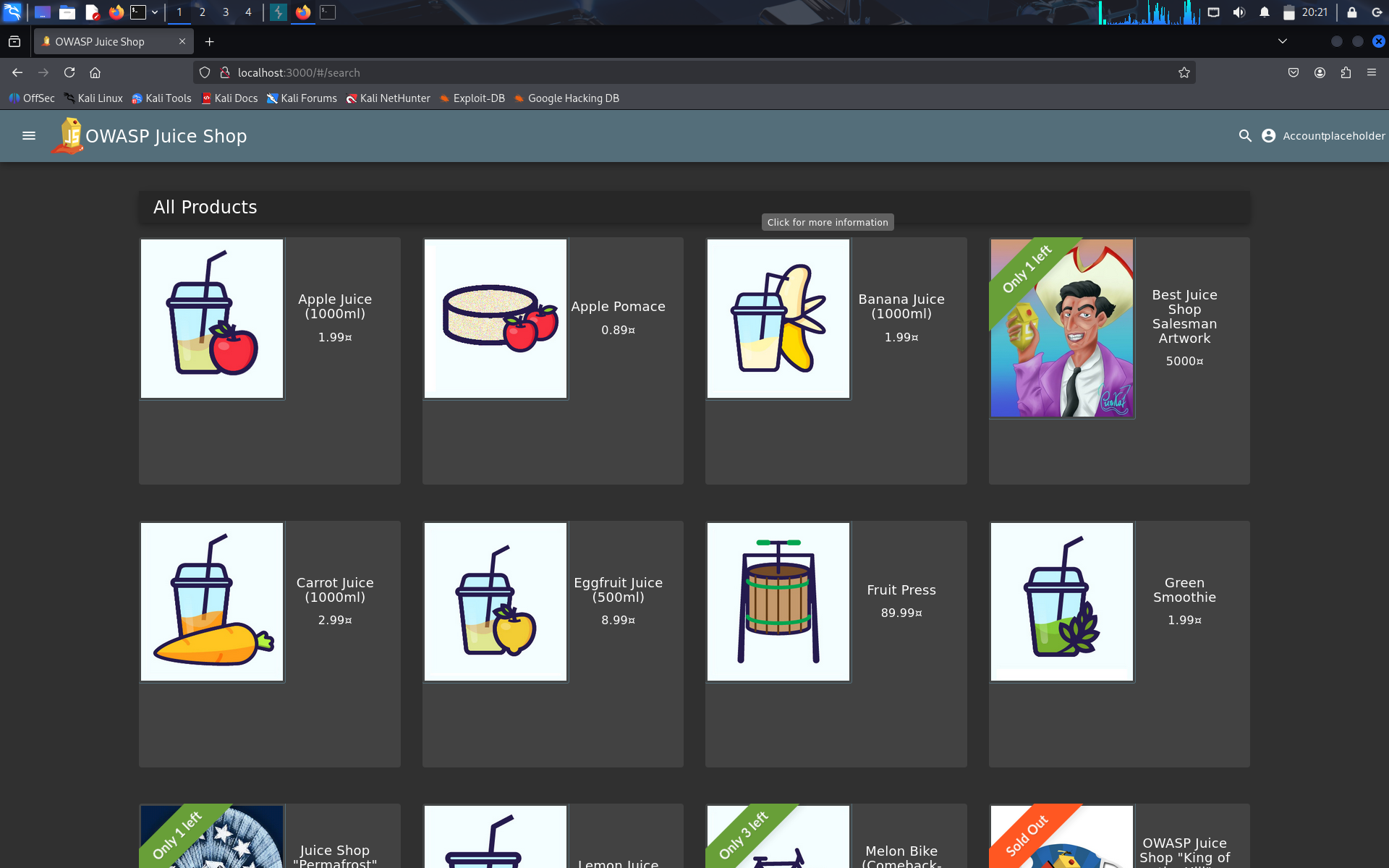
Setup Steps:

* - Installed nodejs, npm, and git  
  - Cloned the official Juice Shop GitHub repository  
  - Launched the app using `npm start`

Screenshot: Juice Shop Setup Running via Terminal



Screenshot: Juice Shop in Browser (http://localhost:3000)



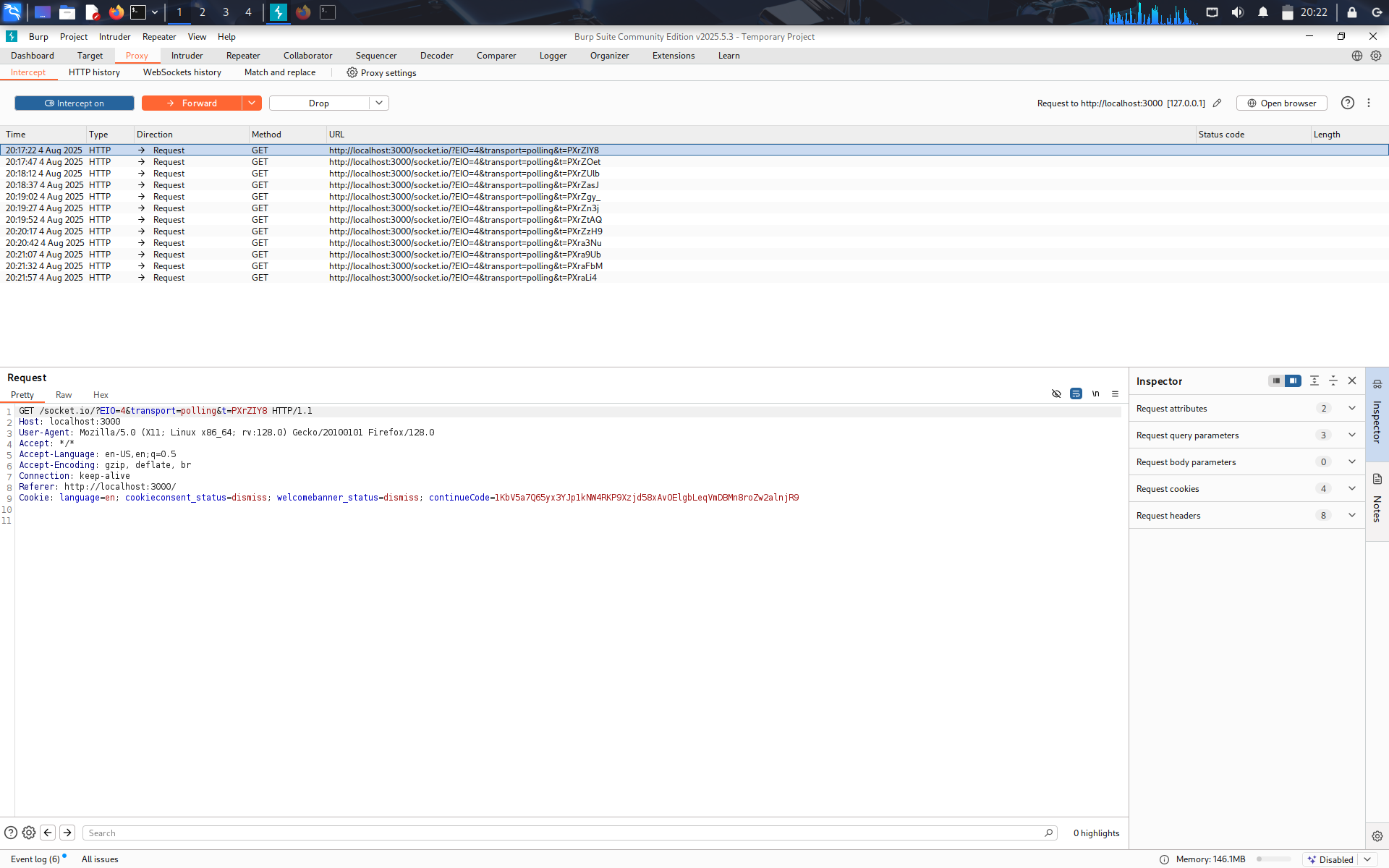
# 2. Tool Configuration – Burp Suite

For request interception and payload manipulation, Burp Suite Community Edition was used.

Actions Performed:

* - Configured Firefox to route traffic through 127.0.0.1:8080  
  - Enabled network.proxy.allow\_hijacking\_localhost in Firefox to proxy localhost  
  - Intercepted live HTTP requests to analyze login, search, and feedback behavior

Screenshot: Burp Suite with Intercept Proxy ON



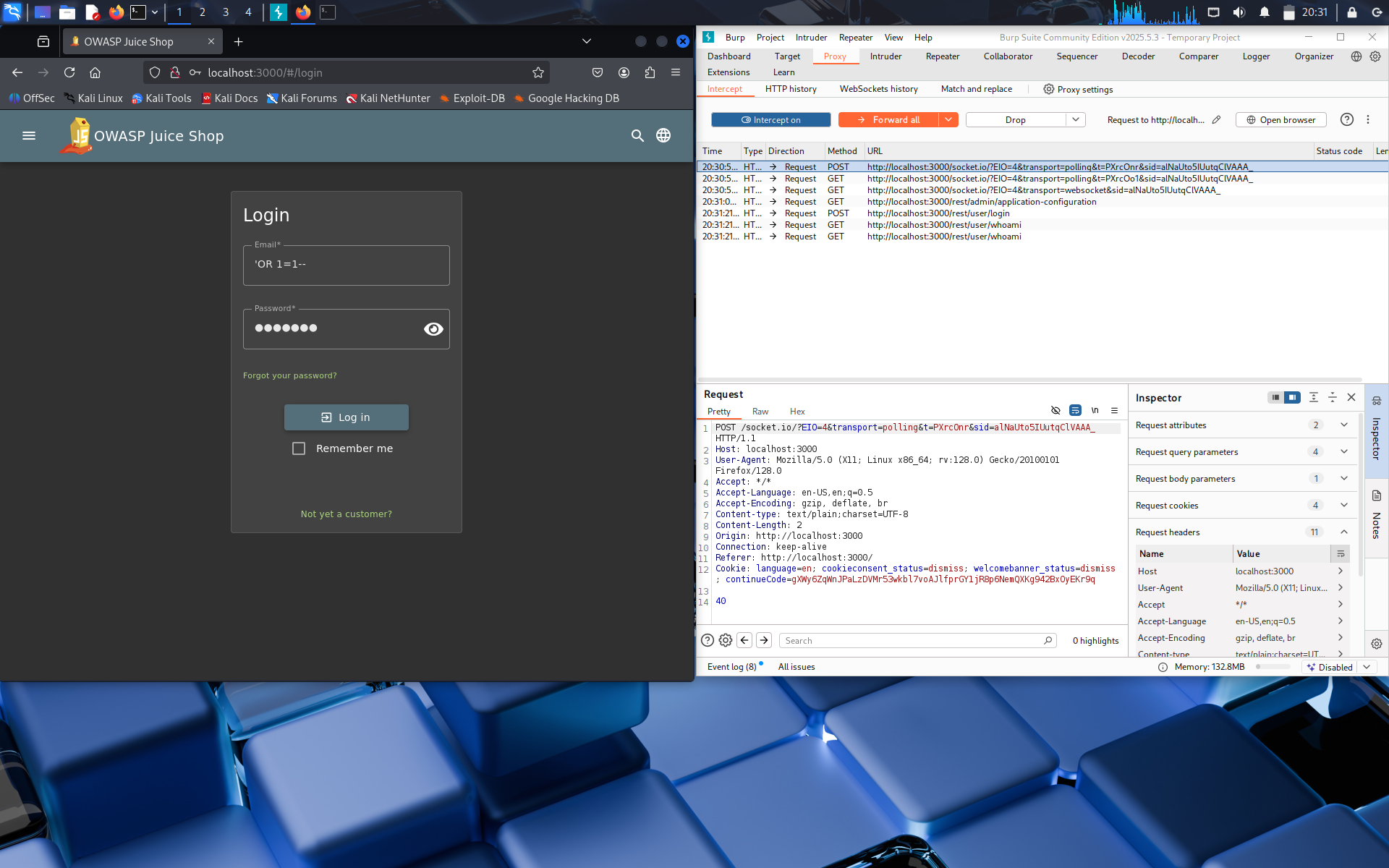
# 3. Vulnerability 1: SQL Injection (Authentication Bypass)

SQL Injection vulnerability was found in the login page, allowing bypass of authentication via manipulated input.

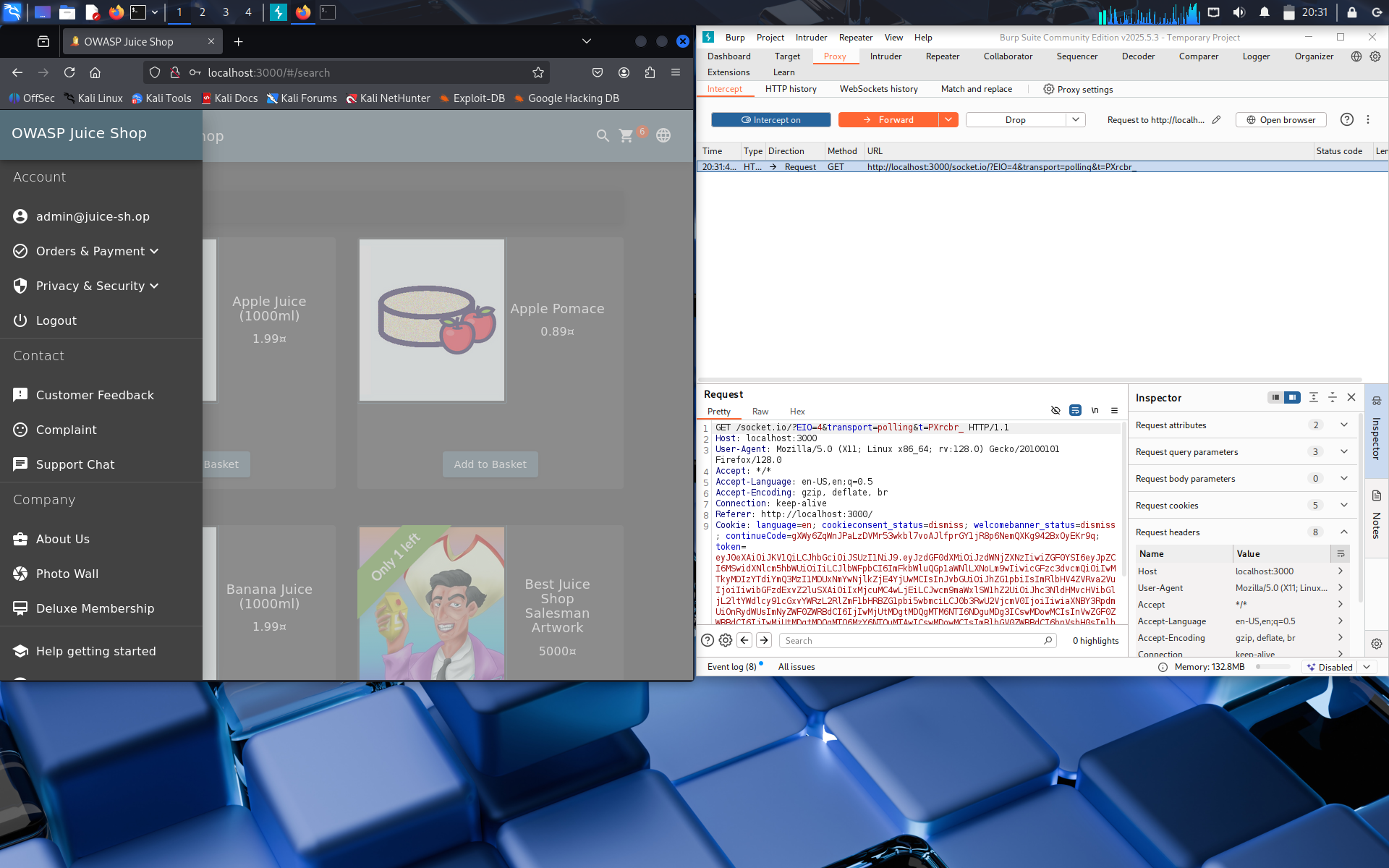
Steps to Reproduce:

1. 1. Navigate to /login  
   2. Intercept POST request in Burp  
   3. Modify login credentials:  
    Email: ' OR 1=1--  
    Password: anything  
   4. Forward the request → Logged in successfully

Screenshot: SQLi Injection via Burp Repeater



Screenshot: Login Success After Injection



Impact: An unauthenticated attacker can gain full access to user or admin areas by bypassing credential validation.

Mitigation:

* - Use parameterized queries (prepared statements)  
  - Avoid dynamic SQL string building  
  - Implement WAF and input validation

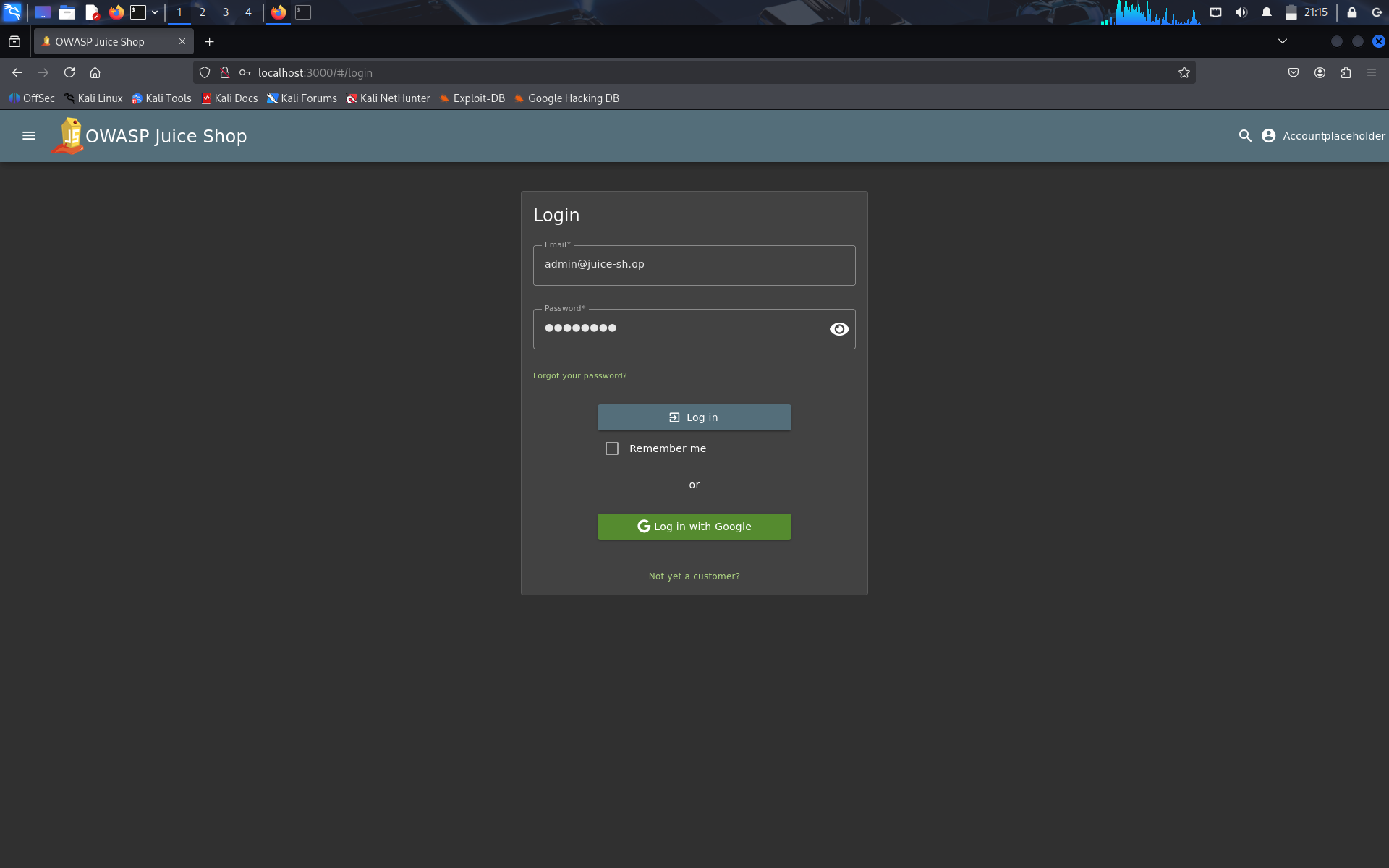
# 4. Vulnerability 2: Weak Authentication (Default Credentials)

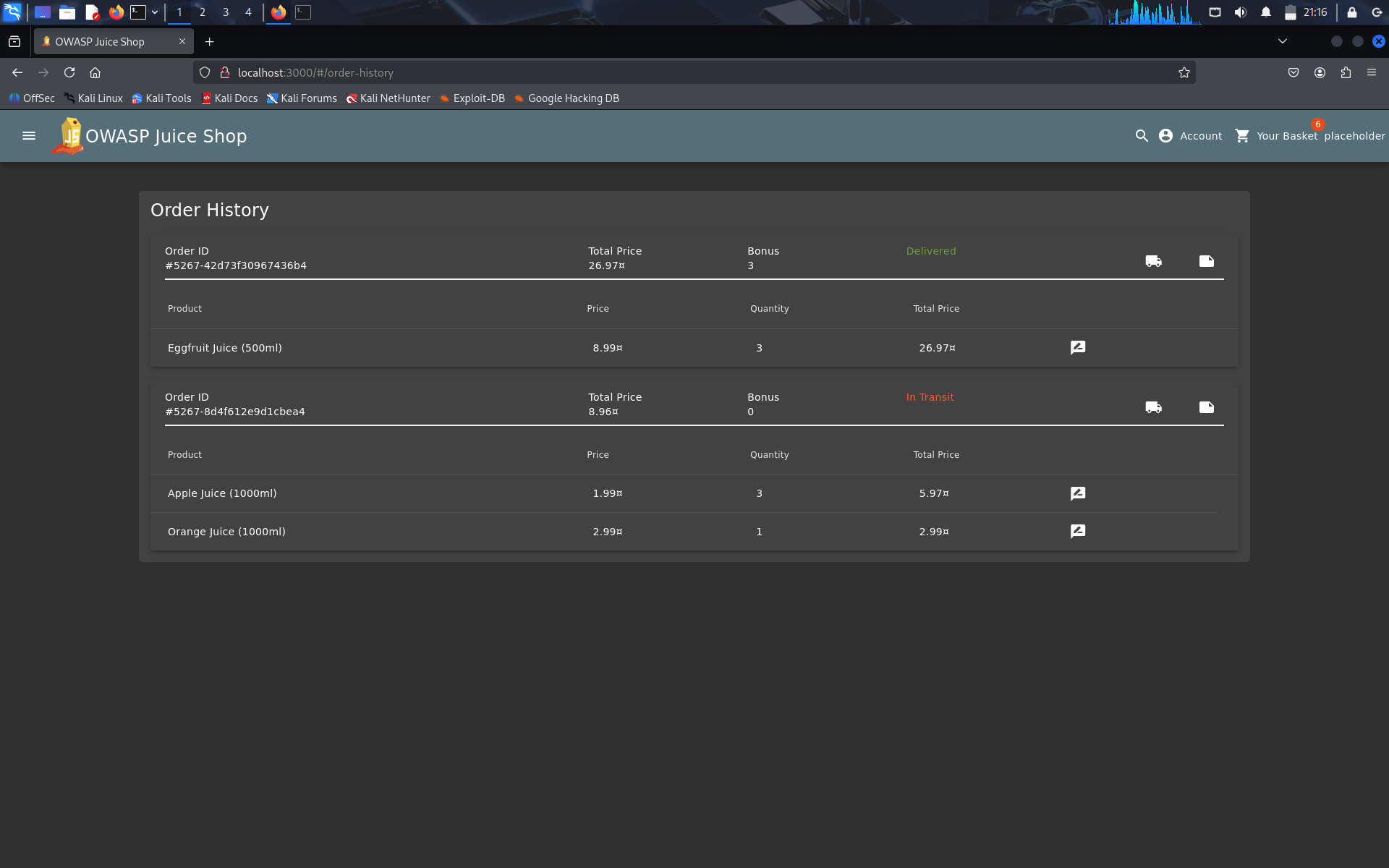
The system accepted weak, known credentials (admin@juice-sh.op / admin123) for administrator login.

Steps to Reproduce:

1. 1. Navigate to /login  
   2. Enter credentials:  
    Email: admin@juice-sh.op  
    Password: admin123  
   3. Access granted to admin-level features

Screenshot: Successful Login Using Default Credentials





Impact: Any attacker aware of default credentials could access privileged functions without effort.

Mitigation:

* - Disable default credentials before deployment  
  - Enforce password strength policy  
  - Require 2FA for sensitive accounts

# 5. Vulnerability 3: Cross-Site Scripting (XSS – Attempted)

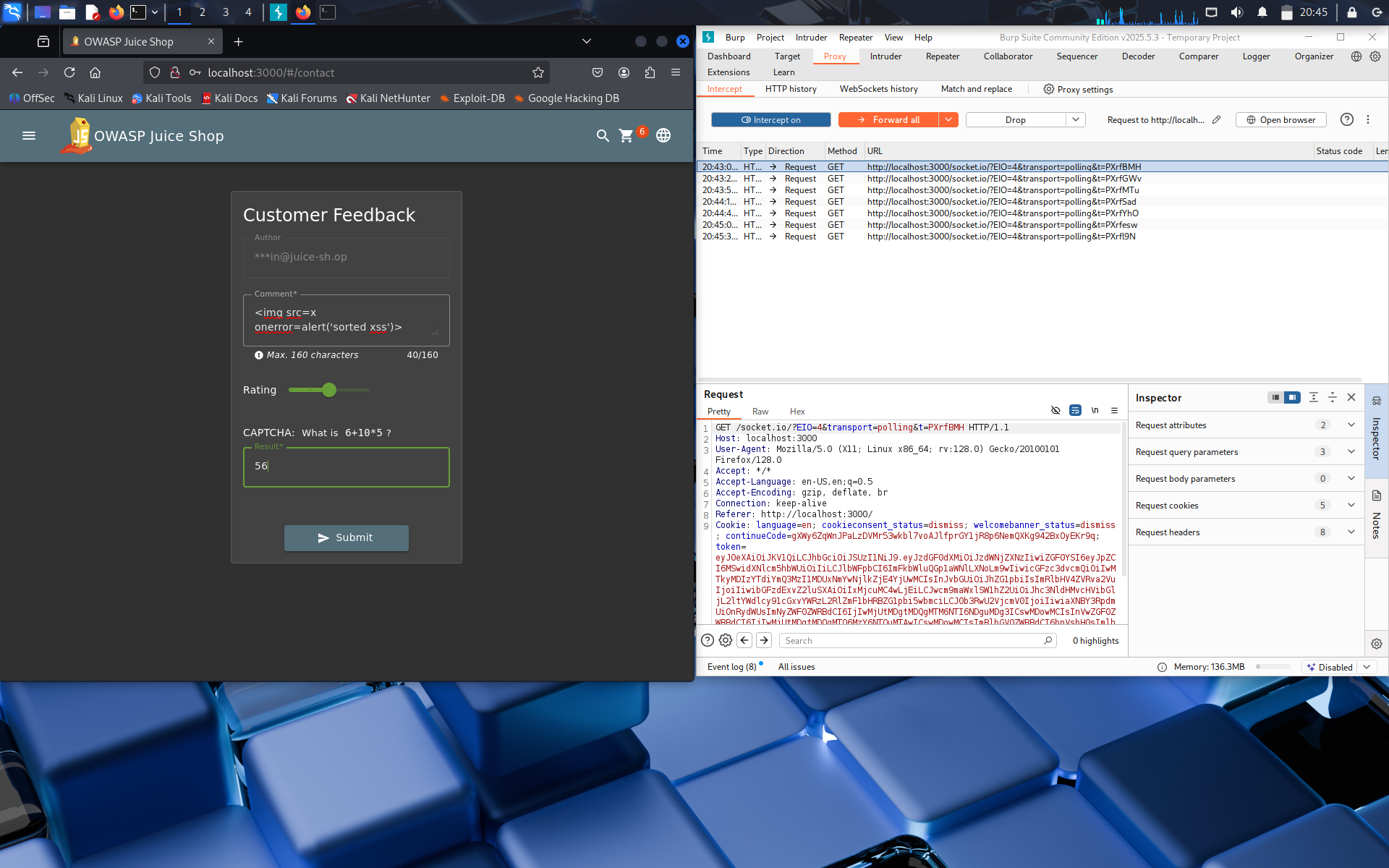
Tested multiple locations for stored XSS, including the Feedback form and Product Review section.

Payloads Used:

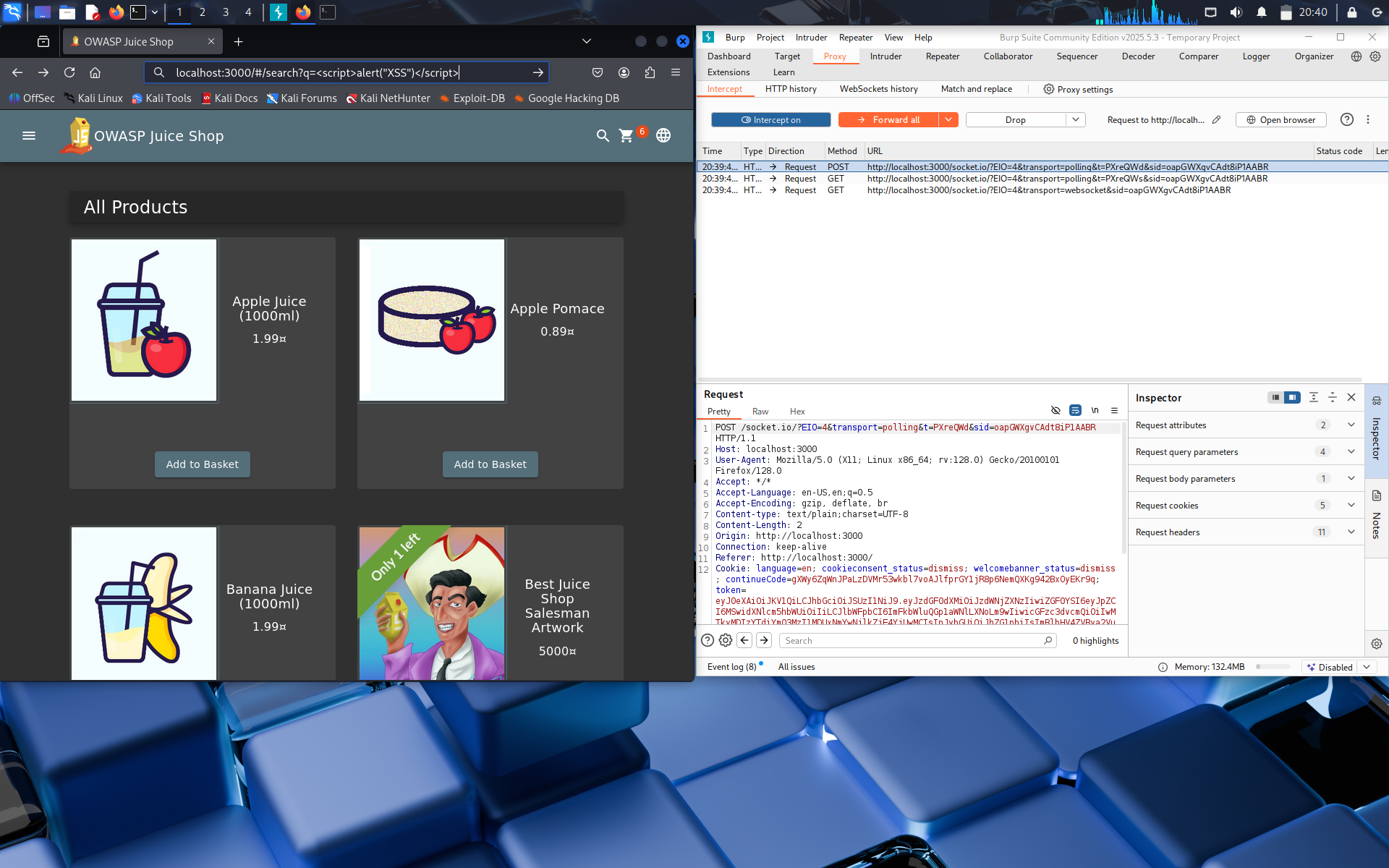
<script>alert('XSS')</script>  
<img src=x onerror=alert('XSS')>  
<svg/onload=alert(1)>  
<iframe srcdoc="<script>alert(1)</script>"></iframe>

Locations Tested: /contact (Feedback), /product/1 (Review)

Screenshot: Payload Submitted in Feedback



Screenshot: Payload Displayed in Product Review



Result: Payloads were stored and rendered, but escaped safely. No scripts executed.

Conclusion: The app reflects input but sanitizes it properly using output encoding.

Mitigation:

* - Contextual encoding of HTML characters  
  - Use of frontend sanitization frameworks  
  - Recommend adding Content Security Policy (CSP)

# 6. Summary Table of Findings

| # | Vulnerability | Type | Status | Risk Level |  
|---|------------------------------|----------------------|------------------------|-------------|  
| 1 | SQL Injection | Authentication Bypass | Confirmed ✅ | High |  
| 2 | Weak Admin Credentials | Default Passwords | Confirmed ✅ | High |  
| 3 | Stored XSS (Sanitized) | Input Reflection | Attempted, Escaped ✅ | Medium |

# 7. Tools & Technologies Used

* - Node.js / NPM – Running Juice Shop  
  - Burp Suite – Intercepting HTTP requests  
  - Browser (Firefox) – Manual testing  
  - Kali Linux – Testing environment  
  - GitHub – Documentation

# 8. Conclusion

This report demonstrates:  
- Exploitation of real-world vulnerabilities (SQL Injection, Authentication Flaws)  
- Secure development understanding (e.g., XSS mitigation)  
- Use of professional testing tools like Burp Suite  
- Capability to document and explain technical findings clearly

This task helped enhance my understanding of:  
- Web application vulnerabilities  
- Manual and tool-based testing  
- Writing structured security reports