Capstone Project: Full VAPT Engagement

Target: OWASP Juice Shop (10.49.177.96)
Author: Gyanesh Chand
Date: 26-02-2026
Tools Used: Kali Linux, Burp Suite

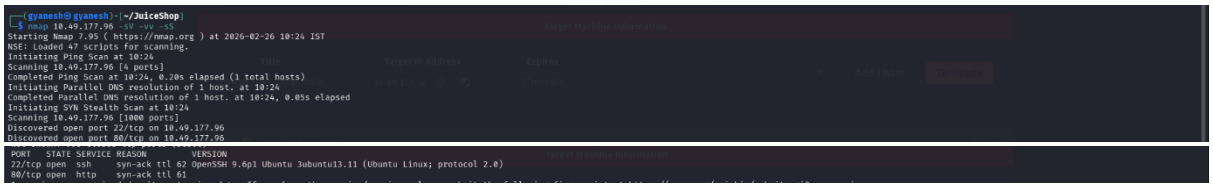
1. Executive Summary

A full Vulnerability Assessment and Penetration Testing (VAPT) engagement was conducted against the OWASP Juice Shop application hosted on the TryHackMe platform. The assessment followed the PTES (Penetration Testing Execution Standard) methodology including reconnaissance, enumeration, exploitation, and validation.

2. Reconnaissance & Enumeration

Command Used:

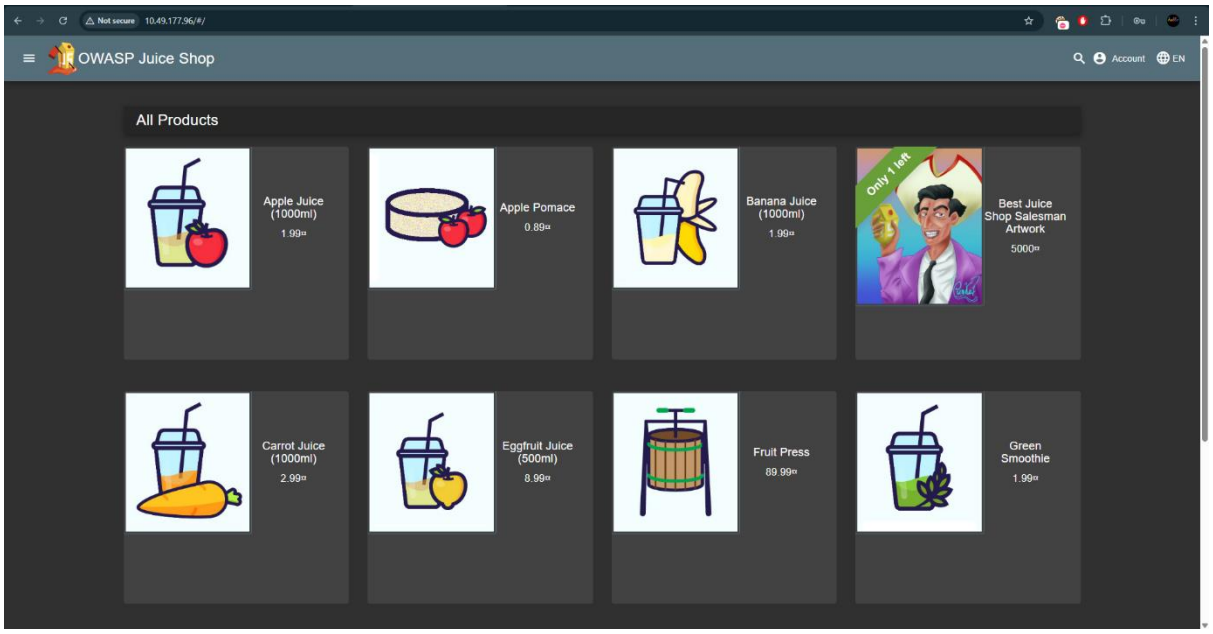
nmap -sV -sC 10.49.177.96



Findings:

- Port 22 → OpenSSH 9.6p1 Ubuntu
- Port 80 → HTTP service hosting Juice Shop

Attack surface confirmed as web-based application.



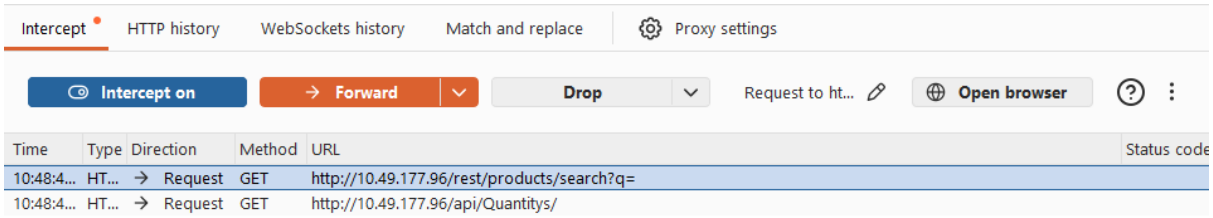
3. Web Application Analysis

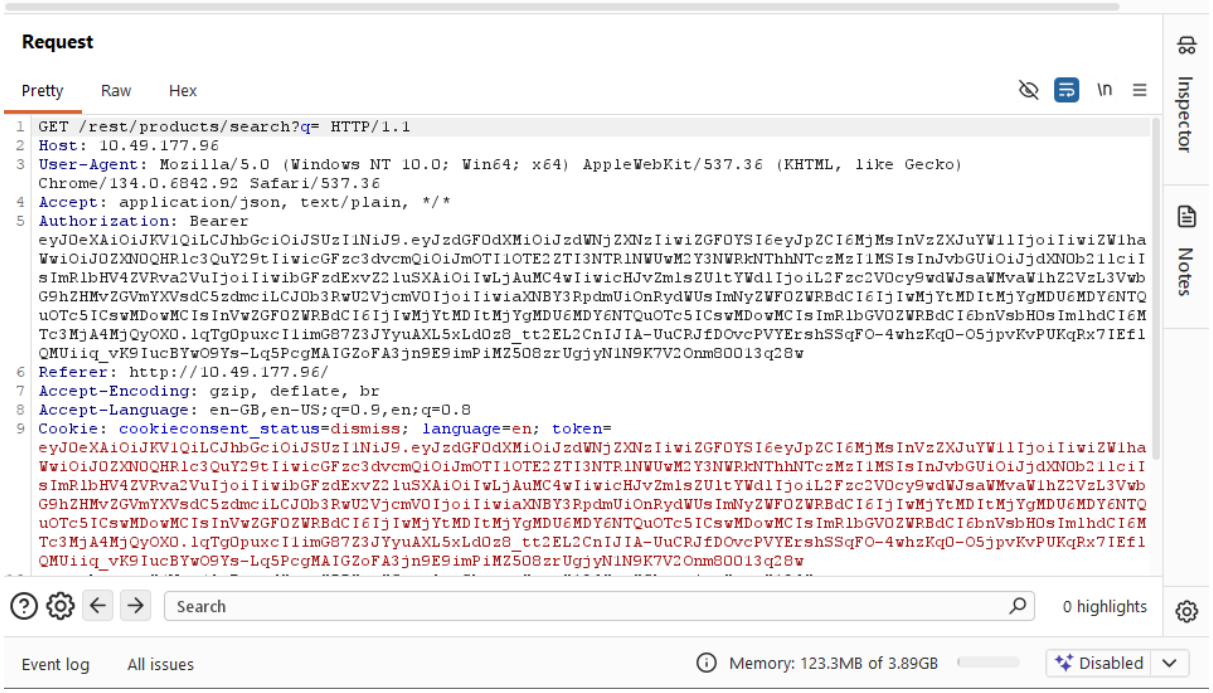
The application was accessed via HTTP.

Burp Suite was configured as an intercepting proxy to analyze requests.

Identified endpoint:

/rest/products/search?q=





This parameter appeared to reflect user input in the response.

Observation:

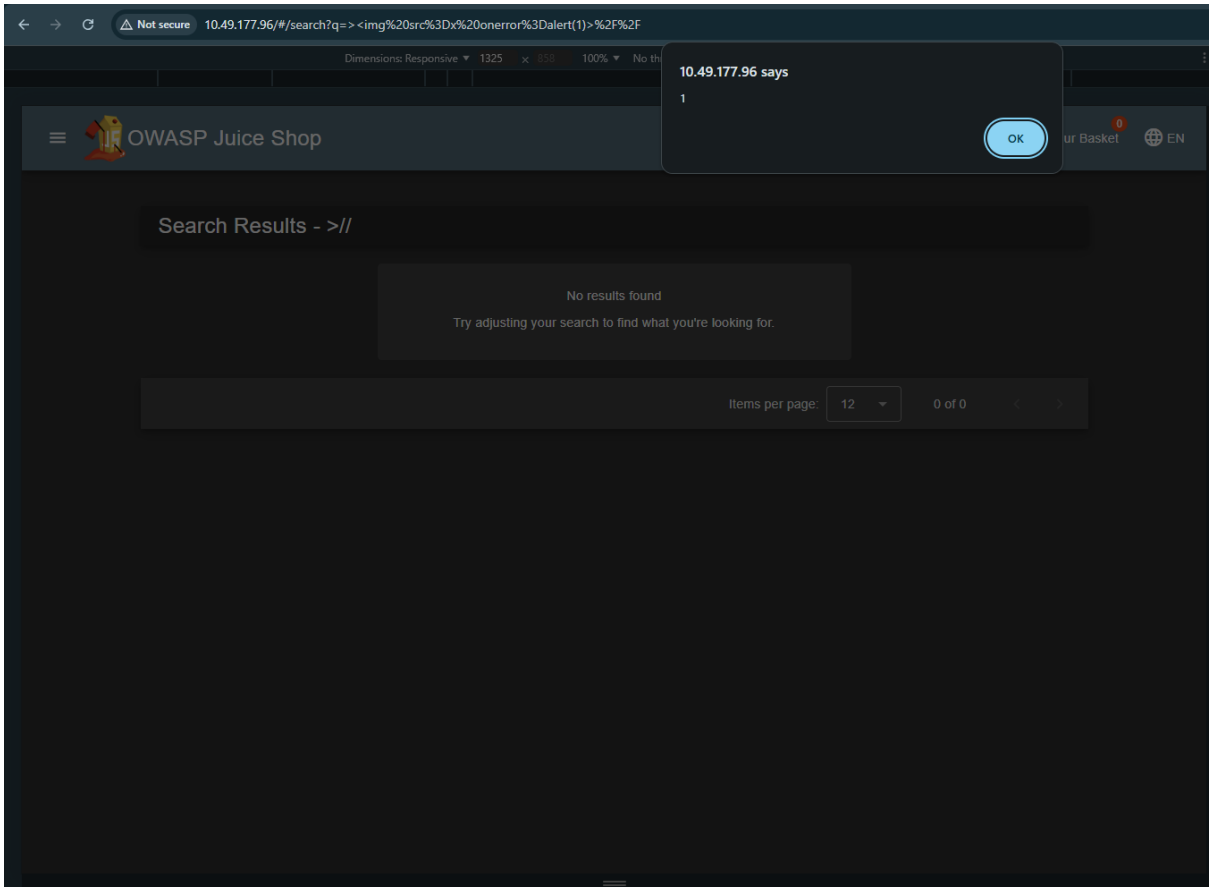
- Search input reflected in response
- No proper output encoding

```
<span _ngcontent-ng-c3242600146 id="searchValue">hi
</span> == $0
```

4. Exploitation – Reflected XSS

Payload Injected:

><img src=x onerror=alert(1)>//



The payload was URL encoded and injected into the search parameter.

The application executed the script successfully, triggering a JavaScript alert.

Technical Impact:

- Arbitrary JavaScript execution
- Session token theft possible
- DOM manipulation
- Account takeover potential

5. Remediation Plan

1. Implement strict server-side input validation
2. Apply context-aware output encoding
3. Deploy Content Security Policy (CSP)
4. Sanitize all user inputs before rendering
5. Use secure frameworks with built-in protection

Non-Technical Briefing

During the security assessment of the OWASP Juice Shop application, a critical security weakness was identified in the product search functionality. The issue allows attackers to inject malicious scripts into the application, which are executed in the user's browser. This type of vulnerability, known as Cross-Site Scripting (XSS), can enable attackers to steal login sessions, impersonate users, or manipulate displayed content.

Although no sensitive data was accessed during this controlled test, the vulnerability presents a significant business risk if exploited in a real-world environment. Immediate remediation is recommended by implementing proper input validation, secure coding practices, and browser security policies.

Addressing this issue will significantly improve the application's resilience against client-side attacks and enhance overall security posture.