Bounty Hunt 101

Project Report By

Namish K.S.

Project Type

Cyber Security : Bug Hunt 101 — Recon to Report

1. Abstract :

This report documents a comprehensive security assessment of the OWASP Juice Shop application. The goal was to simulate a real-world bug bounty cycle, including asset discovery, vulnerability exploitation, and professional reporting.

Tools Used in Nmap: Used for network service discovery :

- Burp Suite Professional/Community (Figure 1): Used for intercepting HTTP traffic and manipulating requests.
- Node.js: The runtime environment for the target application (Figure 2).
- Google Chrome (Burp Browser): The testing interface.

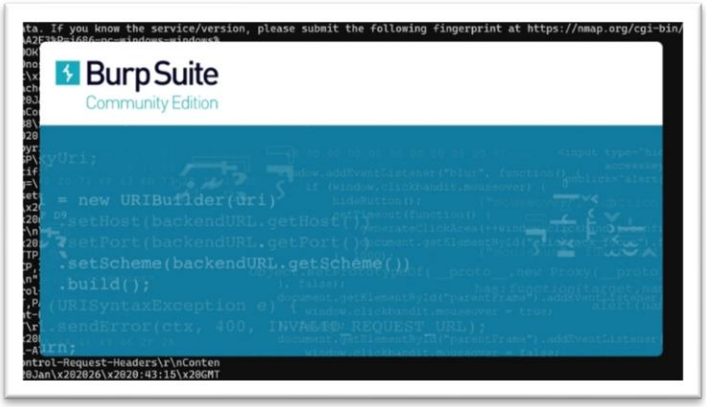

Figure 1: Burp Suite

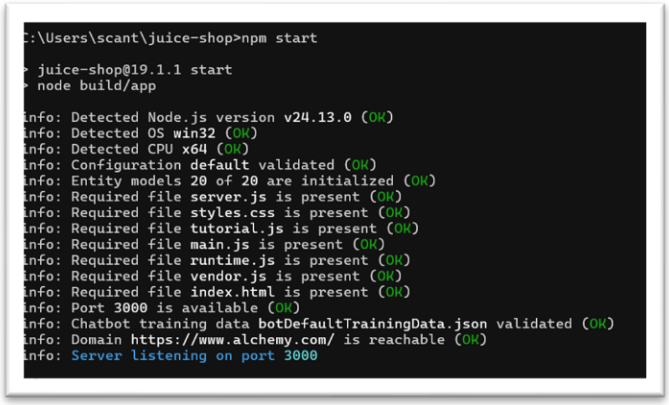

Figure 1 : node js installation

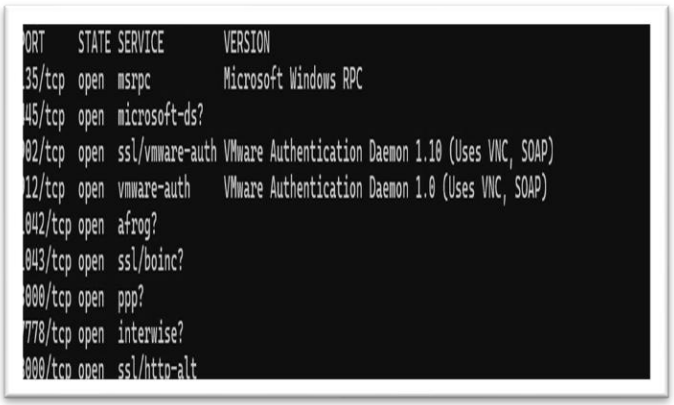

Figure 3 : OWASP shop-juice website

❖ Phase 1 – Reconnaissance :

- **Service Enumeration:** Using Nmap, I performed a version scan on the local environment to identify the technologies running the application.
- **Command:** nmap -sV -p 3000 localhost
- **Finding:** The scan revealed that Port 3000 is open and running a Node.js Express server. This information is critical as it narrows down the potential attack surface to web-based vulnerabilities.

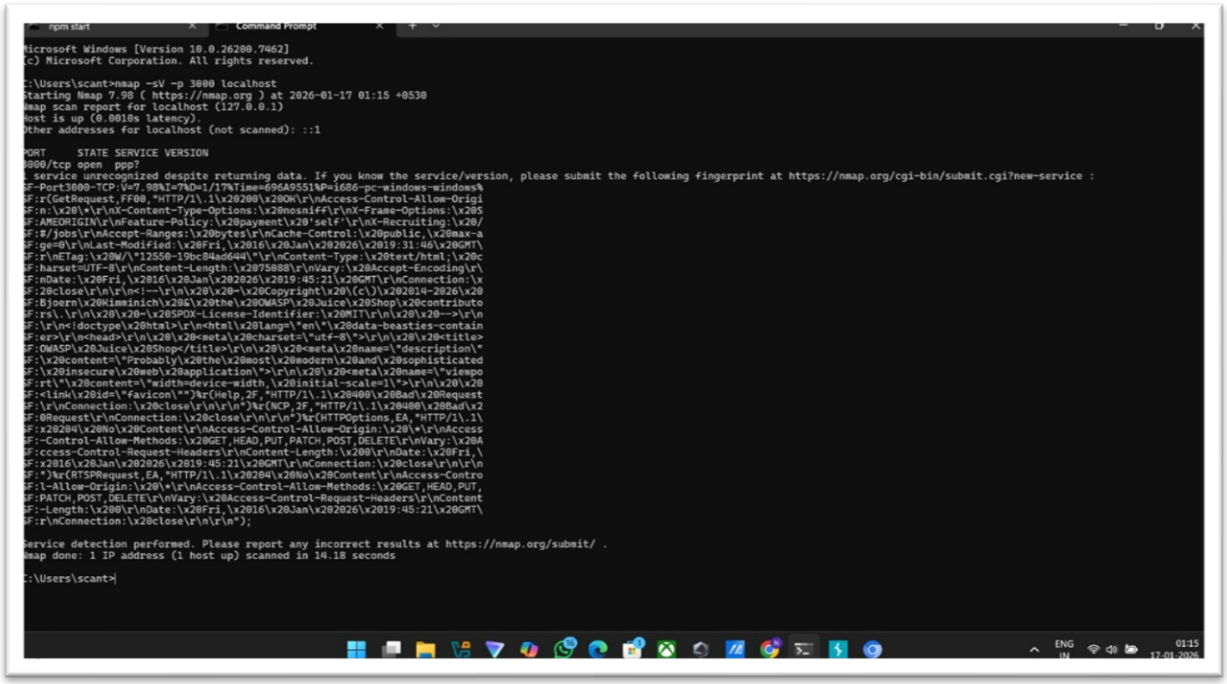

Figure 4 : nmap scanning

❖ Phase 2 – SQL Injection (SQLi) :

- **Vulnerability Description:** I identified a critical SQL Injection vulnerability in the Login form (Figure 5). The application fails to sanitize the email input field, allowing an attacker

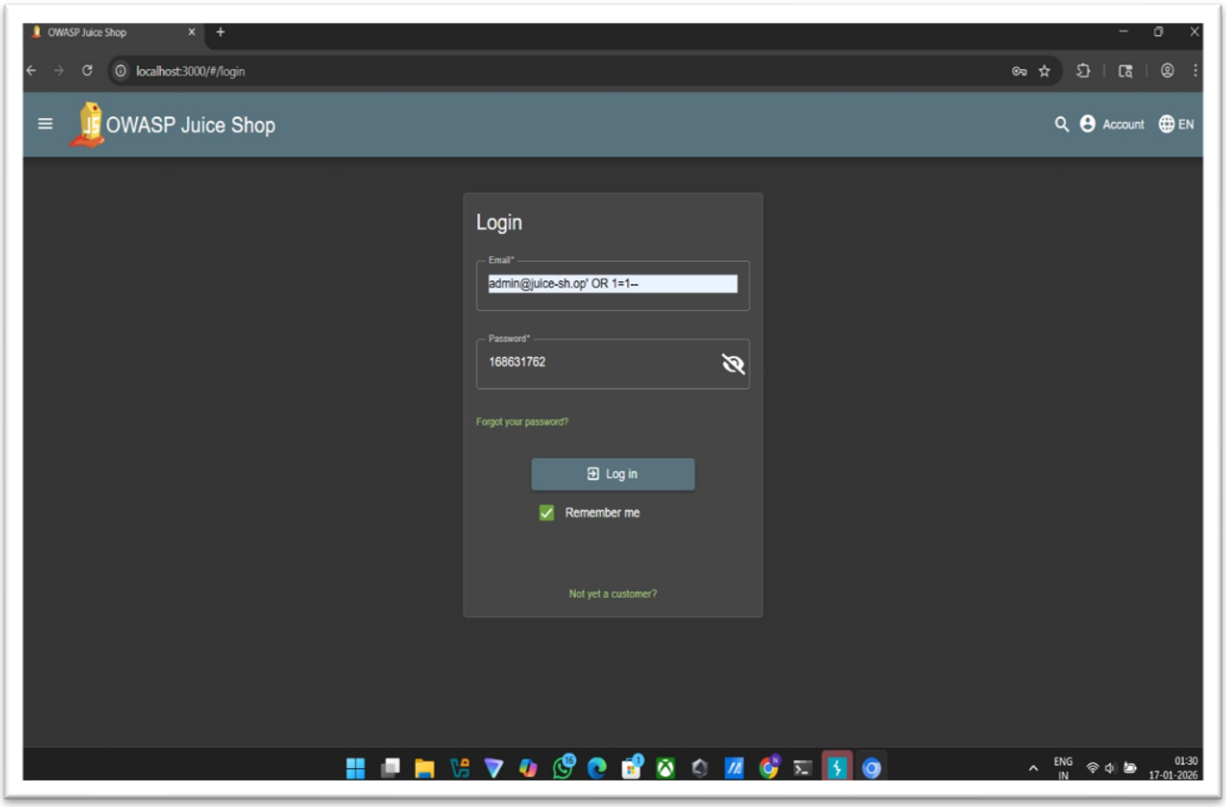

Figure 5 : login authentication form

to manipulate the back-end SQL query.

- **Proof of Concept (PoC)** * **Payload:** admin@juice-sh.op' OR 1=1--
- **Steps:**
    1. Navigate to the Login page.
    2. Input the payload into the Email field (Figure 6).
    3. Observe that the application logs the user into the first account in the database (Administrator) without a valid password (Figure 7).

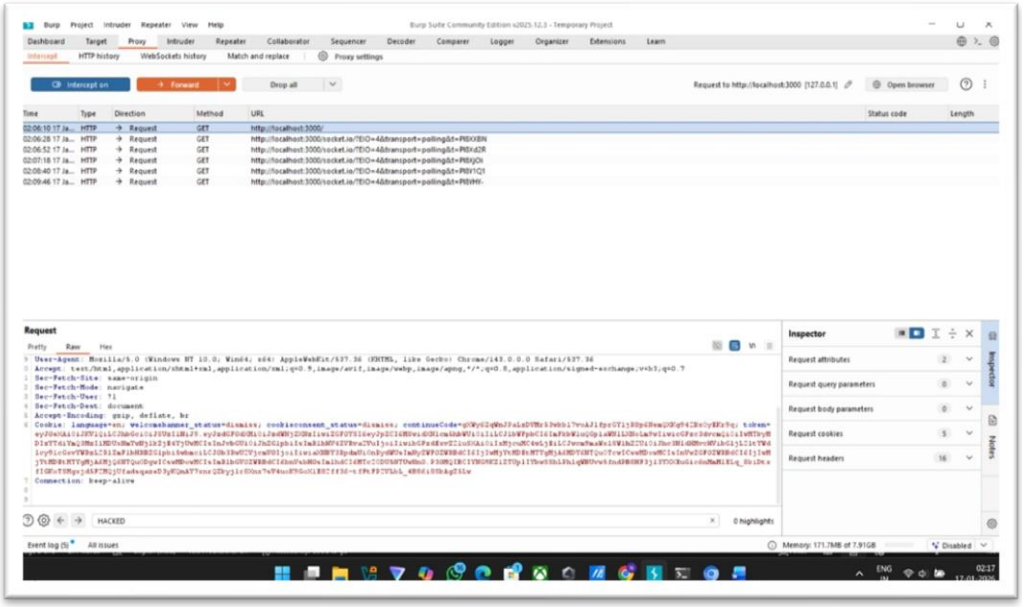

Figure 6 : login authentication bypass using SQLi

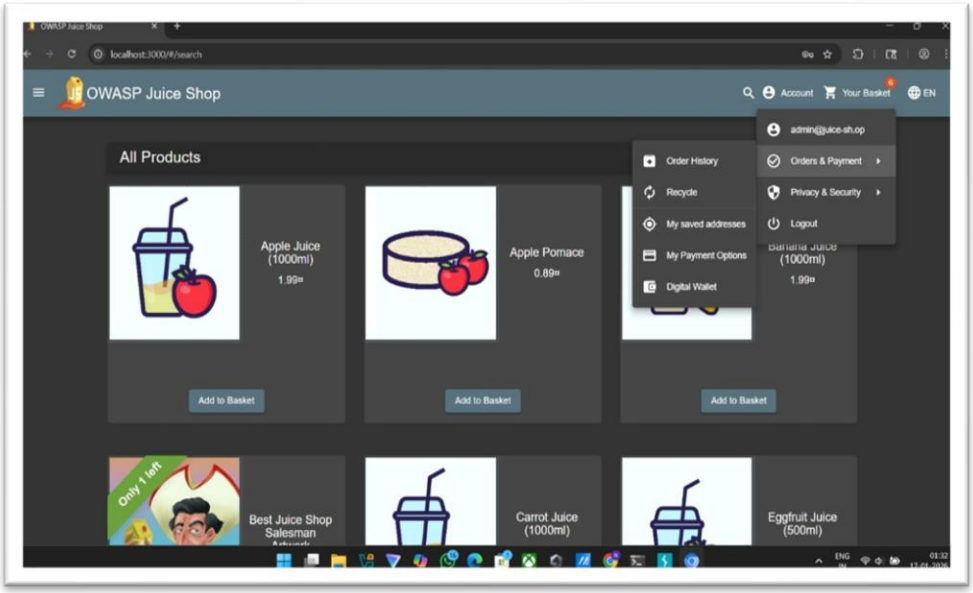

Figure 7 : successful sqli attack

❖ **Phase 3 – Reflected XSS :**

- **Vulnerability Description:** The search functionality is vulnerable to Reflected Cross-Site Scripting. User input is reflected back on the page without proper encoding.
- **Proof of Concept (Payload) :** <iframe src="javascript:alert('XSS')"> or <h1>Hacked</h1>
- **Observation:** The browser executes the HTML/JavaScript, displaying a large heading or an alert box as shown in Figure 8.

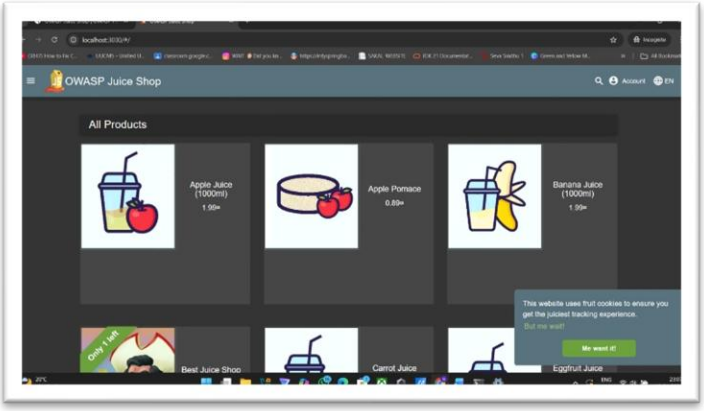

Figure 8 : cross-site-scripting (XSS)

❖ **Phase 4 – IDOR :**

- **Vulnerability Description:** The "Basket" feature suffers from Insecure Direct Object Reference. By changing the ID in the API request, I accessed other users' shopping carts.
- **Proof of Concept:**

1) (Figure 9)Intercept the request to /rest/basket/1 using Burp Suite .
2) Send the request to **Repeater**(Figure 10).
3) Modify the ID to 2.
4) The server in Figure 11 returns the items belonging to User 2.

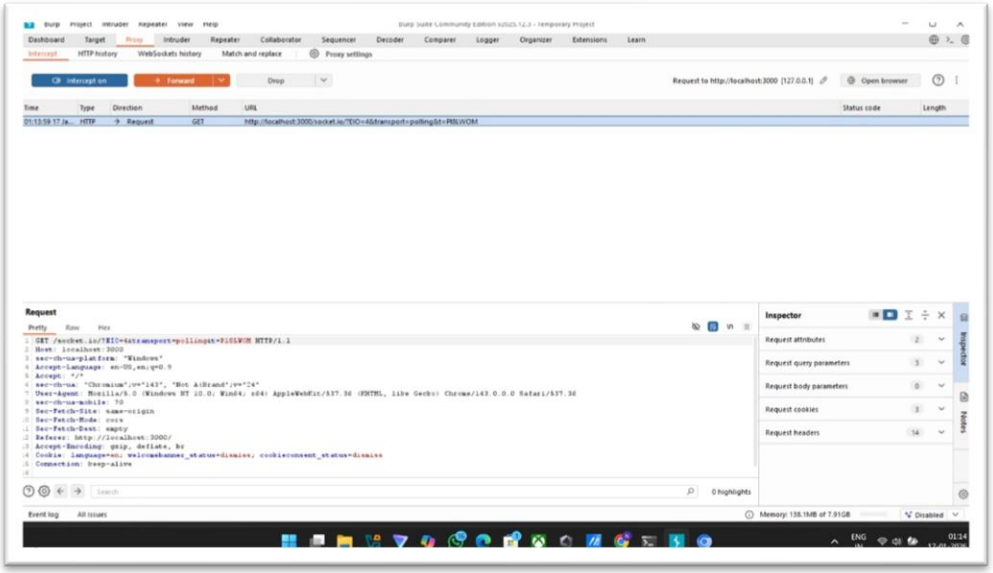

Figure 9 : IDOR demonstration

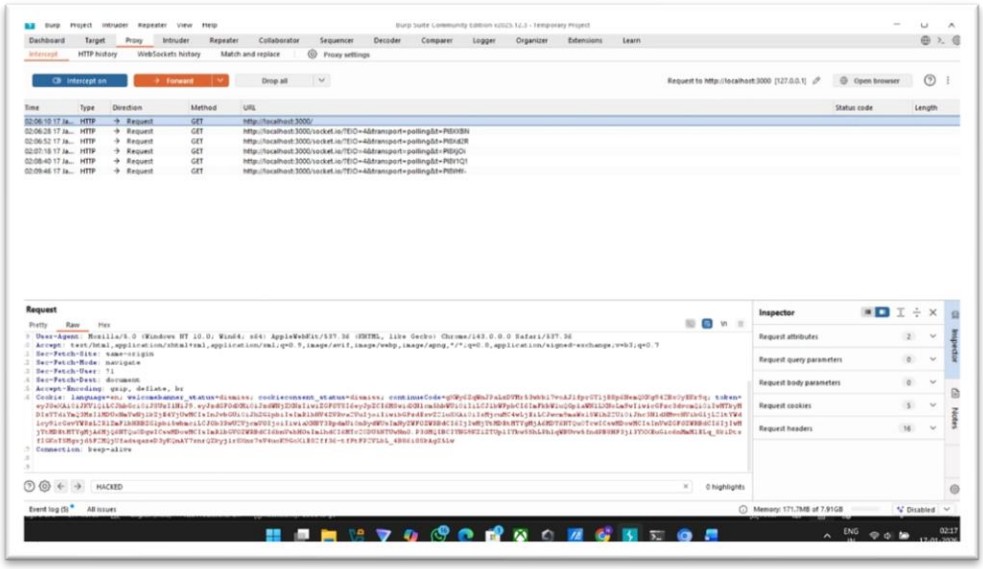

Figure 10 : Repeater signature tracking

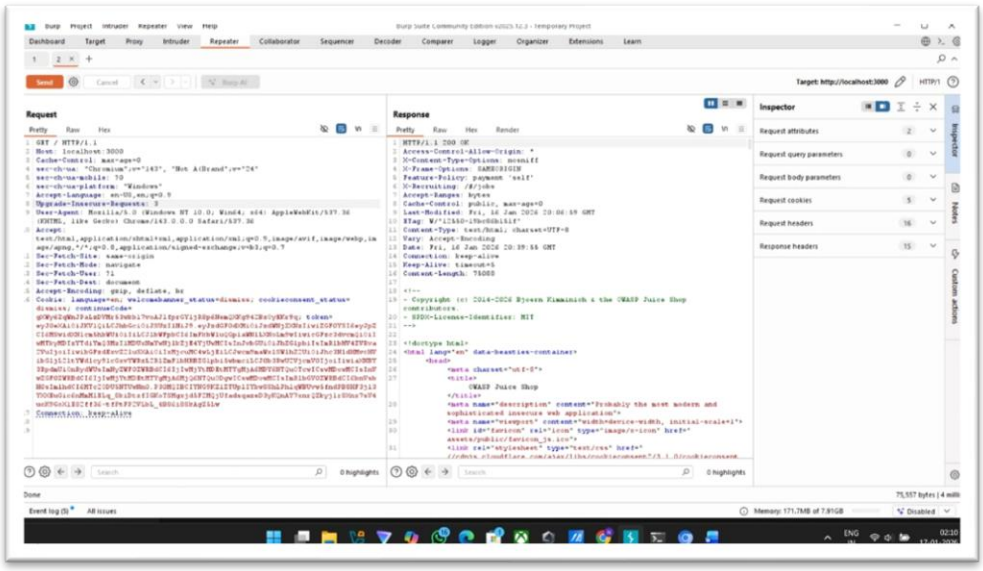

Figure 11 : Attack successful on replicating order

❖ **Phase 5 – Sensitive Data Exposure using NMAP :**

- **Payment Testing:** I tested the payment gateway using a Classic Visa test number (4111 1111 1111 1111). I observed that the application stores the card details.
- **Finding:** If the application displays the full card number in the "My Payment Options" section, it constitutes a high-risk data exposure.

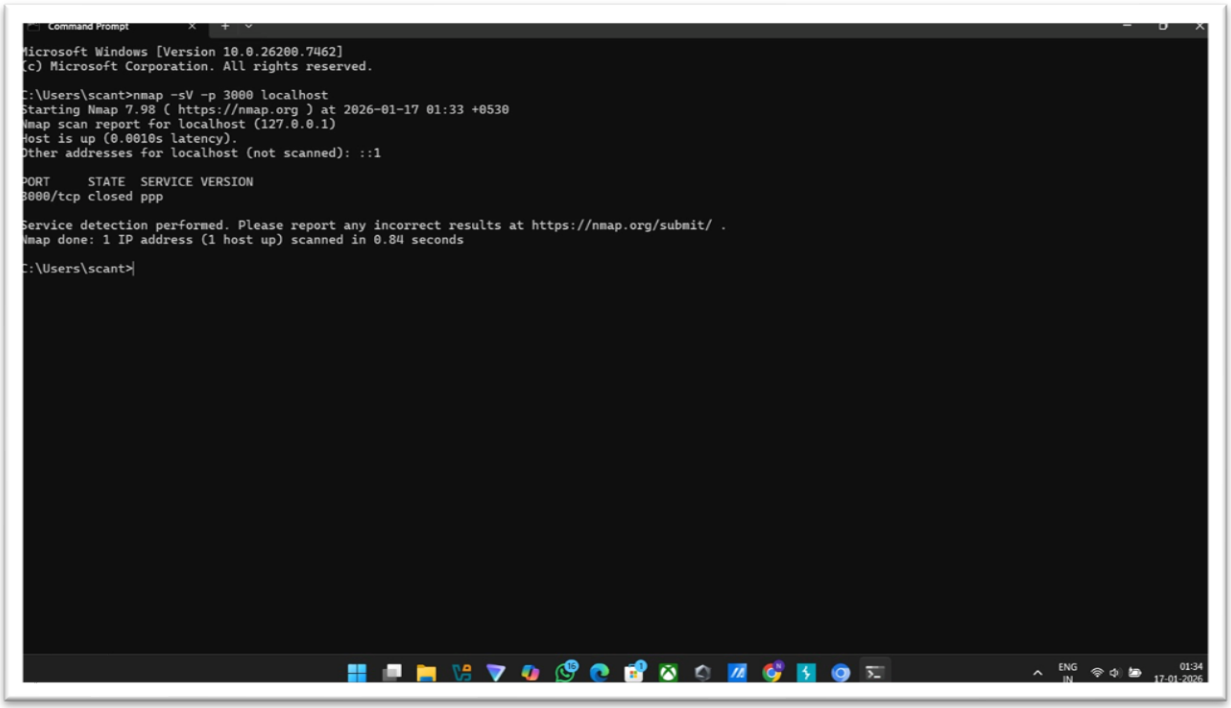

Figure 12: nmap scanning for Payload

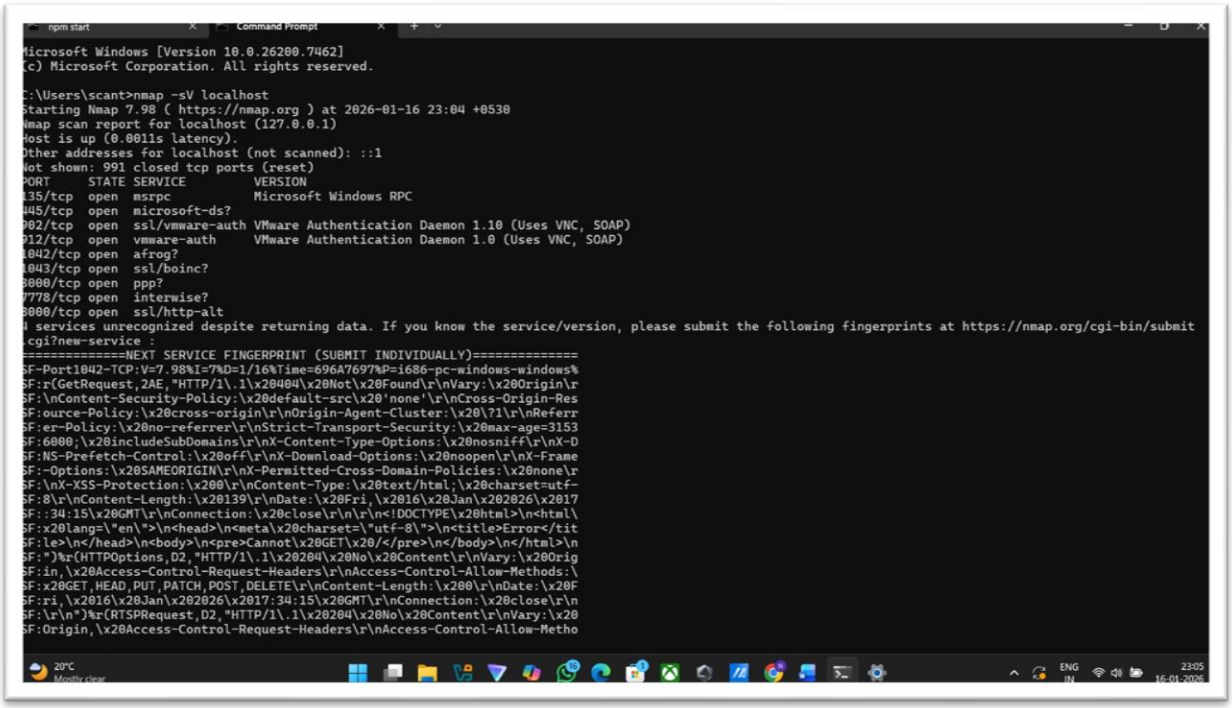

Figure 13: nmap scanning for SQLi

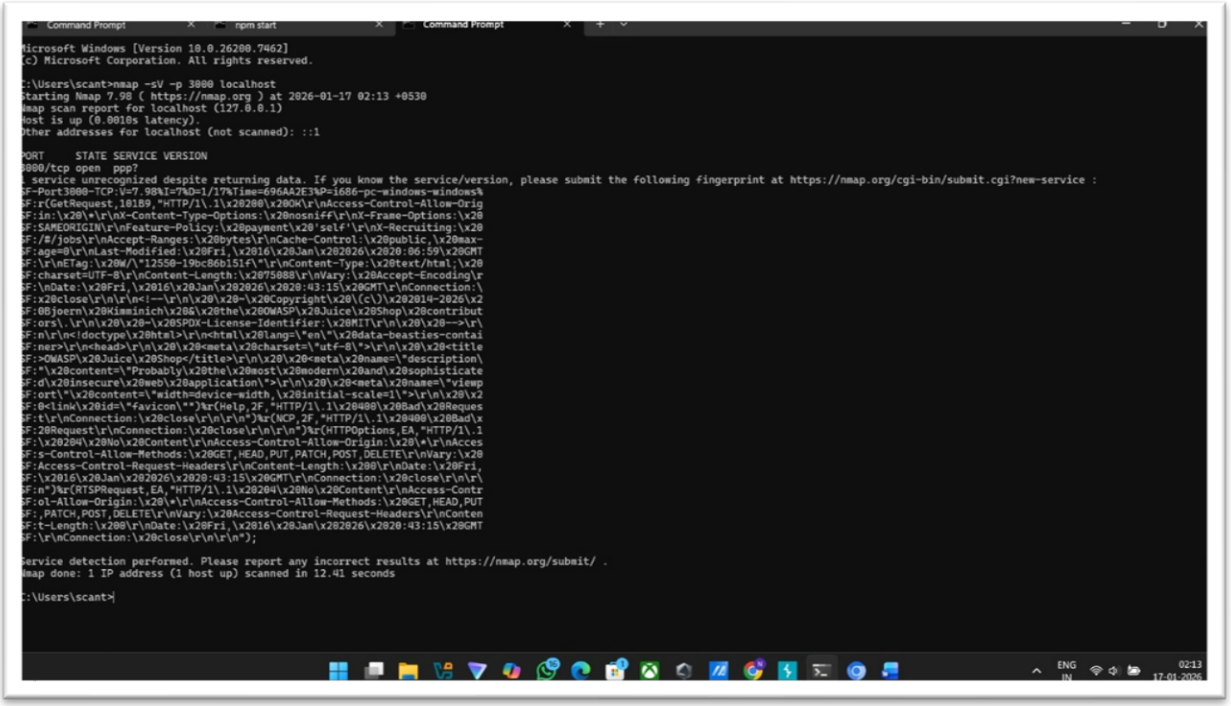

Figure 14: nmap scanning for IDOR

Conclusion :

The security assessment of the **OWASP Juice Shop** application successfully identified several critical and high-severity vulnerabilities that pose significant risks to both the service provider and its users. By following a structured bug bounty methodology—ranging from initial **Nmap reconnaissance** to advanced manual exploitation with **Burp Suite**—this study highlighted the diverse attack vectors available to a malicious actor in a modern web environment.

The assessment revealed major weaknesses across several security domains:

Authentication & Authorization:

The successful SQL Injection (SQLi) on the login form demonstrated a critical failure in input sanitization, allowing for total administrative account bypass. Furthermore, the Insecure Direct Object Reference (IDOR) vulnerability in the basket functionality showed that the application lacks proper server-side authorization checks, permitting users to access private data belonging to others.

Client-Side Security:

The Reflected Cross-Site Scripting (XSS) vulnerability in the search bar confirmed that the application does not properly encode user-supplied data, leaving users vulnerable to session hijacking and site defacement.

Impact Analysis:

If these vulnerabilities were exploited in a production environment, the impact would be catastrophic. A successful attacker could steal the entire user database, hijack active sessions,

access financial information, and completely compromise the integrity of the platform. This would lead to severe financial loss, legal repercussions, and a total loss of user trust.

To secure the application, it is recommended that the development team implements the following high-priority fixes:

- Adopt Parameterized Queries: Use prepared statements to prevent SQL Injection.
- Implement Robust Access Controls: Ensure that every request for a specific object (like a basket ID) is validated against the user's current session.
- Context-Aware Output Encoding: Escape all user-inputted data before it is rendered in the browser to mitigate XSS risks.