WEBSCANPRO

Automated Web Application Security Testing Tool

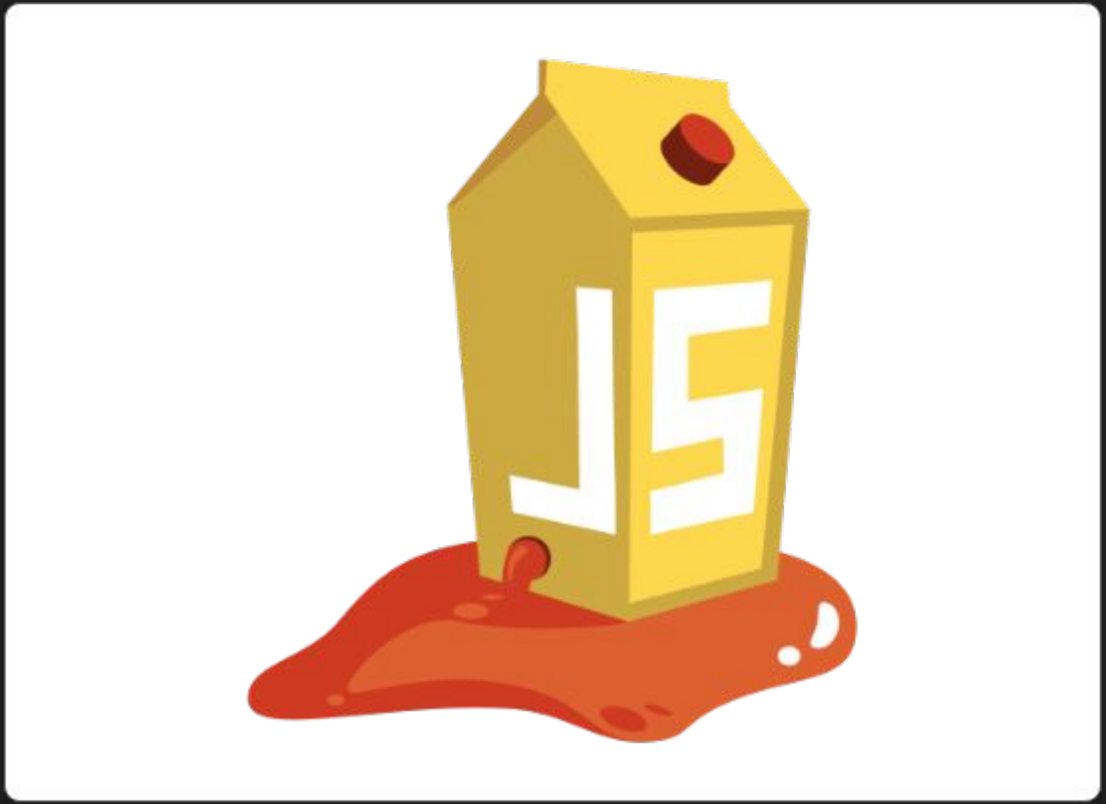
Final Internship Project | January 2026

By- Samar Vijay Vishwakarma

PROJECT OVERVIEW

THE PROBLEM

Manual penetration testing is time-consuming, repetitive, and prone to human error. Modern Single Page Applications (SPAs) are difficult to scan with traditional static analysis tools.



THE SOLUTION

WebScanPro: A custom, modular automation tool designed to detect OWASP Top 10 vulnerabilities including SQL Injection, XSS, and IDOR in a controlled Docker environment.

TECHNOLOGY STACK

PYTHON 3.X

Core logic and modular architecture.

SELENIUM

Dynamic browser automation for SPA testing.

DOCKER

Isolated containerization of the target.

SYSTEM ARCHITECTURE

⚙EXECUTION FLOW

>> **Initialize:** Launch Selenium WebDriver.

>> **Pre-Check:** nuke_popups() clears UI obstructions.

>> **Scan:** Crawl for endpoints & vectors.

🐞ATTACK MODULES

>> **Injection:** SQLi & XSS Payload testing.

>> **Auth:** Brute Force & Session Analysis.

>> **Report:** Aggregate JSON & Generate HTML.

THE TECHNICAL PIVOT

Challenge: Initial implementation using BeautifulSoup failed because Juice Shop is an Angular-based SPA. Static parsers cannot read client-side rendered JavaScript.

Solution: Migrated the entire core engine to Selenium WebDriver.

- >> Enables real user simulation.
- >> Handles dynamic DOM updates.
- >> Bypasses complex UI overlays.

MODULE 1: SQL INJECTION

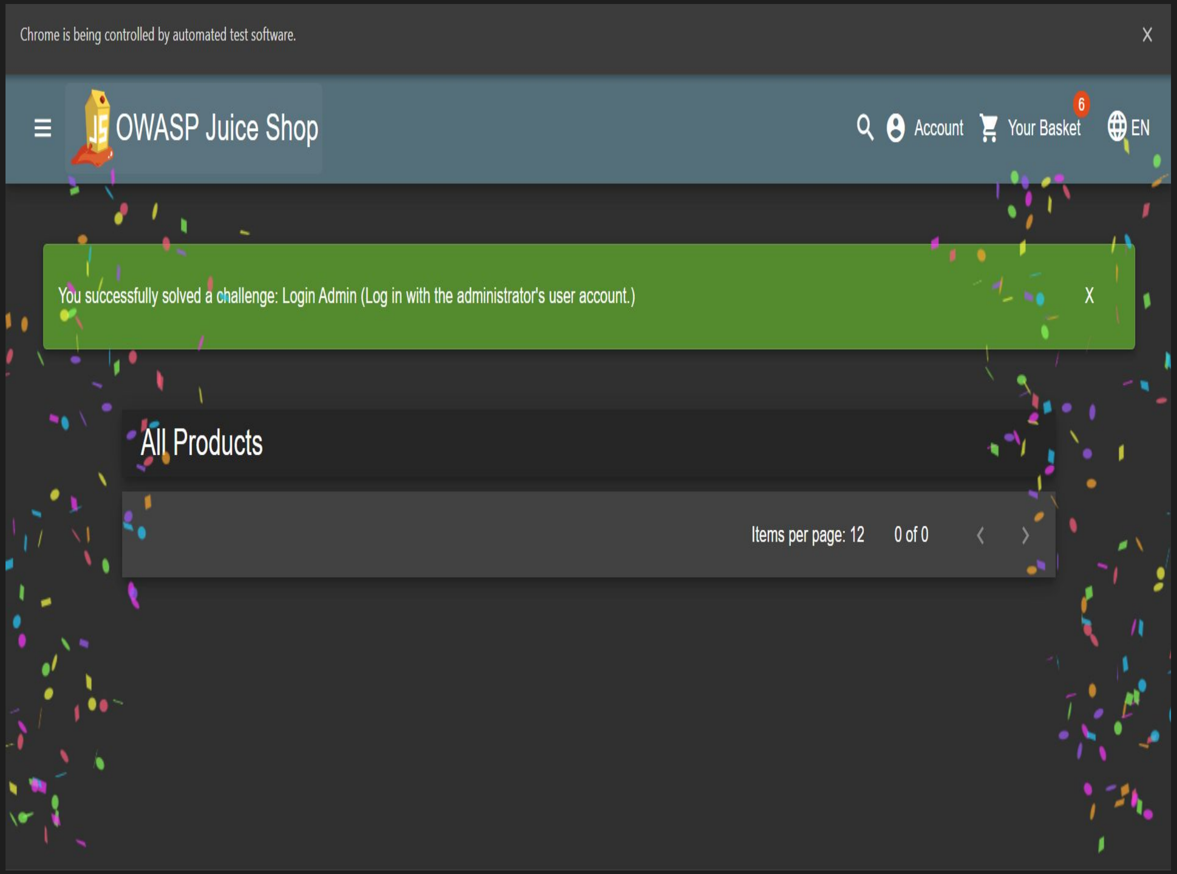
AUTHENTICATION BYPASS

Target: Login Page Email Field

Payload: ' OR 1=1 --

Mechanism: The payload injects a "True" condition, forcing the database to ignore the password check.

Result: Successfully logged in as Administrator without credentials.



Chrome is being controlled by automated test software.

You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)

MODULE 2: CROSS-SITE SCRIPTING

REFLECTED XSS

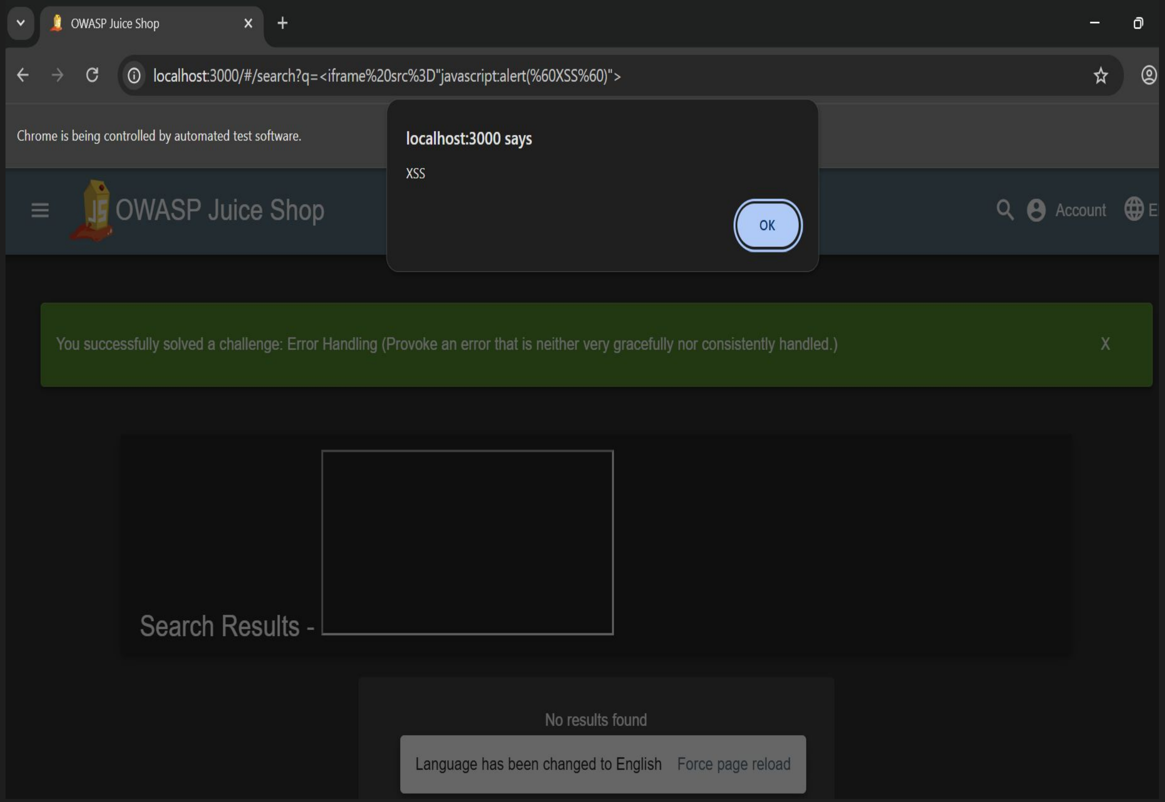
Target: Search API

Payload: </div>

Mechanism: Used direct URL injection to bypass UI timing.

The browser executed the injected JavaScript.

Result: Triggered automated browser alert.



```
localhost:3000/#/search?q=<iframe%20src%3D"javascript:alert(%60XSS%60)">
```

Chrome is being controlled by automated test software.

localhost:3000 says

XSS

You successfully solved a challenge: Error Handling (Provoke an error that is neither very gracefully nor consistently handled.)

Search Results -

No results found

Language has been changed to English

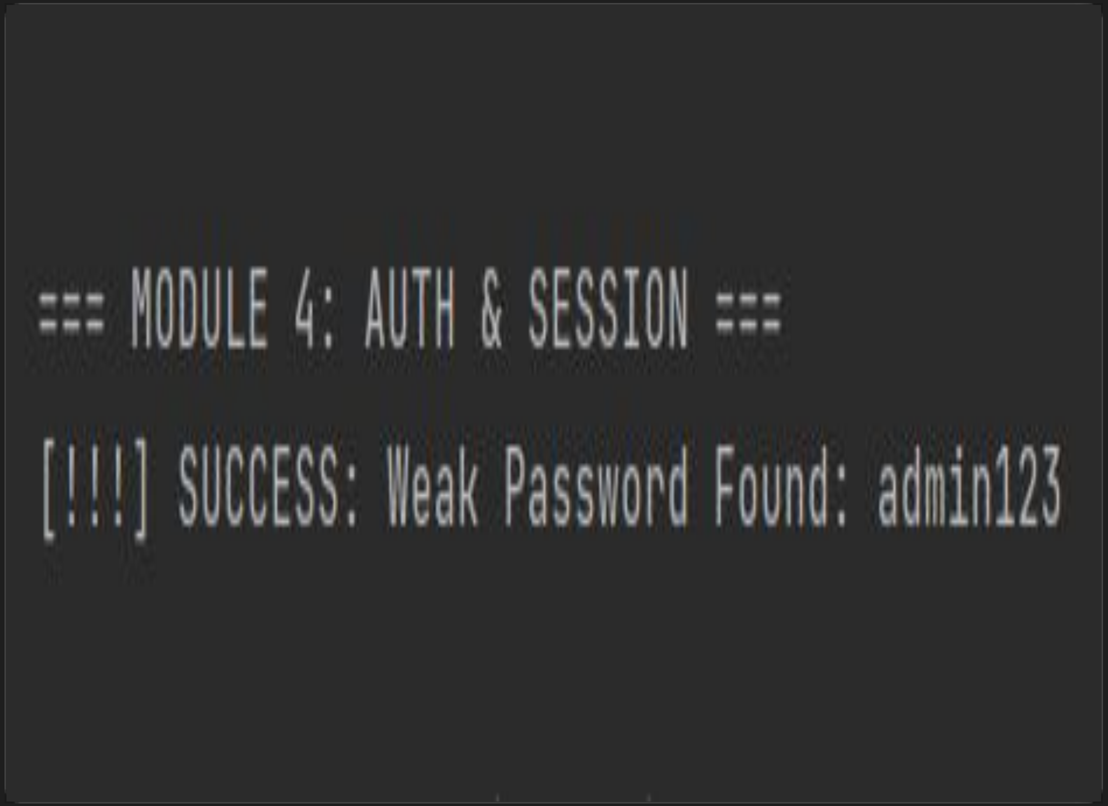
MODULE 3: AUTH & SESSION

BRUTE FORCE & SESSION ANALYSIS

Dictionary Attack: Automated login attempts against the Admin account using common weak passwords.

Session Analysis: Analyzed post-login cookies, identifying missing Secure flags and JWT tokens stored insecurely in LocalStorage.

Result: Cracked admin password: admin123.



```
=== MODULE 4: AUTH & SESSION ===
[!!!] SUCCESS: Weak Password Found: admin123
```

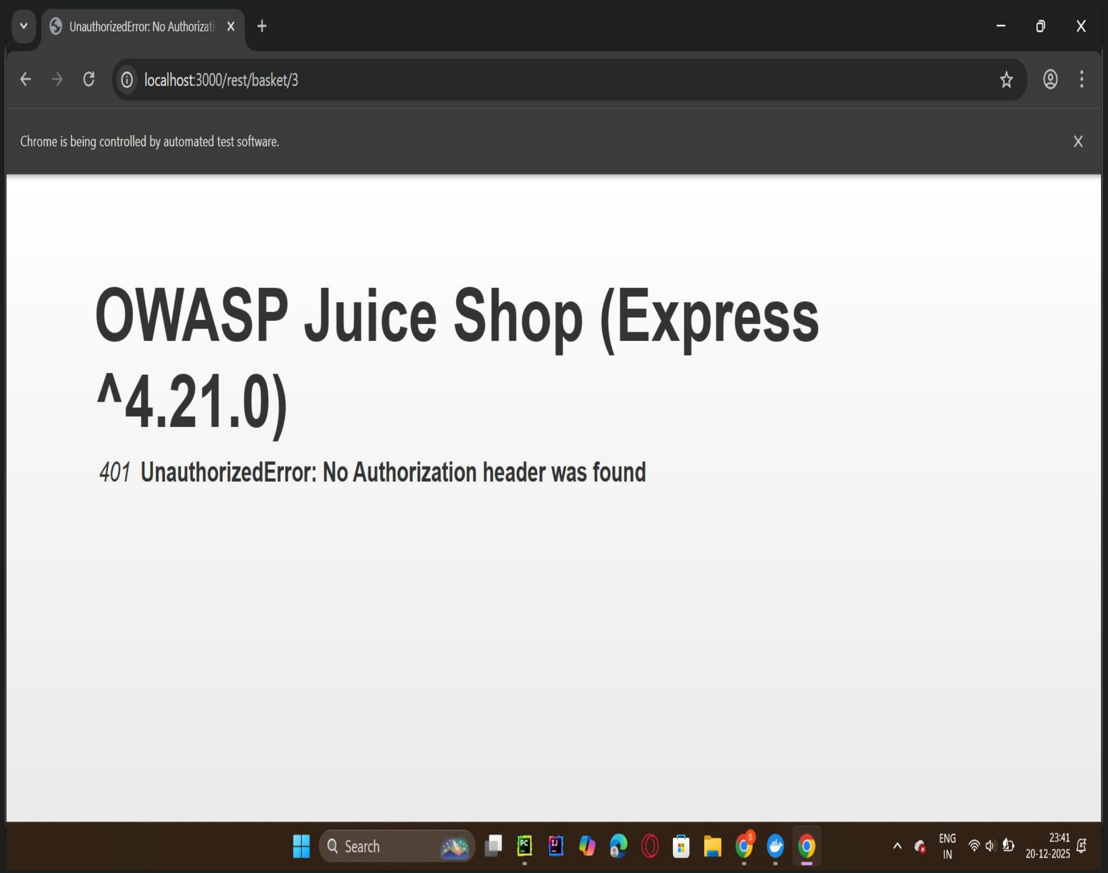
MODULE 4: ACCESS CONTROL (IDOR)

INSECURE DIRECT OBJECT REFERENCE

Target: Shopping Basket API (/rest/basket/id)

Technique: Parameter Manipulation. Changed Basket ID from user's own ID to 1 (Admin).

Result: Server returned full JSON data of the Administrator's shopping cart, proving Broken Access Control.



UnauthorizedError: No Authorizati
localhost:3000/rest/basket/3
Search
ENG
IN
23:41
20-12-2025

Chrome is being controlled by automated test software.

OWASP Juice Shop (Express ^4.21.0)

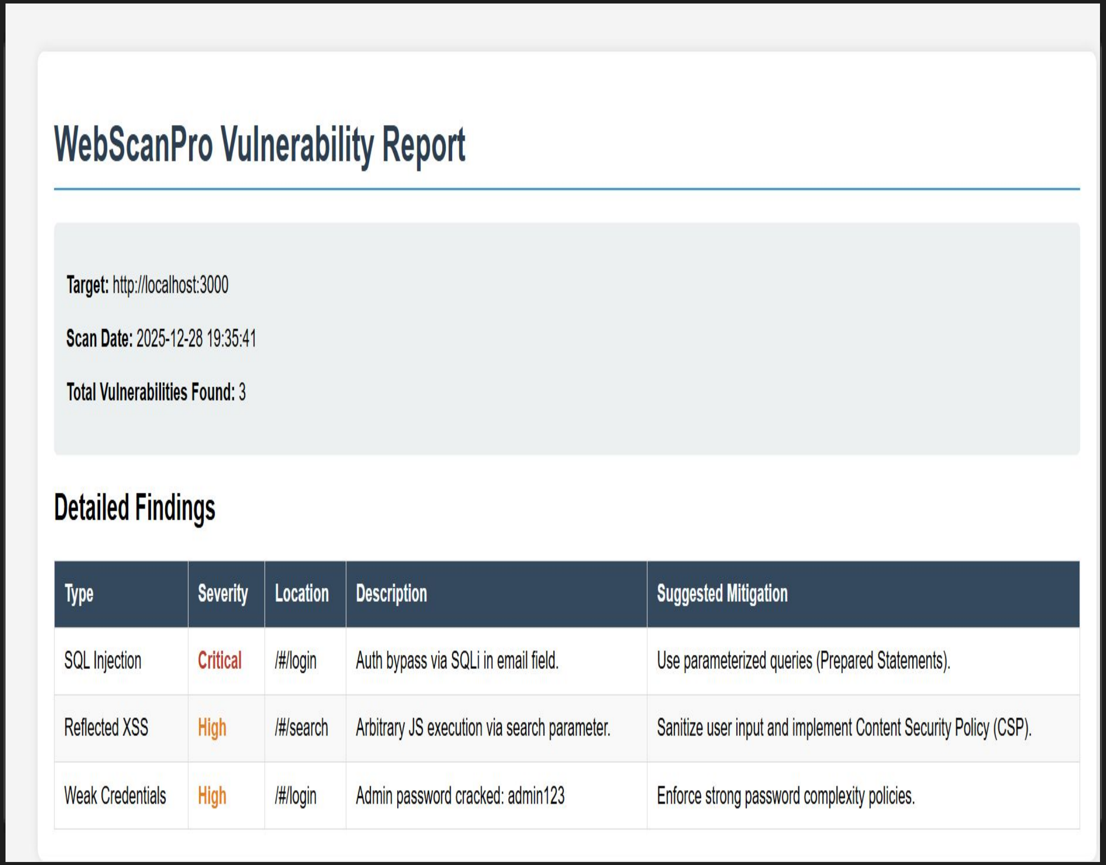
401 UnauthorizedError: No Authorization header was found

FINAL REPORTING ENGINE

AUTOMATED DELIVERABLE

The tool aggregates all findings into a client-ready HTML report.

- >> Summary Table: Scan metadata.
- >> Severity: Color-coded (Critical/High/Medium).
- >> Actionable: Includes mitigation steps.



WebScanPro Vulnerability Report

Target: http://localhost:3000

Scan Date: 2025-12-28 19:35:41

Total Vulnerabilities Found: 3

Detailed Findings

Type	Severity	Location	Description	Suggested Mitigation
SQL Injection	Critical	/#/login	Auth bypass via SQLi in email field.	Use parameterized queries (Prepared Statements).
Reflected XSS	High	/#/search	Arbitrary JS execution via search parameter.	Sanitize user input and implement Content Security Policy (CSP).
Weak Credentials	High	/#/login	Admin password cracked: admin123	Enforce strong password complexity policies.

CONCLUSION & FUTURE SCOPE

🏁ACHIEVEMENTS

Successfully built an end-to-end security scanner. Demonstrated critical vulnerabilities (SQLi, XSS, IDOR) and produced professional audit documentation.

FUTURE SCOPE

Integrate tool into CI/CD pipelines (DevSecOps). Expand modules to detect Stored XSS and perform multi-threaded scanning for speed.

THANK YOU

Questions & Answers

https://github.com/SamarVishwakarma2006/infosys-VI-project