PREPARED BY LAIBA ARIF

AUGUST 3, 2025

WEEK 1

Security Assessment Report

DEVELOPERSHUB

WEEK 1: SECURITY ASSESSMENT REPORT

Date: August 3, 2025

Intern Name: Laiba Arif

Repository: GitHub - internshipDHC

1. OBJECTIVE

THE MAIN GOAL OF THIS TASK WAS TO DO A BASIC SECURITY CHECK ON A MOCK WEB APPLICATION. I STARTED BY SETTING UP AND EXPLORING THE APP TO UNDERSTAND HOW IT WORKS. THEN, I USED BOTH AUTOMATED TOOLS (LIKE OWASP ZAP) AND MANUAL TESTING METHODS TO FIND POSSIBLE SECURITY ISSUES. MY FOCUS WAS ON SPOTTING COMMON WEB VULNERABILITIES, LIKE XSS OR WEAK LOGIN SETUPS. IN THE END, I NOTED DOWN ALL THE ISSUES I FOUND AND GAVE SUGGESTIONS FOR HOW THE APP CAN BE IMPROVED.

2. APPLICATION SETUP

Application Used:

A mock Node.js web application cloned from GitHub.

Setup Steps:

- Cloned the repository
- Installed dependencies using `npm install`
- Launched the application using `npm start`
- Accessed the app at: `http://localhost:3000`

Explored Pages:

- **Signup Page**
- **Login Page**
- **User Profile Page**
- **About Page**

Each page was explored for inputs, behavior, and potential weaknesses.

3. VULNERABILITY ASSESSMENT

Tools Used:

Tool	Purpose
OWASP ZAP	Automated scanning for known web vulnerabilities
Browser Dev Tools	Manual testing for XSS via script injection
Manual Testing	SQL injection attempts during login

📋 Findings

#	Vulnerability	Description	Tool Used	Risk Level	Status
1	XSS (Cross-Site Scripting)	Injected `<script>alert('XSS')</script>` in multiple input fields. No alert appeared, input was escaped.	Browser Dev Tools	Low	Not Vulnerable
2	SQL Injection	Attempted login using `admin' OR '1'='1` for username and password. Login failed, meaning input likely sanitized.	Manual	High	Not Vulnerable
3	Missing HTTP Security Headers	ZAP reported missing headers like CSP, X-Frame-Options, and HSTS.	OWASP ZAP	Medium	Vulnerable
4	Insecure Cookies	Cookies did not have `Secure` and `HttpOnly` flags set, which may allow interception or client-side access.	OWASP ZAP	Low	Vulnerable
5	Directory Browsing Enabled	ZAP flagged that directory listing might be enabled on certain routes.	OWASP ZAP	Low	Vulnerable

4. DETAILED VULNERABILITY BREAKDOWN

🔷 1. Missing Security Headers

- **Issue**: HTTP responses did not include important headers like:
    - `Content-Security-Policy`
    - `X-Frame-Options`
    - `Strict-Transport-Security`
- **Risk**: Increases chances of XSS, clickjacking, and MITM attacks.
- **Recommendation**: Add these headers in the server configuration.

🔷 2. Insecure Cookies

- **Issue**: Application set cookies without using `Secure` and `HttpOnly` flags.
- **Risk**: Cookies could be stolen by attackers if sent over non-HTTPS or accessed via JavaScript.
- **Recommendation**: Always set:
- `res.cookie('token', value, { secure: true, httpOnly: true });`

🔷 3. Directory Browsing

- **Issue**: Some routes or directories might expose their contents.
- **Risk**: Attackers can see file structure or access sensitive files.
- **Recommendation**: Disable directory browsing in server settings (e.g., Express or Apache config).

5. AREAS OF IMPROVEMENT

Area	Recommendation
Security Headers	Implement security headers in server responses using Helmet.js or manual headers
Cookie Security	Use `Secure` and `HttpOnly` flags to protect session cookies
Input Validation	Although current login rejects injections, continue applying server-side validation
Regular Scanning	Perform regular vulnerability scans using OWASP ZAP or similar tools
Error Handling	Avoid displaying raw error messages that may expose backend structure

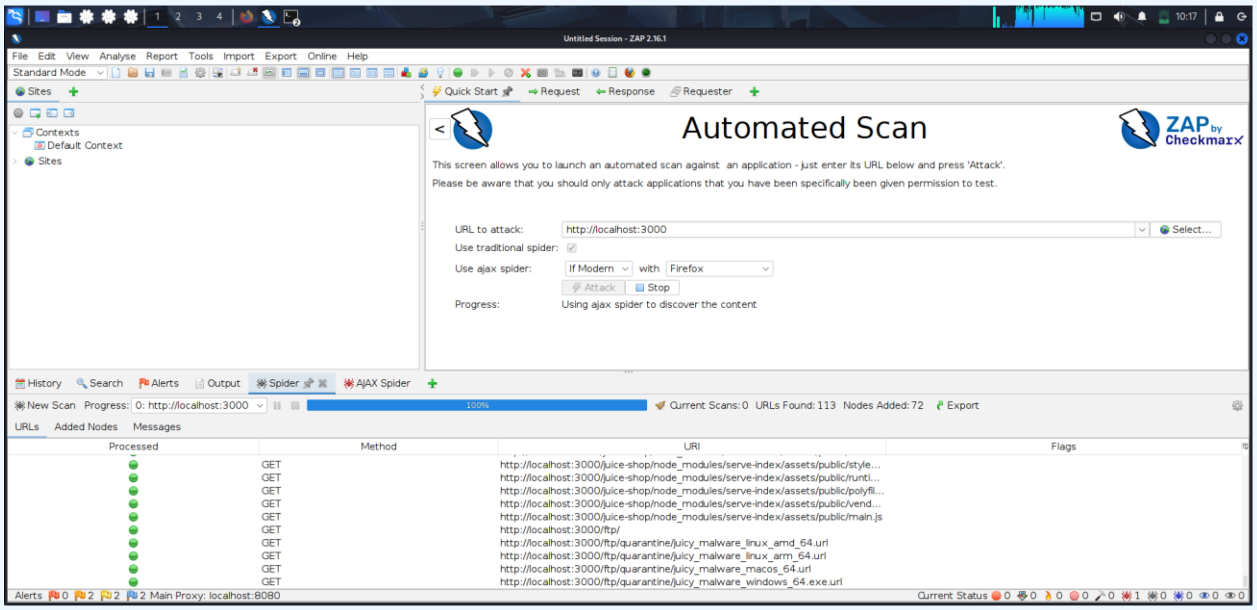
6. SUPPORTING EVIDENCE

✅ ZAP Scan Report:

- **File Name**: `internship-vuln-scan.pdf`
- **Location**: GitHub Repo – internshipDHC

🖼 Screenshots:

- OWASP ZAP scan progress (Spider and Active Scan)



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

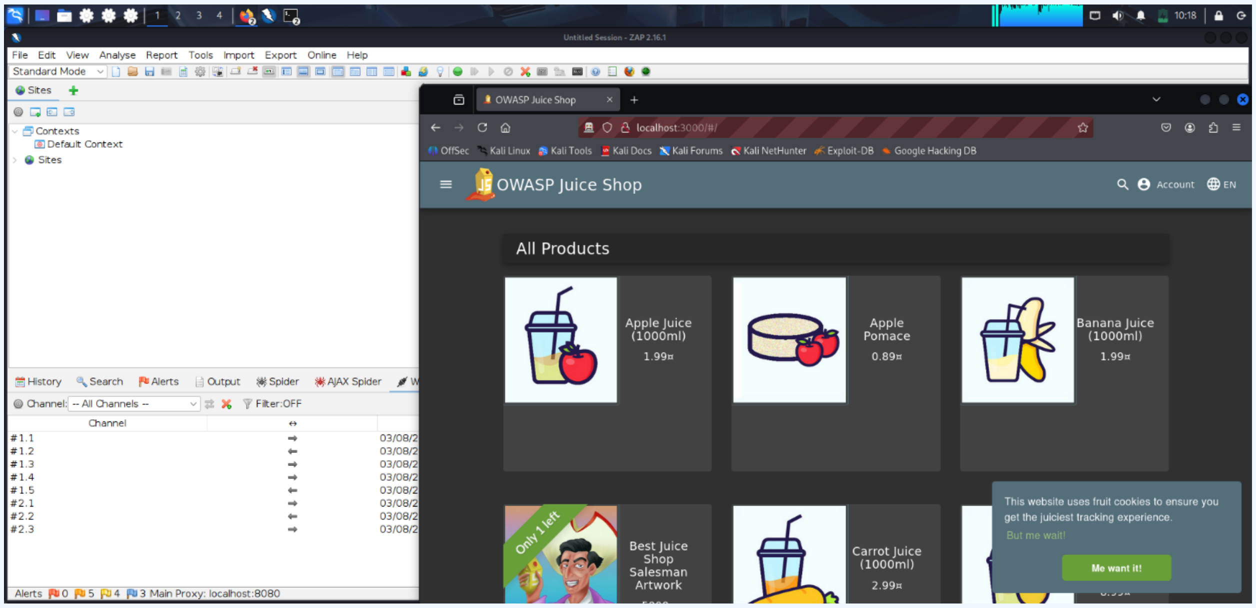
URL to attack: http://localhost:3000
Use traditional spider: ✓
Use ajax spider: If Modern with Firefox
Attack Stop
Progress: Using ajax spider to discover the content

New Scan Progress: 0: http://localhost:3000 100% Current Scans: 0 URLs Found: 113 Nodes Added: 72 Export

Processed	Method	URI	Flags
●	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/style...	
●	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/runti...	
●	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/polyfil...	
●	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/vend...	
●	GET	http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/main.js	
●	GET	http://localhost:3000/ftp/	
●	GET	http://localhost:3000/ftp/quarantine/juicy_malware_linux_amd_64.url	
●	GET	http://localhost:3000/ftp/quarantine/juicy_malware_linux_arm_64.url	
●	GET	http://localhost:3000/ftp/quarantine/juicy_malware_macos_64.url	
●	GET	http://localhost:3000/ftp/quarantine/juicy_malware_windows_64.exe.url	

Alerts 0 2 2 2 Main Proxy: localhost:8080 Current Status 0 0 0 0 0 1 0 0 0 0

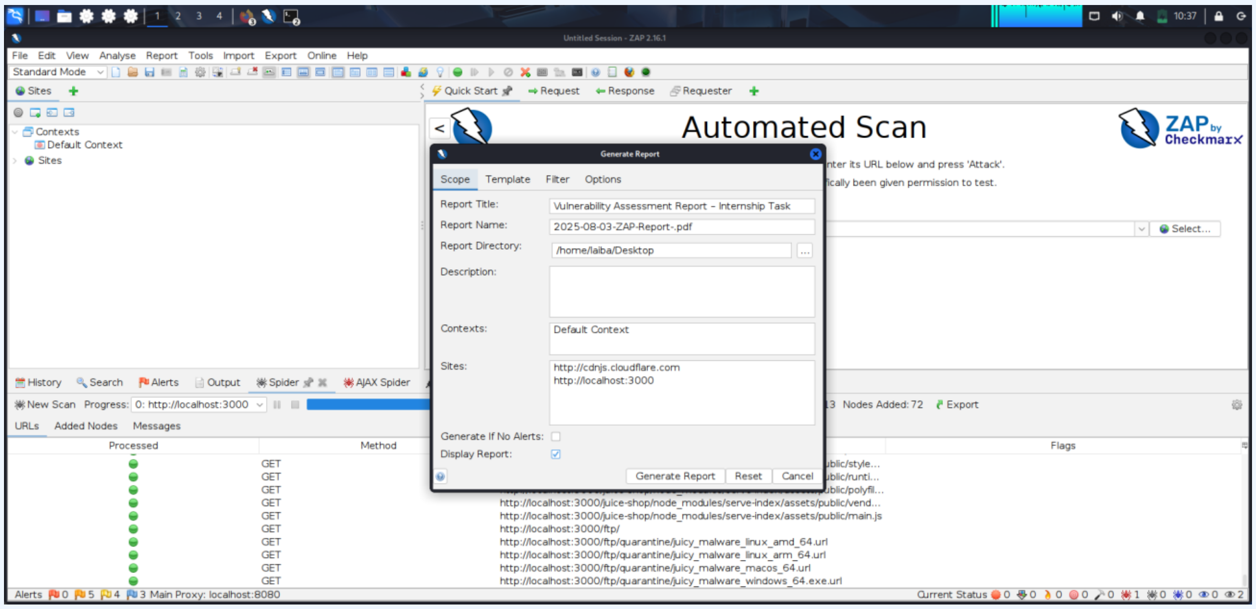
- Attack page with Firefox pop-up


Untitled Session - ZAP 2.16.1
OWASP Juice Shop
localhost:3000/#/
All Products
Apple Juice (1000ml) 1.99¤
Apple Pomace 0.89¤
Banana Juice (1000ml) 1.99¤
Only 1 left
Best Juice Shop Salesman Artwork
Carrot Juice (1000ml) 2.99¤
Me want it!
Alerts 0 5 4 3 Main Proxy: localhost:8080

This website uses fruit cookies to ensure you get the juiciest tracking experience.

But me wait!

- Report generation window


Untitled Session - ZAP 2.16.1
Automated Scan
ZAP by Checkmarx
Generate Report
Scope Template Filter Options
Generate Report Reset Cancel
Nodes Added: 72 Export
Alerts 0 5 4 3 Main Proxy: localhost:8080

Report Title: Vulnerability Assessment Report - Internship Task
Report Name: 2025-08-03-ZAP-Report-.pdf
Report Directory: /home/laiba/Desktop
Description:
Contexts: Default Context
Sites: http://cdnjs.cloudflare.com
http://localhost:3000
Generate If No Alerts: ☐
Display Report: ✓

Progress: 0: http://localhost:3000

```
GET  http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/vend...
GET  http://localhost:3000/juice-shop/node_modules/serve-index/assets/public/main.js
GET  http://localhost:3000/ftp/
GET  http://localhost:3000/ftp/quarantine/juicy_malware_linux_amd_64.url
GET  http://localhost:3000/ftp/quarantine/juicy_malware_linux_arm_64.url
GET  http://localhost:3000/ftp/quarantine/juicy_malware_macos_64.url
GET  http://localhost:3000/ftp/quarantine/juicy_malware_windows_64.exe.url
```

7. CONCLUSION

The mock application shows **good initial security** practices such as rejecting basic SQL injection and XSS attacks. However, there are still **important areas of improvement**, especially regarding **HTTP headers and cookie protection**.

Regular testing and secure coding practices are essential to keep web applications resilient against evolving threats.