Cybersecurity Internship– Week 6 Report

Intern Name: Muhammad Ali Kashif

Project: Advanced Security Audits & Final Deployment OWASP Juice Shop

Week: 6

Objective

The objective of this week was to conduct advanced security audits on the OWASP Juice Shop application, evaluate compliance with industry security standards, and prepare the application for secure deployment.

Multiple automated and manual security testing tools were used, including **OWASP ZAP, Nikto, Trivy, and Burp Suite**, to identify vulnerabilities, misconfigurations, and potential attack vectors.

The assessment revealed several security weaknesses such as missing security headers, vulnerable dependencies, exposed cryptographic keys, and cross-domain risks. These findings highlight the importance of proactive security testing before production deployment.

Overall, this security audit demonstrates the implementation of industry-standard penetration testing practices and secure deployment methodologies.

1. Application Deployment

The OWASP Juice Shop application was deployed locally using Docker to simulate a real-world web application environment suitable for penetration testing.

The containerized setup ensured consistency, isolation, and efficient vulnerability analysis.

Screenshot: OWASP Juice Shop running on localhost inside Docker

2. Security Audits & Compliance

2.1 OWASP ZAP Scan

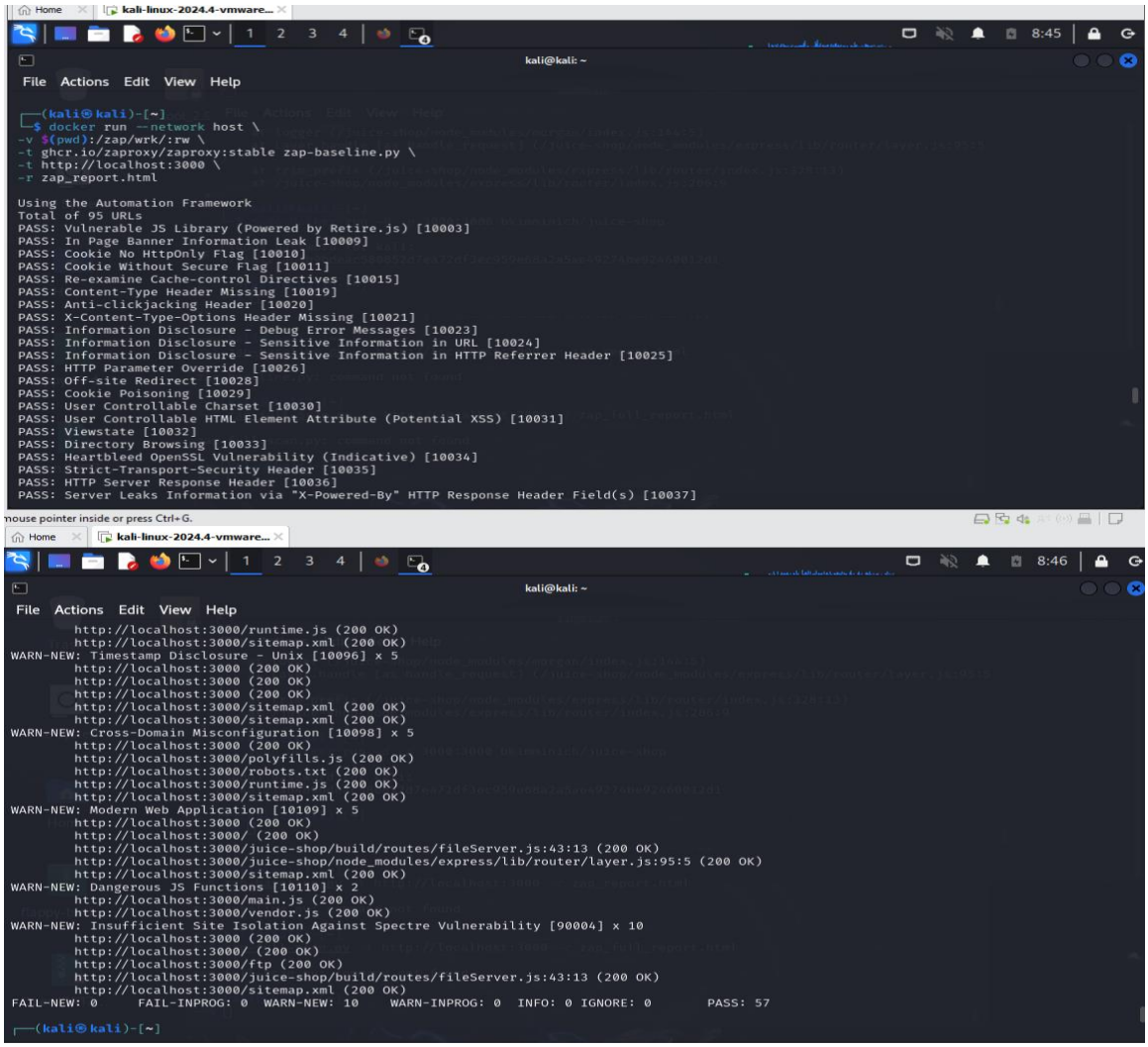
An automated baseline scan was performed using OWASP ZAP to identify common web vulnerabilities.

Key Findings:

- Missing Content Security Policy (CSP)
- Cross-domain misconfiguration
- Dangerous JavaScript functions detected
- Suspicious comments in source files
- Insufficient site isolation protections

These vulnerabilities could potentially allow attackers to inject malicious scripts, exploit browser behavior, or access sensitive data.

Screenshot: OWASP ZAP scan highlighting security warnings.



2.2 Nikto Web Server Scan

Nikto was used to detect server misconfigurations and exposed files.

Key Findings:

- Presence of uncommon HTTP headers
- Potential backup and certificate files accessible
- Missing X-Content-Type-Options header
- Robots.txt exposing sensitive directories

Such issues may assist attackers in reconnaissance and increase the attack surface.

Screenshot: Nikto scan results showing web server weaknesses

2.3 Container Vulnerability Scanning (Trivy)

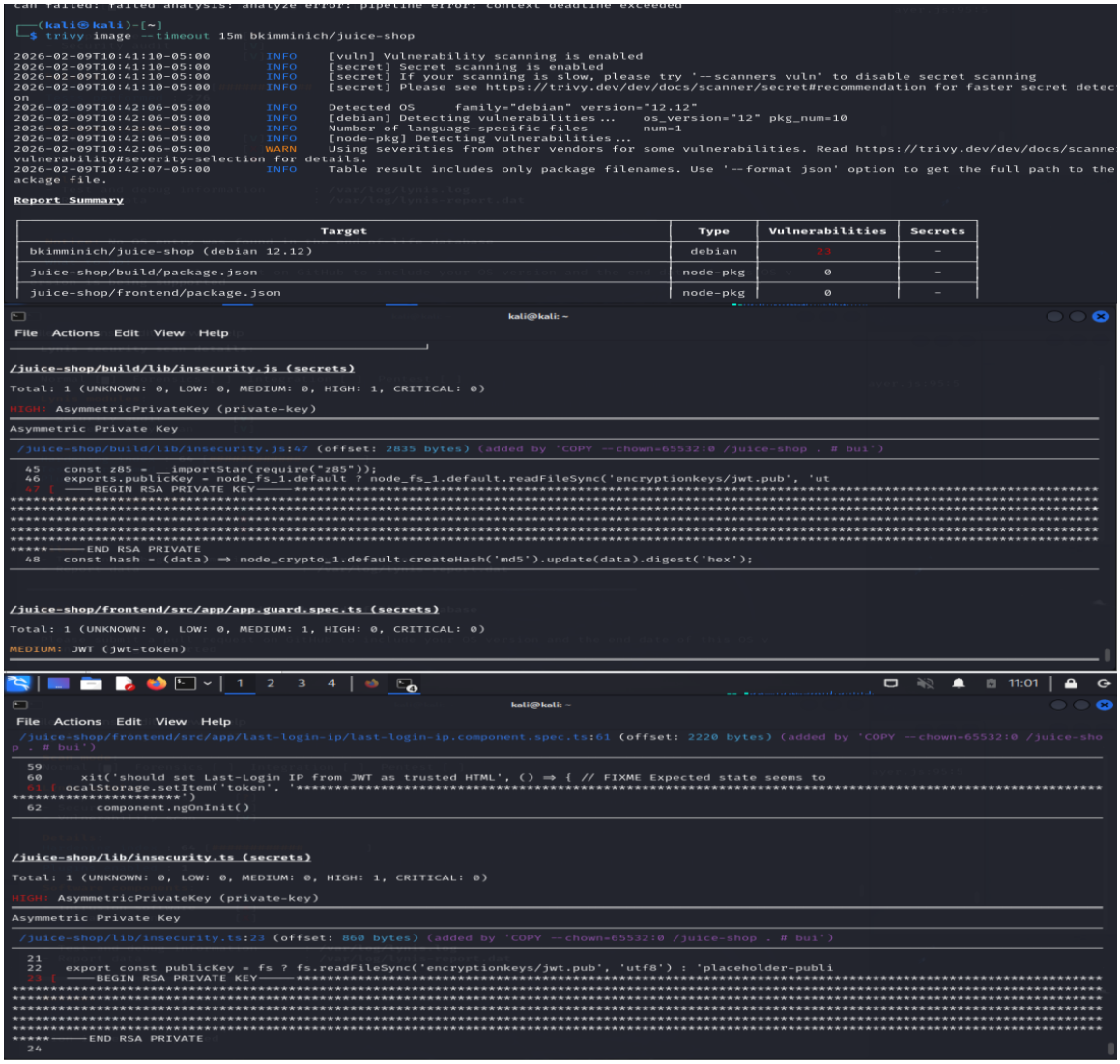
Trivy was used to scan the Docker image for vulnerable dependencies and secrets.

Critical Observations:

- **CRITICAL:** vm2 sandbox escape vulnerability
- **HIGH:** Denial-of-service risk in WebSocket package
- Detection of an exposed **RSA private key**
- Hardcoded JWT tokens found in project files

Exposure of private keys represents a severe cryptographic risk that could compromise authentication mechanisms.

Screenshot: Trivy container scan identifying critical vulnerabilities.



3. OWASP Top 10 Compliance Check

The application was evaluated against OWASP Top 10 security risks.

OWASP Risk	Observation
Broken Access Control	Restricted admin routes detected
Cryptographic Failures	Private key exposure identified
Injection	Potential risk due to unsafe JS functions
Security Misconfiguration	Missing headers and CSP

OWASP Risk	Observation
Vulnerable Components	Multiple outdated dependencies
Identification & Authentication Failures	JWT tokens present in files
Software Integrity Failures	Vulnerable packages detected

The assessment confirms that intentionally vulnerable applications like Juice Shop expose risks commonly found in real-world systems.

4. Secure Deployment Practices

4.1 Dependency & Image Scanning

Container images were scanned prior to deployment to identify exploitable libraries.

This proactive approach reduces the likelihood of deploying vulnerable software into production environments.

4.2 Docker Security Practices Followed

- Used official container images
- Performed vulnerability scanning
- Maintained isolated runtime environment
- Avoided exposing unnecessary ports

Screenshot: Running Juice Shop container.

5. Final Penetration Testing

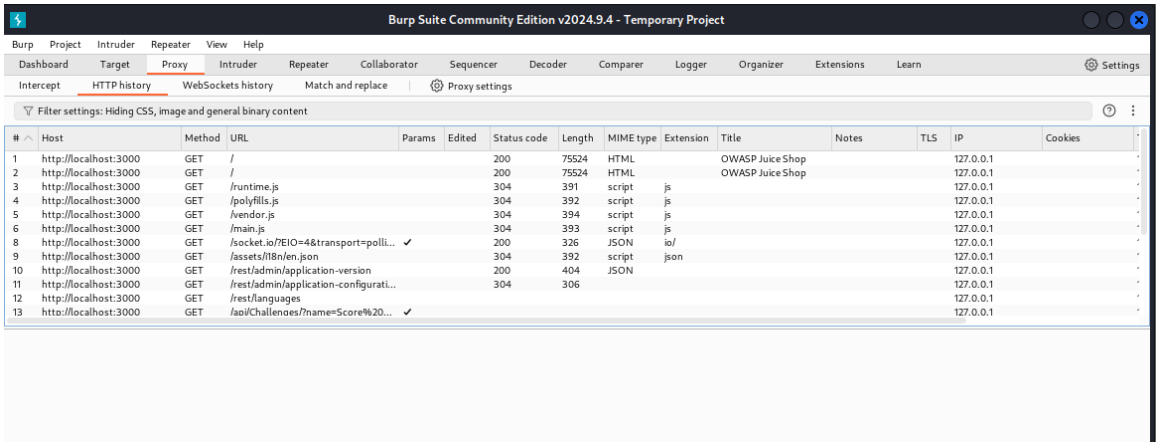
Manual penetration testing was conducted using **Burp Suite** to intercept and analyze HTTP traffic between the client and server.

Activities Performed:

- Intercepted HTTP requests
- Analyzed authentication behavior
- Observed session handling
- Evaluated request/response structure

This testing provided deeper insight into application behavior beyond automated scans.

Screenshot: HTTP request intercepted during manual penetration testing.



6. Security Improvements Recommended

The following actions are recommended before deploying any production-grade application:

- Implement a strong **Content Security Policy**
- Remove hardcoded secrets and private keys
- Regularly update dependencies
- Enable secure HTTP headers
- Perform continuous vulnerability scanning
- Apply least-privilege access control
- Encrypt sensitive data
- Conduct periodic penetration testing

7. Challenges Faced

During the security assessment, several technical challenges were encountered:

- Docker networking configuration issues

- Tool installation errors
- Connectivity problems during automated scans

These were resolved through systematic troubleshooting and configuration adjustments.

8. Conclusion

This week focused on performing comprehensive security audits and preparing the application for secure deployment.

By leveraging automated scanners and manual penetration testing tools, multiple vulnerabilities were successfully identified and analyzed.

The exercise reinforced the importance of integrating security testing throughout the software development lifecycle to minimize cyber risks and strengthen application resilience.

End of Report