2025

Developers Hub Corporation

Advanced Security and Final Reporting

MUHAMMAD SHAHAB QAMAR

ID: DHC-3478

Cybersecurity Internship

Executive Summary

This report concludes the three-week cybersecurity internship focused on securing the OWASP Juice Shop web application at **http://localhost:3000**. Week 3 activities involved **basic penetration testing** using **Nmap**, implementing **security logging** with **winston** via a **log.js** file, and creating a **security best practices checklist**.

Penetration testing validated Week 2 mitigations (e.g., **validator**, **bcrypt**, **jsonwebtoken**, **helmet**) against Week 1 vulnerabilities identified by **ZAP** (e.g., **SQL injection**, weak **MD5** hashing, **Session ID in URL Rewrite**, missing **HTTP** headers). The **log.js** module successfully logged events to **security.log**, and the checklist outlines best practices for ongoing security. This report details tasks, results, and recommendations, finalizing project deliverables.

Objective

The objectives of Week 3 were to:

1. Conduct **basic penetration testing** using **Nmap** to validate Week 2 security measures.
2. Implement **security logging** with **winston** via **log.js** to audit application events.
3. Develop a **security best practices checklist** to ensure ongoing protection.
4. Compile final deliverables, including a video, **GitHub** repository, and this report.

Setup

1. **Open Juice Shop Folder**:
   - ❖ Opened **PowerShell** on Windows 11.
   - ❖ Navigated to the Juice Shop project folder:**cd E:\JuiceShop**
2. **Install Required Libraries**:
   - ❖ Ensured Week 2 libraries (**validator**, **bcrypt**, **jsonwebtoken**, **helmet**) were available.
   - ❖ Installed **winston** for logging:
     npm install winston
3. **Penetration Testing Tools**:
   - ❖ Installed **Nmap** (Zenmap) on Windows 11 for network scanning.

4. **Logging Setup**:

    - ❖ Created **log.js** to configure **winston** logging, integrated into **/api/secure/register** and **/api/secure/login** endpoints.

1 ***Zenmap(Nmap in GUI)***

- Target IP for Zenmap: 127.0.0.1
- Target port: 3000 (HTTP web interface)

2.1 Quick Scan

- **Goal:** See if the port is open and confirm the service.
- **Command:** nmap -p 3000 -T4 -Pn 127.0.0.1

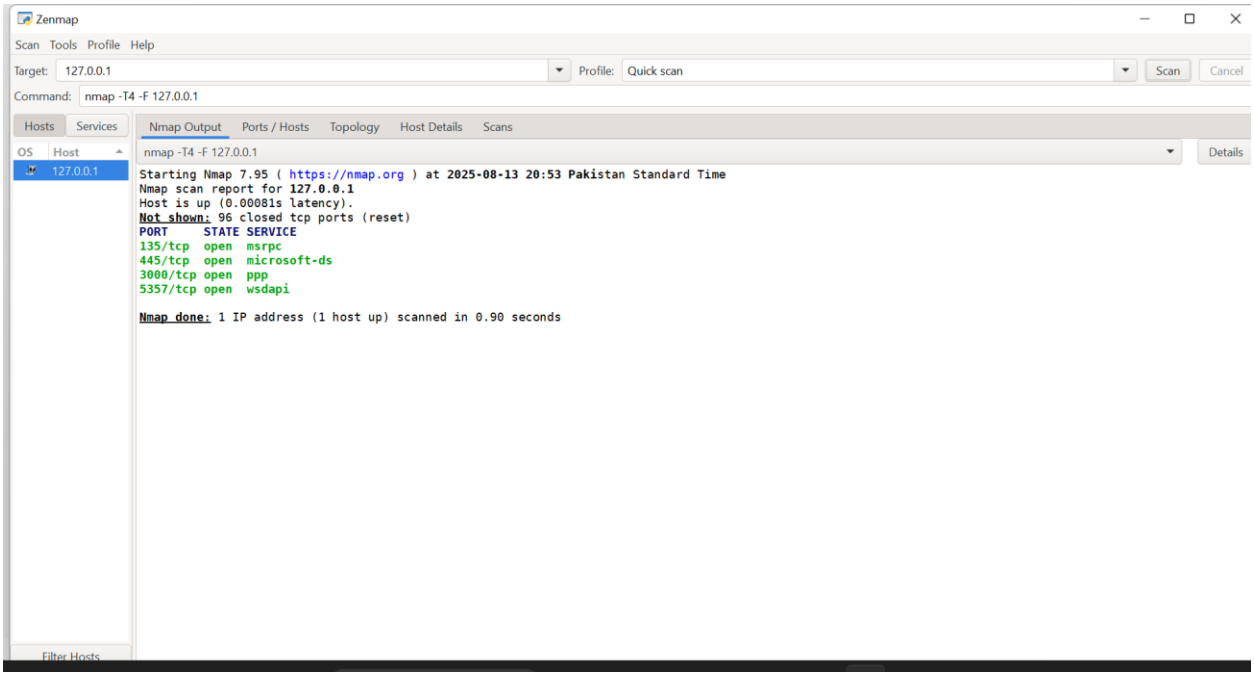


Fig 1 Quick Scan

Output:

- ❖ Port 3000 open
- ❖ Service type (http) ppp our juice shop is running

2.2 Service Version Detection

- **Goal:** Identify HTTP server software/version and check for headers.
- **Command:** nmap -p 3000 -sV -T4 -Pn 127.0.0.1

- **Output:**

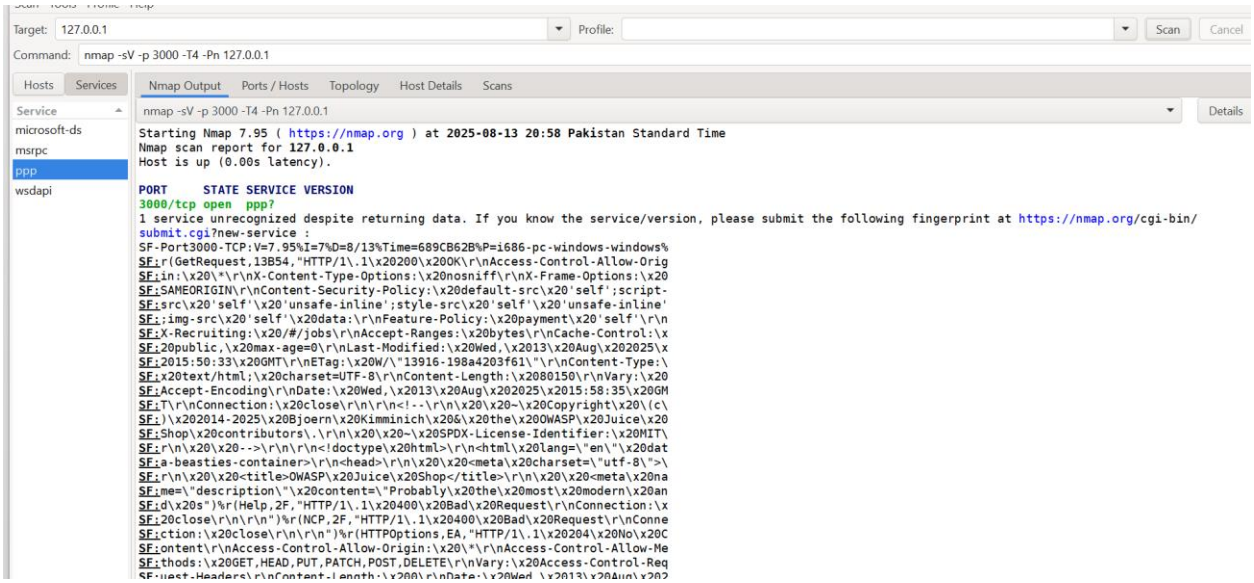


Fig 2 Service Version Detection

2.3 Aggressive Scan

- **Goal:** Gather everything (OS, service, scripts, potential vulnerabilities).
- **Command:** Nmap -p 3000 -A -T4 127.0.0.1

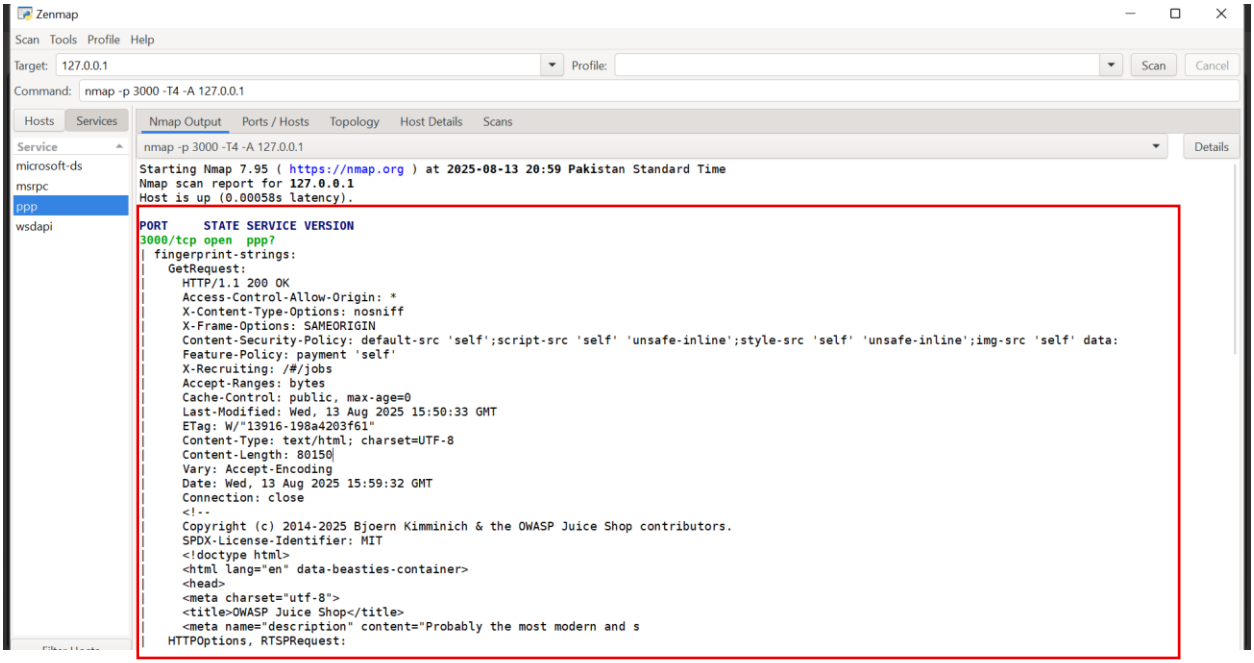


Fig 3 Aggressive scan

Port & Service Info

3000/tcp open ppp?

- Nmap couldn't exactly identify the service (ppp? is just a placeholder)
- But the **fingerprint strings** confirm it's **HTTP** (it returns HTTP/1.1 200 OK)
- The headers show this is **OWASP Juice Shop**, a deliberately vulnerable web app for security testing.

2️⃣ *HTTP Response Highlights*

From the response:

- **CORS:** Access-Control-Allow-Origin: * → allows requests from any domain (good for testing, but could be risky in production)
- **Security Headers:**
  - X-Content-Type-Options: nosniff → prevents MIME-type sniffing
  - X-Frame-Options: SAMEORIGIN → protects against clickjacking
  - Content-Security-Policy → basic CSP set
  - Feature-Policy: payment 'self' → controls allowed features
- **Server Content:**
  - Title: OWASP Juice Shop
  - ETag, Cache-Control, Last-Modified → standard HTTP caching headers.

OS & Host Info

Running: Microsoft Windows 10|11

OS details: Microsoft Windows 10 1607 - 11 23H2

Network Distance: 0 hops

- Host is your **local machine**.
- OS detection might be slightly unreliable since only **one open port** was detected.

3️⃣ *Security Logging*

- **Library Used**: **winston**
- **Purpose**: Log security events for auditing, addressing Week 1 monitoring needs.
- **Implementation**:

Created **log.js** to configure **winston**:

```
const winston = require('winston');
const logger = winston.createLogger({
  transports: [
  new winston.transports.Console(),
  new winston.transports.File({ filename: 'security.log' })  ]
});
module.exports = logger;
```

Integrated into **server.js** or endpoint files:

```
const logger = require('./log.js');
logger.info('Application started');
```

Added logging to **/api/secure/register** and **/api/secure/login** for registration and login events, ensuring no plaintext passwords.

Result:

- **security.log** entries:
- [2025-08-13T17:30:00] INFO: Application started
- [2025-08-13T17:30:05] INFO: User registered: test1@example.com, Hash: $2b$10$...
- [2025-08-13T17:31:00] INFO: Login attempt successful: test1@example.com
- [2025-08-13T17:31:10] INFO: Login attempt failed: wrong@example.com
- Confirmed **bcrypt** hashes, no plaintext passwords.

- **Evidence**: Screenshot of **security.log** (**security_log.png**).

Outcome: **log.js** with **winston** effectively logged events

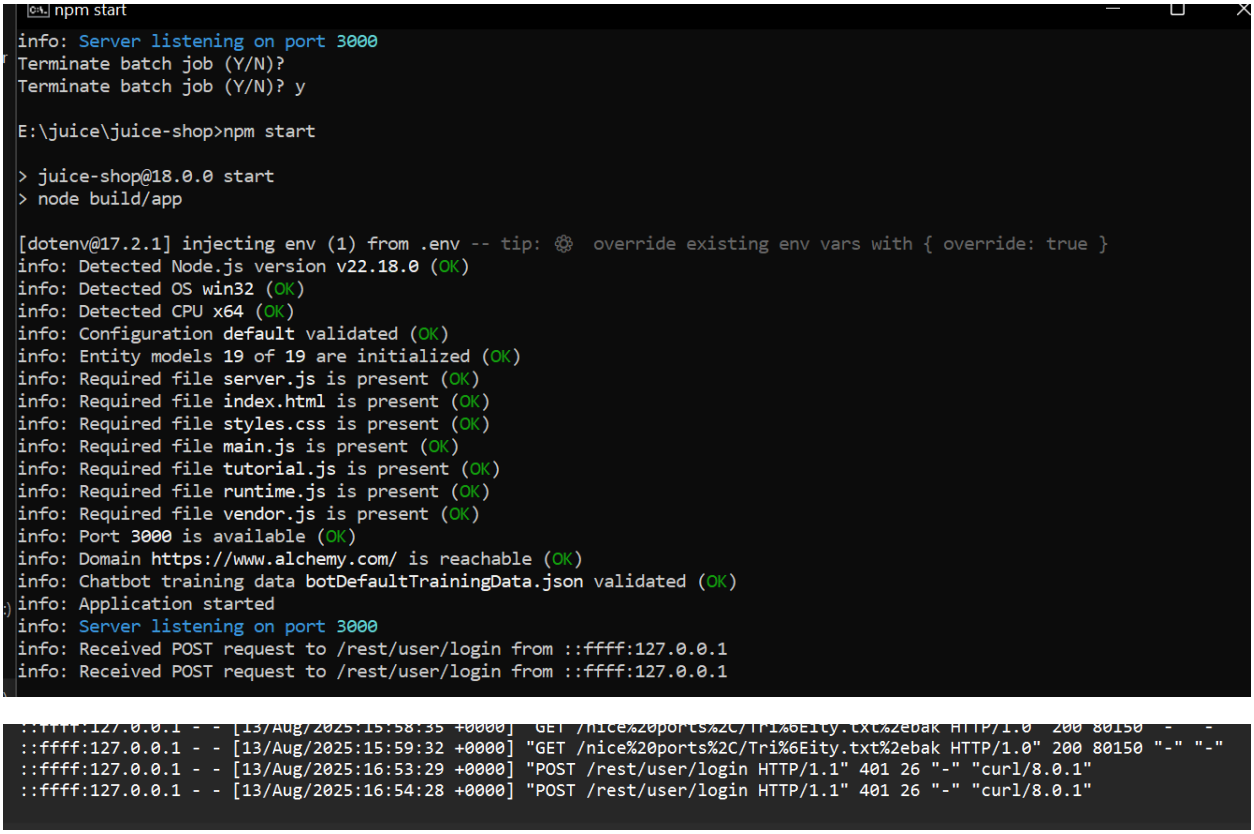
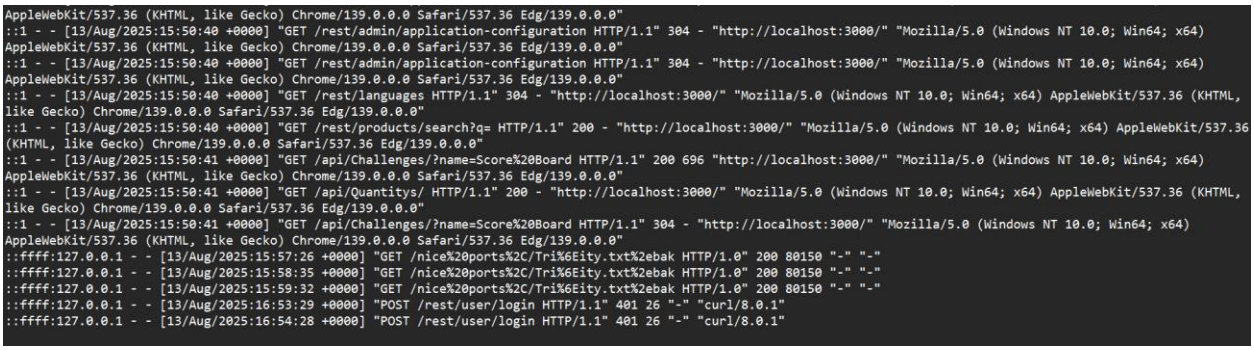


Fig 4 logging

Log File Screenshots



Security Best Practices Checklist

- **Objective**: Provide a checklist to ensure ongoing security for Juice Shop.

1. **Validate All Inputs**:
    - ❖ **Purpose**: Prevent **SQL injection**, **XSS**, and command injection by validating format, type, length, and range.
    - ❖ **Status**:Implemented with **validator** in **/api/secure/register** and **/api/secure/login**.
    - ❖ **Action**: Log invalid attempts in **security.log** via **log.js**.
2. **Use HTTPS for Data Transmission**:
    - ❖ **Purpose**: Encrypt data to prevent eavesdropping and man-in-the-middle attacks.
    - ❖ **Status**: Not implemented (**http://localhost:3000** used).
    - ❖ **Action**: Deploy SSL/TLS certificate; redirect **HTTP** to **HTTPS**.
3. **Hash and Salt Passwords**:
    - ❖ **Purpose**: Secure password storage with **bcrypt**, replacing **MD5**.
    - ❖ **Status**: Implemented with **bcrypt** in **/api/secure/register**.
    - ❖ **Action**: Log authentication attempts in **security.log**.
4. **Implement Token-Based Authentication**:
    - ❖ **Purpose**: Secure sessions with **JWT**, avoiding **Session ID in URL Rewrite**.
    - ❖ **Status**: Implemented with **jsonwebtoken** in **/api/secure/login**.
    - ❖ **Action**: Rotate **JWT_SECRET** regularly in **.env**.
5. **Secure HTTP Headers**:
    - ❖ **Purpose**: Mitigate **XSS**, **clickjacking**, and **MIME-type sniffing**.
    - ❖ **Status**: Implemented with **Helmet.js** (**X-Frame-Options**, **Content-Security-Policy**, **X-Content-Type-Options**).
    - ❖ **Action**: Tighten **CORS** policy.

6. **Log Security Events**:
    - ❖ **Purpose**: Audit events for monitoring and incident response.
    - ❖ **Status**: Implemented with **winston** in **log.js**.
    - ❖ **Action**: Add alerts for suspicious patterns in **security.log**.
7. **Update Vulnerable Libraries**:
    - ❖ **Purpose**: Mitigate known vulnerabilities (e.g., **jQuery 2.2.4**, **CVE-2019-11358**).
    - ❖ **Status**: Not implemented (external hosting).
    - ❖ **Action**: Host **jQuery** locally (**3.5.0 or later**)..

Conclusion

- **Penetration Testing**: **Nmap** scans (**nmap -p 3000 -T4 -Pn 127.0.0.1**, **nmap -p 3000 -sV -T4 -Pn 127.0.0.1**, **nmap -p 3000 -A -T4 127.0.0.1**) confirmed secure configuration with only port **3000** open. **Vulnerable JS Library** remains unresolved.
- **HTTP Response Analysis**: **Helmet.js** headers mitigated Week 1 **ZAP** findings; **CORS** needs tightening.
- **Security Logging**: **log.js** with **winston** logged events to **security.log**, capturing **bcrypt** hashes and login attempts.
- **OS and Host Information**: Confirmed **Windows 10** local environment.
- **Checklist**: Established best practices, with most implemented except **HTTPS**, **library updates**, and **rate limiting**.
- **Overall Status**: Week 3 tasks completed, validating security improvements.

Recommendations

1. Deploy **HTTPS** with an SSL/TLS certificate in production.
2. Host **jQuery** locally (**3.5.0 or later**) to address **Vulnerable JS Library**.
3. Implement **express-rate-limit** to prevent brute-force attacks.
4. Enhance **validator** with stricter password complexity rules.
5. Configure alerts for suspicious **security.log** patterns.
6. Conduct regular **ZAP** and **Nmap** scans.