# CYBERSECURITY INTERNSHIP – FINAL SECURITY ASSESSMENT & IMPLEMENTATION REPORT

**Student Name: Muhammad Shahid**

**Course: Cybersecurity Internship**

**Submission Date: 14 January 2026**

---

## 1. Introduction

This project was completed as part of my Cybersecurity Internship to gain hands-on experience in identifying and fixing common security issues in a web application. Instead of only studying theory, I worked on a real vulnerable application to understand how security testing and protection are done in practice.

The main goal of this task was to analyze a user-based web application, find security vulnerabilities, apply basic security fixes, and finally document everything clearly for final submission.

---

## 2. Application Used and Environment Setup

For this task, I used **OWASP Juice Shop**, which is a deliberately vulnerable web application designed for cybersecurity learning and practice.

### Tools and Software Used

- Git & GitHub
- OWASP Juice Shop (GitHub Repository)
- Node.js & npm
- VS Code
- PowerShell
- XAMPP
- Google Chrome Browser
- Browser Developer Tools

- OWASP ZAP
- Nmap
- Zap
- Microsoft JDK.17.0.17

### Application Setup Process

1. Cloned the OWASP Juice Shop project from GitHub using Git.
2. Opened the project folder in VS Code.
3. Using PowerShell, installed the required dependencies:

   ```
   npm install
   ```

4. Started the application:

   ```
   npm start
   ```

5. Accessed the application at http://localhost:3000

Explored sections: Signup, Login, Search, Cart, Profile, Admin Pages.

---

## 3. Week 1 – Security Assessment

### 3.1 Automated Vulnerability Scanning (OWASP ZAP)

The application was scanned using OWASP ZAP. Key issues found: - SQL Injection - Missing Content Security Policy (CSP) - Missing Anti-Clickjacking Header - Session ID appearing in URL - Vulnerable JavaScript libraries - Application error disclosure - Cross-domain misconfigurations - Missing X-Content-Type-Options header - Information disclosure through comments

### 3.2 Manual XSS Testing

Manual testing with input:

```
<script>alert('XSS')</script>
```

No alert popup appeared; ZAP still flagged potential risks due to missing CSP headers.

### 3.3 SQL Injection Testing

Login tested with: - Username: admin' OR '1'='1 - Password: admin' OR '1'='1

Application showed unusual behavior, indicating weak input validation.

### 3.4 Weak Password Handling

Passwords were intentionally stored insecurely in the app for learning purposes.

---

## 4. Week 2 – Implementing Security Measures

### 4.1 Input Validation

Using **Validator** library:

```
const validator = require('validator');
if (!validator.isEmail(email)) return
res.status(400).send('Invalid email');
```

### 4.2 Password Hashing

Using **Bcrypt**:

```
const bcrypt = require('bcrypt');
const hashedPassword = await bcrypt.hash(password, 10);
```

### 4.3 Authentication Improvement

Using **JWT**:

```
const jwt = require('jsonwebtoken');
const token = jwt.sign({ id: user._id }, 'your-secret-key');
res.send({ token });
```

### 4.4 Secure HTTP Headers

Using **Helmet.js**:

```
const helmet = require('helmet');
app.use(helmet());
```

---

# 5. Week 3 – Advanced Security Measures

## 5.1 Basic Penetration Testing (Nmap)

Port scanning using Nmap:

```
nmap 127.0.0.1
```

### 5.2 Logging Implementation (Winston)

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

### 5.3 Security Checklist

- Validate inputs
- Hash & salt passwords
- Use secure authentication
- Apply HTTP security headers
- Monitor logs regularly
- Use HTTPS

**Security Checklist (SECURITY_CHECKLIST.md) - Input Validation:**

Email validated, SQLi prevented, XSS tested - Authentication & Authorization: JWT auth implemented, token expiry, invalid login rejected - Password Security: bcrypt hashing, no plain-text passwords - Secure HTTP Headers: Helmet.js enabled - Logging & Monitoring: Winston logs configured, events logged - Testing Tools Used: OWASP ZAP, Browser Dev Tools, manual SQLi - Best Practices: input validation, password hashing, token-based auth, secure headers

---

## 6. Tools Summary

| Tool | Purpose |
|---|---|
| Git & GitHub | Source code management |
| VS Code | Code editing |
| PowerShell | Running commands |
| XAMPP | Local server support |
| OWASP Juice Shop | Vulnerable test application |
| OWASP ZAP | Automated vulnerability scanning |
| Nmap | Penetration testing |
| Browser Dev Tools | Manual testing |
| Validator | Input validation |
| Bcrypt | Password hashing |
| JWT | Authentication |
| Helmet.js | Security headers |
| Winston | Logging |

---

## 7. Conclusion

This internship task provided hands-on experience in web application security. Using OWASP Juice Shop, Nmap, OWASP ZAP, GitHub, VS Code, and PowerShell, I successfully performed vulnerability assessment, implemented security fixes, and documented all steps. This work increased my practical understanding of cybersecurity and prepared me for real-world security testing.

---

## 8.Final Submission Status

- Report: Complete