Week 1: Security Assessment — Application Understanding

Application Name: RetireEasy (NodeGoat-based mock web app)
Local URL: http://localhost:4000
Objective: Explore and understand the structure and functionality of the mock web application before testing vulnerabilities.
Tools Used: Browser (Firefox via ZAP proxy), OWASP ZAP 2.16.1
Test Environment: Localhost (Windows 10 x64, npm server)

DevelopersHub Cyber Security Internship

Week 1 — Security Assessment Report (NodeGoat)

Intern Name: Muhammad Rehan
Date: 12-11-2025
Repository Link: https://github.com/destro99912/nodegoat-week1-security-assessment

---

Objective

Perform a basic vulnerability assessment on the OWASP NodeGoat web application using OWASP ZAP and browser developer tools. The goal is to identify and analyze common web security flaws, including **Cross-Site Scripting (XSS)**, **Weak Password Storage**, and **Security Misconfigurations**, and to provide mitigation recommendations for improving the application's overall security posture.

Vulnerability 1 — Stored Cross-Site Scripting (XSS)

- **Location:** /profile → First Name field
- **Impact:** Injected JavaScript executes when profile data is viewed, allowing attackers to steal sessions or redirect users.

Steps to Reproduce

1. Login to the NodeGoat app.
2. Go to Profile → Edit Profile.
3. In the First Name field, enter:
   <script>alert('XSS');</script>
4. Click **Submit**.
5. An alert popup appears, confirming stored XSS execution.

Evidence Screenshots

1. 1_XSS_Alert_Popup.png — Alert execution in browser
2. 2_XSS_ZAP_Response.png — Payload found in HTML response

Mitigation Recommendations

- Encode all user inputs before rendering in HTML.
- Implement strict input validation and output sanitization on both client and server side.
- Use frameworks with auto-escaping features (e.g., React, Angular).
- Add a Content Security Policy (CSP).

Tools Used

- OWASP ZAP 2.16.1
- NodeGoat Web App (Localhost 4000)
- Firefox Browser

Vulnerability 2 — Weak Password Storage

Location:
User Signup → /signup POST request

Description:
During OWASP ZAP analysis, the registration request showed the **user password transmitted and logged in plaintext**. This indicates that passwords are not being hashed or encrypted before storage or transfer.

Evidence:

- 4_WeakPassword_Request.png — Signup POST request showing password in HTTP body
- 4_WeakPassword_Response.png — Response confirming successful submission without encryption

Impact:
Attackers intercepting network traffic (via MITM or proxy) could obtain raw credentials, leading to **account compromise** or reuse attacks on other systems.

Mitigation Recommendations:

- Use secure password hashing (bcrypt, Argon2).
- Force HTTPS for all requests using TLS 1.2+.
- Implement password strength and salting policy.
- Never log or store plaintext passwords.

Vulnerability 3 — Security Misconfigurations

Location:
Detected globally across application responses via OWASP ZAP Active Scan.

Description:
The OWASP ZAP scan revealed several configuration weaknesses within the NodeGoat web application. These included missing security headers, lack of CSRF protection, and exposure of internal framework information through response headers.

Evidence:

- 5_Security_Misconfiguration.png — ZAP scan alert summary showing missing CSP, CSRF tokens, and disclosure of framework information.

Impact:
These weaknesses increase the risk of:

- **Clickjacking attacks** (due to missing X-Frame-Options)
- **Cross-Site Request Forgery (CSRF)** attacks (due to missing tokens)

- **Information disclosure** about backend technologies
- **Reduced resistance** against modern browser-based attacks

Mitigation Recommendations:

- Implement a **Content Security Policy (CSP)** to control browser-side behavior.
- Add **CSRF tokens** in all forms and verify them server-side.
- Disable the X-Powered-By header to prevent framework exposure.
- Enforce standard security headers:
    - Strict-Transport-Security
    - X-Frame-Options: DENY
    - X-Content-Type-Options: nosniff
    - Referrer-Policy: no-referrer

- **Conclusion**
  The security assessment of the OWASP NodeGoat application successfully identified three key vulnerabilities: **Stored Cross-Site Scripting (XSS)**, **Weak Password Storage**, and **Security Misconfigurations**. These issues demonstrate common flaws in web applications that could lead to unauthorized access, credential theft, or exploitation of client-side weaknesses. Implementing the recommended mitigations—such as input validation, secure password hashing, and proper security headers—will significantly enhance the application's resilience against attacks and improve its overall security posture.