🛡️ Task 07: Web Application Vulnerability Testing

Task Name: Web Application Vulnerability Testing

By : NROUPSINH GOHIL

🔹 Objective

The objective of this task is to understand common web application vulnerabilities listed in the OWASP Top 10 and perform basic vulnerability testing such as SQL Injection and Cross-Site Scripting (XSS) on a deliberately vulnerable web application.

🔹 Environment & Tools

- **Target Application:** OWASP Juice Shop
- **Testing Method:** Manual testing using browser
- **Reason for Manual Testing:**
  For this task, vulnerabilities were tested manually by injecting payloads directly into application input fields to understand vulnerability behavior. Burp Suite is commonly used for interception, but manual testing is a valid initial approach.

🔹 Vulnerability 1: SQL Injection (Authentication Bypass)

🔴 Vulnerability Name

SQL Injection (Authentication Bypass)

🔴 Vulnerable Component

Login functionality of OWASP Juice Shop

🔴 Payload Used

```
' OR 1=1--
```

🔴 Description

SQL Injection occurs when user input is directly included in SQL queries without proper validation. In this case, the login form accepted malicious SQL input, allowing authentication to be bypassed without a valid username or password.

🔴 Impact

- Unauthorized access to user accounts
- Authentication bypass
- Potential data leakage or modification

🔴 Evidence

- Screenshot showing SQL payload entered in login form
- Screenshot showing successful login without valid credentials

🛡 **Mitigation**

- Use parameterized queries (prepared statements)
- Validate and sanitize all user inputs
- Avoid direct concatenation of user input into SQL queries

🔹 **Vulnerability 2: Cross-Site Scripting (XSS)**

🔴 **Vulnerability Name**

Cross-Site Scripting (Reflected XSS)

🔴 **Vulnerable Component**

Search functionality of OWASP Juice Shop

🔴 **Payload Used**

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

🔴 **Description**

Cross-Site Scripting (XSS) occurs when an application fails to properly sanitize user input, allowing attackers to inject malicious JavaScript code. In this case, injected JavaScript was executed in the browser through the search feature.

🔴 **Impact**

- Execution of malicious scripts in user browser
- Session hijacking
- Phishing and defacement attacks

🛡 **Mitigation**

- Validate and sanitize all user inputs
- Encode output properly
- Implement Content Security Policy (CSP)

🔹 **Observations**

- The application was vulnerable to basic SQL Injection and XSS attacks.

- Public Juice Shop instances may become unstable after attack attempts, indicating lack of proper error handling.
- Manual testing is effective for identifying common web vulnerabilities at an initial level.

🔹 Conclusion

This task provided hands-on experience in identifying common web application vulnerabilities. SQL Injection and Cross-Site Scripting were successfully exploited, demonstrating the importance of secure coding practices, proper input validation, and adherence to OWASP Top 10 guidelines.

🔹 Final Outcome

- Improved understanding of OWASP Top 10 vulnerabilities
- Practical experience in web application security testing
- Awareness of mitigation techniques to prevent web attacks