Future Interns Cyber Security Internship — Task 1 Report

Prepared by: Shubham Gautam (RedVortex)

Future Interns Cyber Security Internship — Task 1

Vulnerability Report — SQL Injection in id Parameter

Title:
SQL Injection in id Parameter

Severity:
High

Description:
The id parameter in the target application (DVWA SQL Injection module) fails to properly sanitize and validate user-sup
This allows an attacker to inject arbitrary SQL statements into the query, potentially bypassing authentication, retrievin

Proof of Concept (PoC):
Steps to Reproduce:
1. Navigate to the SQL Injection page in DVWA (http://localhost/DVWA/vulnerabilities/sqli/).
2. Enter the following payload into the id input field:
   1' OR '1'='1
3. Observe that the application returns all user records from the database, indicating the query logic was altered.

Captured Request:
GET /DVWA/vulnerabilities/sqli/?id=1%27+OR+%271%27%3D%271&Submit=Submit HTTP/1.1

SQLMap Confirmation:
sqlmap -r request.txt --batch --dbs
[INFO] the back-end DBMS is MySQL
available databases:
[*] dvwa
[*] information_schema

Impact:
An attacker could:
- Read or dump the entire database, including usernames, password hashes, and other sensitive data.
- Modify or delete data.
- Execute administrative operations on the database.
- Potentially escalate the attack to the underlying server if combined with other vulnerabilities.

Fix / Recommendation:
- Use Parameterized Queries (Prepared Statements) to separate SQL code from data.
- Implement strict input validation and reject unexpected input types.
- Apply least privilege to the database account used by the application.
- Implement Web Application Firewall (WAF) rules to detect and block injection patterns.

References:
- OWASP SQL Injection Prevention Cheat Sheet: https://owasp.org/www-community/attacks/SQL_Injection
- MySQL Prepared Statements: https://dev.mysql.com/doc/refman/8.0/en/sql-prepared-statements.html

Vulnerability Report — Reflected Cross-Site Scripting (XSS) in name

Title:
Reflected Cross-Site Scripting (XSS) in name Parameter

Severity:
High

Description:
The name parameter in the target application (DVWA Reflected XSS module) does not properly sanitize user input bef
This allows an attacker to inject malicious JavaScript code into the web page, which will be executed in the victim's bro
Such attacks can be used to steal cookies, hijack sessions, or perform unauthorized actions on behalf of the user.

Proof of Concept (PoC):
Steps to Reproduce:
1. Navigate to the Reflected XSS page in DVWA (http://localhost/DVWA/vulnerabilities/xss_r/).
2. In the input field labeled name, enter the following payload:
   <script>alert('XSS')</script>
3. Click Submit.
4. Observe that a JavaScript alert box with the text XSS appears, confirming code execution in the browser.

Captured Request:
GET /DVWA/vulnerabilities/xss_r/?name=%3Cscript%3Ealert('XSS')%3C/script%3E&Submit=Submit HTTP/1.1

ZAP Confirmation:
- Ran OWASP ZAP Active Scan against http://localhost/DVWA/vulnerabilities/xss_r/
- Detected Reflected XSS vulnerability on the name parameter.

Impact:
An attacker could:
- Steal session cookies and impersonate legitimate users.
- Deface the web application.
- Redirect users to malicious websites.
- Execute keylogging scripts to capture sensitive information.

Fix / Recommendation:
- Apply HTML entity encoding to all user input before rendering in the browser.
- Use frameworks or libraries that automatically escape dynamic content.
- Implement Content Security Policy (CSP) to reduce the risk of script execution.
- Perform strict server-side input validation and reject any script tags or dangerous characters.

References:
- OWASP XSS Prevention Cheat Sheet: https://owasp.org/www-community/attacks/xss/
- MDN HTML Sanitization: https://developer.mozilla.org/en-US/docs/Learn/Forms/Security