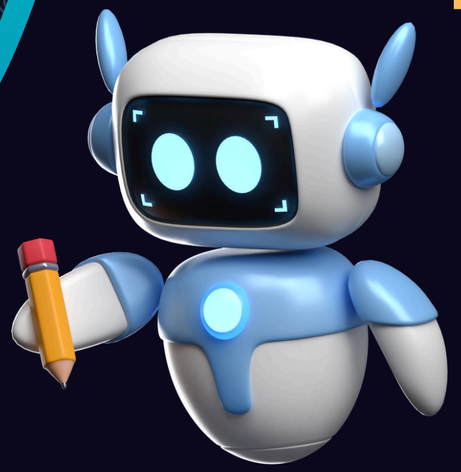
TALENTO TECH

TIC

REGIÓN 3

CAUCA – NARIÑO

LESSON 5



Universidad Tecnológica de Pereira | faceIT

Lesson 5: Reading
Identifying Vulnerabilities and Fixing Them: A Practical Guide

Web applications are a crucial part of modern life, but they are also frequent targets for cyberattacks. Understanding how to identify vulnerabilities and fix them is essential for anyone working in cybersecurity or web development. This guide explains how to detect vulnerabilities using tools like OWASP ZAP, and offers simple solutions such as data validation and secure queries to keep applications secure.

1. How to Identify Vulnerabilities

Vulnerabilities in web applications can expose sensitive data or allow attackers to take control of systems. Identifying these weaknesses is the first step toward securing an application.

Common Vulnerabilities to Watch For:

- SQL Injection (SQLi):Attackers manipulate SQL queries to access or alter database information.
- Cross-Site Scripting (XSS): Malicious scripts are injected into web pages to attack users.
- Insecure Direct Object References (IDOR): Users can access data or functionality they shouldn't by manipulating requests.
-

Using OWASP ZAP to Detect Vulnerabilities

OWASP ZAP (Zed Attack Proxy) is a free, open-source tool that automates vulnerability detection for web applications. It is user-friendly and suitable for both beginners and experts.

How OWASP ZAP Works:

1. Set Up a Proxy: ZAP acts as a middleman between your browser and the application, intercepting and analyzing traffic.
2. Scan the Application: ZAP can perform automated scans to find issues like missing security headers, SQLi, or XSS vulnerabilities.
3. Analyze the Results: ZAP provides a report listing the vulnerabilities it found, along with severity ratings and recommendations for fixing them.

Example: Suppose you're testing an online store. ZAP might detect that the site's search functionality is vulnerable to SQL Injection. It will highlight this issue in its interface and show how the input field can be exploited.

2. Simple Solutions to Fix Common Vulnerabilities

Once vulnerabilities are identified, the next step is to fix them. Here are simple yet effective solutions for two common types of vulnerabilities:

A. Data Validation: Data validation ensures that user inputs are checked and cleaned before being processed.

- What It Does: Verifies that inputs match expected formats (e.g., numbers for a phone number field).
- How to Implement It:
  1. Use libraries like Validator.jsin JavaScript or Flask-WTF in Python.
  2. Reject unexpected or potentially dangerous inputs.

Example: For a contact form that asks for an email address, ensure the input matches a standard email format using regular expressions. This prevents malicious scripts from being submitted.

B. Secure Queries: Using prepared statements and parameterized queries protects against SQL Injection by treating user input as data, not executable code.

- **What It Does:** Prevents attackers from altering database queries.
- **How to Implement It:** Use secure database interaction methods provided by your programming language.

Example in Python:

```
cursor.execute("SELECT * FROM users WHERE username = %s", (user_input,))
```

This code ensures that any input provided by the user is treated as a value, not as part of the SQL command.

3. Demonstrating Vulnerability Detection with OWASP ZAP

To better understand how tools like ZAP work, let's walk through a real-world demonstration.

Scenario:

You are testing the login functionality of a web application. An attacker might use SQL Injection to bypass authentication.

Steps with OWASP ZAP:

1. Intercept Login Requests: Use ZAP to capture traffic between your browser and the application when submitting login credentials.
2. Inject Malicious Input: Test the vulnerability by sending inputs like admin' OR '1'='1 in the username field.
3. Analyze Results: If ZAP detects that the application allows this input to bypass authentication, it will flag it as a SQL Injection vulnerability.
4. Recommendations: ZAP provides suggestions, such as using prepared statements, to fix the issue.

Key Takeaways

- Proactive Detection: Regularly scan your web applications using tools like OWASP ZAP to identify vulnerabilities.
- Best Practices: Apply simple fixes like data validation and secure queries to reduce risks.
- Stay Updated: Cybersecurity threats evolve constantly. Regularly update your tools, techniques, and knowledge.