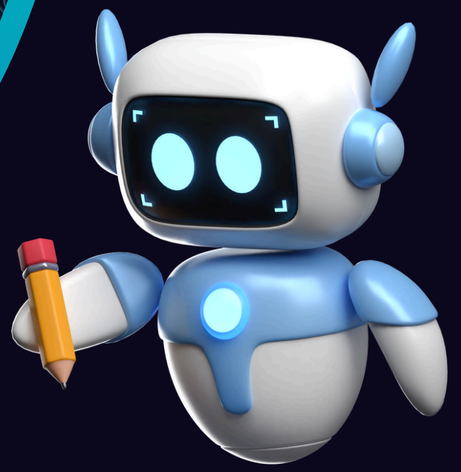
TALENTO TECH

TIC

REGIÓN 3

CAUCA – NARIÑO

LESSON 1



Universidad Tecnológica de Pereira | faceIT

Lesson1: Identifying and Mitigating Vulnerabilities based on OWASP Top 10.

1)Idiom of the day

Digital footprint: The trail of data left behind by users online.

Example: "Be careful about your digital footprint—it can be used against you."

2)ACTIVITY 1: Warmup activity: "Spot the Vulnerability"

Objective: Engage participants in identifying potential vulnerabilities and raise awareness of the OWASP Top 10 as a framework for mitigating them.

1. **Instructions**

- Create a short, fictional scenario about a poorly secured application. Include 2-3 examples of vulnerabilities from the OWASP Top 10 (e.g., SQL injection, weak authentication, or insecure APIs).
- Prepare a slide or handout listing the OWASP Top 10 categories briefly.

2. Scenario Example 1: “Imagine you are reviewing the security of an online shopping website. The site allows users to log in, view products, and make purchases. While testing, you notice the following issues:

- Users can input anything into the search bar, and the website returns an error with detailed database information.
- The admin login page is publicly accessible, and it accepts simple passwords like ‘12345.’
- The checkout process sends credit card data in plain text over the internet.
  What vulnerabilities can you identify?”

Scenario Example 2: An online store allows customers to browse products, register accounts, and complete purchases.

- The search bar allows users to type any query, but entering '; DROP TABLE products;-- causes the website to crash, indicating no input sanitization.
- The admin portal uses a default username (admin) and a weak password (password123) that hasn’t been updated since deployment.
- During checkout, customers’ credit card information is sent in plain text over HTTP instead of HTTPS.

Scenario example 3: A new social media app encourages users to share photos and connect with friends.

- The comment section doesn't sanitize input, allowing users to post malicious JavaScript, which executes when others view the comment.
- The API endpoint /user/profile returns all user details, including email, phone number, and address, even when accessed by unauthenticated users.
- The application's debug mode is enabled in production, exposing sensitive stack traces when errors occur.
-

Scenario example 4: A fitness tracker app syncs user data with a backend API to show stats and provide personalized recommendations.

- The API endpoint /update-stats allows unauthenticated requests, letting attackers alter any user's data by submitting arbitrary values.
- The endpoint /admin/dashboard can be accessed by any logged-in user without proper role verification.
- The app uses an outdated version of an open-source library with publicly documented vulnerabilities.

.

3) Group Discussion: Ask participants to work in pairs or small groups to:

- Identify the vulnerabilities in the scenario.
- Relate them to any OWASP Top 10 category (if familiar).
- Discuss why these vulnerabilities are problematic.

4) Debrief: Invite groups to share their findings. Highlight key points and connect their observations to specific OWASP Top 10 categories, such as:

- SQL injection (e.g., the search bar issue).
- Weak authentication controls (e.g., simple admin passwords).
- Insecure data transmission (e.g., unencrypted credit card data).

Here is a brief listing of the OWASP Top 10 categories:

1. Injection: Occurs when untrusted data is sent to an interpreter, leading to the execution of malicious commands (e.g., SQL injection).
2. Broken Authentication: Insecure authentication mechanisms that allow attackers to impersonate users or gain unauthorized access.
3. Sensitive Data Exposure: Inadequate protection of sensitive data, such as passwords, credit card numbers, or personal information.
4. XML External Entities (XXE): Vulnerabilities in XML parsers that allow attackers to interfere with the processing of XML documents, potentially exposing internal files or services.
5. Broken Access Control: Flaws that allow unauthorized users to access resources or perform actions outside their intended permissions.
6. Security Misconfiguration: Insecure configurations in applications, servers, or databases, often due to default settings or incomplete setups.
7. Cross-Site Scripting (XSS): Attacks where malicious scripts are injected into web pages viewed by other users, enabling data theft or unauthorized actions.
8. Insecure Deserialization: Issues related to deserializing data that can lead to remote code execution, authentication bypass, or other exploits.
9. Using Components with Known Vulnerabilities: Using outdated or vulnerable software components, such as libraries or frameworks, which attackers can exploit.
10. Insufficient Logging & Monitoring: A lack of proper logging and monitoring makes it difficult to detect, respond to, and investigate security breaches.