SQL Injection (SQLi)

1. Definición

SQL Injection é unha falla que permite a un atacante **inxeccionar e executar comandos SQL maliciosos** nunha base de datos a través dunha aplicación web. Isto ocorre cando a aplicación constrúe consultas SQL **sen validar ou sanear correctamente a entrada do usuario**.

En termos simples: se unha aplicación pega directamente datos dun usuario nunha consulta SQL, un atacante pode engadir código SQL extra que non debería executarse.

2. Como funciona

Exemplo típico en PHP (sen protección):

```
$user = $_POST['username'];
$pass = $_POST['password'];

$query = "SELECT * FROM usuarios WHERE username='$user' AND password='$pass'";
$result = mysqli_query($conn, $query);
```

Se o atacante introduce:

```
username: ' OR '1'='1
password: ' OR '1'='1
```

A consulta construíase así:

```
SELECT * FROM usuarios WHERE username='' OR '1'='1' AND password='' OR '1'='1';
```

O resultado é **sempre verdadeiro**, polo que o atacante podería entrar sen credenciais válidas.

3. Tipos principais de SQL Injection

1. **In-band SQLi** (o máis común)
    - O atacante recibe os resultados directamente da mesma canle usada para o ataque.
    - Subtipos:
        - **Error-based SQLi**: usa mensaxes de erro da base de datos para obter información.
        - **Union-based SQLi**: usa `UNION` para combinar resultados e obter datos arbitrarios.
2. **Inferential SQLi (Blind SQLi)**
    - Non se reciben erros visibles; o atacante deduce a información facendo preguntas á base de datos e observando respostas ou tempos de carga.
3. **Out-of-band SQLi**
    - O atacante usa funcionalidades da base de datos para enviar datos a un servidor que controla, por exemplo, vía DNS ou HTTP.

4. Consecuencias dun SQL Injection

- Roubo de datos sensibles (usuarios, contrasinais, correos, etc.).
- Modificación ou borrado de información.
- Execución de comandos no sistema operativo (en casos avanzados).
- Compromiso completo da aplicación e da base de datos.

5. Prevención

- **Usar consultas preparadas (prepared statements)** con parámetros en lugar de concatenar strings.
- **Sanear e validar toda entrada do usuario**.
- Limitar privilexios da base de datos: nunca usar un usuario con todos os permisos.
- Configurar alertas de seguridade e revisar logs.