XSS (Cross-Site Scripting)

1. Definición

XSS (Cross-Site Scripting) é un tipo de vulnerabilidade que permite a un atacante inxectar código malicioso (normalmente JavaScript) nunha páxina web que logo é visualizada por outros usuarios. O nome "cross-site" ven de que o ataque afecta á experiencia de usuarios que confían na páxina web “legítima”.

En termos simples: o atacante consegue que o navegador doutra persoa execute código que non debería.

2. Tipos principais de XSS

1. **Reflected XSS (XSS reflectido)**
    - O código malicioso forma parte da solicitude HTTP (URL, parámetros, formularios).
    - Non se almacena no servidor; simplemente se “reflexa” na resposta inmediata.
    - Exemplo: unha páxina de busca que amosa o termo buscado sen limpar os caracteres especiais. Se alguén pon `<script>alert('Hacked')</script>` como termo de busca, o script execútase no navegador do usuario.
2. **Stored XSS (XSS almacenado/persistente)**
    - O código malicioso almacénase no servidor, por exemplo nunha base de datos ou nunha entrada de comentarios.
    - Cada vez que outro usuario visita a páxina, o script malicioso se executa automaticamente.
    - Exemplo: un comentario de blog con `<img src=x onerror=alert('Hacked')>`.
3. **DOM-based XSS**
    - A vulnerabilidade non está no servidor, senón no **DOM** (Document Object Model) do lado do cliente.
    - O JavaScript da páxina procesa datos do usuario de forma insegura e executa código malicioso.
    - Exemplo: unha función JavaScript que toma o `window.location.hash` e insértalo directamente no HTML sen escapar caracteres.

3. Consecuencias dun XSS

Un XSS pode ser moi perigoso dependendo do acceso que o atacante consiga:

- Roubo de **cookies** ou sesións, permitindo secuestrar contas.
- Redirección a páxinas de phishing.

- Executar accións en nome do usuario (por exemplo, enviar mensaxes, comprar algo).
- Modificación do contido mostrado ao usuario.
- Instalación de malware localmente (menos frecuente, pero posible con combinación de técnicas).

4. Como se prevén

- **Escapar correctamente os datos** antes de inserilos no HTML:
    - Por exemplo, transformar `<` en `<` e `>` en `>`.
- **Usar frameworks que xestionen escaping automaticamente**, como React, Angular ou Vue.
- **Validar e sanear todos os datos do usuario** no servidor.
- **Configurar políticas de seguridade no navegador (CSP)** que restrinxan que scripts se poden executar.
- **Evitar execución de código inline** e inxección de HTML arbitrario.

Un **XSS é unha falla de seguridade que permite executar código malicioso no navegador dun usuario confiado**, e pode ser reflectido, almacenado ou baseado no DOM. A prevención céntrase en escapar, sanear e controlar rigorosamente os datos de entrada e saída.