

L U M B R E



Cómo rompen tu app: inyección SQL, XSS y CSRF explicados con dibujos

Materia: Desarrollo de Software con IA

07/10/2026

Antes de empezar

Para defender una app no hace falta ser hacker: hace falta entender tres trampas clásicas con las que la IA rellena código sin pensarlo. La inyección SQL engaña a tu base de datos, el XSS mete scripts ajenos en las páginas de tus usuarios y el CSRF hace que el navegador actúe sin permiso. Aquí los ves en dibujos y aprendes la defensa que debes exigir en cada prompt.

QUÉ APRENDERÁS

- Describir los tres ataques clásicos (inyección SQL, XSS, CSRF), su efecto y su defensa estándar.

Conoce las tres trampas que la IA olvida cerrar



La seguridad no es un feature: es capas que se suman. Inyección, XSS, secretos, hash: aquí aprendes a exigir auditorías y a no tragar con la primera respuesta.

Cuando pides «pon un buscador» o «muestra los comentarios», la IA escribe lo que funciona, no necesariamente lo que es seguro. Tres ataques son los que se cuelan una y otra vez. No necesitas explotarlos: necesitas **reconocerlos y exigir su defensa**. Los verás como dibujos, no como código de hacker.

Inyección SQL: el dato que se vuelve orden

En un buscador mal hecho, lo que escribe el usuario se pega directamente a la consulta. Un listo puede escribir texto que la base de datos lee como **instrucción**, no como dato: y se lleva todo.

El mismo texto: dato vs orden

```
BUSCO: un nombre en el buscador de clientes\n\nSi el código PEGA tu texto a la consulta:\n SELECT * FROM clientes WHERE nombre = '[lo que escribi]'\n\nEscribo: x' OR '1'='1\nY la consulta se convierte en:\n ... WHERE nombre = 'x' OR '1'='1'\n-> devuelve TODOS los clientes\n\nDEFENSA: consultas parametrizadas (el dato nunca se mezcla con la orden).
```

XSS: script ajeno en tu página

Si guardas y muestras comentarios sin limpiarlos, alguien puede «comentar» un pedazo de JavaScript. Tu navegador lo ejecutará **creyendo que es tuyo**, y puede robar sesiones.

XSS en una línea

Un usuario escribe como comentario un trozo de JavaScript que roba la sesión. Otros lo ven; su navegador lo ejecuta creyendo que es parte de tu página. La defensa: **escapar/sanitizar** todo lo que se muestra y nunca tratar la entrada como código.

CSRF: el navegador traicionado

Una página maliciosa hace que tu navegador, ya conectado a tu banco, dispare una acción sin que tú la pidas (un «transferir dinero»). La defensa es un **token anti-CSRF** que solo tu formulario real conoce.

Para reforzar la defensa SQL

Prueba

1. La defensa estándar contra la inyección SQL es:

- ☐ Ocultar el buscador en el frontend.
- ☐ Usar consultas parametrizadas, donde el dato nunca se concatena a la orden.
- ☐ Acortar la contraseña del usuario.
- ☐ Reiniciar el servidor cada día.

2. Mostrar los comentarios del usuario tal cual fueron escritos es inocuo.

- ☐ Verdadero
- ☐ Falso

Detecta el código inseguro

ACTIVIDAD
INTERACTIVA
DISPONIBLE EN
LUMBRE

3. Lees la explicación de una IA sobre cómo hizo tres funciones. Señala la que introduce una vulnerabilidad:



4. Revisa tu proyecto: ¿dónde metes texto del usuario en una consulta, dónde muestras texto ajeno y dónde hay acciones sensibles? Para cada uno, escribe la defensa que le exigirás a la IA.

La seguridad no es un módulo aparte: son decisiones en cada punto donde entra o sale texto del usuario.

Referencia curricular

Materia:	Desarrollo de Software con IA
-----------------	-------------------------------

Versi\u00f3n interactiva completa (con actividades, v\u00eddeos y autoevaluaci\u00f3n):



<https://empresa.lumbre.lat/object/fsi-m11-oa01>

Material educativo generado desde Lumbre · lumbre.lat