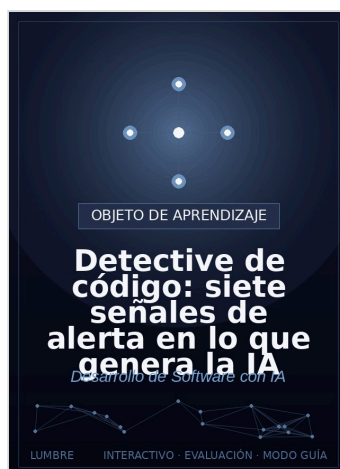


L U M B R E



Detective de código: siete señales de alerta en lo que genera la IA

Materia: Desarrollo de Software con IA

07/10/2026

Antes de empezar

No necesitas leer código como programador para detectar código peligroso: necesitas las siete señales. Secretos escritos a la vista, SQL armado con más, validaciones delegadas al navegador, permisos en abierto, errores que se tragan en silencio, CORS cruzado de cualquier lado y datos personales que se van al log. Este OA entrena tu ojo con un fragmento real de código generado —las siete en veinte líneas— y te da la rutina de escaneo de dos minutos que las encuentra sin saber sintaxis.

QUÉ APRENDERÁS

- Aplicar una lista de señales (secretos en texto, SQL concatenado, permisos abiertos, sin validación) a código ajeno.

Detective no es programador: es quien trae la lista

La IA te devolvió el login y funciona. Pero "funciona" no es "es seguro": el código generado puede funcionar exactamente porque hace las cosas mal —guardar la contraseña en texto plano para poder compararla, quitar el permiso para que no salga error—. Tu ventaja como builder: no necesitas entender cada línea; necesitas escanear con **lista de señales**, como un inspector de vivienda que no construyó la casa pero sabe dónde huele a humedad.

ACTIVIDAD INTERACTIVA DISPONIBLE
EN LUMBRE

1. La rutina de escaneo: cuatro pasadas



Las siete señales

Las siete las puedes buscar sin leer código: son palabras, símbolos y ausencias. Un escaneo de dos minutos con la lista encuentra más que una hora de lectura intuitiva.

#	Señal	Cómo se ve	Por qué duele
1	Secretos en el código	const API_KEY = "sk-abc123..." a la vista en un archivo	viaja al repo, a la copia, al navegador: filtración eterna
2	SQL cosido con más	WHERE email = ' + correo + '	la puerta clásica del SQL injection (M10)

#	Señal	Cómo se ve	Por qué duele
3	Validación solo del lado de afuera	el required está en el formulario y no en el servidor	cualquiera llega al servidor con una petición directa
4	Permisos en abierto	ruta de borrado/edición sin chequear quién es	el usuario 2 borra al usuario 1 "porque funciona"
5	Errores tragados	catch { } vacío: si algo falla, nadie se entera jamás	el fallo silencioso es el bug que nunca se arregla
6	CORS con estrella en algo con sesión	access-allow-origin para todo el mundo en rutas privadas	invita a otros sitios a montar tu API (M10-OA02)
7	Datos personales al log	el log imprime correo, teléfono o la petición completa	la bitácora se vuelve la filtración que no atacaron

Escena del crimen: veinte líneas, cinco señales

Un solo pantallazo: cinco de las siete señales activas

```
const API_KEY = "sk-prod-9f2c8a1b..."; // señal 1: secreto en el
codigo\n\napp.get("/busca", (req, res) => {\n  const q = req.query.q;\n  // señal
2: SQL cosido con +\n  const sql = "SELECT * FROM clientes WHERE nombre LIKE '%" +
q + "%'";\n  db.run(sql, (err, rows) => {\n    if (err) { }\n  })\n  // señal 5: error tragado\n  console.log("busqueda:", q, rows); //
señal 7: datos al log\n  res.json(rows);\n});\n\n// señal 6: cors
estrella en ruta con sesión\napp.use((req, res, next) => {\n
res.setHeader("Access-Control-Allow-Origin", "*");\n  next();\n});\n\n// señal 4:
la ruta de arriba no chequeo quien es: cualquiera busca todos los clientes
```

Para reforzar

Prueba

2. En el código del ejemplo, un atacante escribe como búsqueda: ' OR 1=1 -- . ¿Qué ocurre por la señal 2?

- ☐ El servidor devuelve "búsqueda inválida" y registra el intento.
- ☐ El texto cosido convierte su frase en parte de la consulta: el 1=1 hace que la condición sea siempre cierta y se devuelvan TODOS los clientes.
- ☐ La base de datos rechaza la comilla automáticamente.
- ☐ Nada: LIKE es inmune a inyecciones.

3. La señal 5 (catch vacío) solo es un problema estético: el error igual ocurrió.

☐ Verdadero

☐ Falso

Detecta el escaneo mal hecho

ACTIVIDAD
INTERACTIVA
DISPONIBLE EN
LUMBRE

4. Cuatro alumnos revisan código generado antes de aceptarlo. Señala el escaneo que NO encuentra las siete señales:



5. Pasa el escaneo de cuatro pasadas al último archivo crítico que la IA te generó (login, subida de archivos o el que toque datos). Anota las señales encontradas —o "cero señales" con la evidencia de cada pasada— y pídele la corrección de cada una con la señal nombrada: «tienes SQL cosido con más (señal 2); parametrízalo».

La desconfianza con lista es inspección; sin lista es mala onda.

Referencia curricular

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

Versión interactiva completa (con actividades, videos y autoevaluación):



<https://empresa.lumbre.lat/object/fsi-cnf-0a02>

Material educativo generado desde Lumbre · lumbre.lat