

# Unidad

## Inyección SQL

Bases de Datos Aplicada

v1.0 – Agosto 2023



Universidad Nacional  
de La Matanza

**DIIT**  
Departamento de Ingeniería e  
Investigaciones Tecnológicas

# Inyección SQL (SQL Injection)

- Se dice que existe o se produjo una inyección SQL cuando se inserta o "inyecta" código SQL invasor dentro del código SQL programado, con el fin de alterar el funcionamiento normal del programa y lograr así que se ejecute la porción de código "invasor" incrustado, en la base de datos.
- Puede ser utilizada para realizar ataques maliciosos, como robo de información, modificación o eliminación de datos.

# Inyección SQL (SQL Injection)

- Los ataques de inyección de SQL tienen el fin de introducirse en la base de datos de un sitio web. A veces solo quieren eliminar datos, pero otras veces lo que buscan es editar la base de datos, especialmente en el caso de sitios web financieros.
- Podemos tener **diferentes tipos** de ataques de inyección SQL:
  - mediante la introducción de datos del usuario
  - mediante la modificación de cookies
  - mediante variables de servidor
  - mediante herramientas de hackeo automáticas
  - Ataques SQL de segundo orden

# Inyección SQL (SQL Injection)

## Inyección de SQL mediante la introducción de datos del usuario

- Es la forma más sencilla de perpetrar un ataque de inyección de SQL.
- Muchos de sitios web recopilan las entradas del usuario y las transmiten al servidor. Sin un saneamiento de entrada seguro, un formulario con campos para rellenar o un recuadro para poner comentarios constituyen una vulnerabilidad.
- En lugar de cumplimentar estos formularios con contenido y respuesta normales, se introduce una secuencia de comandos de código SQL.

# Inyección SQL (SQL Injection)

Inyección de SQL mediante la introducción de datos del usuario

Cuando iniciamos sesión en un sitio web lo que ocurre es una consulta SQL. En el siguiente ejemplo podemos ver como se iniciará sesión exitosamente.

## Iniciar sesión

Nombre de usuario:

Contraseña:

☒ Mostrar contraseña



```
-- Cuando realizamos un Log In      --  
-- en un sitio web lo que se esta  --  
-- realizando es una consulta SQL  --  
-----  
  
select * from usuarios  
WHERE usuario='princessLeia' AND contraseña='leia123'
```

Results Messages

id	usuario	contraseña
3	princessLeia	leia123

# Inyección SQL (SQL Injection)

- Las contraseñas suelen estar cifradas por lo tanto se utiliza el casillero de usuario para poder inyectar código SQL.

## Iniciar sesión

Nombre de usuario:

Contraseña:

☒ Mostrar contraseña



```
select *
from usuarios
WHERE usuario='princessLeia' or 1=1 -- - AND contraseña='leia123'
```

100 %

Results Messages

	id	usuario	contraseña
1	1	lukeSkywalker	kywalker123
2	2	darthVader	vader123
3	3	princessLeia	leia123
4	4	obiWanKenobi	kenobi123
5	5	voda	voda123

*1=1 siempre será verdadero por lo tanto se ejecuta la consulta y se ingresará a la página aún no teniendo ningún usuario válido*

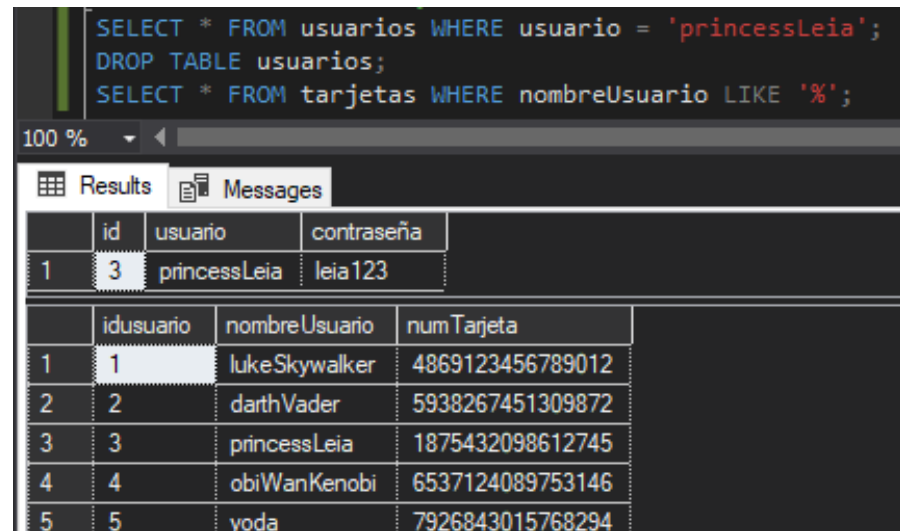
# Inyección SQL (SQL Injection)

- Se insertó en el usuario: **princessLeia'; DROP TABLE usuarios; SELECT \***  
**FROM datos WHERE nombreUsuario LIKE '%'**

## Iniciar sesión

Usuario:

Contraseña:



```
SELECT * FROM usuarios WHERE usuario = 'princessLeia';  
DROP TABLE usuarios;  
SELECT * FROM tarjetas WHERE nombreUsuario LIKE '%';
```

	id	usuario	contraseña
1	3	princessLeia	leia123

	idusuario	nombreUsuario	numTarjeta
1	1	lukeSkywalker	4869123456789012
2	2	darthVader	5938267451309872
3	3	princessLeia	1875432098612745
4	4	obiWanKenobi	6537124089753146
5	5	yoda	7926843015768294

*Por lo tanto, se ejecutará la consulta en el orden dado, se seleccionarían todos los registros con el nombre 'princessLeia', se borraría la tabla 'usuarios' y finalmente se seleccionaría toda la tabla "tarjetas", que no debería estar disponible para los usuarios web comunes.*

# Inyección SQL (SQL Injection)

## Inyección de SQL para obtener información de la base de datos

Por ejemplo, dada una aplicación de venta de productos que muestra los mismos en diferentes categorías. Cuando el usuario hace click, por ejemplo, en la categoría teclados, su navegador solicitará la siguiente URL:

<http://insecure-application.com/products?category=keyboards>

Realiza la siguiente consulta SQL para recuperar los productos de dicha categoría:

**SELECT \* FROM products WHERE category = 'keyboards' AND published = 1**

En este ejemplo la aplicación no implementa ninguna defensa contra este tipo de ataques



# Inyección SQL (SQL Injection)

## Inyección de SQL para obtener información de la base de datos

Por lo que se puede construir un ataque modificando el parámetro de la url para que muestre todos los productos independientemente de la categoría:

<http://insecure-application.com/products?category=keyboards'+OR+1=1-->

Y nuestra consulta SQL quedaría de la siguiente forma:

```
SELECT * FROM products WHERE category = 'keyboards' OR 1=1--' AND  
published = 1
```

La consulta modificada devolverá todos los elementos donde la categoría sea keyboards o, donde 1 sea igual a 1. Como 1=1 siempre es verdadero, la consulta devolverá todos los elementos.

# Inyección SQL (SQL Injection)

- Para **prevenir la inyección SQL**:
  - Se puede utilizar consultas parametrizadas, que permiten separar los datos de las instrucciones SQL.
  - Es importante validar y filtrar adecuadamente los datos ingresados por el usuario antes de utilizarlos en consultas SQL.
  - Uso de ORM (Mapeo Objeto-Relacional)
  - Limitar los permisos de la DB: asegurarse de que la cuenta de usuario tenga solo los permisos necesarios.

# Inyección SQL (SQL Injection)

- Para **prevenir la inyección SQL** podemos utilizar **sp\_executesql**. Este es un procedimiento almacenado que ejecuta consultas dinámicas y maneja parámetros de manera segura. Siguiendo con nuestra tabla usuarios, ejecutamos la siguiente consulta:

```
DECLARE @sqlQuery NVARCHAR(MAX)
DECLARE @paramUsuario NVARCHAR(50) = 'princessleia'
DECLARE @paramContraseña NVARCHAR(50) = 'leia123'

SET @sqlQuery = N'SELECT * FROM usuarios WHERE usuario = ''' + @paramUsuario + ''' AND contraseña = ''' + @paramContraseña + ''''
EXEC(@sqlQuery)

--Acá efectivamente se va a mostrar el usuario y la contraseña ya que ambos datos son válidos.
```

100 %

Results Messages

	id	usuario	contraseña
1	3	princessLeia	leia123

*Efectivamente se va a mostrar el usuario y la contraseña ya que ambos datos son válidos.*

# Inyección SQL (SQL Injection)

- ¿Pero qué pasaría si quisiera inyectar código SQL en mi consulta?

```
DECLARE @sqlQuery NVARCHAR(MAX)
DECLARE @paramUsuario NVARCHAR(50) = 'user1'
DECLARE @paramContraseña NVARCHAR(50) = ''' OR 1 = 1 --'

SET @sqlQuery = N'SELECT * FROM usuarios WHERE usuario = ''' + @paramUsuario + ''' AND contraseña = ''' + @paramContraseña + ''''
EXEC(@sqlQuery)
```

100 %

Results Messages

	id	usuario	contraseña
1	1	lukeSkywalker	kywalker123
2	2	darthVader	vader123
3	3	princessLeia	leia123
4	4	obiWanKenobi	kenobi123

*Al concatenar los parámetros directamente en la cadena, la consulta se vuelve vulnerable a la inyección. En este caso @param2 contiene " OR 1 = 1 --, por lo tanto, la consulta devolverá todas las filas de la tabla, ignorando la condición original.*

# Inyección SQL (SQL Injection)

- Utilizando sp\_executesql

```
DECLARE @sqlQuery NVARCHAR(MAX)
DECLARE @paramUsuario NVARCHAR(50) = 'user1'
DECLARE @paramContraseña NVARCHAR(50) = '' OR 1 = 1 --'

SET @sqlQuery = N'SELECT * FROM usuarios WHERE usuario = @paramUsuario AND contraseña = @paramContraseña'

-- Ejecutar la consulta dinámica de forma segura
EXEC sp_executesql @sqlQuery, N'@paramUsuario NVARCHAR(50), @paramContraseña NVARCHAR(50)', @paramUsuario, @paramContraseña
```

Results Messages

id	usuario	contraseña
----	---------	------------

***La consulta no me devolverá información, previniendo la Inyección SQL.***

*Al declarar y pasar parámetros de esta manera, se protege la consulta contra manipulaciones maliciosas, ya que los parámetros se manejan de manera segura, evitando la concatenación directa de valores en la cadena SQL*

# ¿Dudas?



Universidad Nacional  
de La Matanza

**DIIT**  
Departamento de Ingeniería e  
Investigaciones Tecnológicas