## How to detect SQL injection vulnerabilities

Systematic set of tests against every entry point in the application:

* single quote character ' and looking for errors or other anomalies.
* some SQL-specific syntax that evaluates to the base (original) value of the entry point, and to a different value, and looking for systematic differences in the resulting application responses.
* Boolean conditions such as OR 1=1 and OR 1=2
* Submitting payloads designed to trigger time delays when executed within an SQL query, and looking for differences in the time taken to respond.
* Submitting OAST payloads designed to trigger an out-of-band network interaction when executed within an SQL query, and monitoring for any resulting interactions.

**How to exploit?**

1. Comments:
   1. --
      1. to by pass
         1. logic: show me more elements
         2. password:
      2. leverage:
         1. get in as user ‘admin’
   2. #
2. Tautologies
   1. ‘ OR 1=1
3. Data from other tables
   1. ‘ UNION SELECT username, password FROM users
4. Examining the database
   1. SELECT \* FROM v$version
   2. SELECT \* FROM information\_schema.tables

## Second-order SQL injection

## First-order SQL injection arises where the application takes user input from an HTTP request and, in the course of processing that request, incorporates the input into an SQL query in an unsafe way.