SQL Injection

- SQL injection happens when the attacker uses SQL commands by inserting malicious script inside of it , SQL injection is the result of incorrectly filtered or escaped and can lead to authentication bypass
- there is two types of SQL injection
  1. classic
  2. Blind
  3. No SQL
- one of the most famous and used injections is the (--) where this stand for a comment in SQL , so that the attacker can escape the password part of the authentication using only the user name.

```
select Id FROM users
where username ='admin';--'&password='password';
```

```
// the prevoius commands retreive the username from the data base
   without checking for password by using the special character
```

- knowing and using SQL commands is a must in SQL injection the syntax can vary fro the purpose as (select , union, update ,set , ...etc) depending on the logic needed the use of all of them.
- attackers must use the comment (--) after the injection payload so that the rest of the SQL commands don't miss up the logic of the injection .
- all the last SQL injections are first order SQL injection there another type of it which is **second order SQL injection** it's where the input from the user is filtered and processed and stored on the database at the server but the retrieval of the data is wrong and the server trust the data and marked it as safe .
- lets imagine a website signup page that allows the user to input the user name and password , using that the attacker uses's ```

```
username ="name'UNION SELECT Username, password From Users;--
"&password=password
```

```
// those injected SQl commands are stored in the string username at the
   database so that when the attcker try to signin the server will
   use select title , body from emails where username="username" and then
```

```
because the username of the attacker it will be as the following
select title , body from emails where username='name' union select
username , password From Users;-- this will return all the usernmaes
and passwords in the database as email titles and bodies inside the
http response
```

- while using the SQL statements we may use the (%) to specify the starting with target | e.x: select * from users where username like 'a%' **this will return all users starting with a**
- the same thing can be done to get the ending of the target name ex: select * from users where username like 'a%' **this will return all users ending with the letter a**
- the prevention of SQL injection have two main methods , first (*prepared statements , and allow lists*)
    1. first the prepared statements works by making the developer specify the logic and structure of the SQL query and then the user input is added before the execution so that the user input is treated as a Blaine text instead of a logical flow.
    2. second allow list allows the user to use some statements that are harmful to the code logic and won't make a change in the structure of execution.

---

port swagger labs :causes malicious queries to be executed. When a web

1. lab (1) is a lab introduction to SQL injection there is no input field to check for SQL injection so we look for any data retrieval to check if there is any SQL injection so we test the filter (ex.gifts) it returns three items only so we inject a (') in the parameter we get internal server error means that we can find SQL injection so i add a (') to the filter before the name of the category (ex. /filter?category= ' OR 1=1 -- ; Food+%26+Drink) this displays all the products there is (**solved**).
2. lab (2) the injection is in the login field where i added the '-- after the username to comment the password and access the account (ex. administrator ' --) this gave me access without needing passwords.(**solved**)
3.

Hunting for SQL Injection:

1. first we check every and each user input field by adding (') because the quote defines the ending if an SQL command and if the website is protected from it will not show anything because it's treated as a plain text , if not an Error will appear.
2. there maybe no errors that occur so we use time stamp or union based injection, the time stamp injection is produced like every other SQL injection after using a parameter that has user input or data retrieval (ex.filters) so we can inject ' OR SLEEP(100);-- another way also is the pg_sleep(100);--
3. we use SQL map for automation but i don't trust it the manual work is a must in SQL injection
4. if there is no parameter for the target while using SQL map we use tools like **arjun** to fuzz the parameters.