**What is a SQL Injection?**

**SQL is the acronym for Structured Query Language**. It is used to retrieve and manipulate data in the database.

SQL Injection is an attack that poisons dynamic SQL statements to comment out certain parts of the statement or appending a condition that will always be true.

**How SQL Injection Works**

The types of attacks that can be performed using SQL injection vary depending on the type of database engine. **The attack works on dynamic SQL statements**. A dynamic statement is a statement that is generated at run time using parameters password from a web form or URI query string.

Let’s consider a simple web application with a login form. The code for the HTML form is shown below.

[?](http://www.guru99.com/learn-sql-injection-with-practical-example.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <form action=‘index.php’ method="post">    <input type="email" name="email" required="required"/>    <input type="password" name="password"/>    <input type="checkbox" name="remember\_me" value="Remember me"/>    <input type="submit" value="Submit"/>    </form> |

**HERE,**

* The above form accepts the email address and password then submits them to a PHP file named index.php.
* It has an option of storing the login session in a cookie. We have deduced this from the remember\_me checkbox. It uses the post method to submit data. This means the values are not displayed in the URL.

Let’s suppose the statement at the backend for checking user ID is as follows

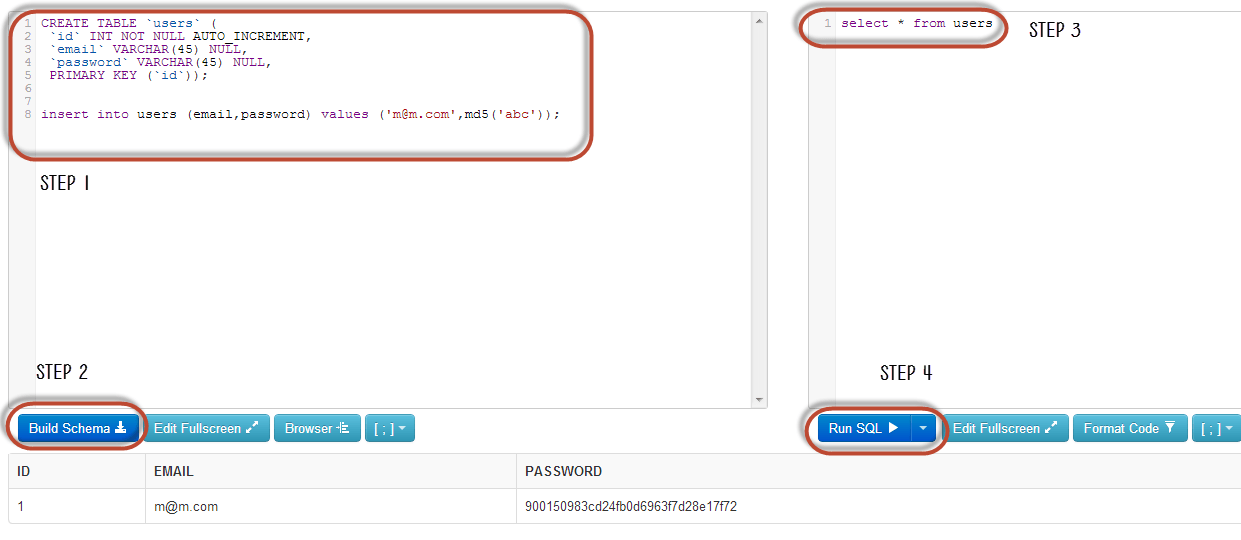
SELECT \* FROM users WHERE email = $\_POST['email'] AND password = md5($\_POST['password']);

**HERE,**

* The above statement uses the values of the $\_POST[] array directly without sanitizing them.
* The password is encrypted using MD5 algorithm.

We will illustrate SQL injection attack using sqlfiddle. Open the URL <http://sqlfiddle.com/#!2/3286e/1> in your web browser. You will get the following window.

Note: you will have to write the SQL statements

[](http://cdn.guru99.com/images/EthicalHacking/Article_13_2.png)

**Step 1)**Enter this code in left pane

[?](http://www.guru99.com/learn-sql-injection-with-practical-example.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE TABLE `techpand`.`users` (      `id` INT NOT NULL AUTO\_INCREMENT,      `email` VARCHAR(45) NULL,      `password` VARCHAR(45) NULL,      PRIMARY KEY (`id`)); |

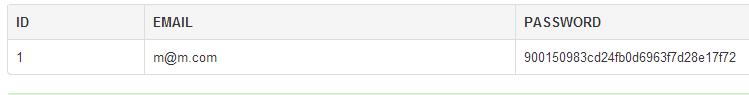
insert into users (email,password) values ('m@m.com',md5('abc'));

**Step 2)** Enter this code in right pane

select \* from users;

**Step 3)**Click Build Schema

**Step 4)**Click Run SQL. You will see following result

[](http://cdn.guru99.com/images/EthicalHacking/Article_13_3.png)

Suppose a user supplies [admin@admin.sys](mailto:admin@admin.sys) and 1234 as the password. The statement to be executed against the database would be

SELECT \* FROM users WHERE email = 'admin@admin.sys' AND password = md5('1234');

The above code can be exploited by commenting out the password part and appending a condition that will always be true. Let’s suppose an attacker provides the following input in the email address field.

xxx@xxx.xxx' OR 1 = 1 LIMIT 1 -- ' ]

xxx for the password.

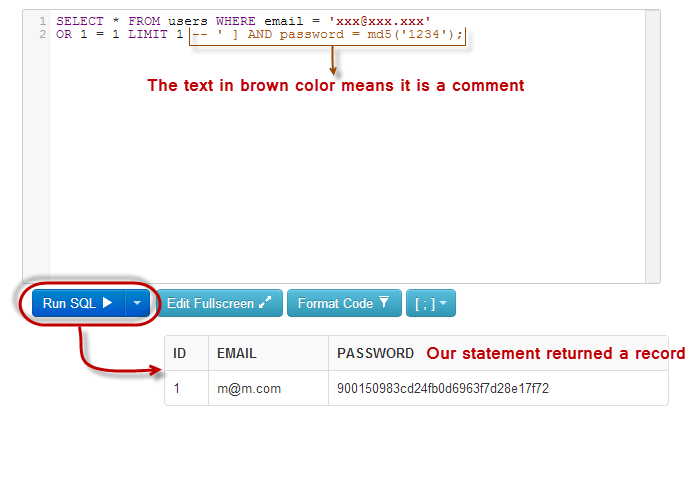
The generated dynamic statement will be as follows.

SELECT \* FROM users WHERE email = 'xxx@xxx.xxx' OR 1 = 1 LIMIT 1 -- ' ] AND password = md5('1234');

**HERE,**

* [xxx@xxx.xxx](mailto:xxx@xxx.xxx) ends with a single quote which completes the string quote
* OR 1 = 1 LIMIT 1 is a condition that will always be true and limits the returned results to only one record.
* -- ' AND … is a SQL comment that eliminates the password part.

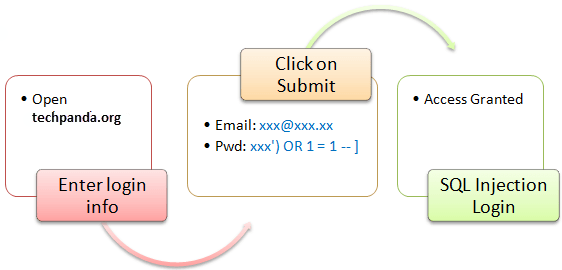
Copy the above SQL statement and paste it in SQL FiddleRun SQL Text box as shown below

****

**Hacking Activity: SQL Inject a Web Application**

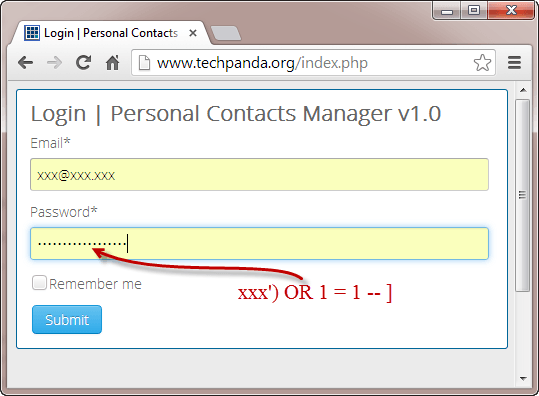
We have a simple web application at <http://www.techpanda.org/> **that is vulnerable to SQL Injection attacks for demonstration purposes only.** The HTML form code above is taken from the login page. The application provides basic security such as sanitizing the email field. This means our above code cannot be used to bypass the login.

To get round that, we can instead exploit the password field. The diagram below shows the steps that you must follow

[](http://cdn.guru99.com/images/EthicalHacking/Article_13_5.png)

Let’s suppose an attacker provides the following input

* Step 1: Enter xxx@xxx.xxx as the email address
* Step 2: Enter xxx') OR 1 = 1 -- ]

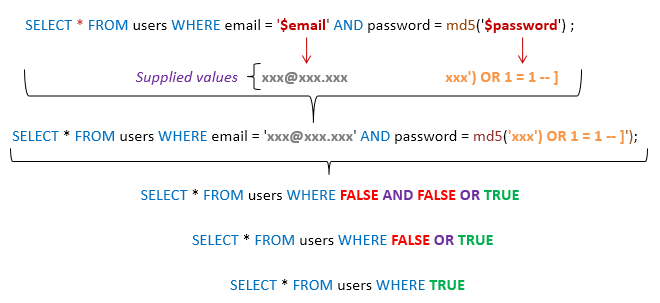


* Click on Submit button
* You will be directed to the dashboard

The generated SQL statement will be as follows

SELECT \* FROM users WHERE email = 'xxx@xxx.xxx' AND password = md5('xxx') OR 1 = 1 -- ]');

The diagram below illustrates the statement has been generated.



**HERE,**

* The statement intelligently assumes md5 encryption is used
* Completes the single quote and closing bracket
* Appends a condition to the statement that will always be true

In general, a successful SQL Injection attack attempts a number of different techniques such as the ones demonstrated above to carry out a successful attack.

**Other SQL Injection attack types**

SQL Injections can do more harm than just by passing the login algorithms. Some of the attacks include

* Deleting data
* Updating data
* Inserting data
* Executing commands on the server that can download and install malicious programs such as Trojans
* Exporting valuable data such as credit card details, email and passwords to the attacker’s remote server
* Getting user login details etc

The above list is not exhaustive; it just gives you an idea of what SQL Injection

**Automation Tools for SQL Injection**

**In the above example, we used manual attack techniques based on our vast knowledge of**SQL. There are**automated tools that can help you perform the attacks more efficiently and within the shortest possible time**. These tools include

* SQLDict - <http://ntsecurity.nu/toolbox/sqldict/>
* SQLSmack - <http://www.securiteam.com/tools/5GP081P75C.html>
* SQLPing 2 - <http://www.sqlsecurity.com/downloads/sqlping2.zip?attredirects=0&d=1>
* SQLMap - <http://sqlmap.org/>

**How to guard against SQL Injection Attacks**

An organization can adopt the following policy to protect itself against SQL Injection attacks.

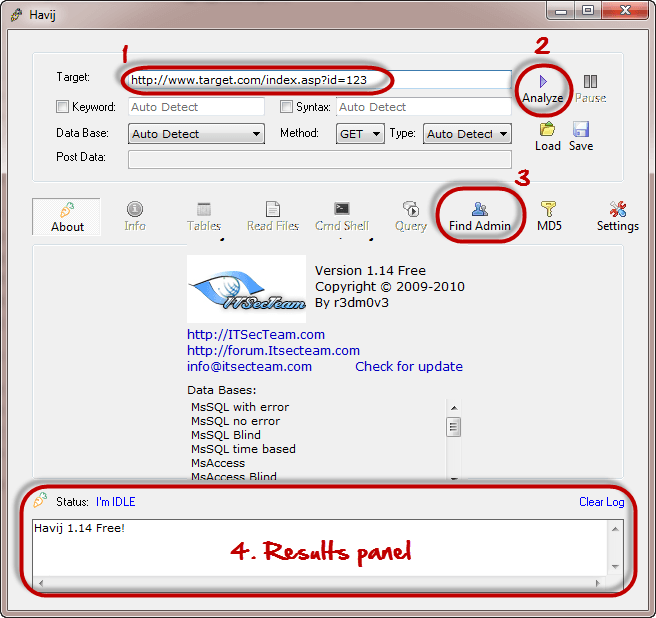
* **User input should never be trusted.**It must always be sanitized before it is used in dynamic SQL statements.
* **Stored procedures –**these can encapsulate the SQL statements and treat all input as parameters.
* **Prepared statements –**prepared statements work by creating the SQL statement first then treating all submitted user data as parameters. This has no effect on the syntax of the SQL statement.
* **Regular expressions –**these can be used to detect potential harmful code and remove it before executing the SQL statements.
* **Database connection user access rights –**only necessary access rights should be given to accounts used to connect to the database. This can help reduce what the SQL statements can perform on the server.
* **Error messages –**these should not reveal sensitive information and where exactly an error occurred. Simple custom error messages such as “Sorry, we are experiencing technical errors. The technical team has been contacted. Please try again later” can be used instead of display the SQL statements that caused the error.

**Hacking Activity: Use Havij for SQL Injection**

In this practical scenario, we are going to use Havij Advanced SQL Injection program to scan a website for vulnerabilities.

Note: your anti-virus program may flag it due to its nature. You should add it to the exclusions list or pause your anti-virus software.

The image below shows the main window for Havij



The above tool can be used to assess the vulnerability of a web site/application.