**Gathering The Information And Structure Of The Data Base Using**

**Logically Incorrect Queries**

Nikhil V, Aashish Raj, Srinivas

18BIT0196 ,18BIT0113, 18BIT0007

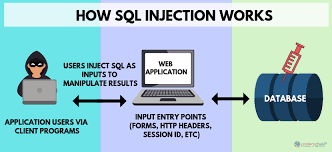
School Of Information Technology

**Abstract-** **SQL injection is a technique where the attacker injects an input in the query in order to change the structure of the query intended by the programmer and gaining the access of the database which results modification or deletion of the user’s data. In the injection it exploits a security vulnerability occurring in database layer of an application. SQL injection attack is the most common attack in websites in these days. Some malicious codes get injected to the database by unauthorized users and get the access of the database due to lack of input validation. Input validation is the most critical part of software security that is not properly covered in the design phase of software development life-cycle resulting in many security vulnerabilities. This paper presents the techniques for detection and prevention of SQL injection attack. There are no any known full proof defences available against such type of attacks. In this paper we have made a website based on charity works to donate funds as it contains the information of all the account details and there history of donations .**

**Keywords— web application, SQLIA, detection, prevention, vulnerabilities, web architecture**

1. **INTRODUCTION**

Now a days web application is widely used in various applications it is the reliable and efficient solution to the challenges of communicating and conducting the various organisation, business or commerce over the internet. Now each and every important assignment is done by using the web application which is connected through the internet. For example electricity bill, online shopping, gaming, banking, messaging, shopping, conferences, etc. So the increase of web application involving the various security issues in the web world. The SQLIA (structured query language injection attack) is a code injection attack technique commonly used for attacking websites in which an attacker injects some SQL codes in place of the original codes to get access the database. There are variety of techniques are available to detect SQLIA. The most preferred are Web Framework, Static Analysis, Dynamic Analysis, combined Static and Dynamic Analysis and Machine Learning Technique. Web Framework provides filters to filter special characters but other attacks are not detected. Static Analysis checks the input parameter type, but it fails to detect attacks with correct input type. Dynamic Analysis technique is capable of scanning vulnerabilities of web application but is not able to detect all types of SQLIA. Combined Static and Dynamic Analysis includes the benefit of both, but this method is very complex in order to proceed. Machine Learning method can detect all types of attacks but results in number of false positives and negatives.



1. **SQLIA MECHANISMS**

Types of SQL Injection Attacks There are multitude techniques of attack for exploiting SQL injection vulnerabilities in web applications. In this paper we describe the major types of attacks.

**Illegal/Logical Incorrect Query**:

This kind of attack manipulate the where clues to get the names of the tables and columns that the inquiry works on through using the "having" condition of the "select" statement to generate error message by the server. **Select \* from usertable where username = ‘ ” + g\_name + ” ’ and password = ‘ “ + g\_password + ” ‘ “ ; "** having 1=1- - Result: Column 'usertbl.ID' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

1. **PROPOSED APPROACH**

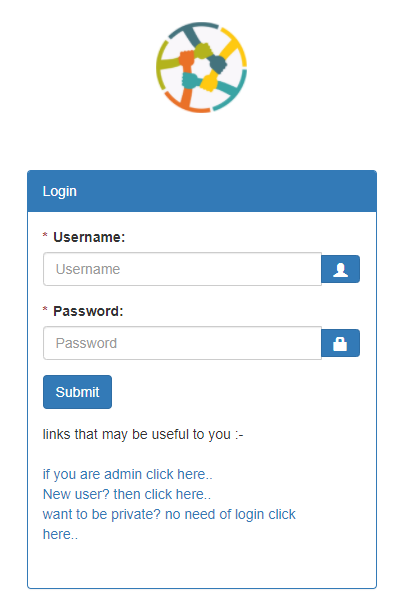
Here we’ll use different logical operators to go sneak inside the website and then entering in it without the correct username and the passwords and by just using vulnerabilities of SQL queries.

1. **Using OR operator**

We will first close the string of the SQL string statement and then we will add an **TRUE statement** with **OR** which will eventually join the whole SQL statement and we’ll get the output of that query as **TRUE** or we can say that the query produces results.

**Example:**

Explaining it with the query used in the charity website mentioned above.



**FIGURE: 3.1**

As we can see from figure 3.1 that in the username form we can fill any thing this will not affect the query and in password column we will write this query:

**" or ""="**

The SQL statement will look like:

SELECT \* FROM users WHERE u\_name ="fasd” AND u\_pass ="**" or ""=""**

The query will produce some results as the whole statement after where returns true

There can be another way to use the same **OR query** by using **comment.**

This statement in the username field:

**" or ""="" –**

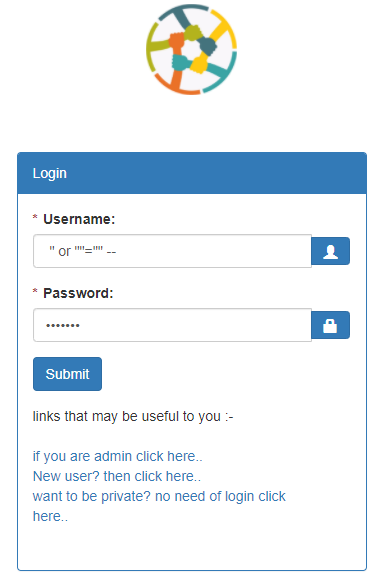
And in password any thing can be written as its going to be comment it out.

The resulting **SQL**statement we will get after this will be

SELECT \* FROM users WHERE u\_name ="**" or ""="" --**

AND u\_pass ="FASDF**"**

The whole underlined part will be commented and the part executed will whole become true and return some value for the result.



**FIGURE: 3.2**

1. **Using XOR operator**

We will first close the string of the SQL string statement and then we will add an **TRUE statement** with **XOR** which will eventually join the whole SQL statement and we’ll get the output of that query as **TRUE** or we can say that the query produces results.

NOTE:

The whole statement will be true as we know that the previous value of the username we entered is wrong as we include many random texts and it can’t belong to a person, and xor of two of these things will give the result as true.

**Example:**

Explaining it with the query used in the charity website mentioned above.

As we can see from figure 3.1 that in the username form we can fill any thing this will not affect the query and in password column we will write this query:

**" xor ""="**

The SQL statement will look like:

SELECT \* FROM users WHERE u\_name ="fasd” AND u\_pass ="**" xor ""=""**

The query will produce some results as the whole statement after where returns true

There can be another way to use the same **XOR query** by using **comment.**

This statement in the username field:

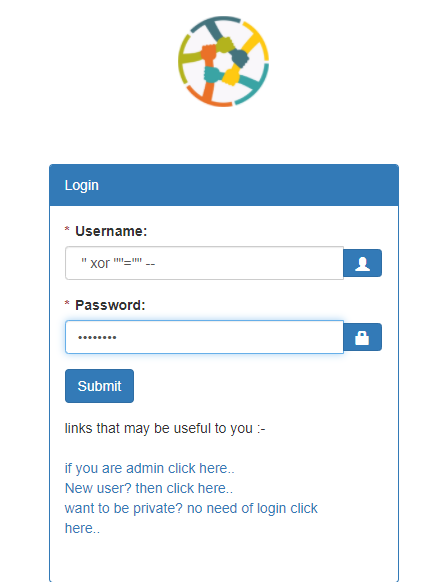
**" xor ""="" –**

And in password any thing can be written as its going to be comment it out.

The resulting **SQL**statement we will get after this will be

SELECT \* FROM users WHERE u\_name ="**" xor ""="" --**  AND u\_pass ="FASDF**"**

The whole underlined part will be commented and the part executed will whole become true and return some value for the result.



**FIGURE: 3.3**

1. **RESULT**

We performed logical incorrect queries of **SQL injection** we are successfully able to use the vulnerabilities of the SQL queries and we get inside the website breaking its security.

We can remove this vulnerability by having the SQL page conforming the username and password in the separate page so that the error result sent must not be appeared in the login window.

We can change our approach of password confirming by having the password checked only if the username of a particular person exists. Like we will run query for the person username then after that if the username exists then we will see its password and we will compare it with the given value. This will not allow the system to accept the query even if it is producing some results for the query.

1. **CONCLUSION**

We performed logical incorrect queries of **SQL injection** we gave a method to protect the website from this type of attack.