14 Basic Cyber Security measures - SQL injection

SQL injection is an attack where an attacker can inject or execute malicious SQL code via a form input which is processed by the application server

It can be used to expose sensitive information like user's contact numbers, email addresses, credit card information and so on.

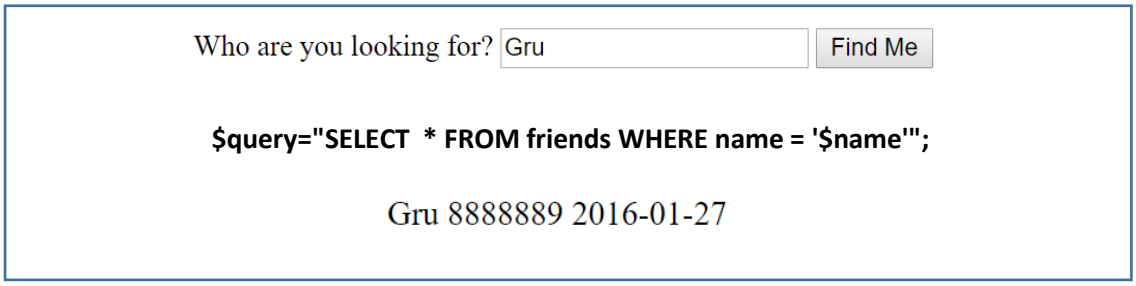
An attacker can even use it to bypass authentication process and get access to the entire database. A simple example on how it works:

How SQL Injection Works

This is a simple example of searching for a friend called Gru

$query="SELECT * FROM friends WHERE name = 'Gru’";

Using input from a form where Gru is entered, results in the word Gru being stored in the variable **$name** and would give the following result.



If the user is an attacker, instead of entering a valid search word in the form input field they could enter something like:

anything' OR 'x'='x

In this case, this SQL query will be constructed:

$query="SELECT * FROM friends WHERE name = 'anything' OR 'x'='x'";

This statement is a valid SQL statement and since **WHERE 'x'='x'** is always true, the query will **return all rows from the *friends* table**.

You can see how easily an attacker can get access to all the sensitive information of a database with just a little dirty trick.

If the *users* table is quite large and contains millions or rows, this single statement can also lead to denial-of-service attack (DoS attack) by overloading the system resources and make your application unavailable for legitimate users.

Warning: The consequences of ignoring SQL injection vulnerability can be even worse if your script generates a DELETE or UPDATE query. An attacker can delete data from the table or change all of its rows permanently.

Preventing SQL Injection

Always validate user input and make no assumptions. Never build SQL statements directly from user input.

In PHP and MySQL you can use **mysqli_real_escape_string()** function to create a legal SQL string.

Here's a very basic example of user authentication using PHP and MySQL that demonstrates how to prevent SQL injection while taking input from users.

Instead of:

```
$username=$_POST['username'];
```

Use:

```
$username = mysqli_real_escape_string($conn, $_POST['username']);
```

This should be used for most form inputs – search, login, register, update – but **NOT a PASSWORD** field!!

Using **mysqli_real_escape_string()** encodes the following characters. In other words – they are ignored (like being commented out)

NUL (ASCII 0), \n, \r, \, ', ", and Control-Z

For example, entered into form:

anything' OR 'x'='x

Escaped version:

anything\' OR \'x\'=\'x

This means all the ' are ignored and the query won't work.

This example demonstrates how it can be used for the friends search

Task 1 – try this out

```php
<?php include "connaddressbook.php";

// Reminder, this check is needed to make sure the form has been sent
if(isset($_POST['search'])){

//User input is stored in variable called $name
    $name=$_POST['name'];
    $escapedname = mysqli_real_escape_string($conn, $_POST['name']);

    echo $name.'<br>';
    echo $escapedname.'<br>';


    $query="SELECT * FROM friends WHERE name = '$escapedname'";
    $result=mysqli_query($conn, $query);

    if(mysqli_num_rows($result) > 0){

        while ($row = mysqli_fetch_array($result)){
          $name=$row['name'];
          $phone=$row['phone'];
          $birthday=$row['birthday'];


  echo $name. " " . $phone. " " . $birthday. "<br />";

    }//close of while
  }// close of if - num_rows
    else echo "No records matching your query were found.";
} //close of if - isset

?>
```

Note – this example has some extra code for demonstration only. The only code you would need to change is:

```php
if(isset($_POST['search'])){

    $name = mysqli_real_escape_string($conn, $_POST['name']);
    $query="SELECT * FROM friends WHERE name = '$escapedname'";
    $result=mysqli_query($conn, $query);

    if(mysqli_num_rows($result) > 0){
```

Task 2

1. Secure your Shoestore products search from SQL injection
2. Secure your insert into the Shoestore suppliers table from SQL injection

Further info can be found here:

http://php.net/manual/en/mysqli.real-escape-string.php

https://www.w3resource.com/php/function-reference/mysqli_real_escape_string.php