

Report - Final

by Cai Perry

Submission date: 14-Jan-2025 12:41PM (UTC+0000)

Submission ID: 248405916

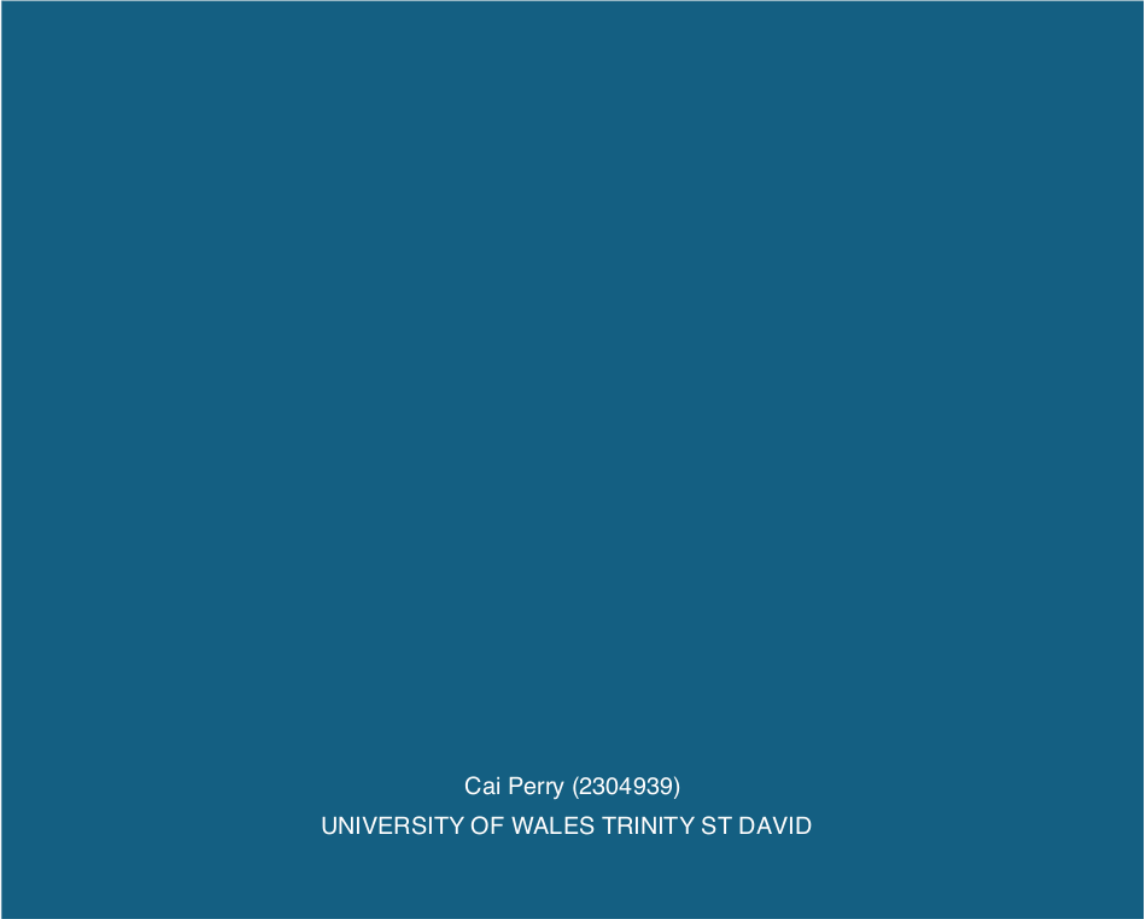
File name: 50391_Cai_Perry_Report_-_Final_630034_1557238782.docx (179.85K)

Word count: 648

Character count: 3000



WEBSITE REPORT



Cai Perry (2304939)
UNIVERSITY OF WALES TRINITY ST DAVID

Contents

SQL injection overview 2

Anti SQL Methods in My Project..... 2

SQL injection overview

SQL injection is a type of malicious attack where SQL queries are inserted into input fields so that they can authorise SQL commands that otherwise wouldn't be possible [1]. This could lead to possible data leaks or data breaches. To complete an SQL injection attack, the attacker will exploit a weakness in an input field such as a search box if they haven't been fully sanitised.

SQL injection will work when a web application is improperly validating user inputs [2], for example if a website were to take a username and password and insert it directly into an SQL query without sanitising it first, the attacker can then manipulate this query to perform actions that weren't intended by the designer.

An example of an SQL injection query would be a search query, for hospital patient information, if you were to inject the code " OR 1=1--" into the search query, it would show all of the records for a few clever reasons. First, the initial " will close the string, breaking out of the structure of the query. Next, as the logical condition 'OR 1=1' is always set to be true, this will be inserted into the query and will always come back as true therefore bypassing any authentication. Last, the '--' will let you create a comment in SQL, this makes the rest of the query get ignored to prevent any syntax errors from showing after the code is injected.

To prevent this from happening, you can utilise prepared statements. In my project, I have used '?' as placeholders for an input, for example a username and password. I have also used the statement Bind_param, this binds the variables (un and pw) to the placeholders (in this case '?'). This ensures the user input is treated as data and not part of the executable SQL code, this is known as 'sanitising' [3].

Anti SQL Methods in My Project

In my project, I have multiple instances of input forms and have also sanitised all data, this next section will be displaying all parts of my database where anti SQL injection measures were necessary.

```

<?php
session_start();
include "coninfo.php";

$un = $_POST["un"];
$pw = $_POST["pw"];

$query = "SELECT * FROM vidlogin WHERE username = ? AND password = ?";
$stmt = mysqli_prepare($link, $query);

mysqli_stmt_bind_param($stmt, "ss", $un, $pw);
mysqli_stmt_execute($stmt);

$result = mysqli_stmt_get_result($stmt);
$numrows = mysqli_num_rows($result);

if ($numrows == 0) {
    echo "Incorrect password.";
    echo '<a href="VidIndex.php">Back to the home page</a>';
} else {
    $_SESSION["username"] = $un;
    echo "Successful login, you can access the admin page here:";
    echo '<br><a href="AdminPage.php">Admin Page</a>';
}

mysqli_stmt_close($stmt);
?>

```

This is my check login file. Here it takes what has been input in the previous file and matches it with the username and passwords in the database, I have added a prepared statement in this code (highlighted in red) to make sure the information has been properly sanitised. To test this, I will be injecting malicious code to make sure the expected outcome shows, this would be the error message and return to home link.

Login

Please login:

Username:

Password:

[Return Home](#)

Incorrect password.[Back to the home page](#)

These two screenshots show me attempting to inject malicious SQL code into the password and username fields and then after clicking login, you are shown the message, 'incorrect password', meaning that the anti-SQL injection methods on this page work.

```

}if ($ft == "t") {
    $query = "SELECT * FROM games WHERE Title LIKE ?";
} else if ($ft == "rd") {
    $query = "SELECT * FROM games WHERE ReleaseDate LIKE ?";
} else if ($ft == "g") {
    $query = "SELECT * FROM games WHERE Genre LIKE ?";
} else if ($ft == "d") {
    $query = "SELECT * FROM games WHERE Developer LIKE ?";
} else {
    $query = "SELECT * FROM games WHERE Price LIKE ?";
}

$stmt = mysqli_prepare($link, $query);

//Because of LIKE usage, % wildcard is used to ensure query still functions
$gamesearch = "%" . $gamesearch . "%";

mysqli_stmt_bind_param($stmt, "s", $gamesearch);

mysqli_stmt_execute($stmt);

```

This is the second place where I have implemented prepared statements, in this case it is a search field. Here I had to also implement a % wildcard. This is because I have used LIKE as an operator. This allows me to search partial matches of records because I have allowed the user to first choose the category that they'd like to search by. If this wasn't implemented, the query wouldn't work as intended and would only search exact matches when checking against the database.

Search for a game

Search:

" OR 1=1--"

Search By:

Title: ☒

Release Date: ☐

Genre: ☐

Developer: ☐

Price: ☐

Submit

Sorry, nothing found
[Return to the home page.](#)

This is the outcome of inputting malicious SQL code into the search bar, instead of showing all records, it tells the user that nothing has been found and to return to the home page.

These are the only two places in my database where a user is asked to input something in an input field, as to edit and delete a record are displayed by a list rather than searching for a record to change or delete first.

Report - Final

GRADEMARK REPORT

FINAL GRADE

55/100

GENERAL COMMENTS

Database Structure and Design - You appear to have 3 stand-alone tables here. The review for the games are not related to the actual games on the site - you seem to be able to review any game. It would be better to link to games you have on the website. You could normalise by genre and developer.

4.5/10

Range of back-end coding -

Make sure when developing websites that your homepage is either home.php or index.php so that it loads automatically.

I couldn't add a game review at first - turned out the date needed to be in the future. Would be better just to date it today rather than give the user the choice.

I managed to log in, but you don't seem to have submitted AdminPage.php. Also, login does not appear to hold - you could remove the login link once the user has logged in. I did manage to find Delete and Edit in your folder - Edit also needs validation as I was able to

There is a problem with login in that I was able to delete a record without being logged in. You need to secure from URL attack.

Search is OK - genre would be better as a pulldown menu as there are a limited number of genres. This would give the user a better idea of what to search for.

27/50

Overall integration of website - A bit basic on the home page, you could have had a bit more information what the site was about.

Site is not consistently styled and some pages are not styled at all. You could use include files to help you with this.

5.5/10

SQL Injection overview - A good summary, but a little brief. You could have mentioned attacks other than the classic 1=1 attack 9/15

SQL Code implementation/proposal - You've shown where you input into the search, but add and edit are still vulnerable. I am not sure why you say edit is displayed by a list - you have to enter information in order to edit the record. 8.5/15

General comments:

You must avoid the use of "I", "my" and so on in academic reports - particular at Levels 5 and 6.

Overall mark for PHP assignment: 54.5%

Overall mark for module (including Divya's assignment) - %

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5
