

Assignment 1 Report

by Josh Perry

Submission date: 02-Dec-2022 02:00AM (UTC+0000)

Submission ID: 190737619

File name: 19625_Josh_Perry_Assignment_1_Report_460452_1698346858.docx (224.22K)

Word count: 1143

Character count: 5945



DATABASE-DRIVEN WEBSITE APPLICATION

Databases and Application Development

Abstract

A short report detailing the definition and impact of SQL injection and broken access control and potential defence mechanisms. This report also includes a section reviewing the implemented security features on the project site for this assignment and details potential further implementations to improve security.

Josh Perry
1903738

Table of Contents

1. Security Issues with Websites Using PHP and SQL	1
1.1. SQL Injection	1
1.2. Broken Access Control	2
2. Implementation of Security Features and Possible Future Security Features	3
2.1. Defending against SQL Injection	3
2.2. Impact of Broken Access Control and Potential Security Implementations	4
3. References	5

1. Security Issues with Websites Using PHP and SQL

1.1. SQL Injection

SQL Injection is a term for the malicious injection of SQL into a user input field of a website, such that when a query is carried out on this field, it will evaluate this field as SQL statements, meaning that the end user could gain access to and steal or otherwise edit sensitive or private organisational and user data. There is no end to the destructive capabilities of SQL injection. Although not recent, this example proves that SQL injection can have very real and very severe real-world impacts on many people. In 2007 the American retailer 7-Eleven was targeted by an SQL injection attack that saw over 130 million credit card numbers stolen, costing a total of \$300 million dollars in losses. [1]

Although an old technique, it is still very much a risk, no matter how well protected a site is, many hackers will find a way in wherever they can. Many sites are labelled as PCI compliant (A policy employed by credit card companies to ensure the protection of online transactions) and have still been hacked, much like the example mentioned earlier. As with any industry in the tech sector, new vulnerabilities are always being discovered and there is always room for mistakes and error in any project. This is why there is a constant cycle of updates and security testing to limit the number of vulnerabilities at any one time.

SQL Injection works on the principle of if there is no data validation enforced on the website to prevent a user inputting wrong data, then the malicious party will be able to inject a statement in a way that displays all records from that table(s) with a statement that looks like the example below.

User Name:

Password:

Figure 1-SQL Injection Example [1]

The resulting SQL Statement would look like:

```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""="";
```

As can be seen, the user has tricked the system into thinking the input data is blank by inserting closed quotes, then injecting their statements. OR""="" will always return true. So, in this case the user will see all available data in the Users table.

1.2. Broken Access Control

Broken access control is a growing problem. According to Contrast Security between September and November 2020, the average RiskScore raised to an all-time high, topping the chart of potential threats and spending the most amount of time in the 'Top 3' Security Threats.

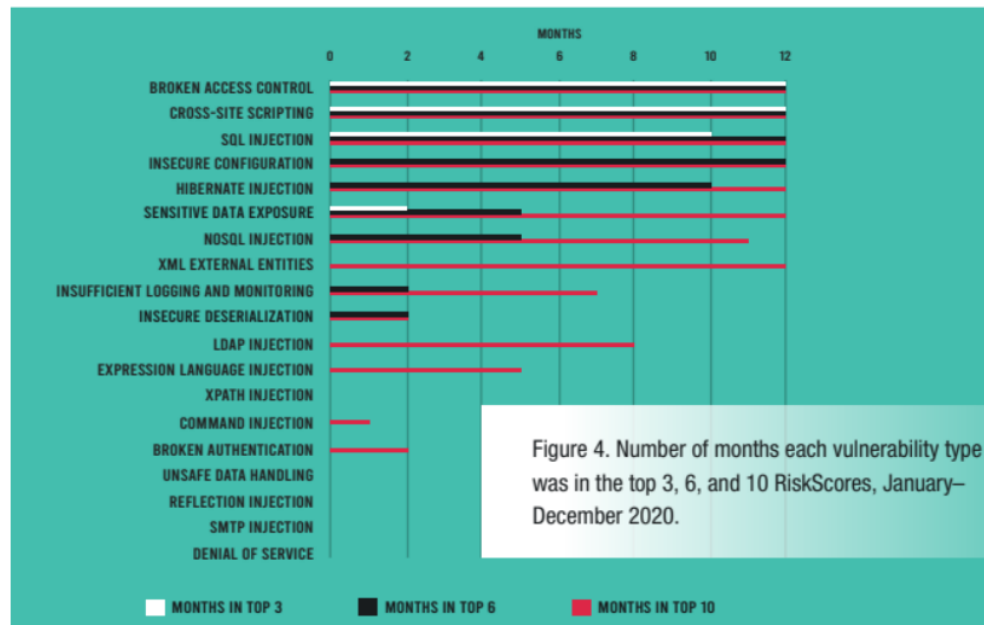


Figure 4. Number of months each vulnerability type was in the top 3, 6, and 10 RiskScores, January–December 2020.

Figure 2 Contrast RiskScore 2020 Review [2]

This is where a user can perform action outside of their level of access. For example, a basic user who is only meant to be able to read data, would be able to perform actions above their intended level of control. This typically results in malicious users being able to access many features they should not be able to. Including but not limited to:

- API features without being from a trusted source
- access any user account through SQL statements. This happens due to poor validation.

This can be prevented by implementing and reusing user access control systems throughout every page of a website.

2. Implementation of Security Features and Possible Future Security Features

2.1. Defending against SQL Injection

There are certain measures that can be taken to prevent SQL Injection. One method is to use prepared statements.

```
function submitForm($conn, $name, $email, $cmt){ //function to submit contact form, same as create user but in a different context for a different page and table.
    $sql = "INSERT INTO contact (contactName, contactEmail, contactComment) VALUES (?, ?, ?)";
    $stmt = mysqli_stmt_init($conn);
    if(!mysqli_stmt_prepare($stmt, $sql)){
        header("Location: ../contact.php?error=stmtfailure");
        exit();
    }
    mysqli_stmt_bind_param($stmt, "sss", $name, $email, $cmt);
    mysqli_stmt_execute($stmt);
    mysqli_stmt_close($stmt);
    header("Location: ../contact.php?error=none");
    exit();
}
```

Figure 3- Screen Capture of Project Site Prepared Statement

The project site for this assignment uses prepared statements. A prepared statement works by taking parameter values that are transmitted later via a different protocol. Simply put, the query is sent, and then the data, thus eliminating the possibility of SQL Injection.

This type of statement has been implemented throughout the site in every function that requires user input.

2.2. Impact of Broken Access Control and Potential Security Implementations

Broken access control can have a massive negative impact on a site's data security as can be seen by the 2014/15 attacks on Snapchat and Facebook Business Pages [3]. When handling the data of other people, it is imperative that security is maintained. Such a breach for the project site could result in user account details being leaked and potentially stolen, as well as access to unused or hidden tables in the database that house sensitive data (i.e., in this scenario the unused table 'finances')

Implementing User Access Control is a technique above the scope of the developer's current ability. However, the developer has implemented a rudimentary system to protect hidden pages in lieu of a full-scale system. The idea behind this is to provide a base line of protection so that unauthorised users cannot access certain pages that require a user to be signed in.

```
if(isset($_SESSION['sid'])) { // if statement to show the page only if the user is logged in, or will redirect.
```

Figure 4- If Statement Used to Provide Basic User Access Control

implementing such a mechanism would greatly improve the security of the website and the database it relies on. The main goal here would be to implement such a system throughout the entire site. This would be done by creating an additional field in the users table that denotes their access level something like the following example:

usersID	usersUid	usersPwd	usersAccess
1	User1	test	admin
2	User2	Password!	user
3	User3	Pizza123	user

After adding this field, the developer would need to implement checks to see the current users access level and make a decision on which page(s) or features they should have access to. This would be done by using the isset function and the SESSION method to perform a validation check in each protected page to determine the access rights of the current user. However, implementing this is outside of the developer's ability. After trying this method, it has been found that the developer is unable to get the authentication to work successfully, hence the rudimentary system with no varying access levels. Ideally the developer would want to use the varying levels of access so that admins can make changes to the site without disrupting regular user access of the site, and so that users do not have access to admin level functionality.

3. References

- [1] L. Vaas, "Hackers sentenced for SQL injections that cost \$300 million," Naked Security by Sophos, 19 2 2018. [Online]. Available: <https://nakedsecurity.sophos.com/2018/02/19/hackers-sentenced-for-sql-injections-that-cost-300-million/>. [Accessed 20 11 2022].
- [2] W3 Schools, "SQL Injection," [Online]. Available: https://www.w3schools.com/sql/sql_injection.asp. [Accessed 23 11 2022].
- [3] Contrast Security, "PRIORITIZING APPLICATION SECURITY RISK MANAGEMENT WITH THE CONTRAST RISKSCORE," Contrast Security, 2021.
- [4] I. Chua, "Real Life Examples of Web Vulnerabilities (OWASP Top 10)," Horangi Cyber Security, 4 1 2022. [Online]. Available: <https://www.horangi.com/blog/real-life-examples-of-web-vulnerabilities>. [Accessed 1 12 2022].

Assignment 1 Report

GRADEMARK REPORT

FINAL GRADE

59/100

GENERAL COMMENTS

Instructor

WEBSITE

Database structure and design - catDesc and catDev need normalising - there is a lot of repeating data in this field. 6/10

Range of PHP demonstrated -

Signup needs some validation - I submitted it blank and it brought up some errors - it looks like you have written a function to cover it but it hasn't kicked in. Validation is OK on login section.

A well managed login generally, although I was able to access the catedit.php page via the URL while not logged in.

On the Edit section - I'm not sure exactly what I'm editing. It's awkward for the user to remember the name of the plugin and get it exactly right - you need to have an edit button next to each record on the previous page and populate the fields on catedit.php. I did not manage to edit a record.

Can't find any delete or search facilities?

27/50

Overall integration of website - I filled in the Contact form - would be nice to have some user

feedback such as "Thank you for contacting us" after submission. Generally a good layout and navigation structure. 7/10

REPORT

Investigation/analysis of two security issues -

SQL Injection is generally good, I would have liked a few more examples other than the 1=1 problem that was covered in class. ("=" in your case, but same principle!) Broken access is a good topic I would have liked a bit more technical detail on how/why it might occur 10/15

Mitigation of security issues in your site -

Prepared statements have been shown well. The example given for Broken Access was covered in class, shame you couldn't get it working. A few more examples would have been good here. too 9/15

General Comments: All academic reports should be structured with an introduction and conclusion - a short one would be OK here given the low word count

TOTAL: 59/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6
