

Database-Driven Website

1st Given Edward Patch
Software Engineer Student (of HND Year 2)
Database and Application Development
University Trinity St. Davids (of Gayner Thomas)
Swansea, Wales
Student ID: 1801492

CONTENTS

I	Introduction	1
II	Website Features and Security Risks	1
II-A	Login	1
II-B	Show Products and Delete Products . .	1
II-C	Edit Products:	1
II-D	Search Products	2
III	Security	2
III-A	SQL Injection (SQLi)	2
III-B	Password Encryption	2
IV	References	2

Abstract—Security analysis and dynamic website is documented, which works and communicates with a MySQL database.

Index Terms—mysql, security, sql injection, php, js

I. INTRODUCTION

In this report, security analysis and design of a dynamic website explained and analysed—the website named ‘Stay Safe’, designed to work inside a mobile application. The website is optimised for both desktop and mobile. As the application side of Stay Safe is aimed towards customer use, the website side is aimed towards staff members. Product and order control achieves this.

Languages/Templates used:

- HTML/CSS
- PHP
- JavaScript
- Bootstrap

II. WEBSITE FEATURES AND SECURITY RISKS

A. Login

The login page will redirect, based on the user role. If the user is a student, they get turned to the Stay Safe’s GIT Repository.

If the user is a staff member, it will show the dynamic website’s Add Products page. However, if the login failed, the user would get redirected back to the login page.

Security Risks

A SQL query that selects the Accounts table is required for the PHP login script. To reduce **SQL injection** risks in the login script, SQL Injection (SQLi) Prepared Statements are used. Refer to section III-A SQL Injection (SQLi); Page 2, password encryption provides two protected areas. If the PHP script were to be intercepted by SQLi or other means of hacking, consumers password would be at risk. The second protection method would be if a database engineer accessed these tables to generate reports or gain unauthorised access to the database. At least the user passwords are encrypted.

B. Show Products and Delete Products

Show Products page shows a list of categories initially, pulled from the Stay Safe database. Using a mix of JavaScript and PHP, the use of collapsible products helped keep the page modern and clean. **Delete Products** feature is located next to the Edit button, under the product container. The deleted product asks for confirmation to delete a product, upon the first request. After the verification is confirmed, the product record is deleted from the database, via the product ID.

Security Risks

The security risks are limited. The SQLi Prepared Statements can play a part, to improve performance rather than for security purposes. It is not required for security in this scenario, the Show Products and Delete Products forms/queries does not ask for user input. This case indicates that the data is safe, compared to log in, search and edit form.

C. Edit Products:

The Edit Product form stored is on the same page as Show Products. The edit products form shows below the products, collapsing the product list.

Security Risks

This method requires access to the Products table in the Stay Safe database. The Products table doesn’t hold sensitive information, compared to the Orders and Accounts table. Although, by practising SQLi Prepared Statements, it increases understanding of how security measures work. Prepared statements are easy to read and code on the developer’s point of view. Refer to under SQLi Prepared Statements Performance, under section III-A SQL Injection (SQLi); Page 2, to read how SQLi Prepared Statements affect performance time.

D. Search Products

The navigation bar contains a search form. The user can search for products that have specific attributes in its data fields. This includes data fields that are, but not limited to:- Product Names, Allergies, Calories. The search bar uses % logic, which in SQL, counts as a part of a word.

Security Risks

The security risks of Products is similar to the edit forms. The staff does not need to break the search to show all products, as they already have access to this information. However, SQLi Prepared Statements were used regarding the performance and practice side of PHP scripting.

III. SECURITY

A. SQL Injection (SQLi)

According to acunetix [1], the attacker can add to a statement 'OR 1=1' by using SQL Injection. acunetix [1] uses the example as follows:

```
SELECT id FROM users WHERE username='username' AND password='password' OR 1=1
```

The example above, will query all the records, from the first ID. Hacksplaining [2] and acunetix [1] suggests, by using PHP's Prepared Statement, it can prevent SQL Injection. Due to using the Prepared Statement method, the function checks data integrity and prevents illegal moves.

Login, Add, Edit, Search submission PHP scripts are where MySQLi Prepared Statements used. A query statement could look like:

```
SELECT * FROM accounts WHERE username = ?, password = ?
```

Instead of:

```
SELECT * FROM accounts WHERE username = '$username', password = '$password'
```

SQLi Prepared Statements Performance

Based on the Blog, MySQLi Prepared Statements [3], Prepared Statements escape 13% faster than standard statements, when escaping. Whereas, Prepared Statements, without escaping perform, 8% faster than standard statements. This indicates that SQLi Prepared Statements offer excellent performance, as well as SQLi security.

B. Password Encryption

Password Encryption is important to provide safety to the users that sign up to the service. PHP offers BCrypt, used in the Application Side of Stay Safe. When logging into the website, the PHP's password_verify function will compare plain text entered from the user input and the encrypted password stored on the webpage. If the function returns a true Boolean, then the user can login.

The examples are exemplified of how the passwords are encrypted before stored:

```
$2a$11$1P7jTHANGwerhoA6rzQMOiGHbOf0AbvnBAxEUbnVNUqAhgGpRf6Tm
$2a$11$QuzaKTayy55oj6hsiDq3HuYkZecnBID15uI4hoZ88kWH7K0xHZdES
```

IV. REFERENCES

REFERENCES

- [1] acunetix, "What is SQL Injection (SQLi) and How to Prevent Attacks." [Online]. Available: <https://www.acunetix.com/websitesecurity/sql-injection/>
- [2] Hacksplaining, "Protecting Against SQL Injection." [Online]. Available: <https://www.hacksplaining.com/prevention/sql-injection>
- [3] "PHP performance and memory usage: MySQLi prepared statements," Aug. 2012. [Online]. Available: <https://we-love-php.blogspot.com/2012/08/mysqli-prepared-statements.html>
- [4] NishanthReddy7, "How to delete a file using PHP ?" Jul. 2019, section: PHP Programs. [Online]. Available: <https://www.geeksforgeeks.org/how-to-delete-a-file-using-php/>
- [5] "PHP: session_destroy - Manual." [Online]. Available: <https://www.php.net/manual/en/function.session-destroy.php>