International Journal of
Advances in Scientific Research and Engineering (ijasre)

E-ISSN : 2454-8006

DOI: 10.31695/IJASRE.2019.33615

Volume 5, Issue 11
November - 2019

Mitigating SQL Attacks on Enterprise Database

MADUAKO Nnanyereugo C[1], OWOLABI Olumide[2]

Research Scholar[1], Lecturer[2]

[1-2]Department of Computer Science,

University of Abuja

Nigeria

ABSTRACT

In today's world, data is generated at a very rapid speed and final destination of such data is the database. Database security assures the security of the information stored in the database against threats, be it insider or outsider threats. Data is stored in the database for easy and efficient way to manage these data. Considering the importance of data in an organization, it is absolutely essential to secure the data stored in the database. A secure database is one which is shielded from different possible attacks. For data protection, enforcement of access control policies based on data contents, subject qualifications and characteristics and other relevant contextual information, such as time mechanisms are used. Security models are required in the design and development of effective and efficient database systems. In this work, some of the attacks and threats that are encountered in database systems and the corresponding counter-measures, as well control methods were discussed. Ensuring security for database is a very critical issue for companies. Hence as the complexity of database increases, we may tend to have more complex security issues of database. The effectiveness of the developed system in protecting data stored in a database was demonstrated by attempting some attacks on the database. The system was able to thwart the attacks.

Key Words: SQL Injection, Database Attack, Security, FTP Solution, Mitigation.

1. INTRODUCTION

Information plays an important role in any organization, hence its protection against unauthorized disclosure (secrecy), and unauthorized or improper modifications (integrity), while ensuring its availability to legitimate users (no denials-of-service) is of paramount importance [1]. A database can be defined as a collection of data that is saved on a computer system's hard drive. Databases allow any authorized user to access, enter and analyze data quickly and easily. It's a collection of queries, tables and views. The data stored in the database are usually organized to model aspects that support processes that require information storage and retrieval. Major chunk of data is stored in the repository called database. The user interface for databases is called a Database Management System (DBMS). DBMS is a software application that interacts with the authorized user, other applications and the database itself to capture and analyze data. It helps to organize data for better performance and faster retrieval by maintaining indices [2].

2. STATEMENT OF THE PROBLEM

Think of a storage place for a manufacturing organization where it keeps all the important raw materials required for making its products. The organization uses these materials to run its production. The organization cannot perform its day-to-day operations without access to the materials. Will the organization allow anyone to walk into the storage facility off the street and have access to the materials? Unless the place is properly secured, unauthorized persons can enter and steal or destroy the materials. Not even all authorized persons may be permitted access to all parts of the storage area. Sensitive and critical materials maybe off-limits to most people.

For a modern organization, its database system is even more significant than many other types of assets. Many organizations such as financial institutions and travel companies cannot survive even a single day without their database systems. Any type of destruction of or unauthorized access to the database system has serious impact. Obviously, an organization must ensure that its database system is adequately guarded against accidental breaches of security or theft, misuse, and destruction through malicious intent.

3. AIM AND OBJECTIVES OF THE STUDY

The aim of this research is to demonstrate how the database system of an enterprise application that collects user input for report or processing results, can be made secure against SQL injection and to improve the filtration level of a user input from real one and a malicious one on a web application.

The objectives are as follows:

- To design sales report website for Unilever Nigeria Northern Region with two login pages. One will be programmed insecure to demonstrate how a login page can be attacked with SQL injection, while the other login page will follow the entire programming standard to protect the login against SQL injection attacks and the code will be tested to see if it works.
- To design a user interface front end of the proposed system using twitter bootstrap technology comprising of HTML (Hypertext Markup Language), CSS (Cascading Style Sheet), and JQuery framework of javascript.
- To design a sales report database with the login information and the report of sales by supervisors of the organization gotten from the excel spread sheet by one of the Unilever supervisors. This database is for testing the attack mitigating techniques discussed in this work.

To demonstrate the effectiveness or otherwise of these attack mitigating techniques.

4. LITERATURE REVIEW

A security strategy is an overall plan to mitigate risk. While many organizations today have a security strategy, what is sometimes missed or not adequately addressed is database security. A security strategy must mitigate the overall possibility of harm or loss to a company's data. Furthermore, a security strategy must address the business data concerns from a legal, statutory and contractual perspective. Multiple regulatory requirements and standards, such as PCI, HITECH/HIPAA, GLBA, SOX, privacy laws and others have required organizations to address information security risks. According to the SANS Database Audit and Compliance Survey, 74 percent of the respondents felt their organizations utilized data that is or might be considered regulated. Additionally, contracts are increasingly requiring organizations to consider security at all layers. Addressing database security in a proactive manner can save organizations a significant amount of money and reduce the overall risk exposure [3].

4.1. Database Overview

A database is a self-describing collection of integrated records. A record is a representation of some physical or conceptual object. A database is self-describing in that it contains a description of its own structure. This description is called metadata- data about the data. The database is integrated in that it includes the relationship among data items, as well as including the data items themselves [4].

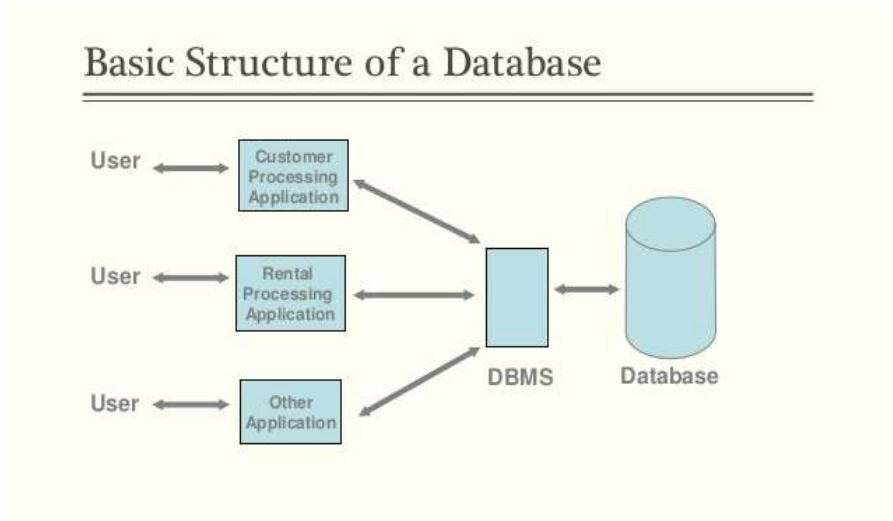

Fig. 4.1: Basic Structure of a Database

DOI: 10.31695/IJASRE.2019.33615

4.2. Database Security

Database security issues exist as long as database storing has expanded dramatically. Viruses, hackers, worms, trojans, phishing, malware, and many other challenges threaten the security of databases. Many organizations today tend to focus on using technologies for building, maintaining, and accessing databases while not devoting much focus to security [5].

Many organizations are under pressure about their security as hackers find more sophisticated ways to find vulnerabilities to exploit, in order to gain valuable data from databases [6]. Currently, 83% of the organizations in the U.S. "believe they have made their data safer by installing or upgrading antivirus software, installing or upgrading a firewall, implementing intrusion detection/prevention technologies, and implementing vulnerability/patch management systems on their networks" as reported by Communications News [6] p.8). The truth is that no matter how sophisticated the protection becomes, so do the methods for breaking into databases and acquiring precious data from a database. In 2005, the number of security breaches reported reached 100 million and is still growing according to Privacy Rights Clearinghouse, who is a non-profit consumer information and advocacy program [6].

Currently, many organizations use passwords to prevent unwanted individuals from gaining access to databases. This of course is a very common method of security, but it is not always effective. Many passwords used, unfortunately are not effective, because many of them are too easy or common for an intruder to discover [5]. In order to have an effective password, it is recommended that a password be at least twelve characters long and include capital letters and numbers [5]. However, though a more complex password can help to protect the system, it also creates a new problem for users, who, too often seem to forget such passwords.

One way (recommended) to improve database security is by adding security layers. Multiple security methods are deployed here to protect the system instead of using only just one. The methods can include having firewalls, access controls, passwords, and encryptions as well as various monitoring systems all of which would be simultaneously active [5]. This approach is ideal for databases, which have large amounts of sensitive data to protect.

Currently, there are database security tools that are being used today in order to test the security on a network. Symantec has a security tool that is being used by hospitals in the Boston area, since 2006 for alpha testing. Though this tool does not stop unwanted intruders from entering the network, it monitors all activity on the network and can show an organization if suspicious activity is present. It can also alert an organization if someone is attacking a database and trying to alter or steal information. Although this tool does not directly protect the database, it is still a huge improvement over previous database tools, which did not monitor network activity, nor did they test to see whether a database or its associated network is safe [7].

4.3. Database Threats

Databases today are facing different kind of attacks. Before describing the techniques to secure databases, it is preferable to describe the attacks which can be performed on the databases. The major attacks on databases can be categorized as follows:

Excessive privilege abuse: this is described as users or applications being granted database access privileges that exceed their required job functions.

Granting excessive permissions are problematic for two reasons: about 80% of the attacks on the company data are mostly executed by employees or former employees. When too many granted privileges are not revoked in time, it makes it unnecessarily simple for those having the privileges to execute their mischief.

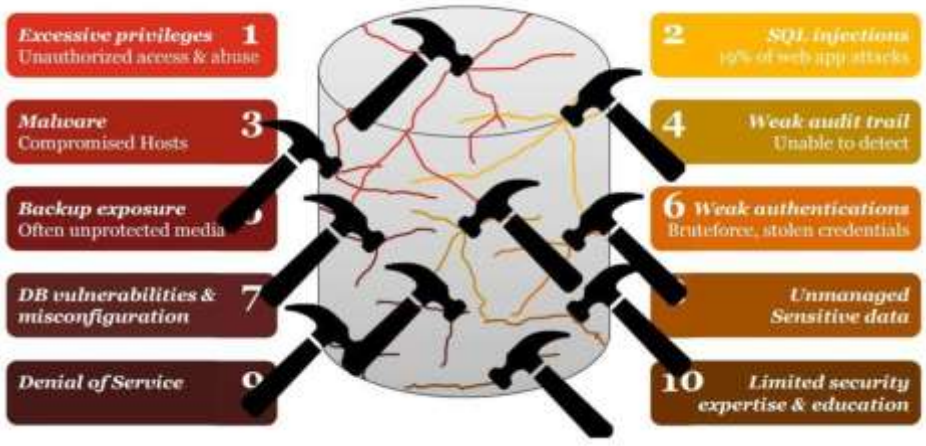

Fig. 4.2: Database threats

DOI: 10.31695/IJASRE.2019.33615

4.4. SQL Injection

The class of database vulnerabilities known as SQL injections continues to present an extremely high risk in the current network threat landscape. In 2011, SQL injection was ranked first on the MITRE Common Weakness Enumeration (CWE)/SANS Top 25 Most Dangerous Software Errors list. Exploitation of these vulnerabilities has been implicated in many recent high-profile intrusions. They are one of the biggest threats to databases, much like web apps. They can be launched on either the database or the web app that acts as a front-end to the database, yet due to the prevalence of SQL injection flaws in web apps and how easy they are to exploit, they are more common than attacking the database. SQLi occurs when input is unsanitized before being executed in the database or web app hosting the database, and attackers crafting a malicious input would allow them access to sensitive data, give them escalated privileges, and in especially dangerous exploits, access over the databases operating system commands and the database itself.

4.4.1 Causes:

SQLi simply stated are caused by software applications that accept data from untrusted sources (internet users), but fail to properly validate and sanitize the data, and subsequently use that data to dynamically construct an SQL query to the database backing that application. Example, imagine a simple application that takes inputs of a username and password, which may ultimately process this input in an SQL statement of the form:

String query = "SELECT * FROM users WHERE username = "" + username + " ' " password = ' " + password + " ' ";

Since this query is constructed by concatenating an input string directly from the user, the query behaves correctly only if password does not contain a single-quote character. When the user enters "joe" as the username and "nathan" OR 'a'='a as the password, the resulting query becomes:

SELECT * FROM users WHERE username = 'joe' AND password = 'nathan' OR 'a'='a';

The "OR 'a'='a' clause always evaluates to true and the intended authentication check is bypassed as a result.

4.4.2 Impacts:

Many of the high-profile intrusions in which SQLi has been implicated have received attention because of the breach of confidentiality in the data stored in the compromised databases. This loss of confidentiality and the resulting financial costs for recovery, downtime, regulatory penalties, and negative publicity represent the primary immediate consequences of a successful compromise.

In Ponemon's SQL Injection Threat Survey of 2014, 65% of organizations surveyed had experienced a successful SQL injection attack in the past year alone. 47% of the respondents either did not scan for active databases or scanned irregularly and 49% of respondents rated the threat level of an SQL injection occurring in their organization a 9-10 rating.

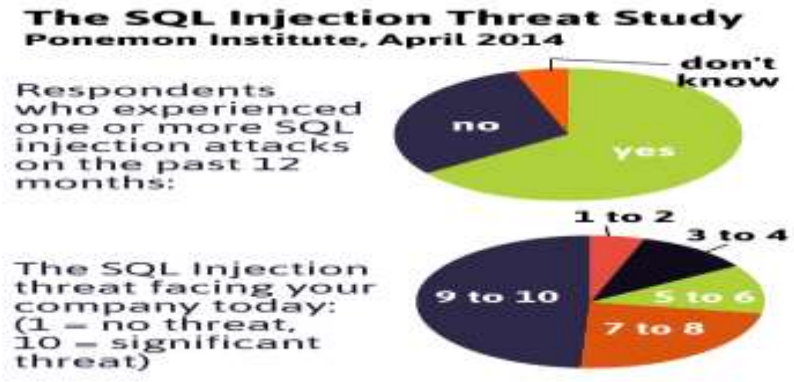

Fig. 4.3: SQL Injection threat study

5. REVIEW OF THE EXISTING SYSTEM

Injecting a web application is the synonym of having access to the data stored in the database. The data sometimes could be confidential and of high value like the financial secret of a bank or list of financial transactions or secret information of some kind

DOI: 10.31695/IJASRE.2019.33615

of information system. An unauthorized access to this data by a crafted user can be a threat to their confidentiality, Integrity, and authority.

5.1. Demerit of the Existing System

As a result, the system could bear heavy loss in giving proper services to its users or it may face complete destruction. Sometimes such type of collapse of a system can threaten the existence of a company or a bank or an industry. If it happens against the information system of a hospital, the private information of the patients may be leaked out which could threaten their reputation or may be a case of defamation.

The Unilever Sales Report application is like a simulation of how to prevent SQL injection attacks on login forms of any web application. The application developed is for only authorized user (who has the username and password for access), to view the sales of two commodities in two days.

5.1.1 Public Area

The public area is composed of two login forms for only authorized user to access, one is secured against SQL injection, while the other is programmed in an insecure way that allows SQL logics to break through the login form and give the user access to the company data. The system is illustrated in figure 5.1

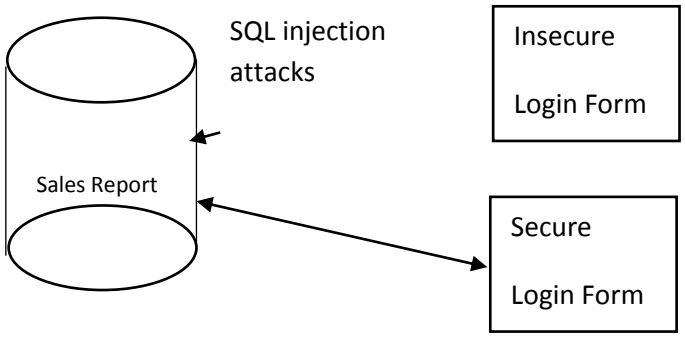

Fig. 5.1: Unilever Sales Report Application

5.1.2. Insecure Login Form

The user is required to enter a username value: “user” and a password value: “test” before access is given, but a hacker can bypass this form by entering the values as follows: username: ‘ OR ‘1’ = ‘1’ and password: ‘ OR ‘1’ =’ 1’. Both of these values will return true in the SQL query for the username and password and give access to the sales report as follows:

"SELECT * FROM login WHERE username='' OR ‘1’=’1’' AND password='' OR ‘1’=’1’'"

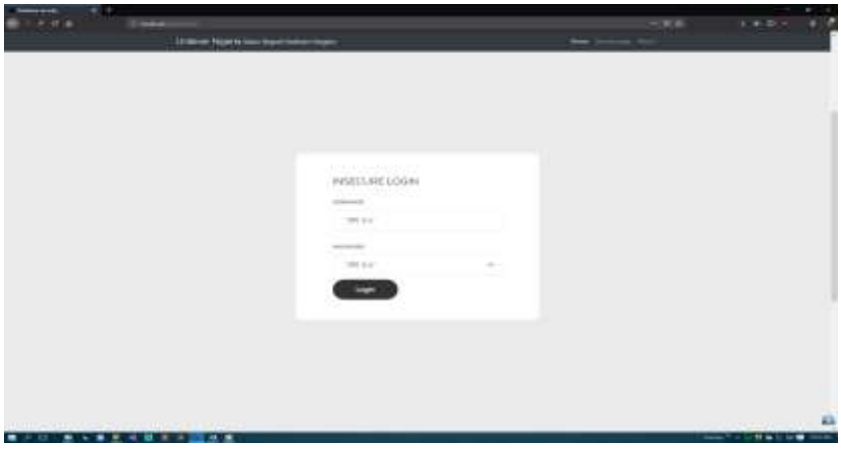

Fig. 5.2: Showing the insecure login form

5.1.3. Secure Login Form

The solution to prevent SQL injection attack on a login page is the use of PHP parameterized functions to clean out or disallow any addition kind of character that will be added to the username or password input on the form, such as:

DOI: 10.31695/IJASRE.2019.33615

- tim() function: is used to remove the white spaces and other predefined characters from the left and right sides of a string.
- preg_replace('/[\s\t\n\r\s]+/', ' ',$value): perform a regular expression search and replace it with a null value
- strip_tags($value): used to strip a string from HTML and PHP tags and return a string with all NULL bytes from a string value in variable $value
- htmlspecialchars($value): converts some predefined characters to HTML entities such as – &(ampersand) becomes & “(double quote) becomes &quot: ‘(single quote) becomes &#039, < (less than) becomes &lt. and > (greater than) becomes &gt.

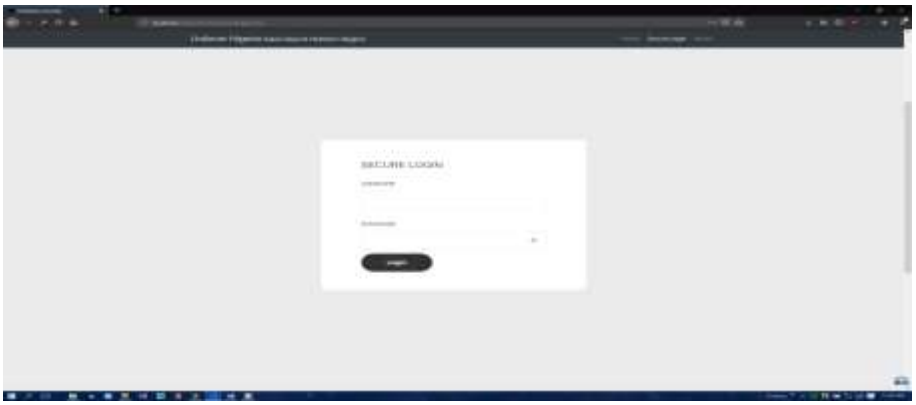

Fig. 5.3: Shows the secure login form

6. DATA MODELLING

6.1. Data Collection

It was very challenging getting an organization to give out their business information for research purpose, but some data was given by a sales supervisor northern region for Unilever Nigeria in an excel spread sheet, and the data on the spread sheet were entered into the database created for the application with SQL queries. The sales data spread sheet is shown in figure 6.1 and 6.2.

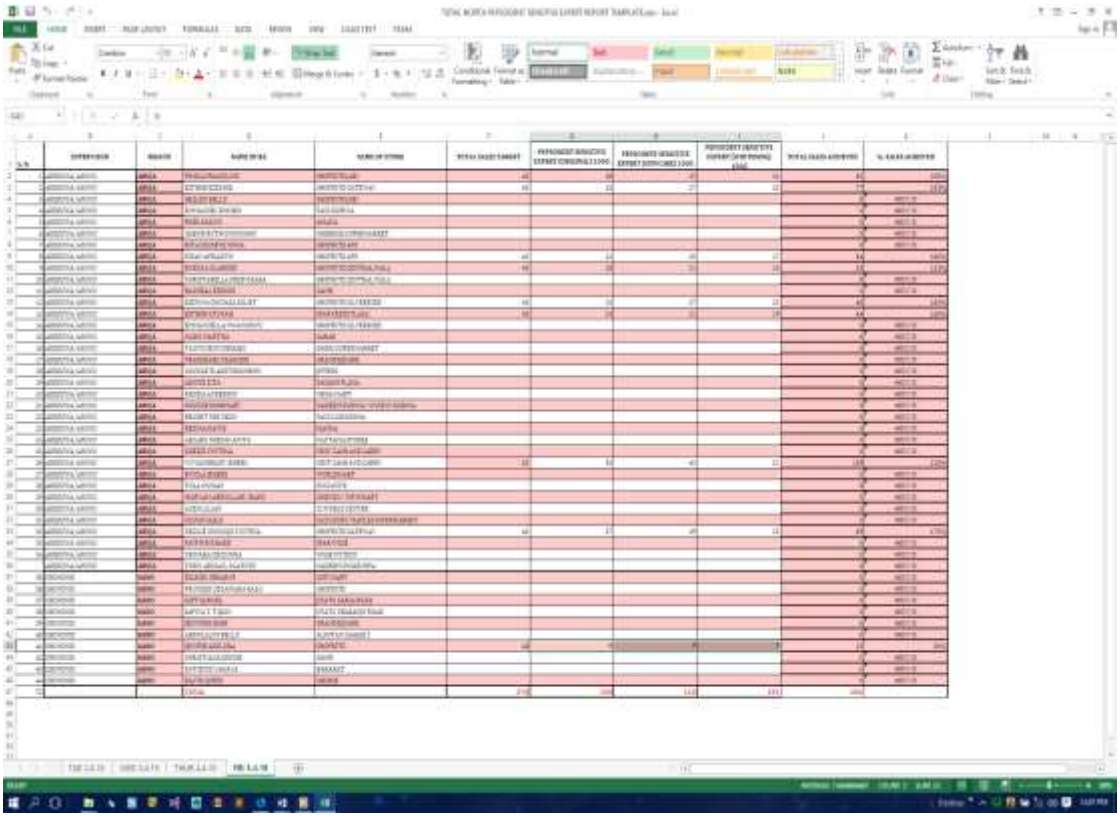

Fig. 6.1: Pepsodent mouth wash sales report

DOI: 10.31695/IJASRE.2019.33615

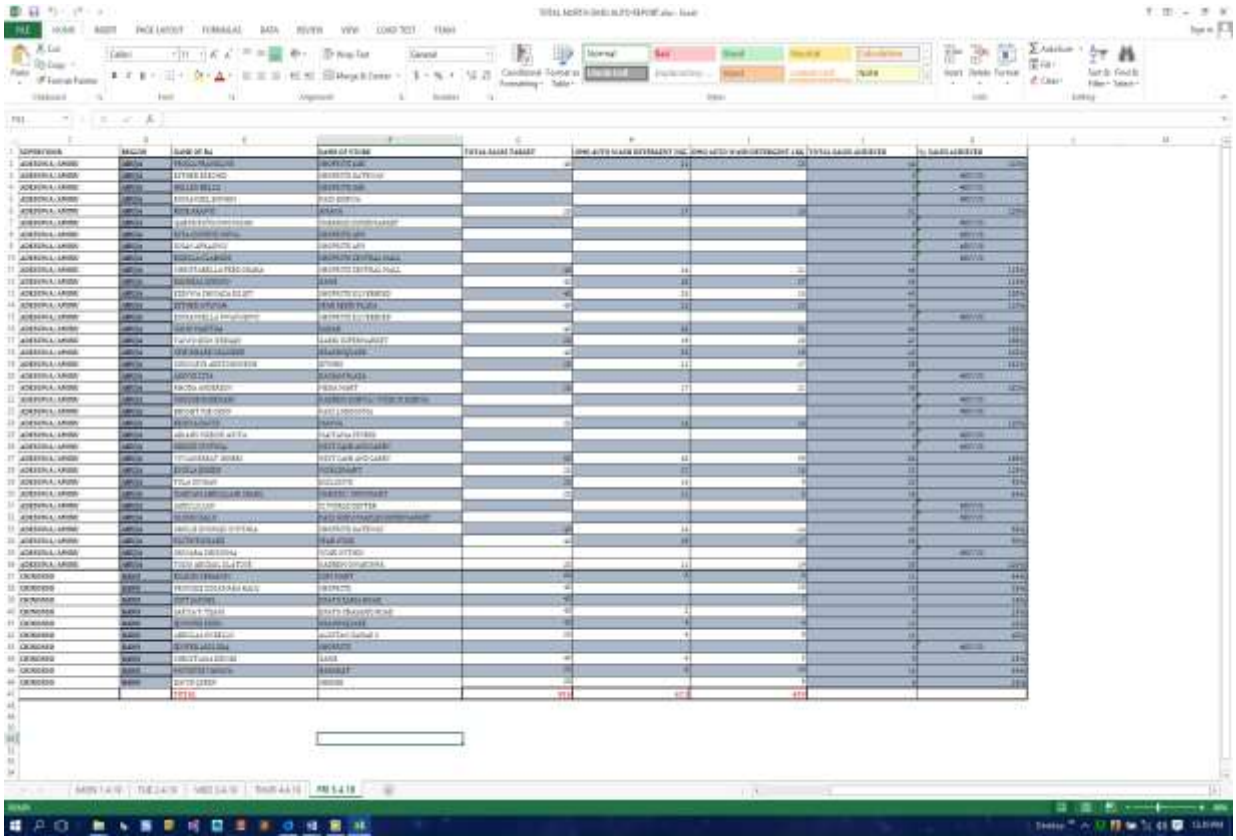

Fig. 6.2: Omo detergent sales report

6.2. Identifying Data Items

The information of the user login and sales of commodity is kept in a database called "unilever" on the MYSQL relational database management system. The followings are the names of the tables created in the database: table "login", "omo_sale", and "pepsodent_sale".

6.2.1. Database Schema

The tables created in database "unilever" are described in Table 6.1 as follows:

Table 6.1 login:

Field	Type	Null	Key	Default	Extra
Id	Int(11)	No	PRI	NULL	auto_increment
username	Varchar(25)	No		NULL	
password	Varchar(25)	No		NULL	

Table 6.2 omo_sale:

Field	Type	Null	Key	Default	Extra
Id	Int(3)	NO	PRI	NULL	auto_increment
Saledate	Date	NO		NULL	
supervisor	Varchar(25)	NO		NULL	
Region	Varchar(25)	NO		NULL	
storename	Varchar(25)	NO		NULL	
salesnumber	Int(3)	NO		NULL	

Table 6.3 pepsodent_sale:

Field	Type	Null	Key	Default	Extra
Id	Int(3)	NO	PRI	NULL	auto_increment
Saledate	Date	NO		NULL	
supervisor	Varchar(25)	NO		NULL	
Region	Varchar(25)	NO		NULL	
storename	Varchar(25)	NO		NULL	
salesnumber	Int(3)	NO		NULL	

DOI: 10.31695/IJASRE.2019.33615

6.3. Process Modelling

6.3.1. Interface Design

The interface of the sales report application is developed using frameworks such as jquery UI (user interface), Twitter bootstrap, which were all built on HTML(Hypertext Markup Language), CSS(Cascading Style Sheet), and javascript. These frameworks are what gave the pages sleek look other than the conventional HTML ugly looks. The login form interfaces were developed on the twitter bootstrap user interface framework as well as the form response.

The Sale Report Output

The report of the products sold is structured in a tabular form for easy understanding and analysis of the information. The interface is shown in figure 6.3.

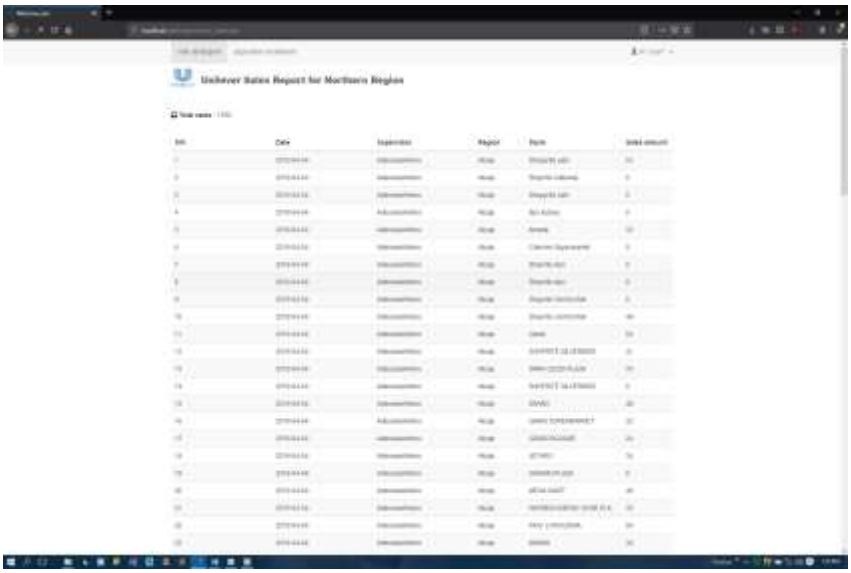

Fig. 6.3: The Sale Report output of the application

6.4. The process flow diagram of the application.

The application is just a simulation of how to write codes in PHP login form to protect against hackers using SQL injection to break through the login form to access organizational information that may be private. The insecure login is tested with the username value to be: ‘ OR ‘1’ =’1’ and password valued to be ‘ OR ‘1’ =’1’ . Then the login button is clicked and the program will see these values as true and give access to the hacker. But, when the same process is repeated on the secure login page, access is denied, due to the parameterized PHP functions to clean the values, figure 6.4 illustrates the process.

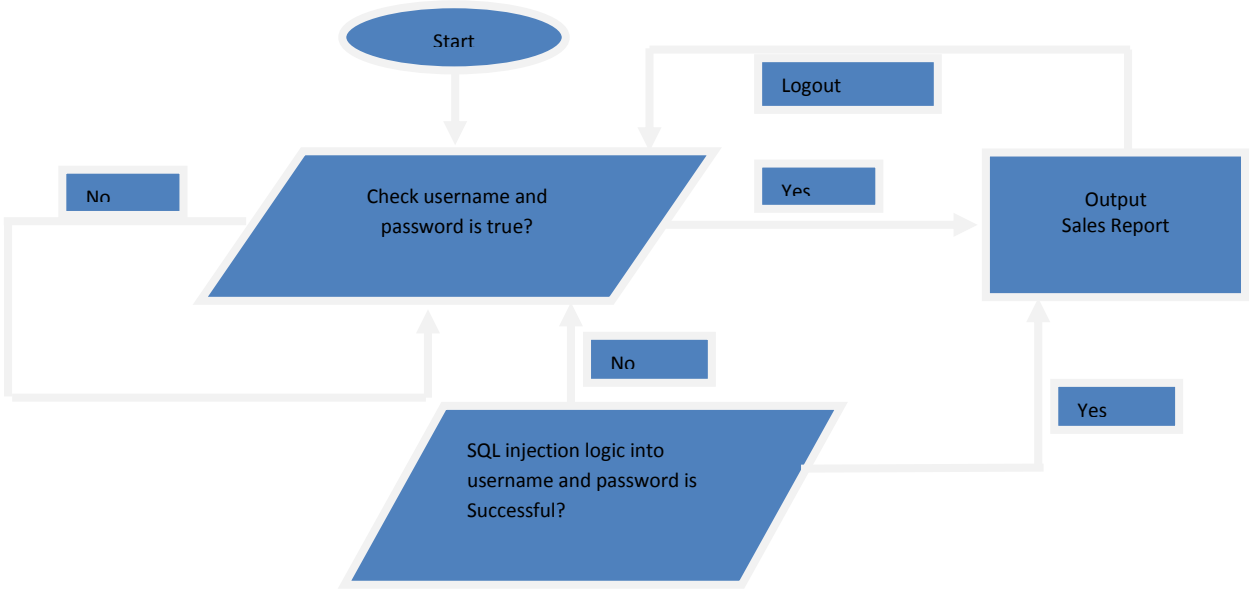

Fig. 6.4: The Process flow diagram of the application

DOI: 10.31695/IJASRE.2019.33615

6.5. Program Codes/Scripts Implementation

In this research, the browser considered in testing the application is Mozilla firefox. The front-end programs are interpreted by these browsers to create an interface between the users and the database. The Apache server processes the PHP codes used in interacting with the database and output the result to the browser with the HTML Tags. PHP (PHP: Personal Home Page) interacts with the MYSQL relational database in five steps:

1) Create a connection to the database:

The mysqli_connect() function is used in creating a connection with MYSQL relational database. The function is implemented as follows: <?PHP mysqli_connect("localhost name", "username", "password") ;?>

2) Select a database:

The mysqli_select_db() function is used in selecting one of the database names created for the project in MYSQL The following is implemented as follows: <?PHP mysqli_select_db("database name")?>

3) Perform database query:

The msqli_query() function is used to querying the database selected to respond to the user request. The function is implemented as follows: <?PHP $result=mysqli_query("make SQL query to the table")?>

4) Return the data of the query result (if any):

The mysqli_fetch_array() function is used to return the query result to the browser. The function is implemented as follows:

<?PHP while($row = mysqli_fetch_array($result)){

Echo $row ["table field"];

}

?>

5) Close connection to MYSQL:

The mysqli_close() function is used to close the connection to the database after the result of the user request have been sent to the browser. The function is implemented as follows: <?PHP mysqli_close($dbc); ?>

7. SYSTEM TESTING

The scripts are saved in the WAMP server directory folder named "www" in the hard drive of the computer. When the server has been put on, the output of the scripts is viewed by launching a browser and entering http://localhost/sitefolder into the URL (Uniform Resource Locator) bar of the browser, then click enter. It outputs the interface shown in Figure 7.1.

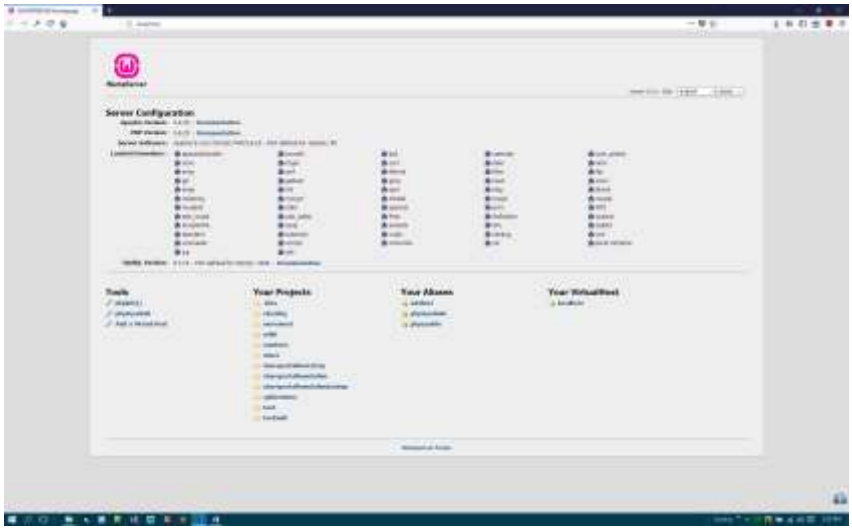

Fig.7.1: The WAMP Server Project Page

DOI: 10.31695/IJASRE.2019.33615

The project folders saved in the server are listed and when the particular project folder is clicked the web browser leads us to the index page of the web site. The file list of the project is shown below in figure 7.2.

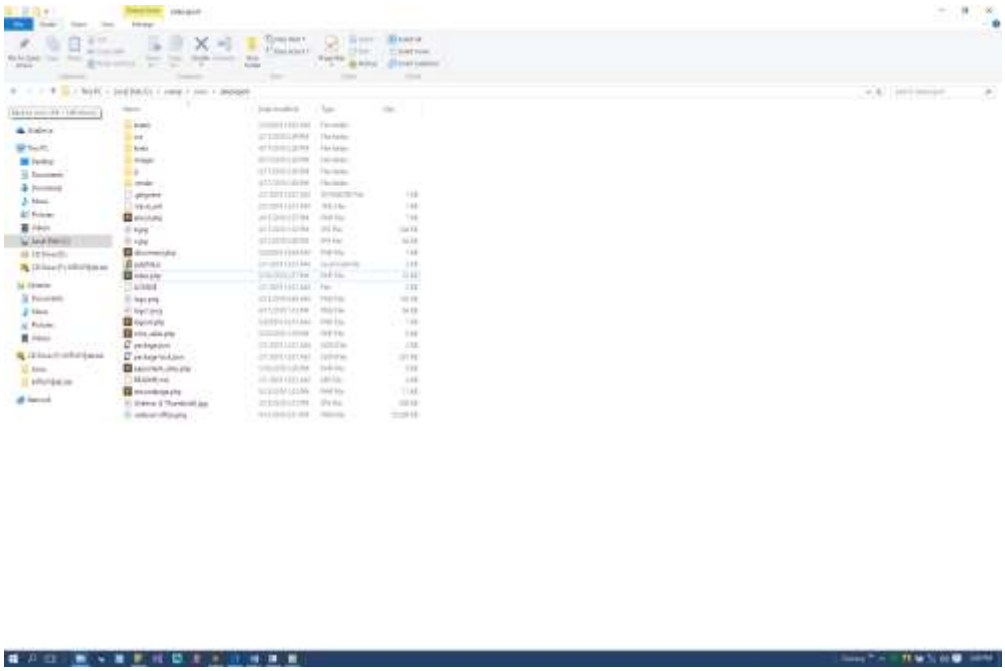

Fig. 7.2: File list of the project folder

7.1. Hosting the Application

The project is a web application that needs to be hosted on an internet server for full utilization. First it is mandatory that one gets a domain name. The domain name used is registered from a free web hosting company http://www.000webhost.com and the sub-domain name used is "naproject.co.nf". The following information below was set up for the successful hosting of the site folder:

- Hostname: files.000webhostapp.com
- FTP Username: maduakonnanyereugo
- FTP Password: nanydamko@gmail.com
- FTP Port: 21

The Filezilla interface is shown in figure 7.3.

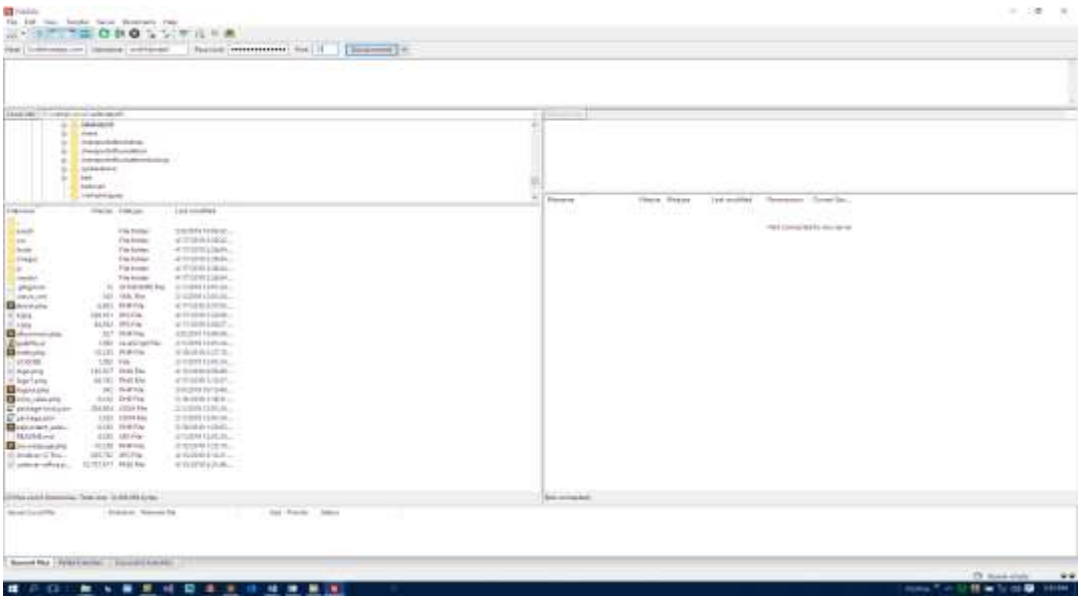

Fig. 7.3: The Filezilla FTP client software to upload website to internet server.

DOI: 10.31695/IJASRE.2019.33615

8. SUMMARY

The proposed system has been able to protect user login from hackers by using specialized functions to clean-up the values entered by users. The challenge of the development of the system was that, no organization would accent to give out their information for the purpose of the research and it took more time to get the data used for the project.

9. CONCLUSION

The proposed system will make it difficult for anyone to use SQL injection attack to break through the login form of the application. In this research, I have reviewed the most popular existing SQL Injections related issues and prevention techniques.

REFERENCES

1. Gertz M, and Jajodia S "Handbook of Database Security - Applications and Trends" Springer, 2008.
2. Malik M, Patel T, 2016 "Database Security - Attacks and Control Methods" International Journal of Information Sciences and Techniques (IJIST) Vol.6, No.1/2. pp.175-183.
3. Baccam T, 2009 "Making Database Security an IT Security Priority" A SANS Whitepaper – November 2009.
4. Kroenke, D., & Auer, D. (2007). *Database Concepts.* 3rd ed. New York, NY: Prentice.
5. Britt, P. (2007, February). Tightening Security in 2007. InformationToday. Retrieved March 23, 2007 from http://0-earch.ebscohost.com.janus.uoregon.edu:80/login.aspxdirect=true&db=buh&AN=23878734&loginpage=login.asp&site=ehost-live.
6. Petersen, J. (2007). *Absolute Beginner's Guide to Databases.* Indianapolis, ID: Que.
7. Mesmer, E. (2006, January). Caregroup checks out Symantec database security tool. Network World. Retrieved March 6, 2007, from http://www.networkworld.com/news/2006/010906-caregroup.html

DOI: 10.31695/IJASRE.2019.33615