CSE 3501 INFORMATION SECURITY ANALYSIS AND AUDIT

REVIEW 3 PROJECT REPORT

on

SQL INJECTION, DETECTION, PREVENTION AND RESPONSE ON A WEBSITE.

Prepared by

Harsh Rajpal - 20BCI0271
Mrinal Sharma – 20BCI0247
Divakar Singhal – 20BCI0261

Under the supervision of

Dr. Saritha Murali

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering
Vellore Institute of Technology, Vellore.

November 09, 2022

Table of Contents

Abstract

SQL injection is a very serious threat and it can be used to exploit user information. It is one of the most dangerous threats as described by OWASP. It has always been on the top of OWASP (Open Web Application Security Project) exploits as it is easy and very dangerous. Even after taking proper precautions, there can still be a back door to exploit the database information. It is a big threat to web-based applications and hackers can hack their way through to modify and delete data as per need. There are many ways to perform SQL injection and Several Prevention techniques as well. In an instance of an injection attack known as a "SQL injection attack," SQL instructions are inserted into data-plane input in an effort to influence the execution of prepared SQL commands. In this project, we are going to demonstrate the whole process from attack to prevention and retrieval.

1. Introduction

SQL is a commonly used database management programme that organises data and information into structures and creates connections between them. The primary purpose of SQL is to communicate with and modify databases. Hackers employ the method of SQL injection to insert code—in this case, a SQL query—into a request URL or an open port in order to modify data or increase their level of access.

The login page is typically the target of SQL injection attacks, which typically include stealing database data using straightforward commands such appending 'I' OR 'I'='I to the URL.

The hacker only needs to insert this one line of code to gain access to the database linked to the login page if the web app does not have any SQL protection measures.

A database is a collection of information and data that has been organised to make it simple to access, handle, and control. Rows, tables, and columns are used to organise the data. Additionally, it is indexed to make it simpler to find the relevant details and information. When new data and information are added, the existing data will be updated, enlarged, and deleted.

Data leaks pose a major threat to an organisation since they could damage its reputation and result in financial losses.

Database attacks are becoming more frequent as data becomes the new energy source of the twenty-first century. Additionally, hackers frequently profit handsomely from the sale of data, particularly private data like credit card numbers.

2. Literature Study

Research of SQL Injection Attack and prevention Technology [1]

This paper introduces common SQL injection attack and defence technologies. The detection techniques use type-safe SQL parameters in addition to validating user input. The detection processes are used to develop a SQL injection protection model that is effective against SQL injection vulnerabilities. One of the most dangerous security flaws in Web application systems is the SQL injection attack; most of these flaws are brought on by the usage of SQL parameters and a lack of input validation. In the article, common SQL injection attack and defence technologies are described. The detection techniques leverage type-safe SQL parameters along with user input validation. The detection processes are used to establish a SQL injection defence model that is effective against SQL injection vulnerabilities.

Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention [2]

The storage of the enormous volume of big data Internet exchanges originating from cloud-hosted web apps and Internet of Things (IoT) smart devices depends critically on the back-end database. Intruders continue to use the SQL Injection Attack (SQLIA) attack of choice on weak online applications to steal private information from the database, potentially leading to negative outcomes. The solutions that are now available, which are mostly signature techniques, were all developed before the recent issues of big data mining and as a result, lack the functionality and capacity to handle new signatures that are masked in web requests. In order to identify and prevent SQLIA, a substitute machine learning (ML) predictive analytics offers a functional and scalable mining of massive data. Unfortunately, it is a well-known problem in SQLIA research that there aren't any ready-made strong corpora or data sets with patterns and historical data items to train classifiers. The development of a data collection featuring extraction from well-known attack patterns, such as SQL words and symbols present at injection locations, is explored in this article.

SQL Injection Attacks Prevention System Technology [3]

In this paper, we review PHP techniques and other techniques for protecting SQL from the injection, methods for detecting SQL attacks, types of SQL injection, causes of SQL injection via getting and Post, and prevention technology for SQL vulnerabilities. Hackers can access private and sensitive data thanks to the flaws in the majority of online apps. Web applications are significantly threatened by structured query injection, one of the most prevalent and pervasive information theft techniques. When users' input is not filtered for certain special characters and symbols contained within structural query sentences, or when the quality of the information, whether it be textual or numerical, is not checked, hackers can take advantage of flaws in system design or existing gaps. This leads to unpredictability in the implementation's results.

A SQL Injection Detection Method Based on Adaptive Deep Forest [4]

This paper proposes an adaptive deep forest-based method to detect the complex SQL injection attacks. SQL injection may seriously affect the network due to its many variants and quick changes, which can lead to data leaking and website paralysis. SQL injection detection is still a difficult challenge because of the heterogeneity of attack load, the diversity of attack tactics, and the variety of attack modes. The emphasis and frontier of online security nowadays is how to defend against SQL injection attacks successfully. As a result, this research suggests a deep forest-based adaptive technique to identify sophisticated SQL injection assaults. In their study, the deep forest structure is first optimised, and each layer's input is composed of the average of previous outputs and the raw feature vector. Experiments demonstrate that their suggested approach successfully addresses the issue that the with more layers, deep woods lose some of their original characteristics.

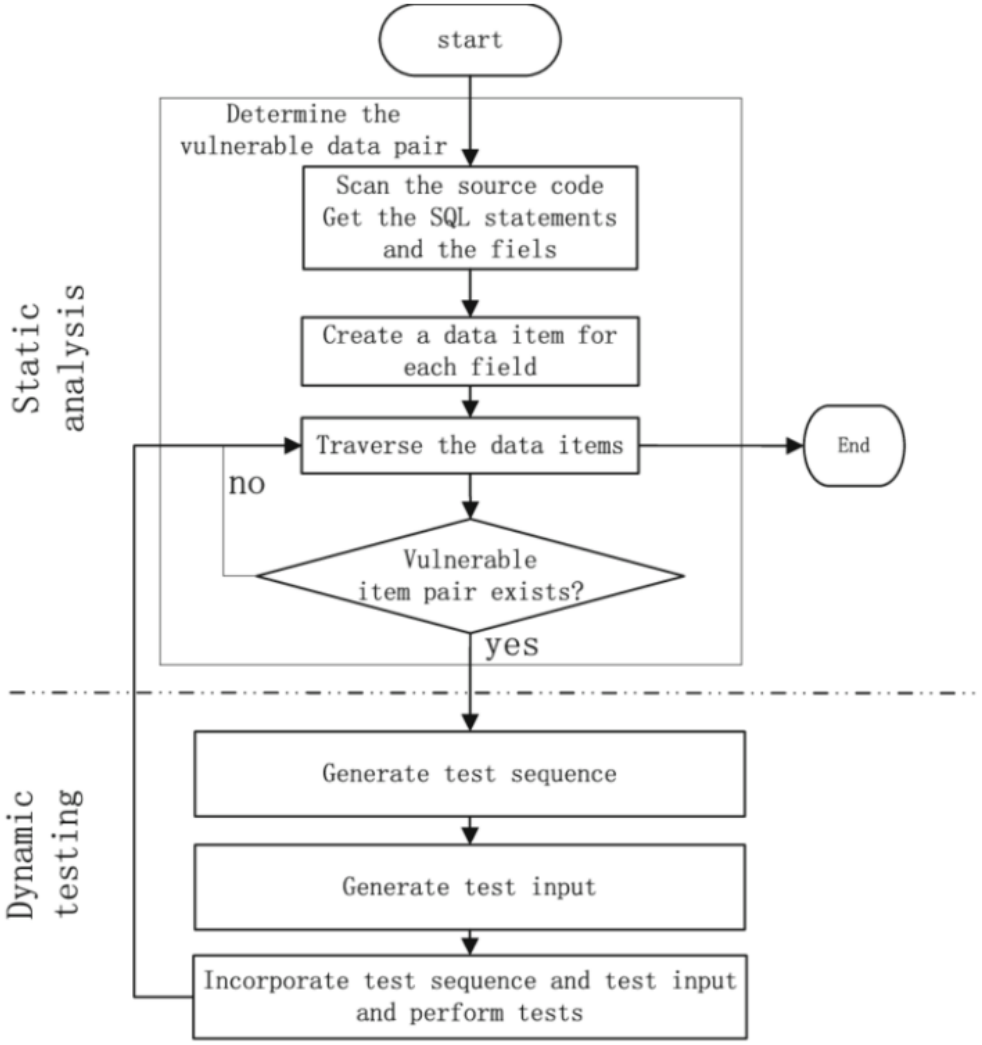
3. Problem Statement and Objectives

A extremely dangerous threat, SQL injection can be used to exploit and steal user information. According to OWASP, it is among the most dangerous risks. It has consistently ranked first among the top 10 OWASP exploits since it is both simple and extremely risky. There may still be a back door to the database information even after taking the necessary protections. Web-based applications are at serious risk because hackers can get in and change and destroy data as needed. There are several prevention strategies as well as numerous ways to carry out SQL injection. In this project, we'll show how the entire procedure works, from attack to defence to recovery.

In this project, we intend to imitate a hack, stop one, get our website back, and safeguard it. SQL injection is used in hacking, and vulnerabilities were scanned and eliminated.

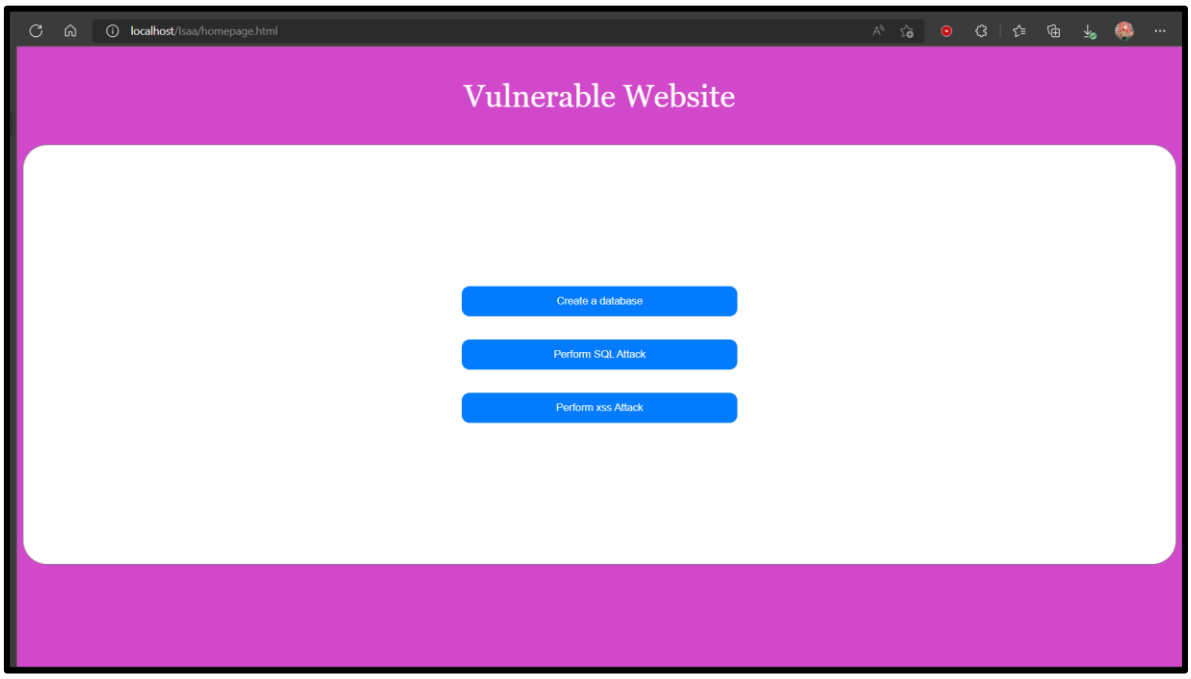
The theft of data from a database connected with an online-based app is one of the worst examples of SQL injection. Sometimes, SQL injection occurs as a result of a web-based vulnerability, an application, or a user's ignorance of database security. There are various ways that SQL can be injected so that outside hackers can use it. The protection of web-based applications must receive significant attention in order to prevent SQL Injection and to guarantee that all data is kept secure in the database.

4. Design

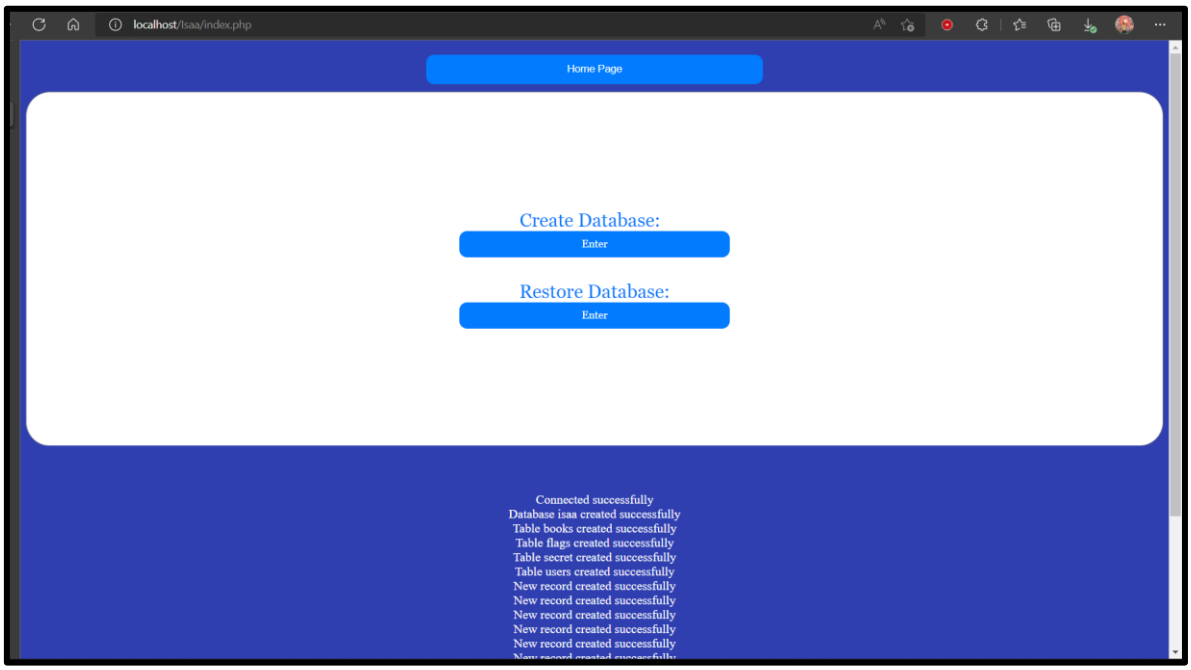

start
Determine the vulnerable data pair
Scan the source code Get the SQL statements and the fiels
Create a data item for each field
Traverse the data items
End
no
Vulnerable item pair exists?
yes
Static analysis
Dynamic testing
Generate test sequence
Generate test input
Incorporate test sequence and test input and perform tests

5. Implementation

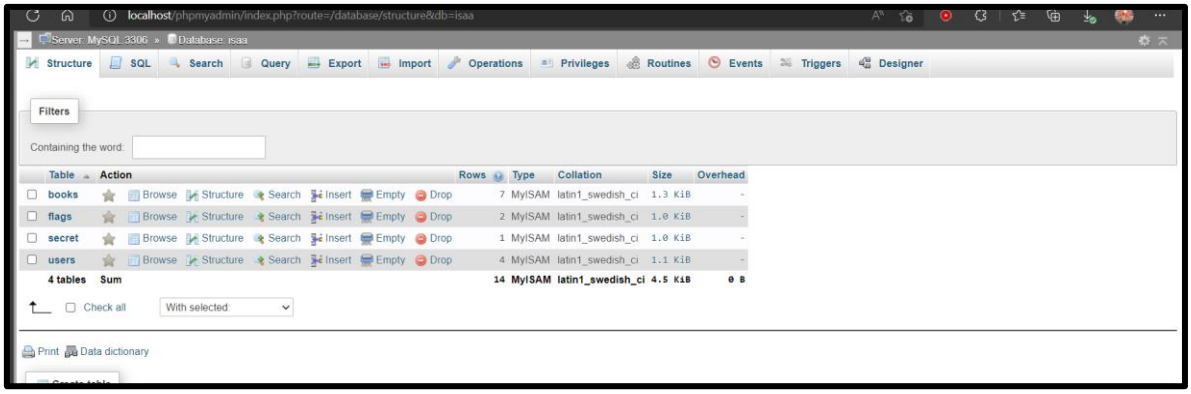
Homepage:



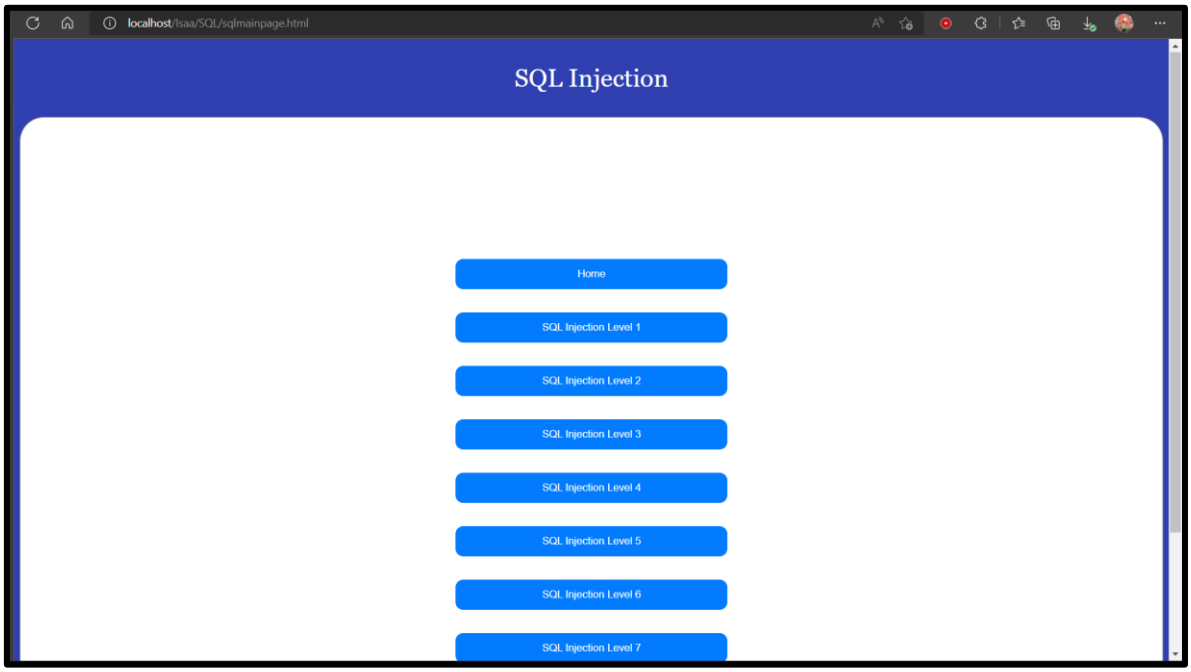
Creation of database and insertion of records to perform SQL attack:



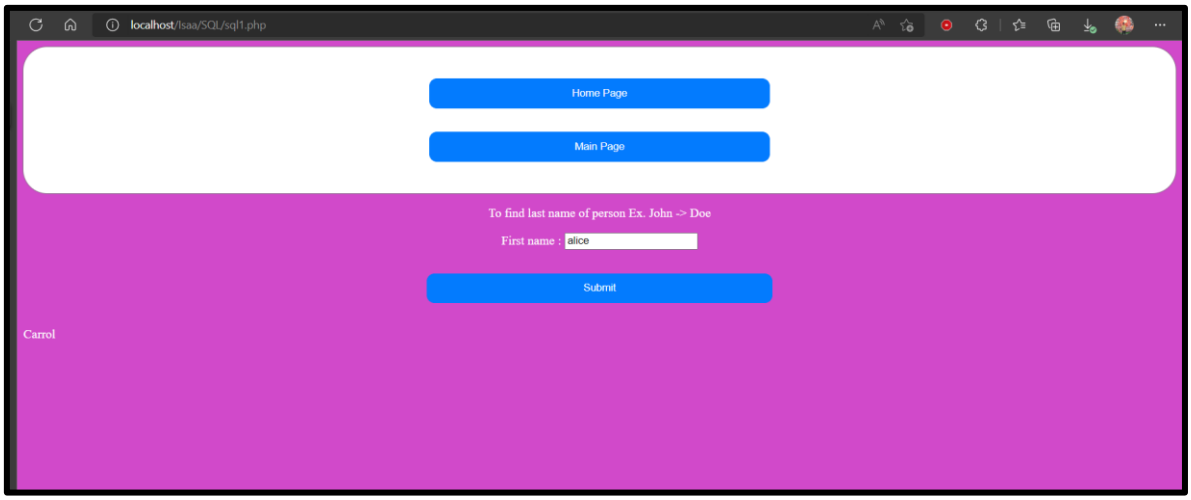
Various Tables created in SQL server:



Various methods available to perform SQL attacks with various increasing level of security:

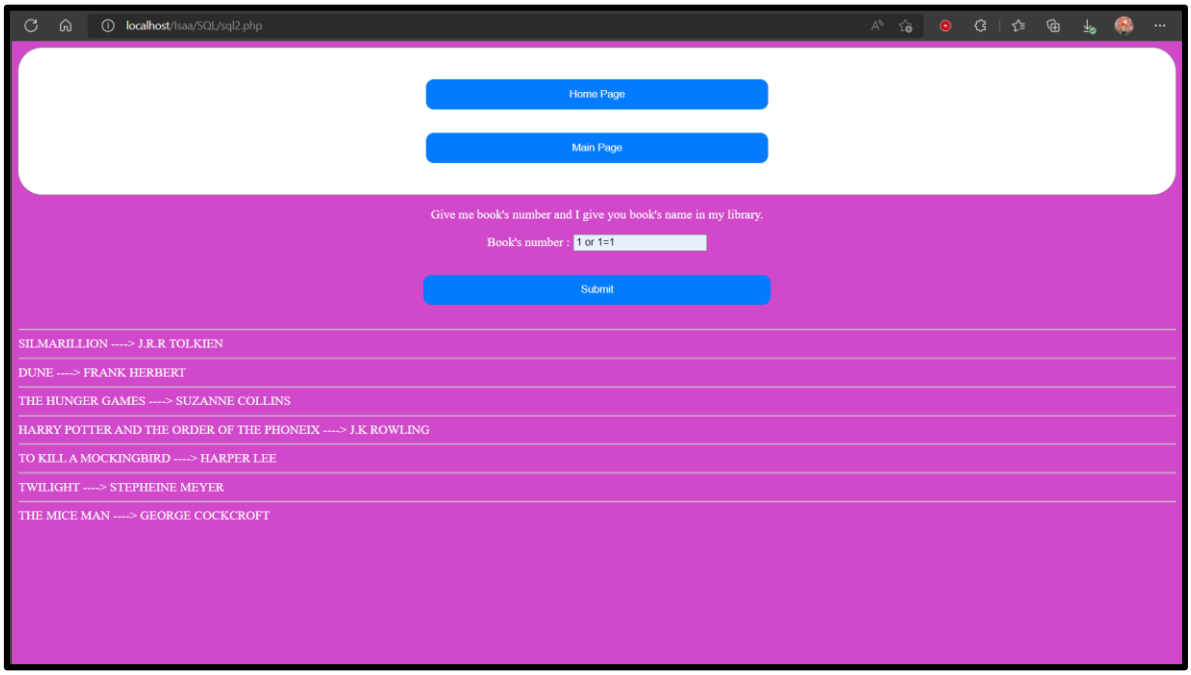


Attack 1: injection using string: ' or 1=1-- && John' or '1'='1

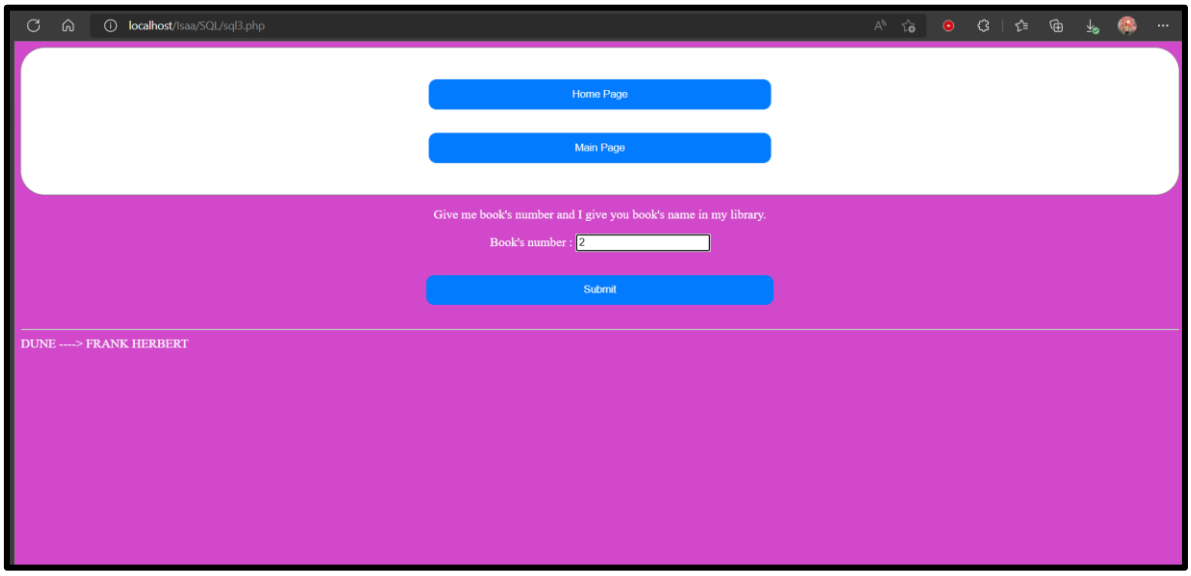


Performing level1 injection using string: ‘ or 1 = 1 –'

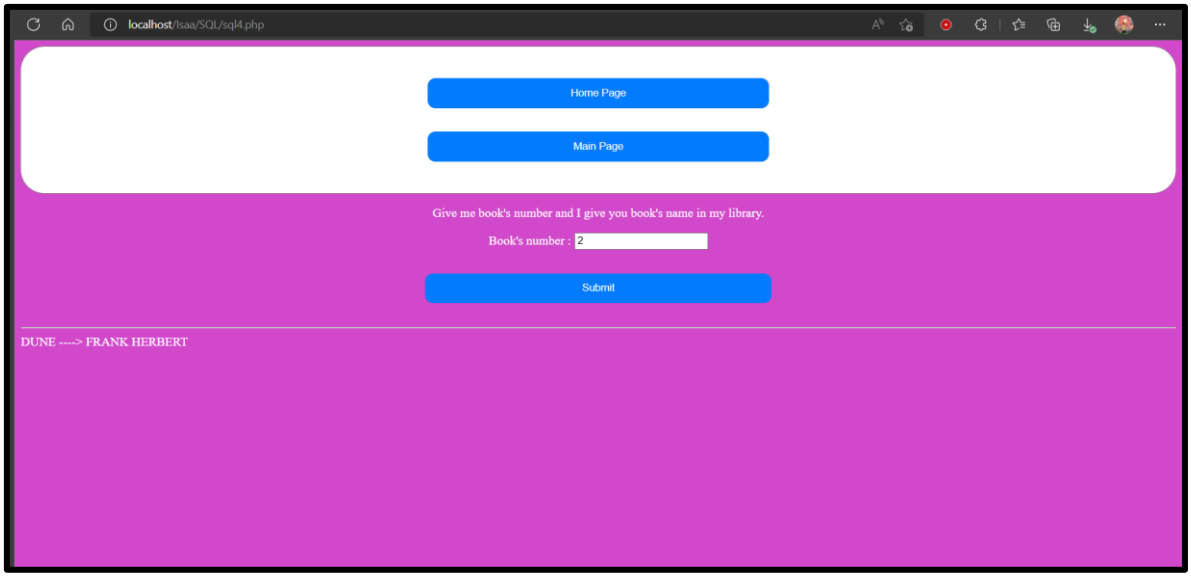
Attack 2 injection using string: 1 or 1 = 1:



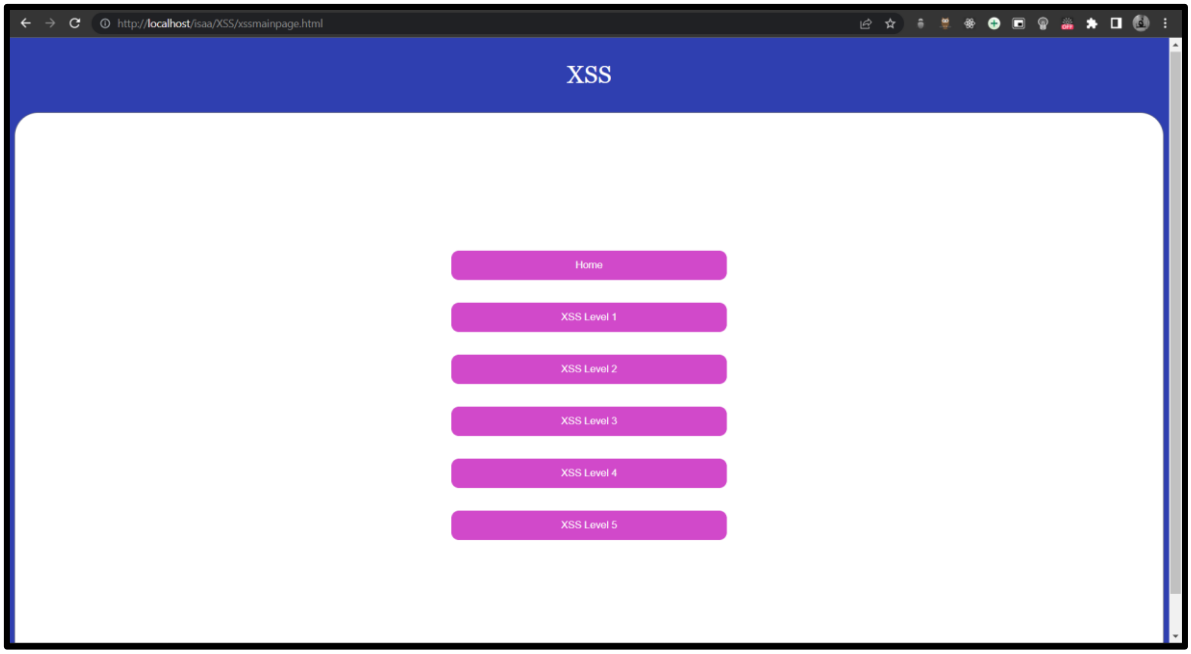
Level 3 injection:



Performing level 4 injection:



Various levels of XSS attacks:



6. Conclusion

Issues with SQL Injection are difficult to eradicate. The particular preventative measures are impacted by the SQL database engine, the programming language, and the subtype of the SQLi vulnerability. You should follow a few basic strategic principles in order to keep your web application safe.

Step 1: Maintain and practise attentiveness

Everyone engaged in developing your online application has to be aware of the dangers of SQL Injections in order to make it secure. You should provide all of your developers, QA employees, DevOps, and SysAdmins the appropriate security training.

Step 2: Never rely on user input

Consider all user input to be unreliable. An SQL Injection risk exists whenever user input is utilised in a SQL query. Authenticated and/or internal user input should be treated similarly to public input.

Step 3: Use whitelists rather than blacklists.

Blacklists shouldn't be used to censor user input. A crafty attacker can almost always find a way past your blacklist. If at all possible, only validate and filter user input using strict whitelists.

Step 4: Use the most recent technology

There is no SQLi protection in earlier web development technologies. Utilize the most recent versions of the development environment, language, and technology related to those elements. Use PDO rather of MySQLi, for instance, in PHP.

Step 5: Use dependable mechanisms

Avoid attempting to construct SQLi protection from scratch. The majority of contemporary development tools can provide you with SQLi protection measures. Use such systems rather than attempting to innovate. Use stored procedures or parameterized queries as examples.

Step 6: Keep checking sysem frequently

Developers may add SQL Injections or other libraries, modules, or applications may introduce them. Use a web vulnerability scanner like Acunetix to routinely check your online apps for vulnerabilities. Install the Acunetix plugin if you use Jenkins for automated build scanning.

Bibliography

[1] Li Qian, Zhenyuan Zhu, Jun Hu and Shuying Liu, "Research of SQL injection attack and prevention technology," 2015 International Conference on Estimation, Detection and Information Fusion (ICEDIF), 2015, pp. 303-306, doi: 10.1109/ICEDIF.2015.7280212.

[2] Z. Xiao, Z. Zhou, W. Yang and C. Deng, "An approach for SQL injection detection based on behavior and response analysis," 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017, pp. 1437-1442, doi: 10.1109/ICCSN.2017.8230346.

[3] S. O. Uwagbole, W. J. Buchanan and L. Fan, "Applied Machine Learning predictive analytics to SQL Injection Attack detection and prevention," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 1087-1090, doi: 10.23919/INM.2017.7987433.

[4] Q. Li, W. Li, J. Wang and M. Cheng, "A SQL Injection Detection Method Based on Adaptive Deep Forest," in IEEE Access, vol. 7, pp. 145385-145394, 2019, doi: 10.1109/ACCESS.2019.2944951.

[5] A. Ramesh, A. Bhowmick and A. V. Lal, "An authentication mechanism to prevent SQL injection by syntactic analysis," 2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15), 2015, pp. 1-6, doi: 10.1109/ITACT.2015.7492650.

[6] Salem, Asad. (2020). Mechanism to detect and prevent SQL injection attack from programmer side.