Detection and Prevention Technique of SQL Injection Attack to Protect Dynamic Web Application and Services

MSc Research Project
Cyber Security

Shariar Parves
Student ID: x20173024

School of Computing
National College of Ireland

Supervisor: Mr. Niall Heffernan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shariar Parves
Student ID:	x20173024
Programme:	MSc in Cyber Security **Year:** 2020-2021
Module:	Research Project
Supervisor:	Mr. Niall Heffernan
Submission Due Date:	11.11.2021
Project Title:	Detection and Prevention Technique of SQL Injection Attack to Protect Dynamic Web Application and Services
Word Count:	5747 **Page Count** 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Shariar Parves

Date: 10.11.2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Detection and Prevention Technique of SQL Injection Attack to Protect Dynamic Web Application and Services

Shariar Parves
x20173024

Abstract

Structured Query Language Injection (SQLI) which is the common injection techniques that uses the malicious code. It helps to the hackers or attackers for bypassing the security of the application with malicious code replacement through the SQL statement. Successful SQL injection in the web application, helps to the attacker to modify, edit, delete as well as retrieve content in the SQL database. According to the lists of OWSAP (The Open Web Application Security Project), SQLI is the number one threats in terms of the top 10 security risk of the web application. In this proposal it will be explained new approach for tracing the attempt of SQL injection for defending the dynamic web application. This will also help for tracing most of the details of the attackers which are like device details, browser details, operating system, IP address of the users, users' location. Information of the attackers will store to the database within the current time. Moreover, it will provide the instant notification in system admin. If attackers use botnet to attack to the system or application, it will also trace. Even it will also store the configuration of the device with high rate of the accuracy. Moreover, it will also provide the email notification. PHP regex, which is the powerful algorithm will use to implement the proposal. It will help for detecting SQL injection that is related to the statement in application. IP can be blocked through the cPanel if Suspicious activities that occurs through manual SQLI, botnet attack.

1 Introduction

Due to the infrastructure of technology and internet, it has been high demand to store sensitive data in the web application as well as database which will be available, accessible to the authorised users while they need. Dependency of the digital information increased. Moreover, web-based application used for creating the interface and it helps for the user to insert, edit information which is secure. Uses of dynamic web application has been used and developed more and more. Now it is the good target for the hackers, aim to get access to the sensitive information of the users. SQLI attack is very dangerous attack for the web based dynamic application, it helps to breach the confidentiality as well as data integrity for the system users. It becomes the serious threat in terms of security for any organisation which used dynamic web-based application (Acunetix 2021).

SQLI attack which is successful that will help to the attacker to read, modify, steal sensitive data from the system. Even they can also execute the administrative operation in the database, entire content can be recovered from the database as well as command for the operating system can be executed. SQLI which is remain the top ten cyber security concern in terms of the web application after the twenty years of the discovery. Moreover, according to the OWSAP top 10 of the web-based application risk, SQLI is the number one risky threat.

Corresponding to the CBR report there are around 65% of the cyber-attack happens based on the SQLI to the web-based application within 2017 to 2019. In recent Flaticon and Freepik which is owned in Freepik, that is the stock image dynamic websites. There are almost 8.3 million users has been affected by the SQL injection attack. Hackers successfully could steal the email as well as password hashes from the database. Popular WooCommerce plugin in the WordPress being used of the ecommerce business and it has been targeted and attempted through the wave of the SQL injection to exploit the system. British royal navy, Kaspersky's of Malaysian site, Microsoft SQL server, MySQL official page, record of Harvard university, Stanford university has affected with the SQL injection attack. Early year of 2021, social network service GAB and it has almost 4 million users affected and 70GB of data has been exfiltrated from the web application by using the most dangerous SQL injection attack (Cyware Social 2021).

Various researchers suggested different solution already for addressing the solution of the SQL injection attack. After analysing all the research there is no definite solution that can be able to ensure entire safety of the SQL injection attack. For detecting and preventing the SQL injection most of the existing techniques are like parameterised query with input validation, remove parameter, secure programming, sanitize the query statement. PHP regex algorithm was not discussed in any of the research papers that helps for detecting SQLI for protecting the web-based application.

Within the research, we focus how to create SQLI detection and prevention techniques with high performance, high accuracy through the PHP regex algorithm in the dynamic web application. Moreover, I will figure out below question in this research:

1. How to implement the process of detection for the SQLI in web-based application?
2. How to use the PHP regex for detecting the attempt of SQL injection in the system?
3. How to execute the different SQLI in dynamic web application?
4.What are the possible steps that can help for protecting the SQL injection in the web application?
5. What are the ways attacker's information can be traced, stored within the database?
6. How the hackers' details can be triggered as notification to the system admin by using the email?
7. Is that possible to detect the SQLI if it happens by using the botnet and how?
8. How to use IP blocker from cPanel to block suspicious IP to restrict unauthorised access in the web application?

New approach of SQLI regarding for detection as well as prevention techniques has illustrated for keeping the web application secure through PHP regex algorithm in this paper.
It will provide the email notification as well as attackers information will be stored to the database in case of SQL injection attempt to the system. Attackers IP address can block to restrict user access in the application. The paper has been planned where section 2 will discuss about the relevant work that done previously with comparison by specifying foreseen contribution. In the section 3, methodology of the research will be illustrated including the methodology evaluation. Section 4, specification of the design will be presented with implementation of architecture. Section 5, implementation of proposed solution will illustrate. Section 6, comprehensive analysis will be done where findings and result will be deemed. Moreover, Future work will be anticipated including reference in section 7.

2 Related Work

In terms of the various parameter like storage, computability of the browser, space, security the web application improving progressively. Still there are various chances to get attack and system can be damaged of the secure application. There are different types of the methodologies presented that helps for the detection, prevention as well as recover about the SQL injection attack (O. Ojagbule *et al.*,2018). Researchers have been studying various types of the detection and prevention method regarding to the SQL injection attack, on the other hand attackers also keep trying to get vulnerabilities in the targeted application by using the SQL injection statement. Various researchers' works been done with the purpose of SQLI detection and prevention techniques. Comparison and contrast regarding to the preceding work will be discussed in this section, specify the objective of the past effort and foreseeable contribution related to the SQLI prevention and detection techniques.

The author (W. H. Rankothge et al.,2020) proposed the techniques of identifying and mitigating SQLI by the help of Web VIM tool. Moreover, it helps for scanning vulnerabilities of all the pages of web application, it also analyses the source code. If any pages are vulnerable for the SQLI attack, chosen tools provide security automatically. This tool helps to take URL which is needed for assessing SQLI attack vulnerability according to the user input. At the beginning it will try to get the subpages in the web application, there will be creation of sitemap, subpages link will be extracted. Chosen tool will inject payload of SQLI to the application input field, if positive result is found then it will provide the confirmation about the application has the SQLI vulnerabilities. In terms of the mitigation process author (W. H. Rankothge *et al.*,2020) proposed about the stored procedure, escaping as well as parameterized queries. WebVIM uses only the user's input validation. Moreover, sanitized process also used in the system for protecting the SQLI. The chosen tools can perform well in the Linux platform.

For detecting SQLI, dynamic analysis, static analysis as well as combined analysis used by the author (M. Al Rubaiei *et al.*,2020). For findings the weakness, shortcomings to the system it is standard to use the static analysis. This is the standard that helps for identifying the SQLI attack. In case of the dynamic analysis, it can be produced in the runtime. It is the strategy that can analyse with runtime SQL question created in the client input information in web application. The tools of AMNSIA used for detecting SQLI attack within the system which will monitor the system. It only can work on the application which is JAVA based. For protecting web application author (M. Al Rubaiei *et al.*,2020) used diverse technique which are like input validation, data sanitization, prepared statement with session tokenization (W. H. Rankothge *et al.*,2020) (M. Al Rubaiei *et al.*,2020).

PHP has been used for developing the WebVIM tool, it can be able to perform better on the Linux platform. In accordance to the Nikto, Nessus, OpenVAS mitigation process with the scanning tools not compatible. It gives the detection and mitigation process within the system by using the single platform (W. H. Rankothge *et al.*,2020). In contrast, the author (M. Al

Rubaiei *et al.*,2020) has been used various types of secure programming techniques, input validation, sanitization of data (M. Al Rubaiei *et al.*,2020). It was not compatible within the security gateway. There are different types of overviews to the SQL injection attack as well as process of defense (Tasevski and Jakimoski, 2020). It can be prevented through input based analyse that will described in the following section.

There are different type's attacks uses for executing the SQL injection attack. Attackers could be able to bypass access, authenticate, modify and delete database (Tasevski and Jakimoski, 2020). Types of SQL injection attack are Union based SQLI, error based SQLI, out of band SQLI, blind SQLI. Patterns of the attacks are different in the web application. Common types of attack are the error-based attack. It basically based on the invalid input which is unexpected with the user interface. In that case database server reply error result that contains target information including the OS (Operating System), structure and the version of the system (Tasevski and Jakimoski, 2020). In terms of the union based SQLI attack it provides the result with original query, extended through the union operator. Multiple queries can be run if it is same structure. In case of the blind SQLI attack, it doesn't give the error message which is complicated for exploiting. Out of band SQLI attack, which is not the common types SQLI attack, it has depended to the enable feature for the database server. It can be occurred if the attacker does not use the same channel to execute attack (Tasevski and Jakimoski, 2020).

By analysing various types of attack author has proposed different mitigation techniques about the SQLI attack. According to context, filtering can help for sanitizing the user data. During time of input phone number, email can be filtered which will help for allowing character in email filed as input, number should allow on the phone number filed. Any others input should not allow to the application except the input which is defined. Website protection of Cloudflare is the protection to unauthorised attempt for getting the access in the target website, information of client (Tasevski and Jakimoski, 2020). It has to be enabled data encryption through the HTTPS while data send through internet which is the defense of SQLI. Patch and update is also important for implementing security in the system. password hashing through the encryption, which is vital, monitoring tools that helps to utilize machine learning for protecting attack.

Input verifier model, input categorization can be able to help for detection as well as prevention SQLI (A. Jana et al.,2020). malicious user input which will generate SQL query as well as execute lead the SQLI attack. With the proposed plan author (A. Jana et al.,2020) defined about the user input in the keywords which is four categories, numbers of alphabet as well as special characters. Possible string input can be generated in four categories. In accordance with the user input some of the input can be safe, some may be unsafe for the user input. Input basically verifies the identity of the actual input that is malicious as well as not safe for the application. It also helps for protecting the application with database from the SQLI (A. Jana et al.,2020). Various types of attack explained in various protection in the case of protection filter used for checking input, verification of input (Tasevski and Jakimoski, 2020) that implemented for checking the vulnerabilities in terms of input. To identify input verification author uses the code in the programme (A. Jana et al.,2020). It has been needed for checking the detection, prevention methods of the SQLI before the production of the web application (Tasevski and Jakimoski, 2020). Green SQL input model that helps for optimising for constructing the patterns of input and the whitelist of the input (P. Lin et al.,2020), it will discuss as well as method of counter measure SQLI attack (Alsobhi and Alshareef, 2020).

Database which is played the important role for strong the information, it is also important for protecting the security. (P. Lin et al.,2020). It can be surveyed through the company of data security, there are almost six thousand decision makers of the technology in the various size of companies related to the SQL injection attack. Almost half of the respondent respond about the SQL injection is the dangerous attack for the web application. Green SQL method is the database firewall which is opensource that has been used by the author (P. Lin et al.,2020). To do the operation there are almost four modes that are database IDS mode, IPS mode, firewall mode as well as learning mode. Performance of the protection regarding to the Green SQL to the database security that depends on the evaluation, SQL command recognition, generation of whitelist. For optimising the green SQL attack to the learning mode, it has been required for building input of attack with category. In contrast (Alsobhi and Alshareef, 2020) author has explained various kinds of SQLI which impacts to the web application. In case of counter measure about the SQLI attack, author anticipated about the escape function, error message custom, keywords prohibited as well as prepared queries.

The author (P. Lin et al.,2020) has been proposed about the SQLI detection though the predefined input in accordance with the categories. In the time of the web application development, developer must maintain the secure programming, coding practice (Alsobhi and Alshareef, 2020). Input should be given based on the existing attack characteristics with limitation (P. Lin et al.,2020). It gives the information about the SQLI, provide the highlight of the database protection importance (Alsobhi and Alshareef, 2020). Secure shell that has the checking of validation process which will help to verify about the request signature, it also proposed through the author (D. Patel et al.,2020) about the SQLI that will be explained in below. Query statement sanitizing has been proposed, experimented by the author (Hlaing and Khaing, 2020).

Due to the preventive query of the SQL statement, it will help to protect hackers for executing the malicious SQL query in the web application for getting access inside the server, database. It will not be possible to go hackers request within the secure shell, it also helps to reject the request (D. Patel et al.,2020). It will make impossible to the hackers for getting access in database as it will react as the secure shell. It will verify the process of checking in the signature model of checking. If it has not been matched though the store procedure, then it will deny request, attackers not be able for reaching the database (D. Patel et al.,2020). The author (Hlaing and Khaing, 2020) proposed query sanitizing is the steps that will consist of another two steps. First will be query of tokenising the user input, this is the process that helps to detect white space, sign sharp, string before the symbol. In second steps after the tokenising query detected of string token by predefined the lexicon (Hlaing and Khaing, 2020).

The author (Hlaing and Khaing, 2020) has been proposed approach for preventing the SQLI by using the terms of tautologies, union queries, malformed queries, piggy bank queries, store procedure. It has not been used the input validation for protecting the SQLI, store procedure has been used by the author which will match the signature. It will also help for preventing the unauthorised access in the system (D. Patel et al.,2020). Author (S. Saleem et al.,2020) has proposed machine learning for detecting and protecting the SQL injection through the partial analyse of HTTP request, it will also prevent SQLI (K. Kuroki et al.,2020).

To detect the attacks in the normal activities machine learning has been used. It can consider three kids of attacks in the two methods. Simple extraction method can be taken in the form that has limited SQLI (S. Saleem et al.,2020). Text based classification also applied for the logs of web server (S. Saleem et al.,2020). SQL query in the HTTP request also helps for estimating the intension within syntax analysis. In the proposed (K. Kuroki et al.,2020) method for identifying SQLI accurately almost 83.1% and 71.9% to the artificial dataset.

3 Research Methodology

In this report, detection and prevention techniques of SQLI attack within the web based dynamic application by using PHP regex algorithm has proposed. Proposed algorithm is powerful algorithm which will help for filtering user input, it will also check matching pattern within the string. If it is found that input string is vulnerable of SQLI, it helps to detect as well as store the hackers’ details with device configuration, IP address of attackers, location. All the details will be stored to the database, it will also trigger email notification. If any malicious which is insider try to do the manual SQLI which can trace through suggested method. Attack by using the botnet also can be tracked as well as provide the email notification to the system admin.

3.1 Method of Detection:

In accordance with the discussion above regarding to the literature review, various method of detection has been proposed. PHP regex is fast, accurate, simple as well as powerful algorithm for detecting the SQLI attack. PHP regex will be used to implement in this research. For the successful SQLI hackers uses the common payload which is injectable by using the query statement. It can be used as 'OR 1 --, ' or '', ' OR'' =', ''OR'' '' = '','OR'1, etc. These patterns will be used for defining the PHP regex. It will send email notification instantly if any one input any string to the given form that matched with predefined PHP regex algorithm.

3.2 Method of Mitigation:

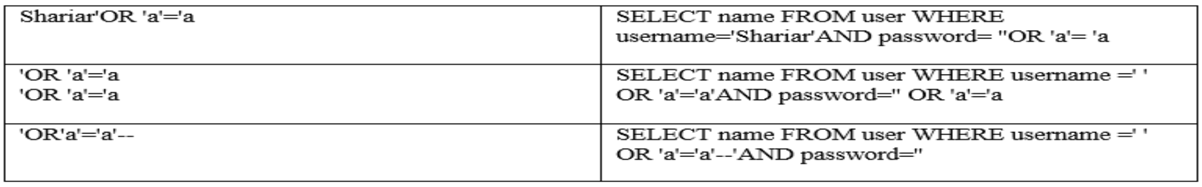
In terms of the SQLI prevention techniques, it has been illustrating various types of prevention techniques in the section of literature review. For example, parameterised query, input validation, storing and escaping procedure etc. In the suggested method, PHP regex algorithm will be used besides the validation of user input as well as parametrised query. By using that proposed method it will help for detecting the IP address of the target users.

Suspicious IP can be able to block by using the Cpanel for restricting the user access to the application. Various types of the SQLI are tautology, piggyback queries, malformed queries, union queries as well as stored procedure that can mitigate by using the PHP regex algorithm accurately. Proper analysis as well as documentation will explain in following section including project plan. Plan can be evaluated in accordance with the research question.

3.3 Analysis:

For storing the data into the database web application has been used and user can be able to get access anytime for retrieving information in accordance with the authorisation. If it is not valid the user input, not performed properly it will not allow access within the system. if it finds the weakness of the application various types of SQL injection can be implemented.

Tautology: It is the attack that basically bypass authentication of the users as well as extract the data after inserting the tautology by using where clause in SQL query. Query convert which is the main condition of this method, it affects row of database in the open table for the unapproved user.

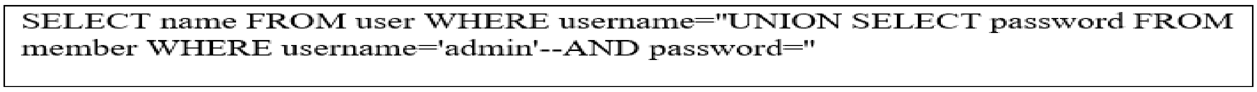


Shariar'OR 'a'='a	SELECT name FROM user WHERE username='Shariar'AND password= "OR 'a'= 'a
'OR 'a'='a 'OR 'a'='a	SELECT name FROM user WHERE username =' ' OR 'a'='a'AND password='' OR 'a'='a
'OR'a'='a'--	SELECT name FROM user WHERE username =' ' OR 'a'='a'--'AND password=''

In the above statement uses the query that is true, and it can be added through tautology '--' and ('a'='a') to parser in SQLI. Rest of the statement will be as comment, and it will not execute on the system.

Malformed Queries: It happens while the attacker try to implement or insert the illegal SQL token in the application. Application will reject inserts and provide error message which will provide the debugging message containing database information. To recognise parameter that injectables can utilise by the syntax error.

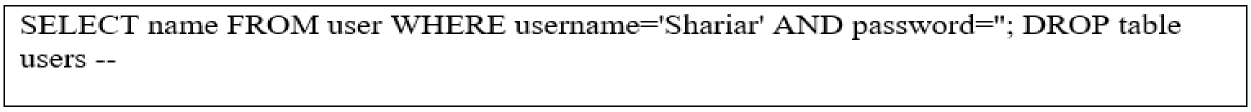
Union Queries: This is the method where the unacceptable statement joined within the valid query with the keyword of UNION. Hackers uses query statement for the structure of UNION for making that it is legitimate. It will help for return information in the database table.



```
SELECT name FROM user WHERE username=''UNION SELECT password FROM
member WHERE username='admin'--AND password=''
```

'--' will comment if the query successfully executed in the system.

Piggy-backed Queries: Within the piggyback query hackers tried extra queries to be used in query string. In the first query it will use ';' to add another query with that.



```
SELECT name FROM user WHERE username='Shariar' AND password=''; DROP table
users --
```

In above, original query is the first part which help for executing the query in database, remaining second part will act as the malicious query to database.

Stored Procedure: This code will store within the database, directly executable by using database engine. In terms of activation the SQLI for the stored procedure as the following:

```
SELECT name FROM member WHERE username=''; SHUTDOWN; --password=''
```

3.4 Uses Software Tools:

For doing the experiment as well as implementation about the proposed research project, working web dynamic web application needed with MySQL database connection. It is also needed for hosting the web application in the Linux based operating system. following tools will be needed for implement the project which are:

Tools	Version
cPanel	94.0
PHP	7.2.34
Bootstrap	4.0
Apache	2.4.48
MySQL	10.3.30
SMTP (Simple Mail Transfer Protocol)	Used for sending mail to system admin.

3.5 Evaluation of Approach:

According to the evaluation of approach, SQL injection detection as well as prevention techniques for the web application in terms of the research question has been provided in below:

Time	Detect Instantly.
Accuracy	100% accuracy.
Send notification	Instantly
Device configuration	Accurate configuration
IP Block	IP can block manually

Time of Response:
Time of the response is faster, accurate. If any hackers try to insert malicious code within the application, it will provide instant response for the system admin. It will store the attacker's details into database, even it will provide the email notification to the system admin instantly.

Performance:
PHP regex is the powerful algorithm which match input patterns and it will use. It provides the accurate result, find the matching patterns which is vulnerable for the system input. It can perform precisely for detecting various types of SQL injection to the system.

Time of Computing:
It is the time that required for computing the result to the normal query with the respect of the injected query. It will provide usual result for the system.

Secure Coding:
By maintaining the secure programming application should be built where URL validation, input validation will implement. Sanitization of data, tokenisation of session, prepared statement will use during query execution. It will give extra layer security for protecting application.

Information of attacker:
It will capture the attacker information (browser, OS, IP, location) instantly if anyone try to inject malicious SQL statement within the application. In that case it stores hackers' or attackers' details in the database, email notification will be triggered to the system admin instantly. In case of bot driven SQLI, it will provide mail notification that will contain same IP many times in the database as well. Through the suggested method it can be caught the malicious insider easily while he tries SQLIA.

Block IP:
By logging in the web application system admin can get the attacker IP details. It helps for analysing manually about the IP. If the IP is suspicious, admin can be able to block the target IP from the cPanel for restricting the user to access to the web application. There are different types of method about the SQLIA detection and prevention technique explained in above literature review. Evaluation of the existing method as well as proposed method given below:

Detection and Prevention Method	Adjust source code	Detect attack types
Proposed Method	No need, it can predefine as well.	All

4 Design Specification

4.1 SQL Injection attack flow:
In accordance with the figure 1, attacker try to inject malicious SQL injection by using the web application, it basically sends the HHTP request to the business logic. It will match the requested query as well as it will forward to database for retrieving the information according to the client request.

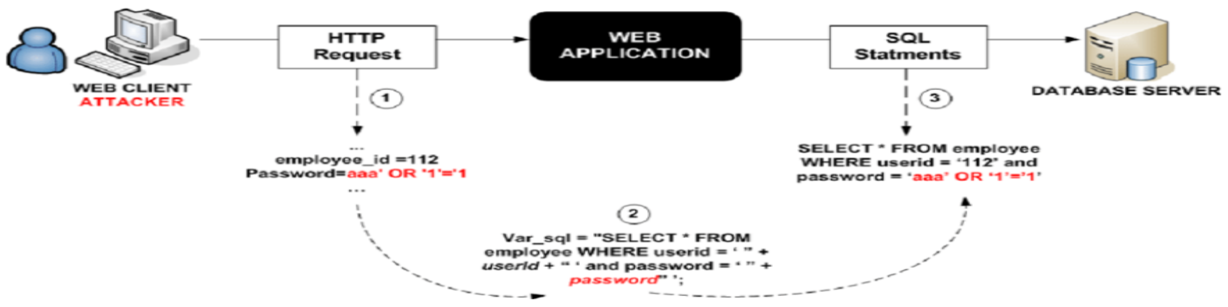

WEB CLIENT
ATTACKER
HTTP Request
WEB APPLICATION
SQL Statments
DATABASE SERVER
1
...
employee_id =112
Password=aaa' OR '1'='1
...
2
Var_sql = "SELECT * FROM employee WHERE userid = ' " + userid + " ' and password = ' " + password" ';
3
SELECT * FROM employee WHERE userid = '112' and password = 'aaa' OR '1'='1'

Figure 1: SQL injection attack flow.

4.2 Proposed approach flow:

According to the figure 2, if the hackers intend to execute the malicious code in the system it will match up the query within the predefined PHP regex algorithm. If any matched found within the patterns which is vulnerable then it stores the hackers' details in database, before the execution of the malicious code it will provide the email notification to the admin. System admin can be able to login to the application for viewing the hacker's information, view IP address. From the cPanel admin can block the IP address by using the IP blocker for restricting the user accessing to the application. In terms of the bot driven SQLI it also stores the information in the database, send the mail notification to the admin.

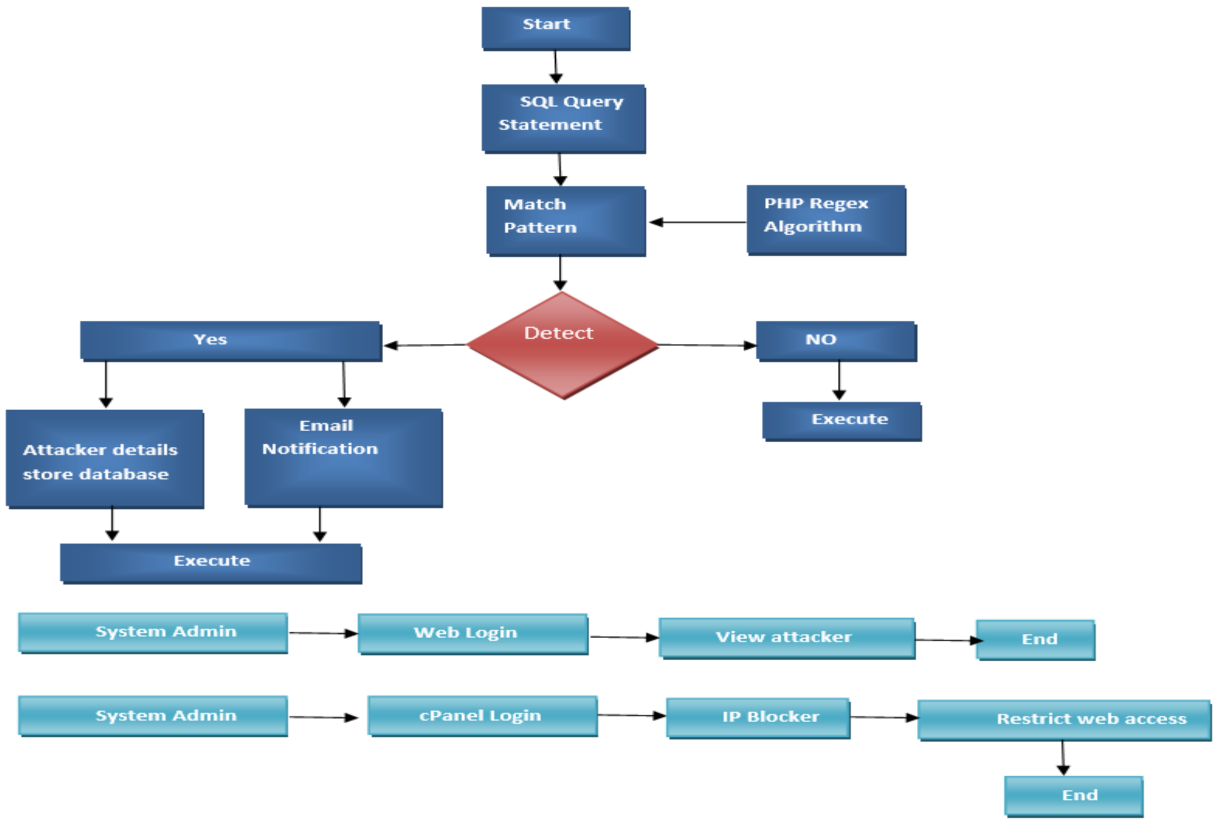


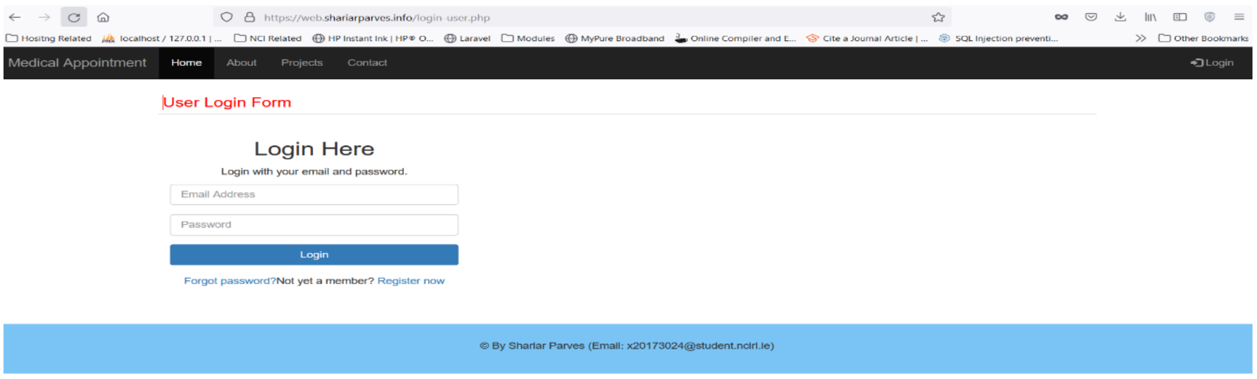
Figure 2: Proposed flow diagram of approach.

5 Implementation

Working web application with database connection is required to implement the project. Sign in form needed for the front end of the application where input will be given.

5.1 Login Access:

By using the bootstrap 4, HTML and PHP login form has been implemented to maintain validation of input. Database connection has been made with the form. It helps for checking user input to give the access successfully to the dynamic web application. If any user tries to input malicious statement of SQLI through the form it will give email notification as well as it will store the user device configuration including the location in database.


https://web.shariarparves.info/login-user.php

Medical Appointment | Home | About | Projects | Contact | Login

User Login Form

Login Here

Login with your email and password.

Email Address

Password

Login

Forgot password?Not yet a member? Register now

© By Shariar Parves (Email: x20173024@student.ncirl.ie)

Figure 3: Form of Login.

5.2 Parameterized Query with validation of input:

To avoid the SQLI, parameterised query has been implemented including the input validation. Users must use the valid email address and proper password by maintaining the password policy to enter the system.

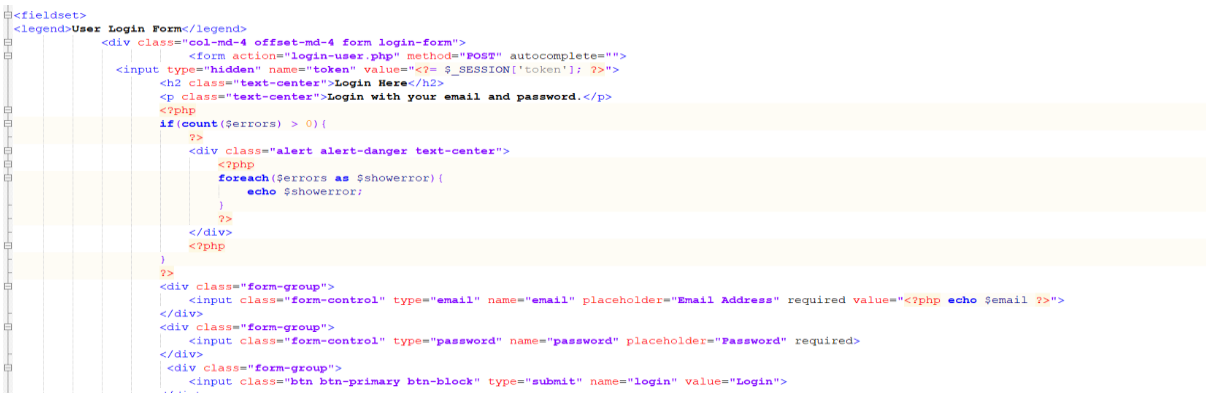

```
<fieldset>
<legend>User Login Form</legend>
            <div class="col-md-4 offset-md-4 form login-form">
                  <form action="login-user.php" method="POST" autocomplete="">
              <input type="hidden" name="token" value="<?= $_SESSION['token']; ?>">
                    <h2 class="text-center">Login Here</h2>
                    <p class="text-center">Login with your email and password.</p>
                    <?php
                    if(count($errors) > 0){
                        ?>
                        <div class="alert alert-danger text-center">
                            <?php
                            foreach($errors as $showerror){
                                echo $showerror;
                            }
                            ?>
                        </div>
                        <?php
                    }
                    ?>
                    <div class="form-group">
                        <input class="form-control" type="email" name="email" placeholder="Email Address" required value="<?php echo $email ?>">
                    </div>
                    <div class="form-group">
                        <input class="form-control" type="password" name="password" placeholder="Password" required>
                    </div>
                     <div class="form-group">
                        <input class="btn btn-primary btn-block" type="submit" name="login" value="Login">
```

Figure 4: Input Validation.

Parametrised query which is effecive techniques to get protection from the SQL injection attack.

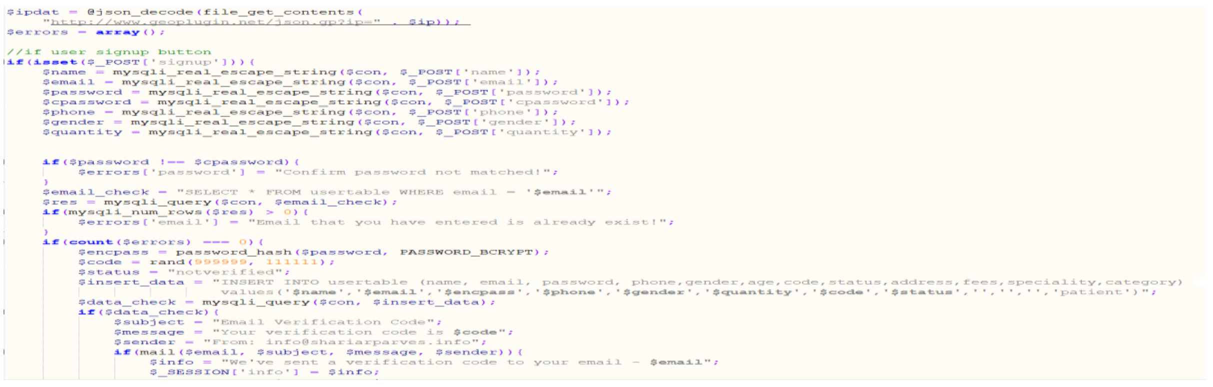
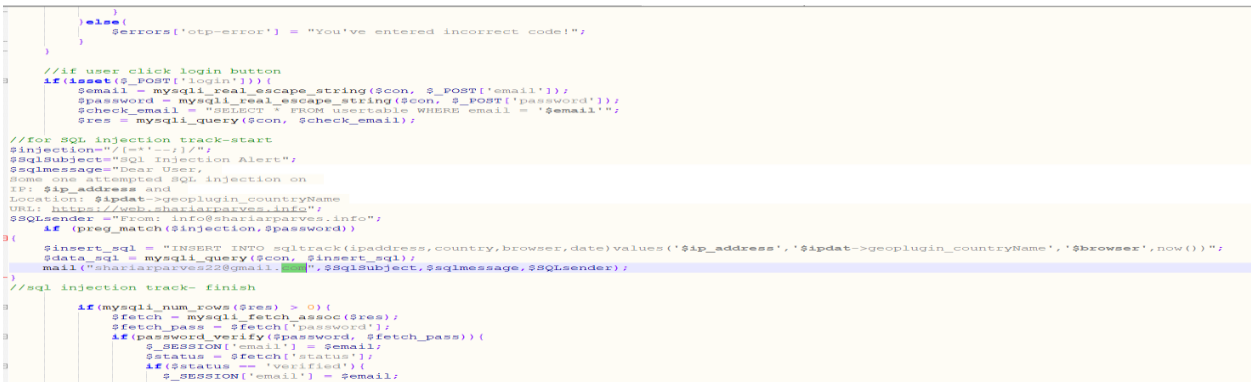

```
$ipdat = @json_decode(file_get_contents(
    "http://www.geoplugin.net/json.gp?ip=" . $ip));
$errors = array();

//if user signup button
if(isset($_POST['signup'])){
    $name = mysqli_real_escape_string($con, $_POST['name']);
    $email = mysqli_real_escape_string($con, $_POST['email']);
    $password = mysqli_real_escape_string($con, $_POST['password']);
    $cpassword = mysqli_real_escape_string($con, $_POST['cpassword']);
    $phone = mysqli_real_escape_string($con, $_POST['phone']);
    $gender = mysqli_real_escape_string($con, $_POST['gender']);
    $quantity = mysqli_real_escape_string($con, $_POST['quantity']);


    if($password !== $cpassword){
        $errors['password'] = "Confirm password not matched!";
    }
    $email_check = "SELECT * FROM usertable WHERE email = '$email'";
    $res = mysqli_query($con, $email_check);
    if(mysqli_num_rows($res) > 0){
        $errors['email'] = "Email that you have entered is already exist!";
    }
    if(count($errors) === 0){
        $encpass = password_hash($password, PASSWORD_BCRYPT);
        $code = rand(999999, 111111);
        $status = "notverified";
        $insert_data = "INSERT INTO usertable (name, email, password, phone,gender,age,code,status,address,fees,speciality,category)
                        values('$name','$email','$encpass','$phone','$gender','$quantity','$code','$status','','','','patient')";
        $data_check = mysqli_query($con, $insert_data);
        if($data_check){
            $subject = "Email Verification Code";
            $message = "Your verification code is $code";
            $sender = "From: info@shariarparves.info";
            if(mail($email, $subject, $message, $sender)){
                $info = "We've sent a verification code to your email - $email";
                $_SESSION['info'] = $info;
```

Figure 5: Implementation of Parameterised Query

5.3 Regex algorithm and PHP Code:

Version 7.2, PHP code has been used, it has also been included the SQL =*'--;, patterns. If it has been found any matched input, it will store the user details (Location, IP address, Browser, time of SQL injection, OS) in the database. It will also provide the email notification to the given email shariarparves22@gmail.com by using the info@shariarparves.info.

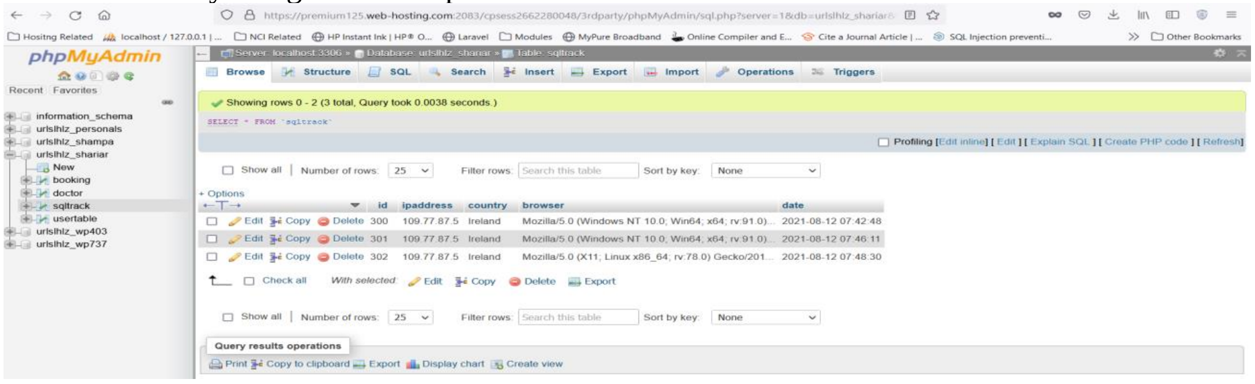


```
            }
        }else{
            $errors['otp-error'] = "You've entered incorrect code!";
        }
    }

    //if user click login button
    if(isset($_POST['login'])){
        $email = mysqli_real_escape_string($con, $_POST['email']);
        $password = mysqli_real_escape_string($con, $_POST['password']);
        $check_email = "SELECT * FROM usertable WHERE email = '$email'";
        $res = mysqli_query($con, $check_email);

//for SQL injection track-start
$injection="/[=*'--;]/";
$SqlSubject="SQL Injection Alert";
$sqlmessage="Dear User,
Some one attempted SQL injection on
IP: $ip_address and
Location: $ipdat->geoplugin_countryName
URL: https://web.shariarparves.info";
$SQLsender ="From: info@shariarparves.info";
    if (preg_match($injection,$password))
{
    $insert_sql = "INSERT INTO sqltrack(ipaddress,country,browser,date)values('$ip_address','$ipdat->geoplugin_countryName','$browser',now())";
    $data_sql = mysqli_query($con, $insert_sql);
    mail("shariarparves22@gmail.com",$SqlSubject,$sqlmessage,$SQLsender);
}
//sql injection track- finish

        if(mysqli_num_rows($res) > 0){
            $fetch = mysqli_fetch_assoc($res);
            $fetch_pass = $fetch['password'];
            if(password_verify($password, $fetch_pass)){
                $_SESSION['email'] = $email;
                $status = $fetch['status'];
                if($status == 'verified'){
                    $_SESSION['email'] = $email;
```

Figure 6: Implementation of PHP Regex algorithm.

5.4 Database Management:

MYSQL database used for storing the information of attacker in the system that connected with front end by using the admin panel.



✓ Showing rows 0 - 2 (3 total, Query took 0.0038 seconds.)

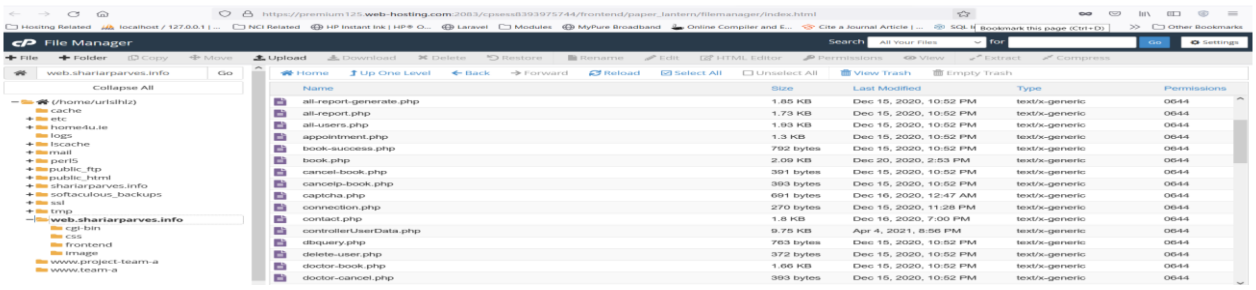
```
SELECT * FROM `sqltrack`
```

id	ipaddress	country	browser	date
300	109.77.87.5	Ireland	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0)...	2021-08-12 07:42:48
301	109.77.87.5	Ireland	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0)...	2021-08-12 07:46:11
302	109.77.87.5	Ireland	Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/201...	2021-08-12 07:48:30

Figure 7: Design of Database.

5.5 C Panel:

Cpanel will be used for storing files for getting access in the system.

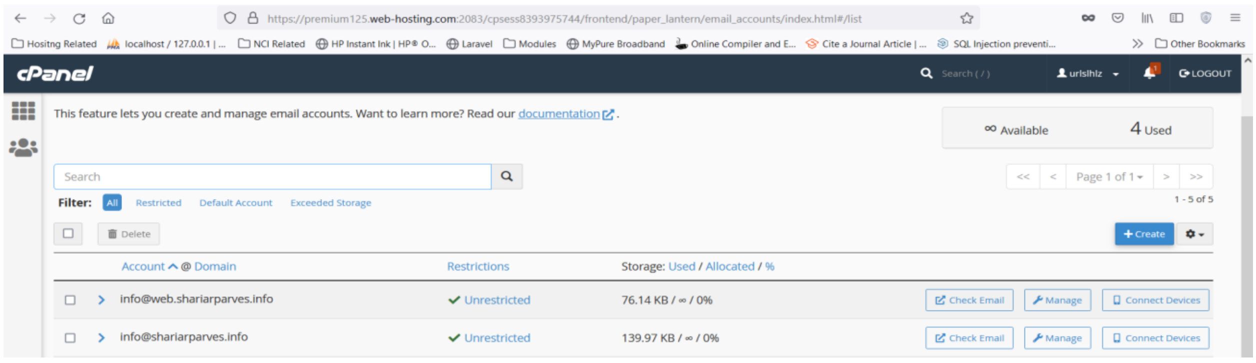


Name	Size	Last Modified	Type	Permissions
all-report-generate.php	1.85 KB	Dec 15, 2020, 10:52 PM	text/x-generic	0644
all-report.php	1.73 KB	Dec 15, 2020, 10:52 PM	text/x-generic	0644
all-users.php	1.93 KB	Dec 15, 2020, 10:52 PM	text/x-generic	0644
appointment.php	1.3 KB	Dec 15, 2020, 10:52 PM	text/x-generic	0644
book-success.php	792 bytes	Dec 15, 2020, 10:52 PM	text/x-generic	0644
book.php	2.09 KB	Dec 20, 2020, 2:53 PM	text/x-generic	0644
cancel-book.php	391 bytes	Dec 15, 2020, 10:52 PM	text/x-generic	0644
cancelp-book.php	393 bytes	Dec 15, 2020, 10:52 PM	text/x-generic	0644
captcha.php	691 bytes	Dec 16, 2020, 12:47 AM	text/x-generic	0644
connection.php	270 bytes	Dec 15, 2020, 11:28 PM	text/x-generic	0644
contact.php	1.8 KB	Dec 16, 2020, 7:00 PM	text/x-generic	0644
controllerUserData.php	9.75 KB	Apr 4, 2021, 8:56 PM	text/x-generic	0644
dbquery.php	763 bytes	Dec 15, 2020, 10:52 PM	text/x-generic	0644
delete-user.php	372 bytes	Dec 15, 2020, 10:52 PM	text/x-generic	0644
doctor-book.php	1.66 KB	Dec 15, 2020, 10:52 PM	text/x-generic	0644
doctor-cancel.php	393 bytes	Dec 15, 2020, 10:52 PM	text/x-generic	0644

Figure 8: Cpanel files.

5.6 Email Creation:

Email created for the cPanel which to send email notification.


https://premium125.web-hosting.com:2083/cpsess8393975744/frontend/paper_lantern/email_accounts/index.html#/list

This feature lets you create and manage email accounts. Want to learn more? Read our documentation.

∞ Available 4 Used

Filter: All Restricted Default Account Exceeded Storage

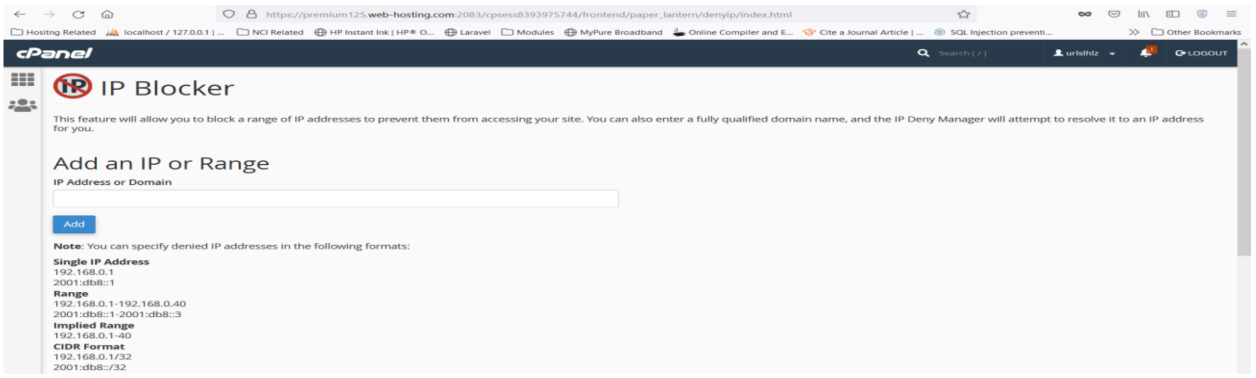
1 - 5 of 5

Account @ Domain	Restrictions	Storage: Used / Allocated / %
info@web.shariarparves.info	✓ Unrestricted	76.14 KB / ∞ / 0%
info@shariarparves.info	✓ Unrestricted	139.97 KB / ∞ / 0%

Figure 9: Email to send SQLI alert.

5.7 IP Blocker:

Suspicious IP that will block by adding IP in the IP blocker which has been shown in below:


https://premium125.web-hosting.com:2083/cpsess8393975744/frontend/paper_lantern/denyip/index.html

IP Blocker

This feature will allow you to block a range of IP addresses to prevent them from accessing your site. You can also enter a fully qualified domain name, and the IP Deny Manager will attempt to resolve it to an IP address for you.

Add an IP or Range

IP Address or Domain

Note: You can specify denied IP addresses in the following formats:

Single IP Address
192.168.0.1
2001:db8::1
Range
192.168.0.1-192.168.0.40
2001:db8::1-2001:db8::3
Implied Range
192.168.0.1-40
CIDR Format
192.168.0.1/32
2001:db8::/32

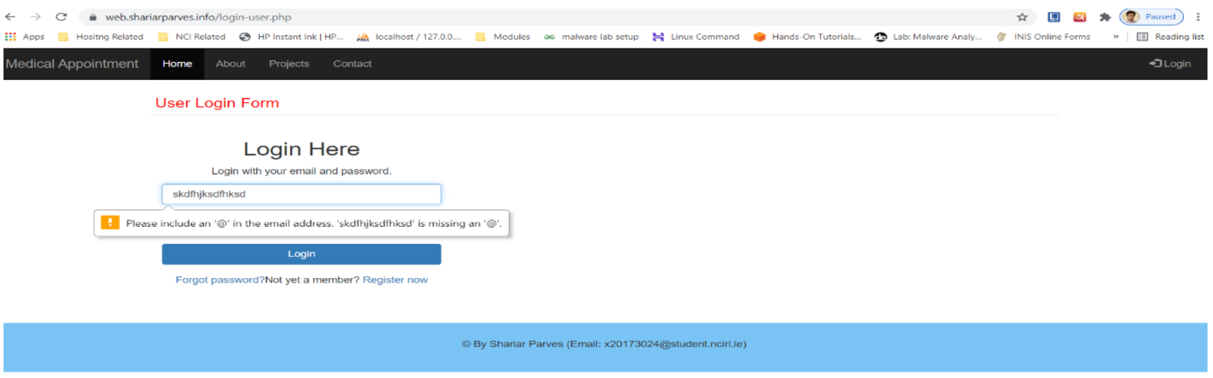
Figure 10: Implementation of IP Blocker.

6 Evaluation

Finding and thoroughly analysis will present in the following section.

6.1 Parametrised query and input validation:

Login form has been maintained strictly by using input validation, parameterised query used for preventing SQL injection.


web.shariarparves.info/login-user.php

Medical Appointment | Home | About | Projects | Contact | Login

User Login Form

Login Here

Login with your email and password.

skdfhjksdfhksd

Please include an '@' in the email address. 'skdfhjksdfhksd' is missing an '@'.

Login

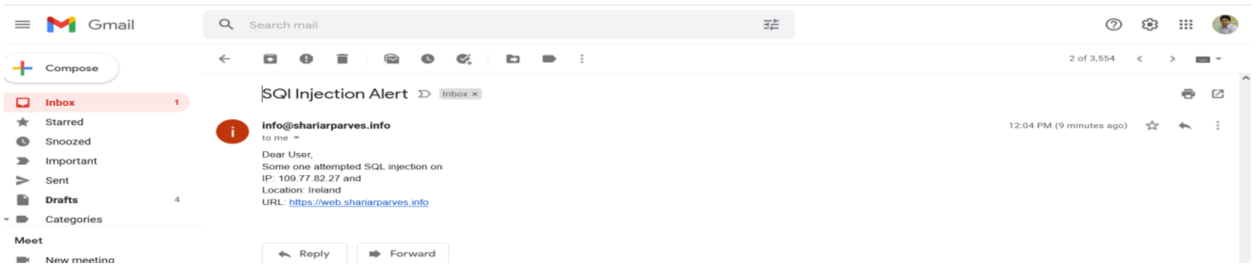
Forgot password?Not yet a member? Register now

© By Shariar Parves (Email: x20173024@student.ncirl.ie)

Figure 11: Parametrised query and input validation.

6.2 Manual SQLI attack trace:

If any hackers try for injecting the SQL injection through the given form it will match the given input that has assigned to the PHP regex like /[=*'--;]/. Then it will provide the email notification, attacker information which are like below:


SQl Injection Alert

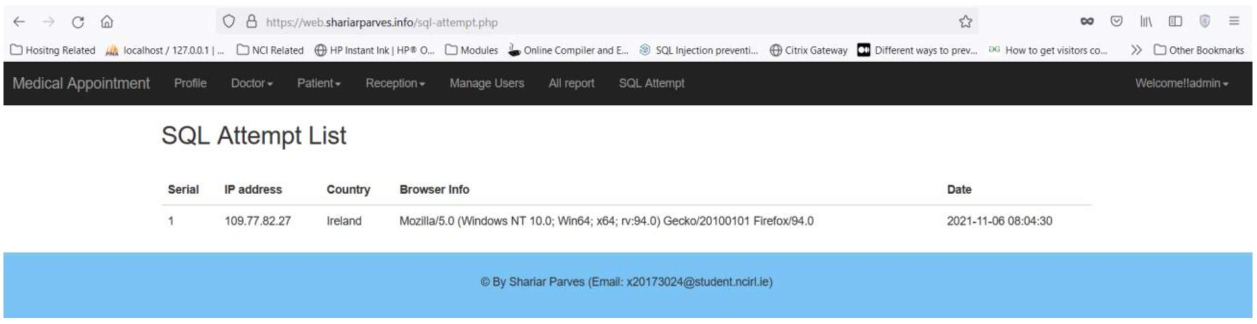
info@shariarparves.info

to me

12:04 PM (9 minutes ago)

Dear User,
Some one attempted SQL injection on
IP: 109.77.82.27 and
Location: Ireland
URL: https://web.shariarparves.info

Figure 12: Email notification for SQL injection attempt.


https://web.shariarparves.info/sql-attempt.php

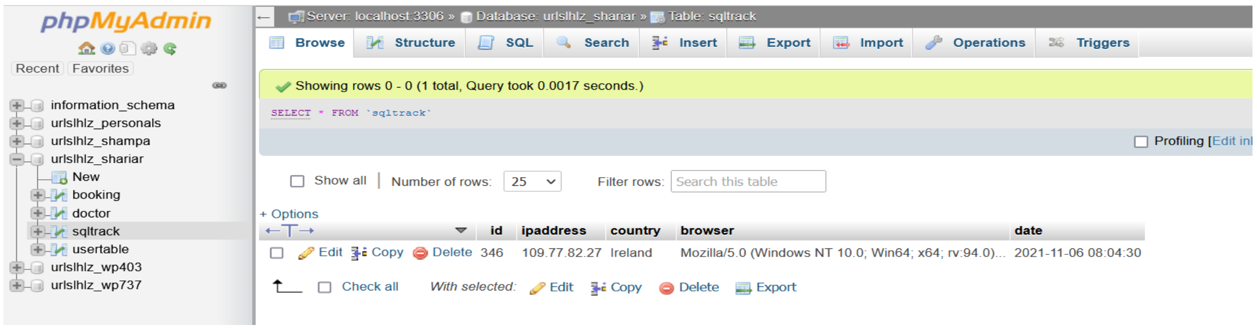
Medical Appointment | Profile | Doctor | Patient | Reception | Manage Users | All report | SQL Attempt | Welcome!!admin

SQL Attempt List

Serial	IP address	Country	Browser Info	Date
1	109.77.82.27	Ireland	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0	2021-11-06 08:04:30

© By Shariar Parves (Email: x20173024@student.ncirl.ie)

Figure 13: Admin panel SQLI attempt details.

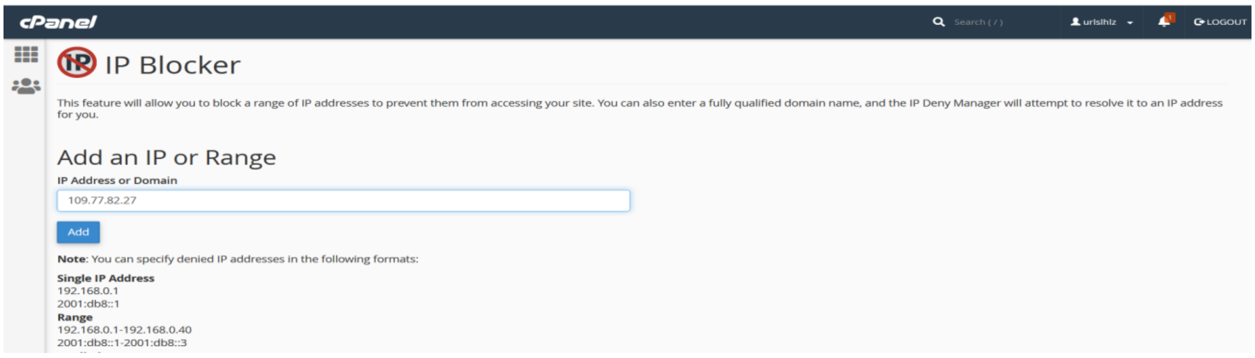

Server: localhost:3306 » Database: urlslhlz_shariar » Table: sqltrack

Showing rows 0 - 0 (1 total, Query took 0.0017 seconds.)

```
SELECT * FROM `sqltrack`
```

	id	ipaddress	country	browser	date
Edit Copy Delete	346	109.77.82.27	Ireland	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0)...	2021-11-06 08:04:30

Figure 14: SQLI notification store in database.



IP Blocker

This feature will allow you to block a range of IP addresses to prevent them from accessing your site. You can also enter a fully qualified domain name, and the IP Deny Manager will attempt to resolve it to an IP address for you.

Add an IP or Range

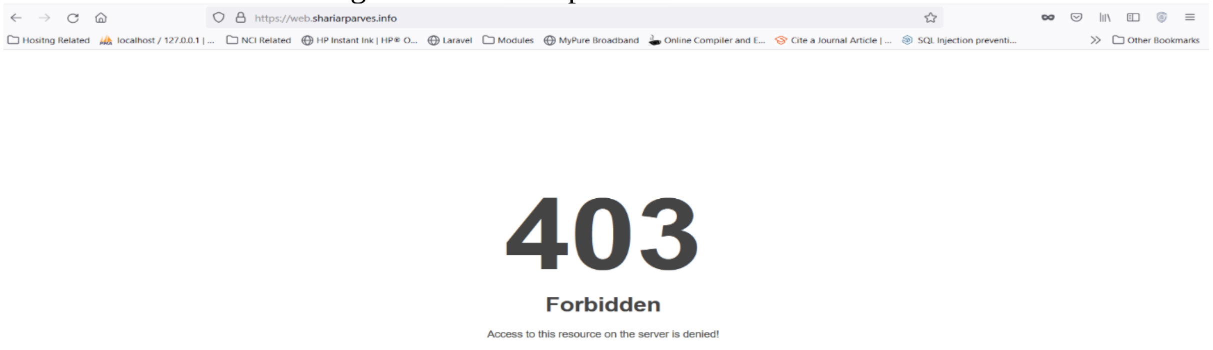
IP Address or Domain: 109.77.82.27

Add

Note: You can specify denied IP addresses in the following formats:

Single IP Address
192.168.0.1
2001:db8::1
Range
192.168.0.1-192.168.0.40
2001:db8::1-2001:db8::3

Figure 15: Add suspicious IP to IP blocker.


https://web.shariarparves.info
Hositng Related
localhost / 127.0.0.1 | ...
NCI Related
HP Instant Ink | HP® O...
Laravel
Modules
MyPure Broadband
Online Compiler and E...
Cite a Journal Article | ...
SQL Injection preventi...
Other Bookmarks

403

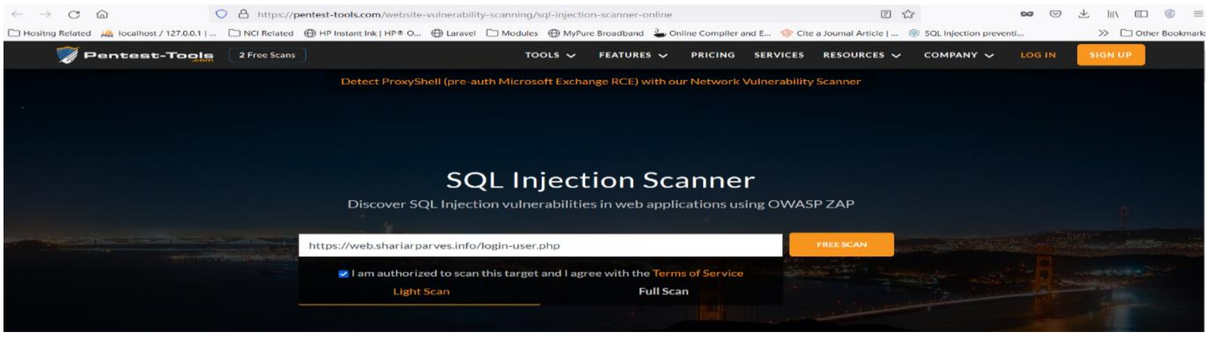
Forbidden

Access to this resource on the server is denied!

Figure 16: Specific IP block to restrict access.

6.3 Automated SQLI attack trace:

To do the pen test, automated tools has been used for executing SQL injection towards the targeted web application. In that case for the implementation tools (https://pentest-tools.com) has been used.


https://pentest-tools.com/website-vulnerability-scanning/sql-injection-scanner-online
Hositng Related
localhost / 127.0.0.1 | ...
NCI Related
HP Instant Ink | HP® O...
Laravel
Modules
MyPure Broadband
Online Compiler and E...
Cite a Journal Article | ...
SQL Injection preventi...
Other Bookmarks
Pentest-Tools
2 Free Scans
TOOLS
FEATURES
PRICING
SERVICES
RESOURCES
COMPANY
LOG IN
SIGN UP

Detect ProxyShell (pre-auth Microsoft Exchange RCE) with our Network Vulnerability Scanner

SQL Injection Scanner

Discover SQL Injection vulnerabilities in web applications using OWASP ZAP

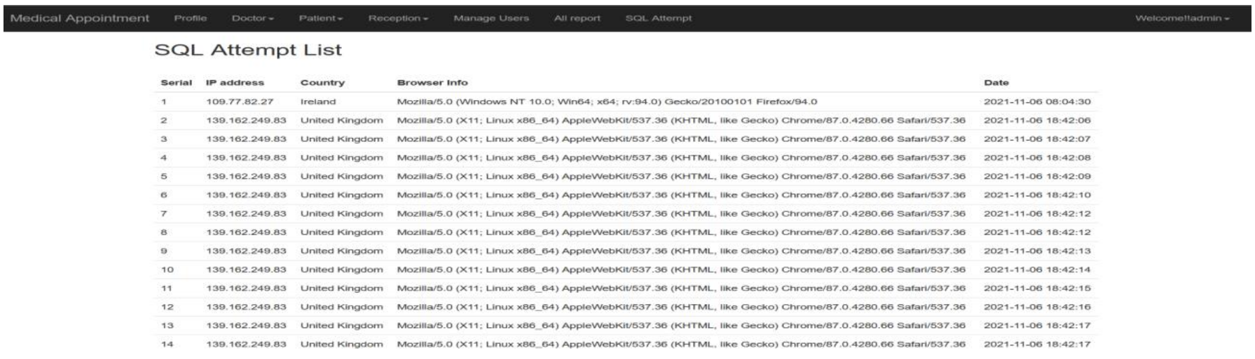
https://web.shariarparves.info/login-user.php

FREE SCAN

☑ I am authorized to scan this target and I agree with the Terms of Service

Light Scan   Full Scan

Figure 17: Automated SQLI attack by Pentest Tool.



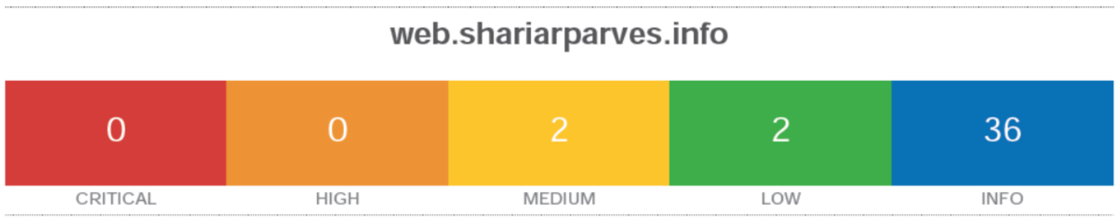
SQL Attempt List

Serial	IP address	Country	Browser Info	Date
1	109.77.82.27	Ireland	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0	2021-11-06 08:04:30
2	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:06
3	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:07
4	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:08
5	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:09
6	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:10
7	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:12
8	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:12
9	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:13
10	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:14
11	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:15
12	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:16
13	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:17
14	139.162.249.83	United Kingdom	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36	2021-11-06 18:42:17

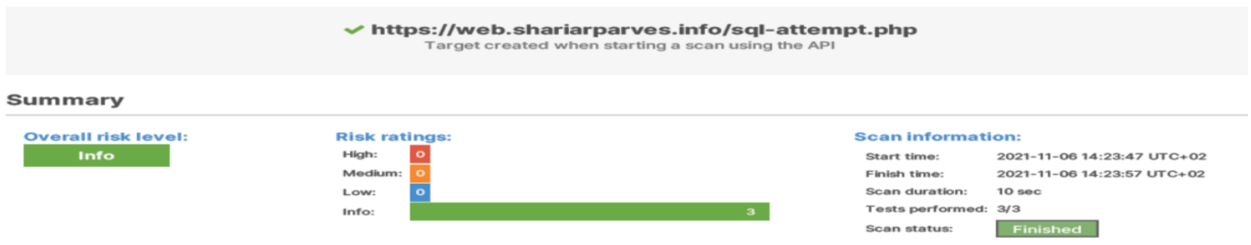
Figure 18: Notification on admin panel

6.4 Vulnerability Scan SQL:

Nessus scan: has been used to confirm and check the SQL vulnerability in the system.



Pentest Scan: After doing SQL pentest through online tools nothing found about the SQLI vulnerability.


https://web.shariarparves.info/sql-attempt.php
Target created when starting a scan using the API

Summary

Overall risk level: Info

Risk ratings:

High:	0
Medium:	0
Low:	0
Info:	3

Scan information:

Start time:	2021-11-06 14:23:47 UTC+02
Finish time:	2021-11-06 14:23:57 UTC+02
Scan duration:	10 sec
Tests performed:	3/3
Scan status:	Finished

6.5 Discussion

In terms of the SQLI attack various types of tests has been executed for checking the result and outcome by using different input to the form. Input validation and parameterised has been checked in the beginning, implemented successfully by using the proper evaluation.

Second steps, SQLI inputted manually to the application, and it triggered email notification instantly to the admin by including attacker details. IP address, Location, OS information, browser details will store in database. Moreover, admin of the system can view hackers' details by accessing admin panel. IP address can be blocked from the cPanel by Admin access which will help to restrict access.

In terms of the third steps, by using the automated tools SQLI attack has been implemented. However, it triggered alert successfully to system admin. Result was successful in case of the notification.

At the end on fourth steps, Nessus scanner, pen test tools used for scanning the application for checking SQLI vulnerabilities and result was nothing found in the system, and it is safe application.

After the successful evaluation, if suspected IP can block automatically which would be more convenient instead of blocking the IP manually from the Cpanel.

7 Conclusion and Future Work

SQLI injection is most common and leading threat for dynamic web-based application. Information in the database that relate to the web application which needs to be protected from threats for keeping confidentiality, integrity of the system. By the improvement of the technology, dependency of the digital technology is increasing, and we keep important and sensitive information to the system to get available access anytime. In that case, hackers trying to invent different strategy for making cyber-attack to the target system by using the SQLI attack. However, SQLI attack is considered as the number one attack as well as security threat to the dynamic web-based application in terms of the top ten security in OWSAP. In accordance with the different research, it has been found that SQLI is the second vulnerability which exploit and threat for security of WordPress CMS after cross site scripting (XSS) attack. Consequence of the SQLI is very harmful, in that case organisation must be very careful while they develop any application that contain and store sensitive information in the database. Proposed method that will help for the web application to detect code injection to provide the protection of the system to keep the database safe.

In terms of the future work and extension, evaluation can be extended, different and more testing tools can be used to increase more accuracy about the application. If suspicious IP can block automatically that will be more convenient as well.

References

Acunetix. 2021. SQL injection attack. [ONLINE] Available at: https://www.acunetix.com/websitesecurity/sql-injection/. [Accessed 20 October 2021].

Cyware Social. 2021. Recent SQL injection attack. [ONLINE] Available at: https://cyware.com/news/sql-injection-is-still-a-critical-attack-vector-e535dcc3. [Accessed 21 October 2021].

S. Saleem, M. Sheeraz, M. Hanif and U. Farooq, "Web Server Attack Detection using Machine Learning," 2020 International Conference on Cyber Warfare and Security (ICCWS), Islamabad, Pakistan, 2020, pp. 1-7, doi: 10.1109/ICCWS48432.2020.9292393.

K. Kuroki, Y. Kanemoto, K. Aoki, Y. Noguchi and M. Nishigaki, "Attack Intention Estimation Based on Syntax Analysis and Dynamic Analysis for SQL Injection," 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 2020, pp. 1510-1515, doi: 10.1109/COMPSAC48688.2020.00-41.

D. Patel, N. Dhamdhere, P. Choudhary and M. Pawar, "A System for Prevention of SQLi Attacks," 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2020, pp. 750-753, doi: 10.1109/ICOSEC49089.2020.9215361.

Z. C. S. S. Hlaing and M. Khaing, "A Detection and Prevention Technique on SQL Injection Attacks," 2020 IEEE Conference on Computer Applications(ICCA), Yangon, Myanmar, 2020, pp. 1-6, doi: 10.1109/ICCA49400.2020.9022833.

P. Lin, W. Jinshuang, C. Ping and Y. Lanjuan, "SQL Injection Attack and Detection Based on GreenSQL Pattern Input Whitelist," 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 2020, pp. 187-190, doi: 0.1109/ICISCAE51034.2020.9236824.

H. Alsobhi and R. Alshareef, "SQL Injection Countermeasures Methods," 2020 International Conference on Computing and Information Technology (ICCIT-1441), Tabuk, Saudi Arabia, 2020, pp. 1-4, doi: 10.1109/ICCIT-144147971.2020.9213748.

A. Jana, P. Bordoloi and D. Maity, "Input-based Analysis Approach to Prevent SQL Injection Attacks," 2020 IEEE Region 10 Symposium (TENSYMP), Dhaka, Bangladesh, 2020, pp. 1290-1293, doi: 10.1109/TENSYMP50017.2020.9230758.

I. Tasevski and K. Jakimoski, "Overview of SQL Injection Defense Mechanisms," 2020 28th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2020, pp. 1-4, doi: 10.1109/TELFOR51502.2020.9306676.

M. Al Rubaiei, T. Al Yarubi, M. Al Saadi and B. Kumar, "SQLIA Detection and Prevention Techniques," 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, 2020, pp. 115-121, doi: 10.1109/SMART50582.2020.9336795.

W. H. Rankothge, M. Randeniya and V. Samaranayaka, "Identification and Mitigation Tool for Sql Injection Attacks (SQLIA)," 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS), RUPNAGAR, India, 2020, pp. 591-595, doi: 10.1109/ICIIS51140.2020.9342703.

O. Ojagbule, H. Wimmer and R. J. Haddad, "Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP," SoutheastCon 2018, St. Petersburg, FL, USA, 2018, pp. 1-7, doi: 10.1109/SECON.2018.8479130.

Pentest Tools. 2021. SQLI Scanner. [ONLINE] Available at: https://pentest-tools.com/website-vulnerability-scanning/sql-injection-scanner-online. [Accessed 4 November 2021].

Tenable. 2021. Nessus Scanner. [ONLINE] Available at: https://www.tenable.com/products/nessus. [Accessed 4 November 2021].