Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 135 (2018) 678–687

Procedia Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Computer Science and Computational Intelligence 2018

Prevention Structured Query Language Injection Using Regular Expression and Escape String

Benfano Soewito[a], Fergyanto E. Gunawan[b], Hirzi[c], Frumentius[a]

[a]*Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480*
[b]*Industrial Engineering Department, BINUS Graduate Program - Master of Industrial Engineering, Bina Nusantara University, Jakarta, Indonesia 11480*
[c]*Investment Coordinating Board of the Republic of Indonesia, Jakarta, Indonesia 12190*

Abstract

Information technology enables for new way of commerce, which is a commerce activities through online media (e-commerce). Security becomes an important issue in online-system, because such system is accessible by anyone through the global network - internet. Security in terms of confidentiality, integrity, and availability becomes goals that must be achieved by any system generally, and commerce system especially, because this kind of system contains many sensitive data like customer data and transaction data. SQL Injection is a vulnerability and threat, with the most occurrence in a web based system. In this research we evaluate and analyze source code against SQL injection, and we use regular expression and escape string to prevent the SQL injection. The results of this study are findings of system vulnerability against SQL injection, which are proven by the ability to get data from database, with SQL injection techniques. System vulnerabilities were analyzed, in order to design and implement the solution for it. The solutions then tested, to validate that it has proven effectively fix the vulnerabilities, and can prevent the exploitation by SQL injection. In the end, the conclusion is that initially the system is vulnerable against SQL injection, but then the solution that being implemented has proven effectively fix the issue.

Keywords: SQL Injection; regular expression; escape string; e-commerce vulnerability

1. Introduction

The development of internet and world wide web (WWW) technology with its web page pages, provides opportunities for new business opportunities. A service or product can be closer to the customer or prospect by utilizing

* Corresponding author. Tel.: +62-817-237-9554 ; fax: +0-000-000-0000.
E-mail address: benfano@gmail.com

1877-0509
10.1016/j.procs.2018.08.218

internet technology and website. Electronic Commerce (E-commerce) is one example of internet utilization. Service providers or products can penetrate new market shares and enlarge their customer coverage, products and services can be more easily accessed, viewed and recognized, even purchased by customers.

On the other hand with the increasingly widespread use of technology for various purposes in particular the field of business, the risks and threats of security threats to e-commerce system also developed[1]. In e-commerce where the customer can not see directly the seller or his products require high trust, so the threat of security threats can lower the level of customer trust to conduct online transactions and can make the main purpose of e-commerce system development becomes less or even ineffective.

Until now according to the survey there are various types of vulnerabilities and threats or attacks that exist and can be done into a website or web-based applications, such as: XSS or abbreviations of Cross Side Scripting, Injection (SQL, XML, LDAP, SMTP Headers, etc.), Broken Authentication and Session Management or often known as Session Hijacking (session hijacking), and many other vulnerabilities and attacks[2]. Based on the same survey also explained that the impact of exploitation and attack on the security gap can be very fatal impact for the website and web application itself or for organizations and businesses.

SQL Injection (SQL Injection) is the most important (most done, and most easily done) of websites or web based application applications that have data bases[3]. Under such circumstances, we need an effort to identify vulnerabilities and analysis of the design of preventive solutions to SQL Injection attack[4], to this e-commerce web application. In many systems and online transactions connected to a free and wide network like the internet, anyone can access and then exploit the vulnerability of existing system vulnerabilities. SQL injection is the most common type of attack and is also very easy to do by skilled and unskilled people[5]. Problems that may arise from exploits via SQL Injection to online shop or e-commerce include:

1. Damage and or loss of data.
2. Leaking the data to unauthorized parties.
3. Use of unauthorized access rights.
4. Loss of access ability by the authorities.
5. Web piracy or fraud, by hijackers.

In order for a web site and web application to be free of SQL injection attacks, it is necessary to validate of inputs, URL parameters, and other variables which is dynamically be queried into the database so that the contents of the input entries can match with the desired format. Practice of penetration or simulation through input data manipulation and URL parameters, as well as scanning using software, can be used to test and see if web applications built are currently free of SQL injection attacks or there is still a loophole in SQL injection vulnerability. In this work, we have analyzed a web-based online sales (e-commerce). We identify the vulnerability gaps that exist on the system, which can be exploited by SQL Injection attacks and provide and implement solutions to resolve loopholes and minimize possible exploitation of the system.

2. Literature review

Web applications are applications that are accessed through a network such as the Internet or intranet, and the computer software and app is run using a browser. The computer software is created with a programming language code supported by the browser (such as JavaScript, combined with Hyper Text Markup Language [HTML] and Cascading Style Sheet [CSS])[6,7].

According to Elshazly, K.[8], In the beginning web applications built using only a language called HTML. In subsequent developments, a number of scripts and objects were developed to extend HTML capabilities such as PHP and ASP on scripts and applets on objects. Web applications can be divided into two types: static web apps and dynamic web. Static Web is formed by using HTML[9]. The shortcomings of such applications lie in the need to maintain the program continuously to keep up with every development that occurs. This weakness is addressed by a dynamic web app model. In dynamic web applications, information changes in web pages are done without program changes but through data changes. As an implementation, web applications can be connected to the database so that information changes can be made by the operator and not the responsibility of the webmaster. Web application architectures in-

clude clients or users, web servers, middleware and databases[10]. The client interacts with the web server. Internally, web servers communicate with middleware and middleware that communicate with databases. Examples of middleware are PHP and ASP. In dynamic web application mechanism, an additional process occurs that the server translates PHP code into HTML code. PHP code that is translated by PHP engine that will be accepted by client.

According to Danile Minoli and Emma Minoli[11], e-commerce is a buying and selling activity of goods or services through the internet and online ordering system. Buying and selling activities via the internet is made possible by the existence of a technology called the world wide web or web for brevity. Through this system the customer can buy enough goods by accessing the website of the provider of goods and services, choosing the desired product and ordering, even to the payment on the web page, which then the product can be directly owned, or delivered to the desired address.

SQL is a programming language specifically designed to manage databases contained in the relational database management system (RDBMS)[12]. Paul Wilton and John W. Colby[13] suggests that SQL has 3 main functions, namely:

1. Create a database along with its structure
2. Querying the database to display the data needed to answer a question
3. Controlling database security

The query itself means retrieving data from the database, so the SQL query can be interpreted as any SQL code used to retrieve data from the database. SQL has a special sub-language concerning query and data manipulation. This sub-language is known as the data manipulation language (DML). There are 4 main commands in DML that can be described as follows:

Table 1. Visual Representation Table student_data.

Student_Reg_Number	Name	Date_of_birth	Gender

2.1. INSERT

The INSERT command is useful for inserting data into tables in the database. For example, suppose there is a student data table with the following structure:

TABLE student_data:
Student_Reg_Number (INT)
Name (VARCHAR 50)
Date_of_birth (date)
Gender (CHAR 1)

The structure of the table above can be visualized as in Table 1. To enter data into the table with the INSERT command, as follows: INSERT INTO VALUES student data ('1812401368', 'Jhon', '2002-12-05', 'Male'); then the table will be filled with the data just entered and can be visualized to be like in Table 2.

Table 2. Visual Representation Table student_data after INSERT data.

Student_Reg_Number	Name	Date_of_birth	Gender
1812401368	Jhon	2002-12-05	Male

2.2. SELECT

The SELECT command is used to display and retrieve data or set and a set of data from the database. Example: Using the table student_data in Table 2, if you want to display the name of a student, then can be made SQL query

as follows: SELECT name FROM data WHERE Student_Reg_Number = 1812401368; WHERE statements are combined with the SELECT command, in order to filter the data that you want to display, in the example, the name of the student to be taken, filtered only by Student_Reg_Number 1812401368.

2.3. UPDATE

The UPDATE command is useful for making changes to existing data in the database. For example, suppose from table student_data in Table 2, student name with Student_Reg_Number 1812401368, want to be changed into Jessica and its gender want to be changed to Female, then the SQL syntax as follows: UPDATE student_data SET Name = 'Jessica', gender = 'Female' WHERE Student_Reg_Number = 1812401368.

2.4. DELETE

As its command name, the DELETE command serves to delete one or more rows of data from within the database. DELETE FROM student_data; This syntax will delete all data in the table student_data. DELETE FROM data WHERE Student_Reg_Number = 1812401368; WHERE statements like the above example provide a filtering function in the DELETE command, where the data to be deleted is only data with Student_Reg_Number 1812401368.

3. Methodology

SQL injection can be interpreted as a form of attack and also as a form of weakness. According to Mayang Namdev et al.[12] that SQL injection is a vulnerability in a web application, where an attacker exploiting this vulnerability can send SQL commands, in order to access the database of the web application, which the command will be executed by the web application itself.

SQL injection can also be seen as a form of attack by injection and insert SQL queries into an input medium, which is done from the client side to a web application. SQL injection that allows attackers to access into the application database, allowing attackers to read and even manipulate (insert, modify, and delete) sensitive data, resulting in threats of security threats such as

1. Identity fraud
2. Modification to data destruction
3. Unauthorized transactions
4. Complete data disassembly by attackers
5. Data can no longer be accessed as it should by the authorities
6. Takeover of database administration rights.

Anhar[5] describes an attacker's general methodology in the activities of exploiting web application systems by utilizing SQL injection as shown in the diagram in Fig. 1. This also will be our methodology to do our research.

First of all in the process of enumeration analysis and scanning web application, it will be collected data data such as elements of e-commerce application system itself, such as architecture of web application, and databases related to this web application system, as well as other documentation[14]. In the next stage will be selected tools to attack into the system using SQL injection techniques. Automatic assault test tools are also collected that can scan and exploit SQL injection of web applications automatically.

From the test results will be identified and analyzed the point of security vulnerabilities contained in the program code, so that can be designed solutions or handling solution to prevent the attack. The process of solution design can be done by library and desk research method in searching and collecting techniques of SQL injection prevention techniques in general[15], then techniques will be applied to the application, by modifying the program code. Finally, the designed and implemented solutions will be validated using the same test when identifying the application system security holes. In our experiment we did manual injection testing and then proposed the security handling solution. Manual testing scenarios (not using automatic injection tools) to be performed such as:

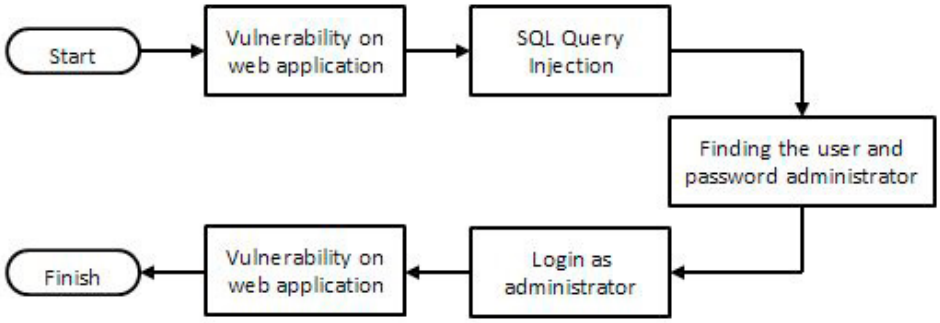

Fig. 1. General Steps of Web Exploits With SQL Injection.

- Firstly, with incorrect query techniques, check whether the input through the box, or design URL parameters directly into the query. Testing can be done by inserting a single (‘) quote or double quotes (”). If the execution results show an error message from SQL means that the input is not filtered so that the directly executed quotes cause errors in the SQL syntax. After knowing whether the user input is filtered or not, then if that does not mean the application is vulnerable to SQL injection.
- Then with the UNION query injection technique try to dig the data and display the data which is in the database. In order for the UNION query technique to be executed correctly, it is necessary to align the number of columns that are queried in the existing SQL command in the program code, with the UNION query command injected, so the next step is to find out the number of columns with the alleged technique. The alleged technique is done by entering integer numbers as UNION query columns, starting from one column, and adding them one by one until execution does not produce an error message.

In general, the methodology and environment which is done in this research can be described as follow,

- The research is conducted in a separate environment from the original system is located, meaning that the system installed in the test environment so as not to interfere with the original system in used by company.
- The test environment is an environment built using devices that have been translated into research instruments.
- In addition to tracking security holes (vulnerability) that identify the presence of loopholes, penetrate into the gap was done as well to get data from database server.
- Penetration in research conducted in two ways, first, tracking by manually injection into all user input fields and URL parameters, and the second performs scanning automatically using the SQL Map scanner tool.
- The result of scanning is dumped into a file to be read and used for analysis purposes.
- The results of both manual and automatic tracking and penetration in further analysis to make appropriate repair and handling solutions for each gap.

Our analyze method start with identifying the security gap, and then perform the design analysis of the solution to overcome the gap, initially from aggregated assault and handling data is studied and understood in advance. It then identifies the vulnerabilities of pages, ie the pages that have input form and the url parameter. On those pages will be scanned using SQL injection scanning software tools (software tools), and further traced and studied the program code page and other pages concerned with the page, to see if there is a weakness in the program code and made code of injection code to perform penetration testing. The results of the penetration test and scanning with tools will show the vulnerability gaps found in the application. From the data security gap that were found, then will be designed a solution for the improvement of the program code, as well as its data base. Once the solution is implemented, validation will be performed by performing another penetration test and scanning done earlier. If there is still a security hole will be created again handling solutions and re-validated the solution again. If no longer found security hole (vulnerability) then the analysis process is complete.

4. Results and discussion

We did the experimental on an e-commerce website because of according to Mayang Namdev [12] and also based on authors experience, in order to prevent SQL Injection we must pay attention to the format and characters that input through the form. All the forms in websites are the same. In this research, we started select on of an e-commerce website from our client. We have identified that the website has 79 box that contain <form> </form> from 18 pages. Then, we made identification of the user input space or box into 10 categories, based on the type and the execution query command performed on the input received through the box.

Table 3. Query execution commands and type.

Numer	Type	Query Execution Commands
1	text	SELECT
2	text	INSERT
3	text	UPDATE
4	textarea	Not Queried
5	password	SELECT
6	password	INSERT
7	password	UPDATE
8	select	SELECT
9	select	INSERT
10	select	UPDATE

From the Table 3 can be seen and filtered that the category-4, the column with the type textarea can not be injected with SQL because the data entered in the column is not queried further to the database, so to the textarea no need to test SQL injection. For the above identified URL Parameters have the same characteristics, which is data from those parameters are executed further or query to the database with the SELECT command, so it can be entered into one group and in the SQL injection test to one of the parameters only.

4.1. Manual Injection Testing

This test is done to the user input field with the password field type. The field used for the trial is a password log in column located on the user's Log In page in /user/login.php. Assume its SQL query command format is SELCT <column_name> FROM <table_name> WHERE <nama_column_email> = <input_from_column_email> AND <name_column_password> = <input_of_column_password> . IIn this test, because on the user's Log In page requires an input other than the password field but also in the email column as the user's identification, then before performing the test, the tester attempts to register an account with the following information; Email: frumentius.kwee@gmail.com Password: 12345678

4.1.1. Security Handling Solution

From the above test results found that e-commerce web applications vulnerable to SQL injection. Vulnerability occurs due to unfiltered input or parameters before being queried to the database, so the attacker can manipulate the input or parameter contents into SQL injection code.

The concept of handling SQL injection is actually very simple, ie ensuring the format of the input data or the contents of the parameters to be queried in the form of the desired format. For example when wanting to give an id in the form of an integer number, then before query input id need to be sure that id really true in the form of integer, or for example want query name of person, hence before data queried need to be sure that data is true string which consists only of alphabet A to Z or a to z or space.

Other prevention concepts can also, if we can not ascertain the pattern of the input string or character because of the breadth of possible inputs, then at least it must be ensured the input or parameter content is not in the form of SQL command or syntax. Based on the above concept there are several techniques that can be used in its implementation of fixing the program code to close SQL injection vulnerabilities, which are:

1. Regular Expression Regular expression is a function that can search for a pattern on a string, with regular expression can be designed pattern such as, pattern number filtering, email format pattern, username pattern pattern, etc. to filter input user input or URL parameter.
2. Prepare Statement The actual prepare statement, having a primary function so that the program code that performs queries with the same pattern repeatedly can be efficiented by creating a prepared statement for queries with that pattern and can be used together for the query's need with that pattern. At the same time prepare statements also have the advantage of avoiding SQL injection due to the execution process of preprocessing statements in which first statements are compiled and recorded without the contents of the parameters, in which the contents of the statement parameter can be either the contents of the user input field or the contents of the URL parameters, then the parameters are connected with the contents and executed. With this process model the SQL pattern is pre-stored so it can not change again even though the contents of the parameters are SQL code, until the content of those parameters even containing the SQL injection code does not have the ability to change the SQL query command format and is considered just an ordinary string.
3. MySQL Escape String Escape string is a useful function to detect important characters in forming a SQL syntax, if the character is detected it will automatically be modified or affixed another character (escape) so as not to form SQL syntax, so it becomes syntax error or can so executed into a normal string. Characters to be escaped in this case include:

   (a) \x00 modified to \\x00
   (b) \n modified to \\n
   (c) \r modified to \\r
   (d) ' modified to \'
   (e) " modified to \"
   (f) \x1a modified to \\x1a

With the concept and the three techniques above, the prevention of SQL injection can be implemented into the following program code:

4.1.2. URL parameters (using regular expression)

Firstly regular expression is applied to the configuration file " .htaccess " to filter the pattern of the contents of the parameter to fit the proper format. It is as if rewriting the page URLs into different forms (rewrite URL). Still the same test is implemented on the article details page, along with its .htaccess code:

```
RewriteEngine on
RewriteRule ^([0-9]+)/([0-9]+)/([a-zA-Z0-9 _-]+) \.html$
artikel.php?url_thn=$1&url_id=$2&url_title=$3
```

The command of "RewriteEngine on" is useful to enable URL rewrite feature, then the pattern is expressed in RewriteRule in regular expression. You can see the parameters of the URL / blog / article.php there are three ie url_thn, url_id, and url_title. Each of these parameters applies the regular expression pattern as follows (each regular expression by character "/" [slash]).

([0-9] +), this expression is applied to the url_thn and url_id parameters. This expression filters strings and only allows numbers to be the contents of those parameters, where we know that the year and id are only numbers.

([a-zA-Z0-9_-]), this expression filters the contents of the url_title parameter, so that the contents of the parameter are numbers, alphabet lowercase and capital letters, underscore characters, and "-" characters. By applying the above technique the URL of the blog article page with article id 13, changed from /blog/artikel.php?id=13 to /blog/2015/13/latest-loves-body-oils-by-nars.html. Regular expression is also applied to filter when the parameters are taken into php file, before then was queried to the database. Here's a snippet of the program php code.

```
$url_thn = '';
if(isset($_GET['url_thn'])) $url_thn = preg_replace('/[^0-9]/s', '',
$_GET['url_thn']);
```

```
$url_id = '';
if(isset($_GET['url_id'])) $url_id = preg_replace('/[^0-9]/s', '',
$_GET['url_id']);
$url_title = '';
if(isset($_GET['url_title'])) $url_judul = preg_replace
('/[^0-9a-zA-Z_-]/s', '', $_GET['url_title']);
```

In php, preg_replace code has 3 parameters. The first parameter is where the desired expression pattern is found, then the second parameter is the surrogate character for the pattern found, and the last parameter is the string you want to be.

In the code implementation above thn and id have the same regular expression ie [^ 0-9] to find all characters in string (parameter 3) which is not number (not marked by character " ^ " on regular expression) and replace it with characters that are not declared or blank. In other words, it will delete its character. While in url_title applied regular expression [^ 0-9a-zA-Z_-] to find all characters that are not numbers, not the alphabet of capital letters and lowercase, not underscore, and not dash or minus, and replace it with undeclared character or blank. When parameters are queried into the database, prepare statement techniques also was implemented as well for their query execution commands, from the old program code of the old query execution command, as follows:

```
$data_detil_artikel = $mysqli->query("SELECT * FROM article WHERE
id = '".$url_id."' AND url_title = '".$url_title."' AND year(tgl)
= '".$url_thn."' AND stat = '1'");
```

It changed to as below to apply prepare statement techniques:

```
$data_detil_artikel = $mysqli->prepare("SELECT * FROM article
WHERE id = ? AND url_title = ? AND year(tgl) = ? AND stat = '1'");
$data_detil_artikel->bind_param('isi', $url_id, $url_title,
$url_thn);
$data_detil_artikel->execute();
$data_detil_artikel->store_result();
```

It can be seen in the prepared statement technique above the indirect query command is fully executed as in the old code, but a SQL query command pattern is set up with the parameters encoded with the "?" Character (question mark), then the parameter is linked to its contents with function bind_param, and then just executed with execute function and the result is stored with store_result function.

4.1.3. Column or box for inputing query (Using escape string)

In columns or box with text type, SQL injection prevention techniques are also applied when the input content to the column is captured before then quaried into the database. Taking the same example with the test before, following the application code of the MySQL Escape String implementation in the product search field:

```
$teks_cari_produk = '';
if(isset($_POST['teks]_cari_produk']))
$teks_cari_produk = $mysqli->real_escape_string
($_POST['teks_cari_produk']);
```

In the case of product search box, an escape string is applied instead of a regular expression because the keyword in the search for a product contains broad characters can be a capital, alphabet, numbers, punctuation marks, and some symbols, therefore instead of designing a complex regular expression and length to specify one by one permitted characters, then we using escape strings to filter only characters that are not allowed, ie character characters that make a string a SQL syntax. The code to execute SQL query commands is also changed to use the prepare statement techniques as follows,

Old code:

```
$data_produk = $mysqli->query("SELECT id, name, price,
price_promo, url_gbr, status FROM product WHERE id_single_item
LIKE '%".$teks_search_produk."%' OR nama LIKE
'%".$teks_search_produk."%' OR kk LIKE '%".$teks_search_produk."%' AND
status = '1' OR status = '2' AND id_sub_kategori > '0' ORDER
BY tgl DESC");
```

New code:

```
$data_produk = $mysqli->prepare("SELECT id, name, price,
price_promo, url_gbr, status FROM product WHERE id_single_item
LIKE '%?%' OR name LIKE '%?%' OR kk LIKE '%?%' AND status = '1' OR
status = '2' AND id_sub_kategori > '0' ORDER BY tgl DESC");
$data_produk->bind_param('sss', $teks_cari_produk,
$teks_search_produk, $teks_search_produk);
$data_produk->execute();
$data_produk->store_result();
```

4.2. Automatic Injection Testing

Testing with the automatic scanner is also done. The automated scanner tool that was used is SQL Map. SQL Map can be used to perform injection tests via URL parameters. The page used in testing with this SQL Map is Product Category page, with parameter URL 'id'. Below is the following command explanation, which is used to scan with SQL Map.

```
sqlmap.py -u "http://localhost/justmiss-project/productcategori.php?id=2" --time-sec=10
--random-agent --dbms=MySQL --keep-alive --level=5 --risk=3 --dbs ?Table
```

From the result above scans we know that the database name used is db_jm_project, along with the tables that exist in the database is also displayed. From these results then we want to display the columns in the tables whose data would like to see, in this test is also taken user_login table.

```
sqlmap.py -u "http://localhost/justmiss-project/product/categori.php?id=2" --time-sec=10
--random-agent --dbms=MySQL --keep-alive --level=5 --risk=3 -D db_jm_project -T
user_login ?columns
```

the result as follow,

```
Database: db_jm_project
Table: user_login
(8 columns)
```

Column	Type
email	varchar(150)
id	tinyint(255) unsigned
login	datetime
name	varchar(100)
password	char(32)
stat	tinyint(1) unsigned
status_user	tinyint(255) unsigned
username	varchar(8)

Lastly after successfully ensuring that indeed the user_login table has important data containing logged in information such as username and password, then through the -dump command, SQL Map pulls out all the contents of user_login table.

```
sqlmap.py -u "http://localhost/justmiss-project/product/categori.php?id=2" --time-sec=10
--random-agent --dbms=MySQL --keep-alive --level=5 --risk=3 -D db_jm_project -T
user_login ?dump
```

The basic security handling solutions are the same that explained in section 4.1.1, 4.1.2, and 4.1.3

5. Conclusion

In this work, the vulnerability to SQL injection is investigated, the process begins by identifying all the URL parameters and user input columns or box, then grouping them by their SQL type and query type. We found that the URL parameters and input columns or box with the text type are vulnerable to SQL injection, The vulnerability of the system is caused by the input and content of URL parameters that are explicitly queried to the database without any validation, filtering, or processing first, therefore it is implemented a handling solution with regular expression, prepare statement, and MySQL escape string techniques. The improved Application system with these three handling solutions is validated by retesting, just like the previous test, and the results of the system are secure and can not be exploited with SQL injection, so it can be concluded that validated handling solutions effectively prevent system overlaid by injection SQL.

References

1. Jamar, R., Sogani, A., Mudgal, S., Bhadra, Y., Churi, P.. E-shield: Detection and prevention of website attacks. *Recent Trends in Electronics Information & Communication Technology (RTEICT)* 2017;:706–710.
2. OWASP, . The ten most critical web application security risk. *OWASP Top 10* 2013;:Available: http://www.owasp.org.
3. OWASP, . Sql injection. *OWASP SQL Injection* 2015;:Available: http://www.owasp.org/index.php/SQL_Injection.
4. Qbea'h, M., Alshraideh, M., Sabri, K.E.. Detecting and preventing sql injection attacks: A formal approach. *Cybersecurity and Cyberforensics Conference (CCC)* 2016;:123–129.
5. Anhar, . Hacking website for newbie. *Book* 2013;:10–54.
6. Ballard, , Phil, , Moncur, M.. Sams teach yourself ajax, javascript, and php all in one. *Book* 2009;:101–154.
7. Deitel H. M., P.J.D., Goldberg, A.B.. Internet & world wide web how to program. *Book* 2004;.
8. Elshazly, K., Fouad, Y., Saleh, M., Sewisty, A.. A survey of sql injection attack detection and prevention. *Journal of Computer and Communications, vol 2, no 8* 2014;:1–9.
9. Sebesta, , W., R.. Programming the world wide web. *Book* 2003;.
10. Kahonge, A.M., Okello-Odongo, W., Miriti, E.K., & Abade, E.. Web security and log management: An application centric perspective. *Journal of Information Security, vol 4, no 3* 2014;:138–143.
11. Minoli Daniel dan Minoli, E.. Web commerce technology handbook. *Book* 1998;.
12. Namdev M., H.F..S.G.. Review of sql injection attack and proposed method for detection and prevention of sqlia. *International Journal of Advance Research in Computer Science and Software Engineering, vol 2, issue 7* 2012;.
13. Wilton P. & Colby, J.W.. Beginning sql. *Book* 2005;.
14. Slay Jill dan Kronios, A.. it security & risk management. *Book* 2006;.
15. Sandeep D Sukhdeve, H.C.. The code sanitizer: Regular expression based prevention of content injection attacks. *International Journal of Computer Trends and Technology, Volume 35, Number 1* 2016;:21–28.