International Journal of scientific research and management (IJSRM)
\||Volume||3||Issue||5||Pages|| 2786-2789||2015||
Website: www.ijsrm.in ISSN (e): 2321-3418

Proactive Mechanism of Protection against SQL Injection Attack

Surya Pratap Singh[1], Avinash Singh[2], Upendra Nath Tripath[3], Manish Mishra[4]

[1]Department of Computer Science
Deen Dayal Upadhyay Gorakhpur University , Gorakhpur -273001
spsingh8161@hotmail.,com

[2]Department of Computer Science
Deen Dayal Upadhyay Gorakhpur University , Gorakhpur -273001
avinashnero2007@gmail.com

[3]Department of Computer Science
Deen Dayal Upadhyay Gorakhpur University , Gorakhpur -273001
untripathi@gmail.com

[4]Department of Electronics
Deen Dayal Upadhyay Gorakhpur University , Gorakhpur -273001
mamnishm@yahoo.in

Abstract: In this fast growing environment most of the organizations are now computerized and those are not seeking computerization. As the use of computer increases the need of the database is also increasing and hence the database security is very important aspect. Various types of attackers and database hackers are trying everything to break the security of the database and to spoof the information contained in the database. The SQL Injection Attack is a very big security threat in the modern databases. An attacker tries to exploit faulty application code by causing maliciously crafted database queries. The attackers are allowed to obtain unauthorized access to the backend database by submitting the malicious SQL query segment to change the intended application. Various database researchers and practitioners provide different approaches to prevent from SQLIA but none of the approach is fully capable to. In this paper we propose some methods of protection against SQL injection attack.

Keywords: SQL Injection attack, Weak Authentication, Malicious query, SQL Query Monitor.

1. Introduction

SQL injection vulnerabilities are one of the most serious threats for the Web based applications. Web based applications that are vulnerable to SQL injection may allow an attacker to gain complete access to their database, just because these databases often contain sensitive information.[1][2] The resulting security violations can include identity theft, fraud and loss of confidential information .In some cases, attackers can use an SQL injection to take control of and corrupt the system that host the Web application too.

SQL injection refers to a class of code injection attack in which data provided by the user is included in an SQL query in such a way that part of the user's input is treated as SQL code.[4]

SQL Injection Attacks (SQLIA's) are one of the most severe threats to web application security. They are frequently employed by malicious users for a variety of reasons like financial fraud, theft of confidential data, website defacement, sabotage, etc. The number of SQLIA's reported in the past few years has been showing a steadily increasing trend and so is the scale of the attacks.[5]

For preventing the SQLIA, defensive coding has been can give a solution but is very difficult to implement, not only developers try to put some controls in their program code but also attackers continue to bring some innovative way to

bypass these controls. Hence it is very difficult to keep ourselves updated according to the last and the best defensive coding solutions. These problems motivated the need for a solution to the SQL injection problems.[6]

2. Definition of SQLIA

SQL injection is a type of attack which the attacker adds Structured Query Language code to input box of a web form to gain access or make changes to data. SQL injection vulnerability allows an attacker to flow commands directly to a web application's underlying database and destroy functionality or confidentiality. [7][9]

Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.

3. Process of SQL injection

The SQL injection consist of direct insertion of code into user-input variables that are connected with SQL commands and executed. A less direct attack injects malicious code into string that are designed for storage in a table as a metadata. When the stored string are concatenated into dynamic SQL command , the malicious code is executed .[8]

The injection process works by prematurely terminating a text string and appending a new command. Because the inserted command may have additional strings appended to it before it is executed, the malefactor terminates the injected string with a comment mark "--". Subsequent text is ignored at execution time.[10]

The following script shows a simple SQL injection. The script builds an SQL query by concatenating hard-coded strings together with a string entered by the user:[11][12]

varShipproduct;

Shipproduct = Request.form ("Shipproduct");

varsql = "select * from OrdersTable where Shipproduct = '" + Shipproduct + "'";

The user is prompted to enter the name of a city. If she enters Redmond, the query assembled by the script looks similar to the following:

SELECT * FROM OrdersTable WHERE Shipproduct = 'Redmond'

However, assume that the user enters the following:
Redmond'; drop table OrdersTable—

In this case, the following query is assembled by the script:
SELECT * FROM OrdersTable WHERE Shipproduct = 'Redmond';drop table OrdersTable--'

4. Review of Literature

The techniques which are currently available can cover a subset of the vulnerabilities of the SQL Injections, some work of the researchers are listed in the following section-

Huang and colleagues [1] propose WAVES, a blackbox technique for testing web applications for SQL injection vulnerabilities. The tool identify all points a web application that can be used to inject SQLIAs. It builds attacks that target these points and monitors the application how response to the attacks by utilize machine learning.

Swaddler [2], analyzes the internal state of a web application. It works based on both single and multiple variables and shows an impressive way against complex attacks to web applications. First the approach describes the normal values for the application's state variables in critical points of the application's components. Then, during the detection phase, it monitors the application's execution to identify abnormal states.

Roichman and Gudes's Scheme – [3] suggests using a fine-grained access control to web databases. The authors develop a new method based on fine-grained access control mechanism. The access to the database is supervised and monitored by the built-in database access control. This is a solution to the vulnerability of the SQL session traceability.

WebSSARI [4] use static analysis to check taint flows against preconditions for sensitive functions. It works based on sanitized input that has passed through a predefined set of filters. The limitation of approach is adequate preconditions for sensitive functions cannot be accurately expressed so some filters may be omitted.

Shaukat Ali et al.'s Scheme – [5] adopts the Hash value approach to further improve the user authentication mechanism. They use the user name and password Hash values SQLIPA (SQL Injection Protector for Authentication) prototype was developed in order to test the framework. The user name and password Hash values are created and calculated at runtime for the first time the particular user account is created

5. Problems in Existing Systems

The SQLIA is the major security threat to the database applications in web environment. There are various approaches to overcome from these problems but the existing SQLIA prevention methodologies have some major problems which are discussed bellow –

a) SQLIA is a hacking technique which the attacker includes SQL statements through web application's input fields or by means of hidden parameters to access the resources which they do not have access to. The lack of proper input validation causes the attacker to be able to intrude in the database system by unfair ways .
   The process of SQLIA is explained by the following example –
   Assume a web application receives a HTTP request from a client as input and generates a SQL statement as output for the database server. For instance the database administrator will be authenticated by empid = 123 and password = administrator. Fig 1 describes a login by an intruder by exploiting SQL injection vulnerability.
   i) An attacker sends the malicious HTTP request to the web application .
   ii) Creates the SQL statement
   iii) Submits the SQL statement to the backend of the database

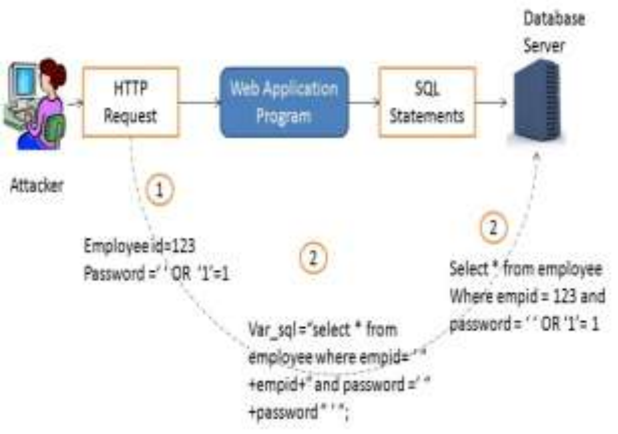

Database Server
HTTP Request
Web Application Program
SQL Statements
Attacker
1
Employee id=123
Password =' ' OR '1'=1
2
Var_sql ="select * from employee where empid=' " +empid+" and password =' " +password " ' ";
2
Select * from employee Where empid = 123 and password = ' ' OR '1'= 1

Fig1. SQL Injection Process

The above SQL statement is always true because of the Boolean tautology we appended (OR 1=1) so, we will access to the web application as an administrator without knowing the right password.

b) The attacker might attempts to modify the present SQL statement by adding elements to the where clause. For example by a simple search application. This application takes the roll number of student as input and provides the search result as output. In this case the web application may run the following query
   Select * from student where roll_number =' <input from the user>'

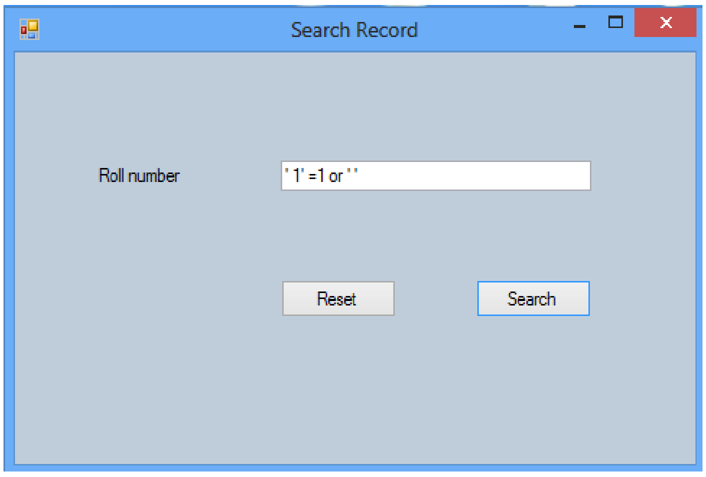
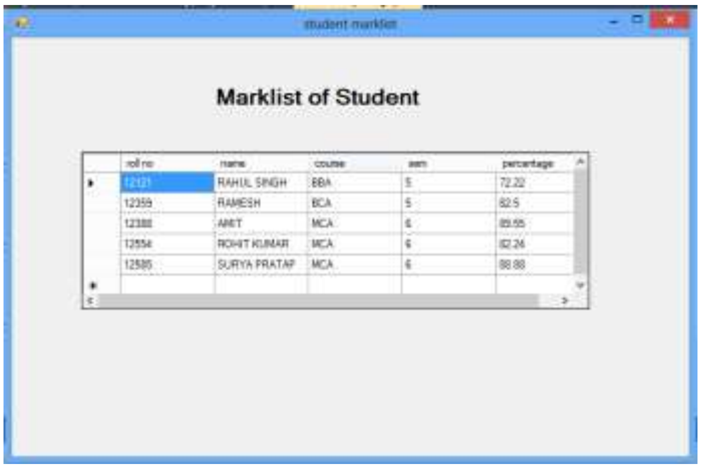


Fig 2. Taking input from the user

Select * from student where roll_number =’1’ = 1 or ‘ ‘



Marklist of Student

roll no	name	course	sem	percentage
12121	RAHUL SINGH	BBA	5	72.22
12359	RAMESH	BCA	5	82.5
12388	AMIT	MCA	6	89.55
12554	ROHIT KUMAR	MCA	6	82.24
12585	SURYA PRATAP	MCA	6	88.88

Fig 3. Result of the Malicious Input

The WHERE clause becomes true for every row and as a result it fetches all entries of the database, in this way the attacker gains access to the application.

6. Proposed Methodology

To overcome from the problems of SQLIA we propose the following techniques. These are the proactive methods of prevention of SQL Injection attack.

a) By use of SQL Query Monitor – we propose the use of SQL query monitor as a tool to analyses all the queries passed to the database server by HTTP request. The SQL Query Monitor Works as bridge between the User application and Database Server. It uses the statistical database log to validate the request generated by HTTP request and allows only if the SQL statements are non malicious. This architecture is explained in the figure 4.

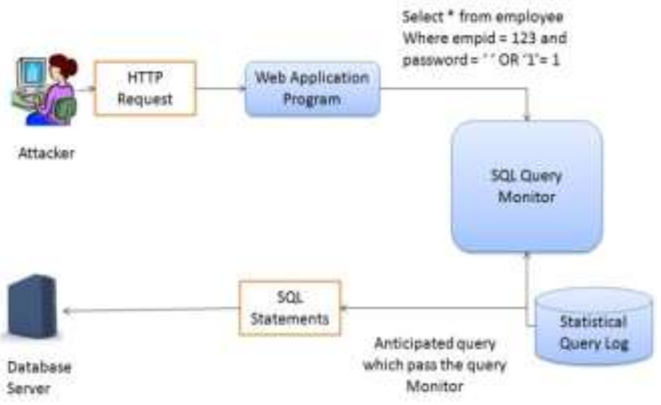


Fig 4. Architecture of prevention from SQLIA by using SQM

In this method when a user generates a HTTP request using web application program it is passed to the SQL Query monitor. The SQL Query monitor then analyses the input query by the use of statistical Query log which contain the statistical data about the allowable values. if the input query is matches to that stored in the Query log then the SQL query monitor grant the permission otherwise it denies the log in process as invalid assess. This process is explained by the uses of following flow chart.

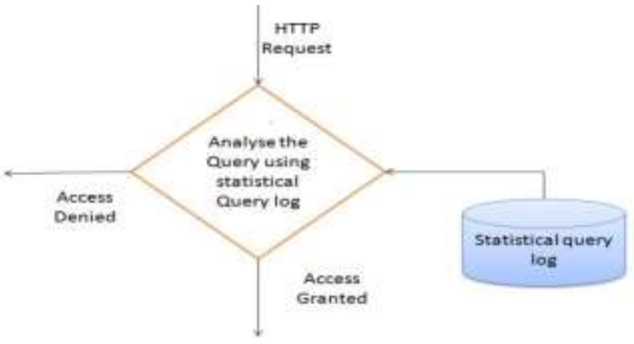


Fig 5. Flow chart of prevention from SQLIA by using SQM

b) Taking user input from predefined choices – in this method we create the application program which enables the user to choose only predefined input choices which is available to them, in this way the user cannot pass the malicious inputs. to implement this approach the search page of the student mark list of the fig2. Can be modified as shown in the fig 6.

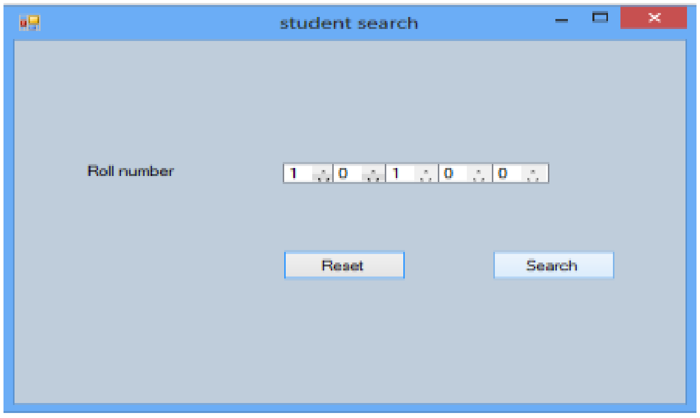


Fig 6. Taking Input from the user using predefined choices

In this way the user now only pass the values which are predefined andd can not pass the malicious inputs

7. Conclusion

The SQLIA is the most vulnerable security threat in the web based database environment in recent years because every attacker or hacker tries to break the database security using such types of attacks. So the proactive mechanism of protection from SQLIA is more acute. Various solutions are given by different researcher but none of the solutions is fully able to prevent the database from these attacks.

So this paper explains the nature an injection process of SQLIA. it also explains the possible cases in which the SQLIA can be done. To protect the system from these attack this paper proposes the use of SQM (SQL Query Monitor) as the validation mechanism by which we can only allowed the predefined inputs. There are various other ways to attack on the database using SQLIA that need to be covered so there is a much scope in this area of research.

References

[1] Y. Huang, S. Huang, T. Lin, and C. Tsai. A Testing Framework for Web Application Security Assessment. *Journal of Computer Networks*, Volume: 48 Issue: 5, Pp:739-761, 2005.

[2] Marco Cova, Davide Balzarotti. *Swaddler:* An Approach for the Anomaly-based Detection of State Violations in Web Applications. *Recent Advances in Intrusion Detection, Proceedings*, Volume: 4637 Pages: 63-86 Published: 2007.

[3] Roichman, A., Gudes, E.: Fine-grained Access Control toWeb Databases. In: Proc. of 12th SACMAT Symposium, France (2007)

[4] Y. Huang, F. Yu, C. Hang, C. H. Tsai, D. T. Lee, and S. Y. Kuo. Securing Web Application Code by Static Analysis and Runtime Protection. In *Proceedings of the 12th International World Wide Web Conference (WWW 04),* May 2004.

[5] Shaukat Ali, Azhar Rauf, and Huma Javed ―SQLIPA:An authentication mechanism Against SQL Injection

[6] K. Amirtahmasebi, S. R. Jalalinia, S. Khadem, "A survey of SQLinjection defence mechanisms," Proc. Of ICITST 2009, vol., no., pp.1-8, 9-12 Nov. 2009

[7] Advanced SQL Injection in SQL Server Applications An NGSSoftware Insight Security Research (NISR) Publication ©2002 Next Generation Security Software Ltd

[8] W. G. Halfond and A. Orso. Combining Static Analysis and RuntimeMonitoring to Counter SQL-Injection Attacks.2005

[9] Vulnerability Management in Web Applications R. Thenmozhi, M. Priyadharshini, V. VidhyaLakshmi, K. Abirami http://www.ciitresearch.org/dl/index.php/dmke/article/view/DMKE042013007

[10] David Litchfield: Web Application Disassembly with ODBC Error Messages .

[11] Martin, B. Livshits, and M. S. Lam. Finding Application Errorsand Security Flaws Using PQL: A Program Query Language.

[12] S. McDonald. SQL Injection: Modes of Attack, Defence, and Why It Matters

Author Profiles

Surya Pratap Singh is MCA and UGC-NET qualified and pursuing Ph.D. In the department of Computer Science DDU Gorakhpur University, Gorakhpur (U.P. India) under the supervision of Dr. U.N. Tripathi. The area of research interest is Database Security, Networking. Mr. Surya Pratap Singh has published 11 papers in different national and international conferences/ Journals.

Avinash Singh is M.Sc. Computer Science, M.Tech and M. Phil and pursuing Ph.D. in the department of Computer Science DDU Gorakhpur University, Gorakhpur (U.P. India) under the supervision of Dr. U.N. Tripathi. The area of research interest is Database Security, Networking. Mr. Avinash Singh has published 21 papers in different national and international conferences/ Journals.

Dr. Upendra Nath Tripathi is Assistant professor in Department of computer science DDU Gorakhpur University, Gorakhpur (U.P. India). He has 13 years of teaching and research experience. He has published 45 papers in various National and International Journals/conferences. His area of research interest is database systems, networking.

Dr. Manish Mishra is Assistant professor in Department of Electronics DDU Gorakhpur University, Gorakhpur (U.P. India). He has 13 years of teaching and research experience. He has published 50 papers in various National and International Journals/conferences. His area of research interest is Computer Technology, fast processor design.