IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

Methodology to Shielding Against SQL Injection Using Internet Protocol Address

Sanjay Mishra[1], Subodh Mishra[2], Vivek Sharma[3]

Student, CSE, TIT College, Bhopal, India[1]

Professor, CSE, TIT College, Bhopal, India[2]

Head of Department, CSE, TIT College, Bhopal, India[3]

Abstract: SQL Injection Attacks (SQLIAs) is a technique through which an unauthorized user can access over database by inserting malicious SQL query segment. The major caused of SQLIAs is improper coding and improper validation of user input. The integrity, confidentiality and availability of web applications are infected by these types of attacks. Now-a-days online services play an important role in our day-to-day life such as email, e-banking, ecommerce, social networking sites, forum etc. However vulnerabilities in these applications may create a wide range of risk as these all contains confidential data such as personal information, banking details and many mores. In this paper we will discuss different types of SQLIAs technique and an algorithm for their preventions against those attacks. This algorithm defeat SQL Injection at different level and protect database to reveal any confidential data from database server when any illegal query is injected for compromising the security. The algorithm using hexadecimal and ASCII value for preventing SQLIAs and a fixed error message is set for protecting database to reveal any valuable information in form of error message.

Keywords: Vulnerability, Structure Query Language Injection Attacks (SQLIAs), Web Application, Hexadecimal, ASCII value, Internet Protocol (IP) Address.

I. INTRODUCTION

Web application plays very vital role in our day-to-day life as it make life efficient and smooth. As use of internet is increasing tremendously in many simple things like shopping is change into e-commerce, banking is changed into e-banking, social life is changed into social networking, and communication via email is very common. With the increasing usages of internet, it is very challenging to secure all the data of user from an unauthorized users and attackers. Web application stores data of users in database and only returns the relevant information when it's receives the request from user after validating the same. But the attack exploits the database server by exploiting its vulnerabilities by various methods.

SQL Injection is one of the most prevalent threats to these web applications from a decade and Open Web Application Security Protection(OWASP) 2013 report these attacks as number one attack [1]. In the report it is shown that the attacks is most damaging factor caused due to illegal query. Vulnerabilities found for these types of attacks using SQL Injection is more than 95% which compromises with integrity, availability and confidentiality of database server. SQLIAs are used by attacker to not only bypass the authentication but it also exploits the data by modification in it. Sometimes it is use to exploit the flows of websites, upgrading or degrading the privileges of any users over web application and for shutting down the database too. There are number of methods and algorithm was given for preventing these attacks but it is unfortunate that they still exits. It is matter of concern to protect it so we have proposed some solution for preventing database server from SQLIAs efficiently.

II. APPROACHOF SQL INJECTION ATTACKS

SQL Injection are using security breaches in any web application for exploiting the database server but there is no any standard definition of these attacks which are defined. Only it is defined as, when illegal SQL code is injected in access which compromises the integrity, availability, confidentiality, of any database server, its termed as SQLIAs [2]. An attacker generally injects an illegal query in form of URL's and cookies. After validating the query the database server gives some error message and sometime it responses differently for attacks which depends on the type and version of database server the web application using. An attacker can also identify the database server by response pattern. In 2004 Microsoft China Technology Center defined SQLIAs in form of two aspects [3].

A. Script Injection Attacks.
B. Malicious user input to affect the implantation of the SQL script.

Whenever the illegal code is injected to affect the implementation of the SQL script, the database server behaves differently and sometimes results in form of error message which is treated as information that allow them to know the database schema information such as table name, column names, data types, and sometimes data value [4]. Many of the times database server tables and field name can be also got by system tables "sysobjects". And when SQL Injection is correct in syntax, attacker can alter the database and also not be detected easily [5]. The attacks are implemented after finding the SQL Injection vulnerability and judging the type and version of database.

III. TYPE OF SQL INJECTION ATTACKS

SQL Injection are implemented because of several reasons such as misuse of delimiters in query string, data types are not distinguish explicitly, flaws in types specification, undefined users input, no check or filtered before execution of databases. For preventing SQL Injection attacks we need to know its type the method through which it can be implemented. SQL Injection are characterised as:

A. Tautology

In this attack the query appends the conditional statements "WHERE" clause as always true query and it produce legal dynamic query that puts all condition true and return all records.

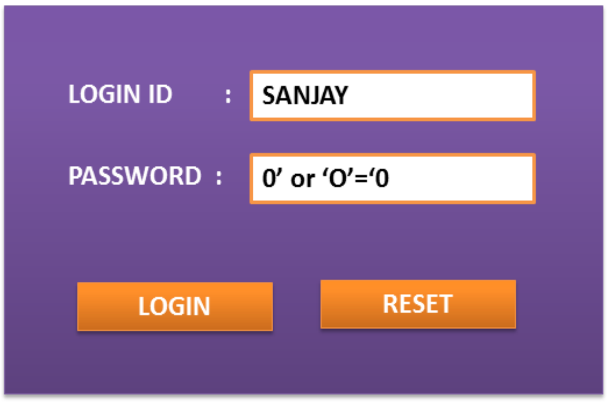

Figure 1: Tautology Attack

Injected Query: select account from USERS where LOGINID= "SANJAY" and PASSWORD =0" or "0"="0. In above case the Login Id is "SANJAY" and Password is 0" or "0"="0. The statement transforms entire WHERE clause into tautology i.e. always true.

B. Illegal/Logically Incorrect Query

This attack is treated as information gathering phase of attacks as when any illegal query is injected, a logical error is produce by the database server reveals valuable information such as server types and it's version, syntax error, type mismatch and other inject able parameter information.

Original Query: http://www.demo.com/users.php?id=109
Injected Query: http://www.demo.com/users.php?id=109'
From the error return by the database server, the attacker knows the type and version of database and sometime the SLQ query pattern for injection.

C. Union Query

In this attack the injected query is concatenated with original SQL query using the keyword UNION to return the dataset by the server when structure of database such as table name and column name are known.

Injected Query: http://www.demo.com/user.php?id=109' union all select 1,2,3- -
By using this desired point of injection is known for extracting the data inside the database.

D. End of comments

By injecting this attack all the query after these symbol "--" are treated as a comment and desired result output comes.

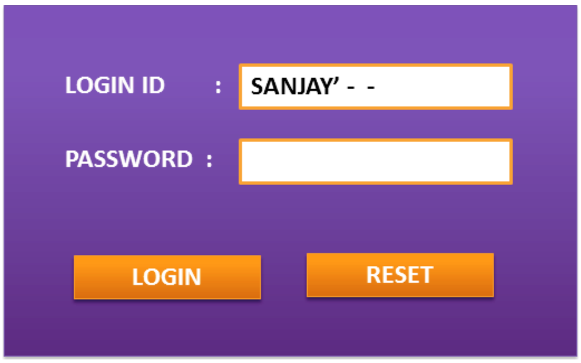

Figure 2: End of Comment Attack.

Here, if only Login Id is known to attackers then they can use this technique to validate the users.

E. Stored Procedure

Attacker uses this type of attack to execute the remote command and for performing privilege escalation. Stored procedure uses special scripting language but sometimes misuse of delimiters gives privileges to the attackers to attack denial of service attack, buffer overflow attack and also leads to run the arbitrary code on server to escalate the privileges of database server [6].

F. Inferences

When database server do not gives any error because web application is configured by error handling method. In this case the attackers observe the response of web application such as the time taken to load the page or response of functions. There are two types of attacks categorised as Inferences attacks.
Blind injection Attacks.
Timing Attacks.

Blind Injection Attacks – In this attack, the attackers inject the query by which server generate an error message as the query is invalid and the error message itself contain database structure. And after gaining information in form of error message the attacker will attempt to reverse engineering.

Timing Attacks – In this attack, the attacker inject code in manner that they are asking from database in a way that in particular condition is true and valid then perform action like delay in replay page loading as if database version contains number like 28 (like 28) have a 20 second delay before replay and load the page [7].

Injected Query for MySQL Server and Oracle: http://www.demo.com/users.php?id=109 and If (version () like "28", sleep (20), "false") - -

Injected Query for Microsoft MySQL Server: http://www.demo.com/users.php?id=109 and If (version () like "28", WAIT FOR (20), "false") - -.

G. Piggy Backed Query
This attack is only implemented if database configured in way that its gives permission to attackers that they can run multiple queries at the same line. This attack includes new distinct queries without any modification in original query. Injected Query: select account from USERS where LOGINID= “SANJAY” and PASSWORD= “password”; drop table USERS - - and PIN= “4321” - -.

H. Alternate Encoding
The database server configured with Intrusion Detection System (IDS) or any signature based filter check system then the attacker’s uses alternate encoding technique to bypass the same systems. This encoding technique is also uses to bypass such systems by modifying text string into Unicode, ASCII, Hexadecimal, Base64 etc. and if require it is used in conjunction with some other methods to penetrate the desired system or to escape from any of the detection tools and methods.

Example: For shutting down of any database server the alternate encoding in ASCII is
exe(Char (115, 104, 117, 116, 100, 111, 119, 110))- -
It is ASCII form of shutdown and if in database it’s defined as SHUTDOWN then, the encoding is
exe(Char (083, 072, 085, 084, 068, 079, 087, 078))- -

Other Encoding forms of “SHUTDOWN” for bypassing the detection system are:
Hexadecimal Encoding of “SHUTDOWN” is 53485554444f574e.
Unicode Encoding of “SHUTDOWN” is \u0053 \u0048 \u0055 \u0054 \u0044 \u004f \u0057 \u004e.
Decimal Encoding of “SHUTDOWN” is 00083 00072 00085 00084 00068 00079 00087 00078.
Base64 Encoding of “SHUTDOWN” is U0hVVERP04.

IV. RELATED WORK
After carefully study about SQL Injection we came at decision that various prevention methodology comprises of method such as input validation, secure coding techniques, tainting approach are able to protect SQL Injection. But still SQLIAs exist as improving coding flows, using parameterized query, tainting approach cannot prevent SQLIAs. An approach to prevent against SQL Injection was given in an algorithm [8]. We will take a part of the approach because the proposed technique was implemented in three phase where in order to minimizing the storing space it save ASCII of both username and password in a same attributes separated by a special symbol say comma(,). But it leaves some improper coding practice and no any method is proposed for preventing attack when number of times attacks is performed by same individual. In proposed method using of comma (,) as a special symbol for splitting the strings make it complex and irrelevant as security point of view. Because now a day a username and password are using special character or symbol such as curly bracket {}, round bracket (), square bracket {}, hash #, colon :, semicolon :, caret ^, comma ,, fullstop ., question mark ?, exclamation ! mark, bar or pipe |, ampersand &, underscore _, back tick „ at @, dollar $, per cent %, slash /, backslash\, arithmetic symbols + - * =, single quote ’, double quotes ” for making the systemsecure. University of Sussex, United Nation has given these facilities for their users [9]. There are number of attributes in database so if we use comma (,) as a special symbol for separation of string cause complexity in database system.

USERID	PASSWORD	ASCII for ARRAY SEPERATED by COMMA (,)
SANJAY	MISHR@	83 65 78 74 65 89, 77 73 83 72 82 64
SUBOD H	PROF@TIT	83 85 66 79 68 72 95 83 73 82, 80 82 79 70 64 84 73 84
VIVEK	HOD@CS	86 73 86 69 75 95 83 73 82, 72 79 68 67 83 84 73 84
TIT_CS	BHOPAL	84 73 84 95 67 83, 66 72 79 80 65 76

Figure 3:Database in which ASCII value of both username and password in single attribute.

The injection is able to protect from all type of injected query such as timing attack and alternate encoding attacks. It is not able to protect from the injection such as Inferences. So prevention from other threat is difficult and security is also major concern for these days.

V. PROPOSED SOLUTION
In proposed methodology we will discuss about the prevention of SQLIAs at various level. The proposed methodology will be implemented at two levels.

Coding Level
While coding the developer have to limit the length of user’s information according to requirement so that SQL injection is not being performed and also convert the input value into ASCII value. Now each ASCII value will multiply by its position number in token which it was divided as single entity. After multiplying ASCII value of each literal in token by its position and sum the value then save it in the database.

Example: select account from USERS where LOGINID= “SANJAY” and PASSWORD= “654321”;
Here, SANJAY is change into ASCII S=83, A=65, N=78, J=74, A=65, Y=89 and then after multiply the ASCII with its position number in token as SANJAY=83*1+65*2+78*3+74*4+65*5+89*6=1602.

Algorithm for storing input value
Step 1: Declare an array, int type and say [n];
Step 2: Store the entered value into string, say str;
Step 3: Convert the string into character type array say ch[20];
Step 4: Store all ASCII value of array ch in array n at respective index after multiplying by its position number in token.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 3, March 2016

Step 5: From a string using the value of the array n, each value is separated using space.
Step 6: Store the string in database.

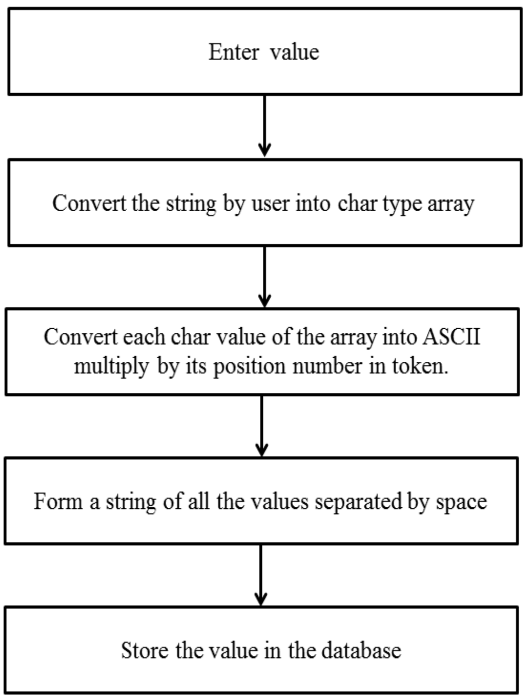
Enter value
Convert the string by user into char type array
Convert each char value of the array into ASCII multiply by its position number in token.
Form a string of all the values separated by space
Store the value in the database

Figure 4: Architecture of value storing method in database.

Algorithm for User Authentication:

Step 1: Store the value entered by the user into the input field into a string say s;
Step 2: Convert the string s into char type array as s1 [20];

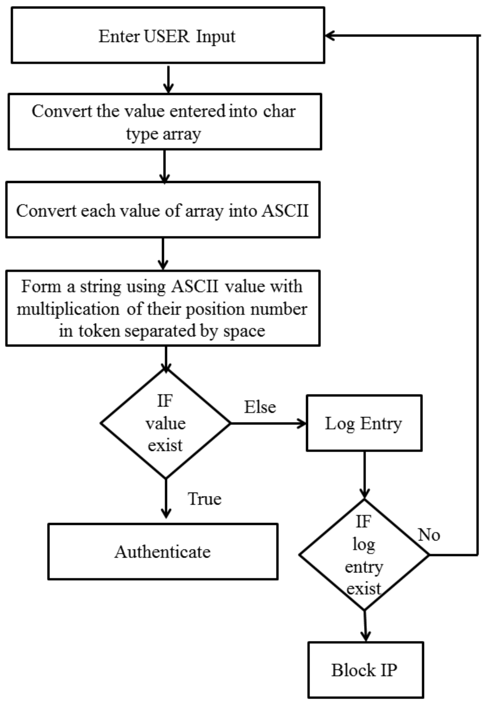
Enter USER Input
Convert the value entered into char type array
Convert each value of array into ASCII
Form a string using ASCII value with multiplication of their position number in token separated by space
IF value exist
Else
Log Entry
True
Authenticate
IF log entry exist
No
Block IP

Figure 5: Architecture for user authentication.

Step 3: Convert each value of array s1 into ASCII code and store them in an int type array say n1 [];

Step 4: Form a string using the value of array s1 in array n1 after multiplying by its position number in token. Each value is separated by space.
Step 5: Compare the entered value in database, if matched then authenticate and go to step 8 else save the log entity in log database.
Step 6: If log entities are matched go to next else go to step 1;
Step 7: Block the IP and display the error message "HTTP-500 Internal Server Error".
Step 8: Display the desired string and authenticate the user.

At Server Level

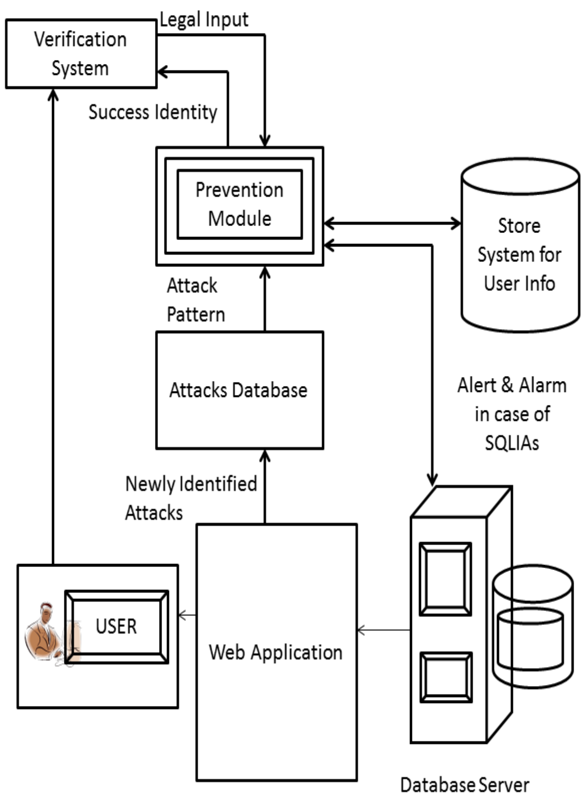

Figure 6: Architecture of proposed model.

Database server will be configured with Intrusion Detection System (IDS) and attack database which identified the newly attacks pattern and if the same user trying to inject illegal query block the user by analyzing the error message with the help of Store System for User Information and save Internet Protocol address in the log file.

As the proposed methodology protect the web application at two levels, it gives better prevention than other model that attacker even bypass then also the secure method will prevent web application by SQLIAs.

Administrator of web application must be ensure that the running account is having minimal privileges [10], and if any alternation is required in database it is only performed by SQL server login and that connection login must be encrypted.

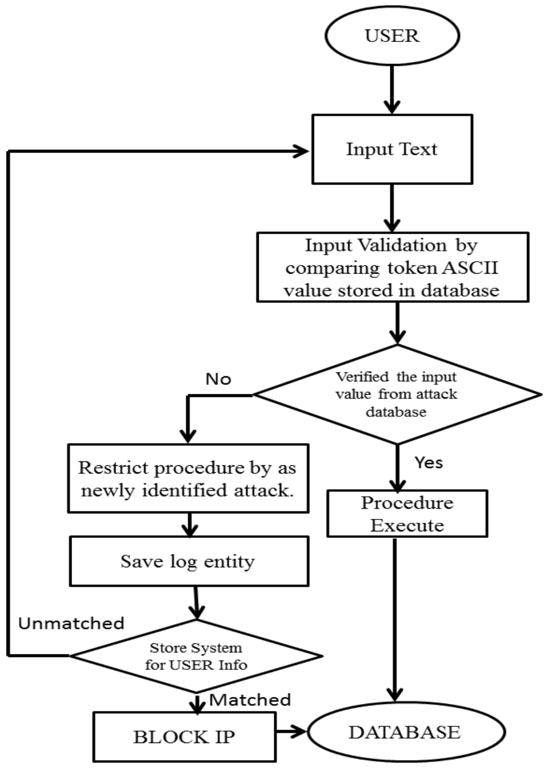
USER
Input Text
Input Validation by comparing token ASCII value stored in database
Verified the input value from attack database
No
Yes
Restrict procedure by as newly identified attack.
Procedure Execute
Save log entity
Unmatched
Store System for USER Info
Matched
BLOCK IP
DATABASE

Figure 7: Flow chart of proposed model.

If any illegal query is injected, database server only returns an error message "HTTP-500 Internal Server Error" in each case of illegal attacks without evaluating the type of attacks.

LOG ENTITY

DATE_TIME	INPUT	STATUS	PAGE_ID	IP ADRESS
11-03-2016, 10:13	Username=1' or 1- -	Invalid	Users.php?id =109	120.11.3.7
11-03-2016, 10:13	Username=0' or 0'= '0	Invalid	Users.php?id =109	110.18.5.1
11-03-2016, 10:13	Username= "admin"	Invalid	Users.php?id =109	100.1.1.1
11-03-2016, 10:13	Username="S ANJAY"	Valid	Users.php?id =109	158.25.4.5

Figure 8: Table for log entry in database.

VI. RESULT AND CONCLUSION

The proposed algorithm is able to handle all SQLIAs. According to result of proposed methodology it effectively and efficiently prevents SQL Injection attacks. For experiment we use ASP.Net base science and MIS framework. This is better solution for SQLIAs as it helps in detection and prevention of attacks from different types of malicious query attacks. This methodology works on any type of server and its version and imposed no any restriction on user. If illegal query is applied two consecutive times using same IP then system block the user and save the log details in database for future prevention.

VII. FUTURE WORK

This methodology protects all SQL Injection by analyzing the injected query pattern and also by error message analyzing. And then blocks the IP address whenever illegal query will be applied using same IP. But there is room for improvement to protect it when attacker uses Virtual Private Network (VPN) having property to switching their IP at regular interval.

REFERENCES

[1]. OWASP Top Ten Project [EB/OL]. 2016-03-10]. https://www.owasp.org/index.php/Category:OWASP_Top_ten_project.
[2]. DebabrataKar; SuvasiniPanigrahi; "Prevention of SQL Injection Attack using Query Transformation and Hashing", 3rd IEEE International Advance Computing Conference (IACC). Pp.1317-1323, 2013.
[3]. Microsoft China Technology Center, SQL Server Safety Review. [URL]http://www.microsoft.com/china/etc/Newsletter104/ctc2-html; 2004.
[4]. Sanjay Mishra; Subodh Mishra; Vivek Sharma; "SQL Injection Prevention by Blocking Internet Protocol Address after Analysing Error Message of Database Server." (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (2) , 2016, 566-569, 2016.
[5]. Xiaobing Chen, hanyu Zhang, liming Luo, He Huang, "SQL Injection Attack and to Prevent Detection Technology (J). Enginnering and Application of Computer, 2007, 43(11).
[6]. E.M Fayo, "Advance SQL Injection in Oracle Database", Technical report, Argeniss Information Security, Black hat USA, 2005.
[7]. A.Sadeghian, M.Zamani, ShahidanM.Abdullab, "A taxonomy of SQL Injection Attacks", DOI 10.1109//CICM.2013.53; IEEE 2013.
[8]. MahimaSrivastava, "Algorithm to Prevent Back End Database Against SQL Injection". 978-93-80544-12-0/14; IEEE 2014.
[9]. My IT Account, [url]http://www.sussex.ac.uk [2016-03-15].
[10]. Microsoft China MSDN, Application Security Solutions in SQL Server. [url]http;//msdn.microsoft.com/zh-cn/library/bb669057.

BIOGRAPHY

Sanjay Mishra received the B.E degree in Computer Science and Engineering from Technocrats Institute of Technology, RGPV Bhopal, Madhya Pradesh, India in 2013. He is a Research Assistant as pursuing M.Tech from RGPV University. His research interests are SQL Injection Attacks, Computer Networks, and Vulnerability Assessments etc.