ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

Sql Injection Attack Prevention Methodology

[1] S.Deepica, [2]T.Edison, [3]Dr.R.Viswanathan
[1][2] Assistant Professor, [3]Associate Professor
[1][2][3] Galgotias University

Abstract: Structured query language injection is a process through which one can intrude the database of a software application (mostly in web applications) through the application's user interface. By adding or editing strings to the SQL query, injection is performed, thereby hacking sensitive user details. The hackers can even create humongous problems by dropping the entire data in the database. The hacker deeds all the security susceptibilities in the injection. When the user input is not captured sturdily and executed unexpectedly, injection could happen.

Key words-Vulnerability, Injection Attack, Secret Key, Private Key.

INTRODUCTION

In the current working scenarios of information technology, any organization is compelled to give more priority to the data security to protect their data from illegal access from the hackers. For any organization, this is the foremost challenge. The companies are even involved in creation of dedicated sections with ethical hackers for their firms; By doing so they will be able to get to know about the vulnerable areas in their application's security. Organizations ranging from startups to Multi-national companies are actively involved in this process. There are even dedicated organizations which provide ethical hacking as a service to other firms. Improper user input data validation is a key security vulnerability issue, if the hacker discovers the availability of minimal security constraints about the format, range and the length of data.

The hacker can trigger a malevolent input using wild cards and single quotes to conciliate such vulnerable application's security. Such attacks are possible only in the public interface. A SQL injection attack is usually referred to as SQLIAs. Apart from structured query language injection, other hacking techniques which are available are Cross site scripting, Denial of service, Buffer overflow, Key logger, Denial of Service (DoS\DDoS), Waterhole attacks, Fake WAP, Eavesdropping (Passive Attacks), Phishing, Virus, Trojan, Click-Jacking Attacks. The rage of their usage is provided in the following figure. From the figure we understand that Cross site scripting is the most widely used hacking technique used by the attackers. But details about Cross site scripting is not explained in this section as it is out of scope of the problem that has been discussed in this paper.

Techniques used by attackers

In SQLIAs, the hackers usually access the input strings that are specially encoded database instructions. During the execution of web application, when the query which contains the encoded database commands are sent to the database, the attack gets succeeded because the embedded commands which are contained with the syntactical elements of SQL, gets executed by the database. The awful consequences of this attack could lead to compromising the sensitive client data, leakage of top business secrets or even annihilation of the entire database contents. SQLIAs have become a very serious security issue due to the availability of unrestricted access to the application's database

2. PREVENTIVE METHODS

To prevent SQL injection for Web Application following Methods are there- Syntax Aware Evaluation

•Do not trust whatever the user enters. Strict type checking is vital.

•If you expect user name to be entered, then check validation whether it contains only alpha-numerical characters.

•Set length limits on any form fields on your site and most importantly don't use real column names which would be a blunder.

•Use pre-prepared statements for the execution of the queries.

•Must not allow multiple queries for execution in a single statement i.e. joint queries shouldn't be permitted.

• For Example: - SELECT * FROM story_author_table WHERE sID=999 AND a. aID=s.aID UNION ALL SELECT 1,2,3,4,5,6, name FROM sysObjects WHERE xtype='U'] This type or multi joint queries should be restricted.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

•Don't leak the database information to the end user by displaying the "syntax errors" only display check box errors etc. This would lead to a way through
which the hackers could an insight about the underlying database details.

Distinct Input Techniques:
This technique includes some restriction to the given input method for example special characters, case sensitive, maximum number of characters (50), minimum number of characters (8) etc.
•Allow only unique and difficult words to guess table and column names. Using simple names for the table and columns make the application less secure
•Use aliases in the query. This will prevent the explicit display of columns used in the queries.
•Escape or filter the special characters and user inputs rise error check box error if any found,
•Validate all your data on the server side at a minimum for content, length and format. This could also be done in addition to the implemented client page level validation if any.

SQL signature
We found that for every web application there must be minimum privileges to access that data.

Inputs Validation
Client system level Ajax validations should also be used in addition to the server level validation. The end-user should always be provided with the access as per his/her role. Providing a super-user access to the normal user of an organization increases the risk of exposing the application to security vulnerabilities. Restricted access can be implemented by creating separated system roles which is the duty of a system security analyst or consultant. System administrator access should be provided to the security analysts who actually take care of the role assignments to the other members of the organization.

Underlying techniques Encryption technique for Prevention of SQL injection:
Encryption technique can be deployed for the prevention of SQL injection. This method proposes Advance Encryption Standard (AES) for preventing structured query language injection Attacks. These techniques are based on secret key and private key. Secret key is used by user always and private key is used by the server. These keys are always unique for every user.

In this method we apply two levels of encryption to the login Query by making use of the secret key and private key-
A. Symmetric key encryption for user name and password by using user's secret key.
B. Asymmetric key encryption for query by using server's private key.
Below are the various phases and their corresponding activities which are performed by means of secret key and private key which are involved for registration/login process:

Registration phase:
In this phase following steps are included-
•Whenever a new user wants to join the server, then user must select unique User Name and Password along with the registration request.
•Server receive user request, then generate unique secret key for that particular user. Server maintains a table which contains columns as username and password and unique key as shown below in table-1.
•Then server will send the registration confirmation along with the secret key.

Login phase:
In this phase user will be able to login to the web page by providing the login credentials created during the process of registration. In other words, user can now access the data base, which include following steps-
•User name and password which is entered by user will be encrypted by AES Advanced Encryption Standard by applying user secret key.
•Then SQL query generator generates the query by using the inputs attached with secret key.
•Then query result is encrypted using RSA cryptosystem which added private key and send that encrypted key to the server.
For example Query result= SELECT * FROM user account WHERE username = 'abc' AND password = 'xyz' AND encrypted username = 'ESecretKey (abc)' AND encrypted password = 'ESecretKey (xyz)'
In the query, you can observe that in addition to usage of username and password, encrypted username and encrypted password is also used in the WHERE condition thereby preventing the execution of any embedded database instruction sent by the attackers through the user interface of the web application.

Verification phase:
It is the phase which includes the final and most important process which happens on the server. The following steps are involved in the verification phase:

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

•When the server receives the login query in encrypted form, it tries to decrypt it. Decryption of the query will be performed using the Server private key.
•It then verifies the corresponding query, password and also the user's secret key.
•Check the decrypted user name and password from the user account table.
•If matches, then accept the user login request.

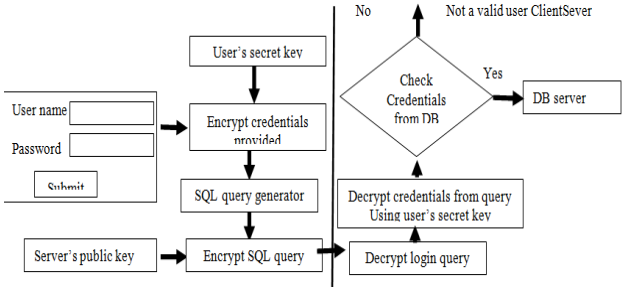


Figure 1.How to Apply AMNESIA

The Prevention of SQLIA's/;

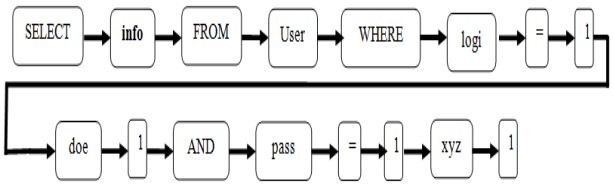
In this paper we discuss about the technique called AMNESIA; The ways to make use of it, and the limitation of this technique is discussed:

Introduction

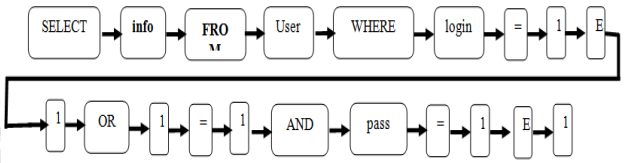
AMNESIA (Analysis and Monitoring for Neutralizing SQL-Injection Attacks), is a mechanism, used to prevent SQL injection attack by checking each query with the pre-defined model of that query developed by the administrator. In this approach the key thing is that it uses both static analysis and dynamic monitoring, which can be explained through following points: -
•It generates a static model of every query used by the web-based application.
•When the user fetches any data on the website than it checks the dynamically created query with the static model
•If the query violets the static model generated in prior, it classifies it into a structured query language injection attack.
•Identify Hotspot:- According to this we have to find out what are the places where in website any database query is executing, and then we have to give an unique Id no. to that spot. We have to ensure that none of the database queries are missed since such misses could have other side-effects during the model construction.
•Construction of Models: - For every Hotspot the administrator should develop a static model which can be cross checked. The model will be a NDFA (Non-Deterministic Finite Automata). The Non-Deterministic Finite Automata will have the transition labels as placeholders, tokens, limiters. For eg- Figure 1.
•Inserting a monitoring Function call:- In this we will insert a Function call for each and every hotspot we defined. This function will be having 2 arguments, hotspot id and the query.

Secure the hotspot by this function

•Execution of Query:- Our application will run as usual until any of the hotspot comes. When the hotspot arrives it will divide the query in blocks as given figure 2, and then compare it with the static model. If both are same then it will execute the query else it will term it as a SQLIA. For eg:- In figure 3 the query (a) will execute and (b) will be rejected.



(a).SELECT info FROM user WERE login = 'doe' AND pass = 'xyz'
Fig 2



(b). SELECT info FROM user WERE login = "OR 1 = 1 ..'AND pass ="
Fig 3

Limitations

According to our mechanism a query is made by arranging strings in an ordered way. According to AMNESIA the developer should create queries by the combination of constant strings and variable with concatenation, appending and insertion. But the mechanism fails when the application that externally stores the query related strings in files, and the mechanism can be violated with good engineering capabilities.

It will also produce a false output if it cannot recognize the keyword as constant string and then it judge it as a SQLIA's. By making use of operations such as insertion, concatenation and appending for combining variables and

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018

strings which are hard-coded queries are created. But it is to be noted that, though this assumption makes it difficult to use AMNESIA, it is not a much restrictive assumption. Using suitable engineering, this assumption related to implementation can be prevented. At times, this technique can create incorrect negatives and positives. In-correct positives results might occur when the analysis of the string is not precise. Incorrect negative results might occur spurious queries are contained in the model generated for SQL query and if the hackers are capable enough to generate the spurious queries. An experimental evaluation of the concept was performed to check the practicalities of the limitations discussed. AMNESIA's performance was excellent in the evaluation; Almost 7 applications were tested for different types of attacks and illegal access. False positive or negative results were not generated.

3. SQL injection prevention using parse tree algorithm

The parsed tree is a data structure for representing our SQL statement. Parsing needs to check the grammar of the statement's language and we can determine that two queries are equal or not, by parsing two statements and comparing their parse trees.

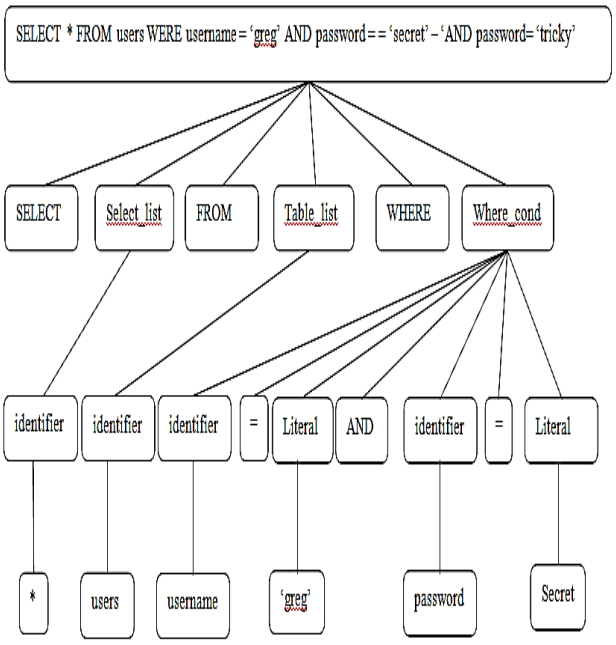


Fig 4

When a hacker tries to inject a SQL query in database query then parse tree of the intended SQL query and the hacker query do not match. The mean is that only when the programmer inserts the query for database the structure of query is predefined. The empty leaf node of the parse tree is used for user's input.

SELECT * FROM users WHERE username=? AND password=? The question marks are place holders for the leaf nodes she requires the user to provide. While many programs tend to be several hundred or thousand lines of code, SQL statements are often quite small. This provides the opportunity to parse a query without adding significant overhead.

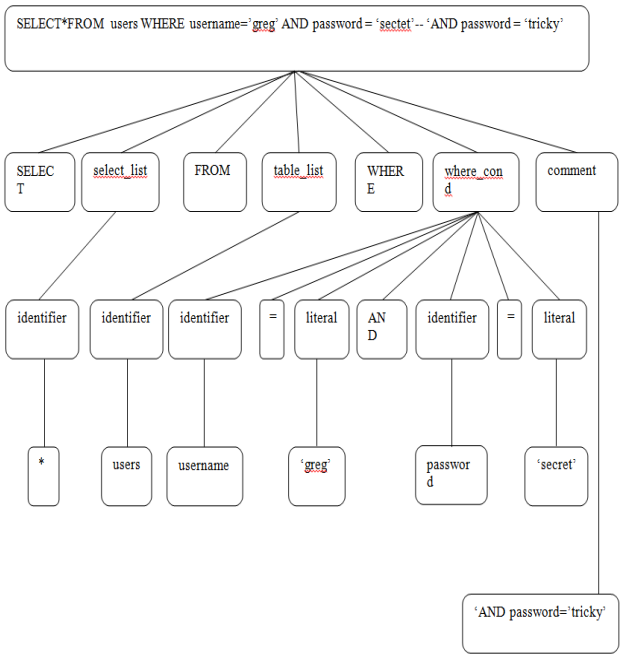
Dynamic queries many web applications which allow the user to shift through tabular results permit sorting the table by any of the columns. Our approach uses the dynamic query concept. As we know the user can give a dynamic input that is he wants to have a output based on a particular condition which he can give using where clause (WHERE subject like '%input%'). So that the query is not statically bound; if we have to search anything on internet, every user has a different search and for this the query is also dynamically created so here we cannot use static query methods for prevention SQL injection. We can also take another example. Let's consider that, we have to display the output in an ordered way. If the user wants the table to be shown in some particular order he has to use ORDERED BY column1 and column2, since here the order is chosen by the user, so it also became a dynamic query and our algorithm can very well satisfy these conditions. One significant advantage over static methods is that we can evaluate the exact structure of the intended SQL query. Many times this structure itself is a function of user input, which does not permit static methods to determine its infrastructure. An example is a search page. One popular free web-based email tool allows users to search through their messages for particular content. This search may or may not include searching through the email body, subject, from field, etc. Typically in these types of searches, if the user leaves one or more of these fields empty, the code does not incorporate them into the SQL query. Normally the programmer will append inputs from these fields on the WHERE portion of the query, such as WHERE subject LIKE '%input%'.

Comment Token Inclusion:

There is a special case of attacks which plays an important role that is the solution is compared with parse tree when in the user supplied fields values have not been inserted and after values are

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 4, April 2018



Figures 1, 2 and 5. The original query is given in Figure 1, with two user-supplied fields (shaded gray). If the user were to supply greg' AND password='secret' { { for the username field and tricky for the password field, the resulting parse tree would be as in Figure 2. The potential vulnerability is that the programmer may assume that since the query parsed properly, the value stored in the request object's password field is the same value which was used to build the query. While the data returned by the query is proper, the state of the program is most likely not the state assumed by the programmer. Fortunately, our parse tree mechanism catches this subtle hazard. The solution is to include the comment as a token in the parse tree. Figure 5 is the result from the same input, with this comment token included. Because the original query does not have a comment token, the resulting parse tree is no longer a match. This does not restrict the programmer from annotating queries. Had a comment existed in the original query, the parse would still have failed because the value (string literal value) of the two tokens would not be equal.

CONCLUSION:

As we come to know that it is not too difficult to prevent SQL injection attacks. By simply following the steps above, we can reduce the
chances of any SQL injection attack against your Web application, if developers are made just little awareness of these points. Its only requires a little bit changes in their coding style to prevent not only SQL injection attacks. In our experience, developer training has been very helpful in increasing their understanding of websites attacks and vulnerabilities. The result normally an improved and maintained security and development teams that leads to more secure websites.

REFERENCES

[1] "Neha Singh and Ravindra Kumar Purwar " SQL Injection –A Hazard To web applications", in International Journal of Advanced Research in computer Science and Software Engineering, vol.2, Issue 6,June 2012, pp. 42-46.

[2] Permulasway Ramasamy and Dr.Sunitha Abburu "SQL Injection attack detection and prevention" in International Journal Of Engineering Science and Technology(IJEST), vol.4, April2012, pp.1396-1401.

[3] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso, "Classification of SQL Injection Attacks and counter measures", ISSSE 2006, March 2006.

[4] San Tsai Sun, Ting Han Wei, Stephen Liu and Sheung Lau, "Classification of SQL Injection Attack" Nov 2007.

[5] Nilesh Khochare, Santosh Kakade and B.B.Meshramm, "Survey on SQL Injection attacks and their Counter measures" IJCEM international Journal of Computational Engineering & Management,ISSN(Online):2230-7893,vol.14, October 2011, 111-114.

[6] William G.J.Halfond and Alessandro Orso, "AMNESIA Analysis and Monitoring for Neutralizing SQL Injection Attacks" November 7-11, 2005.

[7] Atefeh Tajpour, Suhaimi Ibrahim, and Mohammad Sharifi, "Web Application Security by SQL Injection Detection Tools", IJCSI International Journal of Computer Science Issues, vol.9, Issue 2, NO.3, March 2012

[8] Diallo Abdoulaye Kindy and Al-Sakib Khan Pathan, "A Detailed Survey on various Aspects of SQL Injection:Vulnerabilities", Innovative Attacks, and Remedies accepted version for information journal, 2010.

[9] Abhishek Kumar Baranwal, "Approaches to detect SQL Injection and XSS in web applications", IEECE 571b, Term Survey paper, April 2012.

[10] V.Shanmughaneethi and S.Swaminathan ”Detection of SQL Injection Attack in web applications using web services”, IOSR Journal of computer Engineering(IOSRJCE) ISSN:2278-0661, volume 1, Issue 5,May-June 2012, pp.13-20.

[11] Atefeh Tajpour, mohammad JorJor zade and Shooshtari”, Evaluation of SQL Injection Detection and Prevention Techniques”, IJSCE vol 3, march 2015.

[12] Katkar Anjali S and ,Kulkarni Raj B.”Web Vulnerability Detection and Security Mechanism” ,International Journal of Soft Computing and Engineering(IJSCE)ISSn:22312307, volume-2, Issue-4, September2012, pp.237-241.

[13] Anyi Liu , Yi Yuan , Duminda Wijesekera ,and Angelos Stavrou, “SQLProb: A Prloxy-based Architecture towards Preventing SQL Injection Attacks” IJCEM international Journal of Computational Engineering & Management,ISSN(Online):2230-7893,vol.14,October 2013,111-114.

[14] Atefeh Tajpour , Suhaimi Ibrahim, and Mohammad Sharifi, ”Web Application Security by SQL Injection Detection Tools”, IJCSI, International Journal Computer Science Issues,Vol.9, Issue 2,No.3,March 2012,332-339

[15] Stephen W. Boyd, Angelos and D. Keromyti, “SQLrand:Preventing SQL Injection Attacks” oct 2011

[16] Devata R. Anekar and Prof. A. N. Bhute ”SQL Injection Detection and Prevention Mechanism using Positive Tainting and Syntax Aware Evaluation,” International Journal of Advances in Computing and Information Researches, ISSN:2277-4068, Volume 1–No.3,August 2012

[17] William G.J.Halffond and Alessandro Orso”Preventing SQL Injection Attacks UsingAMNESIA”ICSE,2006,Shanghai,China

[18] Etinene Janot and Pavol Zavarsky ”Preventing SQL Injection in onlineapplications,”Study,Recommendations and Java Solution Prototype based on SQL DOM,Application Security Conference,Belgium,19-22 May 2008.