ISSN:0975-9646

Review on SQL Injection Attacks: Detection Techniques and Protection Mechanisms

Sankaran. S , Sitharthan. S , Ramkumar. M

Department of Computer Science Engineering, Saveetha University
Saveetha Nagar, Thandalam Chennai-602105, Tamil Nadu, India

***Abstract*— When a Computer user interacts in the internet through the World Wide Web environment, sending E-Mail, surfing the web and involving in forums and online discussions, lots of data is being created which may have user's personal information. If this data is retrieved by third party tools and techniques, it may cause a break in the user's privacy. Hence, the big question is protecting the web environment against Cyber-Attack. SQL Injection Attacks have been around for over a decade and yet most web applications being deployed today are vulnerable to it. The bottom line is that the web has made it easy for new developers to develop web applications without concerning themselves with the security flaws, and that SQL Injection is thought to be a simple problem with a very simple remedy. For the purpose of security, we have proposed various attack methodologies, and also the testing frameworks and prevention mechanisms.**

***Keywords*— Classification, Detection, Prevention, SQL injection attacks, Web application security.**

I. INTRODUCTION

As per the Open Web Application Security Project (OWASP), the Structured Query Language Injection Attack (SQLIA) is regarded as number one web application vulnerability in the past years. In this modern computer era database has become very essential in any web applications. All the web applications that are being developed has database connectivity in some form, hence making it database dependent. In an average about 71 attempts of attack is performed on an application. Some applications experience about 1300 times in an hour at the peak [1].

A successful SQLIA can

- Read/Extract sensitive or confidential data from database.
- Modify data in database (Insertion/Updating/Deletion).
- Executing operations such as shutdown of the DBMS and other harmful operations.

The insertion or injection of an SQL Query via the input data from the client to the application results in an SQL injection attack. Malicious input statements are given in an SQLIA. This malicious inputs when executed at the database result in unpredicted behavior and thus the security of the database is compromised. The input given by the attacker is executed as it is. The database cannot differentiate between an input given by the clients/attacker and a malicious input [2].

II. TYPES OF SQLIA

In this section the various types of SQL injection attacks are described and discussed in greater detail. Each of the attack types is given a name, one or more intent of attack and attack description in detail, example for the attack and a collection of references that discuss these attacks in a more detailed manner.

Most of these attacks are not performed separately but are combined together to perform some actions depending on the aim of the attacker.

A. Tautologies

Attack Intent: Identifying injectable parameters bypassing authentication, extracting data.

Description: In this type of attack, a code is injected into the input fields of the web application and one or more conditional statements are executed that are always true when executed. This is one of the most widely and commonly used method to bypass authentication and data extraction. If the attack was performed successfully, it returns a set of record, or a result if some action is performed.

Example:

The query for login is:

SELECT * FROM user_info WHERE logID='' or 1=1 -- AND pass1=''

The conditional statement OR (1=1) makes the entire WHERE clause to a tautology. The query evaluated true for all records in the table and returns them. In the above example, there is a value returned and hence a not null value is evaluated, making the authentication process successful allowing the user to login successfully without valid credentials. This method is also employed to extract data from the database.

References: [16, 3]

B. Illegal/Logically Incorrect Queries

Attack Intent: Extraction of data, performing database finger-printing, identify injectable parameters.

Description: When a logically or wrong SQL Query is sent to the database, error messages are sent from the databases which may contain sometimes useful information that can be used for debugging. This information may sometimes be helpful for the attacker in finding vulnerabilities in the system and hence in the database of the application also.

Example-

Select * from <table name> where userId = <id> and password = <wrongPassword> or 1=1;

This is a query for sending an error message for wrong password.

In this type of attack, the attacker gets to know the information of the table such as the name of the table and the field names of the fields in the table, which helps him for a more coordinated attack at the later stage. Credit cards are a classic example for this type of an attack in which the name of the table and details are gathered. Similarly target specific attacks can be performed if the schema of the database is known.
References: [3, 17]

C. Union Query

Attack Intent: Extraction of data, Authentication bypassing.
Description: In this attack type, a dataset is returned that is a result that is an UNION of the original query and the result of the injected query that is inserted into the vulnerable parameter. The UNION query combines two or more queries together and gives the result which gives the fetched rows from the queries participating in the UNION.
Example: An attacker could inject the text "' UNION SELECT cardNo from Credit Cards where acctNo=10032 --" into the login field, which produces the following query:
SELECT accounts FROM users WHERE login='' UNION SELECT cardNo from CreditCards where acctNo=10032 -- AND pass='' AND pin=
The original first query returns a null set since there is no logical equal to "", but the second query returns information from the table. In this example, it returns the column "CardNo" for the account "10032".
References: [4, 5]

D. Piggy-backed Queries

Attack Intent: Extracting data, adding or modifying data, performing denial of service, executing remote commands.
Description: In this type of attack, the statement after the ";" is executed. It is a very dangerous type of attack which could damage the database, sometimes may destroy it also. It could bring large loss of data if it is successfully executed.
Example:
SELECT account FROM user WHERE login='abc' AND pass='; drop table user --' AND pin=123
The above query is generated if the attacker inputs "'; drop table users --" into the *pass* field.
The execution of the query is done, and after the first query the delimiter is read and the second injected query is executed. After the successful execution of the second injected query the table user is dropped, destroying some information that may be valuable. Other queries can be used to insert new values, execute stored procedures, etc.
References: [4, 5]

E. Stored Procedure

Attack Intent: Escalating privileges, denial of service, remote command execution.
Description: The database engine runs the routine that are stored, these are known as Stored Procedures. These stored procedures may be user defined or that are provided by the database by default. The different ways of attacking the database are dependent on the type of stored procedure.
Example:

```
CREATE PROCEDURE DBO.isAuthenticated
@username varchar2, @pass varchar2, @pin int
EXEC("SELECT account FROM user
WHERE login='" +@username+ "' and pass='"
+@password+ "' and pin=" +@pin);  GO
```

In the example, the constructed query string at the line 5 is replaced by a call to the stored procedure. The authentication of the user's credentials is indicated by a true/false value that is returned by the stored procedure. The stored procedure starts executing if the attacker inserts the code "'; SHUTDOWN; --" into the username field or the password field. This generates the query:
SELECT accounts FROM users WHERE login='abc' AND pass=' '; SHUTDOWN; --AND pin=
This is similar to a Piggy backed attack where the statement after ";" is executed resulting in the shutdown of the database. Successful execution of this attack results in huge amount of loss.
References: [4, 18, 6, 5]

F. Blind Injection

Attack Intent: Extraction of data, Theft of data.
Description: The error messages that are displayed by the database are hidden by the developers for security reasons and only the generic error pages are displayed to the user when it is accessed, making difficult for an attacker to get information from the database [19]. It is when an attacker sends true/false questions to steal/theft data.
Example: SELECT name FROM <tablename> WHERE id=<username> and 1 =0 -- AND pass = SELECT name FROM <tablename> WHERE id=<username> and 1 = 1 -- AND pass =
After execution of the queries, error messages are returned. If the web application is secure, there is a chance of injection if the inputs are not validated in advance. After the insertion of first query, if the attacker receives error, he does not know whether it was due to input validation or query formation error. But, the id field is vulnerable if there is no error message after the submission of second query.
References: [4, 6]

G. Inference

Attack Intent: Data extraction, Identify injectable parameter, determine database schema.
Description: In this type of attack, the query is modified to recast it in the form of an action that is executed based on the answer to a true/false question about data values in the database [8]. In this type of injection attack the attacker has difficulty in attacking the site that has been secured so much that there is no detailed error messages that are displayed. Even after a successful injection attack there is no useful feedback or response from the database. Hence, the attacker uses different methods for extracting details from the database. The attacker inserts commands into the site and then observes how the response of the site is. By this method, the attacker comes to know the parameters that are vulnerable to attack and also additional information about the database. Most commonly there are two attacks possible, they are extracting the data, detection of vulnerable parameters.
References: [6, 1]

H. Alternate Encodings

Attack Intent: Vulnerability detection evasion.

Description: In this type, the injection query is modified by alternate encoding, changing characters to some other characters in the queries. By this way, the attacker evades filters for "wrong characters".

All different kinds of SQL injection attack can be hidden using this method.

Example- SELECT name FROM <table_name> WHERE id='' and password=O; exec (char (O x73687574646j776e))

The hexadecimal encoded character are taken as input that is used as char function that returns actual character. This encoded string, shutdowns the database when the command is executed.

References: [6, 1, 3]

III. SQLIA DETECTION AND PREVENTION

There are many methods and tools for detection of SQLIA that have been proposed. Following are some tools invented to detect and prevent the SQL injection attack.

A. JDBC-Checker:

It is a tool that detects and prevents taking into advantage the type mismatch in the queries that are dynamically generated [7].

B. ADMIRE:

ADMIRE is a tool that is used to identify and moderate the effect of SQLIA by thorough and step by step methods [7].

C. SQL-PROB:

This is a detection tool that uses SQL proxy based blocker that fetches input from SQL query of the application and checks with the syntactical structure of the query. It provides protection and security to the frontend web server and the backend database by means of integrating seamlessly with the existing operating environments [8].

D. WAVES:

This is a testing tool that uses a black box testing technique for testing vulnerabilities in web application for SQL injection. This tool finds all possible points and identifies them through which injection can be done. The machine learning is done by which the attacks are build that target these points of vulnerability and also monitors them, how the response of the application is to these attacks [9].

E. SQLRand:

The SQL injection attack of the web server is prevented by this system. Randomized SQL query language is used to detect and abort the injected queries. There is a proxy server that is setup in between the client server and the SQL server. The query is conveyed to the server which is received from the server as de-randomized requests. The proxy parser in the proxy server fails to recognize the randomized query and also reject it, if a successful SQLIA is done [10].

F. POSITIVE TAINTING:

The trusted data is identified and marked in this type. Syntax aware evaluation is performed on the trust marked strings that are tracked. Characters in a string that are not trust marked are not allowed to pass the database [11].

G. AMNESIA:

The SQLIA's are detected and prevented by a means of static and dynamic analysis. The four main steps are:

1. Identify hotspot: The hotspot points are identified in this step that issue the underlying database SQL queries.
2. Building of SQL query model: A model is built for each one of the hotspot representing all the probable queries that may be generated at that hotspot.
3. Instrument Application: Call to each runtime monitor is added at each hotspot in application.
4. Runtime monitoring: It automatically rejects and reports the dynamically generated queries that do match against the SQL query model [12].

H. SQL DOM:

In this detection technique, the API becomes insufficient and cumbersome by creating every per possible operation per column one method and one class table and for one class table one class per table. Unnecessary object duplication can be avoided by statically accessing all the mapping information of the database structure [13].

I. VIPER:

SQL injection attack is detected using a heuristic approach. The generation of SQL queries is guided by the knowledge base of the heuristics. The input forms of the hyperlinks structure is identified at first. Standard SQL attacker are stacked. The output of the web application is matched with library of regular expression that is related to the error messages that a database produces. The attack is continued using the text that is mined from the error messages. It is done with objective of retrieving the schema of the database [14].

J. CANDID:

In CANDID, the actually issued query is compared to the legitimate query structure mined for the same path that is dynamically mined by the program's legitimate query structure at each path by executing the program with the valid and non-attacking inputs [15].

K. Some other Prevention Techniques:

1. Interpretation at web server level

SQL injection attack need to be prevented at the web server itself. It is also one of the most effective ways to prevent SQLIA. An open source web application that can be used is Mod –Security that is installed on the server and alerts the host, whenever a specific keyword is come across [19].

2. Interpretation at language level

The SQLIA's can be prevented my writing source code that is secure, hence the attack becomes difficult to be performed. One of the way is PHP escaping. The most commonly used validation method is data type validation, most of developers do not know it is an in-effective way and can be easily attacked or bypassed.

PHP helps in locating and replacing some specified characters by double quotes (""). The latest version of PHP

should be used for security purpose and care should be given for coding [20].

3. User Privileges

The concept of dividing the admin privileges into admin user accounts rather a superuser that is very dangerous and should be avoided. One admin account should not exceed more than two operations, hence, even if it is compromised there would be not much damage inflicted to the database.

4. Encrypting Data

Encrypting data is another method of prevention. By encrypting the data, the attacker gains access to the encrypted data that is of no use unless decrypted. Sensitive data can be encrypted. The keys of the data that is encrypted can be stored in a separate table or can also be stored in a separate server for high security and can be linked in a manner that is efficient. This is up to the choice of the database admin.

5. Other precautions

The database should be developed in such a manner that only the developer that can be trusted completely knows the full details of the database and only to some extent to the database admin and the staff. Admin should monitor regularly for access by intruders and suspicious activities. The database software should be updated at a regular interval for higher security.

IV. CONCLUSION

In today's modern era, the possibility of an SQL injection attack at the web applications are high. The attacker can modify, delete data, perform database/server shutdown taking advantage of the vulnerabilities in the system. This paper presents the various attack method, their classification using which the system administrators and programmers can understand about SQLIA and secure the web application.

However, as technology develops, so will the threats and techniques used by the malicious users. As storage on in internet is of more trend nowadays care should be taken to secure the data from being stolen by malicious users. Hence securing of the system against SQL injection attack is of great importance.

ACKNOWLEDGEMENTS

I sincerely thank my guide Asst Professor Mr. Ramkumar. M for his proper guidance and valuable suggestions and also the institution for their support.

REFERENCES

[1] J. V. William G.J. Halfond and A. Orso, ―A classification of sql injection attacks and countermeasures,‖ 2006.

[2] A. Tajpour; M. Masrom; M. Z. Heydari.; S. Ibrahim; "SQL injection detection and prevention tools assessment, " Proc. Of ICCSIT 2010, vol.9, no., pp.518-522, 9-11 July 2010.

[3] Indrani Balasundaram. , Dr. E. Ramara. An Approach to Detect and Prevent SQL Injection Attacks in Database Using Web Service. IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, YEAR 2012

[4] C. Anley. Advanced SQL Injection In SQL Server Applications. White paper, Next Generation Security Software Ltd., 2002.

[5] S. Labs. SQL Injection. White paper, SPI Dynamics, Inc.,

[6] Abhishek Kumar Baranwal. Approaches to detect SQL injection and XSS in web applications. EECE 571B, TERM SURVEY PAPER, APRIL 2012

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

[8] Anyi Liu , Yi Yuan , Duminda Wijesekera , Angelos Stavrou,SQLProb: A Proxy-based Architecture towards Preventing SQL Injection Attacks,

[9] Atefeh Tajpour , Suhaimi Ibrahim,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

[10] StephenW.Boyd,AngelosD.Keromyti,SQLrand:Preventing SQL Injection Attacks.

[11] Devata R. Anekar ,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

[12] WilliamG.J.Halffond,AlessandroOrso,Preventing SQLInjectionAttacksUsingAMNESIA,ICSE,2006,Shanghai,China

[13] Etinene Janot ,Pavol Zavarsky,Preventing SQL Injection in online applications:Study,Recommendations and Java Solution Prototype based on SQL DOM,Application Security Conference,Ghent,Belgium,19-22 May 2008.

[14] Angelo Ciampa,Corrado Aaron Visaggio,Massimiliano Di Penta,A heuristic-based approach for detecting SQL Injection vulnerabilities in Web applications,ICSE Capetown,2-8 May 2010,pp 43-49.

[15] Sruthi Bandhakavi,Prithvi Bisht,P. Madhusudan,V.N. Venkatakrishnan, CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluations

[16] M. Dornseif. Common Failures in Internet Applications May 2005. http://md.hudora.de/presentations/

[17] D. Litchfield. Web Application Disassembly with ODBC Error Messages. Technical document, @Stake, Inc.,2002.http://www.nextgenss.com/papers/webappdis.doc. 2002.http://www.spidynamics.com/assets/documents/ WhitepaperSQLInjection.pdf.

[18] F.Bouma.Stored Proceduresare Bad, O'kay?Technicalreport, Asp.Net Weblogs,November2003.http://weblogs.asp.net/fbou ma/ archive /2003/11/18/38178.aspx.

[19] Mod security http://www.modsecurity.org/

[20] Preventing SQL injection by language level interpretation, PHP escaping http://en.wikipedia.org/wiki/SQL_injection