Review of SQL Injection Attack and Method for Detection and Prevention in ASP.NET Web Applications.

Rahul Sharma[1], Sharad Chauhan[2]

[1]Student, M. Tech, E-Max College of Engineering and Technology, Ambala
[2]HOD, Dept. of CSE, E-Max group of Institutions, Ambala

Abstract-- **This paper gives an overview to the SQL Injection attacks and methods to prevent them. we discuss all the proposed models in asp.net to block SQL Injections in web applications. We also describes the technique to prevent SQL injections attacks occurring due to dynamic sequel statements in database stored procedures are often used in e-commerce applications. we know that SQL injection attack can be easily prevented by applying more secure scheme in login phase that is authentication and authorization of the valid users. To address this problem, we present an overview of the different types of attacks with descriptions and examples of how attacks of that type could be performed and their detection & prevention schemes. This paper contains the strengths and weaknesses of various SQL injection attacks. At last we also proposed the scheme to handle the SQLIA and to prevent them.**

Keywords-- **Cybercrime, SQL(Structured Query Language) injection, Attacks, ASP.NET.**

I. Introduction

In the recent years, the World Wide Web (WWW) has a staggering growth of many Web base applications which have developed to meeting various purposes. Now-a-days, Security is the most important attribute for all web sites. Providing secure experience is one of the key principles in the process of gaining customer confidence . Nowadays, almost all the websites are asking to store user's personal information in servers to understand the customer and serve better. It is the responsibility of an organization to confirm that customer's data is safe and accessed in a secured manner.
Security in web application is always a big headache for the programmers but providing secure environments is one of the key principles in the process of gaining customer confidence . In this era of web applications, almost all websites are dynamic, i.e., all the websites are database driven and large data will be accepted from user.
SQL Injection Attacks (SQLIA"s) are one of the most severe threats to web security. They are frequently employed by malicious users for a variety of reasons like theft of confidential data, website defacement, sabotage etc. The number of SQLIA's reported in past years has been showing a steadily increasing trend and so is the scale of the attacks. It is, therefore, importance to prevent such types of attacks, and SQLIA prevention has become one of the most active topics of research in the industry and academia. There has been significant progress in the field and a number of models have been proposed and developed to counter SQLIA"s, but none have been able to guarantee an absolute level of security in web applications, mainly due to the diversity SQLIA's. One common programming practice in today's times to avoid SQL Injection is to use database stored procedures instead of direct SQL statements to interact with underlying databases in a web application, since these are known as parameterized queries and hence are not prone to the basic types of SLQ Injection . However, there are vulnerabilities too in this scheme, most notably when dynamic SQL statements are used in the stored procedures, to fetch the data in the database objects during runtime. Our work is cantered on this particular type of vulnerability in stored procedures and we develop a scheme for detection of SQLIA in scenarios where dynamic SQL statements are used.

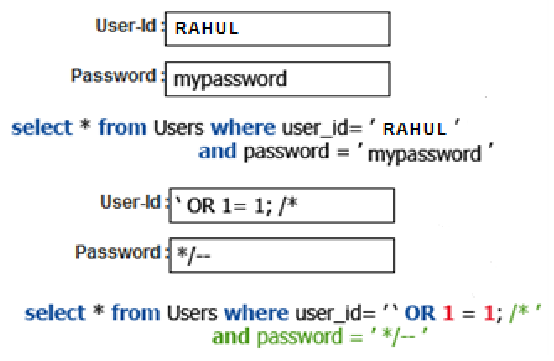
This paper is organised as follows: Section I show the introduction of web attack and how SQLIA is vulnerable cause of attack, section II show briefing about SQLIA, section III show types of SQLIA and in IV section show the various methods used for preventing SQLIA and in last section conclusion is present.

II. SQL INJECTION: THE 'NEED-TO-KNOW'

WHAT is SQL INJECTION ATTACK?

SQL Injection is a type of web application base security vulnerability in which an attacker is able to submit a database Sequel command, which is executed by a web application, exposing the back-end database tier. SQL Injection attacks can occur when a web application utilizes user-supplied data without proper validation or encoding as part of a command or query. The specially crafted user data tricks the application into executing unintended commands or changing data. Sequel Injection allows an attacker to create, read,alter, update or delete data stored in the back-end database. In its most common form, Sequel Injection allows attackers to access sensitive information such as social security numbers, credit,debit card number or other financial data. According to the Software Security Report Sequel Injection is one of the most prevalent types of web application security vulnerability..

.



```
User-Id : RAHUL
Password : mypassword

select * from Users where user_id= ' RAHUL '
                  and password = ' mypassword '

User-Id : ' OR 1= 1; /*
Password : */--

select * from Users where user_id= '' OR 1 = 1; /* '
                  and password = ' */-- '
```

Figure 1: SQL Injection

Key Concepts of SQL Injection

- SQL injection is software vulnerability that occurs when data entered by users is sent to the SQL interpreter as a part of an SQL query
- Attackers provide specially craft input data to the SQL interpreter and trick the interpreter to execute unintended commands
- Attackers utilize this vulnerability by providing specially craft input data to the SQL interpreter in such a manner that the interpreter is not able to distinguish between the intended commands and the attacker's specially craft data. The interpreter is tricked into executing unintended commands
- SQL injection exploits security vulnerabilities at the database tire. By exploiting the SQLI flaw, attackers can create, read, modify, or delete sensitive data .

III. TYPES of ATTACK

Currently, there are many types of vulnerabilities that vary in terms of complexity, detection and recovery.

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands. There are some classification of SQL injection types.

There are different methods of attacks that depending on the goal of attacker are performed together or sequentially. For a successful SQLIA the attacker should append a syntactically correct command to the original SQL query. Now the following classification of SQLIAs are

Tautology*:* This SQLI attack alters the database query by inserting vulnerable SQL tokens into the conditional query statements which always evaluates to true.

Example: SELECT * FROM <tablename> WHERE userId = <id> AND password = <wrongPassword> OR 1=1;

Union Queries*:* This query uses UNION keyword to access the information from other tables in the database. Such queries can be exploited by the attacker to get valuable data from the database.

Example: SELECT * FROM <tablename> WHERE userId = <id> AND password = <rightPassword> UNION SELECT creditCardNumber FROM CreditCardTable;

Piggy-backed Queries*:* In this attack the hacker appends _;' and a query to a database query to be executed on the database which will result in huge loss of data [3]. It could be one of the dangerous attacks that damage or destroy a table.

Example: SELECT * FROM <tablename> WHERE userId = <id> AND password = <rightPassword>; DROP TABLE <tablename>;

Blind Injection*:* Here the web developers hide the error messages from the user coming from the database so that the user is sent with a generic error displaying page. At this point the attacker sends a set of true/false questions to steal 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 =

Both the queries will return an error message in case the web application is secure, however if the input is not validated then chances of injection exist.

IV. PREVENTION OF SQLIA

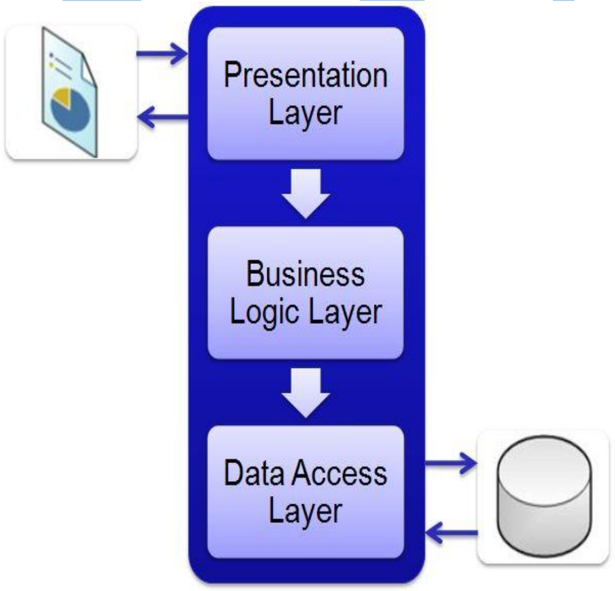
Validate all textboxes in web page with Range Validator or Regular expression Validator*:* Use ASP.NET validator controls such as *RangeValidator* or *RegularExpressionValidator* to constrain the input supplied through server controls.

Avoid using concatenation select queries*:* The concatenation of user input to form SQL commands result in SQL Injection. So use Stored Procedures to create SQL queries. The scanner tool detects if there is a concatenated query and reports leak.

Stored Procedure: Stored procedure is a part of database that programmer could set an extra abstraction layer on the database. As stored procedure could be coded by programmer, so, this part is as inject able as web application forms. Depend on specific stored procedure on the database there are different ways to attack. In the following example, attacker exploits parameterized stored procedure.

CREATE PROCEDURE DBO .is Authenticated @user Name varchar2, @pass varchar2, @pin int
AS EXEC ("SELECT accounts FROM users WHERE login='" +@user Name+ If' and
pass='" +@password+ " „ and pin=" +@pin);
GO For authorized/unauthorized user the stored procedure returns true/false. As an SQLIA, intruder input "SHUTDOWN; - -" for username or password. Then the stored procedure generates the following query: SELECT accounts FROM users WHERE login= 'doe' AND pass=' ';

SHUTDOWN; -- AND pin = after that, this type of attack works as piggy-back attack.
The first original query is executed and consequently the second query which is illegitimate is executed and causes database shut down. So, it is considerable that stored procedures are as vulnerable as web application code.
Inference: By this type of attack, intruders change the behaviour of a database or application. There are two well known attack techniques that are based on inference: blind injection and timing attacks. Blind Injection: Sometimes developers hide the error details which help attackers to compromise the database. In this situation attacker face to a generic page provided by developer, instead of an error message. So the SQLIA would be more difficult but not impossible. An attacker can still steal data by asking a series of True/False questions through SQL statements. Consider two possible injections into the login field:
SELECT accounts FROM users WHERE login= 'doe' and 1 =0 -- AND pass = AND pin=O
If the application is secured, both queries would be unsuccessful, because of input validation. But if there is no input validation, the attacker can try the chance. First the attacker submits the first query and receives an error message because of "1 =0 ". So the attacker does not understand the error is for input validation or for logical error in query. Then the attacker submits the second query which always true. If there is no login error message, then the attacker finds the login field vulnerable to injection.
3-tier Architecture of web application:
1) **User interface tier**: This layer forms the front end of the web application. It interacts with the other layers based on the inputs provided by the user.
2) **Business logic tier**: The user request and its processing are done here. It involves the server side programming logic. Forms the intermediate layer between the user interface tier and the database tier.
3) **Database tier**: It involves the database server. It is useful in storage and retrieval of data.



VI. CONCLUSION

In web applications in order to alleviate the vulnerabilities the web developer should be conscious on two thing.
First one is the developer must be responsible to ensure security of the application from the beginning of coding and another one is developer must check their web applications for leaks before making them public. But implementing security is only part of the solution. Another important part is vigilance. Even if our application has many safeguards, we need to keep an eye on our web application to protect it from newly arrived security attacks. So monitor the web application‘s event logs and perform repeated attempts to log into your application. Continually keep the application server up to date with the latest security updates. Our developed tool considers the web application as three-tiered: Presentation, application and storage. Here web browser is the presentation, ASP.NET is the application and the database is the storage. The scanner tool study the web application source code for leaks and report about the discovered leaks if exist, otherwise report the code as secure.

REFERENCES

[1]. http://www.w3resource.com/sql/sql-injection/sql-injection.php
[2]. W. Halfond, J. Viegas, and A. Orso. A Classification of SQL-Injection Attacks and Countermeasures. Proceedings of the IEEE
[3]. Sincy George, Member, IEEE, and Vivek Agarwal, Senior Member, IEEE"Optimum Control of Selective and Total Harmonic Distortion in Current and Voltage Under Nonsinusoidal Conditions" IEEE TRANSACTIONS ON POWER DELIVERY, VOL. 23, NO. 2, APRIL 2008.
[4]. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5615711&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5615711.
[5]. Security in ASP.NET Websites http://msdn.microsoft.com/en us/library/91f66yxt%28v=vs.80%29.aspx
[6]. Ruse, M., Sarkar, T., and Basu. S., Analysis & Detection of SQL Injection Vulnerabilities viaAutomatic Test Case Generation of Programs. Proc.10th Annual International Symposium on Applications and the Internet, 2010, pp. 31-37.
[7]. http://sqlmag.com/database-security/3-free- tools-prevent-sql-injection-attacks.
[8]. [8].http://weblogs.asp.net/scottgu/Tip_2F00_Trick_3A00_-Guard-Against-SQL-Injection-Attacks