International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-3 Issue-5, October 2013

Build a Web Application with Precautions to Prevent SQL Injection Attack

Mugdha Parande, Bhushan Hajare

Abstract— This paper explain SQL Injection, one of the most commonly exploited vulnerabilities found in web applications and second, In this outline possible steps which we can take to ensure that our website is protected from SQL Injection attacks.

Index Terms— SQLIA, SQL injection attack, Tutology, Blind SQL, Precaution.

I. INTRODUCTION

SQL injection attacks have become one of the most common and dangerous Web application security issues on the Internet. SQL inje*ction vulnerabilities* occur when an application takes user content data and uses it to construct SQL (Structured Query Language) statements without first properly validating or sanitizing that content. *SQL injection attacks* take advantage of SQL injection vulnerabilities to steal sensitive data from the database, modify or destroy the stolen data, execute administrative commands on the database, or in some cases take control of the whole machine. In recent years, SQL injection attacks have been used to store malware in databases and then distribute them through Web sites that are hosted on these compromised databases.

To prevent SQL injection attack while developing website we have to take care of it. Databases are the first target of attacker in web application. Web developers and database administrators should know the impact of this attack and should consider precautions from first step of implementation.

II. WHAT IS SQL INJECTION ATTACK

SQL Injection is a process often used by hackers attacking a website's database. Hackers are able to exploit security vulnerabilities within a website that allows the attacker to input malicious SQL code, which can be used to reveal and damage sensitive data held within the websites database.

These vulnerabilities can occur for several reasons, the most common of which is, in experience, a lack of proper filtering in relation to user input. In other words, it arises because the website is not properly filtering what the user is attempting to input.

Manuscript published on 30 October 2013.
*****Correspondence Author(s)
Ms. Mugdha Parande: B.E,(Information Technology), M.E.(Computer Engineering) form Shri L.R.Tiwari college of engineering , Currently working on network security.
Mr. Bhushan Hajare: B.E,(Information Technology) from D.K.T.E Engineering and textiles,Ichalkaranji, Currently working on database and networking.

III. PROCESS OD SQLIA

When wishing to hack a website the first step that an attacker will take is to locate vulnerability within the site that allows the hacker to manipulate data, take down or deface the site. This can be done by either manually scanning a site or by using a scanning tool such as Acunetix. Such scanning tools have grown in popularity as they require less technical skill than manually scanning a website. In addition, they are normally much faster and more efficient than the majority of hackers would be in manually scanning a site. Scanning tools do not require any administrative logon rights and can be run on any website in the world.

Once the attacker has scanned the site, and has found SQL Injection vulnerability, there are two ways he/she can go about exploiting this vulnerability. The first approach involves manually constructing an SQL string and injecting it into the site. The shortcoming of this approach is that it generally involves a lot of time and patience on the part of the hacker. The second approach involves using an application, such as Havij, that is specifically designed to exploit SQL vulnerabilities.

SQL injection is the use publicly available fields to gain entry to your database. This is done by entering SQL commands into your form fields instead of the expected data. Improperly coded forms will allow a hacker to use them as an entry point to your database at which point the data in the database may become visible and access to other databases on the same server or other servers in the network may be possible. The process of SQL injection attack also shown if following fig.[1]

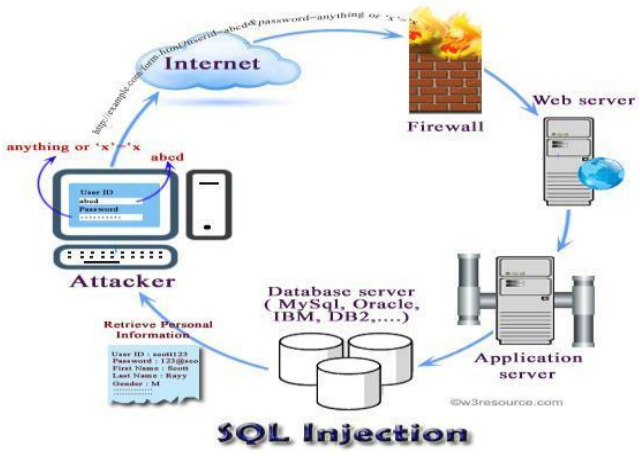

Figure [1]: SQL Injection Process

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

IV. METHODS OF SQLIA WITH EXAMPLE

There are many methods to getting data in SQL injection as following:

A. Basic Injection:

- **Using Tautology:**

Query:
"**SELECT** ***FROM**user_details
WHERE userid = 'uid' and password = 'psw' ";
In the above code userid and password data which are received from an user are stored in uid and pwd . The interpreter will execute the command based on the inputs. Now if an attacker provides **abcd** as userid and **anything' or 'x'='x** as password, then the query will be constructed as
sql = "select * from user_details where userid = 'abcd' and password = 'anything' or 'x'='x' ";
Based on operator precedence, the WHERE clause is true for every row, therefore the query will return all records. In this way, an attacker will be able to view all the personal information of the users.

Figure [2]: Basic SQL Injection Attack

- **Union Query**

Query:
SELECT * FROM user_details **WHERE** userid ="

UNION SELECT * FROM EMP_DETAIS-- ' and password = 'abcd';

The two dashes (--) comments out the rest of the query i.e. ' **and password = 'abcd'**. Therefore, the query becomes the union of two SELECT queries. The first SELECT query returns a null set because there is no matching record in the table **user_details**. The second query returns all the data from the table **emp_details**.

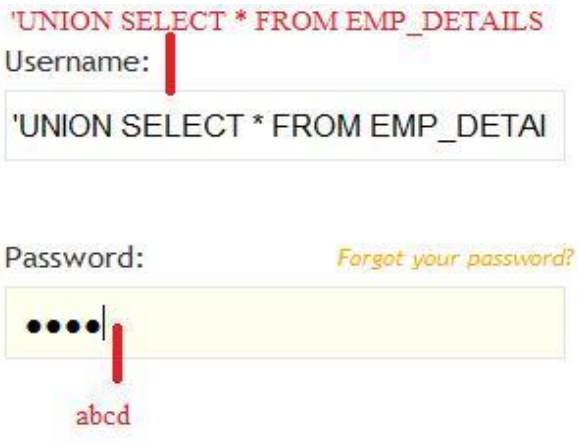

Figure [3]: Union Query Example

- **Piggy-backed Queries / Statement Injection** Query:

SELECT * **FROM** user_details **WHERE** userid = 'abcd' and password = ''; drop table xyz -- ';
After completing the first query (returned an empty result set (i.e. zero rows)), the database would recognize the query delimiter (";") and execute the injected second query. The result of executing the second query would be to drop table xyz, which would destroy valuable information.

Figure [4]: Piggy-backed Example

- **Second order attack**:

The attacker first injects into persistent storage such as a table row and collects some data and using this he will do further to collect more from the database.

- **Using Comments**:

By inserting comments the attackers can truncate some portions like „WHERE" clause in SQL statement.

- **SQL injections by truncation:**

The attacker uses varying buffer lengths to truncate a delimited string and uses another input value to inject his commands of choice into a dynamically constructed SQL statement.

- **Through cross Site scripting**(SQL insertion attacks)

The vulnerabilities can occur when one programming or scripting language is embedded inside another.

- **Injection through cookies**
- **Injection through server variables**

B. Blind injection:

- **Conditional Responses**

One type of blind SQL injection forces the database to evaluate a logical statement on an ordinary application screen.

- **Conditional Errors:**

This type of blind SQL injection causes an SQL error by forcing the database to evaluate a statement that causes an error.

- **Time Delays**

If the response to a query takes too much time, then the attacker can infer something about the database.

V. WHAT CAN BE DONE TO PREVENT SQL INJECTION ATTACK?

1. Trust no data that comes through the system. Validate all textbox entries using validation controls, regular expressions, code, and so on. This involves checking the length of data to avoid instances of buffer over runs.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

If a field expects strings, ensure that the data received is string, not binary data or special characters. This can be done by maintaining and comparing against a white list rather than trying to figure what is bad data. Anything not present in the white list is rejected.

2. Never use dynamic SQL. Use prepared statements, parameterized queries or stored procedures instead whenever possible.

3. Update and patch. Vulnerabilities in applications and databases that hackers can exploit using SQL injection are regularly discovered, so it's vital to apply patches and updates as soon as practical.

4. Firewall: Consider a web application firewall (WAF)

Either software or appliance based – to help filter out malicious data. Good ones will have a comprehensive set of default rules, and make it easy to add new ones whenever necessary. A WAF can be particularly useful to provide some security protection against a particular new vulnerability before a patch is available.

5. Reduce your attack surface: Get rid of any database functionality that you don't need to prevent a hacker taking advantage of it. For example, the xp_cmdshell extended stored procedure in MS SQL spawns a Windows command shell and passes in a string for execution, which could be very useful indeed for a hacker. The Windows process spawned by **xp_cmdshell** has the same security privileges as the SQL Server service account.

6. Use appropriate privileges: don't connect to your database using an account with admin-level privileges unless there is some compelling reason to do so. Using a limited access account is far safer, and can limit what a hacker is able to do.

7. Don't store secrets in plain text. Encrypt or hash passwords and other sensitive data; you should also encrypt connection strings.

8. Exceptions should divulge minimal information; hackers can learn a great deal about database architecture from error messages, so ensure that they display minimal information. Use the "RemoteOnly" customErrors mode (or equivalent) to display verbose error messages on the local machine while ensuring that an external hacker gets nothing more than the fact that his actions resulted in an unhandled error. set debug to false

9. Don't forget the basics; Change the passwords of application accounts into the database regularly. This is common sense, but in practice these passwords often stay unchanged for months or even years. Use a random Username except Admin. Use a random password which contains numbers and special character.

10. Buy better software, Make code writers responsible for checking the code and for fixing security flaws in custom applications before the software is delivered.

11. Employ filters that prevent characters like single or double quotes, backslashes, and colons and so on from being passed from a web form into the SQL Server.

12. Only allow numeric values that are integers to be passed to the SQL Server that can be handled by simply using the ISNUMERIC command to validate the input.

13. Delete stored procedures from the SQL database that are not needed. Examples are xp_sendmail or xp_cmdshell, which are not normally needed, but can be used by hackers to send information or gain access.

14. Check privileges behind SQL commands, such as Startup and RUN, on the SQL Server Security TAB (for Microsoft (NSDQ:MSFT) SQL Server) and make sure the appropriate privileges are assigned for your environment.

15. Use parameterized queries. This is one of the best ways to prevent SQL injection. When you do so, variables are not concatenated to a query string but are sent as parameters. The values of the parameters are searched in the table for a match. So if the query we used in the earlier example should be re-written as:

```
string strQry =
"SELECT Count(*) FROM Users WHERE
UserName=@username " + "AND
Password=@password";
```

The values of @username and @password are compared against the values of the field in the table as against being concatenated. If the hacker typed in

“a‟ or 1=1 –”, the result of the SQL statement is compared against the values in the table. That returns a value of False as there is no user with a name “a‟ or 1=1 –”.

16. Restrict permissions to web user logins to access only the table in question. If the site *is* exploited, this measure restricts damage to just one table. In short, use a least privileged database account.

17. Use escape routines for all user supplied input. Parameterized SQL is the way to go at times. But if for some reason, the only available option is Dynamic SQL, input characters that have a special meaning to SQL Server should be handled to safeguard the input. Escape routines add an escape character to characters that have special meaning to SQL Server, thereby making them harmless. For example, to escape a single quote („) you can use the following routine:

```
[php]private string CleanSqlStmt(string
inputSQL)&lt;br /&gt;&lt;br /&gt;{&lt;br
/&gt;&lt;br /&gt;return
inputSQL.Replace("'","''");&lt;br /&gt;&lt;br
/&gt;}[/php]
```

18. Build strong exception handling routines: Exception handling routines should not provide any information that might assist the hacker in his efforts. On one website, I got this error message when I provided the input “a‟ or 1=1”.

19. Most web applications employ a middleware technology designed to request from a relational database in SQL

20. Prevention in WordPress. WordPress is a database backed platform that is based on PHP scripts, which makes it vulnerable to SQL injection attacks. This means that hackers can use URL insertion attacks to access your database. Once a hacker get access to your website using an SQL injection technique, he/she can easily steal sensitive information, such as customer data or credit card details, modify your website content or even delete your web files. SQL injections can be difficult to detect as it depends on the hacker imagination and experience. However you can prevent such attacks by changing the default database tables prefix and setting strict rules of accessibility in your .htaccess file hosted on your web server or by your web hosting provider.

VI. CONCLUSION

Simple precautions can shield web applications from SQL injection. Making application security a priority at the time of design, and including security.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

testing as part of the non-functional requirements are the beginning of good programming practices. Testing each of the points listed above should safeguard your software against SQL injection.

REFERENCES

[1] By1Prasant Singh Yadav, 2 Dr pankajYadav, 3Dr. K.P.Yadav "A Modern Mechanism to Avoid SQL Injection Attacks in Web Applications",IJRREST: International Journal of Research Review in Engineering Science and Technology ,Volume-1 Issue-1, June 2012.

[2] By MayankNamdev *, FehreenHasan, Gaurav Shrivastav "Review of SQL Injection Attack and Proposed Method for Detection and Prevention of SQLIA"Volume 2, Issue 7, July 2012.

AUTHOR PROFILE

Ms. Mugdha Parande: B.E,(Information Technology), M.E.(Computer Engineering) form Shri L.R.Tiwari college of engineering , Currently working on network security.

Mr. Bhushan Hajare: B.E,(Information Technology) from D.K.T.E Engineering and textiles,Ichalkaranji, Currently working on database and networking.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)