2020 International Conference on Computing and Information Technology, University of Tabuk, Kingdom of Saudi Arabia.
Volume: 01, Issue: ICCIT- 1441, Page No.: 401 – 404, 9th & 10th Sep. 2020.

SQL Injection Countermeasures Methods

Hanan Alsobhi
Faculty of Computer and Information Technology
University of Tabuk
Tabuk, Saudi Arabia
392010376@stu.ut.edu.sa

Reem Alshareef
Faculty of Computer and Information Technology
University of Tabuk
Tabuk, Saudi Arabia
392010377@stu.ut.edu.sa

***Abstract*— Many data created very quickly every day and eventually this data is stored on databases. All processes and processors use the database management system because of the privacy of the stored data and the data must be secured in this system and do not fail in front of the potential database attacks, so it is necessary to use security models, although they differ between them. Database injection has become a common desirable attack targeting web applications, the attacker has access to the database to change SQL queries without authorization. In order to avoid losing data and making it vulnerable to the public, SQL injection must be studied and understood. In this paper, I explain SQL injection and its countermeasures.**

Keywords—SQL, Injunction, Attacks, Database, Countermeasure

I. INTRODUCTION

Everyone's needs databases to store any kind of data, because of the database's speed and reasonable cost. The advantage of using the database is that it automates various actions, saving resources and working hours. For example, instead of manually verifying transactions, users can rely on computer reports stored in the database. The question for everyone's is "is data safe in database?"

Security in today's world is one of the important and difficult tasks that people all over the world face in every aspect of their lives. SQL injection is one of the main methods that attackers use to hack a database, this type of attack exploits vulnerabilities in website or in a server of database. The attacker injects danger query segments created in order to change the intended effect, so that an attacker can gain unauthorized access to a database, read or modify data, make data unavailable to other users or even damage the database server.

In recent years, SQL injection attacks have increased rapidly, and became the first type to target web applications. 2010 During the first half, average daily number of SQL injection attacks worldwide reached about 400,000. [8]

Web applications and their databases require careful configuration and programming as well as effective anti-attack protection mechanisms to ensure security. To address SQL injection problems, many solutions have been suggested by researchers, and despite all this, there is no one definitive solution to ensure complete safety.

II. SQLIA IMPACTS

SQL is a database query language, which enables web-programming languages to communicate with database servers using dedicated libraries. Depending on each language, communication is done by queries. The server returns results based on the query. Web applications can be the target of attack. Injection when queries accept input from user or system.

An injection attack occurs when data comes from an untrusted source, or create dynamic queries based on incoming data, SQL injection can override authentication, data loss, denial of access, and it may damage the entire database or the host takeover. The main consequence of these vulnerabilities is:

Confidentiality: Loss of confidentiality about sensitive data stored at the level of search databases that are viewed by non-owners and unauthorized use of such data in illegal acts. [9]
Integrity: Successful injection provides the possibility to modify or delete data from infected databases. [10]
Authentication: Queries do not validate username and password, which makes an unauthorized user to query the database as an authorized user to access data without knowing the password or even the username. [11]
Authorization: Allows an attacker to change or upgrade privileges, if validity is available on the affected database, and thus more control over the system with greater powers. [12]
Functionality: Simultaneous data processing in databases is an essential feature of enabling users to simultaneously access and share data, and the attack destroys these functions. [13]

III. SQLIA TYPES

Tautologies: Used to inject the code into a conditional clause, so that it is always evaluated correctly, this process is carried out in the absence of input verification: [14]

Example without Injection:

```
SELECT info
FROM users
WHERE user_name = 'bob' AND user_pass =
'123456';
```

Fig. 1. SQL query without injection

Example with injection:

```
SELECT info
FROM users
WHERE user_name = 'bob' OR 1=1 -- ' AND
user_pass = '123456';
```

Fig. 2. SQL query with injection

Piggyback Queries: Attacker sends multiple queries embedded in one query, where the second query contains malicious queries and therefore runs as part of the main query. [14]

978-1-7281-2680-7/20/$31.00 ©2020 IEEE

```
SELECT user_name
FROM users
WHERE user_id = 'admin' AND user_pass= '123';
DELETE FROM users
WHERE user_name = 'admin';
```
Fig. 3. SQL query with injection

Alternate Encodings: This technique used to escape detection of unwanted characters, combine the function "CHAR" with number coding to return the original character. For example, the attacker used char (44) instead of the bad character (').
Illegal/Logical: This method relies on entering data so that the query becomes wrong, and therefore when the system runs the query you get errors making the system shows these errors and this is the important point where the emergence of errors shows to the attacker what will depend on him and also the techniques and gather important information about the type and structure of the backend database and methods he will use and also the elements that will deal together and thus facilitate Attack process. [14]
Stored Procedure: An attacker exploits stored procedures within a database system. These stored procedures are a set of codes that perform some tasks without having to write them every time and contain some dangerous tasks that the attacker can exploit as shutting down the system. [14]

```
SELECT info
FROM users
WHERE user_name = 'bob';
SHUTDOWN; -- AND user_pass = '123456';
```
Fig. 4. Execute shutdown procedures select info

Union query: The secondary query linked to the main query, using the word union, to obtain information about other tables from the database. An attacker can extract data type or information about columns and their properties. [14]

IV. SQLI COUNTERMEASURES

Disable unused features: Disable all features that you do not use. It is not necessary to keep functions of your server that you do not use because they are potentially dangerous for you.
Custom error message: When running the query, run a test to see if it returns an error. If there is an error, put a custom error. SQL errors give too much information to a malicious user.
Escape functions: The escape functions are very easy to set up and allow you to quickly secure your server against most attacks.

```
SHUTWOWN
...
```
Fig. 5. Type of danger function

Prohibit certain keywords:
Do not allow certain characters or keywords. There are indeed words that the user does not have to enter a query. If this kind of word is returned, it must be removed, because it will probably come from an attack.

```
DROP
UNION
...
```
Fig. 6. Type of danger keywords

Limit the size of the data: You can limit the size of the data entered by the user. We saw that some injections required a certain number of characters (you can limit the ID numbers to five characters, a login to 15 for example).

```
<input type="number" name="id" min="1" max="99999" size="5" />
<input type="text" name="lastname" size="15" />
```
Fig. 7. HTML input with limitation of size

Use the Prepare Statements: Use the prepare statements. This solution remains the most effective to protect against SQL injections. It will ask you a little more time (not much more) but will effectively secure your server.

PHP

```
$query = "INSERT `teachers`(`CodeT`, `NameT`, `RankT`, `SpecialtyT`,
`StatusT`) VALUES(:code, :name, :rank, :specialty, :status)";
$datas = array (
        'code'=> $_POST['code'],
        'name'=> $_POST['name'],
        'rank'=> $_POST['rank'],
        'specialty'=> $_POST['specialty'],
        'status'=> $_POST['status']
);
$statement = $db->prepare($query);
$results=$statement->execute($datas);
```
Fig. 8. Example prepare statement using PHP

ASP.NET – C#.NET

```
string query = @"SELECT * FROM Employees WHERE EUsername =
@EUsername AND EPassword = @EPassword";
using (SqlDataAdapter DA = new SqlDataAdapter(query,
this.CNXDatas))
{       DA.SelectCommand.Parameters.AddWithValue("@EUsername",
TB_Username.Text);
DA.SelectCommand.Parameters.AddWithValue("@EPassword",
TB_Password.Text);
DataSet DS = new DataSet();
this.CNXDatas.Open();
DA.Fill(DS, "Employees");
this.CNXDatas.Close();
}
```
Fig. 9. Example prepare statement using C#.net

V. SQLIA EVALUATION

Researchers have suggested several techniques to solve SQL injection problem. These technologies range from development best practices to fully automated frameworks for detecting and preventing SQLIAs. Here we will review these proposed techniques and summarize their advantages and disadvantages.

It is very clear that SQLI attacks are an important class of web application attack. Table 4.1 highlights the most recent while the full list is quite large. Any website is vulnerable to SQLI attacks if it is not properly secured. Although there is a large amount of research conducted on different types of SQLI attacks and mechanisms, there are deficiencies in the research that deals with effective techniques to detect and prevent them from occurring. In this section, the prevention techniques used against different types of SQLI attacks are described.

Web applications developed with HTML have always been vulnerable to SQL injection attacks. By entering the HTML command as input, attackers can manipulate the program to get what they want. SQL-DOM is one of the safeguards developed to handle injected HTML commands. SQL-DOM is able to convert HTML to structured data, which makes it difficult for hackers to enter HTML commands as inputs, and thus the modification is not utilized.

Another prevention method is SQLrand. There is a very smart method that converts the Instruction-Set Randomization application to the SQL language, and a random number is added to the conversion result, where the attacker will not be able to guess the supplement number and he will not be able to make any injection. Accordingly, according to Lloyd and Keromitis, "Any malicious user attempting a SQL injection attack will be frustrated, because the user rating entered in the" random "query will always be classified as a set of keywords, resulting in an expression error" [2].

Many other prevention methods such as AMNESIA, SQL Check, SQL Guard, and CANDID have proven successful in SQL injection prevention, but they are still unable to block all types of SQL injection. All of the methods listed below are unsuccessful in preventing a Stored Procedure attack, but the technique proposed by Havond and others. It uses a positive method of pollution and syntax evaluation to prevent all types of SQL injection.

According to Madman and Munch [18], staining is the best way to prevent SQL injection. It is accurate and efficient because it is purely dynamic.

SQL-DOM: This technology uses static analysis and application code run-time monitoring to detect and prevent SQLI attacks. The four main steps are hotspot identification, SQLquery modeling, tool implementation, and runtime monitoring. This technique used to prevent Tautology, Illegal, Piggyback, Union, Inference, Alternate encoding, SQLIA, SQLI + DNS Hijacking but cannot be prevented Stored Procedure. [1]

SQLrand: This technique uses SQL randomization. Whenever the attacker tries to inject the attack, the database analyst temporarily stores and exits it. This technique used to prevent Tautology, Piggyback, Union, and Inference but cannot be prevented Illegal, Stored procedure and Alternate coding. [2]

AMNESIA: This technology uses static analysis and application code run-time monitoring to detect and prevent SQLI attacks. It has four steps as follows: define hotspots, build SQL query forms, implement the tool, and monitor runtime .This technique used to prevent Tautology, Illegal, Piggyback, Union, Inference and Alternate encoding but cannot be prevented Stored Procedure. [3]

Tainting: This highly automated method uses positive staining and the concept of aware syntax assessment to counteract SQLI attacks. This technology uses trusted data sources and only allows data from these reliable data sources. This technique used to prevent Tautology, Illegal, Piggyback, Union, Inference, Alternate encoding and Stored Procedure. [4]

SQLCheck: Using context-free grammar and translator analysis techniques, this algorithm prevents command injection attack. This technique used to prevent Tautology, Illegal, Piggyback, Union, Inference and Alternate encoding but cannot be prevented Stored Procedure. [5]

SQLGuard: This technique performs a run-time comparison of user input with the analysis tree in the SQL statement before and after user input. The technology detects and eliminates SQLI attacks by linking the intended query structure to the instantiated query. This technique used to prevent Tautology, Illegal, Piggyback, Union, Inference and Alternate encoding but cannot be prevented Stored Procedure. [6]

CANDID: SQL injection attacks detected by dynamically comparing filter inputs with programmable query structure. This technique used to prevent Tautology and partially prevent against Illegal, Piggyback, Union, Inference and Alternate encoding but cannot be prevented Stored Procedure. [7]

Table I: Prevention Techniques

ID	Techniques
A	AMNESIA
B	Tainting
C	SQLCheck
D	SQLGuard
E	CANDID

Table II: Attack types

ID	Techniques
1	Illegal
2	Piggyback
3	Union
4	Inference
5	Stored Procedure

Table III: Comparison of prevention techniques with respect to attack types

	1	2	3	4	5
A	O	O	O	O	X
B	O	O	O	O	O
C	O	O	O	O	X
D	O	O	O	O	X
E	O	O	O	O	X

VI. CONCLUSION

This report allowed us to present some of the best-known SQL injections, thus highlighting the importance of database protection. These injections make it possible to have access to the data, to modify or even to delete it, to usurp a user's account, to access the database structure, to circumvent the management rules, or to obtain some information about the server. Some simple protections prevent the database from being the victim of such attacks, such as the banning of certain keywords, the use of escape functions or the deactivation of error messages that are a little too talkative.

REFERENCES

[1] R. A. McClure, and I. H. Kruger, "SQL DOM: compile time checking of dynamic SQL statements", Software Engineering, 2005.

ICSE 2005. Proceedings. 27th International Conference on. IEEE, pp. 88-96, 2005.

[2] S. W. Boyd, and A. D. Keromytis, "SQLrand: Preventing SQL injection attacks," International Conference on Applied Cryptography and Network Security, Springer, Berlin, Heidelberg, pp. 292-302, 2004.

[3] W. G. J. Halfond, and A. Orso, "Preventing SQL injection attacks using AMNESIA", Proceedings of the 28th international conference on Software engineering. ACM, 2006.

[4] Halfond, William GJ, Alessandro Orso, and Panagiotis Manolios. "Using positive tainting and syntax-aware evaluation to counter SQL injection attacks." Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering. ACM, pp. 795- 798, 2006.

[5] SQL CHECK Constraint

[6] I. Lee, et al, "A Novel Method for SQL Injection Attack Detection Based on Removing SQL Query Attribute Values", Vol.55, No. 1-2, pp.58-68, 2018.

[7] S. Bandhakavi, et al, "CANDID: preventing sql injection attacks using dynamic candidate evaluations", Proceedings of the 14th ACM conference on Computer and communications security. ACM, pp. 12-24, 2007.

[8] IBM Internet Security Systems X-Force® research and development team, "IBM Internet Security Systems™ X-Force® 2009 Mid-Year Trend and Risk Report," Aug. 2009.

[9] D. A. Kindy, and A. S. K. Pathan, "A survey on SQL injection: Vulnerabilities, attacks, and prevention techniques", 2011 IEEE 15th International Symposium on Consumer Electronics (ISCE), pp. 468-471, 2011.

[10] Y. W. Huang, et al, "Securing web application code by static analysis and runtime protection", Proceedings of the 13th international conference on World Wide Web, ACM, pp. 40-52, 2004.

[11] M. R. Borade, and N.A. Deshpande, "Extensive Review of SQLIA's Detection and Prevention Techniques", International Journal of Emerging Technology and Advanced Engineering,, Vol. 3, No. 10, 2013.

[12] R. Johari, and P. Sharma, "A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection", In Communication Systems and Network Technologies (CSNT), 2012 International Conference on, pp. 453-458, 2012.

[13] V. Nithya, R. Regan, and J. Vijayaraghavan, "A Survey on SQL Injection attacks, their Detection and Prevention Techniques", Int. J. Eng. Compu. Sci., Vol. 2, No. 4, pp. 886-905, 2013.

[14] C. Anley, "Advanced SQL Injection in SQL Server Applications", White paper, Next Generation Security Software Ltd., 2002.

[15] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan, "CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks," ACM Trans. Inf. Syst. Secur., vol. 13, no. 2, pp.1-39, 2010.