Advances in Computer Science and Information Technology (ACSIT)
p-ISSN: 2393-9907; e-ISSN: 2393-9915; Volume 3, Issue 3; April-June, 2016, pp. 166-168

http://www.krishisanskriti.org/Publication.html

Detection and Prevention of SQL Injection Attacks in Web Applications

Neetika Srivastava[1], Shivani Thapar[2] and Aruna Bhat[3]

[1,2,3]Department of IT, IGDTUW
E-mail: [1]neeti92sri@gmail.com, [2]sthapar91@gmail.com, [3]abigit06@yahoo.com

Abstract—*Web applications use the database at the backend for storing data and SQL (Structured Query Language) for insertion and retrieval of data. There are some malicious codes that can be attached to the SQL called SQL Injection. SQL injection is a hacking method that is based on the security vulnerabilities of web application. The paper aims to give a comparison between the tools used for detection and discuss a prevention technique recently proposed and its limitations*

1. INTRODUCTION

The World Wide Web has been developed with very rapid progress in recent years. Web applications use the database at the backend for storing data and SQL (Structured Query Language) for insertion and retrieval of data. There are some malicious codes that can be attached to the SQL called SQL Injection. One of the best ways to secure applications is by limiting access to those authorized to access the application. SQL injection is a hacking method that is based on the security vulnerabilities of web application. 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. The attacker's input is transmitted into an SQL query in such a way that it will form an SQL code. Different types of SQL injection attacks are possible. Some of them are mentioned below [1]:

TYPE OF ATTACK	WORKING METHOD
Tautologies	SQL injection queries are injected into one or more conditional statements so that they are always evaluated to be true.
Logically incorrect queries	Using error messages rejected by the database to find useful data facilitating injection of the backend database.
Union Query	Injected query is joined with a safe query using the keyword UNION in order to get information related to other tables from the application.
Stored Procedures	Many databases have built-in stored procedures. The attacker executes these built-in functions using malicious SQL Injection codes.
Piggy-backed queries	Additional malicious queries are inserted into an original injected query.
Inference - Blind Injection - Timing Attacks	An attacker derives logical conclusions from the answer to a true/false question concerning the database. Information is collected by inferring from the replies of the page after questioning the server true/false questions. An attacker collects information by observing the response time (behaviour) of the database.
Alternate encodings	It aims to avoid being identified by secure defensive coding and automated prevention mechanisms. Hence, it helps the attackers to evade detection. It is usually combined with other attack techniques.

Once an attacker realizes that a system is vulnerable to SQL Injection, he is able to inject SQL Query /Commands through an input form field. This is equivalent to handing the attacker our database and allowing him to execute any SQL command including DROP TABLE to the database

An attacker may execute arbitrary SQL statements on the vulnerable system. This may compromise the integrity of our database and/or expose sensitive information. Depending on the back-end database in use, SQL injection vulnerabilities lead to varying levels of data/system access for the attacker.

The aim of the thesis is to find a novel method that helps prevent SQL injection in web applications when it is already deployed on the server and being used.

2. DETECTION OF SQL INJECTION ATTACKS

Detection of SQL injection attack on web applications is very necessary nowadays as it possess a real threat for the application administrators. Databases running in the backend must never be visible to the end user or anyone who might misuse the data. To do the process of detection and having a comparative analysis of tools that work in this area, we make use of the following tools:

a) VEGA: Vega is a free and open source scanner and testing platform to test the security of web applications. Vega can help you find and validate SQL Injection, Cross-Site Scripting (XSS), inadvertently disclosed sensitive information, and other vulnerabilities. It is written in Java, GUI based, and runs on Linux, OS X, and Windows.[4]
b) SQLMAP: SQLMAP is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.[5]
c) HAVIJ: Havij is an automated SQL Injection tool that helps penetration testers to find and exploit SQL Injection vulnerabilities on a web page. It can take advantage of a vulnerable web application. By using this software user can perform back-end database fingerprint, retrieve DBMS users and password hashes, dump tables and columns, fetching data from the database, running SQL statements and even accessing the underlying file system and executing commands on the operating system. The power of Havij that makes it different from similar tools is its injection methods. The success rate is more than 95% at injectiong vulnerable targets using Havij. The user friendly GUI (Graphical User Interface) of Havij and automated settings and detections makes it easy to use for everyone even amateur users. [6]

Work Flow

a) Collected a list of 100 vulnerable sites from [2]. The sites listed are vulnerable to SQL injection and other similar attacks.
b) Vulnerability analysis of the web applications on VEGA.
c) Vulnerability analysis of the web applications on SQLMAP.
d) Vulnerability analysis of the web applications on HAVIJ.
e) Comparative analysis of the above outputs.

Inference

a) Out of the 100 websites under test, 45 were prone to SQLia according to VEGA testing.
b) Out of the 100 websites under test, 30 were prone to SQLia according to SQLMAP testing.
c) Out of the 100 websites under test, 63 were prone to SQLia according to HAVIJ testing.
d) Traditional SQL injection that attempts to retrieve complete tables is rare on websites
e) Blind SQL injection is most prevalent.

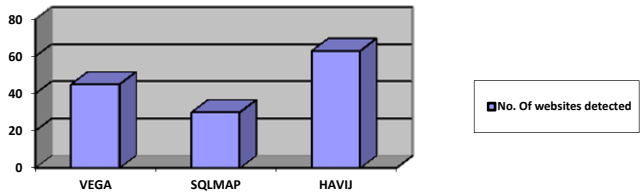

Fig. 1: Comparative analysis of the detection tools used to detect SQLIA.

Hence, false negative ratios (false negative is an error in which a test result improperly indicates no presence of a condition (the result is negative), when in reality it is present.) of the tools are:

- VEGA= 55%
- SQLMAP= 70%
- HAVIJ= 33%

3. PREVENTION OF SQL INJECTION ATTACKS

Quite a number of techniques and tools have been developed by companies and proposed by individuals for prevention of SQL injection attacks in web applications. For preventing the SQLIAs defensive coding has offered as a solution but it is very difficult. Not only developers try to put some controls in their source code but also attackers continue to bring some new ways to bypass these controls. Hence it is difficult to keep developers up to date, according the last and the best defensive coding practices [3].

The latest one proposed works on the principle of hash algorithms and is given in [7]. It uses the hash algorithm to detect and also prevent the malicious query to run on the application's backend. It simply blocks the query on the criteria that the hash of the input query does not match the hash of the query that should actually be passed.

The flowchart of the system is shown below:

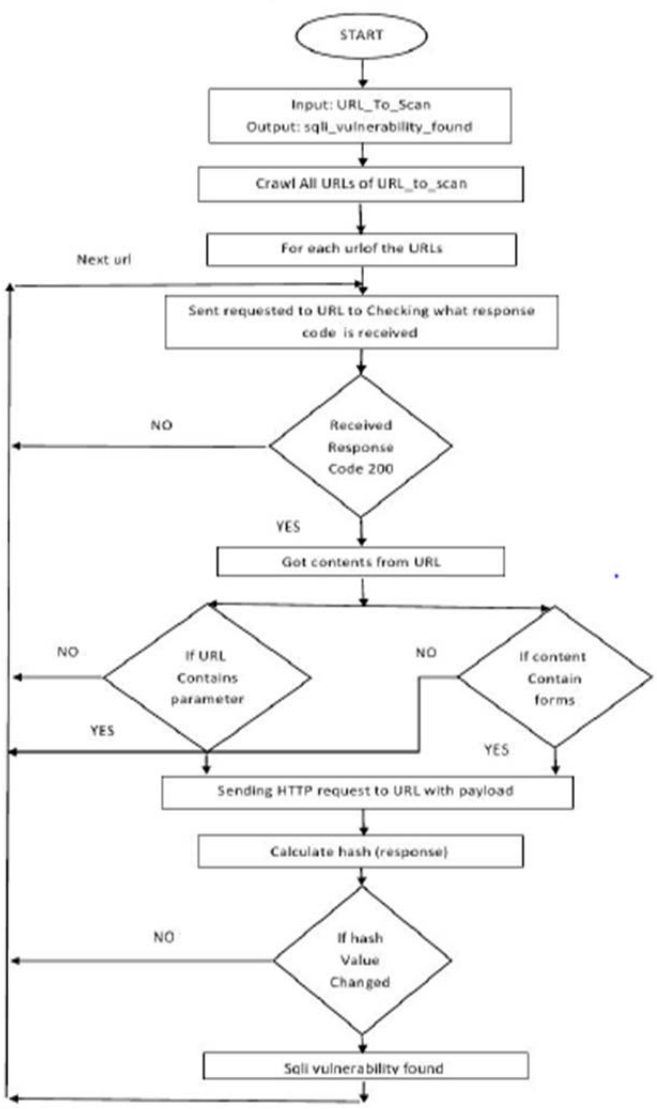

Fig. 2: Flowchart of the system proposed in [7].

The drawback that this proposed method has can be explained using the following example:

Ex 1: **Input query:** Select name from Employee where

ID=009

MD5 Hash Calculated:

22a818a76ec793660da86643bd3f5829

Ex 2: **Input query:** Select name from Employee where

ID=098

MD5 Hash calculated:

634982767e69b44f43cf54dd5c647f51

Point to be noted is even though both the queries are valid and should run fine, but the proposed work does not let the second query run because the hash does not match with the valid query run before in Ex 1. This creates a number of false positives and the system does not run the valid queries too.

4. CONCLUSION

There are a lot of techniques and tools to find bugs, errors or vulnerabilities in web application. Various detection and prevention techniques that have been used or proposed carry their advantages and drawbacks. No method can be said to have no drawbacks.

Future work includes the:

a. Detection of SQL injection using different Hash algorithms.

b. Configuring a firewall that is placed between the User interface and backend database that is responsible for the filtering and cleansing of the query provided by the user. Only after cleansing should the query be sent to the backend for processing. Filtering is to be done using a cheat sheet of possible malicious queries that may cause the attack.

5. ACKNOWLEDGEMENTS

Sincere thanks to Aruna ma'am for valuable guidance in the work done. Big thanks to the head of department and all the professors for their immense support in our work.

REFERENCES

[1] Kindy, Abdoulaye, D., and Pathan, A.K., " A survey on SQL injection: Vulnerabilities, attacks, and prevention techniques.", (2011): 77.

[2] SQL Vulnerable sites [Online]. Available: http://waziristanihaxor.blogspot.in/2015/02/SQL-VULNERABLE- websites-List.html

[3] Tajpour, Atefeh. "Evaluation of sql injection detection and prevention techniques." Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference on. IEEE, 2010.

[4] VEGA Vulnerability scanner [Online]. Available: https://subgraph.com/vega/

[5] SQLMAP: automatic SQL injection and takeover tool [Online]. Available : http://sqlmap.org/

[6] Havij Vulnerability scanner [Online]. Available: http://downloadhavij.blogspot.in/2013/07/download-havij-117-free-full-version.html

[7] Mahdi, Maki, and Mohammad, A.H., "USING HASH ALGORITHM TO DETECT SQL INJECTION VULNERABILITY." (2016).