International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 04 | Apr-2016 www.irjet.net p-ISSN: 2395-0072

SQL Injection Attack :Detection and Prevention

Swayam Charania[1], Vidhi Vyas [2]

[1]*Student, MCA Department, Sardar Patel Institute of Technology, Mumbai,India*
[2]*Student, MCA Department, Sardar Patel Institute of Technology, Mumbai,India*

Abstract— *Web applications comprise of a large degree of functionalities and usefulness. As more and more sensitive data is available on internet attackers are becoming more interested in such data revealing which can cause massive damage. SQL injection is an attack that is used to infiltrate the database of any web application that may lead to alteration of database or disclosing important information. As applications get web based ,attackers provide infected sql queries which can modify the queries and extract configuration information. Stored procedure are considered being more secure, but is not the case .It is necessary to protect web based applications. Detection of Sql Injection becomes necessary and trivial ,where there exist vulnerable datasets. Prevention of such attacks provide a way to secure these datasets ,keeping them unharmed from potential attackers. Various detection measures have been elaborated and described. Tools used in the detection techniques such as sqlmap have been briefly described.*

Keywords: *SQL Injection, SQLMAP,Postgre-SQL,CANDID,WAVES*

1. INTRODUCTION

SQL Injection (SQLIAs) is a technique where attackers can inject SQL queries through input of a web page .Such SQL commands can alter the database and modify the contents. They may even retrieve important information thus harming the integrity of the database.SQL Injections is usually known as an attack vector for websites, but is normally to attack any type of SQL Database. The general concept is to bypass the server level and gain access to the backend. Web applications are often vulnerable to attacks, that provide attackers easily access to the application's underlying database.One such tool depicted is SQL Map, which checks a website for its vulnerability. In this paper we also list classification, detection of SQL Injections and preventive measures to avoid SQLIA's

Web Applications

SQLIA's take place largely in web applications, thus working of a web application needs to be understood first. Web applications comprise of three tiers. The first tier is on the users side and has a basic browser. The second tier contains a dynamic content generator write in php or jsp. Tier three is where the database resides. This is the tier prone to SQL Injection attack. The vulnerabilities in a web application of a system can be exploited to implement this attack. Vulnerabilities include lack of data validations, insufficient concatenation of variables etc. Main categories that the injection attacks are divided into are:

1) First order attack: The attackers write malicious queries which causes the code to be executed immediately.[4]
2) Second order attack: The attacker modifies the contents of the database using SQL statements like insert and delete. An attack is then executed by another action.[4]
3) Lateral Attack: The attackers plays with implicit functions. Example: use of To_char() by changing the environment variables.[4]

2. CLASSIFICATION

Classification of SQL Injection can be classified broadly as:

Tautologies: This type of attack uses conditional queries and inserts SQL tokens into them, which proves to be always true.

Illegal/Logically Incorrect Queries: Attackers use the database error messages to find vulnerabilities in the applications.

Union Query*:* Attackers inject infected queries to the existing safe queries by using the UNION operator and thus retrieve information form the database.

Piggy-backed Queries: Attackers attach delimiters such as ";" to the original query. Doing so causes multiple queries to be executed in which the first query is legitimate followed by infected and illegitimate queries. queries. Normally the initial query is trusted query, whereas following queries could be illegitimate.

Stored Procedure: Stored procedure is a subset of database that a developer could create an extra abstraction layer on the database. Depending on specific stored procedure on the database there are different ways to attack.

Blind Injection: Developers hide error messages which can be useful to attackers to plan and SQL Injection Attack. In this situation the attacker comes across a static page designed by the developers. Hence, attacks is difficult but not impossible as the attackers can ask True/False questions by using SQL commands.

Timing Attacks:

By using timing delays in the database's response the attacker is able to gather the information by observing the amount of time taken to execute a query. The attacker generates a long query using if-else statements and thus measuring the amount of time taken for the page to load to determine if the injected statement is true.

Alternate Encodings: ASCII and Unicode encodings are used by the attackers so that they can escape from the filter which scans "special characters" Example: Using "char(44)" in place of a single quote is an example of alternate Encodings.

.AN EXAMPLE OF DETECTION OF SQL INJECTION ATTACKS

Consider a URL for downloading a file."http://localhost:3627/StaticWeb/dDownload.aspx?id=4EC898F0-57AA-401F-82F5-5F0D856D4301" link is generated as a result of a web request from the user. If an attacker modifies this query as id='abc'; insert into table name values(7,'xyz').This query leads to a generation of two queries ,one for downloading a file and other for inserting values in a table. This will lead to ambiguous changes in database. Similarly consider a login page where user enters in text fields. Such input data can be modified by the attacker so as to attack the database system with SQL injection. A method can be proposed to detect this attack. Web applications can be static and dynamic.[1] In static web application, in which responses generated to a web request are fixed. So it will be generating a fixed query set for static web applications. In case of dynamic applications, queries are varied and thus there is a need to monitor every query generated by users.

In case of dynamic web application, each activity of a valid user can be categorized. Each web request can be categorized, considering a web application of a blog, like login, adding a web article, reading an article, commenting on that article etc. can be used to differentiate the legit mapping. Using this categories non-deterministic mapping can be made accurate up to some extent. This practice session are performed in attack less environment. A user can be granted a web service, input validations are performed and only then the SQL queries are generated. Before retrieving the result from the database a mapping is performed wherein the SQL queries will be checked against a mapping pattern. pattern. Matching will be done on the basis of web request that is diseased with either SQL injection, and the subsequent web query generated which includes the same attack bypassed from previous phase. If matching valid pattern are found then only that query will be fired over database to fetch the results otherwise will be treated as potential threat and can be accessed by admin.

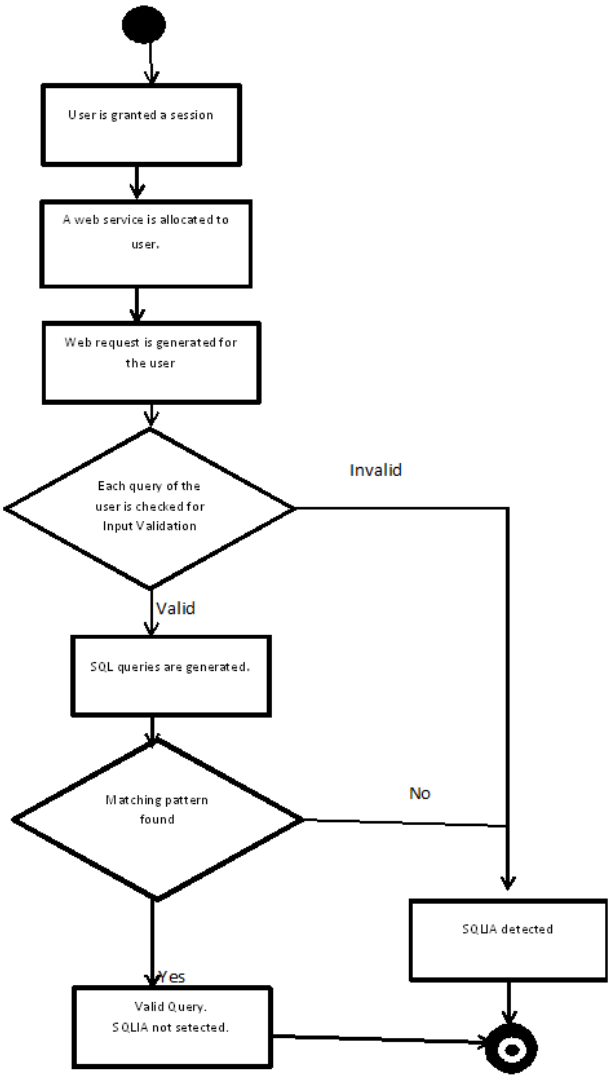

Fig 1.Systematic flow of dynamic web application

Another method can be adopted which builds an efficient detection system between the attacker and the web server, which can shield the web server from the attack. Moreover, some detection system only detect vulnerabilities at the HTTP header in the GET methods. This method in addition to

headers, analyzes the payload(Deep Packet Inspection)of the packets. The first stages can be used to generate different types of SQLI attacks using tools like SQLMap. Infected data packets can be captured by Hardware Network Analyzer.[2]

It inspects the payload contained within the HTTP packet between the client and server and uses a packet matching algorithm to detect if the packet contained an attack. It also ranks the attack based on the type of information acquired and notifies the admin. Identification of the possible threats is necessary, and for this a misuse pattern concept can be applied. A misuse rule is such that it helps programmers design a protective measure for it. In turn the developers can determine as to how a misuse is performed and use it for disaster recovery and incident response purposes.

Numerous other detection methods are deployed to detect SQLIA attacks.

WAVES, a black box technique is used for testing web applications for SQLIA vulnerabilities.[3]

SAFELI,[5] a framework that uses static analysis proposed by Xiang Fu and Kai Qian for compile time identification of SQLIA vulnerabilities. SAFELI can scan the source code and will be able to detect possible vulnerabilities that cannot be discovered by black-box vulnerability scanners.

CANDID [6,7] modifies web applications written in Java through a program transformation. It mines the programmers intended query structure with the actual query issued and thus detects the SQL Injection attack.

3. DETECTING SQLIAs WITH SQLMAP

Introduction

SQLMap is an open source tool. It is used to automate the process of detecting and exploiting SQL injection vulnerabilities of a web application. It is capable to taking over database servers. Using SQLMap an attacker or tester can perform database fingerprinting, execute commands on the underlying operating system, retrieve DBMS details, view or delete database data and even access the file system of the server.

Databases Supported with SQLMap

MySQL, Oracle, Microsoft SQL Server

PostgreSQL: PostgreSQL is an object-relational database management system which as a database server ,has a primary function to store data securely, and to allow retrieval at the request of other software applications.

Microsoft Access: Microsoft Access is a DBMS (also known as Database Management System) from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools.

IBM DB2: DB2 is a family of relational database management system (RDBMS) products from IBM that serve a number of different operating system platforms.

SQLite: SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.

SAP MaxDB: MaxDB is an ANSI SQL-92 (entry level) compliant relational database management system (RDBMS) from SAP AG. MaxDB is targeted for large SAP environments e.g. mySAP Business Suite and other applications that require enterprise-level database functionality.

SQLMap supports six SQL injection techniques: boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band. It is able to recognize hash format of passwords.

Detecting Vulnerability of a Website using SQLMAP

A vulnerable website provides free access to penetration testing such as SQLIAs and cross side scripting attacks. It thus becomes necessary to detect such vulnerabilities at an early stage so as to make it attack-proof. The following case study shows how an SQLIA is detected using SQLMAP.

For our testing purposes we have used a DVWA(Damn Vulnerable Website Application).

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications.[11]

The following is a step by step demonstration of detecting SQLIAs.

Step1:Open the Kali Linux command prompt and enter the following command.

Fig 2 : Command for checking vulnerability

The first half of the command contains the website URL on which SQLIA has to be detected, the further part contains the cookie value retrieved from the Advanced Cookie Manager .

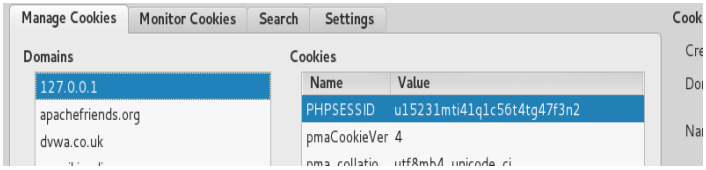

Fig 3: Location of the cookie

The above commands results into the following output which provides the backend details of the website.

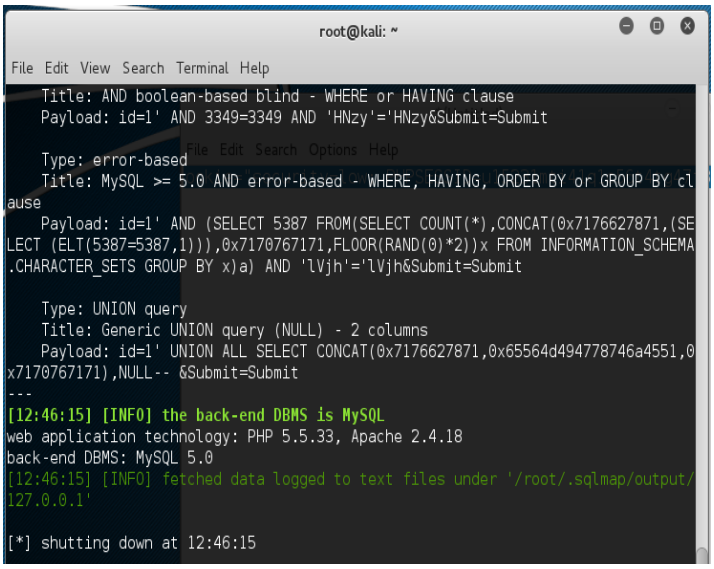

Fig 4 :Backend type revealed

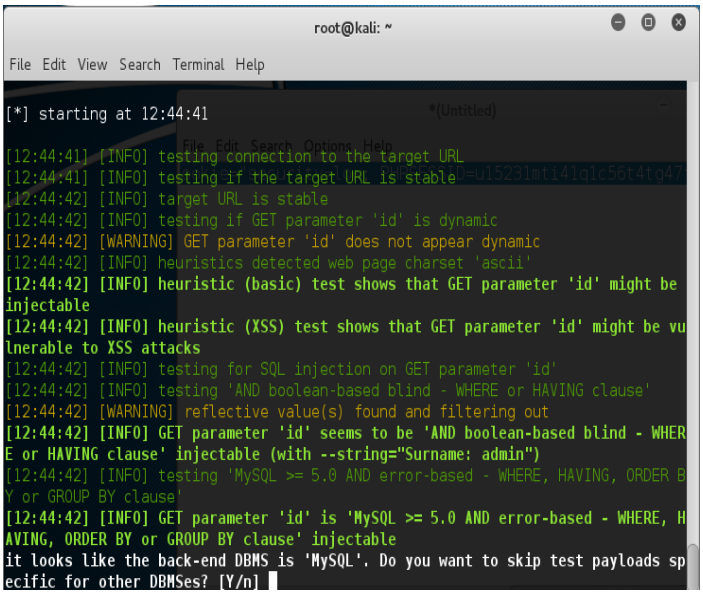

Fig 5:Parameter name that is injectable

Here ,the backend detected is MySQL,also the results show that the parameter “ID” is injectable . This shows that the website is vulnerable to SQLIA and thus a probable SQL Injection Attack is detected.

Step 2: SQLMAP test to detect if the tables of a website can be retrieved or not.

Using the following command on a website enables developers to check if their tables are prone to SQLIA attack.

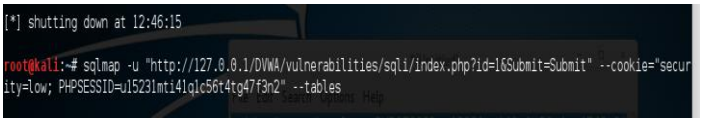

Fig 6:Command for retrieving tables .

Executing the preceding command results in the following output if the website is vulnerable.

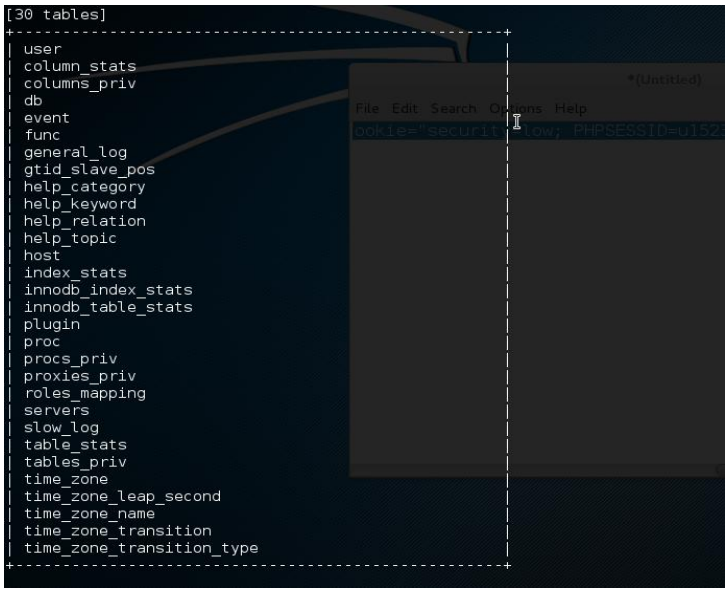

Fig 7: Tables in the Database

Step 3: Similarly, to retrieve columns of a database table the query can be modified as

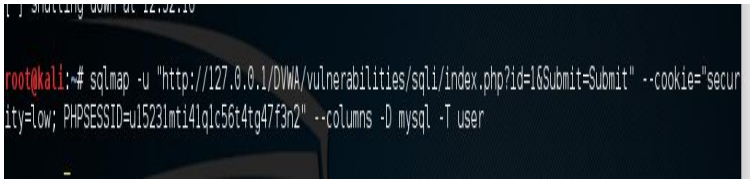

Fig 8: Command for retrieving columns

An attacker may retrieve all columns in a database using this which may lead to a loss of integrity and confidentiality of a website. If the website is a transactional one , the tables

may reveal all the data of the users which may include credit card details and pins triggering a credit card fraud.

The above command provides the following output where all the columns are revealed.

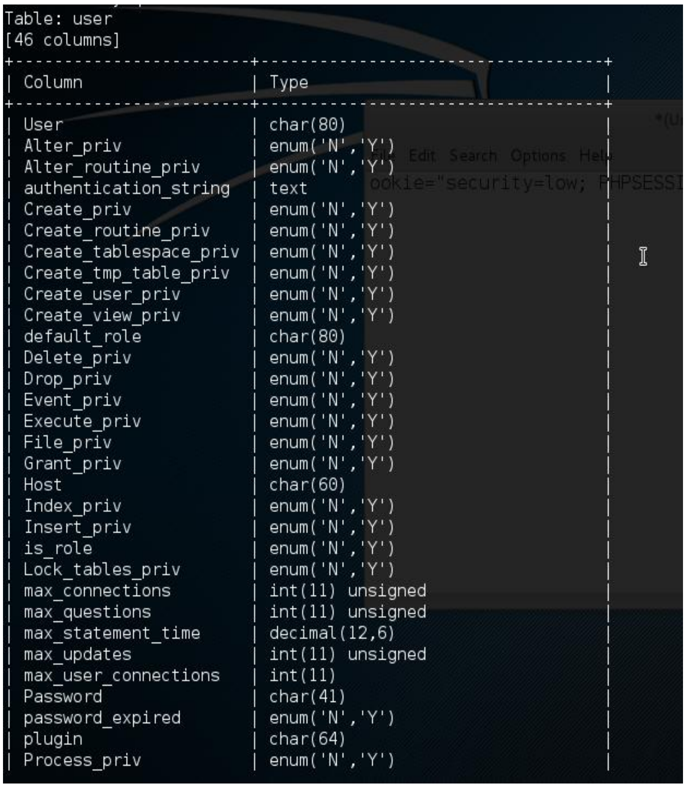

Fig 9: Columns Retrieved

The output shows the columns present in the user table. An attacker may choose any of the columns to alter the table.

Step 4: Fetching values of columns from the table. Using the following statements allow a user to retrieve the all the records of a table.

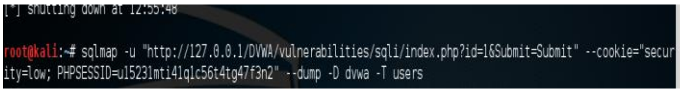

Fig 10: Command for fetching user details

The above statements reveal all user details

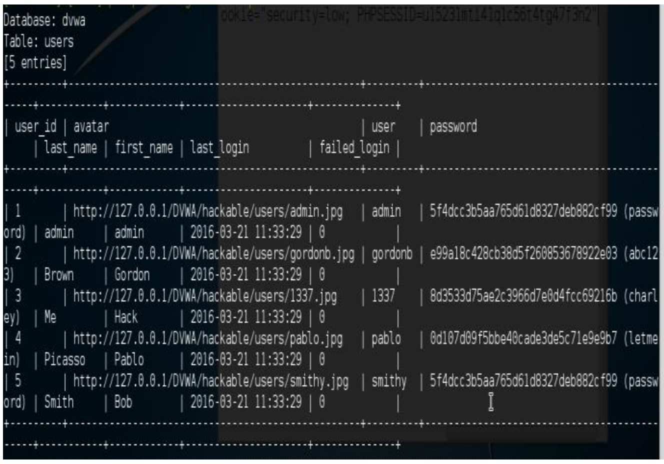

Fig 11: All user details retrieved.

The usernames along with the hashed passwords have been retrieved from the User table. A simple modification in the column record value causes a user to lose his credentials.

The records have been fetched as they were when the database of the website was populated.

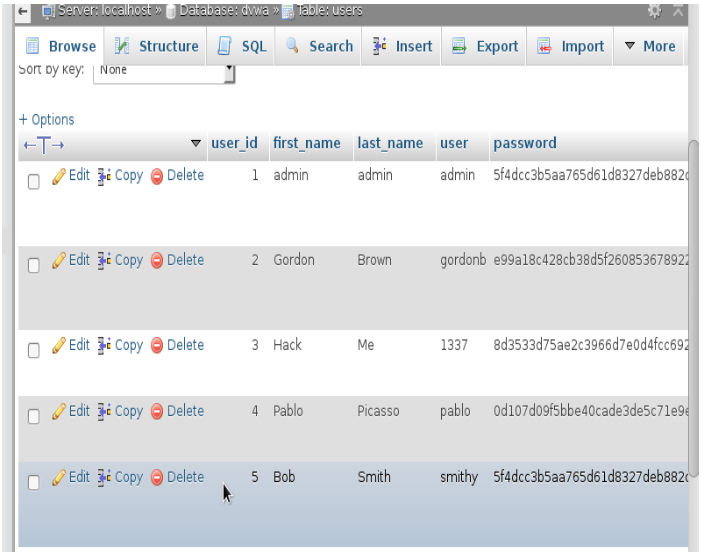

Fig 12: Table containing User details

Thus, an SQLIA can be detected using SQLMAP and the website applications can be validated for these attacks.

Example of a scenario where SQLIA will not be successful. Consider a website with strong validations . Using SQLMAP we detect whether it is SQLIA proof or not. Given that the website has strong validations, the following output is produced.

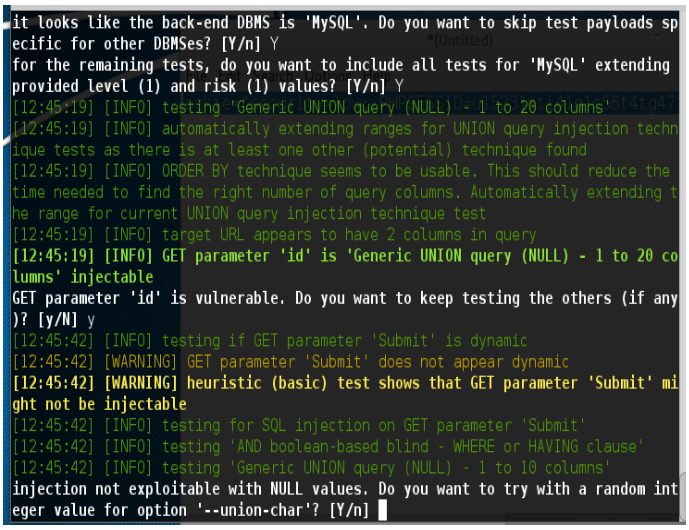

Fig 13: Example showing a parameter that is not injectable

The line clearly states that "Submit" parameter might not be injectable.
Thus building strong websites has become a need of the hour.

4. PREVENTING SQL INJECTION ATTACKS

WAVES: A black box technique, WAVES is efficient in testing web applications for SQL vulnerabilities. It identifies and locates all points in the web application that can be easily found to inject SQLIAs. During the attack phase WAVES, targets specific vulnerabilities as well as monitors the web application by machine learning.[8]

SQLPrevent: SQLPrevent is yet another tool that uses an HTTP request interceptor. When SQLPrevent is deployed into the web server the original data flow is modified. The HTTP requests are stored in a thread-local storage. The SQL interceptor captures the SQL statements and passes them to the detector module. Thereafter, the HTTP request from the thread local storage is tested for its contents of SQLIA. If the statement is infected it would not be sent to the database.[9]

SQLDOM and Safe Query Objects use the concept of query encapsulation that prevent untrusted access to databases. They use a type-checked API which uses systematic query building. Input filtering and strict user type checking is applied in addition to the API. The only reason the approaches fail is because the developer must learn new programming languages. [10]

JDBC Checker: Usually an attacker targets mismatches in a query string that can cause vulnerabilities in a database. JDBC checker is one such tool that prevents such techniques and bars them from gaining access to the database. This tool is capable of capturing SQLIAS that target queries that are not syntactically and type correct.

WebsSARI works on sanitized input that has is produced as a result of several predefined set of filters. It uses static analysis to analyze taint flows.

The only reason this tool proves disadvantageous is due to its requirement for adequate preconditions that cannot be expressed and the some filters are omitted.

Security Gateway is a proxy filtering system that strictly filters input patterns rules on the data going to a Web application. Thus, for transferring parameters from web pages to servers the developers use a Security Policy Descriptor Language (SPDL).Thus it proves useful for developers in deciding which data should be filtered and what patterns must act on the data.

5. CONCLUSION

This paper outlines the various methods for detecting and preventing SQL Injection Attacks. For dynamic web application, a system is proposed which can help detect SQLIAs. In addition, a tool SQLMap is outlined. This tool demonstrate the possible vulnerabilities in the web application by deliberately injecting an SQLIA and thus helps developers to detect the loopholes in the design and eradicating it, which provides security from SQL Injection Attack.

SQL Injections have been long studied and various tools have been proposed, each having their share of limitations. Penetration attacks have been prevalent in the web based domain, web developers can use the detection tools and prevention tools to secure their web applications from the attackers.

ACKNOWLEDGEMENT

The authors express their sincere gratitude to Prof. Aarti M Karande for her constant support and encouragement in carrying out the research work.

REFERENCES

[1] Piyush A. Sonewar, Nalini A. Mhetre "A Novel Approach for Detection of SQL Injection and Cross Site Scripting Attacks", International Conference on Pervasive Computing (ICPC)..

[2] Amith Pramod, Agneev Ghosh, Amal Mohan, Mohit Shrivastava and Dr. Rajashree Shettar,"SQLI Detection System for a safer Web Application",2015 IEEE International Advance Computing Conference (IACC)

[3] Atefeh Tajpour,Mohammad JorJor zade Shooshtari,"Evaluation of SQL Injection Detection and Prevention Techniques",2010 Second International Conference on Computational Intelligence, Communication Systems

[4] Bharti Nagpal,Naresh Chauhan,Nanhay Singh Angel Panesar,"Tool Based Implementation of SQL Injection for Penetration Testing",International Conference on Computing, Communication and Automation (ICCCA2015).

[5] Xiang Fu , Kai Qian. SAFELI–SQL Injection Scanner Using Symbolic Execution. Proceedings of the 2008 workshop on Testing, analysis, and verification of web services and applications. (2008). pp 34-39: ACM

[6] Sruthi Bandhakavi, Prithvi Bisht, P. Madhusudan,CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluations, 2007, Alexandria,Virginia, USA, ACM.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[7] Prithvi Bisht, P. Madhusudan. CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks. Proceedings of the 14th ACM Conference on Computer and Communications Security. 2007. USA: ACM, pp 1–38

[8] Y. Huang, S. Huang, T. Lin, and C. Tsai. Web Application Security Assessment by Fault Injection and Behavior Monitoring. In Proceedings of the 11th International World Wide Web Conference (WWW03), May 2003.

[9] P.Grazie., PhD SQLPrevent thesis. University of British Columbia (UBC) Vancouver, Canada.2008.

[10] R. McClure and I. Kr¨uger. SQL DOM: Compile Time Checking of Dynamic SQL Statements. In Proceedings of the 27th International Conference on Software Engineering (ICSE 05), pp 88–96, 2005.