Double Guard: Detecting SQL Injection Attack in Web Applications

R.Prakash
PG Scholar, Department of IT, SNS college of Technology, Coimbatore.
raman.prakash1@gmail.com

M.Suguna
Associative professor, Department of IT, SNS College of Technology, coimbatore
suguna.marappam@gmail.com

Dr.D.Sharmila
Professor & Head EIE Department, Bannari Amman Institute of technology, Sathyamangalam,
sharmiramesh@reddiffmail.com

Abstract: **Internet services and its applications have become an inextricable part of routine life, Establishing communication and the organizes personnel information from any part of the world web service application are increased day by day .it provides more flexibility and data complexity , web services have a multitier design and it follows to run the front end logic and data are stored in file server or database.**

Intruder can attack the server and front end by using sql query based attacks, SQL injection attack is the most severe thread in application and it has many types, By these attacks attacker can modify and delete most vital information after retrieve their needful information. To provide double guard prevention, it's a newly high automated approach to put off Sql injection attack based on the concept of novel idea of positive tainting and syntax aware evaluation. Here applied the Web Application Sql injection Preventer (WASP) tool technique which able to stop all sql injection attack and never generate false positive.
***Keywords*: Sql injection, Novel idea Approach, Positive Tainting**

I.INTRODUCTION

WEB applications are used by the Internet with help of using several acquiescent Web browser that runs on every operating system and structural design. They have suit ever-present due to the expediency, litheness, accessibility, and interoperability that they have provided. Web applications interconnect with databases that have more vital information such as employee names, position, salary, employee id number, and so on. Web sites erect SQL queries to contact these databases supported, in division, on user-provided input. The goal is that Web applications will bound the variety of queries that can be produced to a secure breaking up of all possible queries, in spite of what input users give. However, insufficient input legalization can allow attackers to expand inclusive contact to such databases. One approach in which this occurs that attackers can present input strings that hold particularly prearranged database commands. When the Web application put up a query by utilizing these strings and proposes the query to its essential database, the attacker's entrenched commands are accomplished by the database and the attack thrives.

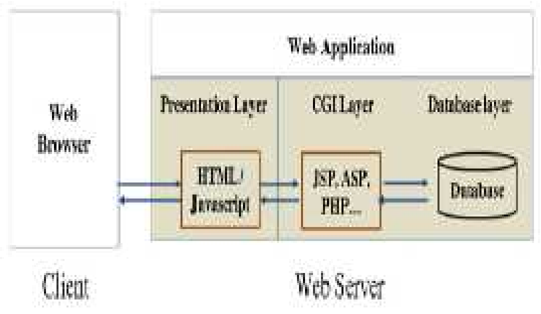


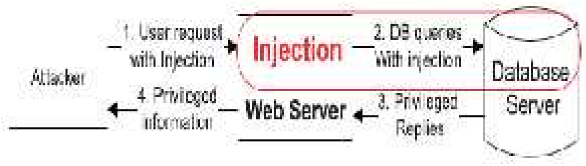
Fig 1: web architecture

The outcomes of these attacks are habitually unsuccessful and can range from revealing of perceptive data (for example, employers details) to the demolition of database information. In this paper, we propose a new vastly automated approach for dynamic detection and prevention of SQLI As. naturally, our method works by discovering "assurance" strings in an application and permitting merely these trusted strings to be utilized to make the semantically significant parts of a SQL query such as keywords or operators.

WASP tool is the important one of the prototype tools, it is implemented in the java and it contains two model and library. It performs well and gives more security to the web applications.

II.SQL ATTACK

Sql attack is the most important attacks and its has variety of attacks each one has done some different function, these attacks are given privilege escalation attack, hijack attack, injection attack, direct db attack, perform vulnerability, weak audit and Denial of service. One example of sql attack



1. User request with Injection
Injection
2. DB queries With injection
Attacker
Database Server
4. Privileged information
Web Server
3. Privileged Replies

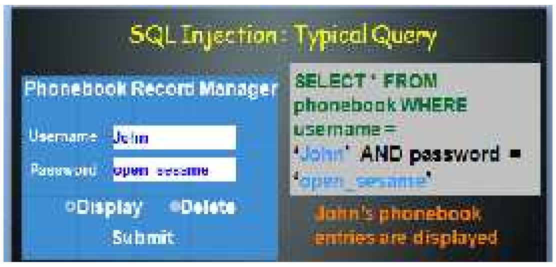
Fig 2 :Direct DB Attack

This diagram explain about direct db attack it is straightly vulnerable the database and collapse the original data. Here more attention given to the injection attack because it has many types of attacks .

III.SQL INJECTION ATTACK

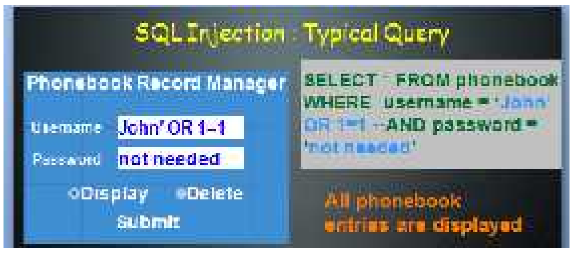
Sql injection attack is a one of the attack of sql attack, sql injection attack mean inject the malicious command with sql queries which can performed in a different way. Attackers give input in to application that moves to the database. Malicious command is integrated with sql query into the front end and accepts as a valid input, which is granting permission to access the database. Finally the databases carry out application command with the construed data and generate several result such as database sleaze, record deletion or operating system compromise

Example of injection attack



SELECT * FROM phonebook WHERE username = 'John' AND password = 'open_sesame'

Fig 3 : normal query



SELECT * FROM phonebook WHERE username = 'John' OR 1=1 --AND password = 'not needed'

Fig 4: Inject malformed query and attacked db

Above these figures are describes the injection basic attack and shows how to attack the database and mentioned how does it work.

IV.TYPES OF SQL INJECTION ATTACK

Sql injection attack has lots of types, these are poorly filtered strings, incorrect type handling, tautologies, union query, illegal /logical incorrect query, piggy backed queries, malformed encoding.

A) POORLY FILTERED STRINGS

This query call is used to select the password from the users , this sql injections pedestal on poorly filtered strings are sourced by user key that is not cleaned for break out typescript. This proceeds that a user be able to enter a key variable that may be accepted on as an SQL proclamation, ensuing in database input manipulation by the end user.

```
$pass = $_GET['pass'];$password =
mysql_query ("SELECT password FROM
users WHERE password = '". $pass . "';");
```

it is most helpful to achieve the sql operation that can be explained above the format and bellow the query also indicated

```
SELECT password FROM users WHERE
password = '' OR 1 = 1 /*
```

B) INCORRECT TYPES HANDLING

This types of sql injection attack happen while giving input not ensured the constraints type. For an example Id field is a numeric, but here filtering is not placed to check whether given input is numeric or not, numeric field only accept the number.

```
(is_numeric ($_GET['id'])) ? $id =
$_GET['id'] : $id = 1;$news=
mysql_query( "SELECT * FROM `news`
WHERE `id` = $id ORDER BY `id`
DESC LIMIT 0,3" );
```

C) TAUTOLOGIES

The main intention of tautologies is to inject vulnerable code in the provisional statements, these conditional always execute to true. This kinds of attack mainly depend on using query result in inside of application. Bypass authentication and extracting data most commonly used. In this method an invader utilize a susceptible input field that query's is utilized in the WHERE conditional.

" for entering input data only taken in the particular field other fields are irrelevant. The Example : An invader given " ' or 1=1 - resulting query is:

```
SELECT accounts FROM users WHERE
login='' or 1=1 -- AND pass='' AND
pin=
```

This provisional logic is enhanced the database inspects every row in the table. If the restrictive statements signify a tautology, the database equivalent and revisits all of the rows in the table as went up against to bout only one Row, as it would usually act in the deficiency of injection.

D) ILLEGAL/LOGICALL INCORRECT QUERIES

This type of attack to allow an attacker to gain the significant details about the pattern and formation of the back-end database in the internet applications. This attack has been considered
As a preliminary step to gaining details for variety of attacks, by this type of attack, vulnerability is influenced and server applications return the error page.

This type attacker main aim is to discover vulnerable parameter, extracting data. When an attacker attempt to infuse statements that affect the syntax structure, this type of fault can reduce data of the convinced column or expand the data. Reasonable error often discloses the name of table and column are affected by the error.

E) PIGGY BACKED QUERIES

it is an another one important type of sql injection attack, this attack intention is adding or modifying and extracting the data, this method an attacker can infuse the extra code with original queries. Here an attacker not changed anything in the original query, as an alternatively, attackers are including a new query with original after that piggy backed on the permanent query.

Query is normally executed that is the first aim, which query is the additionally infused query and it is very harmful, if attack is successful, can inject various type of sql command.

For an example:
SELECT accounts FROM users WHERE login='doe' AND
pass=''; drop table users -- ' AND pin=123

This sql rule is explained the piggy backed queries and attack of the announcement.

F) MALFORMED QUERIES

Union queries and previous queries permit attackers carry out precise queries or execute exact instructions on a database, but need a few former understanding of the database representation, which is regularly unidentified. Malformed queries let for conquering this trouble by captivating benefit of exceedingly vivid fault messages that are produced by the catalog when a malformed query is redundant. When these communications are straightly go backed to the user of the Web submission, as a substitute of being registered for repaired by developers, invaders can craft use of the restoring information to discover susceptible parameters and conclude the plan of the primary database. Invaders utilize this condition by infusing SQL check or waste input that basis the query to enclose syntax errors, type variances, or reasonable errors. regarding as our example, an assailant may perhaps endeavor causing a type disparity error by introducing the pursuing text into the pin input field:

Select acct from users where login='' and pin=convert(int,
Select top 1 name from sysobject where xtype='u'

The end resulting query causing by the Web application.

V.SQLIAPREVENTION ECHNIQUES

WASP: To appraise our approach, we enlarged a sample tool called WASP (Web Application SQL-injection Preventer). In this method using two concept of syntax aware evaluation and Positive tainting , these concept mostly supported to prevent the attack of sql injection, here we discuss about both techniques detail.

POSITIVE TAINTING: Positive tainting vary from conventional tainting (after this, negative tainting) for the reason that it is supported on the recognition, scratching, and tracking of hoped, fairly than untrusted, data. This theoretical disparity has important suggestions for the usefulness of our approach, here that it assists tackle problems caused by incompleteness in the classification of appropriate data to be noticeable. imperfectness, which is one of the main disputes when applying a safety method based on dynamic tainting, has extremely dissimilar consequences in negative and positive tainting. In negative tainting, deficient guide to trusting data that be supposed to not be trusted and, finally, to false negatives. Incompleteness may consequently depart the application susceptible to assaults and can be incredibly hard to detect, in positive tainting, imperfectness may direct to false positives, but it would not at all result in an SQLIA evading detection. furthermore, as clarified in the subsequent, the false positives engendered by our method , if some are possibly to be discovered and simply abolished early throughout prerelease testing. Positive tainting utilized a white-list, fairly than a black-list

SYNTAX-AWARE EVALUATION: Aside from make surely that taint markings are accurately formed and maintained through execution, our technique should be capable to use the taint

markings to discriminate legal from malicious queries. merely forbidding the utilize of untrusted data in SQL commands is not a feasible
Solution since it would pennant any query that holds user input as an SQLIA, leading to lots of false positives. To address this deficiency, researchers have introduced the perception of declassification, which allows the utilize of tainted input as long as it has been processed by a sanitizing function. (A sanitizing operation is naturally filters that execute operations such as normal expression matching or substring replacement.) The plan of declassification is stand on the statement that sanitizing operations are able to remove or neutralize harmful parts of the input and make the data safe.
Syntax-aware evaluation does not rely on some (probably unsafe) suppositions regarding the successfulness of sanitizing operations used by developers. It also permits for the utilize of untrusted input data in a SQL query as long as the utilize of such data does not root an SQLIA.

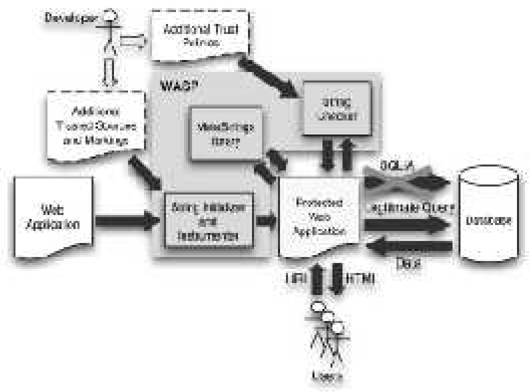

Developer
Additional Trust Policies
WASP
Additional Trusted Sources and Markings
MetaStrings library
String Checker
Web Application
String Initializer and Instrumenter
Protected Web Application
SQLIA
Legitimate Query
Database
Data
URL
HTML
Users

Fig 5 ; high level approach overview tools

Fig. explains the high-level structural design of WASP. As this shape illustrates, WASP consist of a stored (MetaStrings) and two interior part (STRING INITIALIZER AND NSTRUMENTER and STRING CHECKER). The MetaStrings libraries afford serviceable for allocating trust markings to strings and accurately spreading the markings at dynamic. component STRING INITIALIZER AND INSTRUMENTER instruments Web applications to facilitate the utilize of the MetaStrings library

VI.RESULT

Here we see many types of injection attack and retrieve the data from the database, according to the attack the needful information can get from the database by using Sql query. These attacks are given bellow
This diagram explain about the login user, here we have two type of user such as administrator and legitimate user, sometimes attacker can enter as a legal user and can theft the data from the database after retrieving they delete the vital information , these attacks are totally damaged the back end database, they enter the front end and attack the back end data

Fig 6: Attacker enter the front end

Fig 7: Attacker Retrieve Data from backend database

Both fig 6 &7 are shown the front and back end attack and describe about the attacks .

Wasp technique prevent the attack from the attacker and gives the protection to the web pages that given bellow the fig

Fig 8 : Wasp technique

By this method any attacker attempt to enter it can detect and protect from the attack.

VII.CONCLUSION

This paper present a narrated extremely automated move toward for guarding Web applications from SQLIAs. Our technique consists of 1)discovering the marking data that is coming from trusted source. 2)Ttrack trusted data at runtime by using dynamic tainting technique , and 3) only allowing trusted data to form the semantically significant parts of queries such as SQL keywords and operators. Unlike earlier methods based on dynamic tainting, but our approach is based on positive tainting, which clearly identifies trusted (rather than untrusted) data in a program. By this method, we remove the problem of false negatives that may possible result from the imperfect detection of all untrusted data sources. False positives, even though probable in some cases, can classically be easily removing through testing. Our method also afford realistic advantages over the lots of existing practices whose application involves customized and composite runtime environments

REFERENCE

1. Toward Automated Detection of Logic Vulnerabilities in Web Applications V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna, Proc. Usenix security symp 2010

2. Anomaly Detection of Web-Based Attacks .C. Kruegel and G. Vigna Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03), Oct. 2003

3. Database Intrusion Detection Using Weighted Sequence Mining. A. Srivastava, S. Sural, and A.K. MajumdarJ. Computers.vol. 1,no. 4, pp. 8-17, 2006

4. A novel method for SQL injection attack detection based on removing SQL query attribute values,Inyong Lee, Soonki Jeong, Sangsoo Yeo, Jongsub Moon,29 January 2011

5. WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation, William G.J. Halfond, Alessandro Orso, OL. 34, NO. 1, JANUARY/FEBRUARY 2008

6. A Classification of SQL Injection Attacksand Countermeasures,William G.J. Halfond, Jeremy Viegas, and Alessandro OrsoCollege of Computing Georgia Institute of Technology{whalfond|jeremyv|orso}@cc.gatech.edu

7. DoubleGuard: Detecting Intrusions in Multitier Web Applications, Meixing Le, Angelos Stavrou,Member, IEEE, and Brent ByungHoon Kang, 8.Member, VOL. 9, NO. 4, JULY/AUGUST 2012

8 . .C. Anley. Advanced SQL Injection In SQL Server Applications.White paper, Next Generation Security Software Ltd., 2002.

9 E. M. Fayo. Advanced SQL Injection in Oracle Databases. Technical report, Argeniss Information Security, Black Hat Briefings, Black Hat USA, 2005.

10 P. Finnigan. SQL Injection and Oracle - Parts 1 & 2. Technical Report, Security Focus, November 2002. http://securityfocus.com/infocus/1644 http://securityfocus.com/infocus/1646.

11 T. O. Foundation. Top Ten Most Critical Web Application Vulnerabilities, 2005. http: //www.owasp.org/documentation/topten.html.

12 C. Gould, Z. Su, and P. Devanbu. JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications. In Proceedings of the 26th International Conference on Software Engineering (ICSE 04) – Formal Demos, pages 697–698, 2004.

13 C. Gould, Z. Su, and P. Devanbu. Static Checking of Dynamically Generated Queries in Database Applications. In Proceedings of the 26th International Conference on Software Engineering (ICSE 04), pages 645–654, 2004.

1 4 N. W. Group. RFC 2616 – Hypertext Transfer Protocol – HTTP/1.1 Request for comments, The Internet Society, 1999.

15 V. Haldar, D. Chandra, and M. Franz. Dynamic Taint Propagation for Java. In Proceedings 21st Annual Computer Security

Prakash.R is a PG Scholar of Master of Technology in Information technology under Anna university, Chennai .TamilNadu, India.