International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-3, Issue-1, March 2013

# Efficient Solution for SQL Injection Attack Detection and Prevention

**Munqath H. Alattar S.P. Medhane**

***Abstract— SQL injection is the most common attack for web applications and widely used exploit by hackers all over the world. A malicious hacker can do a lot of harm if he wishes to. SQL injection is a security vulnerability that occurs in the database layers of an application. SQL injection is a technique to pass SQL code into interactive web applications that employ in database services. The employment of SQL Injection Attacks, can lead to the leak of confidential information such as credit card numbers, commercial information & table structure. The attackers can get the entire schema of the original database and also corrupt it. In this paper, we have proposed the Detection Model of SQL Injection Vulnerabilities and SQL Injection Mitigation Framework. These approaches are based on SQL Injection grammar to identify the SQL Injection vulnerabilities during software development and SQL Injection Attack on web-based applications.***

***Keywords— SQL Injection; Security Assessment; vulnerabilities; Pattern Matching, SQL Query.***

## I. INTRODUCTION

SQL Injection could be a variety of injection or attack in a very net application, during which the wrongdoer provides Structured Query language (SQL) code to a user input box of an internet kind to achieve unauthorized and unlimited access. The attacker's input is transmitted into Associate in Nursing SQL question in such the simplest way that it'll kind Associate in Nursing SQL code. It's classified joined of the top-10 2010 net application vulnerabilities veteran by net applications in step with OWASP

As shortly because the services of web are rising, all net applications are relied on the web. Example: on-line banking, university admissions, shopping, and numerous government activities. So, we can we will we are able to say that these activities are the key element of today's web Infrastructure. Net Applications are the applications which will be accessed over the web by victimization any applications program that runs on any software package and design. They need become omnipresent because of the convenience, flexibility, handiness, and ability that they supply. Net Applications are susceptible to a spread of recent security threats. SQLIAs are one in all the foremost vital of such threats. SQLIAs are increasing ceaselessly and bouquet terribly serious security risks as a result of they will offer attackers unrestricted access to the info that lie beneath net applications.

SQL injection could be a code injection technique that exploits a security vulnerability occurring within the info layer of Associate in Nursing application; it's wherever

Manuscript received on March, 2013.

**Munqath H. Alattar** Information Technology Department, College of Engineering, BharatiVidyapeeth University, Pune, India.

**Prof.S.P. Medhane,** Information Technology Department, College of Engineering, BharatiVidyapeeth University, Pune, India.

Associate in Nursing wrongdoer will trick a info server into running Associate in Nursing whimsical, unauthorized, unintentional SQL question by piggybacking further SQL components on prime of Associate in Nursing existing, predefined question that was meant to be dead by the appliance. The online application, that is usually, however not essentially, an internet application, this question is shipped to the application's info server wherever it's dead.

## II. LITERATURE REVIEW

Various techniques are projected for preventing SQL injection attacks:

**Boyd, Keromytis**-2004 projected SQL and that uses I instruction set organization of SQL statement to check SQL injection attack. It uses a proxy to a append key to SQL keyword. A de-randomizing proxy then converts the randomized question to correct SQL queries for the info. The secret is not renowned to the wrongdoer, that the code injected by wrongdoer is treated as undefined Keywords and expressions that cause runtime exceptions and therefore the question isn't sent to info. The disadvantage of this method is its complicated configuration and therefore the security of the key. If the secret is exposed, wrongdoer will formulate queries for winning attack. **Russell A. McClure and Ingolf H. Kruger**- 2005 projected SQL DOM (SQL Domain Object Model): a collection of categories that area unit strongly-typed to a info schema. It's supported compile time checking of dynamic SQL statements. Rather than string manipulation, these categories area unit accustomed generate SQL statements. We tend to show a way to extract the SQL DOM mechanically from Associate in existing info schema, demonstrate its relevance to unravel the issues, and valuate its performance.

**Ke Wei dynasty et al**.-2006 projected a completely unique technique to defend against the attacks targeted at hold on procedures. This method combines static application code analysis with runtime validation to eliminate the prevalence of such attacks. Within the static half, we tend to style a hold on procedure computer program, and for any SQL statement that depends on user inputs, we tend to use this computer program to instrument the required statements so as to check the initial SQL statement structure thereto as well as user inputs. The readying of this method is machine-controlled and used on a need-only basis. We tend to conjointly offer a preliminary analysis of the results of the technique projected, as performed on many hold on procedures within the SQL Server 2005 info.

**Cova, Balzarotti et al**.-2007 projected Associate in Nursing anomaly based mostly approach has for the detection of volition of net application. They use "Swaddler" for the analysis of the interior state of net applications and notice the connection between essential points and internal state. By doing this, the Saddler is ready to spot attacks that decide to

bring violation of the meant work flow of an online application.

**Mehdi Kiani et al**.-2008 delineate Associate in Nursing anomaly based mostly approach that utilizes the character distribution of bound sections of communications protocol requests to observe antecedently unseen SQL injection attacks. Our approach needs no user interaction, and no modification of, or access to, either the backend info or the ASCII text file of the net application itself. Our sensible results recommend that the model projected during this paper is superior to existing models at police investigation SQL injection attacks.

**R. Ezumalai and G. A**-2009 used a signature based mostly technique against SQL Injection Attacks. In this technique, they used 3 modules to observe security problems. A observation module that takes input from net Application and sent to analysis module. Associate in Nursing analysis module that finds out the hotspots from application, it uses Hirschberg algorithmic program. Hirschberg algorithmic program could be a string comparison algorithmic program that works on divide and conquer rule. It stores all the keywords within the specifications module.

**AnkitAnchlia and Sheela Jain**-2010 projected a completely unique approach to check the applications in a very comprehensive manner. The approach could be a holistic one; it tests the system beneath real conditions with none artifacts, to avoid potential injection attacks.

**STEP 1: the way to notice SQL INJECTION ATTACK?**

Solution: See fig. SQL Injection Detection Model

**Vulnerabilities Framework / Model**

An approach to style a model which can avoid SQL injection attack

A framework that analyze SQL Injection attack on net applications and info

**Automation Tools (To discover Vulnerabilities)**

A tool is employed to discover SQL Injection Attack loopholes.

To design a model for secure the system or forestall the system from SQL Injection attack that model additionally

Contain the all parameter that come back beneath the class of security policies.

**Analysis of Existing Vulnerabilities**

Examination of User Visible style Flaws.

Mapping existing Vulnerabilities to style call.

**Filter the vulnerabilities corresponding SQL Injection loopholes**

To design mechanism to filter all the vulnerabilities relating to SQL Injection loopholes

**Categorize of SQL Injection loopholes**

Specify the SQL question code similarly as sort that SQL injection attack is feasible.

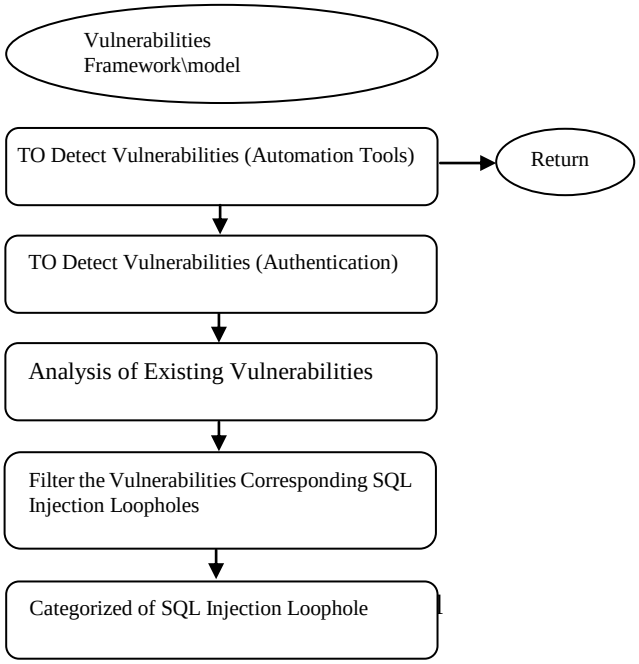


Figure 1: Steps of Attack Detection Framework

**STEP 2: HOW TO MITIGATE SQL INJECTION Attack At Software Andhardware Design Level?**

Solution: See fig. SQL Injection Mitigation Framework

**Security Policy Vulnerabilities:**

It is depend on two factors:

Security assessment framework at design level

Security policy based architecture refinement.

**Security Assessment Framework at Design Level**

A framework to assess the security at software design level

**Security Policy Based Architecture Refinement**

Flexibility to enhance update architecture

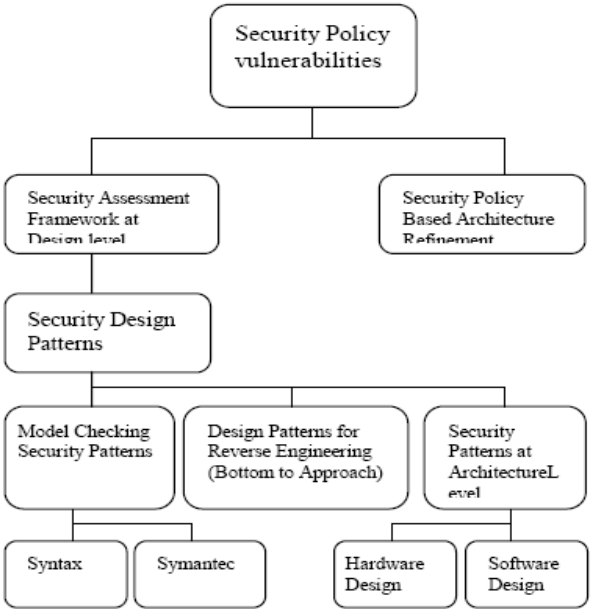


Fig 2: SQL Injection Mitigation Framework

**Security style Patterns**

Model checking security patterns analyze the structure of SQL question commands.

Build a program that may check allowable patterns of SQL statements.

Create a proxy server that may filter SQL commands.

Prevent a SQL injection attack to a info exploitation this proxy server.
Prove that SQL injection will be prevented exploitation the filter developed to figure on the proxy server.
Security patterns at design level

**Security Patterns at Architecture Level**

Implementation of computer code that is hardware freelance

**Syntax**

To design a model that follows rules and regulation that's outlined by security policies.

**Semantic**

Define solution mechanism that feels United States a way to follow the protection policies.

**Hardware style**

There is no probability of loopholes throughout hardware implementation.

**Software style**

No possibilities of loopholes throughout computer code style

**Types of attacks:**

**Tautology attacks:**

In Tautology-based attacks the most intention of the assaulter is to create the conditional statements that square measures continuously appraise to true. Assaulter largely uses the wherever clause of the question. Tautology attack is made once the assaulter is ready to come back all the records of the table or a minimum of is ready to come back one amongst the records from the information.

e.g. SELECT accounts FROM users WHERE
Login="or1=1--AND pass=" AND pin=

In this example the code injected in the conditional (or 1=1) will transfer the WHERE clause in to a tautology and the returned set evaluates to a value which will be not null, which results the application consider that the user authentication was successful.

**Logically incorrect question attacks:**

These varieties of attacks are primarily used for to grasp the structure of the info and therefore the variety of the backend databases. The error messages are useful for aggressor to grasp the structure and kind of the info used.

e.g. SELECT accounts FROM users WHERE login=" AND Pass=" AND pin=convert (int, (select top 1 name from Sysobjects where xtype=‟ u‟ ))

In this example firstly the query will try to extract the first user table that is xtype=‟ u‟ .After that the query will try to convert the table name into an integer. The database will give an error due to not a legal type conversion. If we are using the Microsoft SQL Server then the error will be like "Microsoft OLE DB provider for SQL Server (0x80040E07) Error converting nvarchar value "CreditCards‟ to column of data type int". The attacker is able to know that the database used is a Microsoft SQL Server database and secondly the value of the string cause the Type conversion to occur.

**Union Attack:**

In Union question the assailant uses the union operator. During this the assailant has the entire management of the second injected question, assailant will use that question to retrieve info from any desired table within the information by creating the guess of the table names. The results of the union attack are come within the style of dataset that is results of the mixture of the initial question and therefore the results of the second question that's union attack question.

e.g. SELECT accounts FROM users WHERE login="UNION
SELECT cardno from creditcards where
AcctNO=100 -- AND pass=" AND pin=

In this example there is no login whose value is equal to" ", the first query will return the null set of values, and the second query will return the data from the CreditCards table. The database will return "cardno" for the account "100".

**Piggybacked Query:**

In this attack the attacker tries to inject some extra types of queries in the original query, named as" piggy-back" This technique relies on the server configurations that allow the several different queries with a single string of code. The attacker uses the delimiter";" for this attack, he adds some extra queries after the delimiter and these queries are run on the database.

e.g. SELECT accounts FROM users WHERE login=‟ abc‟ AND
Pass="; drop table users –„AND pin=123

After completion of the first query the database would recognize the delimiter that is";" and lateral will continue execution and will try to drop the table users, if the table exists then it can destroy the information of that particular table of database.

## III. DETECTING SQL INJECTION

In order to shield an online application from SQL Injection attacks, there are two major considerations. Firstly, there's an excellent want of a mechanism to observe and specifically determine SQL Injection attacks. Secondly, information of SQL Injection Vulnerabilities (SQLIVs) could be securing an online application. So far, several frameworks are used andor steered to observe SQLIVs in net applications. Here, we tend to mention the outstanding solutions and their operating ways in short.

**SAFELI -** proposes a Static Analysis Framework in Order to observe SQL Injection Vulnerabilities. SAFELI framework aims at distinctive the SQL Injection attacks throughout the compile-time. This static analysis tool has two main benefits. Firstly, it wills a White-box Static Analysis and second, it uses a Hybrid-Constraint convergent thinker. For the White-box Static Analysis, the planned approach considers the byte-code and deals principally with strings. For the Hybrid-Constraint convergent thinker, the strategy implements associate degree economical string analysis tool that is ready to traumatize Boolean, number and string variables.

**Thomas et al.'s theme -** Thomas et al., in suggestion automated ready statement generation algorithmic rule to get rid of SQL Injection Vulnerabilities. They implement their analysis work mistreatment four open supplies come namely: (i) Net-trust, (ii) ITrust, (iii) WebGoat, and (iv) Roller. Supported the experimental results, their ready statement code was able to with success replace ninety four of the SQLIVs in four open supply comes.

**Ruse et al.'s Approach -** In, Ruse et al. propose technique that uses automatic action generation to observe SQL Injection Vulnerabilities. The most plans behind this framework are predicated on making a particular model that deals with SQL queries mechanically. Adding thereto, the approach identifies the connection (dependency) between sub-queries. supported the results, the methodology is shown to be able to specifically determine the causative set and

procure eighty fifth and sixty nine reduction severally whereas experimenting on few sample examples.

**Haixia and Zhihong's theme** - In, Haixia and Zhihong propose a secure information testing style for net applications. They counsel a couple of things; first off, detection of potential input points of SQL Injection; second, generation of take a look at cases mechanically then finally finding the information vulnerability by running the take a look at cases to form a simulation attack to associate degree application. The planned methodology is shown to be economical.

**Roichman and Gudes's theme** - suggests employing afine-grained access management to net databases. The authors develop a brand new methodology supported fine-grained access management mechanism. The access to the information is supervised and monitored by the intrinsic information access management. This is often an answer to the vulnerability of the SQL session traceability. Moreover, it's a framework applicable to most information applications.

**Shin et al.'s approach** –suggests SQLUnitGen, a Static-analysis-based tool that changes testing for Identifying input manipulation vulnerabilities. The authors apply SQLUnitGen tool that is compared with FindBugs, a static analysis tool. The planned mechanism is shown to be economical as respect to the very fact that false positive was fully absent within the experiments.

**SQL-IDS Approach** - Kemalis and Tzouramanis in Suggest employing a novel specification-based methodology for the detection of exploitations of SQL injection vulnerabilities. The planned query-specific detection allowed the system to perform targeted analysis at negligible process overhead while not manufacturing false positives or false negatives.

## IV. TECHNIQUES

**Real Time Based Positive Tainting**

As we studied in, positive Tainting is based on identification of the trusted data rather than untrusted data. Traditional Tainting (negative tainting) follows the identification of untrusted data and here positive and negative tainting differs. This conceptual difference has significant implications for the effectiveness of our approach, in that it helps address problems caused by incompleteness in the identification of relevant data to be marked. Incompleteness leaves the Web Application vulnerable to SQL injection attacks. In negative tainting detection of attacks is very difficult. Hence we use positive tainting in our approach. Identifying trusted data in Web Application is often straight forward and always less error prone. Here positive tainting will directly sense the real time traffic from set of input web applications.

**Accurate as well as Efficient Taint Propagation**

Taint Propagation is carried at runtime. It consists of identifying taint markings associated with data, while the data is used and manipulated by users at runtime. Taint Propagation needs to be carried out accurately otherwise it would cause the data to be misused. Our approach consists of:

1) Identifying taint markings at correct level of granularity

2) Precisely accounting for the affect of functions that operate on the tainted data.

The data consists of characters. Hence to achieve accuracy tainting at character level is carried in our approach. Here Strings are constantly broken into substrings for building SQL quires.

**Syntax Aware Evaluation**

Positive tainting helps to create taint markings during execution but for achieving more security we must be able to use the taint markings to distinguish legitimate from malicious queries.

The key feature of Syntax aware evaluation is that it considers the context in which trusted and untrusted data is to make sure that all parts of query other than string or numerical ,literals consists only of trusted.

## V. DRAWBACKS

**Draw backs of Defensive coding**

It is difficult to implement

It address only a subset of the possible attacks

The cost and complexity of retrofitting existing code

**Draw backs of Static Analysis**

Generate high rates of false positive

We can't find out vulnerabilities introduced at the run time

Time consuming, if conducted manually

**Draw backs of Traditional Tainting**

Incompleteness

Incompleteness leads to false negatives

Incompleteness may thus leave the application vulnerable to attacks

## VI. CONCLUSION

SQL injection attacks area unit a typical technique to attack on web-based applications. The attacker's area unit used SQL queries for assaultive and therefore these attacks reshape the SQL queries & thus neutering the behavior of the program for the advantage of the hacker. For determination this downside, we tend to project a SQL Injection Detection Model and SQL Injection Mitigation Framework to mitigate the SQL Injection Attacks (SQLIAs). Once mistreatment this potential resolution throughout software system development and once development, then we tend to might say that our net applications area unit secured from SQL Injection Attacks.

## REFERENCES


[1] William G.J. Halfond, Alessandro Orso, and PanagiotisManolios (2008): WASP: Protecting Positive Tainting and Syntax-Aware Evaluation .IEEE Transactions on Software Engineering, Vol. 34, No. 1

[2] Zhendong Su and Gary Wassermann (2006): The Essence of Command Injection Attacks in Web Applications. In ACM Symposium on Principles of Programming Languages (POPL)

[3] "top ten most critical web application vulnerabilities", OWASP Foundation, http://www.owasp.org/documentation/topten.html, 2005.

[4] S.V. Shanmughaneethi, S.C. E. Shyni, and S. Swamynathan (2009): SBSQLID: Securing Web Applications with Service Based SQL Injection Detection. IEEE Conference, Computer Society, pp. 702-704.

[5] H. Shahriar and M. Zulkernine (2008): MUSIC: Mutation-based SQL Injection Vulnerability Checking. The Eighth International Conference on Quality Software, IEEE Computer Society

[6] http://www.owasp.org/index.php/Top_10_2010-A1-Injection,retrieve on 13/01/2010

[7] K. Kemalis, and T. Tzouramanis (2008). SQL-IDS: A Specification-based Approach for SQLinjection Detection. SAC'08. Fortaleza, Ceará, Brazil, ACM: pp. 2153 2158.

[8] X. Fu, X. Lu, B. Pelts verger, S. Chen, K. Qian, and L.Tao. A Static Analysis Framework for Detecting SQL Injection Vulnerabilities, COMPSAC 2007, pp.87-96, 24-27 July 2007

[9] S. Thomas, L. Williams, and T. Xie, On automated preparedstatement generation to remove SQL injection vulnerabilities.Information and Software Technology 51, 589–598 (2009)

[10] M. Ruse, T. Sarkar and S. Basu .Analysis & Detection of SQLInjection Vulnerabilities via Automatic Test Case Generation of Programs. 10th Annual International Symposium on Applications and the Internet pp. 31 – 37 (2010)