International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 8, ISSN No. 2455-2143, Pages 23-29
Published Online June - July 2016 in IJEAST (http://www.ijeast.com)

STUDY ON SQL INJECTION ATTACKS: MODE, DETECTION AND PREVENTION

Subhranil Som
AIIT, Amity University Uttar
Pradesh, Noida, India

Sapna Sinha
AIIT, Amity University Uttar
Pradesh, Noida, India

Ritu Kataria
AIIT, Amity University Uttar
Pradesh, Noida, India

***Abstract:* Web applications are presently utilized for online administrations, for example: long range informal communication, shopping and managing accounts and so forth. Web applications deals with complex user information. Unauthorized access can lead to collapse of a system; even can harass the existence of a company or a bank or a branch. SQL Injection Attacks (SQLIA) is a standout amongst the most hazardous security dangers to Web applications. Researchers are working to control SQLIA at the application layer, but beforehand they are trying to prevent SQLIA at the database level through stored procedures. This paper shows ways to prevent SQLIA in stored procedures. The application is secured from attacks with the technology on two phases because if first phase is unable to protect then second phase can prevent attack.**

Keywords: **SQLIA, SQL Injection, Database Security, SQL Attacks and Prevention, Stored Procedures, Query Tokenization**

I. INTRODUCTION

SQL injection exposures have been communicated greatly unsafe for the database. Vital databases are absolutely accessible by attacker by injecting SQL queries that are retrieved by web application. As customer information is frequently kept in these databases, important information is lost and the security breach. Attackers can even use a SQL injection exposure is used by attackers for controlling and making the web application structure worse.

A class of code-injection attacks is pointed by SQL Injection; customer gives the data which is incorporated into a SQL query in such a way that part of the customer's information to be known by SQL codes. SQL commands given by attacker straight away to the database, through these vulnerabilities. These attacks are dangerous to any Web application that gets data from customers and goes along with it into SQL request to a key database.

This paper, weights on various parts of SQL Survey. This field related work done is exhibited in Section 2. Future investigation orientation to prevent SQLIA and a part of this paper determination and tokenization approach for resolution is contained in section 3. Section 4 presents SQLIA detection approaches and section 5 presents proposed technique. Finally, section 6 discusses conclusion and future work.

II. RELATED WORK

SQL Injection Attack Detection and Prevention Methods: A Critical Review *(Dr. Manju Kaushik et al. 2014*) suggests that by using SQLIA, an attacker can get these lines gain or adjust private/fragile information. There are beside no emphasis is laid on securing set away procedures in the database layer which could experience the bad impacts of SQLIA. As set away methods live on the database front, the procedures proposed by them can't be associated with secure set away frameworks themselves. They proposed a novel strategy to prepare for the assaults centered at set away philosophy. This system joins static application code examination with runtime acknowledgment to take out the occasion of such assaults. In the static part, they lay out a set away strategy parser, and for any SQL decree which depends on upon customer inputs, they use this parser to instrument the vital clarifications with a particular finished objective to differentiate the primary SQL verbalization structure with that including customer inputs. The course of action of this method can be automated and used on a need-simply introduce. [6]

Study of SQL Injection Attacks and Countermeasures *(Sayyed Mohammad et. al. 2013):* This paper gives scientific categorization of strategies to avert and recognize SQLIA. We characterize web application vulnerabilities and how they may bring about SQLIA. At that point, we show an order of SQLIA in view of weakness. A while later, the SQL injection isolation and three unique classes for counteractive action strategies. These distinctive methodologies in the time that balance to SQLIA plausibility. Various SQL recognition and aversion strategies are being talked about in this paper which as of late been proposed by a given attacker. Moreover, the systems were assessed, as for sending prerequisites [7].

A Survey of SQL Injection Countermeasures*(r R.P.Mahapatra et al. 2012):* The various SQLIA types were studied and the prevention techniques which the researcher proposed went under a survey in this paper [8].

Advanced SQL Injection in SQL Server Applications *(Chris Anley 2002):* This document discusses in detail the common 'SQL injection' technique, as it applies to the popular Microsoft Internet Information Server/Active Server Pages/SQL Server platform. It discusses the various ways in which SQL can be 'injected' into the application and addresses some of the data validation and the isolation issues of database identified with these attacks [12].
SQL Injection Attack *(J.makesh et al. 2015):* SQL injection is the database attacking not just attacks the database it have different budgetary result. We say a few methods in this paper and future work is to execute the firewall to the SQL server to maintain a strategic distance from the infusion assaults. A definitive goal for thoroughly destroy the entire idea of SQL injection and to stay away from this system turning into a toy in hands of exploiters. [15]
SQL Injection Attacks in Web Application *(Mihir Gandhi et al. 2013):* This paper exhibits the different diverse procedures of SQLIA. By utilizing these methods the software engineers and framework managers can comprehend the SQLIA all the more altogether and secure the web application from SQLIA. However as the innovation keeps on growing, so will the security dangers and systems utilized by pernicious clients. As the clients of the web move their delicate information into the online environment, it is essential that security be given the most striking in the improvement of web applications. [17]
SQL Injection Attacks: Techniques and Protection Mechanisms *(Nikita Patel et al 2011):* Code injection attack, particularly SQLIA is one of the scandalous issues. Controlling the pernicious SQL code/script on the web application and keeping up the end security is still a key test for the web engineer. Web designers included in creating sites should consider these issues utilizing databases. This paper depicts how an assailant can abuse the web application by utilizing SQL injection attack to get private data from a database. Diverse assurance systems against SQLIAare likewise proposed [18].

III. TYPES OF SQLIA

Tautologies

These kinds of attacks inject SQL tokens to the conditional query statement which are constantly assessed to be genuine. This type of attack uses WHERE clause to extract the valuable information from the input fields which are easily accessible that leads to the failed authenticity of control.

Illustration 1: Think about a web application which collects info through Customer, by means of the above SQL query, the aftereffect is as following.

Assume an attacker gives a name like this:

SELECT * FROM Customer WHERE name = ‘ritu’ OR ‘1’ = ‘1

This statement will give back all lines from the database of customer, instead of ‘ritu’ is a genuine customer name or not since OR is added to the WHERE clause. The result of ‘1’ = ‘1 comparison will be always ‘true’, and the resultant of WHERE clause assesses for all columns in the table to be genuine. On the off chance that this is utilized for validation purposes, the attacker will frequently login as a first or last customer in the table.

Logically Incorrect Queries

At the point when a query is not required, an incorrect text from the database, including required data is returned. These incorrect texts help attackers to find parameters in the application and in this manner the application's database. Without a doubt attackers garbage info or SQL token injected into query language structure mistake, to deliver logical error, syntax error, or type mismatches purposely.

Union Query

By this strategy, the attacker provides the incorrect data with the few correct fields, the SQL query is sent with the ‘Union’ of both correct and incorrect fields. As the result, the dataset from the database is fetched with the correct fields.
Illustration 4:

An attacker could inject the text “’ UNION SELECT card_No from Credit_Cards where acct_No=12450 --” into the login field, which produces the following query:

SELECT acc_inf FROM clients WHERE login=’’ UNION SELECT card_No FROM Credit_Cards WHERE acct_No=12450 -- AND pass=’’ AND pin=

In the first statement, the login is null, hence the query is invalid while the other query fetches the result. In the current situation, the field "Card_no" will fetch out for acc_No="12450". The consequences of the two queries are joined and returned as the output which will show the acct_No corresponding the credit card.

Piggy-backed Queries

In this sort of attack, with the existing query an attacker adds on extra queries and with this type of queries the attacker doesn’t changes the original query rather puts on a new query with the old one resulting into multiple SQL queries received by the database. Initially the existing query is implemented and the substitute query follows the already implemented query. This sort of attack can be exceptionally unsafe. In the event that effective, attackers can embed SQL query basically any sort, in substitute query, including stored procedures and alongside the first query they are executed. This sort of attack is regularly reliant on database designs that contain many queries in one string is the weakness.

Illustration 3: In the event that the input is provided by an attacker " '; DROP TABLE client - - " in the ‘pass’ field the application produces the query:

SELECT acc_No FROM client WHERE login = ‘ritu’ AND pass = ‘’; DROP TABLE client -- 'AND pin = 321
After going the first SQL query and detecting the delimiter (";") injected query is executed automatically by database, which results in losing the client useful information.

Stored Procedure

This process, Attackers pays attention to the stored procedures which are available in the database system. Database engine helps in the working of stored procedures. Stored procedure is just a piece of code which is exploitable. Stored procedure gives true or false values for the authorized or unauthorized clients. For SQLIA, attacker will write “; SHUTDOWN; --" with login or secret key. The below query will be produced by the stored procedure:

Illustration 5:

SELECT acc FROM client WHERE Login= '1231' AND Pass='9999 '; SHUTDOWN;--;

It works like piggyback attack. Firstly the existing query is processed subsequently followed by the other query which gets implemented and leads to shutting down of the database. This states that along with the web application code the stored procedure codes are equally exploitable.

IV. SQLIA DETECTION APPROACHES

Static Approach:

Software engineers give a few rules for SQLIA detection amid web application advancement and this methodology is otherwise called pre-creating approach. For the pre-created technique for identifying SQLIA a compelling legitimacy checking component is required for the info variable information.

Dynamic Approach:

Post-created methods are helpful for examination of element or SQL query on runtime, produced by client information by a web application and consequently this methodology is otherwise called post-created approach. Detection methods works under this post-produced class executing before presenting a query on the database server.

V. PROPOSED TECHNIQUE

In this paper, we have dealt with the security on the ends, i.e. frontend and backend, with no compromisation by proposing the two systems for avoiding SQLIA.

Its two stages are:
a) Frontend Phase
b) Backend Phase

Frontend Phase

Initially at front end we secure Database from any SQLIA. In this methodology we include an additional section in client table to store the Final Hash Code, which is obtained during enrollment time of a client for the first time and is put into client table along with client name and secret key as demonstrated in the Table - 1.

CLIENT	SECRET KEY	FINAL HASH_CODE
Ritu	Ritu09	12HJHOP34TTM
Geeta	Geeta123	21313NKIFIFN3

Table - 1: Client Table

When a client logins,, a genuine client is recognized by matching of client name, secret key and final hash code which was generated at run time using the stored procedure. For final hash code computation we will continue as indicated by the given architecture in the Figure – 1.

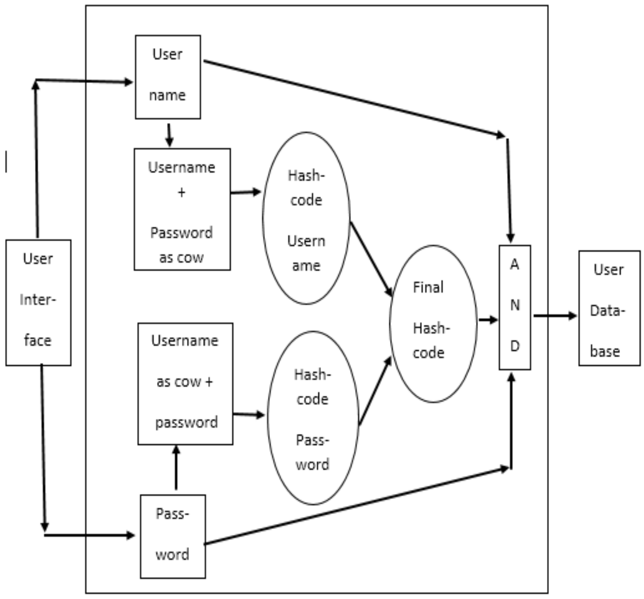

Figure 1: Proposed Architecture

Working Methodology

The working of proposed methodology is defined in two kinds:

1) New Client Registration:

A new client enters the log in details like distinctive name and secret key on client side to get registered. As indicated by the proposed design, the distinctive name and secret key is prepared at the center level.

Below are the stages:

1. To discover hash value of log in name by secret key as cow.
2. To discover hash value of secret key by log in name as cow.
3. Linking the result of step1 and step2 to discover final hash code.
4. Login name, Secret key and final hash code are to be put away into the client table.

2) Login and verification:

The login structure must be filled by the client to get signed into the database.

Below are the given stages:-

1. A distinctive name and secret key is to be entered at client side.
2. The name put away in client table is matched with the entered client name.
3. As per proposed system to discover final hash code at run time, the client name and secret key is handled after the client name is being matched.
4. Final hash code and secret word is checked with stored values in the database.
5. In the event that client is legitimate then he/she can get to data from database or else incorrect text is shown.

Figure - 2 demonstrates the three tier architecture of this working methodology.

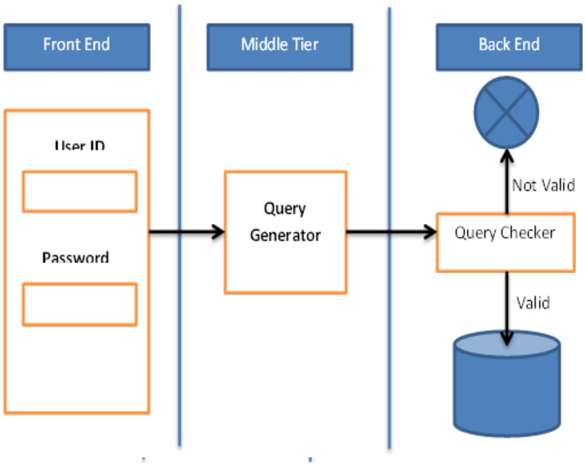

Figure 2: Three tiers proposed Architecture

Backend Phase

The proposed framework notices on how SQLIA on Web applications by tokenization and encryption for detection and prevention. The tokenization process changes over the input query in fruitful token and dynamic table stores it at the user end. Name of field, name of table and information are encoded by AES algorithm is connected by recognizing spaces on the data query, double dashes and single quotes, and so on. The initial encrypted query and table which is tokenized is being sent on the server side. Now the query is decrypted and generated into number of tokens which are then stored into other dynamic table at the server end. After comparing both the dynamic tables, if they are same then it is evaluated that there was no injected query, henceforth the query is carried to the central database for fetching the output. In the event that they are distinctive, query is dismissed and not sent to the server of the database. The incorrect text warning is sent to the client. Figure 4 demonstrates the proposed design of prevention of SQLIA.

I. AES Encryption or Decryption:

The data and attributes of the query are encoded by AES (Advanced Encryption Standard) algorithm needs less storage and this process is quick [14].

As soon as the query is received at server end, it gets decrypted with the similar key and gets transformed into different tokens which are kept in the other dynamic table.

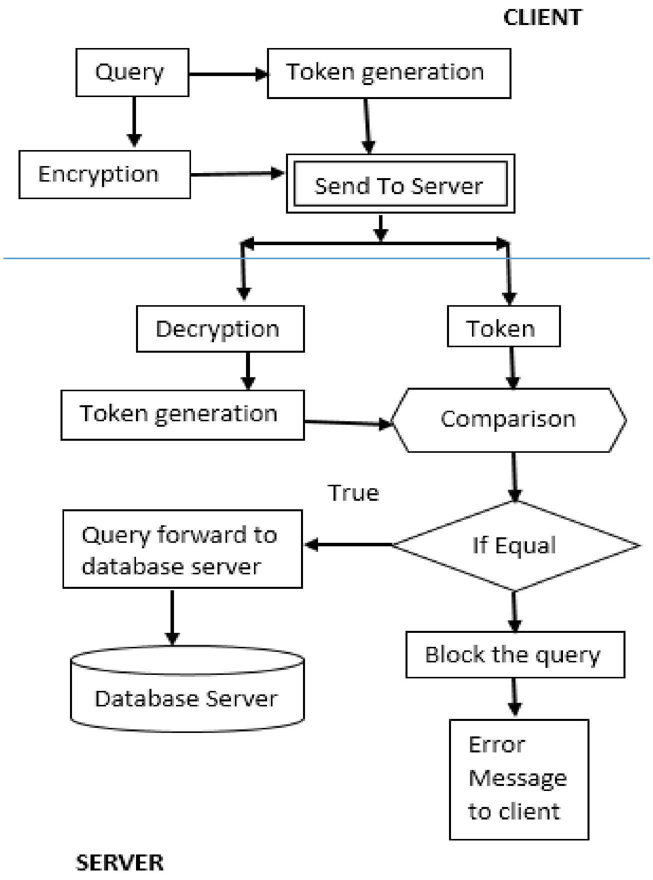

Figure 3: Proposed Architecture of SQLIA Prevention

II. Tokenizing the query:

In this strategy, the tokens are being created from query input given by the client. All strings before a space, a single quote, and double dashes constitute a token.
It is carried out in four stages:-
Step 1: All the unimportant characters are exchanged which could have attacked on the query.
Step 2: Identify the query with single quotes, spaces and, doubles dashes.
The Figure - 4 demonstrates how tokens are made in data query by recognizing single quote, spaces and double dash for the underneath query:

"SELECT e_id, e_name FROM Employee WHERE pay > 1000"

Step 3: The query is broken into different fruitful tokens.
Step 4: The Dynamic table keeps the tokens.
Step 5: Sending query after tokenization, the dynamic token table and the encrypted query are sent to server end

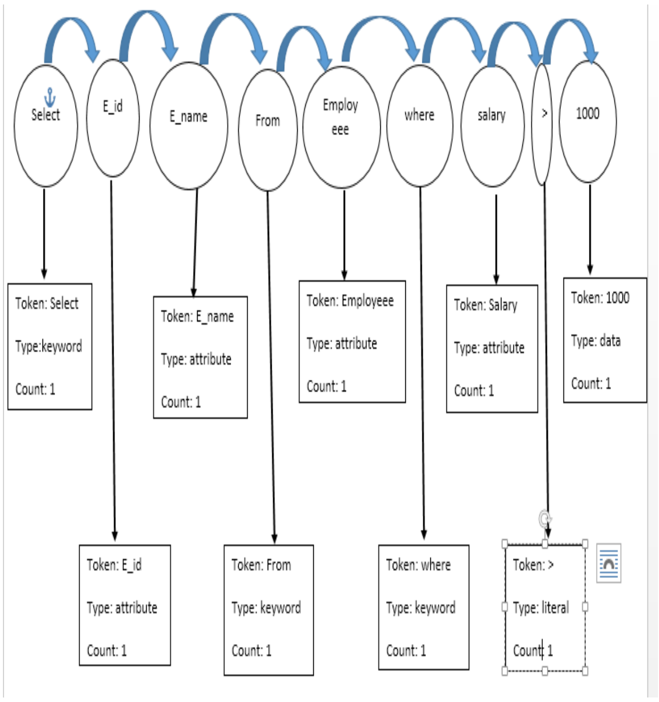

Figure 4: Formation of Tokens

III. Dynamic Tables comparison

Here, both Dynamic tables are matched with each other by their lengths. However if the length of both the dynamic tables are distinctive or regardless of the fact that all the events are distinctive, then injection hasn't attacked and query is forwarded to the database.
In any case, there is injection displayed in the query, when the tables are not of same size, fails to send query to main database. Attacked query gets cancelled and gives back the incorrect text to the user.

VI. CONCLUSIVE DISCUSSION AND FUTURE WORK

This paper has demonstrated a strategy to change over SQL query into number of helpful tokens by applying tokenization and after that encoding all literals, fields, table and information on the query by AES-algorithm to avoid SQLIA.
Our exploratory results demonstrate that a wide range of SQLIA can successfully be prevented by this methodology. It can likewise be effectively connected to some other dialect and database stage without significant changes.
This methodology encourages quick and proficient getting to system with database and keeps away from memory necessities to store the actual query in storehouse.
This methodology because of its low preparing overhead has immaterial impact on execution even at higher burden conditions and does not require real changes to application code.
Further study is done for making use of new algorithm to encrypt data query for preventing SQLIA, the query change plan is required.

VII. REFERENCES

[1] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso, (2006) “A Classification of SQL Injection Attacks and Countermeasures”, 2006 IEEE

[2] Ankita Kushwah, Gajendra Singh (2014) “Sql Injection Attacks: Prevention for All Types of Attacks”, International Journal of Emerging Engineering Research and Technology Volume 2, Issue 2, May 2014, PP 37-42

[3] Gregory T. Buehrer, Bruce W. Weide, and Paolo A. G. Sivilotti (2005) “Using Parse Tree Validation to Prevent SQL Injection Attacks” , 2005 ACM 1-59593-204-4/05/09

[4] Neha Mishra, Sunita Gond (2013) “Defenses To Protect Against SQL Injection Attacks”,International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 10, October 2013

[5] Parveen Sadotra (2015) “Hashing Technique - SQL Injection Attack Detection & Prevention”, International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 3, Issue 5, May 2015

[6] Dr.Manju Kaushik,Gazal Ojha (2014) “SQL Injection Attack Detection and Prevention Methods: A Critical Review” , International Journal of Innovative Research in Science,Engineering and Technology(An ISO 3297: 2007 Certified Organization) Vol. 3, Issue 4, April 2014

[7] Sayyed Mohammad Sadegh Sajjadi and Bahare Tajalli Pour (2013) "Study of SQL Injection Attacks and Countermeasures", International Journal of Computer and Communication Engineering, Vol. 2, No. 5, September 2013

[8] Dr R.P.Mahapatra and Mrs Subi Khan (2012) "A Survey of Sql Injection Countermeasures", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.3, June 2012

[9] Sammangi Gowtam Pratap Kumar, Akula Sai Chanukya ,Ayinavalli Venkata Ramana (2014) "Collaborative Technique to Detect and Prevent SQL Injection Attacks", International Journal of Advanced Computer Communications and Control Vol. 02, No. 02, April 2014

[10] Sruthi Bandhakavi, Prithvi Bisht, P. Madhusudan, V. N. Venkatakrishnan (2007) "CANDID: Preventing SQL Injection Attacks using Dynamic Candidate Evaluations", 2007 ACM 978-1-59593-703-2/07/0011

[11] Pooja Saini, Sarita (2015) "Survey and Comparative Analysis of SQL Injection Attacks, Detection and Prevention Techniques for Web Applications Security", International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169, Volume: 3 Issue: 64148 – 4153

[12] Chris Anley (2002) "Advanced SQL Injection in SQL Server Applications", An NGSSoftware Insight Security Research (NISR) Publication ©2002 Next Generation Security Software Ltd http://www.ngssoftware.com

[13] Tejinderdeep Singh Kalsi, Navjot Kaur (2015) "Detection And Prevention Of Sql Injection Attacks Using Novel Method In Web Applications",Kaur et al., International Journal of Advanced Engineering Technology E-ISSN 0976-3945

[14] Sonam Panda, Ramani (2013) "Protection of Web Application against Sql Injection Attacks", International Journal of Modern Engineering Research (IJMER) www.ijmer.com Vol.3, Issue.1, Jan-Feb. 2013 pp-166-168 ISSN: 2249-6645

[15] J.makesh, S.Thirunavukarasu (2015) "SQL Injection Attack", Special Issue of Engineering and Scientific International Journal (ESIJ) ISSN 2394-187(Online) Technical Seminar & Report Writing - Master of Computer Applications - S. A. Engineering College ISSN 2394-7179 (Print) (TSRW-MCA-SAEC) – May 2015 16

[16] Mihir Gandhi, JwalantBaria (2013) "SQL INJECTION Attacks in Web Application", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January 2013

[17] Nanhay Singh, Khushal Singh, Ram Shringar Raw (2012) "Analysis of Detection and Prevention of Various SQL Injection Attacks on Web Applications", International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 2– No.7, May 2012 – www.ijais.org

[18] Nikita Patel, Fahim Mohammed, Santosh Soni (2011) "SQL Injection Attacks: Techniques and Protection Mechanisms", International Journal on Computer Science and Engineering (IJCSE) ISSN: 0975-3397 Vol. 3 No. 1 Jan 2011 199-203

[19] Atefeh Tajpour , Suhaimi Ibrahim, Mohammad Sharifi (2012) "Web Application Security by SQL Injection DetectionTools", International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, March 2012 ISSN (Online): 1694-0814 www.IJCSI.org 332

[20] Shubham Srivastava, Rajeev Ranjan Kumar Tripathi (2012) "Attacks Due to SQL Injection & Their Prevention Method for Web-Application", International Journal of Computer Science and Information Technologies, Vol. 3 (2), 2012, 3615-3618

[21] Kamlesh Kumar Raghuvanshi , Deen Bandhu Dixit (2014) "Prevention and Detection Techniques for SQL Injection Attacks", International Journal of Computer Trends and Technology (IJCTT) – volume 12 number 3 – Jun 2014

[22] Manisha A. Bhagat, Prof. Vanita Mane (2013) "Protection of Web Application against Sql Injection Attack", International Journal of Scientific and Research Publications, Volume 3, Issue 10, October 2013 ISSN 2250-3153

[23] Bharti Nagpal, Naresh Chauhan, Nanhay Singh (2014) "Protection Of Web Application Against Sql Injection Attack",Injection And Prevention Of Sql Injection Attacks On Web Applications", IJSWS 14-393; © 2014

[24] Khaled Elshazly, Yasser Fouad, Mohamed Saleh, Adel Sewisy (2014) "A Survey of SQL Injection Attack Detection and Prevention", Journal of Computer and Communications, 2014, 2, 1-9, Published Online June 2014 in SciRes. http: // www.scirp.org / journal / jcchttp: // dx.doi.org /10.4236/jcc.2014.28001

[25] Ying Jin, Xiaoying Shen, Chunhui Song "A Filter-Based Approach for Sql Injection Attack Detection", https: // www.ecs.csus.edu / csc / iac / docs / publications/JIN_CATA12.pdf

[26] Etienne Janot, Pavol Zavarsky () "Preventing SQL Injections in Online Applications:Study, Recommendations and Java Solution Prototype Based on the SQL DOM", https: // www.owasp.org / images / 5 / 57 / OWASP - AppSecEU08-Janot.pdf

[27] Zhendong Su, Gary Wassermann (2006) "The Essence of Command Injection Attacks in Web Applications", POPL '06 January 11.13, 2006, Charleston, South Carolina, USA. Copyrightc 2006 ACM 1-59593-02702/06/0001

[28] Asha. N, M. Varun Kumar, Vaidhyanathan.G (2012) "Preventing SQL Injection Attacks", International

Journal of Computer Applications (0975 – 8887) Volume 52– No.13, August 2012

[29] Ms. Mira K. Sadar, Mr. Pritish A.Tijare, Mr. Swapnil N.Sawalkar (2014) " Securing Web Application against SQL Injection Attack: a Review", International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 2 Issue: 3 683 – 687

[30] Neha Singh, Ravindra Kumar Purwar(2012) "SQL INJECTIONS – A HAZARD TO WEB APPLICATIONS" ,International Journal of Advanced Research in Computer Science and Software Engineering Volume 2, Issue 6, June 2012 ISSN: 2277 128X

[31] Shelly Rohilla, Pradeep Kumar Mittal (2013) "Database Security by Preventing SQL Injection Attacks in Stored Procedures", International Journal of Advanced Research in Computer Science and Software Engineering ,Volume 3, Issue 11, November 2013 ISSN: 2277 128X

[32] Derrick Hyatt (2009) "Web 2.0 Injection Infection Vulnerability Class", ISSN: 1939-3555 (Print) 1939-3547 (Online) Journal homepage: http://tandfoline.com /Ioi/uiss20