Volume 6, Issue 12, December 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Review on SQL Injection Vulnerability and Comparative Analysis of Their Detection and Prevention Techniques

[1]Monali Sachin Kawalkar, [2]Dr. P. K. Butey
[1]Department of computer science, R.T.M. Nagpur University, Maharashtra, India
[2]Head of the Department, Department of Computer Science Kamla Nehru, R.T.M. Nagpur University, Maharashtra, India

Abstract- Because of digitalization and rapid growth in technology web applications are widely used like e-commerce, online payments, online banking, money transfer, social networking, etc. As web application interacts with database where critical information is stored over the network. The methodology used is Structure Query language (SQL) and Scripting language. It allows attackers to obtain unauthorized access to the back-end database to change the intended application generated SQL queries. As this information is very sensitive it is necessary to protect from unauthorized access. SQL injections is one of the many web attack mechanism used by hackers to steal such confidential data from organizations. SQL injection is a technique that exploits a security vulnerability occurring in the database layer of an application. The attack takes advantage of poor input validation in code and website administration. Hence Researchers have proposed various solutions to reduce SQL injection problems. But that solution is not enough to solve the problem. In addition the new types of attacks come up every year. To better counteract these attacks, identifying and understanding existing techniques are very important. In this research we describe SQL injection attack types also present and analysis of existing detection and prevention techniques against SQL injection attack and perform comparison analysis between the techniques.

Keywords-OWASP – Open Web Application Security Project (OWASP), SQL – Structure Query language, RDBMS – Relational Database Management System, Web Application, Detection and Prevention Techniques.

I. INTRODUCTION

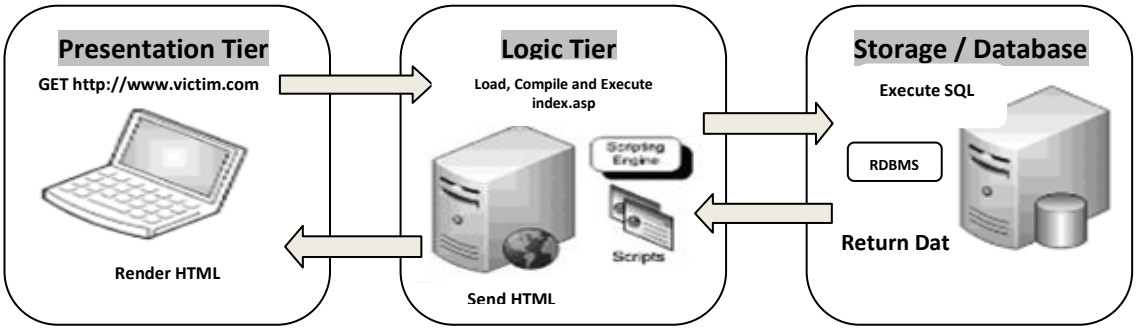
Web Applications are vulnerable to a variety of new security threats getting generated everyday by various sources, these applications which are hosted on Internet which is a widespread information infrastructure. Unaware of the Confidentiality, Integrity, Availability, Security and Privacy, the internet is becoming a repository of Business critical information. No matter which business, Information and data of organization is the most important business asset in today's environment, this can be achieved an appropriate level of Information security.

SQL Injection (SQLI) refers to an Injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server (also commonly referred to as a relational database Management System-RDBMS). Since an SQL injection vulnerability could possibly affect any website or web application that makes use of an SQL-based database, the vulnerability is one of the oldest, most prevalent and most dangerous of web application vulnerabilities. OWASP has released the latest version of "Top 10 Vulnerabilities" based on the previous incidents as well as on the risks associated with the Vulnerabilities. SQL Injection and Cross Site Scripting vulnerabilities are more prominent and harmful and have taken the highest Rank amongst the rest of the Top 10 OWASP Vulnerabilities. By leveraging SQL injection vulnerability, given the right circumstances, an attacker can use it to bypass a web application's authentication and authorization mechanisms and retrieves the contents of an entire database. SQL injection can also be used to add, modify and delete records in a database, affecting data integrity. To such an extent, SQL injection vulnerability can provide an attacker with unauthorized access to sensitive data including customer data, personally identifiable information (PII), trade secrets, intellectual property and other sensitive information. The information might be very confidential from Business perspective.

So, this paper consists of 5 parts. Part I shows Injection principles which include how SQL Injection works? Part II describes types of Attacks. Part III includes related work and Part IV provides comparative analysis between SQL injection detection and prevention techniques. Finally a brief conclusion of this paper is provided in Part V.

II. WHAT IS SQL INJECTION ATTACK?

SQL Injection is a technique employed by unauthorized user to gain access to the database content. Attacker would send specially framed input with SEL keywords embedded that would result in altering the semantic of the query. The concept of injection attack is to insert malicious code into a program so that result in change structure of SQL query. Such an attack may be performed by adding strings of malicious characters into data values in the form or argument values in the URL. Injection attacks generally take advantages of improper validation over input/output data. Most of the web application use multi tier architecture, usually with three tiers. First layer is Presentation layer also called front end which includes information related contents. Second layer is Application layer which includes functionality by performing processing logic and Third layer is data tier called back end consist of database server, keeps data storage.



Simple Three Tier Architecture

Figure-1

As shown in above Figure-1, The Web browser (presentation) sends a request to the middle tier (logic), which services them by making queries and updates against the database (storage).A fundamental rule in three-tier architecture is that the presentation tier never communicates directly with the data tier; in a three-tier model, all communication must pass through the middle wear tier. Conceptually, the three-tier architecture is linear.

The user fires up his web browser and connects to http://www.victim.com. The Web server that resides in the logic tier loads the script from the file system and passes it through its scripting engine, where it is parsed and executed. The script opens a connection to the storage tier using a database connector and executes an SQL statement against the database. The database returns the data to the database connector, which is passed to the scripting engine within the logic tier. The logic tier then implements any application or business logic rules before returning a Web page in HTML format to the user's Web browser within the presentation tier. The user's Web browser renders the HTML and presents the user with a graphical representation of the code. All of this happens in a matter of seconds and it's transparent to the user.[15]

Root cause, impact, identification techniques and counter measure for SQL injection

In this section we encounter various root cause of SQL injection these are:

Root cause –

- **Insufficient input validation:** Execution of any query that consist of INSERT, UPATE, ALTER and some SQL control character such as semicolon and quotation mark. If there is no checking for web applications so it can be abused in SQL injection.
- **Generic Access:** In privileges of accessing database some actions going to perform like, SELECT, INSERT, DELETE for execution of sql query.web application is use for accessing any specific information from database. If application is not secured then it may bypass authentication an attacker gain privileges.
- **User login credentials:** If the variable having uncontrollable variable size for storage of large amount of data so some time can be possible that attacker may enter faked input values.
- **Error Message:** when wrong input values is inserted in web application,error message generates Attacker may get the script structure or information about database .so that attacker may create its own attack.
- **Client side control:** This is actually happened in Cross site scripting. If input validation is implemented in client side scripts only using JavaScript, then by using cross site scripting security function of script at client side can be override and attacker can invalidate input or accessing database.
- **Stored Procedure:** Stored procedures are stored in databases. It is a small program with some function that calls multiple times in execution. When these functions become calls so that stored procedure become executes in place of that functions. The problem with the stored procedure is that an attacker can execute and damage database.

Impact –

- Non availability- By providing inputs like "I; shutdown", the attackers can force shutdown the database.
- Breach of confidentiality – By providing suitable inputs the attackers can view other user's records too.
- Similarly by framing the appropriate inputs, the attackers can also modify the records of other users, thereby breaching the integrity of application which is presumed to be secure and safe by the users.
- Impersonation- logging into accounts without the valid password is also possible thus attackers impersonating himself as one of the valid users of the application.

Identification techniques-

- Provide all possible SQL injections prone strings from the UI
- Check for the response

Counter measures

- Proper input validation on all the input parameters from the application

- Use of stored procedure instead of dynamic SQL queries, since the stored procedure can be protected through access control mechanisms
- Use of prepared statements with set methods to provide user input
- In such a case the input will be treated as string and not as an executable
- Grant only necessary privileges for accounts that are used to connect to DB, for example the use should not be given permission to shut down the database.

So, there are number of way a programmer can prevent the attacks made on their application. But this is not enough to find the vulnerability in sql injection. To prevent SQL injection vulnerability in application is problematic in practice. Therefore many researchers suggested a wide range of techniques which can be used in creating application of defensive coding.

III. INJECTION PRINCIPLES(HOW SQL INJECTION WORKS?)

The principle of basic SQL injection is to take advantage of insecure code on a system connected to the internet in order to pass commands directly to a database and to then take advantage of a poorly secured system to leverage an attacker's access. In order to run malicious SQL queries against a database server, an attacker must first find an input within the web application that is included inside of an SQL query. In order for an SQL injection attack to take place, the vulnerable website needs to directly include user input within an SQL statement. An attacker can then insert a data that will be included as a part of the SQL query and run against the database server.

The following diagram (Figure 2) shows how SQL injection done.

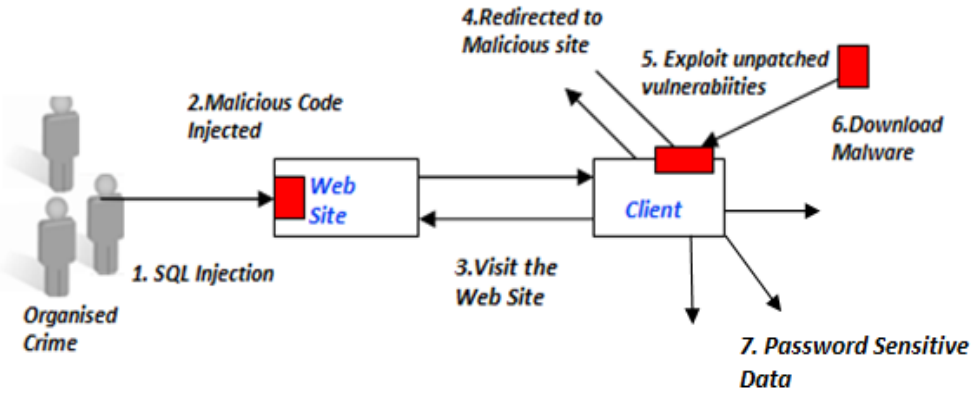


Figure-2

Step 1: The attacker attacks the Web Site using SQL injection. This involves information gathering queries followed by the specific attack that injects the link. The link is injected into string fields of the database so that dynamic web page content generated from the database will contain the link.
Step 2: Once the attack is successful, the attacker injects malicious code to the Web Site.
Step 3, 4 and 5: Users visit the website and are redirected to a malicious site which attacks the desktop using known vulnerabilities in operating system, browser and ActiveX plugins.
Step 6: Once successful, the attacker connects to various other sites to download malware (keylogger, password stealer etc)
Step 7: The attacker has sensitive information and has complete control of the desktop

The following server-side pseudo-code is used to authenticate users to the web applications.

```
# Define POST variables
uname = request.POST['username']
passwd = request.POST['password']
# SQL query vulnerable to SQLi
Sql = "SELECT id FROM users WHERE username=' " +uname + " ' AND password='" + passwd + " '"
# Execute the SQL statement
database.execute(sql)
```

The above script is a simple example of authenticating a user with a username and a password against a database with a table named users, and a username and password column.

The above script is vulnerable to SQL injection because an attacker could submit malicious input in such a way that would alter the SQL statement being executed by the database server.

A simple example of an SQL injection payload could be something as simple as setting the password field to password' OR 1=1

This would result in the following SQL query being run against the database server.

SELECT id **FROM** users **WHERE** username='username' **AND password='password' OR** 1=1"

An attacker can also comment out the rest of the SQL statement to control the execution of the SQL query further.

-- MySQl, MSSQL, Oracle, PostgneSQL, SQLite

OR '1' ='1' –

OR '1' = '1' /*
-- MySQL
OR '1' = '1' #
--Access (using null characters)
OR '1' ='1' %00
OR '1'= '1' %16

Once the query executes, the result is returned to the application to be processed, resulting in an authentication bypass. In the event of authentication bypass being possible, the applications will most likely log the attacker in with the first account from the query result – the first account in a database is usually of administrative users. Such many methods likes above example are included in modes of attack. It is likes hacking of the query string, breaking the query string, database foot printing, adding unauthorized data.

Why SQL Injection Is A Problem?

SQL is a programming language designed for managing data stored in RDBMS, therefore SQL can be used to access, modify and delete data. Furthermore, in specific cases, an RDBMS could also run commands on the operating system from an SQL statement. Keeping the above in mind, when considering the following, it's easier to understand how lucrative a successful SQL injection attack can be for an attacker.

- An attacker can use SQL injection to bypass authentication or even impersonate specific users.
- One of SQL's primary functions is to select data based on a query and output the result of that query.SQL injection vulnerability could allow the complete disclosure of data residing on a database server.
- Since web adding applications use SQL to alter data within a database, an attacker could use SQL injection to alter data stored in database. Altering data affects data integrity and could cause repudiation issues, for instance such as voiding transactions, altering balances and other records.

SQL is used to delete records from a database. An attacker could use an SQL injection vulnerability to delete data from database. Even if an appropriate backup strategy is employed, deletion of data could affect an application's availability until the database is restored. Some database servers are configured (intentional or otherwise) to allow arbitrary execution of operating system commands on the database server. Given the right conditions, an attacker could use SQL injection as the initial vector in an attack of an internal network that sits behind a firewall.

IV. TYPES OF SQL INJECTION ATTACKS

There are various types of attacks performed by attacker, some of them are describe as follows:

Tautologies [16][14]: This type of attack includes conditional statements means attacker injects code in conditional part such that query execute normally. In tautologies, the attacker bypass user authentication to extract data from database.. In this type of injection, the attacker exploits an injectable field contained in the WHERE clause of query.

The outcome of this query gives all the rows in the database for particular action.

Logically incorrect query attacks [17]: This category of attack is happened when it entering some inputs by which it generates an illegal or the logically incorrect queries. The error messages reveal the names of the tables and the columns that cause error. The attacker also comes to know about the application database used in the backend server.

Union Query [14]: In this type of attack, attacker join injected query to the safe query by using database keyword UNION. This category of attack mainly uses to extract data from the table, which is different from the one that was intended in web application by the developer. This technique is mainly used to bypass authentication and extract data.

Stored Procedures [14]: In this type of attack, attackers try to execute SQL procedures which are stored previously by the web application developer.

Piggy-Backed Queries [14]: In this type of attack, an attacker tries to inject additional queries along with the original query as like UNION query result in modify data or add data or drop table. This attack mostly use for damaging table content.

Alternate Encodings: In this attack attackers modify the query by using alternate encoding, such as hexadecimal, ASCII, and Unicode. Encodings is 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.

Because of this attack, attacker can escape from developer's filters which scan input queries.

Inference [14]: Inference means to change the behavior of a database or application. There are two attack techniques that are included in inference one is blind injection and another is timing attacks.

Blind Injection [18]: At the time of creation of application in some condition developers hide the error details and create a generic page in place of an error message so this time attacker can get the information of database .An attacker can extract data by asking a series of True/False questions through SQL statements.

Timing Attacks [18]: When the creation of application it uses an if-then statement at that time Timing attack is happened. Timing attack is related to response time given by database. Such case **WAITFOR** keyword is used for delay response by database. An attacker can gather information from a database by timing delays. In this technique if-then statement is used for injecting queries.

V. RELATED WORK

Various detection and prevention methods are being research in SQL injection attacks. This section describes the related work.

(A) Detection techniques for SQL Injection Attacks-

AMNESIA(Analysis and Monitoring for NEutralizing SQL Injection Attacks)

This is the most relevant detection technique proposed by Halfond et. al. in [19], author suggested that AMNESIA is the effective SQLIAs detection tool. It is a model base technique which combines both static analysis and dynamic analysis for preventing and detecting web application vulnerability at run time. Static phase is used to generate different type of query statements. Dynamic phase is used to interpret all queries before they are submitted to the database and validates each query against the statically built query models. AMNESIA technique stops all queries before they are sent to the database and validates each query statement against the AMNESIA models. Queries that violate the model represent potential SQLIAs and are thus prevented from executing on the database.

SQLGAURD [14][21]-In User input base model SQLGAURD method is useful. This method checked at runtime which is expressed as grammar that only accept legal queries. SQL Guard checks the structure of the query before and after the addition of user-input based on the model. In this approach developer should modify code to use a special intermediate library or manually insert special markers into the code where user input is added to a dynamically generated query.

SQLChecker[14] – This model uses a secret key at runtime checking so security of the approach is dependent on attackers. In SQL Check, the model is specified independently by the developer. It is similar to SQLGaurd by which

Model checks as a grammar that only accept legal queries. In this case developer should have to modify code to use a special intermediate library or manually insert special markers into the code where user input is added to a dynamically generates query.

Tautology Checker [18] – This method provides an analysis framework for security. It is a static analysis and automated reasoning performs for checking any tautology statement contains in coding. The major drawback of this tool is, it having limited scope. So this tool it is not useful as much.

CANDID [18] – In java programming CANDID tool is use as dynamically for program transformation. This tool dynamically mines the programmer-intended query structure on any input and detects attacks by comparing it against the structure of the actual query issued. CANDID's natural and simple approach turns out to be very powerful for detection of SQL injection attacks.

SQL-IDS [14] - Machine learning technique includes this method. It builds a model of typical queries and matches at run time that queries that does not match with original query treat as attacks. This technique detects attacks successfully, but it depends on training seriously.

SQL Prevent [14]-This technique is consists of an HTTP request interceptor. When the original data flow is modified SQL Prevent technique is deployed into a web server. The HTTP requests are saved into the current thread of local storage. Then, SQL interceptor intercepts the SQL statements that are made by web application and pass them to the SQLIA detector module. Consequently, HTTP request from thread-local storage is fetched and checks to determine whether it contains an SQLIA. The malicious SQL statement would be prevented to be submitted to database, if it contains malicious data.

SQLRand [24] - According to the Keromytis and Boyd in SQLRand Proxy server is used between Client (Web server) and SQL server. They de-randomized queries received from the client and sent the request to the server. Portability and security are the advantages of this de-randomization framework.

(B) Prevention techniques for SQL Injection Attacks-

WAVES [25] - WAVES a black-box technique for testing web application for SQL injection vulnerability. This technique uses a Web crawler to identify all points in a Web application that can be used to inject SQLIAs. It target on a specified list of patterns and attack techniques. WAVES then monitors the application's response to the attacks and uses machine learning techniques to improve its attack methodology.

JDBC Checker [21] –This technique is based on static analysis of web application that can reduce SQL injection vulnerabilities and detect type errors. It is uses for dynamically generated query string on basis of mismatching. As we know that most of the SQLIAs consist of syntactically and type correct queries so this technique would not catch more general forms of attacks.

SECURITYFly [14]-This is implemented for java. As compare to other tool this checks string in place of character for any suspicious information and try to sanitize query strings. This tool has a drawback that is numeric fields cannot stop by this approach. Difficulty of identifying all sources of user input is the main limitation of this approach.

SECURITY GATWAY [18] - It technique base on the filtering system that forces the input validation. By using Security Policy Descriptor Language (SPDL), developers provided specify transformation that is applied to the parameters of web application.

SQL DOM [14] - It is an object model for proposing a solution for building a secure communication environment for accessing relational databases from the OOP (Object-Oriented Programming) Languages. Due to this they mainly focus on identifying the obstacles in the interaction with the database via Call Level Interfaces.

WebSSARI [14]-Use for sanitizing input that passed through predefined set of filters. In this case static analysis to check taint flows against preconditions for sensitive functions. The drawback of this approach is that it is not necessary preconditions for sensitive function accurately expressed.

Detection and prevention techniques for SQL injection presented in Part III .Now this would be compared in this section.

VI. COMPARATIVE ANALYSIS

Comparison of SQL Injection Detection Techniques With Respect to Attack Types

Detection techniques are techniques that detect attacks mostly at runtime. Table 1 shows Comparison of SQL injection detection techniques with respect to attack types. The symbol √ is used for techniques that can successfully detect all attacks of that type. The symbol × is used for techniques that is not able detect all attacks of that type.

Table 1. Comparison of SQL injection detection techniques with respect to attack types

Attacks / Tools	Tautology	Piggy Baked	Ileagal Incorrect	Union	Alternate encoding	Timing attack	Blind attack	Stored procedure
ANMESIA	√	√	√	√	√	√	√	x
SQL GUARD	√	√	√	√	√	√	√	x
SQL Checker	√	√	√	√	√	√	√	√
TAUTOLOGY Checker	√	x	x	x	x	x	x	x
CANDID	√	x	x	x	x	x	x	x
SQL IDS	√	√	√	√	√	√	√	√
SQL Prevent	√	√	√	√	√	√	√	√
SQL RAND	√	√	√	√	√	√	√	x

Comparison of SQL Injection Prevention Techniques With Respect to Attack Types

Prevention techniques are techniques that statistically identify vulnerabilities in the code. Table 2 shows Comparison of SQL injection prevention techniques with respect to attack types. Symbol ○ shows techniques that detect the attack type only partially because of natural limitations of underlying approach.

Table 2. Comparison of SQL injection preventiontechniques with respect to attack types

Attacks / Tools	Tautology	Piggy Baked	Ileagal Incorrect	Union	Alternate encoding	Timing attack	Blind attack	Stored procedure
WAVES	o	o	o	o	o	o	o	o
JDBC Checker	o	o	o	o	o	o	o	o
SECURITY Fly	o	o	o	o	o	o	o	o
SECURITY Gateway	o	o	o	o	o	o	o	o
SQL DOM	√	√	√	√	√	√	√	x
WebSAARI	√	√	√	√	√	√	√	√

Comparison of detection or prevention techniques based on deployment and evaluation criteria.

Based on review the detection and prevention techniques can mostly detect SQLAs at runtime. Table 3 summarized Comparison of detection or prevention techniques based on deployment and evaluation criteria.

Table 3.Comparison of detection or prevention techniques based on deployment and evaluation criteria.

Techniques	Detection time	Detection Location	Code Modify
ANMESIA	Run Time	Server Side Application	No
SQL GUARD	Run Time	Server Side Application	Yes
SQL Checker	Run Time	Server Side	No
TAUTOLOGY Checker	Run Time	Server Side Application	No
CANDID	Run Time	Server Side Application	Yes
SQL IDS	Run Time	Server Side Application	No
SQL Prevent	Run Time	Server Side Application	No
SQL RAND	Run Time	Server Side Proxy	Yes
WAVES	Testing Time	Client Side Application	No
JDBC Checker	Coding Time	Server Side Application	No
SECURITY Fly	Run Time	Server Side Application	No
SECURITY Gateway	Run Time	Server Side proxy	No
SQL DOM	Coding Time	Server Side Application	Yes
WebSAARI	Run Time	Server Side Application	No

VII. CONCLUSIONS

SQL injection is a threat to any system connected to the internet with a database backend. An intelligent attacker will find any holes in your code and/or your server security and leverage them to their maximum advantage. An SQL injection is relatively easy to guard against, and should be a consideration of any project before a line of code is written, or a network card plugged it. In this paper, we have discussed a review on various types of SQL injection attacks, vulnerabilities and detection and prevention techniques. To perform this we first discuss about SQL injection then how it works and its root cause. Then we identified the various types of SQL injection attacks. Finally we investigated SQL injection detection and prevention techniques and its comparisons (Please refer Table1, Table 2 and Table 3). Our future work will be to find the best technique among the various methods for their evaluation.

REFERENCES

SQL Injection is not a dark art. A wealth of information is available on the internet regarding how to use it and how to protect against it.

[1] Open Web Application Security Project, "OWASP TOP Project", https://www.owasp.org/SQL_Injection.

[2] "SQL Injection Are Your Web Applications Vulnerable?". SPI Dynamics.2002.http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf

[3] Www.OWASP.org/index.php/SQL_Injection_Prevention_Cheat_sheet

[4] OWASP Top Ten Project - http://www.owasp.org/index.php/Top_10

[5] OWASP Code Review Guide - http://www.owasp.org/index.php/Category:OWASP_Code_Review_Project

[6] OWASP Testing Guide - http://www.owasp.org/index.php/Testing_Guide

[8] OWASP Enterprise Security API (ESAPI) Project - http://www.owasp.org/index.php/ESAPI

[9] OWASP Legal Project - http://www.owasp.org/index.php/Category:OWASP_Legal_Project

[10] http://www.ngssoftware.com/papers/advanced_sql_injection.pdf

[11] http://www.asptoday.com/content/articles/20020225.asp 25 Feb

[12] http://www.wiretrip.net/rfp/p/doc.asp?id=42&iface=6 27 Feb. 2001.

[13] "SQL Injection/Insertion Attacks". insecure.org.

[14] Sh. Bojken, A. Shqiponja, A. Marin, and Xh. Aleksander,"Protection of Personal Data in Information Systems",International Journal of Computer Science, Vol. 10, No. 2,July 2013, ISSN (Online): 1694-0784.

[15] SQL Injection Attacks and Defense(Book) - Justin Clarke

[16] Hussein AlNabulsi, Izzat Alsmadi,, Mohammad Al-Jarrah "Textual Manipulation for SQL Injection attack" I.J. computer Network and Information Security, 2014

[17] Sayyed Mohammad Sadegh Sajjadi and Bahare Tajalli Pour "Study of SQL Injection Attacks and Countermeasures "- ijcce

[18] A Survey on Detection and Prevention Techniques of SQL Injection by Harish Dehariya

[19] W. G. J. Halfond and A. Orso, "Preventing SQL injection attacks using AMNESIA," presented at the Proceedings of the 28th international conference on Software engineering,Shanghai, China, 2006.

[20] A Review on Detection Mechanisms for SQL Injection Attacks - IJIRSET by Aanal Bhanderi , Nency Rawal

[21] Srinivas Avireddy, Varalaxhmi perumal, Narayan Gowraj, Ram Srivastava Kannan"Random4: An Application Specific Randomized Encryption Algorithm to prevent SQL Injection" 11th International conference on trust, Security and privacy in computing and communications IEEE 2012.

[22] J. Duraes, H. Madeira *"Emulation of software faults: A field study and practical approach"* IEEE transaction vol. 32 no.11 pages 849-867 November 2006

[23] R. A. McClure and I. H. Krüger, "SQL DOM: compile time checking of dynamic SQL statements," 2005, pp. 88-96

[24] S. W. Boyd and A. D. Keromytis. "SQLrand: Preventing SQL Injection Attacks", In Proceedings of the 2nd Applied Cryptography and Network Security Conference, pages 292–302, June 2004.

[25] Y. Huang, F. Yu, C. Yang, C. H. Tsai, D. T. Lee, and S. Y. Ku. "Securing Web Application Code by Static Analysis and Runtime Protection", In Proceedings of the 12th International Word Wide Web Conference, May 2004.

[26] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan."CANDID:Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks". ACM Trans. Inf. Syst.Secur, 2010.

[27] Atefeh Tajpour, Maslin Masrom, Suhaimi Ibrahim, Mohammad Sharifi "SQL injection detection and prevention Tools Assessments" IEEE 2010.

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

[29] Qian XUE, Peng HE "On Defence and Detection of SQL Server Injection Attack" IEEE 2011.

[30] Nuno Seixas, Marco Vieira, Jose Fonseca, Henrique Madeira "Analysis of field data on web security vulnerabilities "IEEE Transactions on Dependable and secure computing Vol. 11 No.2 March/Aril 2014.

[31] Hossaian Shahriar, Mohammad Zulkernine, "Information Theoretic Detection of SQL Injection Attacks" International Symposium on high-Assurance systems Engineering, IEEE 2014

[32] M. Martin, B. Livshits, and M. S. Lam "Finding Application Errors and Security Flaws Using PQL: A Program Query Language", ACM Notices, Volume 40, Issue:10 pages,2005.

[33] C. Gould, Z. Su, and P. Devanbu, "JDBC checker: A static analysis tool for SQL/JDBC applications," 2004, pp. 697-698.

[34] R. A. McClure and I. H. Krüger, "SQL DOM: compile time checking of dynamic SQL statements," 2005, pp. 88-96.

[35] Iyano Alessandro Elia, Jose Fonseca and Marco Vieira "Computing SQL Injection Detection Tools Using Attack Injection: An Experimental study" IEEE International Symposium on software reliability Engineering 2012.

