A Survey on Detection and Prevention Techniques of SQL Injection Attacks

Harish Dehariya
Student
UIT, RGPV, Bhopal
Madhya Pradesh, India

Piyush Kumar Shukla,
PhD
Assistant Professor
UIT, RGPV, Bhopal
Madhya Pradesh, India

Manish Ahirwar
Assistant Professor
UIT, RGPV, Bhopal
Madhya Pradesh, India

ABSTRACT

Web applications are widely using nowadays. In these web applications, most of those that are based on money transaction like on-line baking, e-shopping, on-line bill payment, Money transfer, etc. The interaction between the web applications and Database is done with Structured Query Language (SQL) and Scripting Language is used. These queries keep sensitive or personal information of various users. So it is necessary to maintain confidentiality from unauthorized access. SQL injection Attack (SQLIA) is the most common type of vulnerability in which crafted query is inserts as input for retrieving personal information about other users. In this paper, various detection and prevention techniques of SQL injection attacks are described and perform a comparison between them.

Keywords

Web Application, SQL Injection, Vulnerabilities, Detection and Prevention techniques.

1. INTRODUCTION

As the rapid increasing dependency on web applications, organizations connected their database for sharing information. Developer uses various mechanisms for securing the web applications but attacker may get that points where vulnerability may find. All web applications communicate through Databases, which keeps all sensitive information about users. An attacker may get his information and may harm the Database. SQL is a communication medium between Web application and back end database. So the mostly attackers use SQL for accessing a database. A SQL Injection attack (SQLIA) is that in which a malicious mind person injects its own crafted query as input. The back end server executes the query statement and sends the result to the attackers. An attacker obtains the result and back end database. So the mostly attackers use SQL for accessing the database. So the mostly attackers use SQL for accessing a database. A SQL injection attack (SQLIA) is that in which a malicious mind person injects its own crafted query as input. The backend server executes the query statement and sends the result to the attackers. An attacker obtains the result and may use to the purpose of breaking confidentiality of database. This SQL Injection Attack mostly affects financial web applications or secret information system that could be the victim of this vulnerability because the attacker by abusing this vulnerability can threat their authority integrity and confidentiality.

For detection and prevention, various tools have been evolved A detection block model [4] based on calculation on static and dynamic message authentication code (MAC) value. The model works both on client and server side. Client side implements a filter function and server side is based on Information theory. These static MAC value and Dynamic MAC value compared for knowing that the web application is injected by SQL or not. Another approach is SQLRand [10] which is based on a randomization query algorithms

CANDID [26] based on the dynamic candidate evaluation approach for mining the structures of programmer intended queries and a formal basis for this dynamic approach by using symbolic queries. Static analysis and runtime monitoring combine in AMNESIA [30]. SQLGAURD and SQL CHECK [1] proposed a model of expected queries at runtime. In these approaches, the model is expressed as a grammar that only accepts legal queries. Static analysis algorithm is derived from type system and type state for a secure information flow. At the time of analysis codes checks for any vulnerability without user intervention.

The WebSSARI [32] tool uses predefined set of filters for filtering inputs for finding security vulnerabilities in the application. The drawback of this technique is that it assumes that sufficient preconditions for sensitive functions can be precisely expressed adopting their typing system. For many types of applications, this assumption is too strong.

2. MAIN CAUSE OF SQL INJECTION

In this section various cause of SQL injection presented those are:

2.1 Invalidated Input: Any SQL query consist some parameters such as 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.

2.2 Generous Privileges: Privileges are some rules for accessing some database for some object and by object which actions going to be perform. SELECT, INSERT, DELETE are actions of executing SQL queries that included typical privileges.web application is use for accessing any specific information from database. By bypass authentication an attacker gain privileges.

2.3 Uncontrollable variable size: If any variable is using storage for large amount of data so some time can be possible that attacker may enter faked input values.

2.4 Error Message: Error message generates when wrong input values is inserted in web application. Attacker may get the script structure or information about database .so that attacker may create its own attack.

2.5 Clint side only control: If input validation is implemented in client side scripts only, then by using cross site scripting security function of script at client side can be override and attacker can invalidate input or accessing database.

2.6 Stored Procedure: Stored Procedure is small program with some function that calls multiple times in execution. When these functions become calls so that stored procedure become calls in place of that functions. These stored procedures become storedin database. The problem with the stored procedure is that an attacker can execute and damage database.

2.7 Into Outfile support: A text file of containing SQL query result may get by manipulating SQL query. This can be possible by using condition of INTO OUTFILE clause that is benefit of Some Relational Database.

2.8 Sub-select: When a SQL query is inserted in WHERE clause of other SQL query so this is a weakness for Database. This weakness makes the web application more vulnerable.

3. TYPE OF SQL INJECTION ATTACKS

There are different methods of attacks that depending upon the goal of an attacker is performed together or sequentially.

3.1 Tautologies [6]: In this type of attack tautology commands uses for making the SQL query always true. In the SQL query condition uses in WHERE clause so making this condition true the tautology statement applied in WHERE clause.

For example**"SELECT * FROM the employee WHERE userid = '112' and password ='aaa' OR '1'='1'"**

Here tautology statement (1=1) has been added to the query statement so it is always true.

3.2 Illegal /Logically Incorrect Queries [11]: In this type of attack when a query became rejected so an error message is returned that include useful debugging information. An attacker may get the various parameters from this message. So these parameters may help for creating new Query. Junk input is provided for checkingtype mismatches, or logical errors by purpose. For example an URL

http://www.tickitbooking.it/event/?id_nav=8862) is original but in place type

http://www.tickitbooking.it/event/?id_nav=8864' 3)

It will be a SQL injection.

3.3 Stored Procedure [2]:A program coded by Developer for security purpose that is stored in database for providing an abstraction layer is stored procedure. Because it coded by programmer it can also be uses as attack. Depend on specific stored procedure on the database there are different ways to attack. In the following example, an attacker exploits parameterize stored procedure.

For example a stored procedure written as

CREATE PROCEDURE DBO.is Authenticated @userName varchar2, @pass varchar2, @pin int AS EXEC("SELECT accounts FROM users WHERE login='" +@userName+ "' and pass='" +@password+ "' and pin=" +@pin);

3.4 Inference: This type of attackhas two categories are Blind attack and Timing attack.

3.4.1 Blind Injection [1]: When developers hide the error details and provide a generic page in place of an error message so this time attacker can get the information of database structure by asking true/false type of questions through SQL statements.

SELECT accounts FROM users WHERE login= 'doe' and 1 =0 -- AND pass = AND pin=O SELECT accounts FROM users WHERE login= 'doe' and 1 = 1 -- AND pass = AND pin=O

If there is no input validation so query will execute.

3.4.2 Timing Attacks [1]: Timing attack is related to response time given by database. Here **WAITFOR** keyword is used for delay response by database. An attacker can gather information from a database by timing delays. This technique uses an if-then statement for injecting queries.

For example, in the following query:

declare @ varchar(8000) select @ = db_nameO if (ascii(substring(@, 1, 1)) & (power(2, 0))) > 0 waitfor delay '0:0:5'

from the above SQL query it is specify that database will pause for five seconds if the first bit of the primary byte of the name of the current database is 1. Then code is injected to generate a delay in response time when the condition is true. Furthermore, an attacker can ask a series of other questions about this character. As these examples show, the information is extracted from the database using a vulnerable parameter.

3.5 Union Query [2]*:* in this type of attacker join a new query in original query by using UNION keyword and can get data tables from database.

For example

SELECT Name, Phone FROM Users WHERE Id=$id

By injecting Id value and append the following query

$id=1 UNION ALLSELECT creditCardNumber, 1 FROM CreditCarTable

So the executable queries will be as the following:

SELECT Name, Phone FROM Users WHERE Id=1 UNION ALL SELECT creditCardNumber,1 FROM CreditCardTable

3.6 Piggy-backed Queries [11]: As like UNION query this type of attack "**;**"is uses for to append extra query to the original query. With a successful attack database receives and executes multiple distinct queries. Normally, the first query is legitimate query, whereas following queries could be illegitimate. This is mostly use for destroy the table from database.

In the following example, an attacker injects**" 0; *drop table* user"** into the *pin* input field instead of logical value. Then the application would produce the query**:**

SELECT info FROM users WHERE login='doe' ANDpin=0; drop table users

4. SQL ATTACK INJECTION DETECTION TOOLS

4.1Tautology Checker [6] –It is proposed in 2004. This tool provides an analysis framework for security. Static analysis and automated reasoning performs for checking any tautology statement contains in coding. For limited scope of this tool it is not useful.

4.2 CANDID [26] - Dynamically Candidate evaluation tool modify web applications written in java through a 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.

4.3 SQLGAURD [11]-SQL Guard is work on basis of evaluation of the structure of the query before, and after the addition of user-input based on the model.

4.9 AMNESIA [30] - Analysis and Monitoring for Neutralizing SQL Injection Attacks which combines Static and run time analysis. **AMNESIA** is a tool for analysis and monitoring for detecting illegal quires before execution on the database, which is based on static and dynamic analysis. Here on static phase, build a model of queries could be generated in dynamic phase check dynamically generated queries. From the figure it show AMNESIA tool consist three modules is built, in which keeps all possible SQL query queries that may be generated.

Analysis module – In this module identification of hotspots done by scanning of code files and for each hotspot a model

Instrument module - here each hotspot is instrument with a call to the runtime monitor.

Runtime monitoring modules – here query string is give as input and id of hotspot generates the query retrieve the SQL model and check against the model.

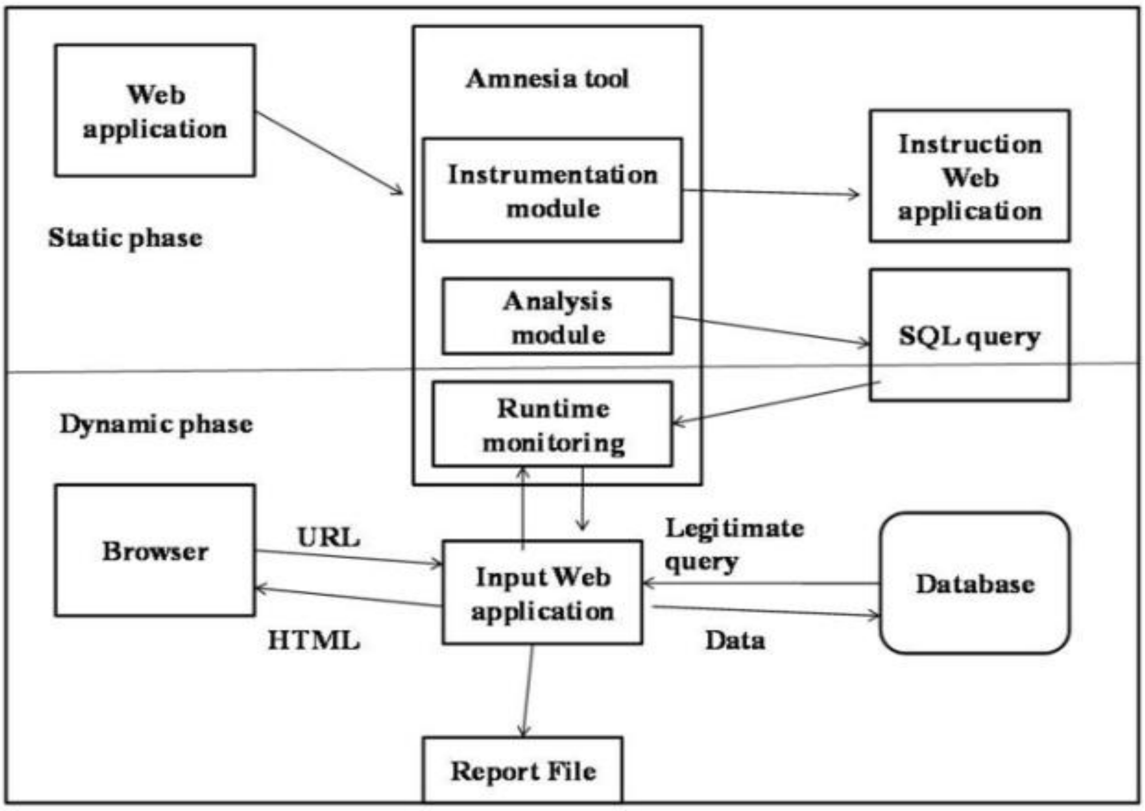

Figure 1:- AMNESIA TOOL et al [30]

4.4 CSSE [29]-This techniques use a Context-Sensitive analysis for detecting SQL tokens in query if an untrusted query id found so query will be rejected. A common drawback of this approach is that it requires modifications at the runtime environment, which affects portability.

4.5 SQL IDS [6] - This tool is based on machine learning technique. It builds a model of typical queries and match at run time that queries those does not match treat as attacks. This tool detects attacks successfully, but it depends on training seriously.

4.6 SQLPrevent[10]– SQL Prevent is works on the interception of an HTTP request. AllHTTP request from web application is stored in storage. When a SQL query generates SQL web application and passes them to SQLIA detector module at this timeHTTP request from thread local storage is fetched and examined to determine.

4.7. SQLRand [10] - 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.

4.8 SQLChecker [1] - In this model is specified independently by the developer. It uses a secret key at runtime checking so security of the approach is dependent on attackers. In approach, 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.

5. SQL ATTACK INJECTION PREVENTION TOOLS

5.1 JDBC Checker [11] - 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.

5.2 WAVES [6] - WAVES a black-box technique that 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.

5.3 SECURITYFly [2] is tool that is implemented for java. Here check 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.

5.4 SECURITY GATWAY [2] - It works 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.

5.5 SQL DOM [6]- 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 because they mainly focus on identifying the obstacles in the interaction with the database via Call Level Interfaces.

5.6 WebSSARI [32]– A WebSSARI tool use for sanitizing input that passed through predefined set of filters. Here 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. WebSSARI’s system architecture is presented in Figure 2. A code walker consists of a lexer, a parser, an AST (abstract syntax tree) maker, and a program abstractor. The program abstractor asks the AST maker to generate a full representation of a PHP program’s AST. The AST maker uses the lexer and the parser to perform this task, handling external file inclusions along the way. By traversing the AST, the program abstractor generates a control flow graph and a symbol table. Based on the prelude files, the verification engine moves through the Control Flow Graph and references the ST to generates type qualifiers for variables and preconditions and post conditions for functions. This routine is repeated until no new information is generated. The verification engine then moves through the control flow graph once again, this time performing type state tracking to determine insecure information flow. It outputs insecure statements (with line numbers and the invalid arguments). For each variable involved in an insecure statement, it inserts a statement that secures the variable by treating it with a sanitization routine. The insertion is made right after the statement that caused the variable to become tainted. Sanitization routines are stored in a prelude, and users can supply the prelude with their own routines.

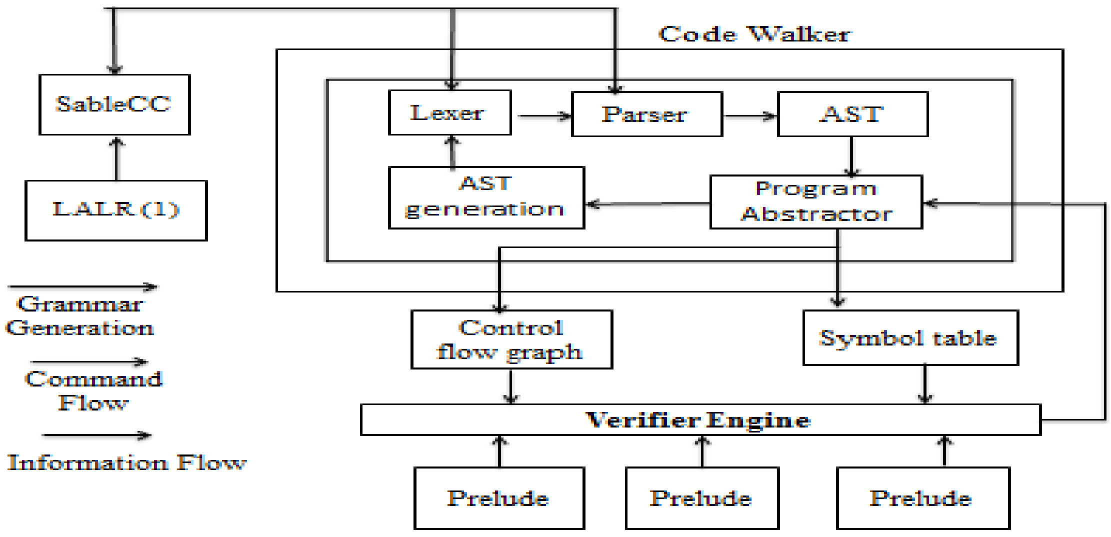

Figure:-2 WebSSARI system Architecture et al [32]

Table 1 Comparison of SQL Injection Attack Detection Techniques with Respect to attack types et al [6, 11]

ATTACK **TOOLS**	**Tautology**	**Piggy Baked**	**Illegal/ Incorrect**	**Union**	**Alternate encoding**	**Timing attack**	**Blind attack**	**Stored procedure**
Tautology checker	√	×	×	×	×	×	×	×
CANDID[29]	√	×	×	×	×	×	×	×
DIWeDa[23]	×	×	×	×	×	√	√	×
CSSE[29]	√	√	√	√	√	×	√	×
SQLRand[10]	√	√	√	√	√	√	√	×
AMNESIA[33]	√	√	√	√	√	√	√	×
SQL IDS[2]	√	√	√	√	√	√	√	√
SQLPrevent[28]	√	√	√	√	√	√	√	√
SQLChecker [6]	√	√	√	√	√	√	√	√

Here sign× Represent attack cannot detected and √ represents attack detected

- From the above table it is clear that Tautology checker and CANDID can detect tautology attack only whereas DIWed can detect TIMIMG and BLIND attacks only.
- SQLRand and AMNESIA can detect all type of attacks except Stored Procedure.
- SQL IDS SQL Checker and SQLPrevent can detect all type of SQL injection Attacks.

Table 2 Comparison of SQL injection Attack Prevention Techniques with respect to attack type's et al [6, 11]

ATTACKS **TOOLS**	**Tautology**	**Piggy-baked**	**Illegal/ Incorrect**	**Union**	**Alternate Encoding**	**Timing Attack**	**Blind Attack**	**Stored Procedure**
JDBC Checker	●	●	●	●	●	●	●	●
WAVES	●	●	●	●	●	●	●	●
Security Gateway	●	●	●	●	●	●	●	●
Security Fly	●	●	●	●	●	●	●	●
SQL DOM	√	√	√	√	√	√	√	×
WebSAARI	√	√	√	√	√	√	√	√

6. CONCLUSION

SQL injection attack is major headache for developers for securing web applications. Various tools have been developed for detection and prevention. In these most of the tools can attack all types of SQLIAs except Stored procedure. In this paper a survey of various detection and prevention techniques of SQL Injection Attack is enlist. Started from Tautology checker that on only check tautology attack then CANDID [26] that can detect more attacks then CSSE [29] by using string evaluation mechanism it made possible to detect all type of attack except Stored Procedure, then AMNEISA [25] which is based on combination of static and runtime monitoring. Other newly method of detection by comparing static MAC value and Dynamic MAC value is done in detection block model [1]. So this is broad area where new tools can be developed for detection as well as prevention of SQL Injection Attacks.

7. REFERENCES

[1] Diksha G. Kumar, Madhumita Chatterjee *"Detection Block Model for SQL Injection Attacks"* I.J. computer Network and Information Security, 2014

[2] Bojken Shehu, Aleksander Xhuvani *"A literature Review and comparaative analysis on SQL injection: Vulnerabiities, attacks and their detection and prevention Techniques"* International Journal of Computer Science Issues, Vol 11,Issue 4, no1 2014

[3] Geogiana Buja, Dr. Kamarularifin Bin Abd Jalil, Dr. Fakariah Bt. Hj Mohd Ali, The Faradilla Abdul *"Detection model for SQL Injection Attack: An approach for preventing a web application from the SQL injection Attack"*IEEE Symposium on Computer Applications and Industrial Electronics, April 2014

[4] 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

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

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

[7] Monali R. Boradel, Neeta A. Despande "Extensive Review of SQLIA's Detection and Prevention Techniques" International Journal of Emerging Technology and Advanced Engineering ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 10, October 2013

[8] Shelly Rohilla, Pradeep Kumar Mittal "Database Security by Preventing SQL Injection Attacks in Stored Procedure" Journal of Advanced Research in Computer Science and software Engineering Volume 3, Issue 11 November 2013.

[9] Jaskanwal Minhas Raman Kumar "Blocking of SQL Injection attack by Comparing Static and Dynamic queries" International Journal of computer network and Information Security 2013

[10] 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"

[11] 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.

[12] Atefeh Tajpour, Suhaimi Ibrahim, Mohammad Sharifi "Web Application security by SQL Injection Detection tools" International Journal of Computer science Issue, Volume 9 Issue 2 No 3 March 2012

[13] Neha Singh, Ravindra Kumar Purwar "SQL Injections – A Hazard to web application" International Journal of Advanced Research in computer Science and Software Engineering Volume 2, Issue 6, June 2012

[14] 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

[15] Kanchana Natrajan, Sarala Subramani "Generation of SQL injection free secure algorithm to detect and prevent SQL Injection attack" ELESE VIER C3IT-2012

[16] Inyong Lee, Soonki Jeong, Sangsoo Yeo, Jongsub Moon "A novel method for SQL Injection attack detection based on removing SQL Query attribute values", ELSEVIER 2012.

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

[18] Jie Wang, Raphael C.W. Phan, John N Whitley, David J. Parish "Augmented Attack Tree Modelling of SQL Injection Attacks" IEEE 2010

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

[20] Ntagwabira Lambert, Kang Song Lin "Use of Query Tokenization to detect and prevent SQL Injection attacks" IEEE 2010

[21] J. Fonseca, M. Vieira, and H. Madeira*, "The web Attacker Perspective –A Field study"* IEEE 2010.

[22] Michelle Ruse, Tanmoy Sarkar, Samik Basu*"Analysis and Detection of SQL Injection Vulnerabilities via Automatic Test Case Generation of Programs".* Annual International Symposium on application and the Internet. 2010

[23] Nuno Auntunes, Nuno Laranjeiro, Marco Vieira, Henrique Madeira *"Effective detection of SQL /X Path Injection Vulnerabilities in web services"* IEEE International conference on services computing 2009.

[24] A. Roichman E. Gudes, *"DIWeDa –Detecting Intrusions in Web Databases"*. Vol. 5094, pp. 313-329 Springer Heidelberg 2008[26] J. Fonseca and Marco Vieira *"Mapping software fault with web security vulnerabilities"* IEEE conference on dependable system and network, June 2008

[25] J. Fonseca and Marco Vieira and Henrique Madeira *"Training Security Assurance Team using Vulnerability Injection"* IEEE Pacific Rim Dependable Computing, December 2008

[26] P. Grazie *"SQL Prevent Thesis"* University of Columbia, Vancouver, Canada 2008

[27] Prithvi Bisht, P. Madhusudan, V N. Venkatraman, Sruthi Bandhakavi *"CANDID Preventing SQL injection Attack using Dynamic Candidate Evaluations"* ACM Transactions on Information and Security (TISSEC) October/November 2007

[28] Fonseca, J. Vieira, M. Madeira, "H. Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks" *IEEE* Dec. 2007.

[29] 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

[30] T. Pietraszek, C. V. Berghe. *“Defending against Injection Attacks Trough Context-Sensitive String Evaluation”* Recent Advanced in Intrusion Detection Volume: 3858, 2006

[31] William G. Halfond and Alessandro Orso *“AMNESIA: Analysis and Monitoring for NEutrializing SQL Injection Attacks*” pages 22-28 St. Louis, MO, USA, May 2005

[32] McClure and I. H. Kruger, *“SQL DOM: Compile time checking and dynamic SQL statements”* Software Engineering ICSE 2005.

[33] Yao-Wen Huang, Fang Yu, Christian Hang, Chuang Hang, Tsai, D.T. Lee, Sy-Yen Kuo *“Securing Web Application Code by Static Analysis and Runtime Protection*” 13th conference on World wide web in ACM New York USA 2004.

[34] Y. Huang S. Huang T. Lin and C. Tsai, *“Web Application security Assessment by Fault Injection and Behavior"* In Proceeding of the 11th International World Wide Web Conference, May 2003.