iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** VII **Month of publication:** July 2017

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VII, July 2017- Available at www.ijraset.com

Study On SQL Injection Attacks: Detection And Prevention

Rishab Garg[1], Priya Gupta[2], Rohan Kr Sachdeva[3]
[1,2,3] *School of Information Technology, Vellore Institute of Technology Vellore, India*

Abstract: ***SQL injection also referred to as SQLI in short. It is the most dangerous way in which our data can be hacked. Hackers can get to our databases prudently through Web Interface and can delete, modify the important content of database. The principle behind SQL Injection is very simple, but utter dangerous and powerful. When the application takes input from user through any form etc. the malicious users get opportunity to enter the diligently crafted data which will be interpreted as the SQL query instead of the data. This query will extract the database details and will let the doors wide open for the opportunists to misuse the data. Not only database modification of database, this will let the hackers to get unauthorized access to the application services also. Injection attack is at the first place of the top 10 web attacks that are executed in 2013. SQL injection is a method for exploiting web applications that use client supplied data in SQL queries. SQL Injection refers to the technique of inserting SQL meta characters and commands into Web based input fields in order to manipulate the execution of the backend SQL queries. The occurrence of SQLI is triggered when hacker changes the functioning of query by inserting SQL commands. Our goal is to implement different SQLI attacks and through the results we will see how important data is compromised by changing the query. This loss of data can cause a company to lose in millions. We will try to analyse various attacks in order to get an in depth knowledge of how these attacks work.***

Keywords: ***SQLI; intrusion; attacks; database; scanning; vulnerability;***

I. INTRODUCTION

The intruder of the SQLIA attack has the main goal to access the database for which he is not authorised. In today's modern digital world more and more companies are reliant on the use of web applications for distribution of services to the users. These applications receive the users or clients request which in turn interacts with the back-end database and returns relevant data for the user. These database mostly contains the confidential and sensitive information of users such as their personal, medical and financial data in which the attackers has keen interest. SQLI is the main attack used by attackers to exploit and compromise the security of the database. It exploits vulnerabilities that exists in the back-end database server which allows hackers to inject a malicious payload of SQL query segments to manipulate the effect of the SQL query, which gives the attackers access to database, read or modify or maybe corrupt the whole database server.

However, there doesn't exist a single complete solution that can solve all the problem that are prevalent today. This paper provides us with the focus on various concepts of SQLI. Section II gives us the idea of various detection and prevention techniques and other work done previously in this field. Section III contains the procedure for the detection of the given SQL attack and what it signifies. Section IV is all about SQLIA Tools Analysis followed by Section V SQLIA Prevention. Then Section VI is Future work and then Section VII Conclusion.

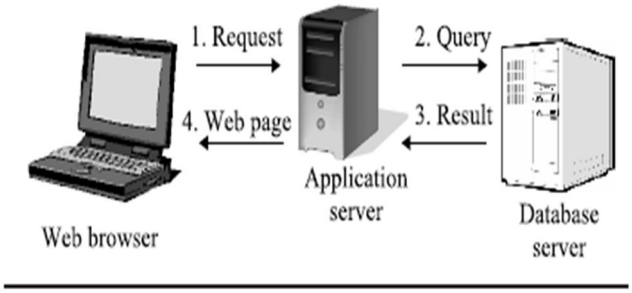


Fig. 1. Generic Request-Reply Architecture

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VII, July 2017- Available at www.ijraset.com

II. RELATED WORK

On reviewing many electronic journals and articles from ACM, IEEE, and host of other websites to know in depth about the various SQLI attacks the following papers covers different important aspects or techniques to prevent SQLI attacks:

A. From "Using Parse Tree Validation to prevent SQLI attacks" ACM, SQLI discovery was covered and the paper discussed various aspects of SQL parse tree validation.[1]

B. From "The Essence of Command Injection Attacks in Web Applications" ACM, various techniques to check and prevent input query which uses SQLCHECK and grammar to validate query. [2]

C. From "Using Automated Fix Generations to Secure SQL Statements" IEEE, the background of SQL statement and vulnerability was covered.[3]

D. From "Automated Protection of PHP Applications against SQLIA" the method originally used to secure application from SQLIA is covered which puts together static analysis, dynamic analysis, and automatic code re-engineering to protect existing properties.[4]

E. From "Preventing SQLI attacks in stored procedures" a simple approach to secure the stored procedures from attack and detect the SQLIA from site. It combines runtime check with static application code analysis so the vulnerability of attack can be eliminated. It changes the structure of the original SQL statement and detects the SQLIA. This method is further divided into two phases, one deals at offline and other at runtime level. In offline phase, parsing, pre-processing and detection of SQL statements is done by stored procedures in the runtime analysis of execution call. In final runtime phase the user input structure is checked with original structure of statement by the process. Once it detects the malicious payload the access is denied and the details about attack is sent back.[5]

III. SQLIA DETECTION

A. Introduction

This section discusses techniques to detect SQLI attacks against an organizations network. There is an ample amount of discussion on this and how to carry them out, their impact, and the prevention techniques using improved and efficient coding and design practise. We take popular open source tools for detecting these attacks.

B. Open Source Tools

There is a grave need to detect the SQLI attacks, to achieve this these are few enumerated open source tools which are capable of scanning and detecting the attacks:

1) *Acunetix Web Vulnerability Scanner Tool[25]:* This tool works on the idea of splitting the attack in the basis of severity. It classifies the attacks as low, medium, hih and informational severity. The tool is capable of detecting web vulnerabilities like SQL injection, URL redirection, Firewalls and SSL, CGI scripting, Cross site scripting.

Among these SQL injection attacks and Cross site scripting scans belong to the high threat label as they are very severe form of attacks.

This scanner tool does a fairly efficient scanning but is found to be slow than other commercial scanning tools in the market.

2) *SQL Map[24]***:** A python application SQL Map is a dedicated SQL injection scanner. The tool aims to detect the SQLI attacks vulnerabilities and it feeds on these outcomes to take advantage on web applications. SQL Map initially detect the loop whole in you site and then use variety of option to perform extensive back-end database management, enumerate users, dump entire or specific DBMS, retrieve DBMS session user and database, read specific file on the file system.

The tool is quite fast than the former mentioned Acunetix, but it still not the best. It also make very few URL injection database as compare to other tools like SQLiX. This tool also doesn't have GUI interface.

3) *Wapiti[23]:* Another python based application tool to detect SQLI. This tool uses command line of Python and relies on its library named Iswww. The library facilitates the tool to crawl on the given website in order to scan it, similar to web crawling. It also uses library like Tidy to clean the HTML pages which are not well formatted. Basically it does black-box scanning. Wapiti scans the all Web Pages available on your site and try to find out scripts and form where it can inject data to check how many types of attack are possible on selected injection point.

Wapiti can detect SQL injection and XSS (Cross Site Scripting) injection. Wapiti has one of the best features that it's able to differentiate temporary and permanent XSS vulnerabilities. It does not provide a GUI interface and you must have to use it from command line interface.

4) *Paros[30]:* A web application used for web application security assessment. This comparing to its contemporaries is a

JAVA application used to evaluate the web pages and their securities. It is open source free application, using Paros's one can exploit and modify all HTTP and HTTPS data among client and server along with form fields and cookies. The core function what this scanner Paros does is explained as follows. According to web site hierarchy, it scans the server, then it checks for server misconfiguration. This is added feature because few URL paths cannot be found and recognized by the crawler. The other automatic scanners are not able to do that.

Basically to be capable of this functionality Paros navigates the URL site and rebuilds the website hierarchy. Paros does three types of server configuration checks as HTTP PUT, Directory indexable, and obsolete file exist. Paros also provides log file, which is create when all the HTTP request and reply pass through Paros. In log panel Paros shows back as request and reply format.

5) *Pixy[29]:* Pixy is the second tool that is written in Java. Pixy can do automatic scans for PHP and also does for the detection for SQL injection and XSS attacks. Pixy functions by taking whole PHP file as an input and producing a report that shows the possible vulnerability section in that PHP file along with some additional information to understand attack. The major disadvantage of Pixy is that it only woks for PHP 4 and not for OOPHP 5.

While SQL injection analysis Pixy divides result in three categories: untainted, weakly tainted, and strongly tainted. It also provide dependence graph and dependence value. Dependent value is nothing but the list of points in program on which the value of variables is depends.

6) *SQLIX[20,21]***:** SQLIX is another SQL injection scanner developed as OWASP(Open Web Application Security Project) project. Talking about OWASP, it is a free and open worldwide community, which focusses on improving application software security. It's based on an Perl Scanner, which identifies the vulnerability of the back-end database, crawls and detects an SQLI attack. Various Perl modules are used by SQLIX. **C**omprehensive **P**erl **A**rchive **N**etwork (CPAN), CPAN, provides an ample amount of Perl software as well as their documentation, which helps any coder to use these libraries and Perl modules in their projects.

As we go one to learn about SQLIX we imbibe that SQLIX is not merely a scanner for injections but has very interesting and important feature of building our own function. Our own function can be deployed to inject malicious code into the database and carry out tests for different vulnerabilities. This feature is not available in any other vulnerability scanners available in market. The intriguing feature helps us in preparing for all kind of possible SQLI attacks. Also instead of giving main URL in command line interface for crawl, we can list the number of Main URLs which we wish to crawl in one file and then call that file. SQLIX involves various modules which determines it's performance factor as discussed in this paragraph:

Crawl is a Perl module to crawl by specifying target and crawl through all the web pages and forms present in that specified URL. The crawler needs Spider.pm Perl module to browse the entire webpage build on the URL. The Mechaniz.pm Perl module, this module handles the job of collecting the URLs to proceed for further process. It is a crucial role and forms the heart of SQLIX tool. Checksite.pm module to check and validate the URL which is present on the website page. The process involves first validating and then stacking on for SQLI operation

IV. SQLIA TOOLS ANALYSIS

This section is all about, the various tools discussed above here are analysed and identified over certain parameters. The table below shows the values of each tool against particular parameters.

TABLE I
VALUE OF EACH TOOLS

TOOLS	EXECUTION TIME	NO. OF INJECTIONS	USER DEFINED FUNTION	NUMBER OF TYPE OF ATTACKS	DATABASE SUPPORTS	LANGUAGE	GUI
SQLIX	2-3	300	YES	2	MY SQL, MS ACCESS	PERL	YES
ACUNITIX	25-30	--	NO	5	MY SQL, MS ACCESS	--	YES
SQL MAP	4-5	41	NO	3	MY SQL, PSQL, MSQL	PYTHON	NO
WAPITI	7-8	XSS 90 SQL 40	NO	2	MY SQL PSQL	PYTHON	NO
PAROS	8-10	40	NO	2	--	JAVA	YES
PIXY	4-5	--	NO	2	--	JAVA	YES

To visualize the table following are the by histograms. The x axis bears tools as in the manner given in the table:

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VII, July 2017- Available at www.ijraset.com

A. Depending Upon the Number of Injections by the Tool

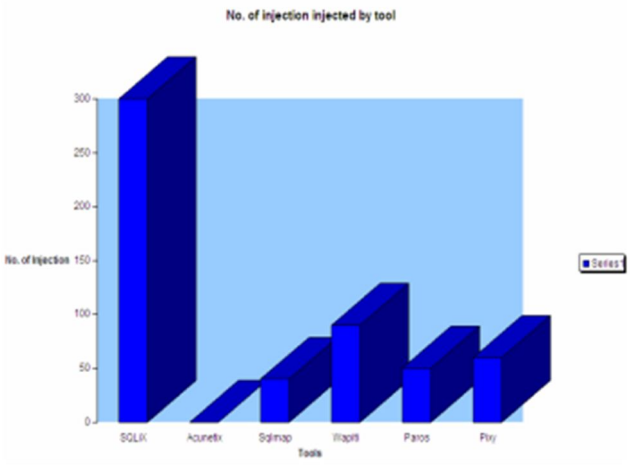


Fig.2. No. of Injection Injected by Tool

B. Depending Upon the Time Taken by the Tool

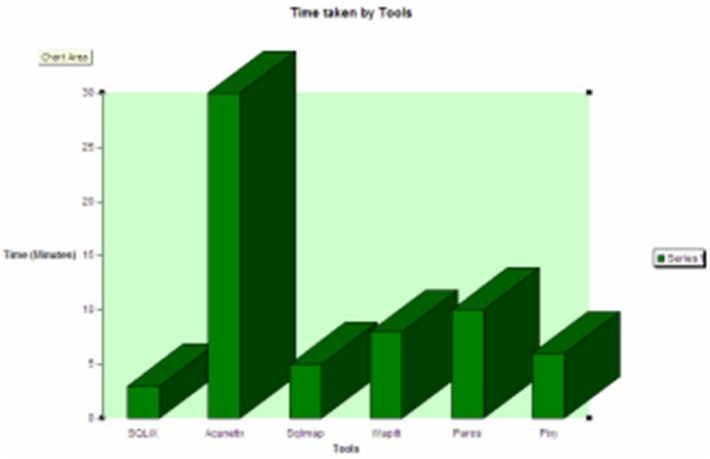


Fig.3. Time Taken by Tool

C. Depending Upon the Type of Attacks Done by the Tool

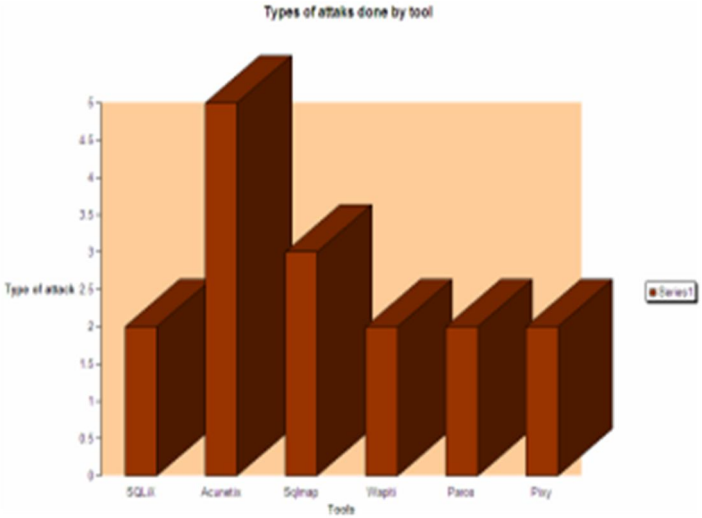


Fig.4. Type of Attacks Done by Tool

D. Depending Upon the Number of Database Supported by the Tool

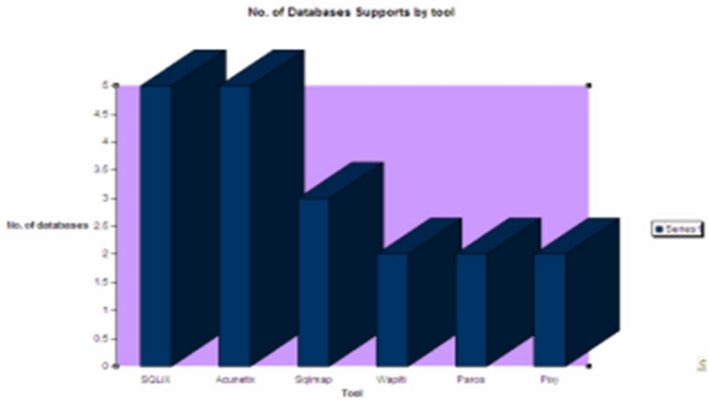


Fig.4. Types of Database Supported by Tool

V. SQLIA PREVENTION

Every time the code differs when we are preventing the SQLA. There are many ways to do it. Some of them are described in this paper. The Code depends on the coder. Here are some ways how we can do it:

A. SQLIA Prevention by Stored Procedure [18]

Stored procedures are like subroutines which can be called by the running applications. The prevention is implemented by the two analysis: they are static and dynamic analysis. Stored procedure parser is used to identify the commands. The runtime analysis does the input checking giving by the user using the SQL checker. This paper deals with the introduction, detection and prevention of the SQL injection attacks. Prevention includes the combination of static and runtime analysis.

B. Adaptive Algorithm [19]

This method comprises of the code conversion methods and the top features of the parse tree validation methods. This algorithm parse the input of the user and verify whether it's prone to danger or not. If the parser detects any danger in the commands then it will use the code conversion techniques and change the input of the user.

C. Comparison

Our main agenda is to increase the efficiency of the prevention algorithms so that the SQLI attacks can be reduced or diminished. Table below shows the ways the different prevention techniques are used against the attacks and their efficiency comparison with each other. This table shows the result of the comparison. These comparisons are useful for us in order to choose the prevention algorithms:

TABLE II
EFFICIENY OF VARIOUS TECHNIQUES

Schemes	Tautology	Logically Incorrect Queries	Union Query	Stored Procedure	Piggy Backed Queries	Inference Attack
AMNESIA	YES	YES	YES	NO	YES	YES
SQLrand	YES	NO	NO	NO	YES	YES
CANDID	YES	NO	NO	NO	NO	NO
SQLGuard	YES	NO	NO	NO	NO	NO
SQLIPA	YES	YES	YES	NO	YES	YES
Negative Tainting	YES	YES	YES	NO	YES	YES
Positive Tainting	YES	YES	YES	YES	YES	YES

D. Advantages of Static Analysis

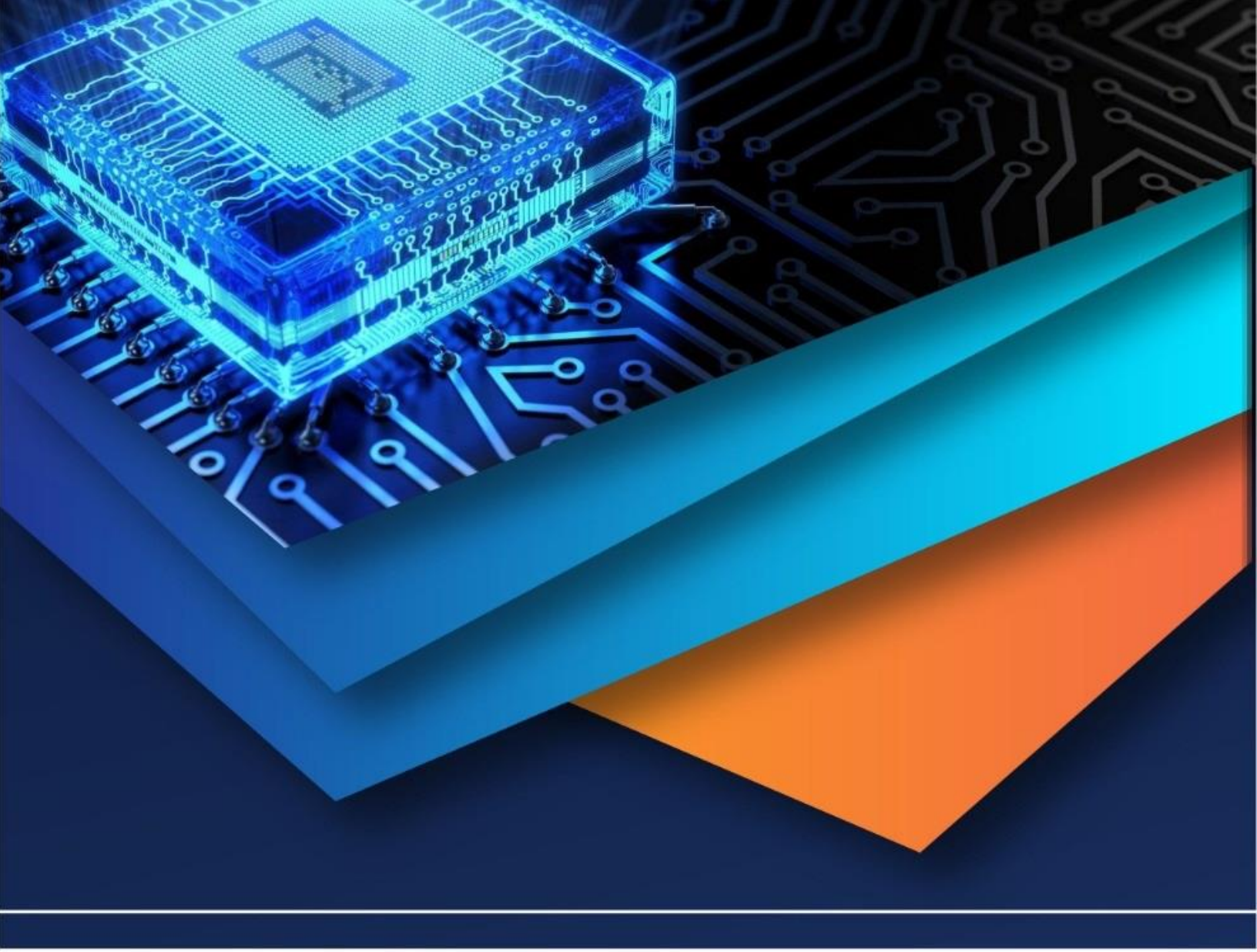
Firstly the query which does not take user input is not included in the SQL graph(control) and also SQL control graph representation counts towards the queries which includes user input to access the database.

VI. CONCLUSION

This paper covered what we initially talked about the SQLI and its various detection and preventive approaches with their performance evaluation. As the future work, we want to evaluate methods using different web based application script with public domain to achieve great accuracy in SQL injection prevention approaches. Integrating SQLIX with http scanner, http scanning and using metasploit will help in detected upcoming vulnerabilities. we also need to add feature to dump old database. This paper covered what we initially talked about the SQLI and its various detection and preventive approaches with their performance evaluation.

REFRENCES

[1] Wei, K., Muthuprasanna, M., & Suraj Kothari. (2006, April 18). Preventing SQL injection attacks in stored procedures. Software Engineering IEEE Conference. Retrieved November 2, 2007, from http://ieeexplore.ieee.org
[2] Thomas, Stephen, Williams, & Laurie. (2007, May 20**).** Using Automated Fix Generation to Secure SQL Statements. Software Engineering for Secure Systems IEEE CNF. Retrieved November 6, 2007, from http://ieeexplore.ieee.org
[3] Merlo, Ettore, Letarte, Dominic, Antoniol & Giuliano. (2007 March 21). Automated Protection of PHP Applications Against SQL-injection Attacks. Software Maintenance and Reengineering, 11th European Conference IEEE CNF. Retrieved November 9, 2007, from http://ieeexplore.ieee.org
[4] Wassermann Gary, Zhendong Su. (2007, June). Sound and precise analysis of web applications for injection vulnerabilities. ACM SIGPLAN conference on Programming language design and implementation PLDI, 42 (6). Retrieved November 7, 2007, from http://portal.acm.org
[5] Friedl's Steve Unixwiz.net Tech Tips. (2007). SQL Injection Attacks by Example. Retrieved November1,2007,from,http://www.unixwiz.net/techtips/sql-injection.html
[6] Massachusetts Institute of Technology. Web Application S ecurity MIT Security Camp. RetrievedNovember1, 2007, from http://web.mit.edu/netsecurity/Camp/2003/clambert-slides.pdf
[7] Massachusetts Institute of Technology. Web Application Security MIT Security Camp. RetrievedNovember1,2007,fromhttp://groups.csmail.mit.edu/pag/readinggroup/wasserman07injection.pdf
[8] Gregory T. Buehrer, Bruce W. Weide, and Paolo A. G. Sivilotti. The Ohio State University Columbus, OH 43210 Using Parse Tree Validation to Prevent SQL Injection Attacks. Retrieved January 2005, from http://portal.acm.org
[9] Zhendong Su, Gary Wassermann. University of California, Davis. The Essence of Command Injection Attacks inWeb Applications. Retrieved January 11, 2006, from http://portal.acm.org
[10] William G.J. Halfond, Alessandro Orso, and Panagiotis Manolios College of Computing – Georgia Institute of Technology. Using Positive Tainting and Syntax-Aware Evaluation to Counter SQL Injection Attacks. Retrieved November 11, 2006, from http://portal.acm.org
[11] William G.J. Halfond and Alessandro Orso. College of Computing Georgia Institute of Technology. Preventing SQL Injection Attacks Using AMNESIA. Retrieved May 28, 2007, from http://portal.acm.org
[12] José Fonseca, Marco Vieira, Henrique Madeira. CISUC, University of Coimbra Dep. of Informatics Engineering 3030 Coimbra – Portugal. Online Detection of Malicious Data Access Using DBMS Auditing. Retrieved March 20, 2008, from http://portal.acm.org
[13] Frank S. Rietta 10630 Greenock Way Duluth, Georgia 30097. Application Layer Intrusion Detection for SQL Injection. Retrieved , Retrieved March 12, 2006, from http://portal.acm.org
[14] Martin Bravenboer, Eelco Dolstra, Eelco Visser, Delft University of Technology The Netherlands. Preventing Injection Attacks with Syntax Embeddings A Host and Guest Language Independent Approach. Retrieved October 3, 2007, from http://portal.acm.org
[15] Yuji Kosuga, Kenji Kono, Miyuki Hanaoka Department of Information and Computer Science Keio University. Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection. Retrieved November 12, 2007, from IEEE Computer Society. http://ieeexplore.ieee.org
[16] Benjamin Livshits and U´ lfar Erlingsson. Microsoft Research. Using Web Application Construction Frameworks to Protect Against Code Injection Attacks. Retrieved June 14, 2007, from http://ieeexplore.ieee.org
[17] José Fonseca CISUC - Polithecnic Institute of Guarda, Marco Vieira, Henrique Madeira DEI/CISUC - University of Coimbra. Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks. Retrieved July 10, 2007, from http://ieeexplore.ieee.org
[18] Hal Berghel. Hijacking the Web Retrieved January 2, 2002, from http://portal.acm.org
[19] Engin Kirda, Christopher Kruegel, Giovanni Vigna, and Nenad Jovanovic Technical University of Vienna. Noxes: A Client-Side Solution for Mitigating Cross-Site Scripting Attacks. Retrieved June 5, 2006, from http://portal.acm.org
[20] http://www.owasp.org/index.php/Category:OWASP_SQLiX_Project
[21] http://www.owasp.org/index.php/Category:OWASP_SQLiX_Project
[22] http://search.cpan.org/~petdance/www-Mechanize- 1.34/lib/www/Mechanize.pm
[23] http://wapiti.sourceforge.net/
[24] http://sqlmap.sourceforge.net/
[25] http://www.acunetix.com/
[26] http://en.wikipedia.org/wiki/Cross-site_scripting
[27] http://en.wikipedia.org/wiki/SQL_injection
[28] http://www.unixwiz.net/techtips/sql-injection.html
[29] http://pixybox.seclab.tuwien.ac.at/pixy/index.php
[30] http://www.parosproxy.ort/index.shtml



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY