ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 9, November 2012

Sql-Injection Tool for finding the Vulnerability and Automatic Creation of Attacks on JSP

[1]Praveen Kumar [2]Himanshu Kumar [3]Remya Joseph
[1]Mtech Software Technology, [2,3]Mtech IT,School of Information Technology and Engineering (SITE),VIT University, Vellore - 632014, Tamil Nadu, India.

***Abstract* — These days' cyber attacks have become a major concern because these attackers can steal important documents and damage websites and access confidential information and may drive many corporations that conduct their business through the web to suffer financial and reputation damages. Out of all those attacks the most dangerous cyber attack is the Structured Query Language (SQL)-injection attack. This type of attack can be easily made via normal web browsers that we use for surfing the net in our day to day life. A characteristic diagnostic feature of SQL injection attacks is that they change the intended structure of queries issued. Most web application developers do not apply user input validation and they are not aware about the consequences of such practices. Due to these inappropriate programming practices a large room for SQL-injection attack is left open which lure the hackers to steal confidential information from the servers' database [4]. In order to handle this vulnerability and detect it, we must enhance the coding structure used for web application development and this requires development of a powerful tool that can automatically create SQL-injection attacks using efficient features (different attacking patterns). Our technique for detecting SQL injection is to dynamically mine the programmer - intended query structure on any input, and to detect attacks by comparing them against the intended query structure.**

***Index Terms*—SQL injection attack, Smart injector Tool,Types of SQL attacks.**

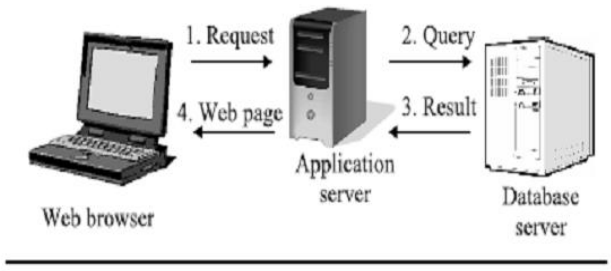


Figure 1: "Web application Architecture "
Source: Gary Wassermann Zhendong Su, Sound and Precise Analysis of Web Applications for Injection Vulnerabilities, University of California, Davis, 2007

I. INTRODUCTION

SQL injection is a basic attack used either to gain unauthorized access to a database or to retrieve information directly from the database. The basic principles underlying SQL injection are simple and these types of attacks are easy to execute and master. SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. An SQL injection is often used to attack the security of a website by inputting SQL statements in a web form to get a badly designed website to perform operations on the database (often to dump the database content to the attacker) other than the usual operations as intended by the designer. SQL injection is a code injection technique that exploits a security vulnerability in a website's software. The vulnerability happens when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.

The SQL Injection attack is a type of attack on the website or the domain through which one can access the database and fetch the data of the database. Database will have table, table will have columns and that columns will have the data, in which sensitive information are stored like username, passwords, email addresses, contact information etc. SQL commands are thus injected from the web form into the database of an application (like queries) to change the database content or dump the database information like credit card or passwords to the attacker. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

Some preventive measures should be taken like code reviewing and penetration testing to address this problem of lack of security in web applications [14, 15]. But, the incredible growth of web applications both in number and in complexity makes it much difficult in realistic world to protect the web applications from being vulnerable to such attacks [13]. Human resources are limited and, in many situations, they are no longer able to deal with the huge amount of source code present in web applications within an organization (and with their ramifications to other systems). Also certain economic constraints restrict the transfer of resources to the security department. Thus the use of automated security tools is mandatory nowadays. In fact, for the security community it is important to build automated tools that allow an easier evaluation and prevention of security problems. Using well-designed query language interpreters can prevent SQL injections. In the wild, it has been noted that applications experience, on average, 71 attempts an hour. When under direct attack, some applications occasionally came under aggressive attacks and at their peak, were attacked 800–1300 times per hour.

II.Existing System

A. *Overview*

There are many such tools and techniques which are used to reduce the input manipulation vulnerabilities during development and testing time to detect or prevent those input manipulation vulnerabilities and to improve the web programs. This section discusses those techniques and compares it with our approach [12].

B. *Manual approaches*

In this section we discuss about the manual approaches used to detect and prevent input manipulation vulnerabilities.

Defensive programming-

The input manipulations can be prevented, by implementing the application in a way that user input cannot contain malicious characters or keywords. Programmers can implement their own input filters by using white lists or black lists. Our approach helps to improve user input validation by providing information on vulnerable method arguments and malicious input.

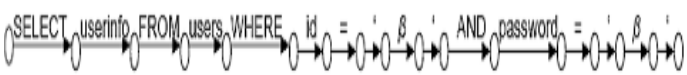


Figure 2: SQL query model.

Code review-

It is known to be effective in detecting bugs with low cost [9]. However, code review is a time consuming task compared to automated static analysis [10] and may be skipped by software development teams rushing to ship an application.

C. *Automated Approaches*

This section describes automated approaches to detect and prevent input manipulation vulnerabilities. In this era of information technology internet is being used worldwide for various purposes like financial transactions and educational endeavors. There is always a risk associated with these online transactions. One of the most important attacks that are nowadays equally worrying the website developers and users is sql injection attack. It is a break-in strategy wherein the attacker tries to access sensitive information from a database by first generating a query that will cause the database parser to malfunction, followed by applying this query to the desired database. Many studies have proposed various tools for securing web against these attacks. Some of them are briefly described below.

1.)Secubat[1]

- Uses black box testing approach.
- Scans web pages for vulnerability using multi-threaded crawling attack and analysis components
- It has ability to auto generate 4 attack components like SQL injection, simple redirecting cross site scripting, encoded reflected cross site scripting and form redirecting XSS attack.

Disadvantage:

- Doesn’t apply various blind SQL injection attack

2.) Sql-Ids Intrusion Detection System[3]

- Monitors java based web application
- First determines the syntactic structure of the SQL query that is originated and executed by the web application system
- It then monitors and detects the web applications when executing the sql queries that are in violation of determined specification of sql queries.
- It detects SQL injection attacks in real time.
- It works independently without affecting the web application or the database.

Disadvantages:

- It overloads[3] operating system and degrades the performance of the web server machine when it receives a large number of requests.
- It only monitors java web based application

3.) Safeli Intrusion Detection System

- This tool involves byte code of java web applications.
- It utilizes the symbolic execution in order to efficiently inspect security flaws in the real time, where an equation is constructed to figure out the initial values of the web controls that may lead to a web security breach in the database
- It then places the created equation at every location where sql query is used to retrieve information.
- A hybrid string solver solves this equation
- It detects SQL injection attacks in real time[4].

Disadvantages:

- It creates load on the web server.
- This technique does not expose the hidden SQL injection vulnerability, but just detect the SQL injection attacks in the real time.
- It is suitable for Java based web applications only

4.) Ardilla Scanning Tool

- It expose web vulnerabilities using automatic input creation which acts as concrete SQL injection and XSS attacks, in order to locate SQL injection and XSS vulnerabilities in the web application[4]
- It does not involve blind SQL injection attacks in the scanning process
- Uses the white box testing approach in its scanning process, which requires examining the source code of the web application line by line

Disadvantages:

- It is useful only for PHP based web applications
- This tool requires the source code to start scanning.

5.) Amnesia Intrusion Detection System[5]

- It uses model based approach that detects unexpected and illegal SQL queries before they are executed.
- It has two parts (a.) static and (b.) dynamic
  a. The static part has program analysis to generate legitimate SQL queries model that can be built by the application
  b. The dynamic part the technique adopts runtime monitoring to inspect the generated queries and to check them against statically built model

Disadvantages:

- Consumes the resources represented in memory and the CPU of the web server
- They need memory to operate and they increase the traffic over the network and cause a situation known as the bottleneck[3].

6.) Mysqll Injector[6]

- It conducts efficient penetration test on PHP based websites to detect SQL injection vulnerabilities.

- It uses various attacking patterns which tricks the web server to display error notifications when it comes across the vulnerability.
- There are a total of 10 such attacking patterns.
- Injecting the attacking patterns comes in 4 different levels and in the final level this tool tries to predict the number of infected columns in the database.
- It is not considered as an additional load on the web server because it is operated remotely as a client side tool

Disadvantages:

- It works only for PHP based web applications

III. Proposed System

SQL injection vulnerability occurs because of the fact that most web application developers do not implement user input validation and they are not aware about the consequences of such practices [2]. SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.

The primary form of SQL injection consists of direct insertion of code into user input variables that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.

However, assume that the user enters the following:
Redmond'; drop table OrdersTable--
In this case, the following query is assembled by the script:
SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond';drop table OrdersTable--'

The semicolon (;) denotes the end of one query and the start of another. The double hyphen (--) indicates that the rest of the current line is a comment and should be ignored. If the modified code is syntactically correct, it will be executed by the server. When SQL Server processes this statement, SQL Server will first select all records in OrdersTablewhere ShipCity is Redmond. Then, SQL Server will drop OrdersTable

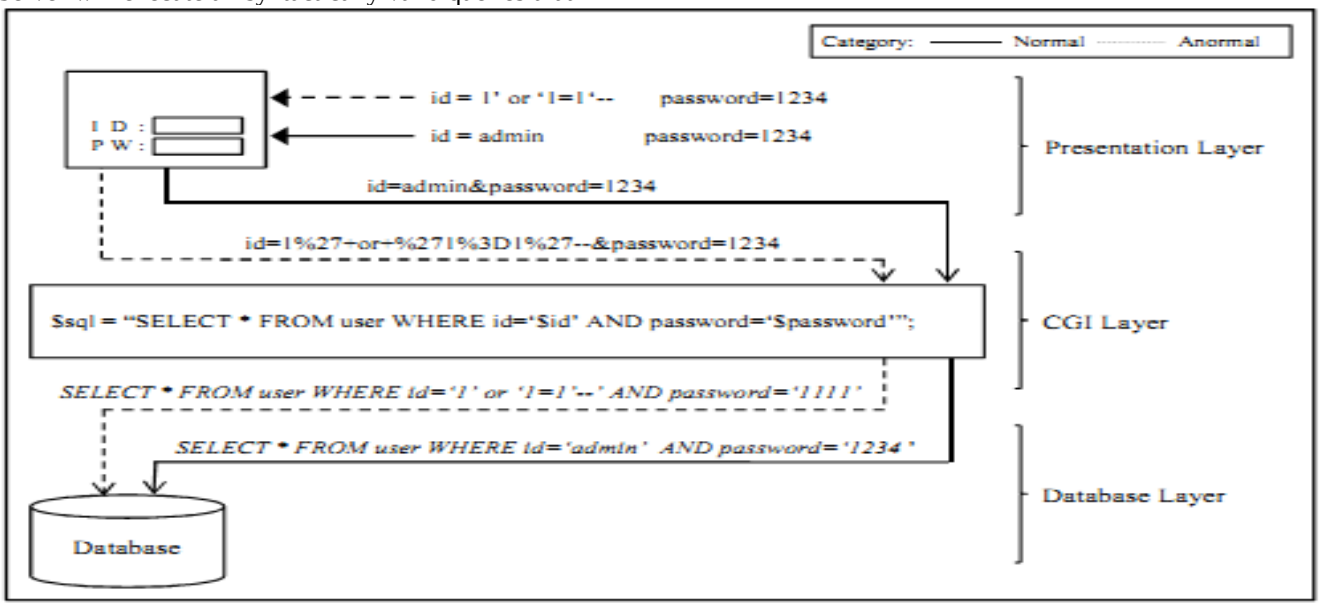


Fig. 2. SQL normal and SQL Injection Attack data flow

The injection process works by prematurely terminating a text string and appending a new command. Because the inserted command may have additional strings appended to it before it is executed, the malefactor terminates the injected string with a comment mark "--". Subsequent text is ignored at execution time.

The following script shows a simple SQL injection. The script builds an SQL query by concatenating hard-coded strings together with a string entered by the user:

```
var Shipcity;
ShipCity = Request.form ("ShipCity");
var sql = "select * from OrdersTable where ShipCity = '" + ShipCity + "'";
```

The user is prompted to enter the name of a city. If she enters Redmond, the query assembled by the script looks similar to the following:

```
SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond'
```

As long as injected SQL code is syntactically correct, tampering cannot be detected programmatically. Therefore, you must validate all user input and carefully review code that executes constructed SQL commands in the server that you are using.

Another example is shown in the above diagram

Here there is an ID and Password field in a web page and we enter some SQL Injection Query like 1' or'1=1'--. This is passed to the CGI layer where it is not properly parsed and filtered for unwanted symbols and thus gets access into the database maintained by the website. Thus a SQL injection attack occurs.

```
SELECT * FROM users
WHERE username = 'bob' and PASSWORD = 'mypassword'; DELETE FROM users
WHERE username = 'admin';
```

Many a tools available in the internet have limited features in shaping efficient attacking patterns and they use brute force techniques to extract data from targeted website which are required to detect hidden SQL injection vulnerability [1,2]. These tools do not show meaningful and detailed information about the detected vulnerability. Thus in our proposed tool we have tried to make all possible types of attacks on the Java Server Page including those types of attacks which are already done. Here first we have identified all features which are supposed to be there in a SQL injection scanning tool which does a scan on php and other web based pages. This was challenging as conducting an efficient penetration test, especially automating the penetration testing process to innovate a new scanning tool is very difficult. The most important aspects of the attacking patterns used in the tool are the ability to trick the web server to display error notifications if it was inappropriately programmed.

In our proposed system we are intending to build an automatic injection tool which will perform SQL Injection Attack on Java Server Pages only. We have hosted a java server page based web application in Tomcat server and are trying different methods of SQL injection attacks that can be performed on it. Here first the Blind SQL Manipulation attack (first order attack) is being tried on the page (which is basically a penetration type of attack) by inserting queries like ' or 1='1, admin' or 1='1, hi' or 1=1-- in login field or writing. The attacker attempts to modify the existing SQL statement by adding elements to the WHERE clause or extending the SQL statement with set operators like UNION, INTERSECT, or MINUS. All such queries were first inserted into a database and then they were extracted from there and given as input to various user login forms. Then secondly the Code Injection attack was tried. Code injection attacks attempt to add additional SQL statements or commands to the existing SQL statement like XXX'; drop table OrdersTable--. This type of attack is frequently used against Microsoft SQL Server applications, but seldom works with an Oracle database.

IV. Smart Injector Tool Development Methodology

Smart injector Tool is a type of SQL injection tool which does attack on Java Server Pages. SQL Injection Attack (SQLIA) can be considered as one of the top 10 web application vulnerabilities of 2007 and 2010 by the Open Web Application Security Project (OWASP).

Before going into the development of an injector tool we should first identify that what we are looking for? The page that allows one to submit data should be checked first like login page, search page, feedback, etc. Some of the HTML pages use POST command to send parameters to another ASP/ASPX/JSP page. Thus the parameters are not visible in the address bar of the browser. However, if we check the source code of the HTML, and look for "FORM" tag in the HTML code then we can find something like this in some HTML codes:

```
<FORM action=login.aspx method=post>
<input type=hidden name=user value=xyz>
</FORM>
```

Everything between the <FORM>; and </FORM> generally contain potential parameters that might be useful. Or we can look for pages like ASP, ASPX, JSP, CGI, or PHP. Try to look especially for URL that takes parameters, like: http://example.com/login.asp?id=10

Is it Vulnerable? Lets start with a single quote trick. This is otherwise known as Blind Injection attack. We can input something like: hi' or 1=1-- into login, or password field of the web page, or even in the URL. Example: Login: ' or 1=1-- Pass: ' or 1=1--. But why ' or 1=1--. Let us take a JSP page that will link us to another page with the following URL:
http://example.com/search.asp?category=sports. In this URL 'category' is the variable name and 'sports' is its value. This request fires following query to be executed on the database in background.

SELECT * FROM search WHERE category='sports'

Where 'search' is the name of a table which is already present in some database. So, this query returns all the possible entries from table 'search' which comes under the category 'sports'. Now, suppose we change the URL into something like: http://example.com/search.asp?category=sports' or 1=1--.Now, our variable 'category' equals to "sports' or 1=1-- ", which executes the following SQL query on database:

SELECT * FROM search WHERE category='sports' or 1=1--'

The query should now select everything from the 'search' table regardless if category is equal to 'sports' or not. A double dash "--" tell MS SQL server to ignore the rest of the query, which will get rid of the last hanging single quote ('). Sometimes, it may be possible to replace double dash with single hash "#". Depending on the actual SQL query, you may have to try some of these possibilities:

' or 'a'='a
' or 1=1--
" or 1=1--
or 1=1--
' or 'a'='a
" or "a"="a
') or ('a'='a
'or"='

Smart injector Tool is new scanning tool that is capable of conducting efficient penetration tests on JSP based websites for detecting the hidden SQL injection vulnerabilities as we described in the above paragraph. This tool involves a number of variables and features in the scanning process. It utilizes the aspect that forms the important feature of the attacking patterns used in this tool is the ability to trick the web server to display error

Table 1- features of the Smart injector tool

notifications whenever it detects an inappropriate programming. This tool utilizes a total of 10 such attacking patterns. In this tool if one pattern fails to expose the vulnerability, the other pattern will surely succeed in tricking the web server to display its vulnerability. Moreover, this strategy will not leave any possible vulnerability without exposing it. In addition, injecting the attacking patterns is done in different levels, the first level is to check the web path against SQL injection vulnerability, and the second level is to ensure that the website is vulnerable, the injection of the other type of attacking vectors starts to take place to check the database version of MYSQL using normal and encoded attacking patterns based on true/false responses and to analyze the website against forbidden protection and filters to prevent the attackers to execute
SQL commands, and to check whether these protections can be broken or not, and is
performed by injecting the website with the third level attacking pattern.

After checking the web server against the existence of the protections, Smart injector Tool tries to predict the number of infected columns in the database by injecting the fourth stage attacks, where these defective columns can be exploited by the attackers by executing several SQL queries through them to extract the data from the database to be shown on the web page. The prediction process may take only a few seconds because smart Scanner intends to calculate the byte size of each HTML page source after injecting the pattern using HTML parser technique and it starts comparing the results to locate the nearest one to the original page source that is always a true response. This is followed by, and then adding the number of defective columns to the URL, which will return a true value or will always return a false value or an error notification.
Then, the Smart injector tool will identify all the possible vulnerabilities present in the database of the website.

It include blind SQL injection using true/false response, blind SQL injection using true/error

SCANNING TOOLS	Variety Of Attacking Patterns	Classical Sql Injection	Blind/First Order SQL Injection Attack	Second Order SQL Injection Attack	Provide Detail Information	Cross Site Scripting Attack	Run in Real-time Server Side
SecuBat	✓	✓				✓	
SQL-IDS							✓
SAFELI							✓
ARDILLA	✓	✓				✓	
AMNESIA							✓
MYSQLInjector							
SmartScanner	✓	✓	✓	✓	✓		

Smart injector incorporates valuable features in shaping the attacking vectors used in the scanning process to increase its ability to trick the web server to display any vulnerability it has.

response, and blind SQL injection using order by, which are not used in the scanning tools listed in below table. In addition, using wide variety of client side tool, and the use of Smart Injector is not conducted frequently as the Intrusion Detection Systems (IDS). IDS systems are executed in the run time in every web page request and considered as server side code.

V. Results And Discussion

Our tool developed works in the following manner. It generates attack on the website whose URL is entered in the corresponding field. It follows a set of patterns for generating the attack and detecting the vulnerability.

attacking vectors and types that were designed keeping in mind the three motioned features through multiple levels with the high sensitivity of HTML parser in utilizing and analyzing the web server responses will certainly expose the Hidden SQL injection vulnerabilities. Moreover, using Smart Injector does not produce any additional load on the web server because it is operate remotely as

Here we first identified certain features which are inherent to such types of SQL Injection tools. They are listed below in Table-1 which also compares the various tools available with the Smart Injector Tool.

Fig.3

It does all the blind injection attacks and displays the vulnerability when it is found.

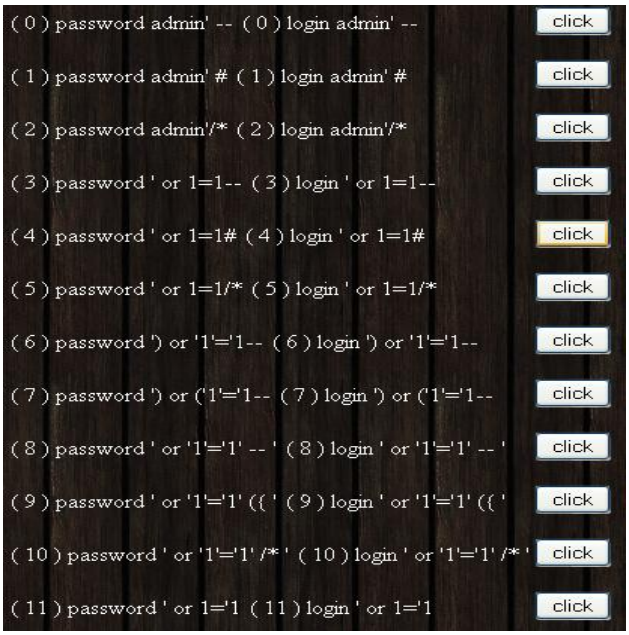


Fig.4

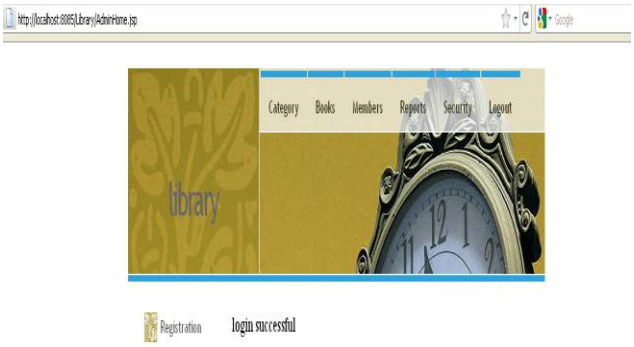


Fig.5

The above screen shot displays that tool has successfully made an SQL injection penetration attack on a library website that we had hosted on the tomcat server.

VI. Conclusion And Future Work

This paper we describes a new web scanning tool which will designed to perform all kinds of SQL Injection attacks like first order or SQL Manipulation attack, second order attack Code Injection attack[8] etc. By using this tool we can automate the penetration test and also able to test the Java Server Pages. Here we used an automated technique for detecting SQL injection vulnerabilities from Java code by converting plain text inputs received from users into prepared statements [7]

References

1.) Kals, S., Kirda, E., Kruegel, C., & Jovanovic, N. (2006). SecuBat: A Web Vulnerability Scanner. International World Wide Web Conference Committee IW3C2, 2, pp. 247-256, Edinburgh, Scotland.

2.) Kemalis, K., & Tzouramanis, T. (2008). SQL-IDS: A Specification-based Approach for SQL-Injection Detection. SAC '08. 2153-2158. Fertaleza, Ceara, Brazil.

3.) V. Cardellini, E. Casalicchio, M. Colajanni, P.S. Yu, ``The state of the art in locally distributed Web-server systems", *ACM Computing Surveys*, Vol. 34, No. 2, pp. 1-49, June 2002.

4.) Kiezun, A., Guo, P. J., Jayaraman, K., & Ernst, M. D. (2009). Automatic Creation of SQL Injection and Cross-Site Scripting Attacks. *ICSE '09*. 199-209. Vancouver, Canada.

5.) Halfond, W. G. J., & Orso, A. (2005). EMNESIA: Analysis and Monitoring for Neutralizing SQL-Injection Attacks. *ASE '05*, 174-183. Long Beach, California, USA.

6.) Abdul Bashah Mat Ali , Ala' Yaseen Ibrahim Shakhatreh, Mohd Syazwan Abdullah,

7.) Jasem Alostad (2011). SQL injection vulnerability scanning tool for automatic creation of SQL-injection attacks WCIT 2010 Procedia Computer Science 3 (2011) 453–458

8.) 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 November9, 2007, from http://ieeexplore.ieee.org.

9.) An Introduction to SQL Injection Attacks for Oracle Developers January 2004 Author: Stephen Kost Copyright © 2004 Integrigy Corporation. All rights reserved.

10.) R. A. Baker, "Code Reviews Enhance Software Quality," In Proceedings of the 19th international conference on Software engineering (ICSE'97) pp. 570 - 571, Boston, MA, USA. 1997

11.) B. Chess and G. McGraw, "Static analysis for security," IEEE Security and Privacy, vol. no. 6, pp. 76 - 79, 2004.

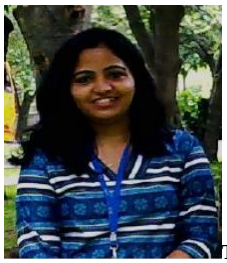
12.) An approach for SQL injection vulnerability detection MeiJunjin cims_hs@yahoo.cn Huangshi Institute of Technololgy, The north street of Guilin ,Huangshi,Hubei province,China,435003

13.) D. Stuttard and M. Pinto, "The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws", Wiley Publishing Inc., 2007

14.) P. Herzog, "OSSTMM 2.2. Open-Source Security Testing .

First Author: Mr. Praveen Kumar: I was Born In Sultanpur(U.P.), India. I Received My B.TECH Electronic and Communication Engineering Degree From Integral University,Lucknow,U.P. , India in 2010. Currently I Am Pursuing My M.Tech In Software Technology At Vit University, Vellore, Tamil Nadu, India. I am a good player in Badminton and Cricket. My Major Research Interests are in Information Security,Software Development and Testing,Cloud Computing,Operating System,Data Structure and Algorithm, RDBMS, AGILE Methodology . I have Completed Successfully Research Project Entitled SQL-Injection Vulnerability Scanning Tool for Automatic Creation of SQL-Injection Attack on JSP.Currently I am doing my final project on Design and Development of Cloud-Computing Based District Resource Planning System.

Second Author : Himanshu Kumar : I have completed my B.E in computer science and Engineering Degree From Manipal University,Manipal,Karnataka,India in 2009. Currently I am pursuing M.Tech In information Technology(Networking) at VIT University,Vellore,Tamil Nadu, India.My Major Research area are Information Security,Software Development and Testing, Cloud computing ,wireless adhoc network,Operating System,Data Structure and Algorithm, RDBMS. I have Completed Successfully Research Project Entitled rainbow table to crack password using md5 hashing algorithm..Currently I am doing my final project on web usage mining trends and navigational pattern.



Third Author: Remya Joseph: I have completed my B.E in computer science and Engineering Degree From Mahatma Gandhi (MACE) University,kothamangalam,Kerala,India in 2010. Currently I Am Pursuing My M.Tech In information Technology(Networking) at Vit University,Vellore,Tamil Nadu, India.My Major Research area are in Information Security,Software development and Testing,Cloud computing,wireless adhoc network,Operating System,Data Structure and Algorithm, RDBMS. I have Completed Successfully Research Project Entitled Efficent web usage mining in formal concept analysis.Currently I am doing my final project on automating IOT Cases in Robot Framework.