ISSN:0975-9646

A Rule Based Verification with Strongly Typed Field based SQL Injection and XSS Attack Detection

Ajay Singh Dikhit [1], Prof.khushboo Karodia [2]

[1]*PG Scholar, Department of Computer Science & Engineering*
Shri Vaishnav Institute of Technology and Science, Indore (M.P), India

[2]*Professor, Department of Computer Science & Engineering*
Shri Vaishnav Institute of Technology and Science, Indore (M.P), India

Abstract: **Computing technologies and evolution of internet reduces the effort required for various processes. Among them the most benefitted industries are web based systems, banking, marketing, e-commerce etc. This system mainly involves the information exchanges continuously from one host to another. During this transition there are so many places where the confidentiality of the data and user gets loosed. Normally the area where there is more probability of attack occurrence is known as vulnerable zones. Web based system interaction is one of such place where multiple users performs there task according to the privileges assigned to them by the administrator. Here the attacker makes the use of open areas such as login or some other places from where the malicious script is inserted into the system. This scripts aims towards compromising the security constraints designed for the system. Few of them related to users inserted scripts towards web interactions are SQL injection and cross site scripting (XSS). Such attacks have to be detected and removed before they make an impact on the privacy and confidentiality of the data. During the last few years various solutions have been integrated to the system for making such security issues resolved on time. Input validations is one of the well known fields but suffers from the problem of performance drops and limited matching. Some other mechanism such as sanitization and tainting will generate high false report showing the misclassified patterns. At the core, both involve string evaluation and transformation analysis towards un-trusted sources for completely interpreting the impact and depth of the attack. This work proposes an improved rule based attack detection with strongly typed text fields for effectively detecting the malicious scripts. The work blocks the normal access for malicious source using and robust rule matching through centralized repository which regularly gets updated. At the initial level of evaluation, the work seems to provide a strong base for further research.**

Keywords- **: Web Security, SQL Injection, Cross Site Scripting (XSS), False Report, Buffer Overflow, Input Validations, Rule Based Attack Detection;**

I. INTRODUCTION

Software vulnerability means the probability of error or theft occurrence which might affect the normal operations of the system. Mainly, it deals with the attackers unauthorized interventions with the system. As the number of software based system gets increased the vulnerability graph is also rising with an abrupt pace. Additionally, as the number of users gets exposed more to the open system like internet, the chances of such attack will raises. Attacker can start any attack sequence from the remote locations for destroying the normal operations of the system. In most of the cases vulnerabilities are caused by improper validation of the data supplied by the user [1]. This undesired condition is used by attackers to inject faults and malicious code into the system that allows them to run their own code and applications. For better understanding vulnerabilities the creation of models that express the set conditions that could lead or originate them is very helpful; additionally when models are well understood they could also be used for prevention. But since it is impossible to guarantee the absence of vulnerabilities in a piece of code during its creation, then it is necessary to have methods to detect them.

Mostly the attacks are probable to open environment based web applications because it is next to impossible to verify the authenticity of user and its operations both simultaneously. There is a wide variety of web based users activity verification techniques for securing such a vulnerable access structure. Web applications enable much of today's online business including banking, shopping, university admissions, email, social networking, and various governmental activities [2]. They have become ubiquitous because users only need a web browser and an Internet connection to receive a system-independent interface to some dynamically generated content. Figure 1 shows typical system architecture for web applications. This three-tiered architecture consists of a web browser, which functions as the user interface; a web application server, which manages the business logic; and a database server, which manages the persistent data.

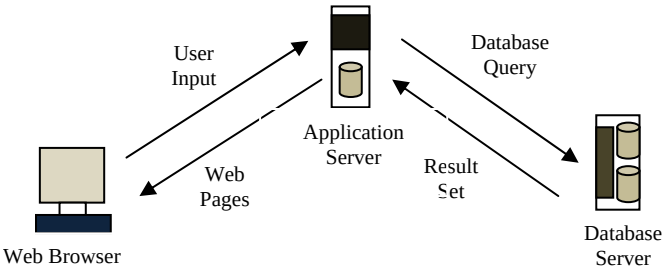

FIGURE 1: SYSTEM ARCHITECTURE OF WEB APPLICATIONS

The web application server receives input in the form of strings from both other tiers: user input from the browser and result sets from the database server. It typically incorporates some of this input into the output that it provides to the other tiers, again in the form of strings: queries to the database server, and HTML documents to the browser, both of which get executed by their respective tiers. The web application server constructs code dynamically, so the code for the entire web application does not exist in any one place at any one time for any one entity to regulate. The flow of data among tiers gives rise to the input validation problem for the web application server; it must check and/or modify incoming strings before processing them further or incorporating them into output that it passes to other tiers to execute. Failure to check or sanitize input appropriately can compromise the web application's security.

II. BACKGROUND

Models are a first approach to deal with vulnerabilities and their understanding. A vulnerable software system can be exploited by attackers and the system could be compromised, the attacker might take control of the system to damage it, to launch new attacks or obtain some privileged information that he can use for his own benefit. Considering this, it is important to know the different types of vulnerabilities, their prevention and detection in order to try to avoid their presence in the final software version of the system and then reduce the possibility of attacks and costly damages. Considering the vulnerabilities towards making the web based system security is input validation. The two most prominent classes of input validation errors are cross-site scripting (XSS) and SQL injection. XSS and SQL injection are the classes of vulnerabilities in which an attacker causes the web application server to produce HTML documents and database queries, respectively, that the application programmer did not intend [3]. They are possible because, in view of the low-level APIs described above for communication with the browser and database, the application constructs queries and HTML documents via low-level string manipulation and treats un-trusted user inputs as isolated lexical entities. This is especially common in scripting languages such as PHP, which generally do not provide more sophisticated APIs and use strings as the default representation for data and code. Some paths in the application code may incorporate user input unmodified or unchecked into database queries or HTML documents. The modifications/checks of user input on other paths may not adequately constraint the input to function in the generated query or HTML document as the application programmer intended. In that sense, both XSS and SQL injection are integrity violations in which low-integrity data is used in a high-integrity channel; that is, the browser or the database executes code from an un-trusted user, but does so with the permissions of the application server. However, both problems involve more than naive integrity level or taintedness tracking because the output gets parsed and interpreted rather than treated as an atomic value.

Typical uses of SQL injection leak confidential information from a database, by-pass authentication logic, or add unauthorized accounts to a database [4]. Although XSS vulnerabilities are more prevalent than SQL injection vulnerabilities, some XSS vulnerabilities cannot be exploited in damaging ways, whereas most SQL injection vulnerabilities can. Typical uses of XSS attacks leak information, such as authentication credentials to a bank website; exploit browser vulnerabilities and perhaps load other malware onto users' systems; or contribute to a social engineering effort to trick users into revealing information, such as passwords. Not only can a single XSS exploit do the things listed above, but XSS vulnerability can be used to create rapidly spreading malware, and thus compound multiply the damage caused by a single exploit [5].

SQL Injection [6] is only one of the many types of injection attacks which mainly occurs on database driven websites. Here the attacker executes unauthorized SQL commands by taking advantage of insecure code on a system, bypassing even deeply nested firewall environments. It causes:

- Bypass Authentication
- Discover Databases (Schemas, Users, Columns, Values)
- Data Disclosure
- From the Database to the Network

Cross site scripting (XSS) [7] essentially applies code injection attacks into the various interpreters in the browser using HTML, JavaScript, VBScript, ActiveX, Flash and other client-side languages. It is an attack on the privacy of clients of a particular web site which involves 3 parties – attacker, a client and the vulnerable website for stealing or manipulating customer's data It causes:

- Steal Authentication Information and Account hijacking
- Changing of user settings
- Cookie theft/poisoning
- Denial of Service - Website Defacement
- Phishing (Pharming and Phear) – embedded links
- Identity Theft! (Impersonation)

Thus a major category of web abased attacks are applied using the above two mechanism. Thus, it is very essential to develop a defence mechanism against both.

III. LITERATURE SURVEY

During the last few years various modifications are performed for improving web security mechanism. Among them the related approaches to the SQL injection and cross site scripting are taken here as a survey.

In the paper [8], an approach toward making the system more robust against the SQL injection and cross site scripting (XSS) detection is suggested. The paper says that the affection vulnerabilities could be made reduced by introducing the filtering approach. The detection filter identifies the maliciousness in the entered text string. Even though the attacker makes a slight changes in the string for making it undetected from the filtering mechanism. Evaluating the approach, it is found that a HashMap is able to provide an easy to implement, a fast running time sanitization function to detect encoded SQL injection and XSS attack strings. A proof of concept is developed in PHP and Java.

The paper [9] presents cross site scripting detection and removal approach. It also presents the comparison of the current filtering and sanitization techniques. According to that XSS sanitization can be difficult to get right as it ties in closely with the parsing behavior of the browser. An improvement over the sanitization correctness using parsing behaviour of the web is also used as a filter for browser. A system administrator may assign HTML filters based authenticated user role as a control structures. However, the administrator can create arbitrary HTML filter policies and assign them to various roles. Thus, the administrator can apply the full HTML filter to all users by default, but explicitly allow trusted users to post comments with links in HTML anchor tags.

The paper [10] presents a study on cross-site scripting on web applications over the last few years. It focuses on developing strong security primitives for web protection against the unauthorized access of the protected content. It also shows performance comparison of two mitigation techniques for Cross-site Scripting (XSS) at the server side based on the parameters like application's platform, middleware technology and browser used by the end user. The paper had implemented Mitigation parsing technique using database and replace technique in different platforms, middleware and checked its performance. From all the results obtained can clearly infer that a web application with mitigation technique with REPLACE function implemented in JAVA can give a good performance in browser FIREFOX.

In the paper [11], a detailed survey and comparative is presented on various kinds of SQL injection, XSS attacks and approaches to detect and prevent them. Approaches to detect SQL Injection for a web application can be protected from attacks by taking into account two major things: Detection algorithm and vulnerability analysis. All the approaches however involve with a limited detection up to some failures at some point. Any unified approach is still not been developed because of its wide applicability. Furthermore some of them even do not provide reasonable security and can be easily bypassed by attackers. Also a few of them are so complex it is almost impractical for a user to use in real situations. The key finding of this paper is analytical survey report on various types of SQL injection and XSS attacks against different methods defined by several authors.

In the paper [12], an automated testing approach, namely µ4SQLi is suggested with its groundwork set of conversion operators. It can generate the input sets which make the systems working affected by executing the systems malicious SQL statements. The paper specifically focuses on the input vulnerabilities using SQLi service under test. If it is used in any SQL statement of the execution of a service and if, through this parameter, an attacker can send malicious inputs that can change the intended logic of the SQL statement. To develop such vulnerabilities, the attacker has to provide inputs that result in executable SQL instruction. Otherwise, the resulting statements are discarded by the record, thus no access or changes to the data are possible.

In a way to achieve its objective of dynamic attributes based vulnerability detection the paper [13] uses supervised learning. The work is a prediction models that are based on classification and clustering in order to predict vulnerabilities, working in the presence or absence of labeled training data, respectively. In this experiment across six applications, the new supervised vulnerability predictors based on hybrid (static and dynamic) attributes achieved, on average, 90% recall and 85% precision, that is a sharp increase in recall when compared to static analysis-based predictions. Here the attributes are based on hybrid static and dynamic code analysis, which characterize input validation and sanitization code patterns for predicting SQL injection and XSS vulnerabilities. Dynamic analysis is used to classify nodes that invoke user-defined or language built-in string replacement/matching functions since classification of such nodes by static analysis could be imprecise.

The paper [14], proposes a lightweight approach to prevent SQL Injection attacks, that it can actually be well defended by using LINQ (Language Integrated Query). LINQ to SQL, when used exclusively for data access, eliminates the possibility of SQL injection in your application for one simple reason: every SQL query that LINQ executes on your behalf is parameterized. Internally, it means that when LINQ to SQL queries the database, instead of using plain values, it passes them as SQL parameters, which means they can never be treated as executable code by the database. The compiler catches a lot of query misuse that might introduce functional defects or other types of vulnerabilities into your application. In contrast, SQL statements you write are parsed and interpreted on the database only at runtime before you know whether it is correct or not. The only attack vector against LINQ to SQL is for an attacker to try to trick LINQ into forming illegal or unintended SQL. Fortunately, the languages and compilers are designed to protect you from that.

The paper [15] covers certain issues and suggest there solution for making the defence more robust against dynamic SQL injection attack. Due to the 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. In order to handle this vulnerability and detect it, enhancement in the coding structure used for web application development is required. It is used to dynamically mine the programmer - intended query structure on any input, and to detect attacks by comparing them against the intended query structure. 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.

IV. PROBLEM STATEMENT

After studying the various approaches related to the defined attack prevention, detection and removal, some of the definite are of work identified is covered here. Mainly the focus with the defined area is to reduce the detection load on performance measures and will increase the

detection accuracy. It depends on the browsers also thus, it needs to be taken care with other factors for a web applications.

Problem (i) Existing sanitisation techniques will only uses limited matching for SQL injection and XSS attacks. There is no central repository holding the complete verification for pattern inserted and will fully automate the process. It causes false report for recent developing attack scripts.

Problem (ii) Large character probability with inputs will increase the probability of scripting attacks. Thus a formal code development and inspection after the complete code will work towards improvements. Always the input must be applied with constraints with definite fields especially with the password or user id texts.

There must be clear separation between the trusted user input and untrusted user input. Thus, some mechanism will work prior before inserting those codes to the system. A focus should also be made towards making the cookies also secure to interpret. Thus context sensitivity will work in this direction. With this work the aim is towards reducing the performance penalty for above detection approaches using light weighted pattern matching operations.

V. WORK EMBODIED

After analysing the various related works towards improving the SQL injection and cross site scripting (XSS) attack detection, the objective of the work are given below.

(i) This work will suggest an automated input validation in combination of other preventing measures for both the attack. Mainly the directions are leading towards sanitization and taint analysis based detection in a combination or hybrid way.

(ii) Work will present a brief study on above mentioned types of attack so as to completely analyze the remaining problems in defense mechanism development.

(iii) The direction of work will also include a visualization mechanism for vulnerability analysis.

(iv) Evaluation of above suggested approach with a futuristic development to make the logical views on strong code base background.

VI. PROPOSED WORK

In today world the web application is changing very rapidly with the use of internet based application and services. This includes the computation process with the secure transition of the user's data. Controlling the operation of computation and other web operation are based on three tier architecture. In this the initial request is made by the end user who processes and forwarded to the application server from where it reaches the original location where the data resides. The data is then reverting back with the request to the source address. During this various types of information theft possibilities are probable on such system. But there is various security mechanisms are developed during the last few years to make the process overall secure. Unauthorized theft of the information comes under the category of the attack. With evolutionary growth of internet based web application the types and variety of the theft occurrence and areas are also increased. Thus to make a robust and overall mechanism for complete security is not practically feasible. Among the traditional security mechanism for web mostly are handled by the web service providers. But still the users controlled information flow is an open area of attack fabrications and insertions. SQL injection and Cross site scripting is the most vulnerable category which is planted by the users text field using an executable queries.

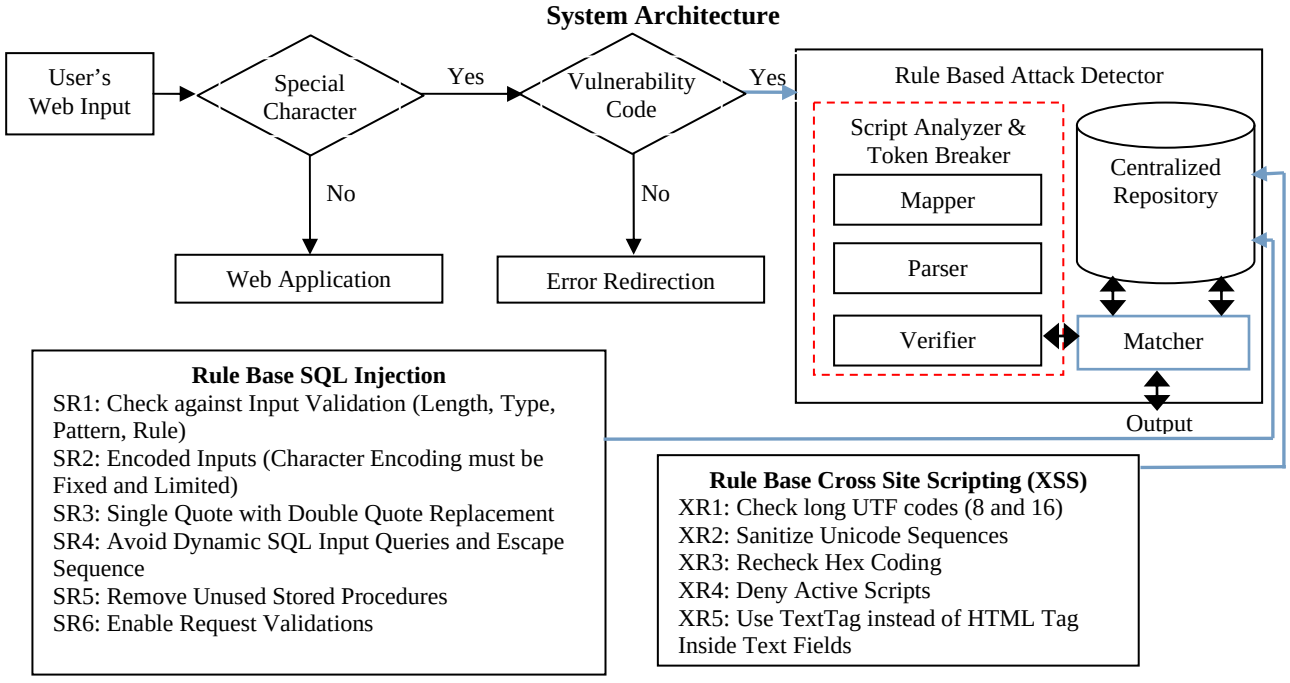

FIGURE 2: A RULE BASED ATTACK DETECTION APPROACH TOWARDS THE SQL INJECTION AND XSS ATTACK

Thus work proposes an improved mechanism using rule based verification with strongly typed text field based SQL injection and XSS attack detection for web browsers. The system performs its operation based on four major components or functionalities:

(i) Script Collector: It is used for collection of script from the various input source of the web. Mainly it handles the character matching for any susceptible codes containing special symbol. It detects whether the symbol inserted is intentional or mistaken and redirects it accordingly.

(ii) Vulnerable Script Pattern Detection: It performs the task of vulnerable script analysis. It can be segregated into mapping, parsing, verifying from central repository.

(iii) Rule Base for SQL Injection: It contains the rules and definition of newly introduced attacks.

(iv) Rule Base for XSS Attack: It contains the rules and definition of newly introduced attacks.

Web application always suffers with the recent attack vulnerability of SQL injection and cross site scripting (XSS). Both the category of attack contains the broader view of user input based text validations because from this instance the system gets affected easily. The process starts with the user gets the system input into the system. It is the open area where if the constraints are not applied then the user is free to apply any type of text into the insertion fields. Here the script can also be passed which could work as a malicious code.

This script based text contains different symbols from which various data orientated things can be accessed maliciously from the web database. Here the system gets a specified special character matching system. This makes the uneven and unidentifiable script and bock the access. If the string is generic then its gets permitted for web application access. If the attacker have modified the string other than the constrained applied with the system or if the normal user insert the same query formally then a further check is made with the vulnerable code detection module working with the browser.

If vulnerability is not found than the user gets permitted to access the web normally. Otherwise the rule based attack detection starts operating. This rule based attack detection contains the script analyser and token breaker module. Here the script is broken into various token which is analysed by the later functionality. Verification system is embossed with three operations: Mapper, Parser and Verifier. All this works collectively for making a complete breaking and analysis of the inserted script. This script after breaking is matched with the central repository of older attack and recently identified attack sequences. This central repository is regularly updated with new definitions and scripts. And if the script contains a portion of the vulnerable string then also it blocks the request. Repository contains the rules from which matching is applied.

The request which violates the rules will be considered as a SQL injected and XSS attack. In this way the system detects the attack as a pattern. If some new attack is detected then also the system gets defined with the new rule easily. The system is easy to use and occupy fewer resources.

APPLICATION AREA

(i) Dynamic web security
(ii) Web services monitoring
(iii) Web performance logger
(iv) Cloud based web computing
(v) Blogging
(vi) Social networking
(vii) Transaction Systems
(viii) Online services marketing

VII. EVALUATION PARAMETERS

With this work the aim is to perform binary classification to predict vulnerable files. A binary classifier can make two possible errors: false positives (FP) and false negatives (FN). A FP is the classification of a neutral file as a vulnerable file, and a FN is the classification of a vulnerable file as neutral. A correctly classified vulnerable file is a true positive (TP), and a correctly classified neutral file is a true negative (TN). A FP may cause a team to do additional testing or inspection of a file only to find no vulnerabilities. A FN may allow a vulnerability to escape to the field after release. For evaluating binary classification models, the work uses recall, precision, inspection rate, and vulnerability rate.

- **Recall (R)** is defined as the percentage of vulnerable files found: R=TP*100/(TP+FN).
- **Precision (P)** is defined as the percentage of correctly predicted vulnerable files:
  P=TP*100/(TP+FP).
- **Inspection rate (IR)** is the percentage of files that were classified as vulnerable:
  IR=(TP+FP)*100/(TP+TN+FP+FN).
- **Vulnerability rate (VR)** is the percentage of the number of vulnerabilities that were included in the files classified as vulnerable:
  VR = (No. of Vuln. in the files classified as vulnerable)*100 / (Total No. of vulnerabilities).

The inspection rate represents the percentage of files required for inspection or testing to achieve the reported recall. Note that VR is similar to recall, but differs where recall counts the number of vulnerable files, VR counts the number of vulnerabilities in each file that was classified as vulnerable.

III.CONCLUSION

Web application development and usages are increased very drastically due to their availability and trust over the system which could be remotely measured. This trust and usages are increase because of their high security and reduced load. But during the last decade this online applications gets affected with SQL injection and cross site scripting (XSS) attacks. This is because the data oriented

web applications host the user's data and somewhere it interacts with the text based user fields and commands. This open area is used by the malicious user to get into the system using the above defined attack sequences. Traditional approaches which are working towards detection are consuming heavy system load. This paper works towards improving the traditional attack detection mechanism with improved accuracy and reduced computation load. The detection and attack mitigation is completely automated with the suggested rule based attack detection having strongly typed fields for preventing the malicious script to be inserted into the system. Analytical evaluations and future direction of the work will prove the same with a implemented system.

ACKNOWLEDGMENT

The authors wish to thanks Dr.Anand Rajavat and Dr.Rajeev Vishwakarma for their support in this work.

REFERENCES

[1] Nenad Jovanovic, Christopher Kruegel and Engin Kirda, "Precise Alias Analysis for Static Detection of Web Application Vulnerabilities", in ACM PLAS, DOI: 1-59593-374-3/06/0006, 2006.

[2] Elizabeth Fong, Romain Gaucher, Vadim Okun and Paul E. Black, "Building a Test Suite for Web Application Scanners", in National Institute of Standards and Technology, 2008.

[3] Etienne Janot and Pavol Zavarsky, "Preventing SQL Injections in Online Applications: Study, Recommendations and Java Solution Prototype Based on the SQL DOM", in OWASP Security Conference, Belgium, 2008.

[4] Yonghee Shin, Laurie Williams, "Toward A Taxonomy of Techniques to Detect Cross-site Scripting and SQL Injection Vulnerabilities", in North Carolina State University, Raleigh, 2008

[5] Michael Martin and Monica S. Lam, "Automatic Generation of XSS and SQL Injection Attacks with Goal-Directed Model Checking", in USENIX Security Symposium, 2008.

[6] Mattia Monga, Roberto Paleari and Emanuele Passerini, "A hybrid analysis framework for detecting web application Vulnerabilities", in University degli Studidi, Milano, Italy, 2009

[7] Raymond Mui and Phyllis Frankl, "Preventing SQL Injection through Automatic Query Sanitization with ASSIST", doi:10.4204/EPTCS.35.3, 2010

[8] Erwin Adi and Irene Salomo, "Detect and Sanitise Encoded Cross-Site Scripting and SQL Injection Attack Strings Using a Hash Map", in Proceedings of the 8th Australian Information Security Management Conference, Edith Cowan University Research Online, 2010

[9] Joel Weinberger, Prateek Saxena, Devdatta Akhawe, Matthew Finifter, Richard Shin and Dawn Song, "An Empirical Analysis of XSS Sanitization in Web Application Frameworks", University of California, Berkeley, 2011

[10] Ravi Kanth Kotha, Gaurav Prasad and Dinesh Naik, "Analysis of XSS attack Mitigation techniques based on Platforms and Browsers", in Department of Information Technology, National Institute of Technology, Karnataka, DOI : 10.5121/csit.2012.2240, 2012

[11] Abhishek Kumar Baranwal, "Approaches to detect SQL injection and XSS in web applications", Term Survey Paper in Masters of Software Systems, University of British Columbia, 2012

[12] Dennis Appelt, Cu Duy Nguyen, Nadia Alshahwan and Lionel C. Briand, "Automated Testing for SQL Injection Vulnerabilities: An Input Mutation Approach", in ACM, DOI:978-1-4503-2645-2/14/07, 2014

[13] Lwin Khin Shar, Lionel C. Briand and Hee Beng Kuan Tan, "Mining SQL Injection and Cross Site Scripting Vulnerabilities using Hybrid Program Analysis", in ICSE 2013, IEEE, DOI: 978-1-4673-3076-3/13, 2013

[14] V. Vilasini and P. Chellamal, "Eliminate Sql Injection Using LINQ", in IJARCST, ISSN : 2347 - 8446 (Online), Vol:2, Issue:1, 2014

[15] Praveen Kumar, Himanshu Kumar and Remya Joseph, "Sql-Injection Tool for finding the Vulnerability and Automatic Creation of Attacks on JSP", in IJARCET, ISSN: 2278 – 1323, 2012