AUSTRALIAN JOURNAL OF BASIC AND APPLIED SCIENCES

ISSN:1991-8178 EISSN: 2309-8414
Journal home page: www.ajbasweb.com

Detection and Prevention of Cross Site Scripting Attacks using Concolic Testing and Pattern Filtering Approach in Web Application

[1]A. Duraisamy and [2]Dr. M. Subramaniam

[1]Research Scholar, Department of Information Technology, University College of Engineering Tindivanam, Tamilnadu, India.
[2]Professor and Head, Department of Information Technology, S.A. Engineering College, Chennai , Tamilnadu, India.

Address For Correspondence:
A. Duraisamy, Research Scholar, Department of Information Technology, University College of Engineering Tindivanam, Tamilnadu, India.
E-mail: durai.ucet@gmail.com

ARTICLE INFO

Article history:
Received 19 September 2016
Accepted 10 December 2016
Published 31 December 2016

Keywords:
Cross Site Scripting Attack, Web Security, Concolic Testing, Pattern Filtering, Reflected and Stored XSS Attacks, JCUTE Tool

ABSTRACT

This paper focuses on detecting and preventing the cross site script attacks in web application. 80 percent of the web applications are vulnerable to security threats, as based on the survey conducted by Open Web Applications Security Project (OWASP). Cross-Site Scripting (XSS) vulnerabilities are due to the lack of input validation that allow attackers to insert malicious scripts in user input and the script is executed at other end. This is frequently found within web pages with dynamic content and it carry out different malicious operations like hijacking user sessions, defaces web sites, redirect the user to malicious sites, password theft etc. In this paper, we detect and prevent the cross site scripting attack in two phases. In first phase, user given URL is extracted and tested for vulnerability using concolic testing approach. It compares concrete and symbolic values and as a result the vulnerable URLs are sent for prevention. In second phase, the URLs whose vulnerability is unknown are injected into Information Leakage Calculator and the decision is taken based on threshold value. The detected XSS attack URLs are prevented using pattern filtering approach. The way of preventing the XSS attack shows the proposed solution effectiveness and convenience.

INTRODUCTION

Companies and Organizations uses web applications to provide better service to the end users. The Database used in web applications frequently contains confidential and Personal information. The Cross-site Scripting (XSS) vulnerabilities are target to be these databases and user personal information. Web applications are interact with back end database to recover persistent data and then present the data to the user as generated output, such as HTML web pages. This is done through a low-level API by dynamically constructing query strings with in a programming language. The structure of the output language does not change at any cost.

Cross-site Scripting (XSS) vulnerabilities are on web applications in which an attacker gets control of a user's browser. That returns a malicious script (usually an HTML/JavaScript code) within the context of trust of the web application's site. The attacker may be able to access, passively or actively, to any sensitive browser resource connected to the web application (e.g: cookies, session IDs, etc.) when the embedded code is successfully executed. We study in the result two main types of XSS Attacks: persistent (stored) and non-persistent (reflected) XSS attacks.

Cross-site Scripting (XSS) vulnerabilities are due to the lack of input validation that allow attackers to insert malicious scripts in user input and the script is executed at other end. We supply a simple example of attacks exploiting XSS vulnerabilities. Examine a search engine that yields the search results contain the same query given by a user. If the user input contains a script and the returned result page does not encode the script

To Cite This Article: A. Duraisamy and Dr. M. Subramaniam., Detection and Prevention of Cross Site Scripting Attacks using Concolic Testing and Pattern Filtering Approach in Web Application. ***Aust. J. Basic & Appl. Sci.,*** *10(18): 66-73, 2016*

into HTML code, the script in the result page will be executed. For example, if user entered an executable script as a query for a search engine that returns with the same query, which is script in this case, the browser will carry out the script and exhibit an alert with a "Hello" message.

Cross Site Scripting attacks can be used to access user's critical information by Stealing HTTP cookies. When a web application wants to keep track of a Communication channel between a web server and a user's browser, a Web Server sends a HTTP cookie that contains the information that the server can identify the user. If an attacker can access the cookie at user's browsers, the attacker can false appearance as the user and can access critical information such as credit card number that only the user was assumed to access. Web sites on which users can publish messages to a message board shared with other users and that require users to login to the system to use the message board are latent vulnerable to this XSS attack. A procedure of cookie stealing is as follows. An attacker finds a web site on which users can post message to a message board shared with other users and that requires users to top, in to the system to use the message board. The attacker posts the script that sends cookie information to the attacker's web site when the script is executed. A user logs in to the web site and reads the message that the attacker has posted. The script is executed and the cookie information of the user is sent to the attacker's site. The attacker can access the user's critical information using the cookie.

Nowadays almost in touch of computer is joined online and to serve millions of user's vast amount of data is stored in databases joined to some web application all across the globe [1]. These users are inserting querying and updating data in these databases. A well- designed user interface is very important for all these operations to occur successfully. Moreover, these applications play an important role in maintaining security of data stored in these databases. Very unsecured web applications are allow well-designed injection to perform unwanted operations on backend database.

An unauthorized user may be able to distinguish this situation by damaging or with theft of trusted users sensitive data stored here. An attacker gains full control over a database or web application as there is a chance of system being fully destroyed is produced the maximum damage. The sensitive data is transferred from one's system to any third party by injecting vulnerability in trusted sites' dynamically generated pages. That third party could be an attacker's server.

In this proposed work user credentials are provided as the input and fetch the relevant data in the database is validated to each of the request and will be authenticated each using the XSS parsing and Script validation. In the proposed system, the user input is validated with Parser and Leakage evaluator. An alarm will be issued to each of the request and XSS to the administrator and the input is also validated in our new system. Our system provides a security to the XSS with more efficiency. This will overcome the performance issue of the already existing system.

Literature Survey:

Imran Yusof et al have Preventing Cross Site Scripting Attack by Applying Pattern Filtering Approach. They proposed a simple filtering technique by filtering input variables as Event Handler, data URI, Insecure Keywords, character encaping, common words in XSS payload and XSS buddies. Mukesh Kumar Gupta et al proposed a Static Analysis Approaches to Detect SQL Injection and Cross Site Scripting Vulnerabilities in Web Applications: A Survey, This paper introduces a classification of security approaches (static, dynamic and hybrid) used to develop secure software in various phases of software development life cycle process.

Binbin Qu, et al proposed a design of Automatic Vulnerability Detection System for Web Application program. In this paper, the attack mode knowledge base and the automata operation library, the vulnerability detection engine reports the presence of vulnerability in the web application. Michelle E Ruse et al , Proposed a system for Detecting Cross-Site Scripting vulnerability using concolic Testing. A concolic testing-based technique for detecting possible vulnerable outputs (those which are directly and indirectly assigned from the user inputs and can contain scripting tags) followed by selective instrumentation for runtime XSS attack detection. Scott and Sharp describe a web proxy that is in between the users and the web application, and it makes sure that a web application attaches to pre written security policies. The main merits and demerits of such policy based approaches is that the creation and management of security policies is a tedious and error-prone task.

Junjin et al proposed a commercial product called AppShield , which is a web application firewall proxy that plainly does not need security policies. AppShield request that it can automatically lessen web threats such as XSS attacks. It is used to learn from the traffic to a specific web application. Because the product is closed-source, it is impossible to verify this claim. Moreover Amutha Prabakar reports that AppShield is a plug and play application that can only do simple checks and it can only provide limited protection because of the insufficiency of any security policies. A client-side solution is the main difference of our approach with respect to existing solutions. The solutions presented in Amutha Prabakar are both server-sides that target are to protect specific web applications.

Munqath et al explain the use of a number of software-testing techniques (including dynamic analysis, black-box testing, fault injection and behaviour monitoring) and introduce mechanisms for applying these

techniques to web applications. The target is to find out and fix web vulnerabilities such as XSS and SQL injection. The target of the presented work is the web application development community. Ajax, which is a modern development in web application that uses asynchronous JavaScript and XML, permits a web application to send and receive data through a XML HTTP request - with no page refreshing. Ajax contains Ajax-based client, which includes page-specific control logic embedded as JavaScript technology. The page communicates with the JavaScript based on events such as the document being loaded, a mouse click, mouse over or focus changes etc. Ajax is a term coined by Jesse James Garrett during 2005.

Engin Kirda et al and O.Ismail et al introduced a client side solution that fully relies on the user's configuration and number of researches has proven that client side solution is not reliable. If a new vulnerability is introduced, the new fix introduced at a central server to stop the hacking cannot defend the user immediately as it needs an update on the client side system. Further according to Kruegel et al, it is not possible to maintain the misuse type IDS (IDS are categorized basically into misuse and anomaly) due to the large dynamic signature in an everyday attack scenario. CERT- Centre of internet security expertise, a federally funded research and development centre states that none of the client side solutions prevent the vulnerabilities completely and it is up to the server to eliminate these issues.

There are entirely a few solutions introduced on the same lines of research. Wes Masri and Andy Podgurski have specified if the information flow is high that information flow based work will enhance the false positives and it is not a suggestive strength. There are validation mechanisms and scanners are launched to stop XSS vulnerabilities. Some software engineering approaches are also launched such that WAVES for security assessment. If tags are allowed in the web applications although none of the solutions are not built for the recent developments and would fail. Also, all the solutions explained above are prone to zero-day attacks. Jayamsakthi et al have provided a solution that is based on the financial and non-financial applications but this does not provided for the XSS attacks emerge from various interfaces.

Server-side XSS Detection System (XSSDS) is based on passive HTTP traffic monitoring and relies upon the firm connection between incoming parameter and reflected XSS issues. The set of all legitimate JavaScript's in a given web application is limited. This makes the basis for two novel detection approaches to establish successfully execute reflected XSS attacks and to find out stored XSS code on the server side. Static analysis techniques analyze program code contains source code, byte code, or binary code. It is used to learn how the control or data would flow at runtime without running the code. Some techniques cannot find out the existence of input validation routines and result false positives due to the complexity and technical limitations. On conclusion, XSS attack is the more common type of attack that acts as a serious problem to the web applications. This type of attack can be detected and prevented using various tools developed by the researchers. We have developed a system that detects the XSS attacks efficiently and the detected vulnerabilities are prevented using an efficient algorithm. Our system will overcome the performance issue of the already existing system.

Implementation And Discussion:

1. Proposed System:

A concolic testing-based technique for detecting possible vulnerable outputs (those which are directly and indirectly assigned from the user inputs and can contain scripting tags) followed by selective instrumentation for runtime XSS attack detection. concolic testing is hybrid software verification technique that performs symbolic execution .This testing is only applied for jsp web pages in the existing system it is not an effective one, then it is converted into java code and then use the existing detection tool JCUTE(Java concolic Unit Testing Tool) for testing the occurrence of XSS attack. In the existing system for prevention of cross-site scripting attack they used pattern filtering approach .Pattern filtering approach is an effective way to preventing this type of attacks but in the existing system which only prevent the persistent type of cross-site scripting attack.

In our work, we detect and prevent the cross site scripting attack in two phases namely Detection and Prevention Phase. In Detection Phase, we get the URL which the user need to access and is passed to the URL extractor. The URL extractor split the given URL into protocol, host, path, query and reference and the extracted components are stored in the Symbolic table. The host of the URL in the Symbolic table is compared with the Concrete table which contains list of malicious link using Comparator. If the given URL is matched with the existing malicious URLs, then it is stored in a trace file. Based on the Trace file, the Flaw Detector detects the flaws in the URL, and then it is treated as malicious. If no flaws are detected in the given URL, then the notification is sent to the Server that the given URL is not malicious and user can access it. If any flaws are detected, then the malicious code is sent to the prevention phase. In Prevention Phase, the non-malicious code is sent to the Information Leakage Evaluator. The Information Leakage Evaluator calculates some value and sent it to the Decider. Decider has some threshold value based on the value the decider allow or deny the URL.

The malicious codes are passed into the XSS Filter, if it is persistent type of attack, analyze all the input tags and passed it to the Input Field Extractor. The Input Field extractor extracts all the input tags in the malicious code and passed it to the Input Filter. The Input Filter filters the scripts, expressions, tags and other

inputs based on the pattern stored in Database. Finally the result is sent to the Reporter. If it is non-persistent type of attack, the malicious code is passed into the Link Filter. The Link Filter analyzes and extracts the links and filters it using existing pattern stored in Database. Finally the result is sent to the Reporter. Reporter sends the result whether to allow or deny the website.

In this section, we introduce two efficient algorithms. One for detecting the Cross Site Scripting attack called concolic algorithm and another one is for preventing the detected Cross Site Scripting attacks called Pattern Filtering Approach. The proposed scheme has the following two phases, 1) Detection Phase and 2) Prevention Phase. In Detection Phase, we get the URL which the user need to access and is passed to the URL extractor. The URL extractor split the given URL into protocol, host, path, query and reference and the extracted components are stored in the Symbolic table. The host of the URL in the Symbolic table is compared with the Concrete table which contains list of malicious link using Comparator. If the given URL is matched with the existing malicious URLs, then it is stored in a trace file and marked as matched. If not matched, then it is marked as unmatched. Then the stored URL in the trace file is sent to the bit calculator, which calculate the number of bits in the URL and send it to the Flaw Detector.

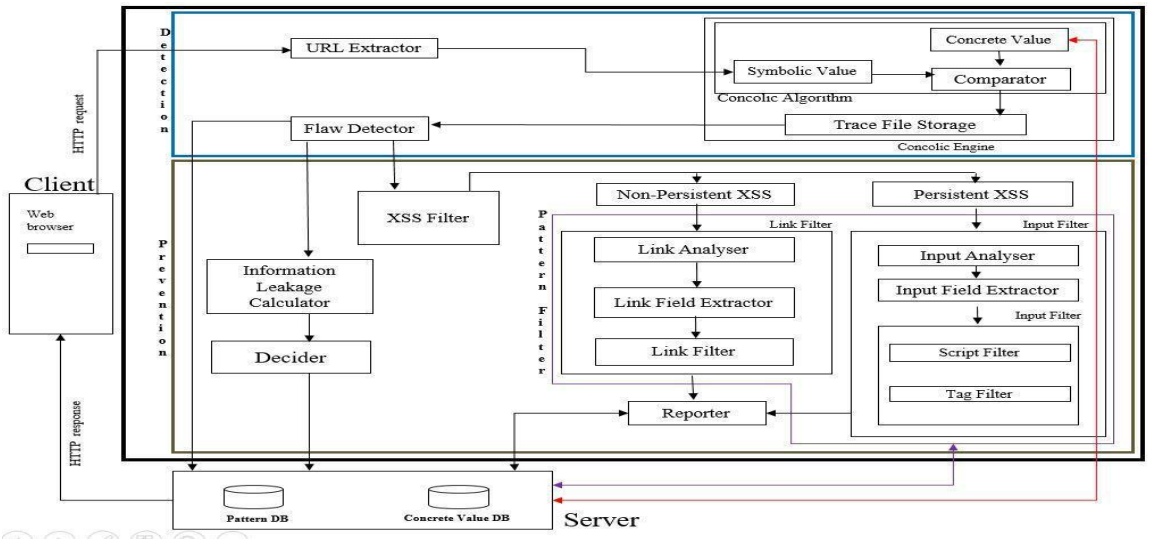

Fig. 1: Proposed System Architecture

If no flaws are detected in the given URL, then the notification is sent to the Server that the given URL is not malicious and user can access it. If any flaws are detected, then the malicious code is sent to the prevention phase. In Prevention Phase, the unmatched vulnerable URLs are sent to the Information Leakage Evaluator. The Information Leakage Evaluator calculates some value and sent it to the Decider. Decider has some threshold value based on the value the decider allow or deny the URL. The malicious code is passed into the XSS Filter, which identifies the type of XSS (Persistent/Non-Persistent) present in it. If it is persistent type of attack, analyze all the input tags and passed it to the Input Field Extractor. The Input Field extractor extracts all the input tags in the malicious code and passed it to the Input Filter. The Input Filter filters the scripts, expressions, tags and other inputs based on the pattern stored in Database. Finally the result is sent to the Reporter. If it is non-persistent type of attack, the malicious code is passed into the Link Filter. The Link Filter analyzes and extracts the links and filters it using existing pattern stored in Database. Finally the result is sent to the Reporter. Reporter sends the result whether to allow or deny the website.

2 .Algorithm Used:

2.1 Concolic Algorithm:

The name concolic is the combination of two words concrete and symbolic. Here in our algorithm we use concrete table to store existing malicious URL and symbolic table to store the given URL. Both the tables are compared and the status is stored in Trace File Storage for the detection of XSS.

BEGIN
GET the URL from the user
ANALYSE and PARSE the URL
SEPARATE the URL as protocol, authority, path, query and reference
STORE the separated URL in the SYMBOLIC table
STORE some of the malicious URL in CONCRETE table

COMPARE the SYMBOLIC table with CONCRETE table
IF (SYMBOLIC == CONCRETE)
Store the URL in the TRACE FILE
ELSE
FORWARD to Information Leakage Evaluator
ENDIF
READ the TRACE file.
CALCULATE the number of bits in the URL stored in trace file
DETECT the level of vulnerability using FLAW DETECTOR.
END

2.2 Pattern Filtering Approach:

BEGIN
GET the input from the DETECTION phase.
IDENTIFY the type of XSS attack in the input using the DATABASE.
APPLY different pattern filtering approach for the input.
IF PERSISTENT
INITIALIZE a filter configuration.
DO filter for (servlet request, response)
GET the string values as a parameter for the variables
MAP these variables with the STORED variables for PREVENTION
DEFINE a XSS script avoiding method
DO
AVOID NULL character
AVOID anything between Script
AVOID anything in a <src> which is unrelated
REMOVE any lone <script> and </script> tag
AVOID expression (eval, javascript, VBscript, Onload)
ENDDO
RETURN the values and SEND it to the reporter
ENDIF
IF NON- PERSISTENT
INITIALIZE a filter configuration
DO filter for (servlet request, response)
INITIALIZE the variables for storing the labels and contents
IMPLEMENT a filtering process on the link field.
AVOID link reference based on pattern filter using pattern DB
MATCH the sting value with the stored variables in DB
IF matches
AVOID the link.
ENDIF
ENDIF
IF ATTACK is present
REPORT to the CLIENT
ELSEIF
RESPONSE the CLIENT with the requested service
ENDIF
END

3. Components of Proposed System:

Our proposed system consists of various numbers of components in order to perform detection and prevention effectively. Components that are available in our proposed system are: *Url Extractor, Concolic Engine, Flaw Detector, Pattern Filter, and Information Leakage Calculator.*

3.1 URL Extractor and Concolic Engine:

The user input URL is given to the URL Extractor as an input. The URL Extractor separates the URL into host, path, query and reference. The separated URL components are stored in a Symbolic table. In Concolic Engine, the Symbolic table which contains the given URL is compared with the Concrete table which has list of existing malicious URL i.e., the host of the URL in the Symbolic table is compared with the Concrete table which contains list of malicious link using Comparator. If the given URL is matched with the existing malicious

URLs, then it is stored in a trace file and marked as matched. If not matched, then it is marked as unmatched. Then the stored URL in the trace file is sent to the bit calculator, which calculate the number of bits in the URL and send it to the Flaw Detector.

3.2 Flaw Detector:

Based on the status (Matched/Unmatched) stored in the Trace file and the bit value, the Flaw Detector detects the flaws in the URL. If the calculated bit value of URL is greater than threshold value (assume 50 bits), it is treated as malicious. If no flaws are detected in the given URL, then the notification is sent to the Server that the given URL is not malicious and user can access it. If any flaws are detected, then the malicious code is sent to the prevention phase.

3.3 Pattern Filter:

If it is persistent type of attack, analyze all the input tags and passed it to the Input Field Extractor. The Input Field extractor extracts all the input tags in the malicious code and passed it to the Input Filter. The Input Filter filters the scripts, expressions, tags and other inputs based on the pattern stored in Database. Finally the result is sent to the Reporter. If it is non-persistent type of attack, the malicious code is passed into the Link Filter. The Link Filter analyzes and extracts the links and filters it using existing pattern stored in Database. Finally the result is sent to the Reporter. Reporter sends the result whether to allow or deny the website.

3.4 Information Leakage Calculator:

If malicious code not belongs to the types of XSS, then it is sent to the Information Leakage Evaluator. The Information Leakage Evaluator calculate some value using the formula given below

$$ILC = (MV - UMV)^2 / (2*LOS) \quad (1)$$

Where,

ILC = Information Leakage Calculator, MV = Matched values

UMV = Unmatched values, LOS = Length of the strings

This equation number (1) returns an anomaly score which is calculated by squaring the difference between the matched and unmatched values divided by twice the length of the string. This anomaly score is compared with the predefined threshold value in order to check the vulnerability level.

Performance Comparison:

The various URLs and scripts found in the given user URL are extracted and their level of vulnerability is checked. The system detects all vulnerable contents present in a web page by downloading the whole web page. The downloaded content is stored in a database and is compared with another database which has vulnerable contents. The time taken to inspect the elements of the web page is average as the whole content is to be downloaded. Then the links are stored in a symbolic table which means that time taken to process the input URL increases when the content present in the given URL is larger.

Table 1: Performance Analysis

Works	Detection of XSS Attacks		Detection Technique	Approach	Prevention	Prevention Technique	Types of Attack Prevention
	Non-persistent	persistent					
DAVDS	No	Yes	Taint Analysis	Static	No	-	-
PPXSS	-	-	-	Dynamic	Yes	Pattern Filtering	Persistent
DXSSCT	Yes	No	Concolic	Dynamic	No	-	-
SMXSS	Yes	No	Zen Framework	Static	No	-	-
APXSSE	No	Yes	No	Server	Yes	Encoding	Persistent
XSS: DAP	Yes	Yes	Concolic Approach	Static / Dynamic	Yes	Pattern Filtering	Both

Abbreviations:

DAVDS: Design of Automatic Vulnerability Detection System for Web Application.

PPXSS: Preventing Persistent Cross Site Scripting Attack by Pattern Filtering Approach

DXSSCT : Detecting Cross Site Scripting Vulnerability using Concolic Testing

SMXSS : Developing a Security Model to Protect Websites from Cross Site Scripting Attacks Using Zen Framework Application.

APXSSE: Analysis and Prevention for Cross-site Scripting Attack Based on Encoding.

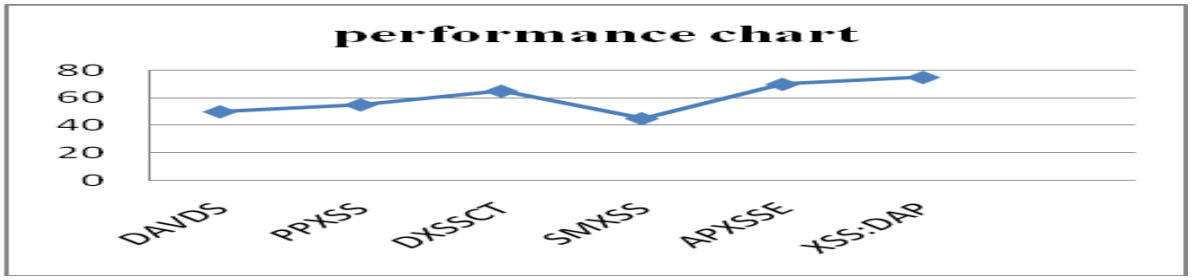

Fig. 2: Performance Chart

The above performance chart is based on the performance calculated for detection and prevention in both types of XSS Attack (Persistent/Non-Persistent). Here DAVDS has 50% of performance, since it performs only detection for persistent type of XSS Attack and did not perform any prevention. As PPXSS perform only prevention of persistent type of attack, it has 55% of performance. DXSSCT has 65% of performance because it performs only detection of non-persistent type of attack. Since SMXSS detect only persistent type of attack, it has 45% of performance. APXSSE has 70% of performance, since it performs detection and prevention of persistent type of attack. In our paper (XSS: DAP), we perform detection and prevention of XSS Attack for both types. So the performance of our paper is 75%.

Conclusion And Future Work:

In this paper, we have proposed a scheme for Detection and Prevention of Cross-Site Scripting (XSS) Attack Using Concolic Testing and Pattern Filtering Approach in Web Applications. This scheme is evaluated by using sample of well-known attack patterns. Initial stages of evaluation shows that the proposed scheme is produce not false positive and false negative. The pattern filtering process takes minimum of O (n) time. In this work a secure tool was developed which written in java language which is called XSS Detection and prevention; this tool works in forum, takes input form field as target to detect XSS attacks by inject malicious code and prevent it using pattern filtering approach. Also we have implemented a solution to notify other attacks using information leakage value calculator based on the threshold value.

Our future work is to estimate the attack of all individual processes to set the Threshold Value dynamically. This System can be improved by making use of the user input with more data which are to be newly produced with the cheat sheet. One shortcoming of the current system is that sometimes allows the injected query in which are not present in the database. This can be improved by including the updated cheat sheet data in the validation part with dynamically in future.

REFERENCES

Abdul Bashah Mat Ali, 2011. SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks. Elsevier World Conference on Information Technology, vol 3, pp: 453-458.

Binbin Qu,Beihai Liang,Sheng Jiang, 2013. Design Of Automatic Vulnerability Detection System For Web Application Program. IEEE 4 th International Conference on Software Engineering and Service Science (ICSESS), pp: 89-92.

Bisht, P., P. Madhusudan and V.N. Venkatakrishnan, 2010. CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks. *ACM* Transaction on Information System and Security, pp: 1-38.

Amutha Prabakar, Dr.M., M. Karthikayenand Prof.K. Marimuthu, 2013. AnEfficient Technique For Preventing SQL Injection Attack Using PatternMatching Algorithm. IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN), pp: 503-506.

Mahapatra, Dr R.P., Ruchika Saini and Neha Saini, 2012. A Pattern Based Approach to Secure Web Applications from XSS Attacks. International Journal of Computer Technology and Electronics Engineering (IJCTEE), vol 2, pp: 196-203.

Imran Yusof, Al-Sakib Khan Pathan, 2014. Preventing Cross Site Scripting Attack By Applying Pattern Filtering Approach. IEEE 5 th International Conference on Information and Communication Technology for The Muslim World (ICT4M), pp: 1-6.

Joaquin Garcia-Alfaro, Guillermo, 2009. "A Survey on Cross-Site Scripting Attacks", https://arxiv.org/abs/0905.4850

Junjin, M., 2009. An Approach for SQL Injection Vulnerability Detection. IEEE 6 th International Conference on Information Technology: New Generations, ITNG '09, pp: 1411-1414.

Lwin Khin Shar and Hee Beng Kuan Tan, 2012. Automated removal of cross site scripting vulnerabilities in web applications. Elsevier Information and Software Technology, vol 54, pp: 467-478.

Michelle Ruse,Samik Basu, 2013. Detecting Cross-Site Scripting Vulnerability Using Concolic Testing. IEEE 10th International Conference on Information Technology: New Generation, pp: 633-638.

Mukesh Kumar Gupta, M.C. Govil, Girdhari Singh, 2014. Static Analysis Approaches To Detect SQL Injection And Cross-Site Scripting Vulnerabilities In Web Applications: A Survey. IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), pp: 1-5.

Munqath, H. Alattar and S.P. Medhane, 2013. Efficient Solution for SQL Injection Attack Detection and Prevention. International Journal of Soft Computing and Engineering (IJSCE) , vol 3(1), pp: 395-398.

Priyanka, and Vijay Kumar Bohat, 2013. Detection of SQL Injection Attack and Various Prevention Strategies. International Journal of Engineering and Advanced Technology (IJEAT), vol 2, pp: 458-465.

Puspendra Kumar, Scholar, and R.K. Pateriya, 2012. A Survey on SQL Injection Attacks Detection and Prevention Techniques. IEEE Third International Conference on Computing Communication & Networking Technologies (ICCCNT), pp: 1-5.

Rahul Johari, and Pankaj Sharma, 2012. A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection. IEEE International Conference on Communication Systems and Network Technologies (CSNT), pp: 453-458.

Rattipong Putthacharoen, and Pratheep Bunyatnoparat, 2011. Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique. IEEE 13th International Conference on Advanced Communication Technology (ICACT), pp: 1090-1094.

William G.J. Halfond, Alessandro Orso and Panagiotis Manolios, 2008. WASP Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation. IEEE Transactions on Software Engineering, vol 34(1), pp: 65-81.

Yousra Faisal Gad Mahgoup Elhakeem, Bazara Barry, 2013. Developing a Security Model to Protect Websites from Cross-site Scripting Attacks Using Zend Framework Application. IEEE International Conference on computing, electrical and engineering (ICCEEE), pp: 624-629.