International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 11 (2018) pp. 9835-9842

Detection and Removal of XSS Vulnerabilities with the Help of Genetic Algorithm

Jyotiraditya Tripathi
M.Tech. Student
Department of Computer Science & Technology
Central University of Punjab, Bathinda, Punjab, India.

Bhawana Gautam
M.Tech. Student
Department of Computer Science & Technology
Central University of Punjab, Bathinda, Punjab, India.

Dr. Satwinder Singh
Assistant Professor
Department of Computer Science & Technology
Central University of Punjab, Bathinda, Punjab, India.

Abstract:

This research paper explains the type of cross-site scripting attacks and their impacts on web application security. This paper proposes a approach that is based on genetic algorithm that gives ability to web application developers that they can detect XSS vulnerabilities from source code by covering minimal test cases. It also proposes a hybrid approach for the removal of vulnerabilities that are found in detection phase. In starting it was not taken as very serious threat to web application because according to security researchers at that time, "it is an attack that cannot root operating system, it can perform privilege escalation then why should we care." But in 2005, world got the answer when XSS attack downs the server of MYSPACE site (a Russian Social networking site).

Keywords: XSS attack, input validation, sanitization, Genetic algorithm, web security, web application, php

INTRODUCTION

Cross-site scripting vulnerabilities are present since the year 1996 during the early days of the World Wide Web (WWW). This was the time when e-commerce was about to take off. In that era, very large numbers of Web pages were under construction and developers have to put more and more effort to make pages little interactive in those days. Suddenly, a light weighted programming language called as JavaScript hit the scene and this acted as an unknown harbinger for cross-site scripting attacks, this changed the whole landscape of web app security forever.

JavaScript gives ability to Web developers that they can create interactive Web page with effects that includes image rollovers, floating menus, and pop-up window. Hackers found a way that they could forcibly load any Web site (bank, auction, store and so on) into an HTML Frame within the same browser window when users visited their Web pages. After this by using JavaScript, they could gain the ability to cross the boundary between the two Web sites, and could read from one frame into the other. In this scenario, hackers were able to extract usernames and passwords typed into HTML Forms, steal cookies, or can compromise any confidential information on the screen. Initially media reported this problem as Web browser vulnerability. Netscape Communications, which was a dominant browser vendor at that time, it introduced and implemented the "same-origin policy," to prevent this vulnerability, a policy that restricts JavaScript on one Web site from accessing data from another. Hackers accepted this technique as a challenge and started discovering many ways to tackle the restriction. In the mid-January of 1999, the cross organization team chose "Cross Site Scripting" from humorous list of proposals:

- Unauthorized Site Scripting
- Unofficial Site Scripting
- Uniform Resource Locator (URL) Parameter Script Insertion
- Cross-site Scripting
- Synthesized Scripting and Fraudulent Scripting

TYPES OF XSS ATTACK

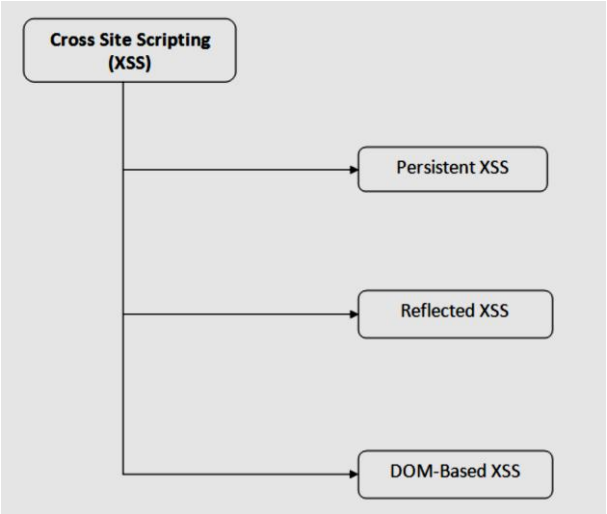


Figure 1: XSS Attacks

1. Persistent Attack

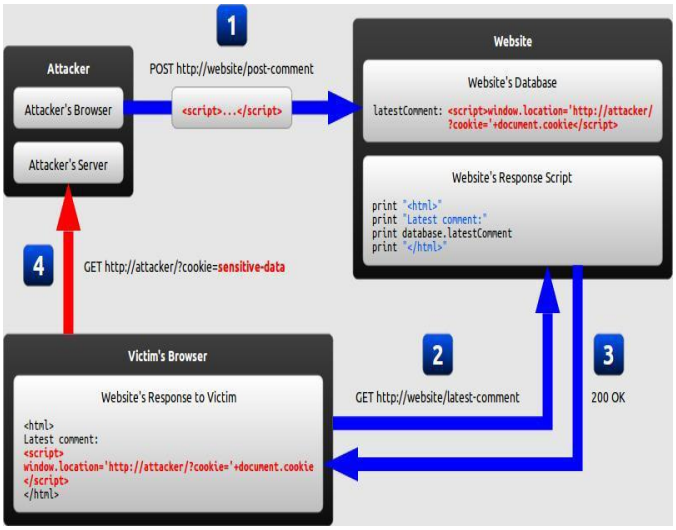
As is with most web-based attacks, exploiting Stored XSS vulnerabilities requires some research. Usually attackers search vulnerable websites that can be used to carry out an attack. There are some types of websites which are prone to such vulnerabilities because they allow content sharing between users, and consequently they constitute the starting points of research in this respect:

- Forums / message boards
- Blogging websites

- Social networks
- Web-based collaboration tools
- Web-based CRM/ERP systems

Once a website is identified as being potentially vulnerable, attackers will try to inject script code into data that is going to be stored on the server. Then, they will access the web pages that are serving back the content they have posted and test if the script executes. This is a type of attack where the malicious code originates from the website's database. This is most dangerous attack among all type of XSS attacks. In this attack, it saves code on server and permanently delivers attack. It can be found most commonly on forums and sites that allow user to POST HTML formatted data.

- Firstly, attacker chooses one of the website's forms and inserts malicious code into the website's database.
- The victim requests a page from the website.
- Now website also includes the malicious code from the database in the response and sends it to the victim.
- After receiving, victim's browser executes the malicious script inside the response, and sends the victim's cookies to the attacker's server.

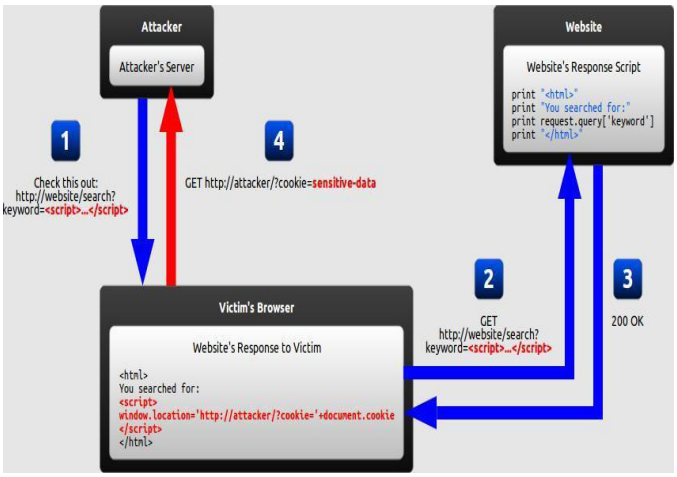


2. Reflected XSS

Reflected XSS attacks, also known as non-persistent attacks, occur when a malicious script is reflected off of a web application to the victim's browser. The script is activated through a link, which sends a request to a website with a vulnerability that enables execution of malicious scripts. The vulnerability is typically a result of incoming requests not being sufficiently sanitized, which allows for the manipulation of a web application's functions and the activation of malicious scripts. To distribute the malicious link, a perpetrator typically embeds it into an email or third party website (e.g., in a comment section or in social media). The link is embedded inside an anchor text that provokes the user to clicking on it, which initiates the XSS request to an exploited website, reflecting the attack back to the user.

In this type of XSS attack, the malicious string is part of the victim's request for website. It is used with URL that appears to be innocent but has XSS attack located within the link. The diagram below illustrates this scenario:

- The attacker crafts a URL containing a malicious string and sends it to the victim.
- In this step, target must be tricked by the attacker into requesting the URL from the website.
- The website includes the malicious string from the URL in the response.
- The victim's browser executes the malicious script inside the response and sends the victim's cookies to the server of attacker.

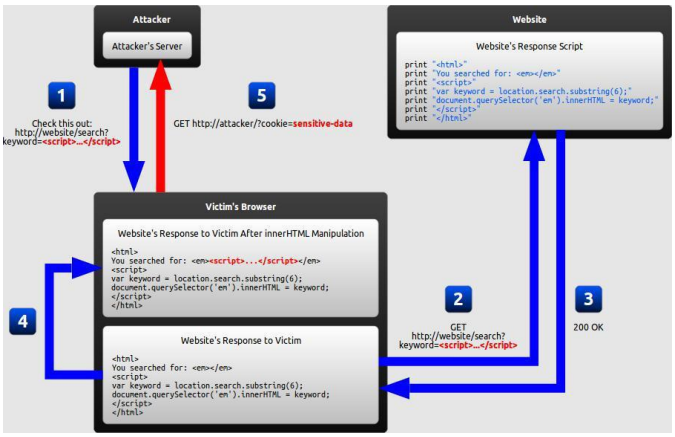


3. DOM-Based XSS

The Document Object Model is a convention for representing and working with objects in an HTML document (as well as in other document types). Basically all HTML documents have an associated DOM, consisting of objects representing the document properties from the point of view of the browser. Whenever a script is executed client-side, the browser provides the code with the DOM of the HTML page where the script runs, thus, offering access to various properties of the page and their values, populated by the browser from its perspective. DOM XSS is a type of cross site scripting attack which relies on inappropriate handling, in the HTML page, of the data from its associated DOM. Among the objects in the DOM, there are several which the attacker can manipulate in order to generate the XSS condition, and the most popular, from this perspective, are the document.url, document.location and document.refferer objects.

DOM-based XSS consists of both persistent and reflected XSS. In this attack, the malicious string is not actually parsed by the victim's browser until the website's legitimate JavaScript is executed. The diagram below illustrates this

scenario for a DOM XSS attack:



PROPOSED APPROACH

There are many approaches for detection of XSS attack vulnerabilities from web application but according to analysis of different type of detection techniques it can be concluded that every approach has some shortcomings and every detection technique requires so many test cases to detect such type of vulnerabilities from source code. In proposed approach, it tries to cover as many as possible security loop holes regarding this type of attack by testing minimum test cases to increase the efficiency of detection technique with the help of genetic algorithm.

This approach consists of following components mainly.

- First component involves converting the source codes of the applications to be tested to Control Flow Graphs (CFGs) using the White Box Testing techniques, where each node will represent a statement and each edge will represent the flow of data from node to node.
- In second phase, taint analysis will be done with the help of RIPS (a static analysis tool for PHP projects). It identifies all tainted variables from source code of selected real time web application.
- In third phase, a fitness function is designed that is required in GA for detection of XSS vulnerabilities from source code.

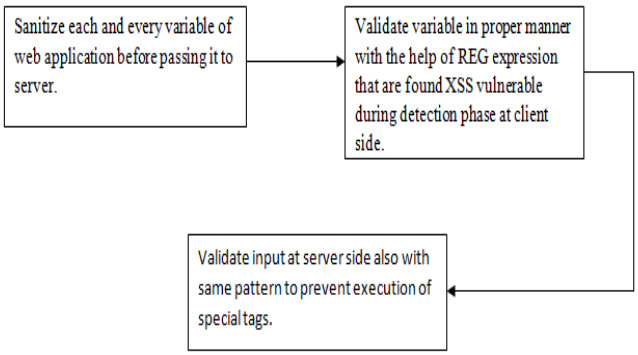
The fitness function should be able to identify as many as possible variables that are responsible for XSS attack by performing minimum iterations. After identifying the XSS vulnerabilities in terms of tainted input variables in web application, these variables must be validated and sanitized to secure the source code. To secure the application, means for removal of detected vulnerability from source code, these steps will be performed:

Removal Approach:

Step 1: Sanitizing User input and Every Variable: By sanitizing the data, any unlawful character from the data is removed. Because the data is sanitized doesn't guarantee that it is legitimately designed. So that validation is also performed which decides whether the data is in a legitimate frame?

Step 2: Validating Input at Client Side: Not trusting client input implies validating it for type, length, arrangement and range at whatever point variable goes through a put stock in limit, say from a Web page to an application script, and after that encoding it proceeding redisplay in a dynamic page. By and by, this implies developers have to survey each point on site where client provided or passing input is taken care of and handled and guarantee that, before being passed back to the client, any values acknowledged from the customer side are checked, filtered and encoded.

Step 3: Validating Input at Server Side: Client side validation can't be depended upon, yet user input can be constrained down to a negligible alphanumeric set with server-side processing before being utilized by Web application in any capacity. Regular expressions can be used in php script to search and replace user input to ensure that input passed from client side is non-malicious. This sanitization and validation ought to be performed on all user input before passing it on to another procedure. For instance, a telephone number field shouldn't acknowledge any accentuation other than enclosures and dashes. You additionally need to encode extraordinary characters like "<" and ">" before they are redisplayed in the event that they are gotten from client input. For instance, encoding the content tag guarantees a program will show <script> yet not execute it. In conjunction to encoding, it is vital that your WebPages dependably characterize their character set so the program won't decipher unique character encodings from other character sets.



[Working Scenario of removal approach]

Functions to Validate Data:

There are several predefined functions in PHP that can be used to validate the user data in efficient manner. Some of predefined function for validating data are listed and explained below with proper example.

PHP Built-in Sanitization and Escaping Functions: Some of predefined and built-in functions that are used for data sanitization are listed below.

Function	What it Does	Example
is_numeric()	Tests if data matches 0 to 9 with optional sign and optional decimal point.	is_numeric($input) will return true if $input == '-9.123'
preg_match()	Test if data matches regular expression.	preg_match('/^[a-z]{2,3}$/', $input) returns true if $input is lowercase letters either 2 or 3 characters long. Note the ^ and $ in the regex.
filter_var()	Test if data conforms to a built-in PHP filter.	filter_var($input, FILTER_VALIDATE_EMAIL) tests if $input is a valid email address. Other useful filters are FILTER_VALIDATE_IP, FILTER_VALIDATE_URL, FILTER_VALIDATE_BOOLEAN. You can find more filters here.
in_array()	Tests if data is one of a range of allowed values.	in_array($input, array('Windows', 'Linux', 'OSX', 'Other')) will return true if $input contains one of the allowed values. Great for <select> fields and radio buttons on web forms.

Function	Output	Description
intval('123AA456')	123	Sanitize integers. [docs]
filter_var('mark<script>@example.com', FILTER_SANITIZE_EMAIL)	markscript@example.com	Sanitize emails. [docs]
filter_var('Testing <tags> & chars.', FILTER_SANITIZE_SPECIAL_CHARS)	Testing <tags> & chars.	Encode special chars. [docs]
filter_var('Strip <tag> & encode.', FILTER_SANITIZE_STRING);	Strip & encode.	Remove tags. [docs]
filter_var('Strip <tag> & encode.', FILTER_SANITIZE_STRING, FILTER_FLAG_ENCODE_LOW | FILTER_FLAG_ENCODE_HIGH | FILTER_FLAG_ENCODE_AMP)	Strip & encode.	Remove tags with extra encoding flags. [docs]

TESTING OF REAL TIME WEB APPLICATION FOR DETECTION OF XSS VULNERABILITY IN SOURCE CODE

FEE Portal is online fee portal coded in PHP for online payment of student fee like bus fee, character certificate fee, migration certificate fee etc. of Central University of Punjab. By analyzing the source code of this project and its metrics, following results are obtained:

Project Metrics

Files	13
Program Units	7
Lines	986
Blank Lines	58
Code Lines	656
Comment Lines	9
Statement	310

Convocation-2017 is a real time web application for the online registration and payment of convocation fee of Central University of PUNJAB alumni. It is a PHP based web application. By analyzing the source code of this project and its metrics, following results are obtained:

Project Metrics

Files	21
Program Units	12
Lines	1586
Blank Lines	103
Code Lines	1056
Comment Lines	10
Statement	536

Student Information is another web based application coded in php. This application is responsible for displaying the particular student details regarding academic and extracurricular activities. The project metrics of this project is as follows:

Files	18
Program Units	9
Lines	1320
Blank Lines	90
Code Lines	984
Comment Lines	8
Statement	498

RESULTS AND DISCUSSION

In this section, experiments are performed on selected real time projects and after applying proposed technique for detection of XSS vulnerabilities that are present in source code, a hybrid technique is applied on the code to remove the vulnerabilities that are found by using the GA, and a simple test is performed with the help of VEGA tool to check the efficiency of hybrid approach for removal of XSS vulnerabilities.

To achieve the objectives of research, first of all a static analysis is done on selected real time projects with the help of tool, RIPS. After performing the static analysis a CFG is generated where each node represents a statement and each edge represents the flow of data from node to node. After this taint analysis is done. In case of XSS vulnerability, tainted variable refers to inputs from user or database, and print statements that append a string into a web page. However it is impossible to cover the all paths of source code in testing. Hence here we selected a subset of paths that are of our interest. It means only focus on vulnerable paths regarding XSS vulnerabilities. If we talk about XSS vulnerabilities then these are the paths where input is executed without validation.

In generated CFG, total vulnerable paths = 4

Results regarding paths covered by variables in project 'FEE-Portal' that are vulnerable.

User Input	**Taints**	**No. of Vul. Paths Covered**
$_POST	22 ,42 ,54	3
$_POST[*]	23 ,44 ,43	3
$_POST[Amount]	59	1
$_POST[DateCreated]	60	1
$_POST[Fee]	8 ,15	2
$_POST[PaymentID]	57	1
$_POST[ResponseCode]	51 ,55	2
$_POST[SecureHash]	51	1
$_POST[TransactionID]	58	1
$_POST[amt]	14 ,16	2
$_POST[dob]	8 ,22	2
$_POST[fee_for]	14 ,15	2
$_POST[pass]	10	1
$_POST[reg]	8 ,20	2
$_POST[secretkey]	12 ,13 ,19	3
$_POST[submitted]	20	1
$_POST[user]	8	1
$_POST[vercode]	16 ,13	2

Genetic Algorithm and Fitness Function Design:

A pseudo code of GA is as follows:

```
population = generate_random_population();
for(T in vulnerable paths) {
while(T not covered AND attempt < max_try) {
selection = select(population);
offspring = crossover(selection);
population = mutate(offspring);
attempt = _ttempt + 1;
}}
```

Fitness Function:

F(x) = ((Cpath% + D) * XSSp%)/100+ Nd

Where Cpath%= Percentage of Vulnerable path covered by variable

D = Difference between Path covered by variable and total vulnerable path

XSSp= Total vulnerable paths

Nd=Number of database queries that uses particular variable

Steps Performed in GA to find XSS vulnerability:

a) Representation:

The most common form of representing or encoding chromosomes in GA is using binary format. However, using binary format in XSS vulnerabilities detection would be very complex since the chromosomes represent patterns of real strings that serve as inputs while testing. Therefore, we decided to use natural numbers as the encoding scheme in our GA.

Representation of variables as chromosomes:

Variable/Parameter	Representation of Chromosomes
$_POST	C1
$_POST[*]	C2
$_POST[Amount]	C3
$_POST[DateCreated]	C4
$_POST[Fee]	C5
$_POST[PaymentID]	C6
$_POST[ResponseCode]	C7
$_POST[SecureHash]	C8
$_POST[TransactionID]	C9
$_POST[amt]	C10
$_POST[dob]	C11
$_POST[fee_for]	C12
$_POST[pass]	C13
$_POST[reg]	C14
$_POST[secretkey]	C15
$_POST[submitted]	C16
$_POST[user]	C17
$_POST[vercode]	C18

b) Initial Population:

The GA population is considered as the set of possible solutions for the problem which is to be solved. These possible solutions are generally known to as chromosomes. In this work, the initial population is a set of test data that is generated and considered according to the path coverage criterion. After the initial population is selected, each individual chromosome is calculated and evaluated for possible inclusion in the next generation and it is based upon the fitness function.

Initial population for selected project Fee-portal is equal to the number of variables used in project and the number of variables is 18.

Hence, number of chromosomes initially=18

c) Fitness Function:

The fitness function is basically a measurement of how good a particular chromosome is in solving the given problem under consideration. So, a chromosome has a higher fitness value if it is closer to solving the problem. In our research, fitness function evaluates the percentage of vulnerable path covered by any variable in CFG. For every variable, fitness function is calculated.

d) Selection:

In this phase, chromosomes are selected based on their fitness value for next generation iteration. In our research, whose fitness values are above 50%, are only selected for next generation and according to this criteria, the selected chromosomes are:

Selected Chromosomes for next iteration	Fitness Value in percentage
C1	76%
C2	76%
C3	77%
C5	53%
C7	52%
C10	52%
C11	53%
C12	54%
C14	52%
C15	76%
C18	52%

e) **Crossover:**

In this phase, two chromosomes are combined to form new one in the hope that newer one will be better than their parents in terms of fitness value. Here those chromosomes are selected for crossover whose fitness values are highest and lowest with the respect of threshold. It means for crossover, C1 and C5 are selected with fitness value of 76% and 52%. Similarly other chromosomes are selected for next iteration. For next iteration fitness value will be increased by 10% means now new fitness value is 60%.

f) **Mutation:**

Here mutation operation is performed after crossover that alters the value of chromosomes. For example, C1 and C5 are mutated and new chromosome is formed whose fitness value is 64%.

After crossover and mutation, if some chromosomes are left whose fitness values are same then we have to check that which chromosome is also interacted with database query and what is depth of that particular chromosome. And on the basis of database depth, chromosomes are selected if fitness values are same.

After several iterations, we have detected that there are some chromosomes that are fittest in solving the problems and these are:

(C1,C2,C7,C10,C11,C12,C14,C15,C18)

It means that these variables have sensitive sinks and they are more dangerous in terms of XSS vulnerability exploitation. To secure the source code, the hybrid approach is applied for the removal of XSS vulnerabilities. Applying the hybrid removal approach means sanitizing the every variable and validating the inputs properly with the help regular expression at client and server both side with same pattern that are found XSS attack vulnerable during the detection phase by applying the genetic algorithm.

Regarding the most frequent fault types if we talk about the XSS vulnerabilities that are present in source code, the proposed approach is analyzed that it is able to detect such type of faults or not. The results are discussed below in tabular form with the description of fault types.

Fault Type Description	Detection by Proposed Approach
Missing "If (*cond*) { statement(s) }"	Yes
Missing function call	Yes
Missing variable initialization	Yes
Wrong logical expression used as branch condition	No
Variable Validation	Yes

A detailed study of XSS attacks has been carried out. On the basis of study and analysis of different detection techniques for finding the XSS vulnerabilities from source code, a comparative study has been carried out between different type of detection techniques and proposed detection technique, the comparison is shown below:

Detection Techniques	Persistent XSS Attack	Non-persistent XSS Attack	DOM based XSS Attack	Number of Test cases covered
Static Approach	X	✓	✓	High
Dynamic Approach	X	✓	✓	Medium
Hybrid Approach	✓	✓	✓	High
Proposed Approach	✓	✓	✓	Low

[Comparison between detection-focused techniques and proposed approach with respect to attack types]

Scanning the Project after applying Hybrid Approach for removal of XSS vulnerability by using VEGA tool:

XSS attack Flaws

Classification	No
Resource	Not found
Parameter	Not found
Method	POST
Detection type	Not found
Risk	No risk against this type attack

CONCLUSION AND FUTURE WORK

In this work, different approaches for the detection of XSS vulnerabilities in any web application have been studied and on the basis of analysis it can be concluded that every detection approach has its own merits and demerits and in almost every approach we have to take several test cases to detect XSS vulnerabilities. The main aim of this research is to detect as many XSS vulnerabilities that are present in source code of web application by covering minimal test cases to reduce the complexity of detection approach and increasing the efficiency of detection with the help of GA. In this research, to find the XSS vulnerability in source code, first of all a static analysis of code is done by using a PHP static analyzer, RIPS. After this a CFG is generated of whole source code and taint analysis is done to identify all tainted values in source code. By collecting this information, GA is applied on tainted variables by considering these variables as a initial population to detect the XSS vulnerabilities in source code. After detection of vulnerabilities a hybrid approach is applied on the target source code for the removal of vulnerabilities that are found in detection phase. This removal approach combines the all major three remedies that are being used by web developers to secure their application against such type of attacks individually at some extent. But if these remedies are applied on code individually then application security can be compromised very easily so that to improve the layer of security of application all three remedies are combined and applied on code collectively. To test the efficiency and accuracy of proposed hybrid approach for removal of such type of vulnerabilities, a web security scanner tool (VEGA) is used to identify the security level of selected project regarding XSS vulnerabilities. In future the detection approach can be applied on some other programming language like Python because now days Python is a language which has been involved in big boy's squad of web application development, it deals with the highly complex web sites like Instagram, Quora, Dropbox etc. There is a strong future scope of automating the process of removal of XSS vulnerabilities at the time coding by developing some application that comprises the all three remedies.

REFERENCES

[1] Hydara I, Sultan ABM, Zulzalil H, Admodisastro N. Current state of research on cross-site scripting (XSS) - A systematic literature review. Information and Software Technology. Elsevier B. V; 2015; 58:170–86. Available from: http://dx.doi.org/10.1016/j.infsof.2014.07.010

[2] N. Jovanovic, C. Kruegel, E. Kirda Pixy: a static analysis tool for detecting Web application vulnerabilities. Security and Privacy, 2006 DEC; 9055132 IEEE Symposium

[3] E. Galan, A. Alcaide, A. Orfila, J. Blasco A Multi–agent Scanner to Detect Stored–XSS Vulnerabilities, July 2010 ; Available from: https://e-archivo.uc3m.es/bitstream/handle/10016/9997/paper%20en%20proceedings%20333-338.pdf?sequence=1

[4] Xiaobing Guo, Shuyuan Jin, and Yaxing Zhang, XSS Vulnerability Detection Using Optimized Attack Vector Repertory, NOV,2015 ; Available from: http://or.nsfc.gov.cn/bitstream/00001903-5/423073/1/1000014290853.pdf

[5] Shashank Gupta and Lalitsen Sharma, Exploitation of Cross-Site Scripting (XSS) Vulnerability on Real World Web Applications and its Defense Dec 2012; in International Journal of Computer Applications 0975 – 8887. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.303.4511&rep=rep1&type=pdf

[6] Shashank Gupta and B. B. Gupta, Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art JAN 2017 ,in Springer. Available from: https://www.researchgate.net/profile/B_B_Gupta/publication/281823720_Cross-Site_Scripting_XSS_attacks_and_defense_mechanisms_classification_and_state-of-the-art/links/5604288608ae5e8e3f2fd025.pdf

[7] Sharma P, Johari R, Sarma SS. Integrated approach to prevent SQL injection attack and reflected cross site scripting attack. International Journal of System Assurance Engineering and Management. 2012 Sep; 26;3(4):343–51. Available from: http://link.springer.com/10.1007/s13198-012-0125-6

[8] Sun Y, He D. Model checking for the defense against cross-site scripting attacks. International Conference on Computer Science and Service System IEEE; 2012. p. 2161 4. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6394855

[9] V. Nithya , S. Lakshmana Pandian and C. Malarvizh, A Survey on Detection and Prevention of Cross-Site Scripting Attack. International Journal of Security and Its Applications 2015. Available from: https://www.researchgate.net/profile/S_Pandian/publication/281601290_A_Survey_on_Detection_and_Prevention_of_Cross-Site_Scripting_Attack/links/5943cbc745851525f890a09d/A-Survey-on-Detection-and-Prevention-of-Cross-Site-Scripting-Attack.pdfhttps://www.researchgate.net/profile/S_Pandian/publication/281601290_A_Survey_on_Detection_and_Prevention_of_Cross-Site_Scripting_Attack/links/5943cbc745851525f890a09d/A-Survey-on-Detection-and-Prevention-of-Cross-Site-Scripting-Attack.pdf

[10] Rattipong Putthacharoen and Pratheep Bunyatnoparat, "Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique",2011.Available from: http://www.icact.org/upload/2011/0579/20110579_finalpaper.pdf

[11] Jose Fonseca, Marco Vieira and Henrique Madeira "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks",SEP,2007.Available from: http://bdigital.ipg.pt/dspace/bitstream/10314/3533/1/Fonseca-CompSQLXSS.pdf