International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 20 (2017) pp. 10176-10183
© Research India Publications. http://www.ripublication.com

Eradicating Cross Site Scripting Attack for a Secure Web Access

K. Vijayalakshmi
Research Scholar, Department of Computer Science and Engineering,
B.S. Abdur Rahman University, Seethakathi Estate, Vandalur,
Chennai, Tamil Nadu, India.
Orcid Id : 0000000187897102

Dr. A. Anny Leema
Associate Professor, Department of Computer Applications,
B.S. Abdur Rahman University, Seethakathi Estate, Vandalur,
Chennai, Tamil Nadu, India.

Abstract

Recent updates of Vulnerability reports of the Open Web Application Security Project confirm that Cross Site Scripting (XSS) is one of the most common and severe web security defects. Cross-Site Scripting occurs when an application takes data from the user and sends it back to a web browser without validation or encoding. It occurs when the web application references the user input in HTML pages when there is no proper validation. An attacker can easily inject the malicious scripts through such inputs in the HTML pages of the application. When a client browses a tapped page, the client's browser which is unaware of the presence of malicious scripts may execute all scripts sent by the application which results in a successful XSS attack. To overcome this attack, this paper presents an Anti XSS Mechanism for mitigating XSS attacks and its vulnerabilities in Web Applications. Our proposed approach identifies the attack and detects it using a data refiner algorithm and secures them with appropriate encoding technique which prevents input values from causing any improper validation and execution of malicious script. We developed an Anti XSS tool, which contains two main mechanism called XSS Gauge and XSS Eradicator, to implement the proposed approach. Using this tool, we tested our proposed mechanism with the standard test bed applications and our work has shown a significant improvement, i.e., the average accuracy rate is 98.4 % which is far higher comparing to the existing systems in detecting and defending XSS Attacks.

Keywords: XSS Eradicator, Data Refiner, Context Analyzer, XSS Gauge, XSS Validator, Anti XSS Mechanism.

INTRODUCTION

The Web Applications are the boon for the internet users and at the same time the attackers are targeting the online applications and web services. Detecting these Web application attacks are very challenging because the attack traffic mimics the legitimate behaviour. A more sophisticated mechanism [1] is required to detect and mitigate such attacks.

The protection mechanism for XSS Attack is a very important need for the present scenario, in order to protect our sensitive information. XSS Attacks are considered extremely dangerous because it leads to identity theft, impersonation etc… Over 70% of organizations reported of having been compromised by a successful cyber attack[2]. During June and July 2006, the e-payment web application PayPal had been exploited by the attackers to steal sensitive data like credit card numbers from its members during more than two years until Paypal's developers fixed this XSS vulnerability [3][4].

Cross site scripting attack is a code injection attack performed to exploit the XSS vulnerabilities existing in the web application by injecting html tag / javascript functions into the web page so that it gets executed on the victim's browser when one visits the web page and successfully accesses to any sensitive victim's browser resource associated to the web application, example cookies, session IDs, etc... By exploiting Cross Site Scripting vulnerabilities in the scripts especially javascript, since it is highly used scripting language on the client side by web developers, the attacker targets the organizations that accommodate large online communities of users who use social networking sites, blogs and online news sites or the organizations that rely on web technology to generate revenue like the providers of online services, services that store personal or financial information such as online payment, banking services, etc...

A typical scenario of XSS is as follows : A URL enclosing a malevolent code is crafted and send by the attacker to the victim. The victim is fooled into ask for the URL from the trusted website. In the response, the trusted website contains the malevolent code from the URL. The malevolent code inside the response get executed in the victim's browser, thereby transferring the victim's cookies to the server of the attacker.

In this manner, the attacker discovers an approach to inject a code into a web page of the server. Whenever a user visits the page, the code will be downloaded and executed by their browser as part of interpreting the page [5]. Rest of our paper is organized as follows: in the related work we present a

review of attacking techniques and different approaches for the mitigation are discussed. System Architecture section presents the proposed system architecture and the major components involved for Anti XSS mechanism. Proposed defense mechanism discusses the various components and their process. Experimental and Evaluation section analyses the results and calculates the attack detection speed and efficiency rate of defense mechanism. Conclusion section concludes the work with its future scope.

RELATED WORKS

The primary objective of XSS attacks are stealing the victim's sensitive information and invokes the malicious actions. A detailed analysis has been done on detection and prevention techniques proposed by various researchers to mitigate XSS attack. The major XSS vulnerabilities can be detected by performing static and dynamic analysis on web application. Many researchers are carrying their work in this domain. Few of them are stated here as follows:

A server side prevention technique against XSS attacks was introduced by M.T. Louw[6] This technique known as BEEP, a browser enforced embedded policies, modifies the browser so that it cannot execute the malicious script. Security policies regulates what the server sends to BEEP enabled browser.

A mechanism for detecting malicious java script was proposed by O.Hallaraker and G.Vigna[7]. This system contains a browser embedded script auditing component and IDS to process the audit logs and evaluate them to the signature of already known malicious activities or attacks.

Web browser with a Sandbox environment was developed by Shasank Gupta and Lalitsen Sharma[8] a technique to mitigate XSS vulnerability. A web browser under the protection of a sandbox submits the user id and password to a web server. Web server generates the cookies and the sandbox protected web browser will receive these cookies.. Thus the cookie value will not be disclose into the windows and it cannot be grabbed by any attacker. The sandbox allows the execution of malicious script on the client's web browser but it cannot give the authority to disclose the cookie out of this protected environment and hence bye pass the XSS vulnerabilities.

A.Kieyzun, P.J. Guo,K. Jayaraman, and M.D. Ernst[9] devised an automatic technique for creating inputs that expose SQLI and XSS vulnerabilities. This technique generates sample inputs, symbolically tracks tainted data through execution including through database accesses and mutates the inputs to produce concrete exploits. The technique creates real attack vectors, has few false positives, incurs no runtime overhead for the deployed application, works without requiring modification of application code, and handles dynamic programming-language constructs. The author implemented the technique in php, a tool Ardilla. This approach was implemented in a tool called BLUEPRINT that was integrated with several popular web applications.

A security model called Browser Dependent XSS Sanitizer (BDS) by Shashank Gupta and B.B. Gupta[10] proposed a defense mechanism on the client-side Web browser for mitigating the effect of XSS vulnerability. The authors used a three step approach to eliminate the XSS attack without degrading much of the web browsing experience on various modern browsers.

The approach developed by P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna[11], goals to pinpoint information outflow by infecting the incoming data of the browser. The protection offered in the paper halts XSSvulnerabilities by tracing the movement of benign data inside the web browser. If it is to be transmitted to a third party, the user can authenticate the allowance of data transmission. Thus the user not only depends on security of web application but also has an extra protection layer while accessing the web.

The major influence of client side solution by K. Selvamani, A.Duraisamy, A.Kannan[12], effectually decreases XSS attacks. This paper presents an XSS defense without depending on web application providers by supporting an extenuation mode that greatly diminishes the amount of associated alert prompts and also it provides protection against stealing sensitive information such as session IDs and cookies. This paper used a procedure to decide if a resource requested is through a native link by examining the Referrer HTTP header and matching its domain with the requested webpage domain. The value of the domain is demarcated by way of splitting and analysing URLs. With the idea got from various research work, we conclude, this as a noble approach but it drops the concert as it follows numerous phases to resolve whether the website is defenseless or not.

A supervised learning based approach for automatically checking the XSS vulnerability of web pages is proposed by A. Nunan, E. Souto, E. M. dos Santos, and E. Feitosa [13]. The features of most frequent XSS attacks are collected from the web page contents and URL contents. Two classifiers Naïve Bayes classifier and Support Vector Machine (SVM) are used. For the classification two types of databases are used, one for positive samples and one for negative samples. Positive samples are collected from XSSed database. For negative samples two databases are used: Dmoz database and ClueWeb09 database. But in this paper a definite accurate solution has neither been attempted nor documented. Therefore in this proposed system we use a different strategy to stop malicious strings by identifying the vulnerable portion of inputs being used by conducting data refinement and context analysis.

SYSTEM ARCHITECTURE

Sensitive information and its database are the honey pot for attackers, therefore the proposed system takes the responsible for more intensive protection in order to protect the information with the two main mechanism called XSS Gauge and XSS Eradicator. The entire Anti XSS Mechanism handles the tainted sources efficiently with Data Refiner Algorithm and XSS Validator.

An attacker can easily inject the malicious scripts through the inputs in the HTML pages of an application[14]. When a client browses a tapped page, the client's browser which is unaware of the presence of malicious scripts may execute all scripts sent by the application which results in a successful XSS attack[15]. Generally all web applications are vulnerable to a variety of malicious attacks. One of the topmost attack is a cross-site scripting vulnerabilities. XSS proliferates as a result of executing non legitimate scripts within an origin[16]. There are three main varieties: Stored, Reflected and DOM-Based attacks. This attacks happens both in the server as well as in the client side. To guard against this attack the preliminary protection methods are :

- Encoding the Meta Characters
- Input Validation
- Content Analyzer
- Code Replacer

This preliminary protection is taken care by the Data Refiner Mechanism, which refines the data from XSS threats and handles the tainted sources. The refined information is sent to the XSS Eradicator where the XSS Validator takes care of eradicating the attack by context analysis and finally with the code replacement where all comments or instance of malicious code is replaced by model which contains trusted code and not making any changes to trusted content. The proposed system architecture, Figure 1, depicts the Anti XSS tool for defending against this attack.

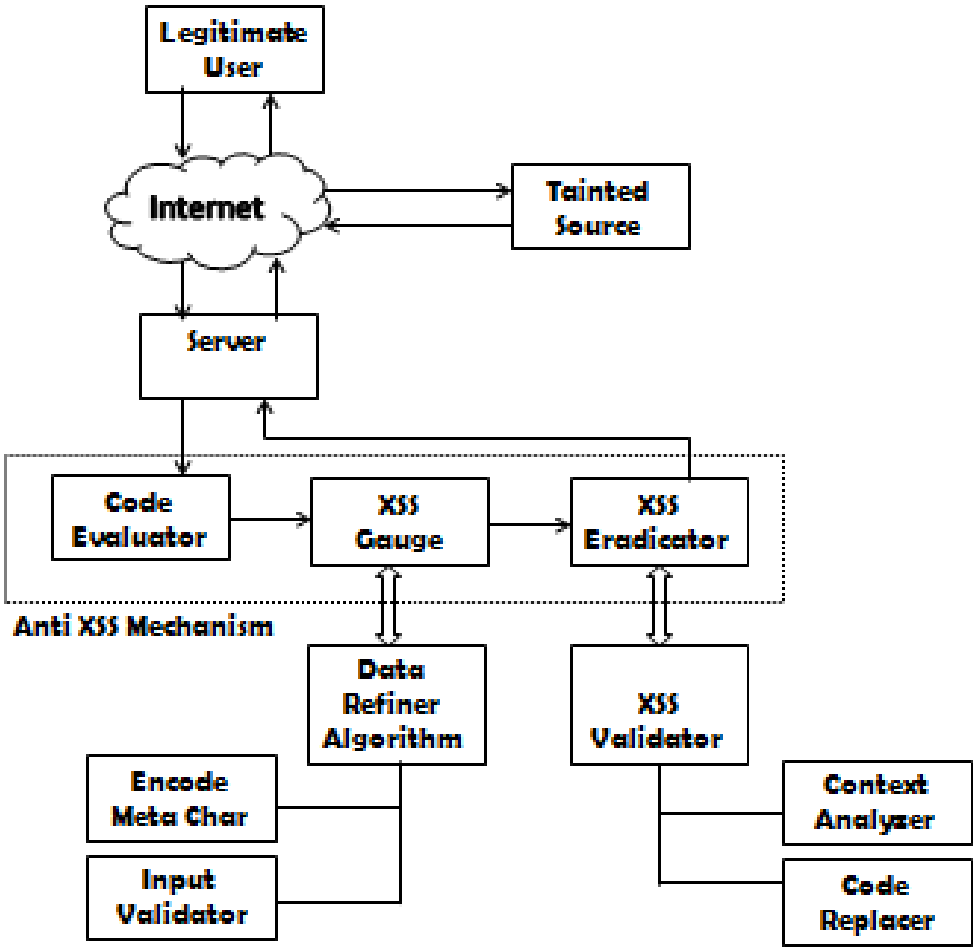

Figure 1: Architecture of Anti XSS Mechanism

The proposed Anti XSS system works as follows:

1. The Legitimate user (client) sends the request to the server.
2. The request is directed to the server which contains a web application as well as the database.
3. The request may contain a suspicious tainted source and it is redirected to an Anti XSS Mechanism.
4. The Code evaluator evaluates the code and identifies the tainted source and sends it for refinement.
5. The Data Refiner mechanism refines it with two main techniques:
   a. Encoding the Meta Characters.
   b. Validating the input properly with Input Validator.
6. The Data Refiner sends the validated URL and other inputs supplied by the tainted source.
7. The XSS Validator, in the XSS Eradicator denies the request if the data refiner had marked the URL Request as malicious.
8. The non malicious URL is send to the Context analyzer where the entire context is analyzed for suspicious code and confirms the attack free context.
9. If the suspicious code is found then it is send to the code replacer , where all comments or instance of malicious code is replaced by model which contains trusted code and not making any changes to trusted content.
10. This trusted content is sent to the server and stored in the database, then the user receives or gains the access.
11. Else the access is denied.

The proposed mechanism has the ability of supporting detection mechanism and as well as defending from attack simultaneously. The overall experimental result justifies the proposed work improves the efficiency and effective in preventing the XSS (Cross Site Scripting) attack.

Proposed Defense Mechanism

In this study, a website in *php* has been developed and hosted on the local host (*XAMPP* server). The experiments to exploit *XSS* vulnerabilities in the website have been performed

Data Refiner Mechanism:

XSS attack is a type of code injection where user input is misinterpreted as program code rather than data, thus secure input handling is needed to prevent this code injection. To mitigate *XSS* attack, the following mechanisms are developed and have been used in the proposed system in order to eradicate the XSS vulnerabilities. The data refiner mechanism contains two main techniques: 1. Encode meta characters (URI Encoding) and 2. Input Validation (HTML Response body, eg., Forms)

Encode Meta Characters:

Encoding of the meta character supplied in the tainted source like uri is handled by urlencode() function in php to overcome XSS vulnerabilities.

The flow of control is depicted in the following steps:

Step 1: Hacker posts vulnerable script with a tainted source

<script> “Vulnerable Code ” </script>

Step 2: assign vulnerable code as vc, vulnerable string

Step 3: urlencode(vc) = evc

Step 4: Encoded input, evc is passed to XSS Eradicator.

Step 5: The Validator interprets script in url as model and not as a script, thus protects from the attack.

Step 6: Proceeds to next step of refinement i.e.,Validation

Any portion of URI can be manipulated for XSS. The major tainted sources like URI, HTML Response body and Html Request header are the vulnerable area where the attackers focuses more. Therefore, Encoding of meta character supplied by this tainted source are handled by urlencode() function in php to eliminate the cross site scripting vulnerability. This encoded script is later supplied to the next phase XSS eradication.

Input Validation:

Input validation is refinement technique where the encoded tainted source like an HTTP request body which contains nothing but the data collected from the users e.g., forms. Since this is the direct input from the user, it can be immediately marked as a tainted source. filter_input_array() function to filter the POST variables received from a form in php is used to prevent the insertion of malicious code into the web application and thus mitigates the execution of XSS vulnerable scripts.

Step 1: Handles HTTP Request Body, typically the Form.

Vulnerable Code can be injected in form

Hacker posts vulnerable script with a tainted source (Form)

<script> “Vulnerable Code” </script>

Step 2: filter_input_array(INPUT_POST, $filters)

(Only content inside the tag is validated)

Step 3: Server inserts input in to the Data validator where each data is validated

FILTER_VALIDATE_post variable

Step 4: Browser interprets this refined input as data not as code.

Step 5: Thus, Eliminates the execution of XSS Vulnerable scripts.

XSS Validator :

In the second phase of protection, XSS validator handles the data from XSS gauge which is again refined and validated for its vulnerabilities. It contains two main components: 1. Context Analyzer and 2. Code Replacer.

Context Analyzer:

It analyze each context and identifies the vulnerable in the data available. Each character is examined and defined by XSS Validator. The context analyzer CA is based on heuristic knowledge analysis on a number of technical aspects and the findings of various threat report identified earlier and stored. In particular, trend analyses of CA provides an understanding of what type of threat occurred in the past as an indicator of what may occur in the future, and insists on short, medium, and longer-term analytic reports. It helps in decision making and creates a new context analysis rules to enhance the defensive mechanism. The task of CA is as follows:

a) Examines and records all harmful elements in the given context based on heuristic knowledge base.
b) Identifies the features of proposed analysis requirements which exacerbate existing protection issues.
c) Target validation is done with heuristic knowledge which contains vulnerable labels.
d) Creates a new target CA rules which is stored in XSS validator and supplied to server for extensive protection by the XSS Eradicator.

The above tasks makes the XSS Validator relevant and effective in achieving its original objectives in a context analysis also identifies the affected part by the vulnerabilities and inequalities which are examined by the context analyzer mechanism.

Code Replacer:

In this technique, all comments or instance of malicious code is replaced by model which contains trusted code and not making any changes to trusted content. It works as follows :

a) If there is no vulnerabilities after the refinement the data is passed to the server without any change thus promotes the web access.
b) Else If vulnerabilities are found by the validator it is converted into a model not treated as data or as script. This model is developed as a Trusted Model by applying the white listing technique.

This model is stored in the database and provided to the server. Therefore the browser finally gets only this information from the server and interprets this trusted model and not the vulnerable code.

Merits of the proposed system:

- Encoding meta chars (in URI) and input validation (in HTTP Request Body) successfully mitigate XSS attack risks.
- These techniques requires minimal configuration.
- These techniques addresses XSS vulnerabilities from all interfaces and the security mechanism is centralized.
- Whenever any up gradation of this system is required only the forms need to be changed for the web application stored for each web page. Web pages do not require changes in it.

Experimental Evaluation

This system needs a standard test bed for testing purpose, we used the open source projects for this testing need and were taken from a site gotocode.com, which provides web applications which are open source projects. The web applications of this site have been used as a test bed in many earlier works[17, 18, 19]. They used this test bed for general web prone attacks and SQL injections attacks.

In this system, the work was tested on 5 open source projects for XSS attacks. The five projects that were taken for study are Health care centre, Transport portal, Registration form, Financial service, Stationary store. These five web applications are considered and developed based on the vulnerability analysis of stevemorgan [20] which gives the top most vulnerable domains for attack. The Burp suite [21] was used as an attacking tool. The inputs to these applications were the XSS attack vectors and commands. This system was able to detect all the attacks injected by burp suite and was able to achieve 95% detection rate. The total number of XSS attacks identified and the total number of detections fixed by the proposed system along with False positive defines the detection rate. The analysis are stated in Table. 1

Table 1: Detection analysis of XSS Attacks with False Positive

Web Application	XSS Identified	FP	Fixed
Health Care Centre	13	1	12
Transport Portal	11	1	10
Registration Form	8	0	8
Financial Service	16	1	15
Stationary Store	9	0	9
Total	**57**	**3**	**54**

The detection speed, comparing with the existing systems [10, 22, 23, 24], the proposed Anti XSS is fast and improves efficiency of detection and also reduces the false positive (FP).

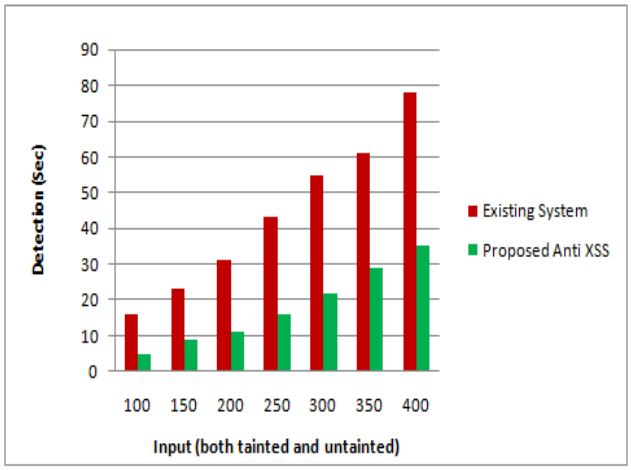

Figure 2: Detection analysis of XSS attack

Figure 2, depicts the analytic results of the detection speed comparing to the existing systems. The existing systems observed in the related work follows either server side or cleint side detection techniques and hence identifies only few types and does not fixes all the classifications of XSS attacks. The proposed Anti XSS is a hybrid approach which identifies all types of XSS attacks and fixes it. The eradication of XSS is governed in figure 3, the average accuracy rate is 98.4 % which is far higher comparing to the existing systems.

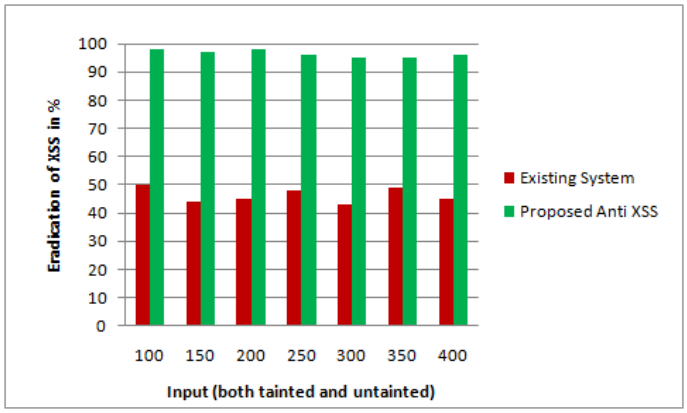

Figure 3: Eradication of XSS attack

The various services and its analysis is shown in figure 4, the entire analysis of the proposed system governs the analytic results of the Anti XSS system which concludes with high accuracy with a 99% of overall security and address and fixes almost all the XSS vulnerabilities.

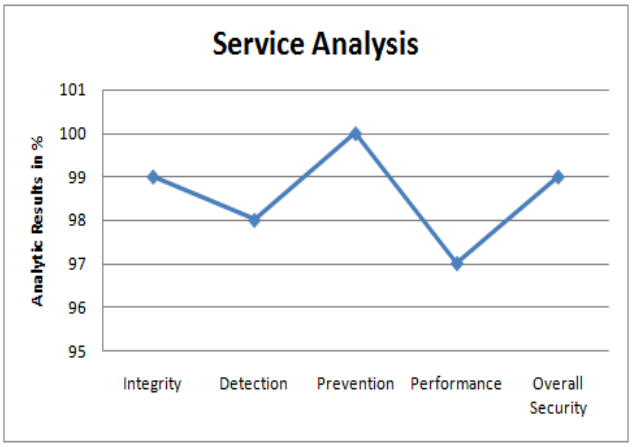

Figure 4: Analysis of various services in Anti XSS

CONCLUSION AND FUTURE WORK

The XSS attack is considered as one of the top five web application vulnerabilities leading to security breach. There are a wide range of defensive techniques are available to prevent XSS attacks. But these techniques are implemented either on the client-side or on the server-side to protect web applications from XSS attacks. A sloppy validation of input on the web application leads to the stealing of cookies from web browser. Now a days, the hackers are more powerful because they attack all sophisticated defensive systems therefore we need to develop new approaches to eradicate XSS attack. The cross site scripting attack is considered as one of the major threat in web vulnerabilities, which needs an efficient approach on the server side and also on the client side to protect the users who depends on web applications.

To overcome this attack, this paper presents an effective approach for mitigating XSS attacks and its vulnerabilities in Web Applications. Our proposed system acts as an Anti XSS tool and protects both server as well as client side. This new approach identifies the attack and detects it using a data refiner technique and secures them with appropriate encoding technique which prevent input values from causing any improper validation and execution of malicious script. This Anti XSS tool, contains two main mechanism called XSS Gauge and XSS Eradicator, to implement the proposed approach. Using this tool, we tested our proposed mechanism with the standard test bed applications and our work has shown a significant improvement in detecting and defending XSS Attacks.

Future scope of this work can be extended because this system addresses the current XSS vulnerabilities and if new XSS attacks are introduced then new Anti XSS tools should be proposed. New algorithms can also be proposed to process the attack vectors to decrease the processing time of user input in the web applications.

REFERENCES

[1] Chih-Hung Wang, Yi-Shauin Zhou, "A New Cross-Site Scripting Detection Mechanism Integrated with HTML5 and CORS Properties by Using Browser Extensions", *Computer Symposium (ICS) 2016 International*, pp. 264-269, 2016.

[2] https://www.netiq.com/promo/security-management/2015-*cyberthreat-defense-report.html*.

[3] Mutton, P. "PayPal Security Flaw allows Identity Theft", *http://news.netcraft.com/ archives/ 2006/06/16/paypal_security_flaw_allows_identity_theft.html*, June 2006.

[4] Mutton, P. PayPal XSS Exploit available for two years?, *http://news.netcraft. com/archives/ 2006/07/20/paypal_xss_exploit_available_for_two_years.html*, July 2006.

[5] Ashwin Garg, Shekhar Singh. A Review on Web Application Security Vulnerabilities. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3 January 2013.

[6] M. T. Louw and V N. Venkatakrishnan, "Blueprint: Robust Prevention of Cross-Site Scripting Attacks for existtng browser" *Proc. 30th IEEE Symp Security and Privacy (SP 09), IEEE CS*, 331-346, 2009.

[7] O.Hallaraker and G.Vigna, "Detecting Malicious JavaScript Code in Mozilla", *In Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems*, 2005.

[8] Shasank Gupta and Lalitsen Sharma, "Exploitation of XSS vulnerabilities on real world web applications and its defense", *IJCA, Volume 60*- No.14, December 2012.

[9] A.Kieyzun, P.J. Guo,K. Jayaraman, and M.D. Ernst, "Automatic Creation of SQL Injection And Cross-Site Scripting Attacks", *ICSE '09 Proceedings of the 31st International Conference on Software Engineering*, 199-209, May 2009.

[10] Shasank Gupta and B.B. Gupta, "BDS: Browser Dependent XSS Sanitizer", *IGI-Global, Handbook of Research*, pp.174-191, Nov 2014.

[11] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna, "Cross site scripting prevention with dynamic data tainting and static analysis". *In Proceeding of the Network and Distributed System Security Symposium* (NDSS07), 2007.

[12] K. Selvamani, A.Duraisamy, A.Kannan "Protection of Web Applications from Cross-Site Scripting Attacks in Browser Side" (IJCSIS) *International Journal of Computer Science and Information Security*, Vol. 7, No. 3, March 2010.

[13] Nunan, E. Souto, E. M. dos Santos, and E. Feitosa, "Automatic classification of cross-site scripting in web pages using document based and URL based features," *IEEE Symposium on Computers and Communications* (ISCC), pp. 702-707, 2012.

[14] Ashwin Garg, Shekhar Singh, "A Review on Web Application Security Vulnerabilities", *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 3, January 2013.

[15] K. Vijayalakshmi, A. Anny Leema, "Extenuating Web Vulnerability with a Detection and Protection Mechanism for a secure Web Access" , *In Proceedings of the IEEE International Conference On Signal Processing, Communications and Networking* (ICSCN'17), 2017.

[16] Pratik Soni, Enrico Budianto, Prateek Saxena, "The SICILIAN Defense: Signature-based Whitelisting of Web JavaScript", CCS'15, October 12–16, ACM, 2015.

[17] Bhawna Mewara, Sheetal Bairwa, Jyoti Gajrani, "Browser's Defenses Against Reflected Cross-Site Scripting Attacks", IEEE International Conference on Signal Propagation and Computer Technology, pp: 662 - 667, July 2014.

[18] Lee, Inyong, Soonki Jeong, Sangsoo Yeo, and Jongsub Moon, "A Novel Method for SQL Injection Attack Detection Based on Removing SQL Query Attribute Values", *Mathematical and Computer Modelling*, Vol. 55, no. 1, pp. 58-68, 2012.

[19] Kanpata Sudhakara Rao, Naman Jain, Nikhil Limaje, Abhilash Gupta, Mridul Jain, Bernard Menezes, "Two for the price of one: A combined browser defense against XSS and clickjacking", *International Conference on Computing, Networking and Communications, Communications and Information Security*, 2016.

[20] https://www.forbes.com/sites/stevemorgan/2016/05/13/list-of-the-5-most-cyber-attacked-industries/

[21] Burp suite, http://portswigger.net/burp/

[22] Shashank Gupta and Brij Bhooshan Gupta," XSS-immune: a Google chrome extension-based XSS defensive framework for contemporary platforms of web applications, *Security And Communication Networks*, Security Comm. Networks, 2016.

[23] R. Pelizzi and R. Sekar "Protection, usability and improvements in reflected XSS filters" Proc. of the 7th ACM Symposium on Information, *Computer and*

Communication security, ASIACCS‟12, Seoul, Korea, 2012.

[24] Gupta, S. & Gupta, B.B. Arab J Sci Eng (2016) 41: 897. https://doi.org/10.1007/s13369-015-1891-7.