Middle-East Journal of Scientific Research 24 (12): 3748-3754, 2016
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2016.3748.3754

Shielding Cross-Site Scripting Attacks Using the State of Art Techniques

[1]Megala Manickam and [2]Uma Maheshwari Govindasamy

[1]Department of Applied Mathematics and Computational Sciences,
PSG College of Technology, Coimbatore - 641038, Tamil Nadu, India
[2]Department ECE, PSG College of Technology, Coimbatore - 641038, Tamil Nadu, India

Abstract: Malicious codes are inserted by the attackers to steal information such as cookies and credentials that appears on the web browser. These codes includes HTML tags and JavaScript functions. It becomes a tough task for the user to sanitize query from malicious code that could be hooked. Nowadays dynamic websites are designed to increase the interactiveness between services that are available over internet. However the features used by the browsers to support these resources increases the security risks and makes way for the new malicious code injection or Cross-Site Scripting (XSS). XSS stands in the top list of threats that are exposed to web applications in the recent years. The main objective of this paper is to conduct a systematic review of the studies that are done on XSS vulnerabilities and attacks. Even though certain studies have addressed through the detection and prevention mechanisms, we have noted that many contributions addresses the dynamic analysis techniques. No single solution can effectively invade the intruders. More researches have to be conducted to detect new attacks and prevent them from affecting source code before it is deployed. We have also made a survey of the applications and limitations of the proposal that are taken for study.

Key words: Cross Site Scripting · Attacker · Vulnerabilities · SQL Injection

INTRODUCTION

With the increased growth of internet and the amount of information available, users exchange sensitive information without considering the security. In recent year crimes related to web hacking has increased. Attackers involved in these activities captures system information and vulnerabilities to penetrate through their system to harm their secured data. The vulnerabilities can be of one of these types: Improper validations on inputs, unpatched software, security controls disabled, default or remember passwords [1]. Hence information security is proposed to deny unauthorised access and destruction to maintain secure transmission of information through web. Information exchange done via web makes the user to interact where the user enters his/her information which is then passed to the web server. The communication is made through the application code that is exposed to the user on his/her session [2]. Web applications use JavaScript code and HTML Tags to embed web pages to support client-server architecture. The code is executed in the web browser. Some times users are unaware of the JavaScript codes that are automatically inserted into web documents which manipulate HTML documents. These types of attack are termed as Cross-site Scripting attacks. Cross-Site Scripting (XSS) is considered as the most common application layer hacking techniques. XSS is of three types: Reflected, Stored and DOM based. Reflected XSS is executed by the browser of the victim and triggered when the victim provides input to the web site. Stored XSS attacks are the malicious code/script stored in the databases, comments or other fields by the attacker that is executed in the client side. Attackers may collect sensitive or secured significant information from the victim system.

The purpose of this work is to forecast the results of the systematic literature review conducted in the recent state of research on XSS attacks. The review is made to cover the period of past 5 years (i.e) from 2010 to 2016. The paper is structured as follows. Section 2 describes the methods that are applied in the study and results are presented in Section 3. We conclude the paper in Section 4.

Corresponding Author: Megala Manickam, Department of Applied Mathematics and Computational Sciences,
PSG College of Technology, Coimbatore - 641038, Tamil Nadu, India.

Table I: Confusion Matrix

	Classification	
Real Class	XSS	Non XSS
Attack (XSS)	TP	FN
Normal (Non XSS)	FP	TN

Research Methodology: This study presents the literature review of certain research studies that are carried on XSS. We have referred the review article of Isatou for organizing the paper [3].

Search Process: As proposed by Isatou we have used the following search terms to collect articles needed for this study. These databases are open access articles which contain context related to our field of research. Certain case studies also use these references for carrying systematic literature reviews in XSS attacks. The database and the URL are shown in Table I. The following search terms were used in this survey:

- Cross-Site Scripting
- XSS Vulnerabilities
- Cross Site scripting Attacks
- Recent Research on Cross Site Attacks
- Scripting Survey Cross Site
- Removal of Vulnerabilities in XSS
- Classification of XSS Attacks

The articles obtained through this process are downloaded from the search criteria made such that the recent publications (5 years) were covered. The topics included for our research are security vulnerabilities such as SQL injection and methods proposed to address the problem. The aim of study, year of publication, problem discussed, solution proposed and type of XSS were addressed.

Organizational Statistics: White Hat Security Statistics have analysed the financial and banking sectors to explore the reasons why the banking websites are frequently vulnerable to cyber attacks. High fraction of vulnerable attacks is exposed by any one of the two methods.

- Frequency of Modification of Source Code of Web Applications
- Static Analysis of Source Code of Web Application

Varieties of defensive mechanisms have been employed by organisation to control the behaviour of web application. These defensive mechanisms include

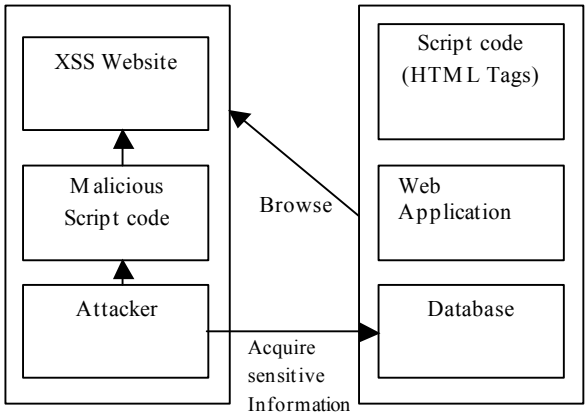

Fig. 1: Process of XSS attack

- Providing computer based software training to the programmers
- Enforcing library in web application to centralise security controls
- Static analysis
- Deployment of Firewall
- Anti-Fraud Monitoring System

According to White Hat Security Statistics Report 92% of the web applications from different organization are subjected to Static Code Analysis and 72% include white list libraries.

Injection of Attacks: Cross Site Scripting works by injecting the malicious XSS payload vector to steal sensitive information from a victim system. XSS normally focus on the un-sanitized inputs and theft of victim browser information including cookies, passwords etc by injecting JavaScript code. Bypassing XSS attacks just disables the scripting language on browser which affects readability of the web page. The step followed in the attack injection is depicted in the Figure 1.

- At first the attacker finds a website that is susceptible to XSS attack, then the attacker inject malicious JavaScript code
- When the victim browser browse the website malicious code that comes in the form of current post in HTTP response message.
- When the malicious code is executed victim credentials (e.g. cookies) will be transferred to the web server of the attacker.

Cookies are used by attackers in session hacking. An XSS attack normally occurs in dynamic web applications that require input from the user end.

```
1. $name = $_GET['name '];
2. echo "Hello " + $name + "!";
```

Fig. 2: Example of a Reflected XSS

Taxonomy of Xss Attacks: XSS attacks can be classified into three categories namely DOM based attacks, Persistent and Non-Persistent XSS attacks. Depending on the type of vulnerability, the methods for detection, avoidance and exploitation may vary.

Un-Sanitized Programming XSS: This type of issue is caused by insecure programming. Handling user-provided inputs in unsafe manner can lead to this type of XSS. Every type of XSS attacks a web application in a specific way.

Reflected XSS: This term denotes all non-persistent XSS issues which occur when the web application echos part of the HTTP request in the respective HTTP response's HTML. In order to make this attack a successful one the attacker has to trick the user into sending fabricated HTTP request. This can be done by including a malicious link or hidden iframe into the page. A sample code is shown in Figure 2.

Stored XSS: Stored XSS refers to the attack in which attacker inject malicious script in the vulnerable application's storage. In this type of attack malicious script remains in the application even after the exploited session ends. Hence the user who receives the injected script is exploited to the attacker without any further actions by attacker. Unless like reflected XSS after embedding malicious script successfully into the application, the actual exploitation rely on the outside of the vulnerable web application.

DOM-Based XSS: This is a special variant of reflected XSS in which the logical errors in the JavaScript cause XSS conditions by careless usage of Client side data. This type of issue occur if a JavaScript uses controlled values to alter the HTML content of the web page [5].

Background and Literature Study: Several works have been done in the past related to the cross-site scripting attacks. The survey of these works have been divided in to two categories

- Detection of XSS Vulnerabilities
- XSS Defensive Techniques

Detection of XSS Vulnerabilities: In [6] authors have proposed a clear solution of finding whether the web application is vulnerable or not. The steps for finding vulnerability it is as follows:

- Open a web site from web browser and browse for the areas which require input from the client's web browser side. Such scenario in web application are login forms, bulletin boards, search fields, comments etc.
- Now input any text in to these input fields and submit to the web server of the web application
- After this test the first condition which states that the check the response web page of the web server for the same text or keyword which was supplied by the user. If the response web page contains same keyword then web application is declared as XSS vulnerable. On the other hand if the response web page does not contain any user supplied keyword then check for the second condition
- The second condition is to simply input any script string (<script>"Hello"</script>) in the user supplied areas and submit this input string to the web server of the web application. After submitting this request to the web server of the web application, if the web server replies with a "Hello" message in the pop-up window, then the website is vulnerable to XSS attacks and if it is not display then check for final condition.
- The final condition states that simply search for the same input java script string (<script>"Hello"</script>) in the source code of the web application. If any part of the string is successfully found, then the web application is vulnerable to XSS attack.

Web Page Classification: Static analysis does not provide examples of input values that must be used to make the application to execute the path that causes XSS vulnerability. Runnable Test cases provides an execution path and evidence of vulnerability mechanisms. This enables the developers to understand the problem before fixing them. Hence a combination of genetic algorithm and concrete symbolic execution is used for the generation of security test case. Concrete Symbolic Values consists of tracing input variables when the program is concretely executed. Hence Boolean expressions are interpreted on decision points as conditions on symbolic inputs.

A solver is resorted for reasoning on symbolic values and to elaborate new values that satisfies conditions that are not satisfied by the previous executions. The symbolic value of an input is the name of input parameter. If the

symbolic value of a variable is not an input variable it can be retrieved by querying the SYMP map. Symbolic value of expression is the string that results from quotation of expression where variables are replaced with their symbolic values. This combined strategy is able to generate test cases with higher coverage and it is suggested that if the genetic algorithm shows poor performance the contribution to the combination is more important to improve the results of concrete symbolic execution. Concrete symbolic execution is proven to be effective in overcoming the limitation of each other combined strategy. This helps in finding input values that satisfies narrow conditions while GA helps to explore search space and possibly finding input values when solver gives up due to complex conditions that are modelled by its language [9].

The automatic classification of web pages is divided into four steps. The first step is the detection of encrypted code. In this step the encoding in the form of hexadecimal, decimal, octal and reference characters that fall under the range of ASCII and extended ASCII encoding are identified. Second step is the decoding phase where the web page is decoded by routines that perform alternative encoding conversion to ASCII format in order to make the text intelligible and enable the analysis and extraction of other features. This aims at the extraction of features that are hardly detected in its encrypted state. The third step is the webpage decoding in which the extraction of decoded features is performed. In this stage features are extracted based on HTML/Javascript schemes and suspicious patterns.

Features related to code encryption only cannot determine whether the code is malicious because the alternative encodings are used in non-malicious code. The next step is the classification step that aims to classify web page sample as XSS or non XSS. To accomplish a classifier a set of training samples is provided to the classifier which designs a predictive model. Machine learning methods such as Naive Bayes and SVM have been used for classification. Naive Bayes and SVM deal with the set of labelled sample data that includes both positive and negative samples that are infected with XSS code or not. The experiments were performed using *Dmoz* and *ClueWeb09* dataset as non-XSS web pages while XSSed pages are used as malicious samples [10]. KameleonFuzz a black box XSS fuzzer is proposed to exploit XSS. This generates malicious inputs and detect vulnerability status. The input generation and evolution is achieved with genetic algorithm guided by an attack grammar. A double taint inference permits to detect precisely whether an exploitation attempt is succeeded. This technique have also proved with no false positives and high revealing capability than other black box techniques [8].

The objective of Dynamic Hash Generation technique is to make the cookies useless for the attackers. This approach can be implemented on the web server without making changes on the web browser. Web servers thus keeps track of the name and attribute in the cookie and send hash value to the browser. At each time when browser makes an active connection the browser has to include hash cookie value into corresponding request so that web server rewrite the hash cookie value to the original value generated by the web server. XSS attack can steal the cookie information stored on the browser to hijack user session. Here the web server is made to dynamically generate hash value of the name attribute and send to the web browser. The web browser includes its hash value into its repository [11].

SQL Injection Attacks: Penetration Testing and Fault Injection is applied to emulate the XSS attack. The testing is done via Scanner SOAP UI tool. This tool injects scripts through the add-on security testing and analysis resulting in the response from server and finally classifying the responses in the web services as vulnerable or not by injection of XSS attack. A fault injector WSInject is used to emulate XSS by acting as a proxy between the client and servers. The interception and modification of SOAP message exchange are transparent between client and servers. There is no need of source code to be altered or viewed and only constraint is that client and server has to be connected through the proxy [7]. Attributes based on hybrid static and dynamic code analysis that characterize input validation and sanitize code patterns for predicting vulnerabilities related to SQL injection and cross site scripting. For a security sensitive program statement the hybrid attributes are collected and their nodes are classified from its data dependency graph. This method is used to build effective vulnerability predictors using both supervised and unsupervised learning methods. It is shown that the hybrid attributes predict vulnerabilities more accurate than static analysis [12]. An integrated model to prevent SQL injection attacks and reflected cross-site scripting have been proposed. This model is divided in to two modes safe mode and a production mode environment. In safe mode a security query model for SQL injection and sanitizer model for reflected XSS for each of the identified SQL queries and input entry points for the reflected cross site scripting attacks. In production environment input entries that create dynamic SQL queries are validated

against security query model generated in safe mode and normal input text entered by the user is validated by sanitizer model instrumented in the code at safe mode [13].

Detection Algorithms

Machine Learning Algorithm: *Genetic Algorithm:* Genetic Algorithms are optimization heuristics that are inspired by natural evolution from biology. At first an initial population composed of random set of input values are evolved until maximum number of generations or final solution is found. For each iteration subset of current population is selected to form next one by giving more chances to the individual that produce final results (Individuals with higher fitness). Selected individuals are made to produce offspring by crossing over their chromosomes and mutating them to breed next population. Here the individuals are turned to HTML requests for the application by encoding them in to corresponding URL with the values passed by GET by representing scheme name = value. Fitness function is calculated such that the amount of target branches that are executed when the application is run with the current individual as population. One point crossover strategy is adopted and mutation changes the value of parameter randomly.

Data Mining Algorithms: *Naive Bayes and SVM:* Naive Bayes is a statistical method that makes classification decisions by calculating probabilities and costs related to each decision. The Bayesian classifiers follows the feature conditional independence by assuming that a feature value on a given class is independent of the values of other features. For example in order to classify a sample "x" the Bayes classifier first calculate the posterior probability. This classification mechanism achieved higher recognition rate by keeping a low computational cost. SVM works by selecting a hyperplane that maximizes a class separating margin in order to divide samples in to different classes. The basic idea of SVM is to nonlinearly map the original feature vector to a higher dimension space in which data can be linearly classified. SVM uses kernel function which allows calculating the hyperplane without performing mapping step.

Decision Tree: Decision tree is a common way to organize classification schemes. Every decision tree starts with a root node and consider it as the parent node for the other nodes. Each node in the tree evaluates an attribute to determine which path it should follow. The Decision tree is tested by comparing a value against some constant. Classification using Decision Tree is performed by traversing the tree from root to leaf node. These classification methods are a classic way to represent the data from a machine learning algorithm that offers a fast and powerful way to express the structures in the data.

Dynamic Hash Algorithm: Dynamic Hash Generation Technique is used to generate the hash of value of name attribute in the cookie. Other attributes are kept constant. Steps followed in Dynamic Hash Algorithm are:

- The user on web browser side submits the User-ID and password to the web server of the web application
- The web server submits the corresponding information from the browser and generates a cookie.
- Now the web server will dynamically generate hash value of name attribute in the cookie and store both values in the form of a table on the server side.
- Subsequently the web server will send the hash value of the name attribute in the cookie in the web browser
- The web browser will store the hash value to the repository.

Since the cookies at the browser database are not valid because of the hash values being replaced, therefore XSS attack will not be able to impersonate the user using stolen cookies which are converted in to hash form. If the browser wants to reconnect to the web server as a part of active connection it has to include cookie with its corresponding request to the web server. The web server will use the information in the table to rewrite back the values of name attribute in the cookie to the original value generated by the web server.

Experiments and Result Evaluation: This section describes the database, algorithm performance and other parameters that are used to conduct experiments and the results are also presented. We have implemented the approach using PHP.

Database: Naive Bayes and SVM works with set of labelled sample data that include both positive and negative samples to detect whether the web pages are infected with XSS code or not. For the positive class 15,000 websites were used. These samples are obtained from XSSed database (http://www.xssed.com) with the attacks occurred from December 2012 to March 2015. The negative samples are taken from *Dmoz* database. The contents of web page are in English and are selected randomly.

Performance Measures: The 10-fold cross-validation was used to evaluate the results. The goal of 10-fold cross-validation is to predict and estimate how a correct model will execute in practice. At first the original dataset is divided into 10 folds. At each run one of the folds is used as the test set and the other 9 folds are put together to form the training set. This process is repeated for 10 times. Confusion matrix is used as a metric for performance measurement which enables the result assessment of positive and false negatives. The confusion matrix shown in Table I are based on the following measures.

$$Detection\ Rate = \frac{TP}{(TP + FN)} \quad (1)$$

$$Accuracy\ Rate = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (2)$$

$$False\ Alarm\ Rate = \frac{FP}{(FP + TN)} \quad (3)$$

Here TN – True Negative indicates the amount of negative samples that are correctly classified and FN – False Negative indicates the amount of malicious samples classified as Negative, FP – (False Positive) indicates the amount of negative samples classified as malicious and TP – (True Positive) indicates the amount of malicious samples correctly classified.

Classification Analysis

Naive Bayes and SVM: Naive Bayes is used with its default configuration parameters. For SVM the polynomial kernel degree is set to 1.0 and regularization parameter "C" as 1.0 (as suggested in [10]). Both the classifiers SVM and Naive Bayes achieved high performance in terms of accuracy, detection and false alarm rate. Naive Bayes provides low computational cost when compared to SVM.

Genetic Algorithm: A population of 70 individuals is evolved for 500 generations. Keeping the 10% of best individual as alive across generations. Execution is stopped over 500 generations or the stopping condition is met. Crossover and mutation probabilities are set to $P_c = 0.7$ and $P_m = 0.01$ respectively. Genetic algorithms are in fact, optimization heuristics that finds local optimum from where improving further is difficult even with the mutation operators. The approach based on the genetic algorithm does not pass the sanity check while choosing population at random search; hence it fails to generate secure test cases. This proves that only easy to cover vulnerabilities can be tested using genetic algorithm.

Table 2: Comparison of Results Obtained By Algorithms Taken for Study

Classifier	Naive Bayes	SVM	Genetic Algorithm	Dynamic Hashing Technique
Detection Rate	95.32	94.09	94.98	98.01
Accuracy Rate	98.96	99.78	98.60	99.89
False Alarm	0.50%	0.25%	0.22%	0.02%

The genetic algorithm is fast and starts to converge over 200 generations to trap in local optima. The final outcome of genetic algorithm states that simple vulnerabilities can be tested and time required is also short.

Hashing Technique: We have implemented the Hash technique with the help of PHP language. It is seen through our experiments that web server is able to generate hash of the value of name attribute for cookie that the browser stores and return this value back to the web server on every subsequent request. The attempt to steal cookie information from browser database also fails since the cookie contains the hash value and not the session information. This technique fails if the attacker finds the method used for randomly generating hash value. For example if the hash code is interpreted as follows:

Set- Cookie: SID = pqrs123

Set-Cookie: SID = abcd456 ; Domain = .trail.com; Path = /area1

Set-Cookie: SID = lmno678; Domain = .trail.com; Path = /area2

Also as suggested in [11] that this degrades the performance of whole system as it is a server side solution. This also increases latency and response time and also cannot intercept the HTTPs and SSL connections. Generating the hash value for the cookies in HTTP header is present in real time applications. The efficiency of the proposed approaches is tested including all the features to compose feature vector to use for classifiers. The dataset comprises of both the training and testing dataset. Table II presents the results attained by Genetic Algorithm, SVM, Naive Bayes and Hashing Technique.

CONCLUSION

Increased use of web paradigm for developing web applications opens a lot of ways for new security threats against the application infrastructure. Hence Cross site

scripting is considered as serious vulnerability. Existing traditional approaches to prevent vulnerabilities are not sufficient thereby making the end users responsible for the protection of web services. Main cause of such attack is the unawareness of the user while accessing a web page therefore XSS attacks are easy to launch and difficult to prevent. Hence web application developers must use the support tools during deployment to guarantee vulnerable free deployment.

In this paper we focus on the problem of preventing XSS attack. We have presented a deep survey of the XSS attacks with detection and prevention techniques. Whether the attacks falls under persistent or non-persistent, there are many solution to solve the vulnerability problem. But some of these solutions may have some failures and does not render enough security. The study reveals us that more improvement needs to be done in web application security to obtain better results

REFERENCES

1. Examples of system security vulnerabilities, Available at: http:// www.crime-research.org/ news/05.05.2004/ 241/html.
2. Web Application Security and the OWASP Top 10, Available at:www.sapient.com/assets/ImageDownloader/813/Web_Application_Security.pdf.
3. Hydara, I., A.B.M. Sultan, H. Zulzalil and N. Admodisastro, 2015. Current state of research on cross-site scripting (XSS)–A systematic literature review. Information and Software Technology, 58: 170-186.
4. Gupta, S. and B.B. Gupta, 2015. Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. International Journal of System Assurance Engineering and Management, pp: 1-19.
5. Johns, M., 2011. Code-injection Vulnerabilities in Web Applications—Exemplified at Cross-site Scripting. It-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik, 53(5): 256-260.
6. Gupta, S. and B.B. Gupta, 2014. BDS: browser dependent XSS sanitizer. Book on Cloud-Based Databases with Biometric Applications, IGI-Global's Advances in Information Security, Privacy, and Ethics (AISPE) series, pp: 174-191.
7. Salas, M.I.P. and E. Martins, 2014. Security testing methodology for vulnerabilities detection of xss in web services and ws-security. Electronic Notes in Theoretical Computer Science, 302: 133-154.
8. Duchene, F., Rawat, S., Richier, J.L. and Groz, R., 2014, March. KameleonFuzz: evolutionary fuzzing for black-box XSS detection. In Proceedings of the 4th ACM conference on Data and application security and privacy (pp. 37-48). ACM.
9. Avancini, A. and M. Ceccato, 2013. Comparison and integration of genetic algorithms and dynamic symbolic execution for security testing of cross-site scripting vulnerabilities. Information and Software Technology, 55(12): 2209-2222.
10. Nunan, A.E., E. Souto, E.M. dos Santos and E. Feitosa, 2012, July. Automatic classification of cross-site scripting in web pages using document-based and URL-based features. In Computers and Communications (ISCC), 2012 IEEE Symposium on (pp: 000702-000707). IEEE.
11. Gupta, S., L. Sharma, M. Gupta and S. Gupta, 2012. Prevention of cross-site scripting vulnerabilities using dynamic hash generation technique on the server side. International journal of advanced Computer Research (IJACR), 2(5): 49-54.
12. Shar, L.K., H.B.K. Tan and L.C. Briand, 2013, May. Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis. In Proceedings of the 2013 International Conference on Software Engineering (pp: 642-651). IEEE Press.
13. Sharma, P., R. Johari and S.S. Sarma, 2012. Integrated approach to prevent SQL injection attack and reflected cross site scripting attack. International Journal of System Assurance Engineering and Management, 3(4): 343-351.