Efficient Prediction of Cross-Site Scripting Web Pages using Extreme Learning Machine

S.Krishnaveni
Department of Computer Application
PSGR Krishnammal College for women
Coimbatore, India
Skvmphil12@gmail.com

K.Sathiyakumari
Department of Computer Application
PSGR Krishnammal College for women
Coimbatore, India

Abstract— Malicious code is a way of attempting to acquire sensitive information by sending malicious code to the trustworthy entity in an electronic communication. JavaScript is the most frequently used command language in the web page environment. If the hackers misuse the JavaScript code there is a possibility of stealing the authentication and confidential information about an organization and user. The attack is based on the malicious JavaScript code inserted into pages by intruders or hackers. Various attacks like redirect, script injection and XSS which usually include to transmitting private data to attacker or redirecting the victim to web content controlled by hacker. A cross-site scripting vulnerability allows the introduction of malicious content on a web site that is then served to users. Therefore filtering malicious JavaScript code is necessary for any web application. The aim of this work is to analyze different malicious code attacks phenomenon, various types of malicious code attacks. The experimental results obtained on XSS classification in web pages using Extreme Learning Machine techniques. ELM approach can be found in its high sparseness, it can also be seen that ELM accomplishes better and more balanced classification for individual categories as well in very less training time comparative to other classification algorithms. The data are collected from the real web pages and various features are extracted to classify the malicious web page using supervised learning algorithms and the results demonstrate that the proposed features lead to highly accurate classification of malicious page.

Keywords— DOM-based attack, ELM, KELM, Malicious code, OWASP, Redirect attack, SOL injection attack, Web surfing, XSS.

I. INTRODUCTION

In general, user access a web application using the web browser at the same time web browser request the resources from the web server of a particular web application, and the web server respond with the resources through HTTP protocol [3 and 8] in which no sessions are maintained. Usually web applications are uses cookies to provide mechanism for creating stateful HTTP sessions. Cookies are used to store session IDs and authentication for the web application [13] so the cookies are the most and very interesting target for the hackers since; the cookies are the essential key for identify and authenticate the users. Nowadays Cross-Site Scripting (XSS) attacks are the common vulnerable attack in web application through the injection of malicious code in advanced HTML tags and JavaScript functions.

Cross site scripting (XSS) vulnerability is mainly caused by the failure of web applications in sanitizing user inputs embedded in web pages. Even if state-of-the-art defensive coding methods and vulnerability detection methods are often used by the developers and security auditors, still XSS flaws remain in many applications because of (i) the difficulty of adopting these methods to detect vulnerability, (ii) the insufficient implementation of these methods, and/or (iii) the lack of understanding the XSS problem. Recently, the attacks against web applications are SQL injection attack and cross site scripting attack, be subject to increase the detection methods against web application attacks such as pattern recognition, parsing and listing methods (web application firewall) have been developed. However, the detection techniques of web application attacks are developing still now. The rapid growth of internet resulted in future rich usage of dynamic web applications increases the features, also introduced completely new underestimated attack vectors. Cross site scripting (XSS) attacks are currently the most exploited security problems in modern web applications [11,12]. These attacks make use of vulnerability code in web applications and resulting in serious consequences, such as stealing of

cookies, passwords and other personal credentials. It is caused by malicious scripts, which do not sanitize the user input. Several server-side countermeasures for XSS attacks are exist, but such techniques have not been commonly applied, because of their deployment overhead. The formation of dynamic websites comprised of a set of objects such as script functions, HTML tags, hyperlinks and advanced features in browsers lead to several resources and inter-activeness in services currently provided on the Internet. However, these features have also increases the security risks and attacks since they allow malicious codes injection or XSS (Cross- Site Scripting). XSS persist at the top of the lists of the greatest threats to web applications in recent years. The existing client-side solutions degrade the performance of client's system ensuing in a poor web surfing experience. The paper introduces a client side solution that uses a step by step approach to detect XSS, without degrading much the user's web browsing experience.

The rest of this paper is organized as follows: Section II introduces the concepts related to cross site scripting. Section III describes the feature extraction for classification of malicious code attacks in Web pages. Section IV has the experimental results and their analysis. Finally, Section VI presents conclusions and future work.

II. CROSS-SITE SCRIPTING

Grossman [4] defines Cross-Site Scripting (XSS) as an attack vector caused by malicious scripts on the client or server, where data from user input is not properly validated. This allows the theft of confidential information and user sessions, as well as it compromises the client's browser and the running system integrity. The script codes used in XSS are typically developed in JavaScript and embedded in the HTML structure [11]. However, technologies such as Active X, Flash or any other technology supported by browsers can also be used as a vector.

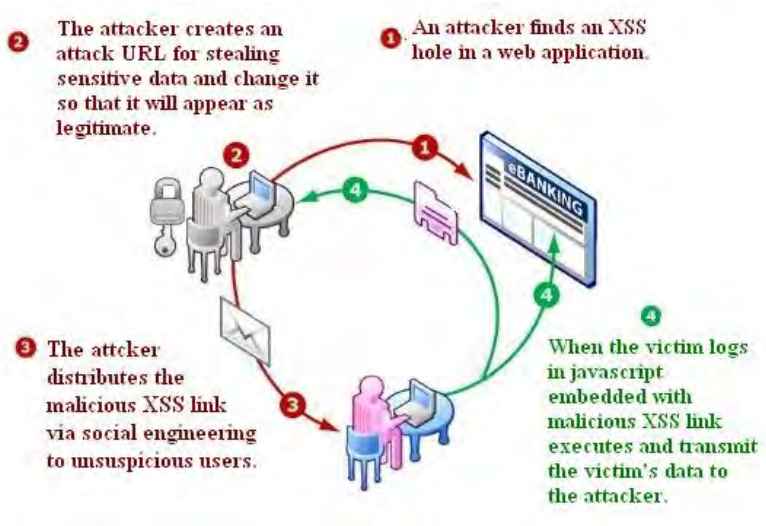

Fig1. An overview of XSS attack

The XSS attacks can be categorized as Persistent, Reflective and DOM-based [13]. In the first case, the malicious code is permanently stored on server resources. Persistent is the most dangerous type of XSS [14]. In the second case, the code runs in the client browser without being stored on the server. This attack is usually made possible through links to malicious code injection. According to the OWASP (Open Web Application Security Project) [10], this is the most frequent type of XSS attack. Finally, instead of using malicious code embedded into the page that is returned to the client browser, the DOM-based XSS enables dynamic scripts on components of the document, modifying the DOM environment (Document Object Model). According to Klein [7], the identification of such an attack requires execution or completion simulation of DOM objects. However, many potentially dangerous schemes [14] that enable this type of attack can be detected prior to its execution. To better illustrate, Fig. 1 presents an overview of XSS attacks. In one of the actions, an attacker can insert a malicious script in resources of a web's server (persistent XSS). Then, it will be permanently displayed on "legitimate" pages returned to other users during regular browsing. In another attack, a hacker can send an e-mail to the target user, which has a link to the malicious script code (reflective XSS) to an URL of a legitimate site. However, this site is hosted on a vulnerable web server. In both attacks, the user receives the requested web page while the browser interprets and executes the page contents and the malicious script code that was injected.

III. FEATURE EXTARCTION

A. Identity Extraction

Identity of a web page is a collection of words that uniquely determines the proprietorship of the website. Identity extraction must be accurate for the successful prediction of malicious web page. In spite of malicious artist creating the script code to steal the user information, there are some identity relevant features which cannot be exploited. The changes in these features are affects the similarity of the web page. This paper anchor tag is used for identity extraction and also used to find the identity of a web page accurately. Features extracted in this identity extraction phase include Meta Tag with Title, Description, Keyword properties and HREF of <a> tag.

B. Feature Extraction

Feature extraction acting an important role in improving the classification effectiveness and computational efficiency. Distinctive features are extracted from the corresponding URL and source code that help to predict the malicious web page accurately. In a HTML source code there are many features that can distinguish the original web page from the malicious web page. A set of 18 features are extracted for each website, 12 features are already presented in [1] and 6 new features newly added which is not presented in [1]. All the 18 features are explained below,

1. URL length

URL Length is the number of characters of an URL. This feature has been used with good results to classify non malicious and malicious URLs on phishing detection.

2. No. of Domain (URL_Chain)

Number of Domains (URL_Chain) this feature corresponds to the number of domains found in the URL. Attacks such as Redirect present URLs in chain, which are inserted to redirect the victim to pages stored on servers controlled by attackers.

3. Duplicate Special Characters

Duplicated Special Characters (Doubled_Char) is corresponds to the identification of an ill-formed special string of characters that is inserted into the opening and closing of tags.

4. Keywords

Keywords are corresponds to the keywords commonly found on page redirects related to the spread of malware and phishing attacks associated with XSS attacks. For example: XSS, banking, redirect, root, password, crypt, shell, spray, eval, etc.

5. HTML Tags

It identifies the presence of potentially vulnerable elements to execution of malicious code such as: <script>, <iframe>, , <img> and etc.

6. Tag Split

It describes the tag chain. That contains the more than one tag nested in it. It will not identified by the antivirus software and the browser so in background it allows the hackers to run malicious script and steal the information.

7. Link

The Link object represents an HTML link element. The link element should be placed inside the head section of the HTML document, and it is used to specify a link to an external resource. A common use of the <link> tag is to link the external style sheets.

8. Applet

The applet tag contains the PARAM name and value; the malicious codes are injected using the param value by the hackers.

9. Object

The <object> tag defines an embedded object within an HTML document. The linked file is actually an HTML file that can contain the XSS code: <OBJECT TYPE=”text/x-scriptlet” DATA=http://www.ha.ckers.org/scriptlet. html><OBJECT>.

10. Embed

The <embed> tag defines a container for an external application or interactive content (a plug-in). If the attributes presents with allowScriptAccess="never" and allownetworking="internal" it can mitigate the risk. For Example: <EMBED SRC="http://ww.ha.ckers/some website link>. The EMBED tag a Flash movie can be added, it may that contains XSS.

11. Cookies

Cookie theft is the process of exploiting the XSS vulnerability (Non-persistent/persistent) and steals the cookie from the victim who visits the infected link.

12. Referrer

Referrer checking will help with some classes of XSS - reflected XSS and DOM XSS but will have no effect on stored XSS. The document.referrer property is set by the browser and represents the page which linked to the current page.

13. Script Functions

JavaScript is the scripting language of the web to create a dynamic web page. There are lots of JavaScript functions are used in the context of malicious code injection such as aler (), eval(), write(), getElementByTagname(), formCharCode() and etc.

14. Form Action

It can be able to changes your web appearance. When the web page is published, the codes that are inserted by the hackers are going to work so that the web appearance is change.

15. Windows

The window object represents an open window in a browser. If a document contains frames the browser creates one window object for the HTML document and one additional window object for each frame. The window object may use to steal user information.

16. Document

When an HTML document is loaded into a web browser, it becomes a document object. The document object is the root node of the HTML document and the "owner" of all other nodes. The document object provides properties and methods to access all node objects, from within JavaScript code.

17. Directly Executed Functions

The directly executed functions are used to exploit the cross site scripting vulnerability while directly executing these functions without use any link or other references to the web page.

18. EventHandlers

An event handler is a callback routine that operates asynchronously and handles inputs received into a program (events). On the input side, events include opening or closing files and data stream operations, reading data from files and so forth. For Example: onClick(), onLoad(), <a ONMOUSEOVER="alert(1)" href="#">read this!.

Thus a group of 18 features describing the characteristics of a webpage are extracted from the HTML source code and the URL of a webpage by developing .NET 2008 code. All the features mentioned above are multi features. The feature vectors are generated for all the three types of attack like 100 of XSS web pages and 100 of Redirected web pages and 100 of Script Injected web pages based on that the training dataset is created.

C. Data Collection

The training dataset is developed using 1500 webpages that have been collected from xssed. It is an archive consisting of collection of malicious webpages and suspected webpages. Out of 1500 websites, 300 webpages are falls under the category of malicious web pages based on the attack it can be categorized. For each webpage, a set of eighteen features are extracted using the URL of the webpage and entire HTML source code. The generated feature vectors are trained with classification algorithms in MATLAB 2012.

IV. EXPERIMENTS AND RESULTS

The experiments have been carried out by implementing classification algorithms such as Extreme Learning Machine (ELM) and Kernelized- Extreme Learning Machine (KELM). These supervised learning algorithms are implemented using MATLAB 2012. The dataset can be separated into training set and test set. For training the models 80% of the data are divided as training set from the final dataset so the training data contains 240 records from the 300 records. As well as the tested data is also contains the 20% of data from the final dataset finally the test set holds 60 records from the 300 records.

Here, the two algorithms such as Extreme Learning Machine (ELM) and Kernelized- Extreme Learning Machine (KELM) are compared based on the learning time and the accuracy. Even though the accuracy level is nearer the learning time is differ for each. The Extreme Learning Machine (ELM) is implemented by using MATLAB. Here the ELM is used to classify the malicious web page. There are two algorithms are used for classification such as Basic-ELM and ELM-Kernel finally the result is compared based on the four criteria like Training Time, Testing Time, Training Accuracy and Testing Accuracy. Fig 2 and 3 tells about the classification result for both ELM and KELM. And table 1 shows the comparative analysis between two algorithms.

TABLE I COMPARISON OF ELM AND ELM-KERNEL

Classifier	Evaluation Criteria			
	Training time (sec)	Testing time (sec)	Training Accuracy (%)	Testing Accuracy (%)
ELM	0.1250	0.0313	0.7784	0.9962
ELM-Kernel	0.1796	0.0065	0.7218	0.9740

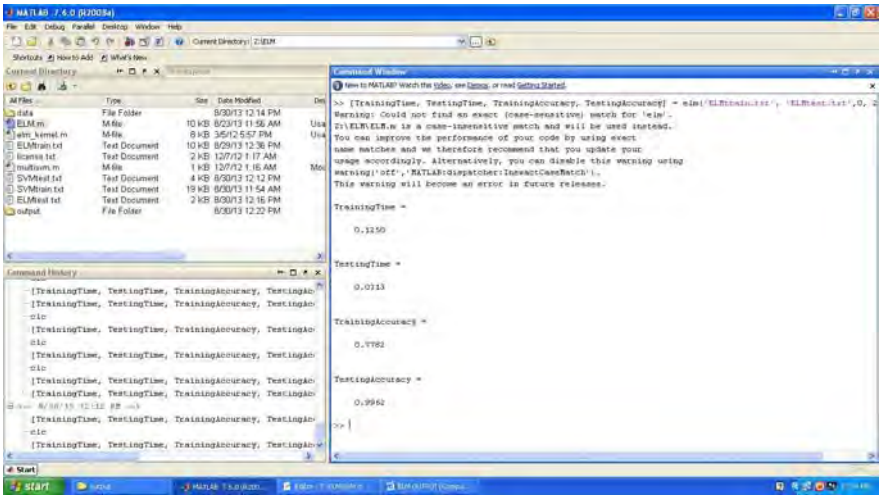

Fig2. Classification result of ELM in MATLAB

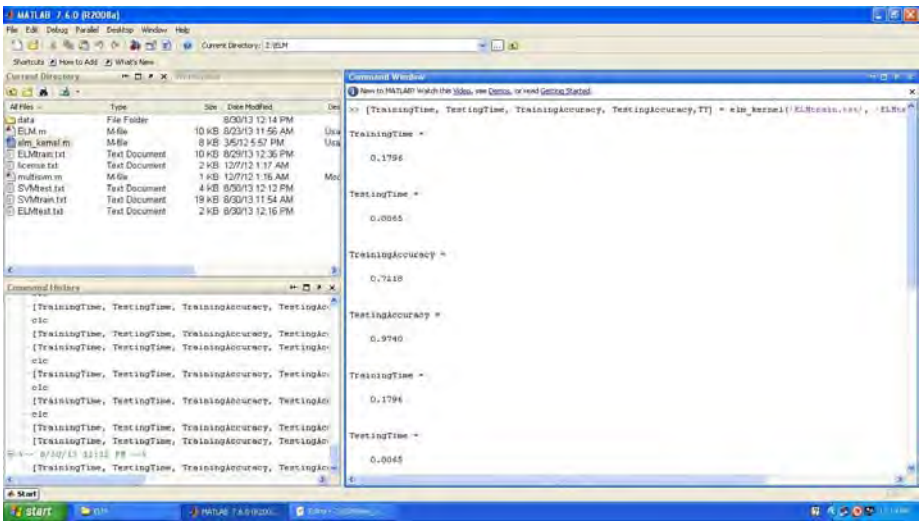

Fig3. Classification result of ELM-Kernel in MATLAB

From the above comparative analysis (Table11), it is found that the predictive accuracy shown by Basic ELM with ELM-Kernel, the Basic ELM has higher accuracy than the ELM-Kernel. The time taken to build the model using ELM-Kernel is more, than ELM algorithm. Fig4. Show the chart representation of performance analysis between two algorithms

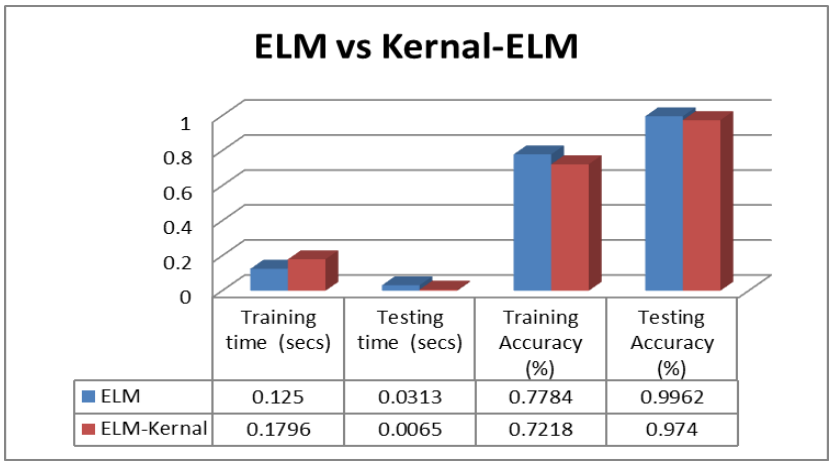

Fig 4. Comparison chart for ELM and KELM

V. CONCLUSION AND FUTURE WORK

This research work describes the modeling of the malicious web page prediction task as classification problem and the implementation of trained model. The trained model has been generated using extreme learning machine and kernelized extreme learning machine. The performance of the trained models is evaluated using 10 fold cross validation based on prediction accuracy and learning time and the results are analyzed. It is observed that about 99% predictive accuracy is shown by ELM based prediction model. As far as the malicious web page prediction is concerned, the training time and the testing accuracy plays major role in determining the performance of the model. In this research work, the implementation of Extreme Learning Machine (ELM) is out performs well for classify Cross-Site Scripting (XSS) web pages. With regards to future enhancement of this research work,

- Number of instances in the training dataset can be increased and the model can be trained with large dataset.
- Other classification algorithms can be used for training and testing.
- More features that help to increase the prediction accuracy can be extracted and used for learning

REFERENCES

[1] Angelo Eduardo Nunan, Eduardo Souto, Eulanda M. dos Santos, Eduardo Feitosa Automatic Classification of Cross-Site Scripting in Web Pages Using Document-based and URL-based Features. In IEEE 2012, pages: 701-707
[2] Feng GR, Huang G-B, Lin QP et al (2009) Error minimized extreme learning machine with hidden nodes and incremental learning. IEEE Trans Neural Network 20(8):1352–1357
[3] D. Gourley, B. Totty, M. Sayer, S. Reddy, and A. Aggarwal, HTTP The Definitive Guide, 1st ed., O'Reilly Media, US, 2002.
[4] Grossman, J., Hansen R., Petkov, D.P., Rager, A. e Fogie, S. "Cross Site Scripting Attacks: XSS Exploits and Defense". Burlington, MA, EUA, Syngress Publishing Inc. 2007.
[5] Guang-Bin Huang , Qin-Yu Zhu, Chee-Kheong Siew "Extreme learning machine: Theory and applications" Elsevier Neurocomputing 70 (2006) pages 489–501.
[6] Huang G-B, Ding X, Zhou HM (2010) Optimization method based extreme learning machine for classification. Neurocomputing 74(12):155–163.
[7] Klein, A. "DOM Based Cross Site Scripting or XSS of the Third Kind: A look at On Overlooked Flavor of XSS". http://www.webappsec.org/projects/articles/071 105.html, November, 2010.
[8] D. Kristol, —HTTP State Management Mechanism, in Internet Society, 2000. Available: http:// www.ietf.org/rfc/rfc2965.txt.
[9] Kunlun LI, Juan ZHANG, Hongyu XU, Shangzong LUO, Hexin LI, "A Semi-supervised Extreme Learning Machine Method Based on Co-training" Journal of Computational Information Systems 9: 1 (2013) pages: 207–214.
[10] OWASP, Foundation. "OWASP Testing Guide", 2008. V3.0. http://www.owasp.org/images/ 5/56/OWASP_Testing_Guide_v3.pdf, October, 2011.
[11] Shar, L.K., Tan, H.B.K. "Auditing the XSS defence features implemented in web application programs" Software, IET on 2012 (Volume: 6, Issue: 4) Page(s): 377 – 390.
[12] Tiwari, S., Bansal, R., Bansal, D., "Optimized client side solution for cross site scripting" Networks, ICON 16th IEEE International Conference on 2008, Page(s): 1 – 4.
[13] Uto, N., Melo, S.P. "Vulnerabilidades em Aplicações Web e Mecanismos de Proteção". Minicursos SBSeg 2009. IX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, Campinas, São Paulo, Brazil, 2009.
[14] W. Xu, S. Bhatkar, and R. Sekar. Taint-Enhanced Policy Enforcement: A Practical Approach to Defeat aWide Range of Attacks. In 15th Usenix Security Symposium, 2006.
[15] Yoan Miche, Antti Sorjamaa, and Amaury Lendasse "OP-ELM: Theory, Experiments and a Toolbox" ICANN 2008, Part I, LNCS 5163, pages 145–154.
[16] Yue, C. e Wang, H. "Charatering Insecure JavaScript Practice on the Web". 18th International Conference on the World Wide Web, Madri. Spain, 2005.
[17] www.extreme-learning-machines.org
[18] www.xssed.com