CLASSIFICATION OF CROSS SITE SCRIPTING WEB PAGES USING MACHINE LEARNING TECHNIQUES

FAISAL SALEH NASSER AL-ASWER

UNIVERSITI TEKNOLOGI MALAYSIA

CLASSIFICATION OF CROSS SITE SCRIPTING WEB PAGES USING MACHINE LEARNING TECHNIQUES

FAISAL SALEH NASSER AL-ASWER

A project report submitted in partial fulfillment of the
requirements for the award of the degree of
Master of Science (Information Security)

Faculty of Computing
Universiti Teknologi Malaysia

JANUARY 2017

I dedicate this project to my beloved parents, brothers and sisters for their endless support and encouragement you have given me throughout all the time

To my respected supervisor, Dr. Anazida Zainal

To my beloved country, Yemen

To all my friends

ACKNOWLEDGEMENT

First and foremost, all prais and thanks are due to Allah, the peace and blessings be upon his Messenger, Mohammed (Peace Be Upone Him). Next, I would like to express my heartfelt gratitude to my supervisor **Dr. Anazida Zainal** for her constant support during my study at UTM. She inspired me greatly to work in this project. Her willingness to motivate me contributed tremendously to my project. I have learned a lot from her and I am fortunate to have her as my mentor and supervisor. Special thanks to Dr. Hamza Hentabli and Dr. Faisal Alsamet for their remarkable help and guidance. I also want to greatly thank my parents, brothers and sisters who cheered me on from the beginning of my study. I thank my dear friend, Eyad for his continuous encouragement and support throughout my project. Last but not least, I am grateful to all my friends for their warm encouragement and support.

ABSTRACT

There are many web application threats such as SQL injection and Cross Site Scripting. According to OWASP 2013 security report, Cross Site Scripting came in third place. Cross Site Scripting is an attack that targets web applications which lack security countermeasures against untrusted data that is provided by the user, and this attack take advantage of these web applications because they do not apply any input validation or output sanitization methods. Few previous works which used machine learning to detect cross site scripting attacks via classification of the web pages into two classes; malicious or benign. The previous works used too many features which considered as irrelevant and noise data because they do not have significant value on accuracy ratio which would cause complexity and decrease the performance of the classification process. They also used URL features which considered unnecessary since URL is considered as the entry point of the attack but cannot activate it since all the different kinds of cross site scripting get activated and run inside the HTML source code. In this study, we focus on how to implement feature selection through Information Gain (IG) to select the most significant features that lead to better performance and less execution time. The selected features used to classify the datasets with three different classifiers to test the performance of these features. The features used in this study were used by previous works, however with IG feature selection, we selected 14 features as the most significant features and the accuracy obtained by using these features was 95.78% compared to when using all features which was 93.11%. The recall was also improved from 88% when all features used to 92.33% when only using the 14 selected features.

ABSTRAK

Terdapat banyak ancaman aplikasi web seperti suntikan SQL dan manipulasi skrip antara laman web. Mengikut laporan sekuriti OWASP 2013, manipulasi skrip antara laman web berada di tempat ketiga. Manipulasi skrip antara laman web adalah serangan yang menyasarkan aplikasi web tanpa perlindungan terhadap data tidak sahih yang diberikan pengguna, dan serangan ini mengambil peluang terhadap aplikasi web tanpa sebarang langkah pengesahan input dan sanitasi output. Beberapa kajian terdahulu menggunakan pembelajaran mesin untuk mengesan manipulasi skrip antara laman web melalui pengklasifikasian kepada dua kelas, iaitu merbahaya atau tidak merbahaya. Kajian terdahulu turut menggunakan terlalu banyak ciri-ciri tidak relevan dan tidak bernilai kerana tidak mempunyai nilai ketara dalam perkadaran ketepatan lalu merumitkan dan melambatkan proses klasifikasi. Mereka turut menggunakan ciri-ciri URL yang dianggap tidak perlu apabila URL ditetapkan sebagai pintu masuk serangan tetapi tidak boleh diaktifkan apabila serangan manipulasi skrip antara web dilancarkan dan beroperasi di sebalik kod HTML. Dalam kajian ini, kami memfokuskan bagaimana untuk mengimplementasi ciri-ciri pilihan melalui dapatan maklumat untuk memilih ciri-ciri paling ketara yang meningkatkan prestasi dan menyingkatkan masa pelaksanaan. Ciri-ciri terpilih digunakan untuk pengklasifikasian kumpulan data dengan tiga klasifikasi berbeza bagi menguji prestasi ciri-ciri tersebut. Ciri-ciri dalam kajian ini turut digunakan dalam kajian terdahulu, namun dengan ciri-ciri pemilihan dapatan maklumat, kami memilih 14 ciri-ciri sebagai yang paling ketara dan ketepatan diperolehi menggunakan ciri-ciri tersebut adalah 95.78% berbanding menggunakan kesemua ciri-ciri iaitu 93.11%. Bacaan kali kedua turut meningkat kepada 88.0% apabila kesemua ciri-ciri tersebut digunakan berbanding hanya 92.33% dengan menggunakan 14 ciri-ciri yang terpilih.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

With the growing of the World Wide Web (WWW) and the Internet-based systems, the web applications are becoming more sophisticated and dynamic. For better user experience and optimized interaction between web applications and their users, functionality and dynamicity of those web applications need to be implemented. However, those web applications might bring threats to the security of the users' data, businesses and the web application itself, and with the increasing growing of web applications, it opens many opportunities for attackers targeting those web services that left vulnerable due to the lack of security awareness and unsecure web development.

An example of a web page that is being requested by a user, the web application server fetches details and information of that web page and sends the response to the web browser that belongs to the requesting user, this response then is interpreted using browser built-in engines such as HTML and JavaScript interpreting engines which will display the web page in the window of the browser. Now, embedded codes give users enhanced experience by implementing dynamic interaction between user and web application, between web application components and web application and the browser as well, such as dynamic menus that response to mouse movement or click and list all options, automatic change of content, hiding or displaying parts of the web

page according to automatic configuration or user action involvement, and storing user information such as cookies in the web browser itself for later use such as re-authentication. Such dynamicity and functionality make web application more interactive, assistive and easy to use, however, attackers who target web applications abuse JavaScript (used for enabling dynamicity and functionality) and use it against those web applications for different goals and intentions such as stealing sensitive information (cookies), misinformation, account hijacking, Denial of Service (DoS), web defacement and many other attacks that are either damaging or just to disturb the users of the website such as meaningless pop up alerts.

There are many attacks that target the vulnerabilities existed in web applications where attackers commit these attacks through JavaScript such as Cross-Site Request Forgery (CSRF), Cross-Site Scripting, Malware Injection and many more. According to OWASP (Open Web Application Security Project) 2013 web security report (OWASP, 2013); a security report released every 3 years cycle, Cross-Site Scripting attacks ranked 3rd in the list. Full list is shown below:

1) Injection flaws: such as SQL injection attacks
2) Broken Authentication and Session Management
3) Cross-Site Scripting (XSS)
4) Insecure Direct Object References
5) Security Misconfiguration
6) Sensitive Data Exposure
7) Missing Function Level Access Control
8) Cross-Site Request Forgery (CSRF)
9) Using Components with known Vulnerabilities
10) Unvalidated Redirects and Forwards

Cross-Site Scripting (XSS) attacks are based on using JavaScript code to attack vulnerable websites that have no validation mechanism for user input and no sanitization for the output coming from the server. More details about XSS, its types and how it works is provided in *Chapter 2* of this report.

The best approach to tackle such attacks is to embrace the need of secure web development where developer should be aware of such security threats and have skills to implement security countermeasures to defend against web applications attacks. However, due to large scopes and unawareness of security, some web applications are left unsecure in some parts of the application code and easily attacked by XSS attacker. Although a great amount of research and many techniques have been introduced to mitigate XSS vulnerabilities and attacks and intensive care of implementing security during development, many of those solutions and web applications still suffer from XSS attacks due to the complexity and sophistication level of attacking approaches and mechanisms.

1.2 Problem Background

Many solutions and approaches have been developed to mitigate XSS either by detect and alert (client side: browser), detect and prevent (client side) or developing secure web applications (server side). For detecting XSS on client side, many techniques had been proposed such as modifying web browser, modifying JavaScript engine, proxy based solution where HTTP request and response get scanned before processed by server (request) and client (response).

A less focused-on approach that is used to detect XSS depends on Machine Learning (ML) techniques. According to the literature reviewed, only few academic papers that highlighted the use of ML in detecting XSS (Likarish et al. (2009), Nunan et al. (2012) and Krishnaveni and Sathiyakumari (2013)). These papers use ML algorithms (such as Naïve Bayes or Support Vector Machine) to classify a web page into two classes which is either *malicious* (infected by XSS) or *benign* (non-XSS). The authors try to detect XSS by classifying HTML code (HTTP response loaded from the server) depending on features which are used to recognize XSS attacks. These features extracted from a dataset and applied on the data (HTML code or HTTP response) provided to the classification algorithm in the training and the testing phases. The

authors used too many features including unnecessary ones such as URL features. URL features were used by the authors because some XSS attacks involve embedding the attack pattern into the URL. However these patterns to be activated they must be passed into the HTML source code and if that web page is secured against XSS then these attacks will not run, thus by just having attack patterns in URL it does not mean that the web page is vulnerable and also all the XSS attacks either based on URL or stored in the web page server, they all get activated inside the HTML source code, so it is important to focus only on the XSS features found in the HTML source code and ignore the URL features which were used in the previous studies.

Training and testing a dataset to develop a robust classifier depends highly on the features extracted and selected from the HTML code which contributes to the accuracy of the classification results (Janecek, 2008). Features extraction depends on what attributes from the data can be used to classify it into classes, and usually these attributes or features have some redundancies or noise which affect the performance of the classifier (Khanna, 2014). The selection of the relevant and most significant features can enhance the accuracy of the classification and lead to a shorter training time which can be done by removing redundant or irrelevant features (noise data).

Thus, building a robust classifier depends highly on how the features (of the data that need to be classified) are extracted and selected, and how can the feature selection process can be optimized so only relevant features are used which will increase the accuracy of identifying at which class a web page (in the dataset) belongs to, XSS or Benign. However, using too many features or irrelevant features in order to detect XSS web pages can unfortunately have bad impact on the results. Therefore, decreasing the performance of the classification algorithm (Khanna, 2014).

To improve the performance of the classification it is very important to invest a good effort on the features selection and apply different techniques such as PCA (Principal component analysis), PSO (Particle swarm optimization) or IG (Information Gain) to choose the best and most significant features that lead to a better and more efficient classification performance. The previous works (Likarish et al. (2009), Nunan

et al. (2012) and Krishnaveni and Sathiyakumari (2013)) had no clear method for features selection, and by applying such techniques, we believe the number of features they used can be reduced to a smaller subset that will lead to same or better results, yet shorter training and testing times.

1.3 Problem Statement

Most of recent XSS classification approaches proposed their ML techniques based on too many features extracted from both URL and web page document. These approaches did not include a feature selection to reduce the number of the used features. Using too many features without feature selection and including unnecessary features lead to an increase of the computation time and complexity of the classifier because of including redundant, irrelevant or unnecessary features causing a decrease in classification performance due to not properly selected by good features selection methods. URL features are considered unnecessary since the environment that is used to run the attack is the HTML source code (web page content), therefore it is important to focus only on the features of XSS based on the HTML document since all the XSS attack types get activated only inside the source code.

1.4 Purpose of Study

This research identifies most significant features of XSS web pages to be used for enhancing classification performance, detection accuracy, decrease false positive and categorize web pages (dataset) into two categories (XSS or Benign) by implementing the features selection via Information Gain (IG) technique.

1.5 Project Objectives

The objectives of this project are listed below:

i. To extract all possible features from the dataset based on the features used in the literature review (Likarish et al. (2009), Nunan et al. (2012) and Krishnaveni and Sathiyakumari (2013)).
ii. To use Information Gain (IG) technique for features selection in order to select the most significant features that lead to a better classification performance and better results.
iii. To implement and compare data classification via ML algorithms Naïve Bayes (NB) and Support Vector Machine (SVM) using the selected features on the training and testing datasets.

1.6 Scope of Study

The scope of this project is as follows:

i. The study focuses on using Information Gain (IG) features selection technique that produce the most significant features which will lead to high percentage of accuracy and better performance from the classification algorithms.
ii. Classifying XSS web pages based on features obtained from HTML code only (web page source code).
iii. The study will use a dataset for the malicious class from XSSed (www.xssed.com) which was used by most of the previous works. For the

benign class, a crawler plugin used to crawl web pages from Google search results where this data is verified to be XSS-free using Vega (a software to scan web page looking for vulnerabilities such as XSS and SQL injection).

iv. The classification process is done using RapidMiner 2016; which is an open source data mining software.

1.7 Organization of Report

The significance of this study underlies on how important it is to improve the feature selection which would lead to a better classification where then we can build better security applications to detect XSS with a small to none false alarm rate. Another significance of this study is to increase the awareness level among researchers and web applications developers on the nature of XSS, how its attacks happen and the threat they can bring to the security of the web applications used by many industries such as ecommerce, educational institutes, banks and governmental agencies which should motivate them to build secure web applications from the scratch and not be only detective but defensive and preventive against these attacks.

1.8 Organization of Report

The rest of this report is organized as follows: Chapter 2 provides a literature of the problem studied and what has been done so far to solve it, Chapter 3 is about the project methodology and a brief on how the data is collected, handled and processed. The design and implementation of the experiments for features extraction and selection is thoroughly explained in Chapter 4. Classification experiments and results are explained in Chapter 5. Finally, conclusion and future work are provided in Chapter 6.

REFERENCES

Acunetix, 2013. DOM-based Cross-Site Scripting (XSS) Explained. Available at: http://www.acunetix.com/blog/articles/dom-xss-explained/ [Accessed: May 2016]

Al Shalabi, L. and Shaaban, Z., 2006, May. Normalization as a preprocessing engine for data mining and the approach of preference matrix. *In 2006 International Conference on Dependability of Computer Systems* (pp. 207-214). IEEE.

Amor, N.B., Benferhat, S. and Elouedi, Z., 2004, March. Naive bayes vs decision trees in intrusion detection systems. *In Proceedings of the 2004 ACM symposium on Applied computing* (pp. 420-424). ACM.

Ankush, S.D., 2014. XSS attack prevention using DOM based filtering API (Doctoral dissertation). *Department of Computer Science and Engineering. National Institute of Technology Rourkela*, Rourkela – 769 008, India.

Brownlee, J., 2014. An Introduction to Feature Selection. Available at: http://machinelearningmastery.com/an-introduction-to-feature-selection/ [Accessed: December 2016]

Brownlee, J., 2016. Naive Bayes for Machine Learning. Available at: http://machinelearningmastery.com/naive-bayes-for-machine-learning/ [Accessed: January 2017]

Cortes, C. and Vapnik, V., 1995. Support-vector networks. Machine learning, 20(3), pp.273-297.

Deepa, G. and Thilagam, P.S., 2016. Securing web applications from injection and logic vulnerabilities: Approaches and challenges. *Information and Software Technology*, 74, pp.160-180.

DuCharme, B. n.d. Data Science Glossary. Available at: http://www.datascienceglossary.org/#feature [Accessed: January 2017]

DuPaul, N., 2013. Cross-Site Scripting (XSS) Tutorial: Learn About XSS Vulnerabilities, Injections and How to Prevent Attacks. Available at: https://www.veracode.com/security/xss [Accessed: May 2016]

Fawcett, T., 2004. ROC graphs: Notes and practical considerations for researchers. Machine learning, 31(1), pp.1-38.

Gerardnico, 2016. Text Mining - term frequency – inverse document frequency (tf-idf). Available at: http://gerardnico.com/wiki/natural_language/tf-idf#tf [Accessed: December 2016].

Gupta, N., 2014. A Study of Existing Cross Site Scripting Detection and Prevention Techniques in Web Applications. *In International Journal Of Engineering And Computer Science* ISSN:2319-7242, Volume 3 Issue 9, Page No. 8445-8450

Hamada, M.H.A., 2012. Client Side Action Against Cross Site Scripting Attacks (Doctoral dissertation, Islamic University–Gaza).

Han, J., Pei, J. and Kamber, M., 2011. *Data mining: concepts and techniques*. Elsevier.

Hydara, I., Sultan, A.B.M., Zulzalil, H. and Admodisastro, N., 2015. Current state of research on cross-site scripting (XSS)–A systematic literature review. *Information and Software Technology*, 58, pp.170-186.

Janecek, A., Gansterer, W.N., Demel, M. and Ecker, G., 2008, September. On the Relationship Between Feature Selection and Classification Accuracy. *In FSDM* (pp. 90-105).

Khanna, V., 2014, Is having a very large number of features in Machine Learning ever a bad thing? Available at: https://www.quora.com/Is-having-a-very-large-number-of-features-in-Machine-Learning-ever-a-bad-thing [Accessed: Dec 2016]

Krishnaveni, S. and Sathiyakumari, K., Efficient Prediction of Cross-Site Scripting Web Pages using Extreme Learning Machine. *International Journal of Computer Science & Engineering Technology (IJCSET)*, 4(11), pp.1395-1400.

Lala, 2015. Guide to Web PenTesting. Available at: http://slides.com/lala/guide-to-web-pentest#/ [Accessed: December 2016]

Likarish, P., Jung, E. and Jo, I., 2009, October. Obfuscated malicious javascript detection using classification techniques. *In MALWARE* (pp. 47-54).

Ma, J., Saul, L.K., Savage, S. and Voelker, G.M., 2009, June. Beyond blacklists: learning to detect malicious web sites from suspicious URLs. *In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1245-1254). ACM.

Manek, A.S., Sumithra, V., Shenoy, P.D., Mohan, M.C., Venugopal, K.R. and Patnaik, L.M., 2014, August. DeMalFier: Detection of Malicious web pages using an effective classifier. *In Data Science & Engineering (ICDSE), 2014 International Conference* (pp. 83-88). IEEE.

MDN, 2015. Mozilla Developer Network: About JavaScript. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript. [Accessed: May 2015]

Nguyen, T. and Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials,* 10(4), pp.56-76.

Nithya, V., Pandian, S.L. and Malarvizhi, C., 2015. A Survey on Detection and Prevention of Cross-Site Scripting Attack. *International Journal of Security and Its Applications*, 9(3), pp.139-151.

Nunan, A.E., Souto, E., dos Santos, E.M. and Feitosa, E., 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. 702-707). IEEE.

Oracle, 2007. Oracle Data Mining, Concepts. 11g Release 1 (11.1). Oracle Corp, 2007. Available at: http://www.comp.dit.ie/btierney/Oracle11gDoc/datamine.111/b28129.pdf [Accessed: January 2017]

Oracle, 2016. Classification: About Classification. Available at: https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/classify.htm#DMCON004 [Accessed: December 2016]

OWASP, 2013. OWASP Top 10 – 2013. Available at: https://www.owasp.org/images/f/f8/OWASP_Top_10_-_2013.pdf. [Accessed in: May 2016].

OWASP, 2015. DOM based XSS. Available at: https://www.owasp.org/index.php/DOM_Based_XSS [Accessed: May 2016]

OWASP, 2016, Cross-site Scripting (XSS). Available at: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS). [Accessed: May 2016]

Raman, P., 2008. JaSPIn: JavaScript based Anomaly Detection of Cross-site scripting attacks (*Doctoral dissertation, CARLETON UNIVERSITY Ottawa*).

Ramaswami, M. and Bhaskaran, R., 2009. A study on feature selection techniques in educational data mining. arXiv preprint arXiv:0912.3924.

Richards, J. (2014). Relevance Of Generalized Linear Models In Oracle Database Mining. [online] Available at: http://www.exeideas.com/2014/11/linear-models-in-oracle-database.html [Accessed January 2017].

Scikit-Learn.Org, 2015. Feature Extraction. Available at: http://scikit-learn.org/stable/modules/feature_extraction.html [Accessed: January 2017]

Systems Sciences, n.d. Machine Learning [image]. Available at: http://systems-sciences.uni-graz.at/etextbook/bigdata/supervised_seg.html [Accessed: January 2017]

Wikibooks.org, 2016. Support Vector Machines - Wikibooks, open books for an open world. Available at: https://en.wikibooks.org/wiki/Support_Vector_Machines [Accessed: January 2017].

Witten, I.H. and Frank, E., 2005. Data Mining: Practical machine learning tools and techniques with Java Implementations (Second Edition). *Morgan Kaufmann Publishers.*