J Inf Process Syst, Vol.13, No.4, pp.1014~1028, August 2017
https://doi.org/10.3745/JIPS.03.0079

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

XSSClassifier: An Efficient XSS Attack Detection Approach Based on Machine Learning Classifier on SNSs

Shailendra Rathore*, Pradip Kumar Sharma*, and Jong Hyuk Park*

Abstract

Social networking services (SNSs) such as Twitter, MySpace, and Facebook have become progressively significant with its billions of users. Still, alongside this increase is an increase in security threats such as cross-site scripting (XSS) threat. Recently, a few approaches have been proposed to detect an XSS attack on SNSs. Due to the certain recent features of SNSs webpages such as JavaScript and AJAX, however, the existing approaches are not efficient in combating XSS attack on SNSs. In this paper, we propose a machine learning-based approach to detecting XSS attack on SNSs. In our approach, the detection of XSS attack is performed based on three features: URLs, webpage, and SNSs. A dataset is prepared by collecting 1,000 SNSs webpages and extracting the features from these webpages. Ten different machine learning classifiers are used on a prepared dataset to classify webpages into two categories: XSS or non-XSS. To validate the efficiency of the proposed approach, we evaluated and compared it with other existing approaches. The evaluation results show that our approach attains better performance in the SNS environment, recording the highest accuracy of 0.972 and lowest false positive rate of 0.87.

Keywords

Cross-Site Scripting Attack Detection, Dataset, JavaScript, Machine Learning Classifier, Social Networking Services

1. Introduction

Nowadays, social networking services (SNSs) are widely used, and the number of their users is increasing rapidly. The huge number of SNSs users and public interest attract hackers; therefore, attacks on SNS applications are more frequent. An attacker can launch different types of attacks on SNSs such as cross-site scripting (XSS), phishing, distributed denial of service (DDOS), and many more [1,2]. An XSS attack is a more dangerous attack, and it has become the favorite choice of attackers for attacking SNSs. Certain features of SNS webpages such as JavaScript and AJAX and frequent communication between users make these pages more susceptible to XSS attack [3]. An XSS attack is typically caused by the inappropriate filtering policies on input text, enabling an attacker to inject XSS script into webpages. When visiting the webpage containing the injected script, a user runs the script on his/her browser and becomes a victim of XSS attack. For instance, if a user profile on SNSs is infected by XSS, the profiles of the user's friends and other connected user profiles can be easily infected by this attack [4]. A typical

Manuscript received March 2, 2017; first revision April 26, 2017; accepted May 30, 2017.
Corresponding Author: Jong Hyuk Park (jhpark1@seoultech.ac.kr)
* Dept. of Computer Science and Engineering, Seoul National University of Science & Technology (SeoulTech), Seoul, Korea ({rathoreshailendra, pradip, jhpark1}@seoultech.ac.kr)

XSS attack method is shown in Fig. 1.

The XSS attack can be categorized into three major types (persistent XSS, non-persistent XSS, and DOM-based XSS [5]) and they are described as follows:

- *Persistent attack* usually happens when input from the user is kept in the target server, e.g., database, comment field, visitor log, and message forum. In the SNS environment, this attack consists of multiple steps. Initially, the adversary stores the attack payload in the vulnerable SNSs server using the webpage form. Subsequently, when the victim user accesses the webpage containing the attack payload, this payload is executed on the browser of the SNS user, thereby introducing a persistent XSS attack.
- *Non-persistent attack* happens when a web application returns the user input immediately in the form of pop-up box, search result, error message, or any other reflected message containing some or all of the user input without permanently storing the data provided by the user. In this type of attack, initially, the adversary lures the victim by providing a URL with harmful obfuscated code. Once the victim uses that URL, the harmful obfuscated code enclosed in the URL is run on the victim's browser, causing a reflected XSS attack.
- In a *DOM-based XSS attack*, DOM stands for Document Object Model, which is used to represent web documents in a browser in structure format. It enables active scripts (JavaScript) for document reference components such as a session cookie or a form field. A DOM-based XSS attack happens when the active content of script, such as a JavaScript function, is altered by a particularly created request so that an attacker can control a DOM element.

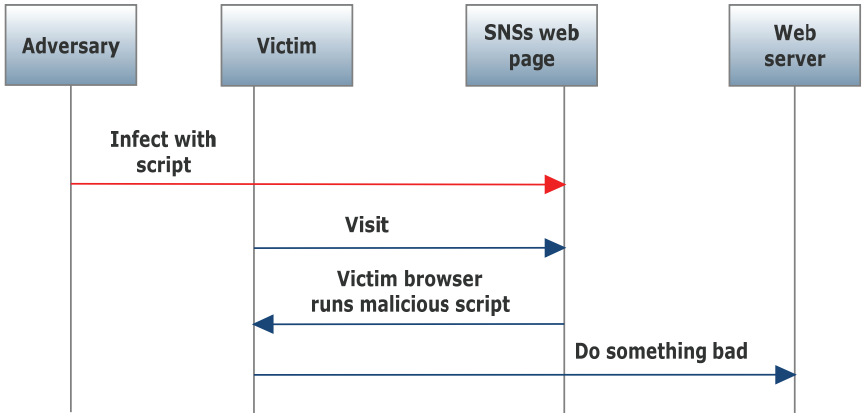


Fig. 1. A typical XSS attack method.

XSS vulnerability is usually exploited in the form of XSS worm on famous SNSs such as Facebook and Twitter. As a malicious payload generally written in JavaScript, an XSS worm breaks the security of the browser to propagate among the users of a website. The adversary can use this worm for malicious intent or to steal personal information such as credit card number or passwords. In the SNS environment, XSS worms can affect many features of SNS websites, such as chat systems and profiles, when these features are not implemented properly or without security awareness. There are various XSS worms that had already impacted many popular SNSs [6].

- *Renren Worm*: Renren is one of the largest Chinese SNS platforms infected by the Renren worm in 2009. This worm propagated in multiple steps and affected millions of Renren users. Initially, the attacker posts a malicious flash movie on his profile. Subsequently, when a victim user views the attacker-infected profile and exploits the flash vulnerability of the flash movie, such flash vulnerability injects malicious JavaScript on the victim's side. Furthermore, the malicious injected script replicates itself on the wall of the victim. When the victim's friends view the infected profile, they are also infected by the worm replication [7].
- *SamyWorm*: MySpace is an SNS platform used by millions of users to keep in touch and share their interests with each other. It was one of the first victims of the XSS worm named Samy (name of the worm creator). This worm infected MySpace users in two stages. In the first stage, Samy, the worm creator, injected an attack payload into his profile. Subsequently, in the second step, any user viewing the Samy-infected profile got infected, and the attack payload spread to the accessing user's profile. Thus, the infected accessing user became a source of further infection [8].
- *Boonana Worm*: Boonana is another Java applet worm released in October 2010. Infection by this worm involved multiple segments. Initially, the attacker posts a malicious Java applet on his profile. Subsequently, when the victim user views the attacker's profile and exploits a Java vulnerability of the malicious Java applet, the Java vulnerability injects malicious JavaScript on the victim's side. The worm replicates itself on the victim user's wall using the stolen cookie. The Boonana worm multiplies over SNSs as more users access the malicious applet [9].
- *SpaceFlash Worm*: Released in 2006, it is a JavaScript XSS worm that exploited a flash vulnerability of the MySpace platform. In the first stage of worm infection, the victim user visits the "About Me" page of the attacker's profile (containing a malicious Flash applet) and exploits the flash vulnerability of the "About Me" page. The flash vulnerability injects malicious JavaScript on the victim's side. Furthermore, the worm replicates itself on the victim's "About Me" page by sending an AJAX request to the server. The SpaceFlash worm multiplies over SNSs as more victims visit the malicious "About Me" page [10].

Recently, XSS attacks have become a major security concern in the area of SNS security. According to the report from Hackgon [10], 12.75% of the total web attacks were XSS attacks, and nearly 70% of all vulnerabilities on the web were categorized as XSS-related vulnerabilities. Thus, many researchers have proposed ideas on detecting XSS attack on the web. Likarish et al. [11] proposed an approach based on machine learning technique to identify malicious JavaScript on the web. Note, however, that it can be used to detect obfuscated malicious JavaScript only; it does not cover all possibilities of XSS attack. Nunan et al. [12] used an automatic classification approach for XSS attack detection. This approach can be applied to SNSs to identify an XSS attack. The experimental analysis of this approach was restricted to the classifier and calculation of its factors. Thus, there may be other classifiers that can be used to classify the XSS attack and attain better results. The existing techniques for XSS attack detection on the web cannot be efficiently applied to SNSs because some specific features of SNSs—such as more trust between two nearer nodes, lower average distance between two nodes, and higher frequency of communication between users—make detecting an XSS attack on SNSs using the existing techniques difficult; hence the need for a method that detects an XSS attack on SNSs by considering their specific features. In this paper, we propose a machine learning-based approach that uses SNSs' specific features to detect an XSS attack on SNSs. The following are the major contributions of this paper:

- We analyze the recent characteristics of SNSs webpage and suggest a novel collection of features that are responsible for XSS attack on SNSs.
- A novel dataset is constructed by collecting 1,000 SNS webpages and extracting suggested features from the webpages.
- This paper offers a novel XSS attack detection approach that relies on machine learning classifiers to classify XSS-infected webpages from non-infected ones.
- We provide a thorough evaluation of the proposed approach to prove its validity and effectiveness in the SNS environment.

The rest of this paper is organized as follows: Section 2 discusses various existing approaches that have been recently proposed for XSS detection in the web application; Section 3 describes the proposed approach and its four steps of feature identification, collection of webpages, feature extraction and training dataset construction, and machine learning classification; In Section 4, we experimentally evaluate our proposed approach and compare it with the existing approaches; Finally, Section 5 presents the conclusions.

2. Related Work

Nowadays, many researchers are working on XSS detection on SNSs, and a significant amount of research work has been done in the past. Likarish et al. [11] proposed a machine learning approach to detect the obfuscated malicious JavaScript in web applications. This approach identified 65 features and trained 4 classifiers for classification. The analysis of this approach suggests that we can use the machine learning technique for XSS attack detection on SNSs. The research of Sun et al. [13] presented a client-side approach to detecting the JavaScript worm via a Firefox plug-in. This approach relies on string comparison to detect the propagation of XSS worm. Note, however, that it defends only a particular client and does not provide protection for the whole web application. In addition, it is vulnerable to simple polymorphic worms wherein the payload signature changes actively throughout the propagation of the worm. Nunan et al. [12] applied the automatic classification of webpages for XSS attack detection. This type of classification is based on the features of URL and webpage-document. The experiment results show the usefulness of the suggested features in XSS attack detection. Livshits and Cui [14] proposed an automatic detection tool to detect JavaScript worms known as Spectator. This tool adopts a means of propagation chain across the social network for searching the worm. It disallows uploading from the infected nodes to control further spread. It is also based on the distributed tagging and tainting technique, which identifies the worm spreading behaviors. As a limitation of this technique, it only identifies the JavaScript worm when a large number of users have been affected.

Cao et al. [6] proposed PathCutter, a detection tool that can detect content-sniffing XSS, DOM-based XSS, and traditional XSS attack. They accomplished their objective using two fundamental steps: request authentication and view separation. PathCutter splits the web application into diverse views, and then separates these views on the browser side. The authors presented both proxy-side and server-side deployment of PathCutter. They implemented the tool on two open-source SNSs: Elgg and WordPress. They also tested their tool on five real-world worms: Yamanner, SpaceFlash, Renren,

MySpace Samy, and Boonana worm. PathCutter does not prevent form content ex-filtration XSS attacks and cookie stealing. Another weakness of this approach is that it consumes longer rendering time on the client side, For instance, to respond to a post containing 45 comments, a system with PathCutter takes 30% longer rendering time compared to the system without PathCutter implementation [6]. Ter Louw and Venkatakrishnan [15] developed a tool for protecting end-users from an XSS attack known as BLUPRINT. It offers powerful affirmation that the web browser will not run malicious scripts in a web application with low integrity output. The tool applies a technique that reduces the trust placed in web browsers when understanding untrusted content. It has been used with numerous real web applications to protect them from XSS worm with less overhead.

Xu et al. [16] introduced a correlation-based system that can detect and mitigate the risk of active worm propagation on SNSs. It creates a surveillance network by allocating "decoy friends" to a certain group of users, and the "decoy friends" will passively monitor the network for suspicious activities. The main weakness of this approach is the difficulty of adding "decoy friends" to the user friend list without the consent of the user. This approach protects against active worms on SNSs such Koobface but does not provide protection against passive worms such as XSS worms. Another disadvantage of this approach is that it requires a certain number of infected users before detection.

Recently, Ahmed and Ali [17] proposed an approach to detecting XSS attack, which uses genetic algorithm for generating a test dataset. They tested their approach on MySQL and PHP-based web applications. The research of Wang and Zhou [18] presented a novel mechanism based on a new technique such as HTML5 to detect XSS attacks. The performance of the proposed mechanism was evaluated with XSSer, a popular tool developed by OWASP.

3. Proposed Approach

In this section, we discuss our proposed approach to detecting XSS attacks on SNSs. The functional block diagram of our approach is shown in Fig. 2. Our approach depends on the machine learning classifiers to classify the webpages into two categories: XSS or non-XSS. It mainly involves four steps: feature identification, collection of webpages, feature extraction and training dataset construction, and machine learning classification. The description of each step is described below.

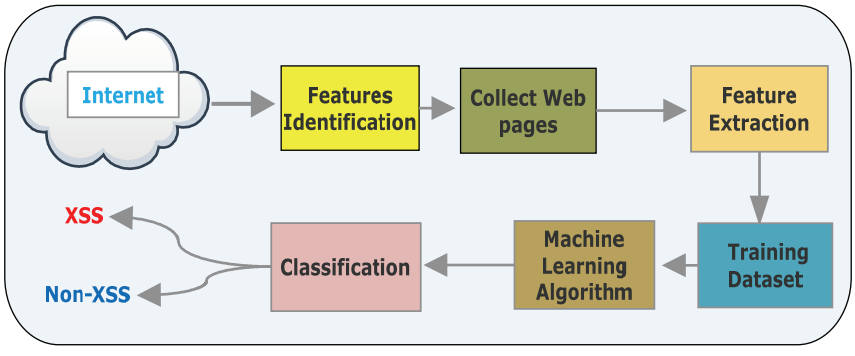


Fig. 2. Functional block diagram of the proposed approach.

3.1 Feature Identification

Feature identification is an important step in the machine learning classification process. As the main classification ingredient, features correctly separate XSS-infected samples from the non-infected ones. In our proposed approach, we define the list of features. These features are extracted from URLs, webpage content, and SNSs to create a feature vector, which is given as an input to the classifier. A classification of XSS features is shown in Fig. 3 and is described below.

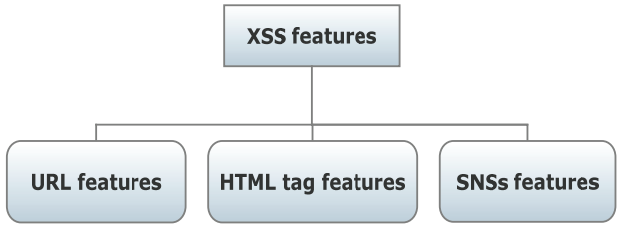


Fig. 3. Classification of XSS features.

1) URL features**:** The uniform resource locator (URL) can be utilized to conceal malicious XSS codes that play a vital role in non-persistent XSS. An attacker can create some decoration on the URLs to lure the user into clicking them, and detection of decorated URLs is more difficult. This type of decoration can also be used as evidence to detect an XSS attack [19]. We studied an XSS attack on a webpage by using decorated URLs and examined some features that separate XSS-infected samples from the non-infected ones. The list of significant URL features is shown in Table 1.

Table 1. List of URL features

No.	Feature	Type
1	Total count of long URLs	Integer
2	The maximum size of URLs	Integer
3	The maximum occurrence of domains found in the URLs	Integer
4	Total count of URLs with maximum number of obfuscated characters	Integer

2) HTML tag features: HTML tags are an essential component for create a webpage. These tags and their attributes (such as value) can be inserted and deleted dynamically by scripts. Therefore, certain HTML tags can be used by an attacker to inject the XSS code scripts from outside. These HTML tags consist of <link>, <object>, <form>, <script>, <embed>, <ilayer>, <layer>, <style>, <applet>, <meta>, <img>, <iframe>, and many more. For instance, the XSS worm "Samy" infected MySpace webpages by injecting a huge quantity of XSS payload in the tag of the webpages [20]. On the other hand, JavaScript language is used in a webpage for embedding tasks, but an attacker can misuse some methods on the embedded XSS payload such as exec(), fromCharCode(), eval, alert(), getElementsByTagName(), write(), unscape(), and escape() [21]. Code obfuscation is also used to hide the XSS code. Therefore, we have also included some other features like external script, harmful keyword, size and number of scripts, maximum number of

encoded characters in JavaScript, having trouble with event handler, etc. In our proposed approach, we selected 18 HTML tag features as shown in Table 2.

3) SNS Features: SNSs add some additional features to enhance functionality compared to normal networks; for example, in an SNS environment, there is greater trust between two nearer nodes compared to those in a normal network, the average distance between any two nodes is much smaller than that in normal networks, and the propagation speed of worms is faster. These additional features increase the chances of XSS attack on SNSs, and the attacker can easily infect a single node with an XSS worm. This XSS worm propagates much faster on SNSs and infects the whole network in a very short time [16]. We selected some other features that are responsible for XSS attack on SNSs as shown in Table 3.

Table 2. List of HTML tag features

No.	Feature	Type
1	Total count of harmful keywords	Integer
2	Total count of Iframe	Integer
3	Total count of external Iframe source	Integer
4	Total count of external links	Integer
5	Having trouble with event handlers	Boolean
6	Presence of malicious java script method	Boolean
7	Total count of DOM-modified keywords	Integer
8	Total count of String-decoding keywords.	Integer
9	Total count of AJAX keywords, and Other Keywords	Integer
10	The maximum size of the script	Integer
11	Total count of maximum size of the script	Integer
12	Total count of encoded links	Integer
13	Maximum count of encoded characters in JavaScript	Integer
14	Maximum size of HTML tags	Integer
15	Total count of maximum size HTML tags	Integer
16	Maximum count of JavaScript strings in HTML tags	Integer
17	The existence of obfuscation code	Boolean
18	Total count of external Script source	Integer

Table 3. List of SNSs features

No.	Feature	Type
1	Total count of suspicious HTML tags in SNSs webpage	Integer
2	Total count of suspicious URLs in SNSs webpage	Integer
3	Number of suspicious JavaScript strings in SNSs traffic in unit time (frequency)	Integer

3.2 Collecting Webpages

In this Step, a database is created by collecting malicious and benign webpages from various trusted Internet sources such as XSSed [22], Alexa [23], and Elgg [24]. The database consists of three kinds of webpages: benign webpages, malicious webpages containing obfuscated code, and SNSs webpages

infected by XSS worm. The benign webpages are collected from the Alexa database, whereas malicious webpages are collected from the XSSed database. Note, however, that SNSs webpages are not available in XSSed and Alexa database. Therefore, we used Elgg 1.12.4, which is an open social network engine with many available plug-INS. The SNS webpages are collected by modifying the Elgg source codebase and subsequently inserting the corresponding modifications. Fig. 4 shows the different categories of the collected webpages and their Internet sources.

3.3 Features Extraction and Training Dataset Construction

This step is responsible for extracting and recording XSS features (identified in step 1) from each of the collected webpages. For features extraction, we use various tools as described in Table 4. Each webpage is manually labeled as XSS or non-XSS based on the extracted features, and a training dataset is constructed. We constructed a training dataset containing 1,000 webpages (400 malicious and 600 benign) and their extracted features.

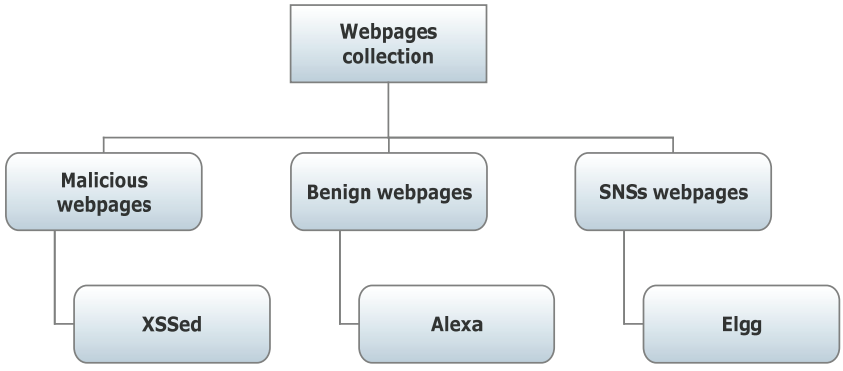


Fig. 4. Collection of webpages from various Internet sources.

Table 4. List of tools for webpage feature extraction

Tool	Description
Website crawler	This tool is used to extract the required information from websites.
Jericho HTML parser	It is a java library that provides a facility for analyzing and manipulating a specific portion of an HTML document, and it consists of tags and keywords.
JavaScript engine	A JavaScript engine is a virtual machine that interprets and executes JavaScript.
Website pattern extractor	This tool is used to extract data from a webpage according to a Regular Expression or a specified predefined pattern.
Link / Header crawler	This tool is used to determine a series of all URLs associated with a webpage. It produces a summary that offers a list of title tags and header responses for all webpages associated with the webpage being tested. This tool can also help identify irrelevant content, unusual redirects, and possible broken links.
Keyword density analysis	This tool takes a URL as input, examines it, and returns a list of the most commonly used phrases or keywords on the webpage. It is used to identify the frequency of keywords and list of harmful keywords.

3.4 Machine Learning Classification

The goal of this step is to categorize the webpages into XSS or Non-XSS. In other words, it is used to determine whether a webpage is infected with XSS or is legitimate. In order to achieve this objective, the training dataset (constructed in an earlier step) is supplied to the machine learning classifier. The classifier generates a predictive model that is further used to detect XSS-infected webpages. In our proposed approach, we used Weka [25] as a tool to generate predictive model.

4. Experiment and Evaluation

4.1 Performance Measurement Methodology

To measure the performance of our proposed approach, we applied the 10-fold cross-validation technique [26] for the classification experiments. It increases the evaluation reliability of the classifier. The aim of this technique is to forecast and estimate how accurate a classification model will work in practice. Initially, it partitions the original whole dataset into ten equally-sized folds, from which nine folds are operated together as a training dataset and one fold is used as a test dataset. Subsequently, the machine learning classifier performed evaluation by applying both datasets. This entire procedure of partitions and evaluation is repeated ten times. Each fold is used exactly once as the test dataset in each evaluation. The final results consist of 10 evaluation results on average [27,28]. The classification results are described using a confusing matrix as shown in Table 5.

Table 5. Confusing matrix

Actual	**Predicted**	
	XSS	**Non-XSS**
XSS	T_p	F_n
Non-XSS	F_p	T_n

where T_p (True positive) shows the number of XSS-infected webpages classified as XSS, F_p (False positive) shows the number of non-XSS-infected webpages classified as XSS, F_n (False negative) shows the number of XSS-infected webpages classified as non-XSS, and T_n (True negative) shows the number of non-XSS-infected webpages classified as non-XSS-infected.

We use the "Confusion Matrix" as a metric to evaluate classifier performance based on our proposed features. This matrix enables the assessment of result of false negatives and false positives as shown in Table 5. We compute the following standard measures based on this matrix:

$$Recall = T_p/(T_p + F_n) \tag{1}$$

$$Precision = T_p/(T_p + F_p) \tag{2}$$

$$F - measure = (2 * Recall * Precision)/(Recall + Precision) \tag{3}$$

$$Accuracy = (T_p + T_n)/(T_p + F_p + F_n + T_n) \tag{4}$$

$$False\ positive\ rate = F_p/(F_p + T_n) \tag{5}$$

4.2 Performance Evaluation of the Proposed Approach

We performed an evaluation to find out how accurately our proposed approach determines whether a webpage is infected with XSS or is legitimate. For the evaluation, we used Weka as a tool for creating and measuring the predictive model. The Weka is a data mining tool that evaluates a classification model using numerous machine learning algorithms. It can process clustering, regression, and classification. The evaluation procedure of our approach involves two phases. In the first phase of evaluation, a set of training datasets (constructed in subsection 2.3) without SNS features was supplied to two classifiers known as Decorate and ADTree. Subsequently, we evaluated the performance of both classifiers. In the second phase of evaluation, we repeated the same process as that in the first phase by supplying a set of training datasets with SNS features to both classifiers, and performance was evaluated for both classifiers. Table 6 details the performance of both classifiers with and without SNS features. From Table 6, it can be easily seen that both classifiers attained good performance results in terms of standard measures as described in the earlier subsection. These results show the effectiveness of the proposed approach in the XSS classification of webpages. Furthermore, better performance was attained with the use of SNS features compared to the performance achieved without using them. In general, the experiment results demonstrate that our approach to XSS detection is more effective in the SNS environment.

We also evaluated which machine learning classifier attains better performance in detecting XSS-infected webpages in the SNS environment. For the evaluation, a set of training datasets with SNS features was supplied to the top 10 classifiers (as shown in Table 7). Consequently, we evaluated and compared the performance results for all classifiers to find which classifier attains good performance. We compared the results for all classifiers in terms of standard measures as described in the earlier subsection. It can be easily observed from Table 7 that each classifier had reasonable performance in detecting XSS-infected webpages. Every classifier had recall, precision of more than 0.950, F-measure of over 0.952, accuracy of more than 0.947, and false positive rate of below 2.24. These results validate the effectiveness of our proposed approach. The overall result of all classifiers shows that the tree classifiers (RandomForest and ADTree) obtained the best performance compared to meta-classifiers (Decorate and AdaBoost.M1). RandomForest is the best classifier with recall of 0.971, precision of 0.977, F-measure of 0.974, accuracy of 0.972, and lowest false positive rate of 0.87.

Table 6. Performance of ADTree and Decorate classifier by using both with SNSs and without SNSs features

	ADTree		**Decorate**	
	Without SNSs features	**With SNSs features**	**Without SNSs features**	**With SNSs features**
Recall	0.956	0.972	0.949	0.968
Precision	0.954	0.970	0.946	0.965
F-measure	0.950	0.971	0.947	0.966
Accuracy	0.952	0.971	0.946	0.966
False positive rate	0.131	0.092	0.152	0.112

Table 7. Performance of 10 classifiers for XSS detection

	Recall	Precision	F-measure	Accuracy	False positive rate
RandomForest	0.971	0.977	0.974	0.972	0.087
ADTree	0.972	0.970	0.971	0.971	0.092
RandomSubSpace	0.969	0.972	0.970	0.970	0.092
Decorate	0.968	0.965	0.966	0.966	0.112
AdaBoost.M1	0.965	0.964	0.964	0.966	0.123
JRip	0.962	0.960	0.961	0.964	0.149
NaïveBayes	0.964	0.966	0.965	0.963	0.151
Support Vector Machine	0.958	0.955	0.956	0.958	0.187
Logistic Regression	0.955	0.950	0.952	0.956	0.146
k-Nearest Neighbors	0.950	0.954	0.952	0.947	0.224

Finally, in order to validate our classification results, the obtained results were compared with the research results of Wang et al. [29]. In this research, similarity and difference-based features were adopted for XSS attack detection on SNSs. A training dataset was constructed by extracting these features of the webpages collected from Dmoz, XSSed, and Weibo database. The performance results were obtained by evaluating the constructed training dataset on two classifiers known as AdaBoost.M1 and ADTree. Figs. 5 and 6 illustrate the comparison of the research results of Wang et al. [29] and the results achieved by our proposed approach for AdaBoost.M1 and ADTree classifiers. The results of the comparison show that our approach obtains excellent result compared to the research results of Wang et al. [29] in terms of precision, recall, and F-measure due to the following reasons: (i) in our approach, we used multiple and selected features of URLs, HTML tag, and SNSs from diverse Internet sources that provide better prediction rate of XSS webpages in the SNS environment, such as XSSed, Alexa, and Elgg; (ii) The accumulated dataset in our approach is not biased since it contains a reasonable number of both malicious and benign webpages; and (iii) Our accumulated dataset is recent, containing recently infected XSS webpages.

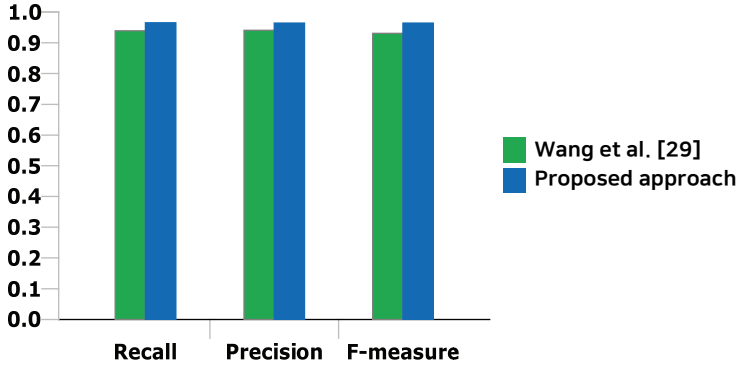


Fig. 5. Performance comparison with the proposed approach by using AdaBoost.M1 classifier.

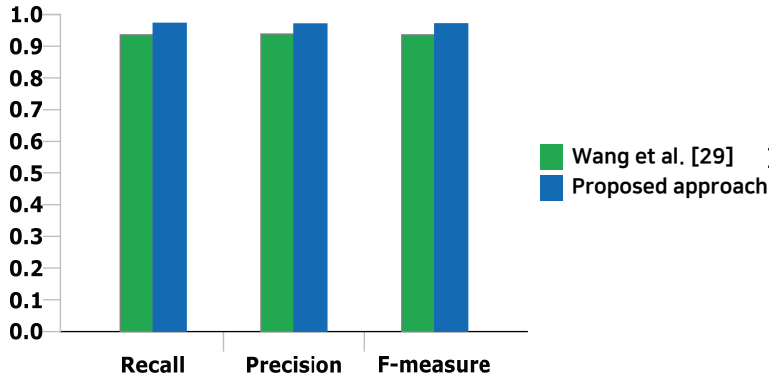


Fig. 6. Performance comparison with the proposed approach by using ADTree classifier.

5. Conclusion

This study investigated the XSS attack on SNSs and provided two major contributions in the area of SNSs security. First, we provided an analysis of the recent characteristics of SNS webpages and recommended a novel set of XSS features on SNSs. Second, we proposed a novel machine learning-based approach for detecting XSS attacks on SNSs. We evaluated the proposed approach by using ten classifiers on the training dataset cooperating with and without SNS features. The evaluation results show that our approach obtained outstanding performance with the lowest false positive rate of 0.87 and highest accuracy of 0.972. Our findings suggest that the recommended set of features can be used to detect XSS attack on SNSs, and that the proposed approach can be incorporated with SNSs for detecting XSS webpages in real time.

In the future, our approach can be enhanced in two directions. The first direction involves enhancement in the proposed feature set, and the second one provides the application of advanced machine learning algorithms, such as deep learning and extreme learning machine for the classification of XSS webpages.

References

[1] D. H. Lee, "Personalizing information using users' online social networks: a case study of CiteULike," *Journal of Information Processing Systems*, vol. 11, no. 1, pp. 1-21, 2015

[2] J. Kim, D. H. Yao, H. Jang, and K. Jeong, "WebSHArk 1.0: a benchmark collection for malicious web shell detection," *Journal of Information Processing Systems*, vol. 11, no. 2, pp. 229-238, 2015

[3] Y. Zhang, X. Wang, Q. Luo, and Q. Liu, "Cross-site scripting attacks in social network APIs," in *Proceedings of Workshop on WEB 2.0 Security and Privacy (W2SP 2013)*, San Francisco, CA, 2013.

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

[5] M. K Gupta, M. C. Govil, and G. Singh, "Static analysis approaches to detect SQL injection and cross site scripting vulnerabilities in web applications: a survey," in *Proceedings of the Recent Advances and Innovations in Engineering (ICRAIE)*, Jaipur, India, 2014, pp. 1-5

[6] Y. Cao, V. Yegneswaran, P. Possas, and Y. Chen, "Pathcutter: severing the self-propagation path of XSS JavaScript Worms in social web networks," in *Proceedings of the Network and Distributed System Security Symposium (NDSS'12)*, San Diego, CA, 2012, pp. 1-14

[7] L. Constantin, "New Chinese social networking worm discovered," 2009 [Online]. Available: http://news.softpedia.com/news/New-Chinese-Social-Networking-Worm-Discovered-120021.shtml.

[8] Technical explanation of The MySpace Worm [Online]. Available: https://samy.pl/popular/tech.html.

[9] G. Cluley, "Cross-platform Boonana Trojan targets Facebook users," 2010 [Online]. Available: https://nakedsecurity.sophos.com/2010/10/28/cross-platform-worm-targets-facebook-users/.

[10] Hackagon, "XSS attack," 2016 [Online]. Available: http://hackagon.com/xss-attack/.

[11] P. Likarish, E. Jung, and I. Jo, "Obfuscated malicious JavaScript detection using classification techniques," in *Proceedings of the 4th International Conference on Malicious and Unwanted Software (MALWARE),* Montreal, Canada, 2009, pp. 47-54.

[12] A. E. Nunan, E. Souto, E. M. dos Santos, and E. Feitosa, "Automatic classification of cross-site scripting in webpages using document-based and URL-based features," in *Proceedings of the IEEE Symposium on Computers and Communications (ISCC),* Cappadocia, Turkey, 2012, pp. 000702-000707.

[13] F. Sun, L. Xu, and Z. Su, "Client-side detection of XSS worms by monitoring payload propagation," in *Proceedings of the 14th European Symposium on Research in Computer Security*, Saint-Malo, France, 2009, pp. 539-554.

[14] V. B. Livshits and W. Cui, "Spectator: detection and containment of JavaScript Worms," in *Proceedings of the USENIX Annual Technical Conference*, Boston, MA, 2008, pp. 335-348.

[15] M. Ter Louw and V. N. Venkatakrishnan, "Blueprint: robust prevention of cross-site scripting attacks for existing browsers," in *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, Oakland, CA, 2009, pp. 331-346.

[16] W. Xu, F. Zhang, and S. Zhu, "Toward worm detection in online social networks," in *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC'10)*, Austin, TX, 2010, pp. 11-20.

[17] M. A. Ahmed, and F. Ali, "Multiple-path testing for cross site scripting using genetic algorithms," *Journal of Systems Architecture*, vol. 64, pp. 50-62, 2016

[18] C. H. Wang and Y. S. Zhou, "A new cross-site scripting detection mechanism integrated with HTML5 and CORS properties by using browser extensions," in *Proceedings of the 2016 International Computer Symposium (ICS)*, Chiayi, Taiwan, 2016, pp. 264-269.

[19] Common Attack Pattern Enumeration and Classification, "CAPEC-72: URL encoding," 2017 [Online]. Available: https://capec.mitre.org/data/definitions/72.html.

[20] Y. S. Hwang, J. B. Kwon, J. C. Moon, and S. J. Cho, "Classifying malicious webpages by using an adaptive support vector machine," *Journal of Information Processing Systems*, vol. 9, no. 3, pp. 395-404, 2013.

[21] R. Wang, X. Jia, Q. Li, and D. Zhang, "Improved N-gram approach for cross-site scripting detection in online social network," in *Proceedings of the Science and Information Conference (SAI)*, London, UK, 2015, pp. 1206-1212.

[22] XSS attacks information [Online]. Available: http://www.xssed.com/.

[23] Alexa, "The top 500 sites on the web," 2017 [Online]. Available: http://www.alexa.com/topsites.

[24] Elgg Foundation, "Introducing a powerful open source social networking engine," [Online]. Available: https://elgg.org/.

[25] Weka 3: data mining software in Java [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.

[26] K. M. Prabusankarlal, P. Thirumoorthy, and R. Manavalan, "Assessment of combined textural and morphological features for diagnosis of breast masses in ultrasound," *Human-centric Computing and Information Sciences*, vol. 5, no. 1, pp. 1-17, 2015.

[27] C. Chantrapornchai and P. Nusawat, "Two machine learning models for mobile phone battery discharge rate prediction based on usage patterns," *Journal of Information Processing Systems*, vol. 12, no. 3, pp. 436-454, 2016.

[28] J. H. Choi, H. S. Shin, and A. Nasridinov, "A comparative study on data mining classification techniques for military applications," *Journal of Convergence*, vol. 7, pp. 1-7, 2016.

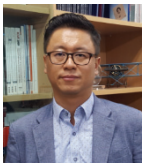
[29] R. Wang, X. Jia, Q. Li, and S. Zhang, "Machine learning based cross-site scripting detection in online social network," in *Proceedings of the 2014 IEEE International Conference on High Performance Computing and Communications (HPSS), 2014 IEEE 6th International Symposium on Cyberspace Safety and Security (CSS), and 2014 IEEE 11th International Conference on Embedded Software and Systems (ICESS),* Paris, France, 2014, pp. 823-826.

Shailendra Rathore

He is a Ph.D. student in the Department of Computer Science at Seoul National University of Science and Technology (SeoulTech.), Seoul, Korea. Currently, he is working in Ubiquitous Computing Security (UCS) Lab under the supervision of Prof. Jong Hyuk Park. His broadly research interest includes Information and Cyber Security, SNS, Digital Forensic, IoT. Previous to joining PhD at SeoulTech, he has worked as an Executive -Technology at Crompton Greaves Global R & D, Mumbai, India from June, 2013 to July, 2014. He received his M.E. in Information Security from Thapar University, Patiala, India and B.Tech. in Computer Engineering from Rajasthan Technical University, Kota, Rajasthan, India

Pradip Kumar Sharma http://orcid.org/0000-0001-6620-9083

He is a Ph.D. scholar at the Seoul National University of Science and Technology. He works in the Ubiquitous Computing & Security Research Group under the supervision of Prof. Jong Hyuk Park. Prior to beginning the PhD program, he worked as a software engineer at MAQ Software, India. He worked on a variety of projects, proficient in building large-scale complex data warehouses, OLAP models and reporting solutions that meet business objectives and align IT with business. He received his dual Master's degree in Computer Science from the Thapar University, in 2014 and the Tezpur Univerity, in 2012, India. His current research interests are focused on the areas of ubiquitous computing and security, cloud computing, SDN, SNS, and IoT. He is also reviewer of Journal of Supercomputing (JoS).



James J. (Jong Hyuk) Park http://orcid.org/0000-0003-1831-0309

He received Ph.D. degrees in Graduate School of Information Security from Korea University, Korea and Graduate School of Human Sciences from Waseda University, Japan. From December 2002 to July 2007, Dr. Park had been a research scientist of R&D Institute, Hanwha S&C Co., Ltd., Korea. From September 2007 to August 2009, he had been a professor at the Department of Computer Science and Engineering, Kyungnam University, Korea. He is now a professor at the Department of Computer Science and Engineering and Department of Interdisciplinary Bio IT Materials, Seoul

National University of Science and Technology (SeoulTech), Korea. Dr. Park has published about 200 research papers in international journals and conferences. He has been serving as chairs, program committee, or organizing committee chair for many international conferences and workshops. He is a founding steering chair of some international conferences—MUE, FutureTech, CSA, UCAWSN, etc. He is editor-in-chief of Human-centric Computing and Information Sciences (HCIS) by Springer, The Journal of Information Processing Systems (JIPS) by KIPS, and Journal of Convergence (JoC) by KIPS CSWRG. He is Associate Editor / Editor of 14 international journals including 8 journals indexed by SCI(E). In addition, he has been serving as a Guest Editor for international journals by some publishers: Springer, Elsevier, Wiley, Oxford University press, Hindawi, Emerald, Inderscience. His research interests include security and digital forensics, human-centric ubiquitous computing, context awareness, multimedia services, etc. He got the best paper awards from ISA-08 and ITCS-11 conferences and the outstanding leadership awards from IEEE HPCC-09, ICA3PP-10, IEE ISPA-11, and PDCAT-11. Furthermore, he got the outstanding research awards from the SeoulTech in 2014. Dr. Park's research interests include human-centric ubiquitous computing, vehicular cloud computing, information security, digital forensics, secure communications, multimedia computing, etc. He is a member of the IEEE, IEEE Computer Society, KIPS, and KMMS.