JURNAL INFOTEL
Informatics - Telecommunication - Electronics

Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

Cross-site Scripting Attack Detection Using Machine Learning With Hybrid Features

Dimaz Arno Prasetio[1*], Kusrini[2], M. Rudyanto Arief[3]
[1,2,3]Universitas AMIKOM Yogyakarta
Jl. Ring Road Utara, Condong Catur, Sleman, Yogyakarta, 55281, Indonesia
Corresponding email: dimaz.1234@students.amikom.ac.id

Received 21 January 2021, Revised 11 February 2021, Accepted 15 February 2021

Abstract — This study aims to measure the classification accuracy of Cross-Site Scripting **(**XSS) attacks by using a combination of two methods of determining feature characteristics, namely using linguistic computation and feature selection. XSS attacks have a certain pattern in their character arrangement, this can be studied by learners using n-gram modeling, but in certain cases, XSS characteristics can contain a certain meta and synthetic this can be learned using feature selection modeling. From this research result, hybrid feature modeling gives good accuracy with an accuracy value of 99.87%. It is better than previous studies in which the average is still below 99%. This study also tries to analyze the false positive rate considering that the false positive rate in attack detection is very influential for the convenience of the information security team. With the modeling proposed, the false positive rate is very small, namely 0.039%.

Keywords – machine learning, hybrid features, logistic regression

I. INTRODUCTION

The internet has become a vital part of our lifestyle, with numerous uses in banking, shopping, entertainment, resource sharing, news, and social networking. Web applications are becoming more common in everyday lives. We literally depend on these applications to accomplish tasks, and they are integral to our day-to-day activities. We communicate with online applications in a dynamic manner [1] when we interact with our email, conduct banking transactions, visit social networking sites, etc. Web applications have a dynamic nature because they can determine how a website reacts to a user's input. In many websites, users' input on the site is not correctly validated, which compromises the integrity of the site. The attack occurs when a user visits a malicious website, which is specifically designed to exploit vulnerabilities within the client and server-side, after which the attack is launched [2]. XSS has been acknowledged as one of the top ten web application security vulnerabilities by the Open Web Application Security Project (OWASP) [3]. Types of XSS attacks [4] are as follows: stored XSS, reflected XSS, and DOM-based XSS.

XSS stored attack usually happens when the personal data of the user is transferred to the intended destination. This attack is conducted in a web environment through multiple stages. Initially, the attacker deploys the attack payload into the vulnerable server by using webpages vulnerability. Finally, a target user receives the attacks after visiting the webpage with the attached XSS attack payload. This popular attack has occurred in the case of social media MySpace [5] where this XSS works like a worm virus where everyone who visits the hacker's profile page in myspace will automatically run the attack and again attack other users who visit their profile page.

Reflected XSS attack happens when a web application returns immediately without storing the data that is provided by the user. This kind of attack happens when the adversary lures the victim by a web page with malicious code written in it. If the victim visits that URL, the embedded code in the URL will run and cause a reflected Cross Site Scripting (XSS) attack. Research on XSS attacks with the reflected type [6] [7] [8] [9] [10] is quite a lot compared to research on XSS stored types.

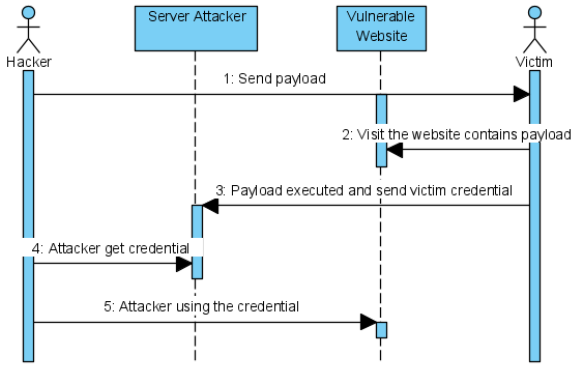

Fig. 1. Flow XSS attack

In a DOM-based XSS attack, the web document is represented in XML format with DOM. This causes an active script code refers to a specific document. A DOM-based XSS attack happens when the active content of JavaScript is changed by a special request.

To defend XSS, it's common to use two approaches: static analysis and dynamic analysis [11]. Static analysis analyzes source codes without running the applications, which, in turn, require very specialized information regarding security. Runtime analysis analyzes the execution records of the program to detect vulnerabilities. Dynamic analysis analyzes incoming attacks that are usually in the form of javascript injection against a website input. Website input can be in the form of fields, cookies, headers, and file uploads.

In research [12] using various ensemble methods to classify XSS attacks, including the bagging method, AdaBoost, bagging classifier with SVM, gradient boosting, these methods obtained an average accuracy of 98%. A study [13] that uses external sources to help classify XSS attacks, namely by using threat intelligence from bad IP list providers and bad domains, using the bagging method, and majority voting an average accuracy of 98% is obtained. The use of decision tree as a classification algorithm to detect XSS attacks [14] did not get good enough results with an accuracy value of only 86%, still less than the two studies above. Research [15] carried out XSS detection using a static code analysis approach where the PHP program code was converted into opcode, which then used the Bi-LSTM algorithm for classification, the results of the study obtained an accuracy of 98%. Research on the use of the n-gram language model has been carried out [16] to detect bugs and it has pretty good results, namely, 98% in this study replacing previous bug detection techniques that use the model rules.

In this research, the detection of XSS attacks uses a dynamic analysis approach, where the attack dataset was into two characteristic forms, namely through the process of linguistic computation and feature selection.

II. RESEARCH METHODS

The purpose of this research is to analyze the accuracy of text classification into two classes, namely attack or non-attack with a model that has never been done before. This study uses a dataset from Kaggle [17] and the addition of datasets from various sources on Github, the number of datasets was 16361 data consisting of XSS attacks and non-XSS attacks.

A. Proposed Model

The following is the modeling proposed in this study.

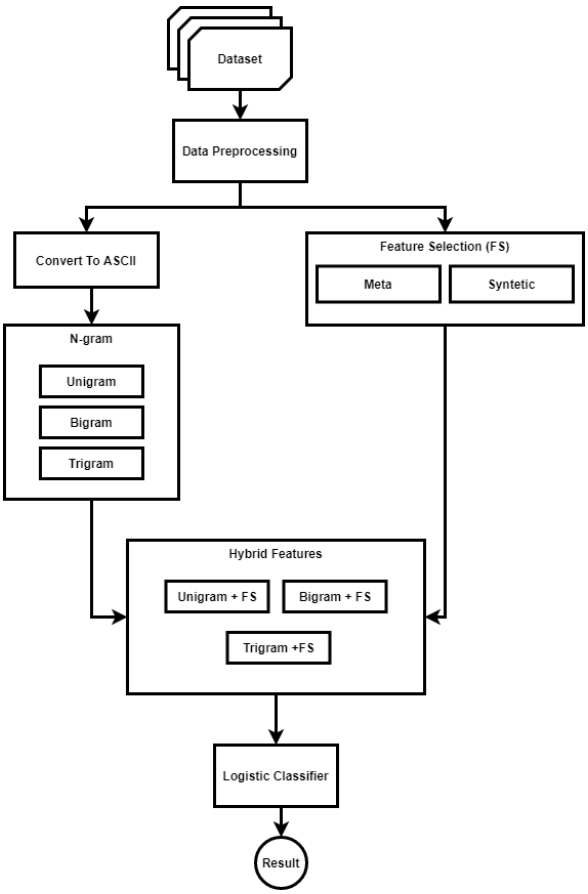

Fig. 2. Proposed Model

First, the dataset will be preprocessed. In this stage, data normalization will be carried out such as removing duplicate data and deleting blank lines, and deleting text that is less than 20 characters long. After that the dataset will be processed to take its characteristics, this modeling is divided into two, the first is computational linguistic, here the classification characteristics are based on the number of appearances of character divisions based on n-grams, in this phase before the n-gram process is carried out the data will first be converted into ASCII form is then performed computational linguistics. Then the second branch is to take the meta-based and synthetic characteristics of the dataset. More details will be explained in the next session.

B. Linguistic Computation

Computational linguistics may be seen as the subfield of computer science dealing with utilizing computational methods to read, interpret, and generate human language [18]. Computational linguistic systems may have several uses, such as helping human-human contact, such as with computer translations; aiding human-machine communication, such as through conversational agents; or supporting both humans and computers through processing and learning from the immense quantity of human language material that is now accessible online.

The n-gram language model consists of words and sentences, where each sentence is constructed of a sequence of words. A dictionary has all terms in a language, and each word is identified by a single letter. The language model uses Markov chains to predict any conceivable sentence in a language. The likelihood of a sentence in a language is determined by randomly producing a term in the list. The likelihood of a term in a sentence is dependent on the previous n - 1 terms. Provided a sentence $s = w_1 w_2 w_3 \cdots w_m$, the likelihood of its interpretation can be calculated as:

$$P(s) = \prod_{i=1}^{m} P(w_i | h_{i-1}) \quad (1)$$

Where $h_i = w_i - n \cdots w_i$ is the historical past. In the n-gram model, the next word is determined solely by the prevocalization of the previous n words. For example, given a sequence length of three, the probability that the sequence would be $s = w_1 w_2 w_3$ is:

$$P(s) = P(w_1)P(w_2|w_1)P(w_3|w_2 w_1) \quad (2)$$

In this paper, we build a predictive model of what is most likely to be used given different contexts. By learning the probability distribution of each token sequence, we further calculate which token sequences are likely to lead to XSS attacks.

In the proposed model, we want to examine whether XSS attacks have special linguistic patterns that can be used to help classify XSS attacks or plain text. By studying the structure of XSS attacks, which generally use symbols such as "<", ">", "/", ";" then even 1 character symbol if interpreted in a sentence structure, the symbol can represent one word. Therefore, each character will be converted first into ASCII, which will be computed later.

This research will compute n-gram, which consists of unigram, bigram, and trigram. The results of the computational accuracy will be validated with a machine learning algorithm, namely logistic regression which is known for its fast processing and good accuracy.

C. Feature Selection

Supervised feature selection techniques are applicable to the problem of classification or regression. The goal of these techniques is to select a subset of features that are able to differentiate examples from different classes (regression). Feature relevance is usually assessed via regression or its related variable such as the class label. The training phase highly depends on the selected features. Classifiers or regression models are trained on a subset of features selected by supervised feature selection. Methods for feature selection do not depend on the learning algorithm, it may be independently carried out as a stand-alone procedure (embedded methods). This trained classifier or regression model can correctly label the unseen samples in the test set. For supervised classification problems, we focus on classification problems and use label and supervision information interchangeably.

In this study, feature selection will be divided into two types, namely meta and synthetic, more details are in Table 1.

Table 1. List of Feature Selection

Feature	Type	Description
Number of urls [11]	Synthetic	Number of HTTP or HTTPS
Number of js event [11] [12]	Synthetic	The number of JS events that appear, JS events are javascript attributes and functions that can trigger javascript execution on a web page
Length [12]	Meta	Number of characters
Number of special character [12]	Meta	The number of special characters that appear, special characters are characters other than alphabet and numeric
Number of non-printable character	Meta	Number of characters that cannot be printed such as \u009, \b, \a
Minimum ASCII	Synthetic	ASCII minimum value
Maximum ASCII	Synthetic	ASCII maximum value

All of the above features are included in the training process, except the number of URL feature is not used because based on the results of experiments that have been carried out it can reduce the value of classification accuracy.

This is the list of js events used in this study.

Table 2. List of JS event

onabort	onmousemove
onafterprint	onmouseover
onanimationend	onmouseout
onanimationiteration	onmouseup
onanimationstart	onmousewheel
onbeforeprint	onoffline
onbeforeunload	ononline
onblur	onopen
oncanplay	onpagehide
oncanplaythrough	onpageshow
onchange	onpaste
onclick	onpause
oncontextmenu	onplay
oncopy	onplaying
oncut	onpopstate
ondblclick	onprogress
ondrag	onratechange
ondragend	onresize
ondragenter	onreset
ondragleave	onscroll
ondragover	onsearch
ondragstart	onseeked
ondrop	onseeking
ondurationchange	onselect
onended	onshow
onerror	onstalled
onfocus	onstorage
onfocusin	onsubmit
onfocusout	onsuspend
onfullscreenchange	ontimeupdate
onfullscreenerror	ontoggle
onhashchange	ontouchcancel
oninput	ontouchend
oninvalid	ontouchmove
onkeydown	ontouchstart
onkeypress	ontransitionend
onkeyup	onunload
onload	onvolumechange
onloadeddata	onwaiting
onloadedmetadata	onwheel
onloadstart	onmouseenter
onmessage	onmouseleave
onmousedown	-

D. Hybrid Features

In this study, a combination of linguistic computation and feature selection was carried out, several scenarios that will be analyzed for accuracy are as follows.

Table 3. List of Hybrid Feature

Features ID	Description
F1	Unigram (1-gram)
F2	Bigram (2-gram)
F3	Trigram (3-gram)
F4	Feature Selection (meta, synthetic)
F5	F1+F4
F6	F2+F4
F7	F3+F4

F1, F2, F3 are research using linguistic computation, F4 is done by classifying based on feature selection with meta and synthetic types, features of meta type are length and number of special characters, while features with synthetic types are number of js event, number of non-printable character, minimum ascii, and maximum ascii. F5, F6, F7 are combination of linguistic computation and feature selection. Each of these features will be measured for accuracy with a logistic regression algorithm.

III. RESULTS

This session provides the results of each of the models previously described.

A. Linguistic Computation Result

The linguistic computation test was carried out with three scenarios, namely unigram, bigram, and trigram, before that, the computation was done first, the conversion of text into ascii form was then carried out by testing the accuracy with logistic regression.

Table 4. List of Linguistic Computation Results

Feature	Accuracy	F1 score	Precision	Recall
F1	99.34%	99.46%	99.54%	99.37%
F2	99.72%	99.77%	99.82%	99.72%
F3	99.83%	99.86%	100%	99.72%

It can be seen that the linguistic computation method has fairly good results with the best results obtained on the trigram with an accuracy of 99.87% with a confusion matrix as follows.

Table 5. Confusion Matrix of Linguistic Computation Results

	XSS	Non-XSS
XSS	3980	11
Non-XSS	0	2542

B. Feature Selection Result

Testing with feature selection is first carried out based on the predefined features. Based on the results of testing with logistic regression, the following results were obtained.

Table 6. Feature Selection Results

Feature	Accuracy	F1 score	Precision	Recall
F4	77.39%	83.25%	76.03%	91.98%

If you look at the results of linguistic computation, the results of feature selection have lower accuracy, which is 77.39% with a confusion matrix as follows.

Table 7. Confusion Matrix of Feature Selection Results

	XSS	Non-XSS
XSS	3671	320
Non-XSS	1157	1385

C. Hybrid Features

This hybrid features test is done by combining linguistic computation with feature selection, the results are as follows.

Table 8. Hybrid Feature Results

Feature	Accuracy	F1 score	Precision	Recall
F5	99.32%	99.44%	99.47%	99.42%
F6	99.75%	99.79%	99.87%	99.72%
F7	99.87%	99.98%	99.97%	99.82%

The following is the confusion matrix result.

Table 9. Confusion Matrix of Hybrid Feature Results

	XSS	Non-XSS
XSS	3984	7
Non-XSS	1	2541

The results obtained are there is an increase in accuracy compared to the two methods above if it runs separately. The best accuracy results are in the F7 feature model, which is a combination of Trigram and Feature Selection, which is 99.87% with false positives using equation (3) of 0.039%.

$$FP\ rate = FP / (FP + TN) \quad (3)$$

which, FP is False Positive, FPrate is False positive rate, and TN is True Negative.

IV. DISCUSSION

We divide the scenario in this study into three main parts, namely examining the accuracy of the classification using linguistic computation, the second using feature selection, and finally combining the two. The combination of the two or hybrid features is the model we propose. By looking at the results of accuracy alone, it can be seen that hybrid features can increase accuracy in this XSS attack classification process.

Based on the results of the accuracy of the linguistic computation method, the greatest accuracy of the trigram model is 99.83%, then the accuracy of the feature selection method is less accurate, which is 77.39%. Then, by combining the two methods, the results obtained were better, namely 99.87% with a combination of trigram and feature selection.

V. CONCLUSSION

The use of hybrid features to classify XSS attacks can increase accuracy with the best results obtained by an accuracy of 99.87% and also false positive XSS detection is very crucial, with this modeling false positives can be reduced to 0.039%. In the future we will try various comparisons with other classification algorithms and more datasets.

REFERENCES

[1] S. Ninawe and R. Wajgi, "Detection of DOM-Based XSS Attack on Web Application," *Lecture Notes on Data Engineering and Communications Technologies*, vol. 33, pp. 633–641, 2020.

[2] H. Choi, S. Hong, S. Cho, and Y. G. Kim, "HXD: Hybrid XSS detection by using a headless browser," in *Proceedings of the 2017 4th International Conference on Computer Applications and Information Processing Technology*, Kuta Bali, 2017, pp. 1-4.

[3] "OWASP Top Ten Web Application Security Risks | OWASP." https://owasp.org/www-project-top-ten/ (accessed Jan. 21, 2021).

[4] 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 *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, May 2014, pp. 1–5.

[5] Y. Cao and V. Yegneswaran, "Pathcutter: Severing the self-propagation path of xss javascript worms in social web networks," *Symp. Netw. …*, 2012.

[6] J. Dahse and T. Holz, "Static detection of second-order vulnerabilities in web applications," in *Proceedings of the 23rd USENIX Security Symposium*, 2014.

[7] A. Kiezun, P. J. Guo, K. Jayaraman, and M. D. Ernst, "Automatic creation of SQL injection and cross-site scripting attacks," in *Proceedings - International Conference on Software Engineering*, 2009.

[8] B. Stock, S. Pfistner, B. Kaiser, S. Lekies, and M. Johns, "From facepalm to brain bender: Exploring client-side cross-site scripting," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2015.

[9] B. Stock, M. Johns, M. Steffens, and M. Backes, "How the web tangled itself: Uncovering the history of client-side web (in)security," in *Proceedings of the 26th USENIX Security Symposium*, 2017.

[10] S. Lekies, B. Stock, M. Wentzel, and M. Johns, "The unexpected dangers of dynamic Javascript," in *Proceedings of the 24th USENIX Security Symposium*, 2015.

[11] X. Guo, S. Jin, and Y. Zhang, "XSS Vulnerability Detection Using Optimized Attack Vector Repertory," in *Proceedings - 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2015, pp. 29–36.

[12] P. Nagarjun and S. S. Ahamad, "Ensemble Methods to Detect XSS Attacks," 2020. Accessed: Jan. 13, 2021. [Online]. Available: www.ijacsa.thesai.org.

[13] Y. Zhou and P. Wang, "An ensemble learning approach for XSS attack detection with domain knowledge and threat intelligence," *Comput. Secur.*, 2019, doi: 10.1016/j.cose.2018.12.016.

[14] Ö. Kasım, "Malicious XSS Code Detection with Decision Tree," *J. Polytech.*, Feb. 2019.

[15] C. Li, Y. Wang, C. Miao, and C. Huang, "Cross-site scripting guardian: A static XSS detector based on data stream input-output association mining," *Appl. Sci.*, 2020.

[16] S. Wang, D. Chollak, D. Movshovitz-Attias, and L. Tan, "Bugram: bug detection with n-gram language models," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, Aug. 2016, pp. 708–719.

[17] S. S. Hussain Shah, "Cross site scripting XSS dataset for Deep learning | Kaggle," 2020. https://www.kaggle.com/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning (accessed Jan. 20, 2021).

[18] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science (80-.).*, vol. 349, no. 6245, pp. 261–266, Jul. 2015.