Improved N-gram Approach for Cross-site Scripting Detection in Online Social Network

Rui Wang, Xiaoqi Jia, Qinlei Li, Daojuan Zhang
State Key Laboratory of Information Security,
Institute of Information Engineering,
Chinese Academy of Sciences, China
Email: {wangrui,jiaxiaoqi,liqinlei,zhangdaojuan}@iie.ac.cn

***Abstract*—Nowadays Online Social Networks (OSNs) have become a popular web service in the world. With the development of mobile networks, OSNs provide users with online communication platform. However, the OSNs' openness leads to so much exposure that it brings many new security threats, such as cross-site scripting (XSS) attacks. In this paper, we present a novel approach using classifiers and the improved n-gram model to do the XSS detection in OSN. Firstly, we identify a group of features from webpages and use them to generate classifiers for XSS detection. Secondly, we present an improved n-gram model (a model derived from n-gram model) built from the features to classify webpages. Thirdly, we propose an approach based on the combination of classifiers and the improved n-gram model to detect XSS in OSN. Finally, a method is proposed to simulate XSS worm spread in OSN to get more accurate experiment data. Our experiment results demonstrate that our approach is effective in OSN's XSS detection.**

Keywords*—*Online Social Networks Security; Cross-site Scripting Attacks Detection; N-gram Model

I. Introduction

As a widely-used web application, more and more online social network (OSN) users are suffering from cybercrimes. The number of OSNs' users, which has increased a lot in recent years, is still in rapid growth. The large number of users and the potential economic interests has drawn much attention from hackers, and as a result, OSN platforms are attacked at an increasing frequency. The common types of cybercrimes are malwares, viruses, scams, thefts and frauds [1].

Cross-site scripting (XSS) is a common type of attacks which has become one of the main threats in OSN [14],[3]. Some characteristics of OSN make it more easily attacked by XSS: 1) The use of some new web development techniques, such as AJAX and JavaScript, leads to more attacks with good user interactive experience. 2) OSN users can build their own relationships in the network, and the frequent communication between nodes makes the network more vulnerable [13]. For example, if a node in the OSN is attacked by XSS, its neighbors will become more easily infected than the users in general networks do. Therefore, we need an effective approach to detect XSS attacks in order to protect our OSNs and to prevent the propagation of XSS at the same time.

Cross-site scripting (XSS) is caused by improper input text filtering policies, which enable attackers to inject client-side script into webpages. The script will run in the victim's browser under certain circumstances, forming a XSS attack. There are three main types of XSS: persistent XSS, non-persistent XSS and DOM-based XSS [2],[15]. **1) Persistent XSS attack:** In a persistent XSS attack, the attacker sends the data with malicious script through webpage forms to the server, and the data is saved on the server. After that, when the webpages containing the malicious code is returned to another user, the script provided by the attacker will run in the user's browser, which leads to a persistent XSS attack. **2) Non-persistent XSS attack:** In a non-persistent XSS attack, the attacker tempts the user with URLs containing malicious script. When the user clicks on the URL, the script contained in the URL will be executed in the user's browser, and a non-persistent XSS attack is formed. **3) DOM-based XSS:** DOM-based XSS occurs in the content processing stages performed by the client. The client-side script will modify the page content dynamically, which does not rely on uploading data to the server. If the data in the DOM is not checked strictly, DOM-based XSS attacks will occur. As shown in Fig.1, user A is attacked by non-persistent XSS and user B is attacked by persistent XSS.

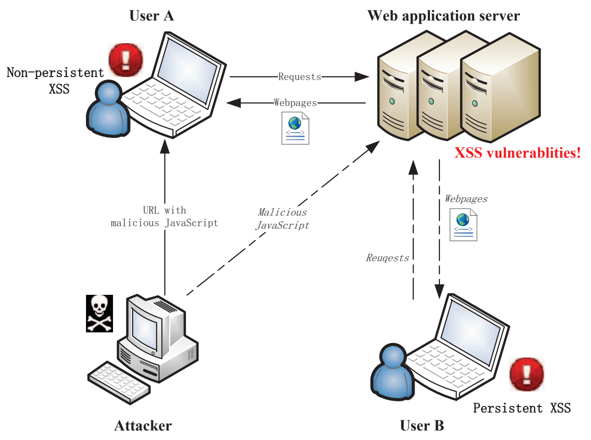

Fig. 1: User A is attacked by non-persistent XSS and user B is attacked by persistent XSS

In this paper, we propose an approach based on classifiers and the improved n-gram model to detect XSS in OSN, and our contributions are as follows:

- We leverage new OSN webpage features by analyzing the OSN webpage characteristics, and build our own classifiers to detect XSS in OSN.
- Drawing inspiration from the n-gram model, we set up a novel model called improved n-gram model which can detect malicious webpages based on feature vectors instead of sequences.
- We implement our approach using the combination of classifiers and the improved n-gram model. The results of the experiments demonstrate the effectiveness of our approach.
- In our experiments, we use some real-world data collected from webpages to simulate XSS worms in OSN to prove the correctness of our approach.

The rest of the paper is organized as follows. Section 2 describes how n-gram model can be used to detect malwares. Section 3 shows the ideas of XSS detection in OSN based on classifiers and improved n-gram model. Section 4 is two kinds of implementations of our approach and Section 5 shows the performances and evaluation for our implementations. Related work is show in Section 6, and finally we talk about some future work and draw some conclusions in Section 7.

II. Applying the n-gram to malware detection

N-gram model is a type of probabilistic language model for predicting the next item in a sequence, and it is similar with n-1 order Markov model. N-gram model is widely used in many fields, such as probability, communication theory, computational linguistics, etc. However, applying n-gram model to the research of information security is still under exploration, such as an n-gram model based method to detect malwares in systems [20].

In n-gram model, we first need to obtain a sequence from a running program in the system, and the sequence of system calls is a good choice. Suppose that the system call sequence of program A is (a1, a2, a3, a4, a5, a6), the 3-gram model for program A will be the set of triples { <a1, a2, a3>, <a2, a3, a4>, <a3, a4, a5>, <a4, a5, a6>}.

With a group of programs containing both malwares and benign programs, 3-gram model based method can be used to detect malwares. Suppose that the set of triples for malwares denotes M, and the set of triples for benign programs denotes B. A library is built to represent the characteristics of malwares, which contains the triples that appearing in some of the malware models but not in any of the models for the benign programs (that is M-B). When the 3-gram model of a program contains more than k instances of 3-grams in the library, this program is considered malicious.

Although n-gram model has some limits (e.g.large parameter space and serious data sparseness), it is still in use due to its simplicity, especially for 2-gram and 3-gram models.

III. Ideas on detection

Classifier is an effective method of malware detection in web applications, while n-gram models can be useful in detecting malwares hidden in systems. After consideration, we propose a novel approach with classifiers and improved n-gram model to detect XSS in OSN.

A. Classifier

We use classifiers to do the classifications in machine learning. The goal of our classification is to detect malicious pages in OSN.

1) Feature extraction: Feature extraction is a crucial step in the process of classification. The feature vector shows the characteristics of the instance (webpage) and is the basis of classification. In our approach, we identify two types of features, *difference-based features* and *similarity-based features*.

a) Difference based features: OSN has its uniqueness compared with other web applications. The propagation of XSS in OSN is quite different from that in other networks. The most obvious difference is that XSS in an OSN network propagates much faster than that in other networks [10], which is the result of OSN's characteristics, such as its topology's higher concentration and shorter shortest path lengths. For example, in 2003, the virus called "Blaster" infected 336,000 computers in 20 hours, while the XSS worm "Samy" infected 1,000,000 users in the same period of time [11],[12].

We come up with 3 correlated features: the spread speed of suspicious JavaScript strings, the spread speed of suspicious HTML tags and the spread speed of suspicious URLs.

In our approach, we take the frequency of the suspicious contents' appearance in OSN traffic as the values of the spread speeds mentioned above. We keep track of the numbers of suspicious contents' appearance in unit time respectively and estimate the spread speed of the suspicious content, in order to judge whether it is XSS malicious code.

b) Similarity-based features: As a web application, OSNs have a lot in common with general networks. In the analysis, we come up with 3 similarities: 1) Both OSN and general network consist of webpages, which contain HTML tags, JavaScript code and URLs. 2) When infected with XSS malicious code, the source code of the webpages will somehow get modified. 3) Both OSN and general networks use some new techniques, such as AJAX. We extract a group of features based on the similarities, which is shown in Table I.

Keyword features. Scripts have some keywords (key functions), which appears in benign and malicious script at different frequencies [8]. Such difference can be used by our classifier to judge whether a webpage is malicious or not. Take *eval* as an example. It is a keyword in XSS worm "Samy", but it appears much less frequently in benign webpages. On the other hand, some keywords with similar functions, such as string decoding functions *unescape* and *fromCharCode*, may not appear in the same malicious webpage. In the feature vectors, the total frequency of several keywords has more influence on the classification result than that of a single keyword. So, In order to reduce the number of features and enhance the effectiveness, we divide the keywords into several groups based on their functions, and take the frequency of each group as a feature.

JavaScript features. In order to bypass the general security detection systems, attackers often use obfuscation to hide their

TABLE I: The description of similarity-based features

Similarity based features		
Keyword features	1	the number of DOM-modified keywords
	2	the number of String-decoding keywords
	3	the number of string-execution keywords
	4	the number of AJAX keywords
	5	the number of other keywords
JavaScript features	6	the number of concatenation of strings
	7	the max length of strings
	8	the number of long string
	9	the max percent of encoded characters
URL features	10	the max length of URLs
	11	the number of long URLs
	12	the max percent of encoded characters in URLs
HTML tag features	13	the max length of HTML tags
	14	the number of long HTML tags
	15	the max number of JavaScript strings in HTML tags

XSS malicious code. An easy and widely used technique is encoding. The encoded script, which changes a lot in appearance, becomes less readable and difficult to get the meaning of the strings. The percent of encoded strings in a webpage is a key feature in judging whether the webpage is malicious or not.

URL features. URLs, which can be used to hide XSS malicious code, are crucial in non-persistent XSS. In order to protect such malicious URLs and tempt the users to click on them, attackers will make some "decoration" on the URLs, and the "decorated" URLs are more difficult to be detected. The extent of this kind of "decoration" can also be evidence in detecting XSS attack.

HTML tag features. HTML tags are the basis of a webpage structure. They can be dynamically inserted and deleted by scripts. Their attributes can also be modified by scripts (e.g. *value*). And even worse, some scripts are able to run automatically in users' browsers (e.g. *src*). So, HTML tags become an ideal place to hide malicious script. When written in a good form, the amount of JavaScript code contained in HTML tags is not likely to be very large, which is not the case for XSS infected webpages. For example, a large amount of JavaScript code will appear in the <div> tag of the webpages infected with XSS worm "Samy" [9]. Whether there are suspicious HTML tags or not becomes crucial evidence in detecting malicious pages.

2) Classification algorithm: There are many researches using Naive Bayes or SVM as their classification algorithms, which have some limitations: 1) The Naive Bayes algorithm is based on the Naive Bayes model [16], so it requires the selected features to be independent, which is not the case for many situations. 2) In the SVM algorithm, the performance of classification relies on the kernel function [17], but it's quite difficult to choose a proper kernel function for the

1. < 1, a_1, a_2>
   a_1: The max length of strings,
   a_2: The number of long strings
2. < 2, b_1, b_2>
   b_1: The max length of URLs,
   b_2: The number of long URLs

Fig. 2: Two instances of a improved n-gram model

classification. 3) It costs the SVM algorithm a lot of time to do the training.

After a lot of researches and experiments, we choose alternating decision tree (ADTree) as our classification algorithm in order to avoid the limitations of Naive Bayes and SVM.

An ADTree [18] is a machine learning method for classification which generalizes decision trees and has connections to boosting. An ADTree is made up of decision nodes and prediction nodes. Decision nodes specify a predicate condition and prediction nodes contain a single number. In ADTrees, root and leave nodes have to be prediction nodes. When an instance is classified, it goes through all paths made up of the decision nodes with true values and sums all the traversed prediction nodes. ADTree takes advantages of decision trees so that they can even deal with samples with partial feature values. In addition, it still can be used when the features are closely correlated, which is exactly what we are faced with. Thus, we choose ADTree as our classifier algorithm.

B. *improved n-gram model*

Unlike sequence-based n-gram model, the improved n-gram model we propose uses vectors or a set of data whose items are not in order but are correlated with each other. In the improved n-gram model, we put the correlated items in the feature vectors into groups according to their correlations, use each group of items to build a new tuple, and finally add the tuples to the model. For example, we have extracted a feature vector (a1, b1, a2, b2, a3, b3) from a sample(e.g. a1 is the max length of strings, a2 is the number of long strings, b1 is the max length of URLs, and b2 is the number of long URLs). We assume that a1 and a2, b1 and b2 in the vector are correlated. So the two groups containing correlated items are the group of a1 and a2 and the group of b1 and b2. The new tuples we make from the groups are <a1, a2> and <b1, b2>, and the final improved n-gram model for this sample is <a1, a2>, <b1, b2>.

Another great difference between improved n-gram model and n-gram model is that the instances in improved n-gram model are not equivalent. So when an improved n-gram model is used in our detection, we cannot match different "n-grams" directly. Take our feature vectors as an example. The improved n-gram model generated from the feature vectors is {<a1, a2>,

<b1, b2>}. <a1, a2> and <b1, b2> should never be matched with each other for the meaning of them are quite different. In order to solve the problem of matching tuples, we add IDs into the tuples to distinguish different "n-grams", and only the tuples with same ID can be matched with each other. Fig 2 shows two instances of the improved n-gram model, and they cannot match with each other due to our rules.

Furthermore, it is hard to determine the value of k in n-gram model, and a fixed value will lower its extensibility. In our improved n-gram model, we use two libraries built for malware and benign programs respectively to solve the problem. When being detected, a program will be considered to be malicious if the number of tuples in its improved n-gram model matches more cases in the library for malware than in the library for benign programs.

IV. XSS DETECTION IN OSN

This section presents two implementations of our XSS detecting approach in OSN using classifiers and the improved n-gram model.

A. Implementation I

In this implementation, the classifier is the main method, and the improved n-gram model works as the auxiliary method.

For classification, especially binary classification, the classifier is a simple and effective solution. However, sometimes we need more accurate results and just using the classifier is quite difficult to improve the detection performance. In order to improve the false positive rate and false negative rate of the classifier, we use the improved n-gram model to do the second-time detection. Fig 3 shows the process of the implementation.

Classifier A and improved n-gram model in Fig 3 are built from training data in the training phase. The test data is obtained from websites for the experiments. The process is shown as follows:

1) Detection with classifier. Firstly, we use the classifier to detect malicious code in the sample. It takes the webpages as input and we denote the output of the classification process as *result1*.

2) Detection with improved n-gram model. In order to reduce the false negative rate, we consider using improved n-gram model to reclassify the samples that have been classified as benign webpages (*result1*=benign) by the classifier. The model takes the "benign" webpages from the classifier as input and generates its result based on the number of detected malicious characteristics in the webpage.

B. Implementation II

In this implementation, the classifier and improved n-gram model are in equivalent position. As both of the models can be used to detect malicious code, we build three units for our detection box, which produces its detection results based on all the classification results of its units.

The detection box in Fig 4 consists of three units: 1) Classifier A. It will classify the sample according to the features and classification algorithm we identified in Section 3. 2) Improved

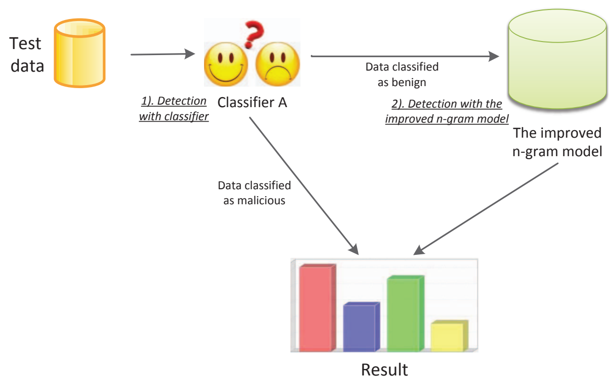
Test data
1). Detection with classifier
Classifier A
Data classified as benign
2). Detection with the improved n-gram model
The improved n-gram model
Data classified as malicious
Result

Fig. 3: Implementation I of our approach

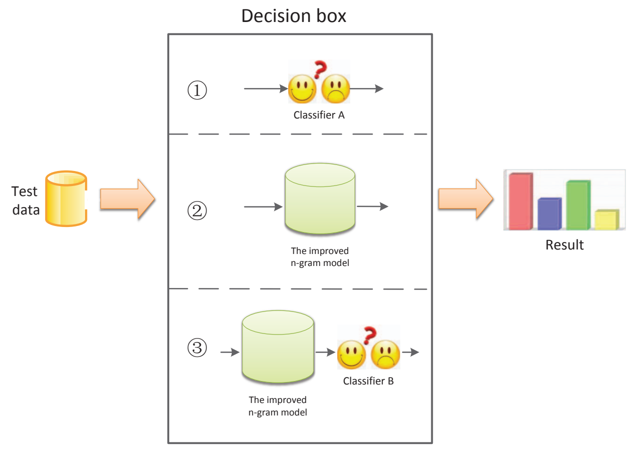

Fig. 4: Implementation II of our approach

n-gram model. In this unit, we firstly build the improved n-gram model for the sample. Then we calculate the numbers of matched instances in malicious library and benign library respectively, and finally classify the sample as malicious or not. 3) The combination of improved n-gram model and classifier B. When a sample is detected by this unit, its improved n-gram model will be built. Then, the unit will calculate the difference between the numbers of matched instances in the malicious library and the benign library, and add the result to the feature vector, forming a new feature vector with one more value. Classifier B is a new classifier other than Classifier A. It takes the new feature vector containing the result from n-gram model as input and generates its own test result. All the classifiers and models used in the three units are obtained from the same training data.

The detection box is used to determine the class of the samples. When a sample enters the detection box, it is examined by each of the units and the final result of classification depends on all the results generated by the three units. A sample is classified as a malicious webpage only if there are at least two units classifying it as a malicious webpage. Otherwise, it is classified as a benign one.

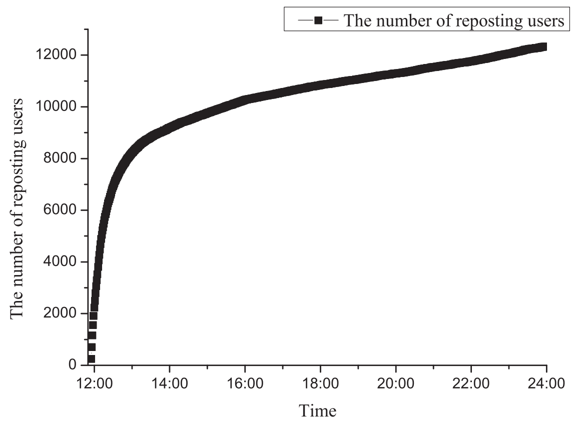

Fig. 5: A hot post propagation in weibo

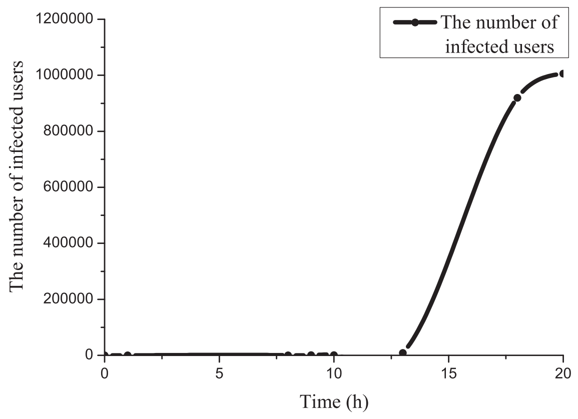

Fig. 6: Samy's spread in MySpace

V. Experiments and evaluation

This section presents the experiments and their results.

A. OSN's XSS worm simulation

It will be more accurate if we build a real online social network and replicate the XSS worms' outbreak, but it is time-consuming and not necessary. Thus, we come up with a method to simulate the propagation of XSS worms, which is based on real-world data and works really well for us.

The main idea of our method is to simulate the propagation of XSS worms using the propagation data of a hot post's reposting. Fig 5 is a hot post's reposting curve, which is based on the data collected from weibo.com, and Fig 6 is "Samy"'s propagation curve in MySpace [9]. It can be found that though not exactly in the same shape, the two curves have a lot in common: 1) They can both infect more users and propagate further than general posts. This is because that they both take advantages of the relationships between OSN users and spread along the topology of the OSN. 2) Both of them spread quickly in the OSN, which is proved by the comparison of the two curves' estimated growth rates and steep increasing trends.

After the work mentioned above, we decide to use a hot post's reposting in real-world OSN to simulate the outbreak of XSS worms.

B. Datasets

1) Data Collection Process: There are two types of data in the datasets, benign samples and malicious samples. Benign samples contain 33,843webpages obtained from DMOZ database (http://www.dmoz.org), and malicious samples consist of 18,700webpages from XSSed database (http://www.xssed.com) and 3,300 webpages from a real site (weibo.com).

2) Training Data and Testing Data: We divide the samples into two sets, training data set and testing data set. The training set consists of 23,840 benign samples and 14,900malicious samples, while the test set consists of 10,000 benign samples and 7,100 malicious samples.

C. Experiment and evaluation

1) Performance evaluation: In the binary classification, according to the four situations of classification, there are four important definitions.

- TP – the number of positive samples classified as positive
- FN – the number of positive samples classified as negative
- FP – the number of negative samples classified as positive
- TN – the number of negative samples classified as negative

In the binary classification, *precision* and *recall* are the most important values used for evaluations. Conventionally, we take the concerned class as a positive result and the other as a negative one. The two values of the positive result are defined as followings:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Besides, false positive rate (FPR) and false negative rate (FNR) are also used for evaluation. They are defined as following:

$$FPR = \frac{FP}{FP + TN}$$

$$FNR = \frac{FN}{FN + TP}$$

2) Results and evaluation: The results of our experiments using the two implementations are shown in Table II and Table III respectively. According to the tables, we make draw some conclusions through analysis. Firstly, the two implementations of our approach have both achieved good performances in *precision* and *recall*. The results demonstrate that the approach based on the combination of classifiers and improved

TABLE II: The results of Implementation I

		Malicious	Benign
Classified as	Malicious	6849	764
	Benign	251	9,236
Precision		0.900	0.974
Recall		0.965	0.924

TABLE III: The results of Implementation II

		Malicious	Benign
Classified as	Malicious	6,777	608
	Benign	323	9,392
Precision		0.918	0.967
Recall		0.954	0.939

TABLE IV: The comparison of FPR and FNR of malicious samples between our implementations and the method only using classifiers

Method	False Positive Rate	False Negative Rate
Implementation I	0.076	0.035
Implementation II	0.061	0.045
Only Classifier	0.078	0.046

n-gram model is quite effective. Secondly, the difference of performance between implementation I and implementation II shows that the ways of combining the improved n-gram model and the classifiers will affect the result of detection. And we can find that both of them have their own advantages in detecting XSS in OSN.

A good detection method has low FPR and FNR. Table IV shows FPR and FNR of our approach. We can draw some conclusions from Table IV. Firstly, the third row is the results of the method using only classifier. Its FPR and FNR are in an acceptable range of the detection system, and the low FPR and FNR demonstrate that the classifier and the features we identified are effective to do XSS detection in OSN. Besides, compare to the implementation using only classifiers in Table IV, both implementation I and implementation II have better FPR and FNR. Moreover, compare to implementation II, implementation I has lower FPR while its FNR is higher than implementation II. For real-world detection, we can choose either of the implementations based on the situation we face.

VI. Related work

XSS worms are prevalent malicious code in web applications, which take advantages of XSS vulnerabilities. OSN, a novel web application, is suffering more from XSS attacks than other applications. The nodes in OSN's topology are easily infected due to the high concentration of the topology. Therefore, nowadays, researchers are paying more and more attention to OSN's XSS detection.

A. XSS detection with machine learning

Machine learning is a simple and effective way in detecting XSS in webpages. This method is a static way to detect XSS malicious code and costs less time in execution. There are several related researches on it. Since the obfuscation of code is the key point in XSS detection, Likarish [6] et al propose 65 features and 4 classifiers for classification to detect the obfuscated malicious JavaScript. The evaluation of it demonstrates that machine learning can be an effective way in XSS detection. Nunan et al [7] propose a method to identify whether there is XSS code in webpages by implementing automatic classification of webpages using document-based and URL-based features. And the results demonstrate the effectiveness of the proposed features.

The extracted features are important to the performance of classification. When we use machine learning in XSS detection in OSN, we should identify a group of features which are characteristics of OSN XSS worms for better results. In our paper, we propose a set of features which are proved to be effective in XSS detection in OSN.

B. N-gram model

N-gram model is used to predict the next item in a sequence which is in the form of a n-1 order Markov mode. It can be used in anomaly-based detection. Li et al [21] describe Fileprint, an n-gram analysis, as a way to detect malware. They apply 1-gram analysis to different types of files to detect malicious code. And the method is designed to be effective and highly efficient. In the research of AccessMiner [20], Lanzi et al propose to use n-grams as the basic technique to model system calls in order to detect malwares. They generate an n-gram model for the characteristics for malwares. A program which has more than a given number of malicious n-gram instances will be identified as malware. The approach proved to be effective in detecting malwares in system.

N-gram model, which can be used to detect malwares, is based on an ordered sequence. In our opinions, comparing with ordering, the relations between items in the sequence is more important. Thus, we propose an improved n-gram model based on feature vectors to detect XSS in OSN.

C. XSS detection in OSNs

There are many achievements in the research of XSS detection in OSN. Livshits [3] propose Spectator, which is an automatic detection and containment solution for JavaScript. Spectator works by searching for worm-like propagation chains across a social network and controls the further spread by disallowing uploading from nodes that have been reported to be infected. Spectator can be implemented in many deployment models. PathCutter [4], another useful architecture, works by blocking two vital propagation paths of XSS worms: DOM access to different views of client and unauthorized HTTP requests to server. Besides the effectiveness in experiments, the implementations of the prototype have been tested in real-world environment and need minimal modifications to be used in the web application. Louw M.T et al [5] design and implement a tool called BLUPRINT to control the XSS worms. The tool uses an approach to minimize trust placed in browsers for interpreting untrusted content. BLUEPRINT has been integrated in several real web applications and works well in protecting them from XSS attacks.

In our paper, we propose a more simple approach to detect XSS in OSN. Our approach is based on classifiers, which have been used in XSS detection, and the improved n-gram model, which is derived from n-gram model. The implementations of our approach achieve good performance in the experiments which demonstrates the effectiveness of themselves.

VII. Conclusion

This paper proposes a novel approach using classifiers and the improved n-gram model to detect XSS in OSN. At first, we use a group of features related to OSN's XSS worms to build classifiers. Then, we present an improved n-gram model according to the ideas of malware detection using the n-gram model. Furthermore, we propose two implementations of our approach which both have their own advantage and can be used according to different detection policies. Finally, we design and perform some experiments to demonstrate the effectiveness of our approach in XSS detection of OSN.

VIII. Acknowledgements

This work was supported by National Natural Science Foundation of China (NSFC) under Grant No. 61100228 and 61402477, the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDA06030601 and XDA06010701, and the National High-tech R&D Program of China under Grant No. 2012AA013101.

References

[1] Elaine Chan. *Norton Cybercrime Report*, Norton by Symantec. 2012.9.

[2] Wikipedia, *Cross-site scripting*, http://en.wikipedia.org/wiki/Cross-site_scripting

[3] B. Livshits and W. Cui, *Spectator: detection and containment of JavaScript worms*, In USENIX 2008 Annual Technical Conference on Annual Technical Conference (ATC'08). USENIX Association, Berkeley, CA, USA, 335-348. 2008.

[4] Y. Cao, V. Yegneswaran, P. Possas and Y. Chen, *Pathcutter: Severing the self-propagation path of xss javascript worms in social web networks*, In Symposium on Network and Distributed System Security (NDSS'12), 2012.

[5] M. TerLouw 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 (SP '09). IEEE Computer Society, Washington, DC, USA, 331-346. DOI=10.1109/SP.2009.33 http://dx.doi.org/10.1109/SP.2009.33,2009.

[6] P. Likarish, E. Jung and I. Jo, *Obfuscated malicious javascript detection using classification techniques*, In the 4th International Conference on Malicious and Unwanted Software (MALWARE), pp. 47-54, IEEE, 2009.

[7] A. E. Nunan, E. Souto, E. M. dos Santos and E. Feitosa, *Automatic classification of cross-site scripting in web pages using document-based and URL-based features*, In the 2012 IEEE Symposium onComputers and Communications (ISCC'12), pp. 702-707, IEEE, 2012.

[8] B. I. Kim, C. T. Im and H. C. Jung, *Suspicious malicious web site detection with strength analysis of a javascript obfuscation*, International Journal of Advanced Science and Technology, Volumn 26 2011

[9] Samy, *I'm Popular*, http://namb.la/popular/

[10] 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). ACM, New York, NY, USA, 11-20, 2010. DOI=10.1145/1920261.1920264 http://doi.acm.org/10.1145/1920261.1920264

[11] M. R. Faghani and H. Saidi, *Malware propagation in online social networks*, In the 4th International Conference on Malicious and Unwanted Software (MALWARE), pp. 8-14, IEEE, 2009.

[12] M. R. Faghani and H. Saidi, *Social Networks' XSS Worms*, In Proceedings of the 2009 International Conference on Computational Science and Engineering(CSE '09), Vol. 4. IEEE Computer Society, Washington, DC, USA, 1137-1141, 2009. DOI=10.1109/CSE.2009.424 http://dx.doi.org/10.1109/CSE.2009.424

[13] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel and B. Bhattacharjee, *Measurement and analysis of online social networks*, In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC '07). ACM, New York, NY, USA, 29-42,2007, DOI=10.1145/1298306.1298311 http://doi.acm.org/10.1145/1298306.1298311

[14] A. Al Hasib, *Threats of online social networks*, IJCSNS International Journal of Computer Science and Network Security, 9(11), 288-93, 2009.

[15] OWASP, *Cross-site Scripting*, https://www.owasp.org/index.php/XSS

[16] Wikipedia, *Naive Bayes clssifier*, https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[17] Wikipedia, *Support Vector Machine*, http://en.wikipedia.org/wiki/Support_vector_machine

[18] Wikipedia, *Alternating decision tree*, http://en.wikipedia.org/wiki/ADTree

[19] P. Likarish and E. Jung, *A targeted web crawling for building malicious javascript collection*, In Proceedings of the ACM first international workshop on Data-intensive software management and mining (DSMM '09). ACM, New York, NY, USA, 23-26.

[20] L. Andrea, B. Davide, K. Christopher, C. Mihai and K. Engin, *AccessMiner: using system-centric models for malware protection*, In Proceedings of the 17th ACM conference on Computer and communications security (CCS '10). ACM, New York, NY, USA, 399-412.

[21] W. J. Li, K. Wang, S. J. Stolfo and B. Herzog, *Fileprints: Identifying file types by n-gram analysis*, In Proceedings of the IEEE Workshop on Information Assurance and Security, United States Military Academy, NY, USA, 2005,64-71.