Detecting Malicious PDF Files Using Semi-Supervised Learning Method

Di Feng[1,2], Min Yu[1,4,*] , Yongjian Wang[3,*] , Chao Liu[1] and Chunguang Ma[2]

[1]Institution of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]College of Computer Science and Technology, Harbin Engineering University, Harbin, China
[3]Key Laboratory of Information Network Security of Ministry of Public Security, the Third Research Institute of Ministry of Public Security, Shanghai, China
[4]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
yumin@iie.ac.cn, wangyongjian@stars.org.cn
*corresponding author

Keywords: malicious PDF files, malicious JavaScript, semi-supervised learning.

Abstract. With the increase in popularity of Portable Document Format (PDF) documents and increasing vulnerability of PDF users, effective detection of malicious PDF documents has become as a more and more significant issue. In this paper, we proposed a way to detect malicious PDF files by using semi-supervise learning method. Compare with previous studies, this method not only improve detection accuracy and generalization ability by combining with three different classifiers, but also effectively utilize the abundant unlabeled PDF files to retrain classifiers and update module by selecting the "useful" files from unlabeled test set.

1. Introduction

With the increase in popularity of PDF documents, malicious PDF files have become a well-known threaten during the past years. PDF files have been one of the most successful attacking vectors, because it is suitable both for sending emails containing malware and for infections via download [1].

In recent years, more and more scholars have begun to focus on the research of malicious PDF files. There are two categories for taxonomy of academic research on detection methods of malicious PDF files: Static analysis and Dynamic analysis. Static analysis including three different methods: based on embedded JavaScript content in PDF [2-3], based on metadata of PDF [4-5], based on document structure [6-7]. These methods have achieved some progress in the detection of malicious PDF files. However, all of them only focus on using labeled samples, none of them has make full use of the unlabeled samples [8].But in many real-world tasks, the number of labeled training samples is limited due to that labeling the examples requires human efforts and expertise. Abundant unlabeled samples still have not been exploited.

Obviously, it is difficult to have good generalization ability by a poor trained classifier. On the other hand, it is wasting resources without using these unlabeled samples. In this paper, a way to

detect malicious PDF files by using semi-supervised learning methods has been proposed. Experiments show that this method can improve detection accuracy and generalization ability by combining with three different classifiers to jointly vote, which can effectively utilize the abundant unlabeled PDF files to retrain classifiers and update module by selecting the "useful" files from test set.

The paper is divided into five Sections including this one. Section1 introduces the background and significance of this study. Section2 introduce PDF format, JavaScript in PDF and how features will be selected in this study. Section3 describes the detail of how to train and retrain the classifiers and how to detect malicious PDF files. Section4 provides the experimental results and analysis. Section5 discusses the limits of our study and point out the future research direction.

2. About PDF Files

2.1. PDF File Format

The object is the basic data type in PDF file, there are mainly eight class object types[9] which shown in Table1.

Table 1 The object types of PDF file.

object type	description
Boolean	Include logical value "true" and "false"
Numeric	Can be divided into two types as integer and real number
String	Consists of a string of bytes, through the "()" contains a string or the "<>" contains a hex string
Name	Consists of "/" and a string of characters, with uniqueness
Array	Can be composed of a series of different types of one-dimensional objects
Dictionary	Consists of a series of entries which have two parts, the first part is the keyword of the entry(usually name object), the second part is the content of the entry
Stream	Consists of a string between keyword "stream" and "endstream", and without length limitation
Null	Represents an null object

PDF file structure is composed of header, body, cross-reference table and trailer.

- Header: Used to indicate the version of the PDF file. Take the form of "PDF - [version number]" in the first line of the PDF file.
- Body: Body mainly contains the contents of the PDF file which is shown to the user. Each part is presented by the object, which constitutes the specific content of the PDF file, such as fonts, pages and images.
- Cross-reference table: Cross-reference table is a kind of special organization in PDF file. It lists the indirect objects and their location in the file, allows random access to objects in cross-reference table, so that reader can quickly locate the objects in the PDF file.
- Trailer: Trailer provides the location of cross-reference table. By parsing trailer, application can find the cross-reference table and other special objects easily. And trailer also contains a dictionary object, which provides some relevant information about PDF file, such as global information (author, keyword, title) and encryption information.

On the other hand, a PDF file can be seen as a hierarchy structure and consist of objects in the body. The core of hierarchy structure is catalog, a kind of dictionary object. Through catalog, other important information about PDF file can be collected, including page dictionary, names dictionary, outlines dictionary and so on. The hierarchy structure can be shown as a tree in Figure.1.

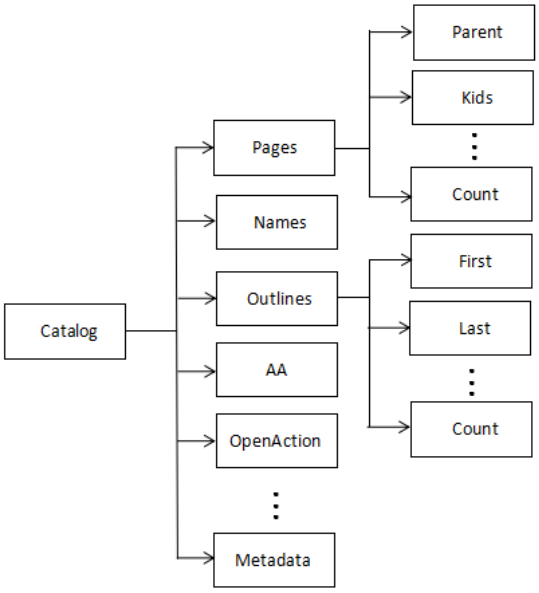

Figure 1 The hierarchy structure of PDF file.

2.2. JavaScript in PDF

There are many attack methods used in malicious PDF files, include attacking on Adobe JavaScript API, heap spray attacks, embedded other malicious files, etc. A lot of investigations and researches show that most malicious PDF files using JavaScript codes when performing malicious functions. So in this paper, we focus on JavaScript codes which embedded in PDF files and select features that based on JavaScript codes contents for later analysis.

JavaScript codes usually appears after the "/JS" by direct references or indirect references. There are many known locations where JS can be found: The AA entry of a dictionary contains an additional annotation dictionary, and every entry in an additional annotations dictionary should be an action dictionary, which can contain JS; The OpenAction entry of a dictionary also can contain JS; A page tree consists of a tree of page tree node dictionaries and page dictionaries. The page dictionaries are the leaves of the tree and they can contain references to JS in the AA and annotations entries; An outline tree consists of a tree of outline dictionaries and outline item dictionaries. The latter are the leaves of the tree, they can contain references to action dictionaries containing JS; Forms are structured in a tree and the leaves can contain references to JS; A PDF file's name tree can contain JavaScript that gets executed on document load.

2.3. Features Select

For the JavaScript codes embedded in PDF files, we adopt N-gram algorithm for features extraction. N-gram is a kind of text or language analysis algorithm that based on Markov chain, which has been widely used in the static test of malicious codes. In the detection of malicious PDF files, N-gram algorithm was adopted to extract the JavaScript codes and analysis whether the codes has certain characteristics or not. These features will be used for detection model building as multidimensional vectors. There are some common features listed in Table 2 which usually be selected in malicious after features extraction.

Table 2 Some common features of malicious PDF files.

feature	description
for	A feature that represent a large number of cycles appear in codes
while	Same as above
eval	Calculation of a string and the implementation of the JavaScript codes
escape	Used to encode string
unescape	Used to decode the string which encoded by escape()
fromCharCode	Used to decode encoded strings for execution using eval()
replace	Replace statement in JavaScript
spray	A keyword of heap spray
%u	Same as above
util.printf	A function used in heap spray attacks
getAnnots	Get annotation objects
customDictionaryOpen	Open the dictionary object

3. The Proposed Framework and Classification Method

3.1. Overview

The process of detecting malicious PDF files in our framework mainly consists of two parts: Known files module and Detection module. Figure.2 shows the framework and the process of detecting and acquiring new malicious PDF files by maintaining the updatability of the known files module and detection module.

Known files module composed of known malicious files, both of the labeled malicious PDF files from original samples set and the new received files which classifiers consider as malicious obviously. When new unlabeled sample PDF files are accepted, it will be compared with the known files module to determine whether it is malicious, by calculating the MD5 value. If the MD5 value of the file has been stored in the module, it will considered as malicious decisively without delivering to the classifiers.

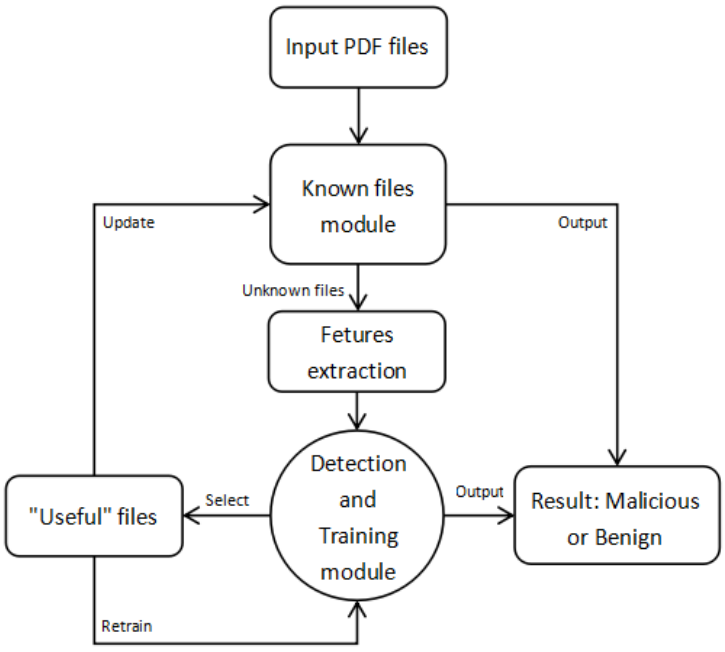

Figure 2 The process of detecting and retraining.

Detection and training module is trained and retrained by tri-training algorithm which Zhou et al. proposed in[10][11]. Tri-training algorithm is a kind of effective semi-supervised learning method. It attempts to exploit unlabeled data using three classifiers, and which focus on how to efficiently select most confidently predicted unlabeled samples to label and produce final hypothesis. It will be described in detail in section3.2. On the other hand, for the remaining PDF files which are unknown and have been transformed into feature vectors, this module will classify these files into two categories: malicious and benign. And this module select the "useful" files from unlabeled files, which utilized to rich known files module and update the classifiers. These processes will be explained in detail in section3.3.

3.2. Tri-training Algorithm

Let L denote the collection of original labeled data, U denote the collect of all unlabeled data. Training set L_1, L_2, L_3 is bootstrap sampled from sample set L. Correspondingly, h_1, h_2, h_3 are three initial classifiers generated by training L_1, L_2, L_3. The purpose is to make three classifiers h_1, h_2, h_3 different from each other, so integrate them to work can improve detection precision and generalization ability. x_i is an arbitrary test sample in U. In each round, if h_1 and h_2 have same classification result for x_i while h_3 have an opposite result, then in next round, add x_i to the training set L'_3, $L'_3 = L_3 \cup \{ x_i | x_i \in U$ and $h_1(x_i) = h_2(x_i)\}$; a similar method to produce L'_1, L'_2. Using these new generated training set to update the classifiers h_1, h_2, h_3 in each round, until the classifiers do not change obviously or satisfied certain conditions.

The Tri-training algorithm can adopt an implicit confidence measurement by majority voting, which improves the efficiency of the training process. However, this implicit confidence measurement might generate false positives and introduce noise into training set. In order to avoid the influence of noise, the criterion based on theoretical results of learning from noisy examples is derived as follow:

If a sequence σ of m samples is drawn, where the sample size m satisfies (1)

$$m \geq \frac{2}{\zeta(1-2\eta)^2} \ln(\frac{2N}{\delta}) \tag{1}$$

where ζ is the hypothesis worst-case classification error rate, η (<0.5) is an upper bound on the classification noise rate, N is the number of hypotheses, and δ is the confidence, then a hypothesis H_i that minimizes disagreement with σ will have the PAC property, i.e.

$$\Pr\left[d\left(H_i, H^*\right) \geq \zeta\right] \leq \delta \tag{2}$$

Where $d(,)$ is the sum over the probability of elements from the symmetric difference between the two hypothesis sets H_i and H^* (the ground-truth). Then, the u can define as

$$u = \frac{2\mu}{\zeta^2} \ln\left(\frac{2N}{\delta}\right) = m(1-2\eta)^2 \tag{3}$$

For each classifier, in order to keep improving the performance in the training process, the u value of the current round should be greater than that in its previous round. Let L^t and L^{t-1} denote the newly labeled data set of a classifier in the *t-th* round and *(t-1)-th* round, respectively. Then the training sets for this classifier in the *t-th* round and *(t-1)-th* round are $L \cup L^t$ of the size of $|L \cup L^t|$ and $L \cup L^{t-1}$ of the size of $|L \cup L^{t-1}|$, respectively. Let e^t and e^{t-1} denote the upper bound of the classification error rate of the hypothesis derived from the combination of the other two classifiers in the *t-th* round and *(t-1)-th* round, respectively. The condition that a classifier's performance can be improved through the refinement in the *t-th* round is shown as (4)

$$0 < \frac{e^t}{e^{t-1}} < \frac{\left|L^{t-1}\right|}{\left|L^t\right|} < 1 \tag{4}$$

In order to ensure that the (4) is established, L^t is randomly subsampled to size s as following.

$$s = \left\lceil \frac{e^{t-1}\left|L^{t-1}\right|}{e^t} - 1 \right\rceil \tag{5}$$

And L^{t-1} should satisfy (5).

$$\left|L^{t-1}\right| > \frac{e^t}{e^{t-1} - e^t} \tag{6}$$

3.3. Detection and Updating

Generally, compared with previous semi-supervised learning method, tri-training requires neither the existence of views nor special learning algorithm. So in our work, we employed the SVM algorithm using the radial basis function kernel function as learning algorithms. We employed the SVM as learning algorithm for the following reasons: 1) SVM has been successfully used to detect worms and malware. 2) The classifier trained by SVM algorithm is black-box that is hard for attackers to understand. 3) SVM has the ability to handle large numbers of features, which makes it suitable for detecting malicious files.

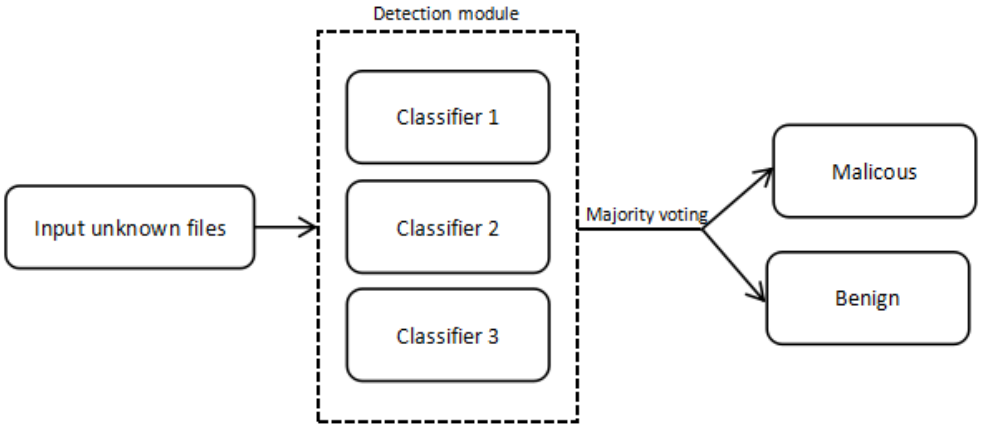

Figure 3 The detection method.

When detection module h received unknown and unlabeled PDF files, these files will be delivered to three classifiers h_1, h_2, h_3 and each of them will be classified independently. At last, through majority voting, detection module h decides these files whether be malicious or not. For a certain file x_i, several results will appear: 1) If all the classifiers consider x_i as benign, detection module also consider it as benign. 2) If all the classifiers consider x_i as malicious, detection module also consider it as malicious. 3) If two of the classifiers consider x_i as malicious or benign and the other one consider it conversely, then three classifiers are combined by majority voting. e.g: $h_1(x_i)=h_2(x_i)\neq h_3(x_i)$, then $h(x_i)=h_1(x_i)=h_2(x_i)$. Which shown as in Figure.3. Unlabeled files which result satisfy one of the latter two conditions will be selected as "useful" files to update the system.

The first type of "useful" files which all the classifiers consider it as malicious, will be labeled as malicious and be acquired by the known files module to rich the malicious PDF files library. The second type of "useful" files which two of the classifiers consider it as malicious or benign and the other one consider it conversely, we adopt "majority teaches minority" strategy, two classifiers will teach the third classifier on this sample. This file will be labeled by the majority classifiers, then add

it to third classifier's training set. After the end of each round of classification, every classifier will be retrained based on a training set that have been updated, the method to retrain classifier and reduce the noise effects have been describe in section3.2.

4. Evaluation

4.1. Evaluation Standard

The recall and precision are two measures that widely used in information retrieval and statistical classification filed to evaluate the quality of the results. So in our study, we use these two measures as evaluation standard.

Let the number of PDF files to be classified is N, the number of malicious files that be detected correctly is N_a, the number of benign files that be detected as malicious is N_b, the number of malicious files that be detected as benign is N_c, the number of benign files that be detected correctly is N_d. Obviously, $N=N_a+N_b+N_c+N_d$; and the number of actual malicious files, expressed in N_m, $N_m = N_a+N_c$. Related definitions can refer to Figure.4.

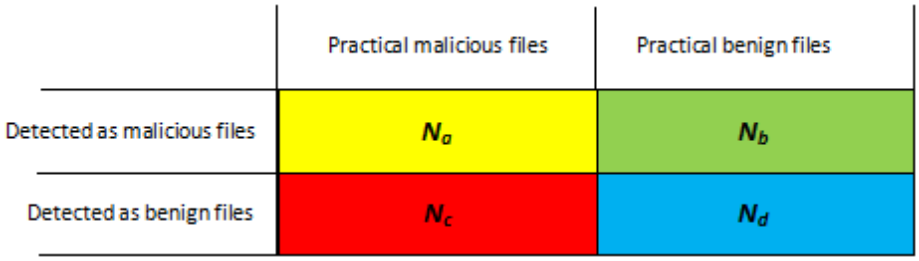

Figure 4 Definition of sample set.

Recall is the ratio of the number of malicious files that be detected to the number of actual malicious files, represents how many malicious files that can be detected; Precision is the ratio of the number of malicious files that be detected to the number of all files that be detected (including both correct and false positives), represent how many file that be detected as malicious are accurate; Let recall expressed in R, and precision expressed in P. Related definitions can refer to (7) and (8).

$$R = \frac{N_a}{N_a + N_c} = \frac{N_a}{N_m} \tag{7}$$

$$P = \frac{N_a}{N_a + N_b} \tag{8}$$

4.2. Results and Analysis

In order to verify the effectiveness of our employed methods, a large number of file samples are needed. There are 9861 PDF files that we collect from Internet in total, including 5518 malicious files and 4343 benign files. 5218 malicious files and 4043 benign files are selected as original sample set to extract features and generate detection model. The remaining files include 300 malicious files and 300 begin files, are divided into three groups, simultaneously test the sample separately. After the end of each group test, classifiers will be retrained by utilizing these test samples. Results can be seen in Table 3.

Table 3 Detected results of each group.

	Group 1		Group 2		Group 3	
	Malicious	Benign	Malicious	Benign	Malicious	Benign
Detected as Malicious	88	2	84	0	91	1
Detected as Benign	12	98	16	100	9	99

In Table 3, it can be seen that the detection method have an acceptable results for the test samples. For these malicious PDF files, about ten documents are classified falsely as benign in each group. It is the malicious samples which are not based on JavaScript codes, or falsely extracted codes that lead to the produced omission; For these benign PDF files, few are classified as malicious. Through analysis, we find that the false positive were produced due to they may use some of the vulnerable methods in a standard manner, like getAnnots().

In order to compare with the semi-supervised learning methods that we employed, a single classifier that based on SVM algorithm also adopted to test the sample. The recall and precision of the two methods in this experiment are presented in Table 4.

Table 4 Comparison of recall and precision of two methods.

	Semi-supervised learning methods			Only adopt SVM		
	Group1	Group2	Group3	Group1	Group2	Group3
Recall	88%	84%	91%	82%	78%	84%
Precision	97.7%	100%	98.9%	97.6%	100%	98.8%

As shown in Table 4, precision of the two methods both are nearly 100%, which means most of the files that be detected as malicious are certainly malicious, both of the methods produce few false positive. But through attentive comparison it can be found that the former is still slightly higher than the latter. About recall, both two methods have omission in detecting malicious, but the methods we employed have higher recall than the other one, means the methods can detect more malicious than the latter as we expect.

5. Conclusions and Future Work

In this paper, a framework that detecting malicious PDF files using semi-supervised learning method were presented. This method is able to take advantage of a large number of unlabeled samples in reality and have a better recall and precision compared with the traditional learning methods in detecting malicious PDF files.

But there are still a lot of problems that we should be paid attention to: Our research only focus on malicious PDF files which based on JavaScript content. And we adopted an implicit confidence measurement with the method of semi-supervised learning, which might not be as accurate as explicit estimation. In future work, select the features of PDF files in other ways will be considered, not only detect JavaScript codes that embedded in PDF files. And we will try to combine semi-supervised learning method with the artificial to improve the accuracy of detecting malicious PDF files.

Acknowledgements

This work is supported by Strategic Pilot Technology Chinese Academy of Sciences (No. XDA06010703), National Natural Science Foundation of China (No. 61173008, 61402124, 61303244), Young Scholar Foundation of Institute (No. 1104005704) and Key Lab of Information Network Security, Ministry of Public Security (No. C15607).

References

[1] AVE-TEST security report 2015/16. https://www.av-test.org/en/news/news-single-view/current-risk-scenario-av-test-security-report-facts-at-a-glance/, 2016
[2] Laskov P, Srndic N. Static detection of malicious JavaScript-bearing PDF documents[C]// Twenty-Seventh Computer Security Applications Conference, ACSAC 2011, Orlando, Fl, Usa, 5-9 December. DBLP, 2011:373-382.
[3] Schmitt F, Gassen J, Gerhardspadilla E. PDF Scrutinizer: Detecting JavaScript-based attacks in PDF documents[C]// Tenth International Conference on Privacy, Security and Trust. 2012:104-111.
[4] Smutz C, Stavrou A. Malicious PDF detection using metadata and structural features[C]// Computer Security Applications Conference. 2012:239-248.
[5] Pareek H. Entropy and n-gram analysis of malicious PDF documents[J]. International Journal of Engineering, 2013.
[6] Srndic N, Laskov P. Detection of Malicious PDF Files Based on Hierarchical Document Structure. 20th Annual Network & Distributed System Security Symposium,2013.
[7] Maiorca D, Ariu D, Corona I, et al. A structural and content-based approach for a precise and robust detection of malicious PDF files[C]// International Conference on Information Systems Security and Privacy. 2015:27-36.
[8] Nissim N, Cohen A, Glezer C, et al. Detection of malicious PDF files and directions for enhancements: A state-of-the art survey[J]. Computers & Security, 2015, 48(779):246-266.
[9] PDF Reference. http://www.adobe.com/devnet/pdf/pdf_reference.html, 2016.
[10] Zhou Z H, Li M. Tri-training: exploiting unlabeled data using three classifiers[J]. IEEE Transactions on Knowledge & Data Engineering, 2005, 17(11):1529-1541.
[11] Zhou Z H. Semi-supervised learning by disagreement[J]. Knowledge and Information Systems, 2010, 24(3):415-439.