MALICIOUS PDF DOCUMENT DETECTION BASED ON FEATURE EXTRACTION AND ENTROPY

Himanshu Pareek[1], P R L Eswari[1] and N. Sarat Chandra Babu[2]

[1]Center for Development of Advanced Computing, Hyderabad, India
[2]Center for Development of Advanced Computing, Bengaluru, India

ABSTRACT

In this paper we present a machine learning based approach for detection of malicious PDF documents. We identify various features in PDF documents which are used by malware authors to construct a malicious file. Based on these feature set we arrive on models which is used to detect malicious PDF documents. Based on these feature sets, detection rate is high as compared to approaches which depends on analysis of JavaScript embedded in the PDF document.

KEYWORDS

Malware Detection, Malicious PDF Document, Heuristics

1. INTRODUCTION

PDF documents are heavily used for launching attacks by cyber criminals [1]. A PDF document with exciting topic is sent to victim and when document is opened, particular vulnerability in the rendering software implementation or configuration is exploited to launch the next step of attack. For example, direct execution of native executable (if code was embedded in the PDF document itself), injection of code into a running process or even downloading binary from the internet and then executing it. In this work, we analyze a data set of benign and malicious PDF documents and observe differences in them using machine learning techniques. Then a document can be analyzed to determine whether it is malicious or benign.

2. RELATED WORK

Pavel Laskov and Nedim Srndic [2] have implemented an approach for static detection of malicious java script bearing PDF documents. Though this approach is very useful as it does not involve any sandbox but would be able to detect malicious PDF documents if they use mostly obfuscated java script. Zacharias Tzermias et al. [3] designed and implemented MDScan which combines static and dynamic analysis for malicious PDF detection. This feature however pushes towards reliable detection of malicious PDF documents but emulation technique increases processing time. It also required emulation software on user system. Florian Schmitt [4] also presents an approach to detect malicious PDF document based on java script analysis. Our work takes these previous works to detect obfuscated java script into account but also adds other parameters like OpenAction, deflate etc. Similar to our work is the one done by Charles Smutz and Angelos Stavrou [10]. In their work, they have taken a huge set of features including font, creation date etc but have not disclosed whether they really affect the classification results. While in our work, we target only few crucial features.

3. DATASETS

We collected malicious PDF files from various malware hosting websites like offensivecomputing.net, contagiodump.blogspot.com, malware.lu and virusshare.com. We decided to keep all these malwares in training set (3000 benign and 3000 malicious) and keep on collecting new malwares from various users and keep them in test set. We kept 700 PDF documents in the test set.

4. OUR APPROACH

Our approach is based on machine learning. We first took classified sets of malicious and benign PDF files and extracted various features from them. Then based on standard machine learning classifier algorithms we build the model. Then these models are used to classify a unknown given file as malicious or benign. We analyzed all the PDF documents in training set manually for the ten features shown in Table 1. Based on our own analysis of malicious PDF document we also added parameter 7 that is the presence of both java script and OpenAction.

These feature set was decided on the work done by Paul Baccas at Sophos [5] to indicate what are prominent features used in malicious PDF documents as compared to genuine documents. Importance of these feature sets can be understood from Adobe PDF Specification [6]. For example:

/OpenAction if present in an object header line specifies a destination page which is displayed when document is opened or may specify an action to be performed when the document is opened. It can also be used in attacks [7]. Similarly 'Launch' specifies to launch an application, usually to open a file. '/JS' and '/JavaScript' indicates towards the presence of JavaScript inside the object. PDF Specification supports many data encoding techniques like 'FlateDecode' to compress data which may help to keep the document size less. In our approach, we pick up such filters and uncompress the data to discover any java script code. If java script is found after deflating the compressed data then we assign a severity of two. We rely on the approach given by Pavel Laskov [4] to detect obfuscated java script and if found that is given a severity of 3.

We also took in account that a PDF document with the intent of exploiting may contain some garbage strings and hence reduce the document entropy. However this may not be a reliable method to detect malicious document we just used this as a parameter with very strict restriction. If entropy of a document goes below than 2 then only flag is raised an a severity of two is assigned.

Following feature vector is calculated for every PDF files for further calculation

Table 1: Features Extracted From a PDF files

S. No.	Feature	Severity Number
1	/JS - The number of JavaScript launched.	1
2	/JavaScript - The number of embedded JavaScript.	1
3	/OpenAction - The presence of an open action	1
4	/Launch - The use of the /Launch statement.	1
5	AsciiHexDecode - The use of the asciihexdecode filter.	1
6	Chars After Last EOF	1
7	(/JS OR /JavaScript) AND (/OpenAction OR /Launch)	3
8	If Stream Entropy less than 2	2
9	JavaScript after deflating	2
10	Obfuscated JavaScript	3

We calculate Euclidean distance score by using following formula.

$$ED = \sqrt{\sum_{i=0}^{i=N} S_i^2}$$

We first used whether Euclidian distance can be used to classify a document as benign or malicious but Euclidian distance showed high false positive rate. Then, we turned to specialized machine learning algorithms to classify the PDF document based on our feature set.

5. Implementation and Evaluation

We used the pdfid.py from Stevens [8] to start with identification of various features from the PDF file and calculate ED score of various malicious PDF documents. Then to utilize the machine learning algorithms, we used Weka implementation [9] of various algorithms. True positive rate of various classifiers are depicted in Figure 1 and false positive rates are depicted in Figure 2.

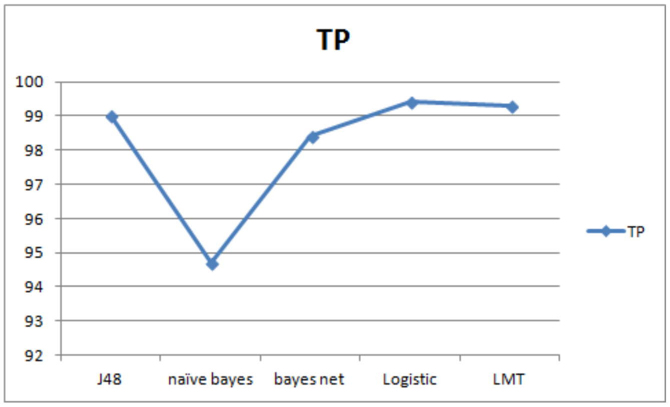

Figure 1: True Positive Rate of Various Classifiers

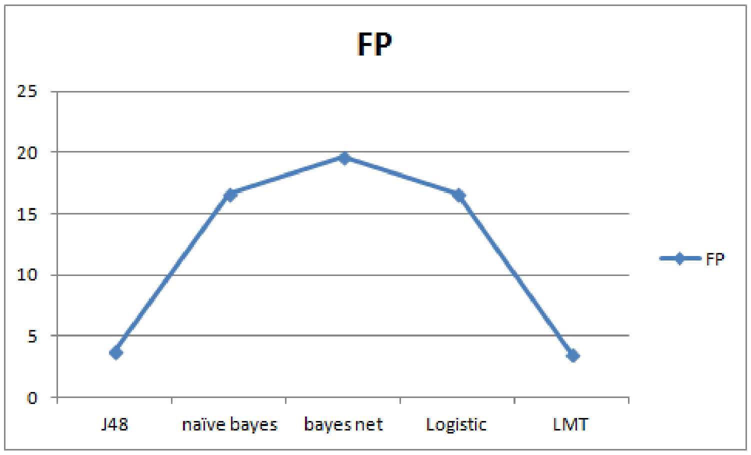

Figure 2: False Positive Rate of Various Classifiers

We also did analysis to determine how much time is taken to process a single PDF file. As approach is based on feature extraction there is no explicit difference between processing times for benign and malicious files. Time increases with the file size. this is shown in the figure 3.

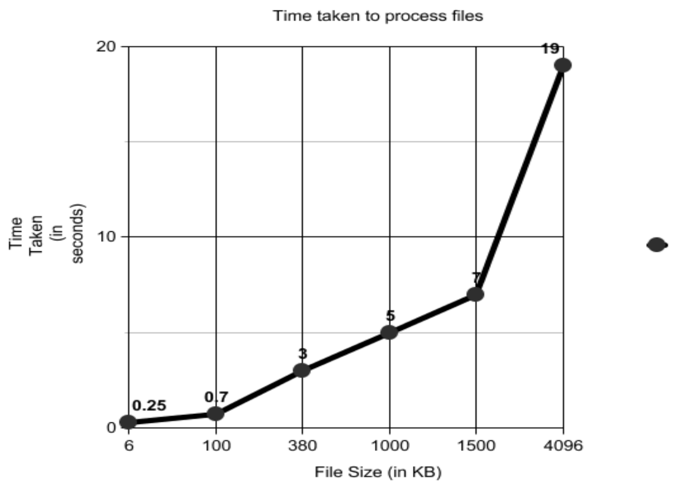

Figure 3: Processing Time for Files.

6. Conclusions

With this paper, we presented an approach to detect malicious PDF documents based on feature extraction and machine learning algorithms. Our approach does not utilize any java script emulation methods. We also show that carefully chosen set for machine learning algorithm can provide better results. However with our set of features LMT performed slightly better than J48 but both were better than Naive Bayes and Bayes Net. If we remove few parameters like \JS and \OpenAction, detection rate falls by fair enough margins.

REFERENCES

[1] Bruce Schneier, "PDF most common malware vector", Available at *httsp://www.schneier.com/blog/archives/2010/03/pdf_the_most_co.html*

[2] Laskov, Pavel, and Nedim Šrndić. "Static detection of malicious JavaScript-bearing PDF documents." Proceedings of the 27th Annual Computer Security Applications Conference. ACM, 2011.

[3] Tzermias, Zacharias, et al. "Combining static and dynamic analysis for the detection of malicious documents." Proceedings of the Fourth European Workshop on System Security. ACM, 2011.

[4] Schmitt, Florian, Jan Gassen, and Elmar Gerhards-Padilla. "PDF Scrutinizer: Detecting JavaScript-based attacks in PDF documents." Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on. IEEE, 2012.

[5] Paul Baccas, "Finding Rules For Heuristic Detection Of Malicious PDFs: With Analysis Of Embedded Exploit Code", Virus Bulletin Conference September 2010

[6] Adobe PDF Specification. Available at *http://www.adobe.com/devnet/pdf/pdf_reference.html*

[7] Exploit Action with PDF OpenAction. Available at http://securitylabs.websense.com/content/Blogs/3202.aspx

[8] Didier Stevens, PDF Tools. "pdfid.py", Available at *http://blog.didierstevens.com/programs/pdf-tools/*

[9] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[10] Smutz, Charles, and Angelos Stavrou. "Malicious PDF detection using metadata and structural features." Proceedings of the 28th Annual Computer Security Applications Conference. ACM, 2012.