Scientific Journal of Impact Factor (SJIF): 4.72
e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 4, April -2017

A STRUCTURAL APPROACH FOR PDF DOCUMENTS CLASSIFICATION

Ritu Choudhary[i]*, Mehak Khurana[ii]

[1]*Department of computer Science ,The NorthCap University, Gurgaon, India*
[11] *Department of computer Science, The NorthCap University, Gurgaon, India*

Abstract: *From last few years, the PDF document has proved to be a great acomplishment vector for malicious infections, making upto 80% of all the exploits found by Cisco ScanSafe. Generating novel PDF files is quite an easy task and the aggregate PDF documents recognized as the malicious has extended beyond the potential of security analysts to analyse them manually. The solution proposed by our paper is to automatically extract the features from the PDF files to analyse the malicious and non malicious behaviour ogf the PDFs and to gather and classify them, so that the resembling malware may be detected without analysing them manually, hence reducing the workload of the malware experts. The features discussed may also be studied to determine the trends followed within the PDF files, vairious exploits or obfuscation technique. Finding homogeneity in PDF files expose the further information about a data set used. In our study we collected dataset from vairious sources and the tested them for the maliciousness so as to classify them.We also report the performance of different classifiers. Finally, we believe that, to reduce the targeted attacks, a more cautious machine learning based detectors are needed.*

Keywords: *PDF, Machine Learnig, Malware, Objects, Streams, JavaScripts.*

1. Introduction

Today the technology has broadened beyond the bounds of Desktop Systems to an extensive number of gadgets like, smart phones, tablets and other devices. Data creation and its distribution are presently at its peak, so the document formats must be tractable across different environments (J. Gantz and D. Reinsel, 2012). The Portable Document Format (PDF) exhibit a standard format to create and read documents over various platforms. PDF documents can be generated and read across operating systems, mobile devices, tablets, and also by printers and scanners. The motility of the PDF has led to its boundless adoption and usability, making it a commonly-used file format now days.
The Adobe Portable Document Format was generated by John Warnock in 1993 and was labeled an open standard for electronic documents by the International Organization for Standardization (ISO) in 2008. In 2010, nearly 90% of systems had a version of Adobe Acrobat or Adobe Reader installed (A. C. Madrigal, 2012). The adaptability of PDF documents found its way to their success: the PDF documents not only support the text boxes and character encodings, but also allows embedded JavaScript and ActionScript, action triggers and dynamic forms (J. Yonts, 2010). Nevertheless, the distinctiveness of the content that is allowed in PDF files entertains the attackers with many features of attack, leading the PDF to evolve into one of the most favored advent for transferring malicious content.
In 2001, the very first malicious PDF document was discovered, and an amiable attack had happened; it contrived Adobe Acrobat users by mailing itself to everybody in a user's Microsoft Outlook contact list. The Common Vulnerabilities and Exposures (CVE) is a prototype which was used by Mitre to distinctly identify the publicly known software defects (D. Waltermire, K. A. Scarfone, 2011). The first CVE addressing Adobe was introduced in 2006, stating malicious PDF files as a severe threat. Since then, the number of PDF document attacks were high in number, constituting about 80% of all the exploits in 2009 (D. Danchev, 2009).
The targeted and untargeted social engineering attacks using PDF files have justified being successful for malware infections via various techniques (Selvaraj, K., et al., 2010). These strategies are employed in mass mailing the PDFs: malicious PDF files are delivered to several users with the aim of accomplishing as many successful attacks as possible. The mass mailed PDF files are also disposed to tempt users by claiming to contain information relating to some catchy contests or prizes. Targeted PDF documents are sent to a less number of users, as they are sent to specific users who may benefit the attackers. With a little amount of knowledge, attackers make these PDF documents more attractive to the targeted users by using trending events and personal information to appear authorized.
Drive-by-downloads happens when a user abruptly encounters an infected web site (Selvaraj, K., et al., 2010). Malicious PDF files are a popular medium in drive-by-download attacks. The PDFs opens in the background of the website and infects a user system without his or her knowledge. Malicious PDF documents which are hosted on web pages are generally smaller in size than emailed PDFs as they need to be executed in a shorter period of time, and are not comprised of any content other than the exploit and the payload, whereas spammed malicious PDF files may contain legitimate appearing text or images along with the exploit.
The adaptability of PDFs creates difficulty in the detection and the analysis of malicious PDFs. A small alteration in one tiny part of the PDF may hinder the document’s detection based on signature, but still injects the same malicious payload. Manual

analysis of a malicious document is a slow process that requires both static and dynamic analysis. Since malicious PDF documents are evident to modify, there are collections of thousands of distinct PDF documents still waiting to be manually analyzed by security experts, which is an impossible attainment. Therefore, this thesis proposes a set of features that are to be used in automated analysis of a large set of malicious PDFs. The results of this analysis compel us to question the possibility of efficient automated detection of malicious PDFs or other malicious documents like Microsoft Word, and if a stricter document structure might make automated detection a bit simpler.
Our thesis is build upon the previous work done on PDFs by focusing on a novel domain: malicious PDF file classification and grouping for the analysis of a large data set of malicious files. Due to the pervasive nature of PDF documents, creating new PDF documents is very easy task and the number of PDF documents classified as malicious has evolved beyond the capabilities of security researchers to analyze manually. The solution proposed by our thesis is to automatically extract the specific features from the PDF documents and classify them, so that similar malware may be detected without manual analysis, thus cutting down the workload of the malware analyst. These features may also be studied to identify trends within the PDF documents, such as similar exploits or other obfuscation techniques.

2. Background

2.1. PDF Document Format

The Portable Document Format (PDF) was introduced by Adobe in 1993 as an open standard for depicting documents unconfined to software, hardware and operating system used. The PDF format was formally released in 2008 as an open standard by the International Organization for Standardization as ISO 32000-1(Horst, J., et al.). However, the Adobe Reader grant for loose interpretation of the model, often opening, and tackle to fix, highly malformed documents. The number of features acquired within a PDF file and the lightly interpreted standard allow malware injectors multiple features for infection with a single PDF file. Some of the features, such as Adobe Flash, are identical to other web exploits, while others, such as JavaScript, have intricacy unique to PDF files. For instance, for the author to access and change distinct parts of the document using JavaScript, Adobe provides its own API for JavaScript. The intensive understanding the PDF document structure is essential for the analysis of malicious PDF files.
A PDF file comprise of four main parts shown below; the header, the main body, the cross-reference table and the trailer (Horst, J., et al.).

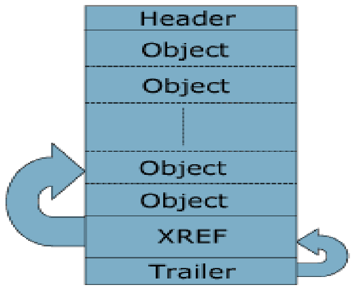
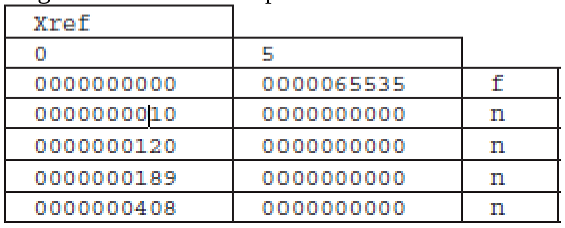


Figure 1: The structure of a PDF document.

The header is always the first line of the PDF file. It should be of the format %PDF-*[version number]*.
The main body is where the contents of the PDF document are found, represented as objects or object streams.
The cross-reference table, the xref table, is a table determining the byte offset of every object specified in the body and contains a list of objects and their position and status within the file. The xref table is used for direct access to a speci fied object in the document without searching. There can be multiple xref tables within a file.



Xref		
0	5	
0000000000	0000065535	f
0000000010	0000000000	n
0000000120	0000000000	n
0000000189	0000000000	n
0000000408	0000000000	n

Figure 2: The Xref table.

The trailer gives the location of the cross-reference table and of several special objects within the body of the file. The most important key for the parser is the /root key. The trailer is then followed by a startxref, the location of the last xref table, and then by %%EOF which should end the file**.**

2.2. Malicious Use of PDF Files

As stated earlier, PDF is an extensive standard for exchanging the documents. Implicitly every computer has a PDF reader installed already, or the user may have a difficulty in participating in society. Also, everyone is aware of the fact that PDF is a highly powerful and flexible format, which can act as a dream come true for an attacker for carrying out various attacks.

3. Related Work

The elementary approach to malicious PDF document detection was introduced via static analysis on the raw document, by using n-gram analysis (Li et al., 2007; Shafiq et al., 2008) and decision trees (Tabish et al., 2009). But, these approaches were not only accustomed for the PDF documents, as they focused to detect many more different file formats, such as EXE, DOC, etc. Since such approach could be employed to perform raw analysis to check for the malicious content over many other file formats apart from PDF documents. However, such approaches fail to counter modern obfuscation techniques which use AES, heap spraying and other polymorphic malware techniques.

Much of the recent work on malicious documents has embarked to define methods for automatic identification of malicious documents for use in intrusion detection and antivirus software. There have been several attempts to identify the best suited method for quickly identifying malicious documents. In one study of malicious documents, Li et. al. found that in practice the attackers often inject malicious code into existing benign documents, and concluded that a hybrid static and dynamic analysis is the best solution for malicious documents (Li, S. Stolfo, et al. 2007). Other studies have used various machine learning techniques to classify benign and malicious PDF documents, with some success (C. Smutz, C. Vatamanu, J. Cross, P. Laskov, D. Maiorca, et al.). Several web-based online tools, such as Wepawet (M. Cova, C. Kruegel, et al.,2011) and VirusTotal, allow a user to upload a malicious document online for the tool to classify it as malicious or benign and list out the various malicious objects contained in them. Wepawet uses different approaches to provide an automated in-depth analysis of the file. VirusTotal avails popular anti-virus software to generate a score of maliciousness for a document. However, novel the techniques used in the file, less likely anti-virus software and thus the cumulative VirusTotal are to state it as malicious.

4. Proposed Approach

Vast majority of recent works on malicious PDF detection focused on the analysis of either the JavaScript code which is for the content based systems or the PDF file structure for the structural systems. Such information is usually processed by a *machine learning* system, i.e., it is converted into a *vector* of numbers (*features*) and sent to a mathematical function (*classifier*).

To extract information, we adopted the tools PeePDF and Pdfid. These tools perform an in-depth analysis of PDF files to detect known exploits, suspicious objects, or potentially malicious functions. Moreover, they will extract and parse, as a separate sample, any *embedded* PDF file. When combined, these tools provide a reliable parsing process in comparison to other ones, which naively analyzes PDF files ignoring their logical properties, thus allowing attackers to easily manipulate them.

4.1. Features

Through manual analysis of PDF documents, we identify several features to analyze a large set of PDF documents. By identifying trends among malicious PDF documents, similar exploits can be identified without manual analysis of each document (D. Maiorca, et al.). The following chapter will identify and explain the features chosen for our sample set, justify the choices, and provide recommendations for automated extraction of features.

General Structure. We extracted features which contained information: **(i)** The *size* of the document; **(ii)** The number of *versions* of the document; **(iii)** The number of *indirect objects* inside the document; **(iv)** The number of *streams* in the document; **(v)** The number of *compressed objects* in document; **(vi)** The number of *object streams* in document; **(vii)** The number of *X-ref streams* in file; **(viii)** The number of objects containing *JavaScript*.

S.NO.	FEATURE
1	*Size of the file*
2	*Versions of the file*
3	*Indirect objects*
4	*Streams*
5	*Compressed objects*
6	*Object Streams*
7	*X-ref streams*
8	*Objects containing JavaScript*

Table 1: Features Extracted from PDF files

4.2. Feature Extraction

To create the feature vectors one first needs to decide what features are relevant to the learning task and then do the required measurements to determine the values of the features on a sufficient number of samples.

In order to begin analysis of the features detailed in above section, the malware analyst must extract the features from the PDF documents. Due to the large size of malicious PDF document collections, an analyst requires automatic extraction of the features. Feature extraction begins with parsing the PDF document, a process for which several open-source tools are readily available. But, we created scripts in python which extracted the features from PDF documents.

In their interactive consoles, peepdf and pdfid have commands that will display the number of Java Scripts, Objects, Streams, etc in a PDF document. The features discussed in this chapter are used to group malicious PDF documents without manual analysis of each document. However the features are not readily extractable. Our method achieves automated extraction of features through the use of open source tools and simple scripts that parse the malicious PDF documents and extract the features. The following sections discuss analysis of these features to group malicious PDF documents and the results of analysis on the data set.

4.3. Dataset

We conducted our experiments using real and up-to date samples of both benign and malicious PDFs in the wild. Overall, we collected various unique malicious samples from Contagio malware dump, a well-known reputable repository which provides information about latest PDF attacks and vulnerabilities. Moreover, we randomly collected various samples of benign PDFs, by means of the public websites like offensivecomputing.net, malware.lu and virusshare.com, etc. We kept a balance between malicious and benign files for the purpose of supervised training.

4.4 Classifiers Used

All classifiers described here are implemented in the Weka machine learning framework (Hall, M., Frank, et al., 2009) which will be used in testing later in the section, and some of the classifier descriptions have been found in the Weka documentation.

5. Comparing Malicious and Benign Documents

This section will present the results of automatically extracted the feature set and using the Features to group a data set of malicious and non malicious PDF documents. We used the pdfid.py and peepdf to start with identification of various features from the PDF file and following are the results generated comparing each feature in malicious and non malicious PDF documents.

Objects

The graph in Figure 3 shows the scale of number of objects in malicious versus non malicious PDF documents. It can be clearly observed that the number of objects is more in 90% of the malicious files. This is due to the fact that the attackers use the objects as a container to the malicious contents which can easily be used to inject the malware inside the PDFs.

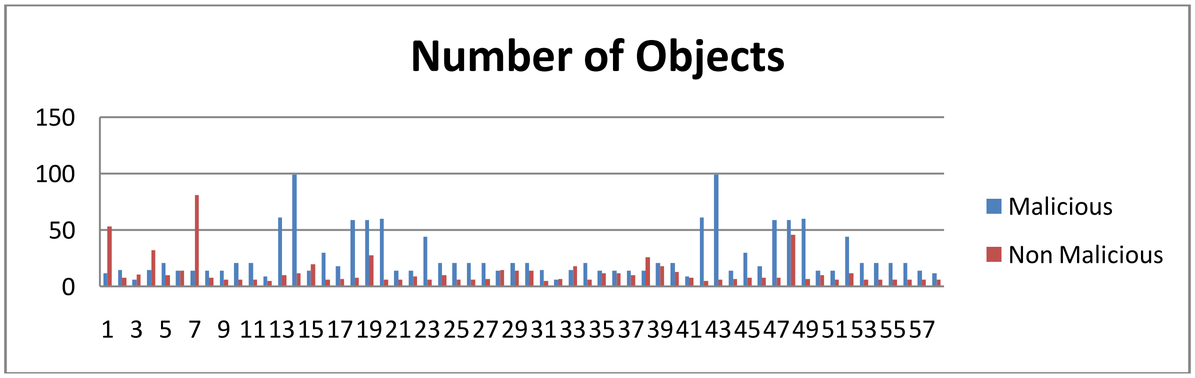


Figure 3: Comparison of number of Objects

X-References

The next feature is the X-Reference, the x-reference table contains the references to all the objects in the document. The purpose of a cross reference table is that it allows random access to objects in the file, so we don't need to read the whole PDF document to locate the particular object. Usually x-references do not make any differences when compared in context of malicious and non malicious documents. Therefore, they alone can never help the analysts to decide whether the document under analysis is malicious or not. This can also be noticed in the graph mentioned below:

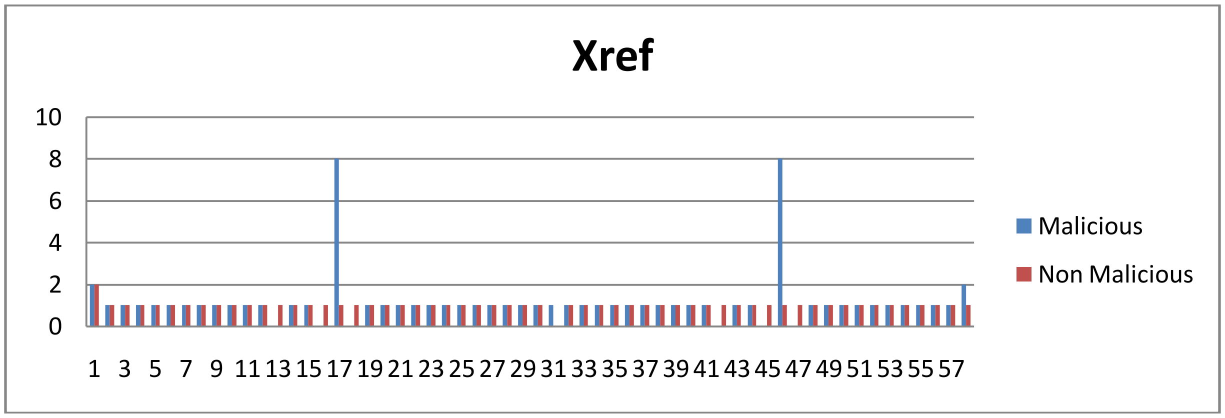


Figure 4: Comparison of number of Xrefernces

Indirect Objects

Any object inside a PDF file can be stated as an indirect object that grants the object a unique object identifier, which can be used by other objects to reference the indirect object. An indirect object is a sequenced object denoted with keywords **obj** and **endobj**. The endobj must be present in the same line, whereas the obj must encounter at the end of object ID line, which is the first line of indirect object.

Considering indirect objects in context of malicious PDF documents, the results found were same as seen in the simple objects discussed earlier section. The indirect objects can also be used to deliver the malware to the victim's machine if proper analysis of the document is not done. More the number of Indirect objects more are the chance that the file may contain suspicious content. Therefore, the graph shows the comparison between the malicious and non malicious documents in context of indirect objects.

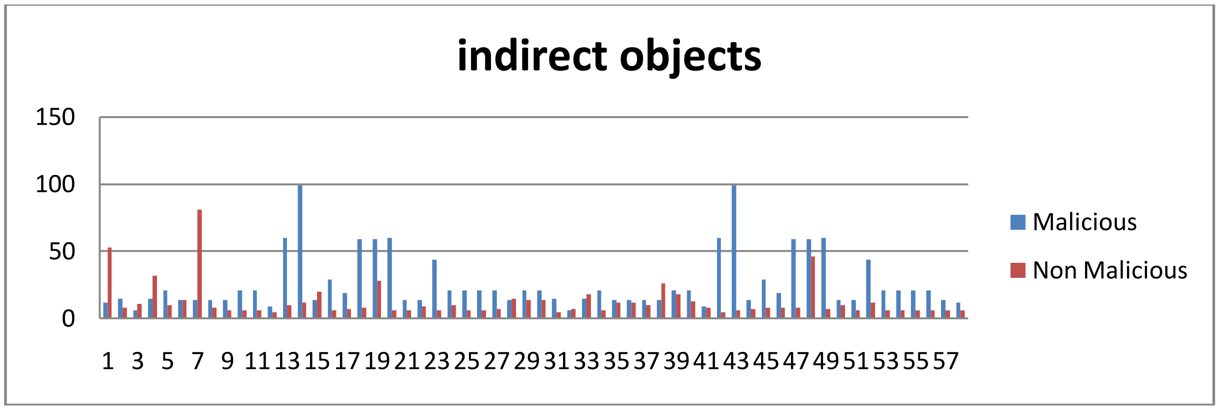


Figure 5: Comparison of number of Indirect Objects

Streams

A stream object is denoted by a sequence of bytes and its length can be limitless, that is why graphics, images and other big data blocks are often represented as streams. A stream object is specified by a dictionary object which is followed by the keywords **stream** and is further followed by **endstream**. Streams may also be subjected to filters that compress or encode the data within them. Streams are capable of hiding the scripts with exploits, shell code or they can even contain data or image inside them which is a great advantage to the attackers to embed the malicious content inside a PDF document. As we can clearly observe in the graph below the number of streams in benign documents is very low as compared to malicious ones.

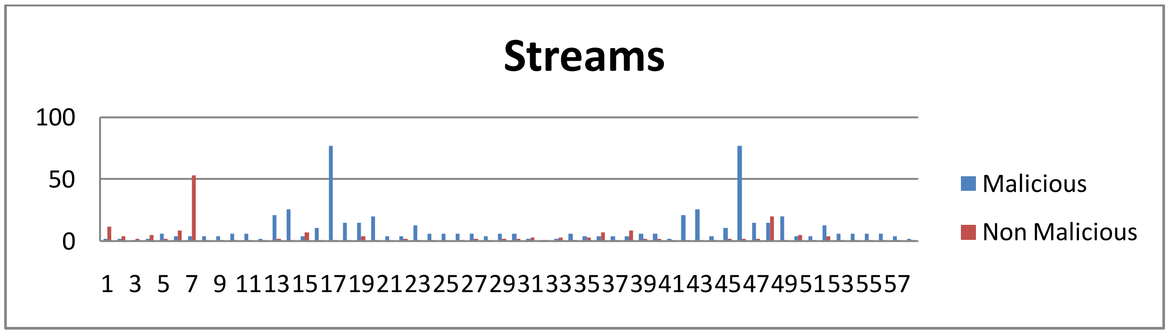


Figure 6: Comparison of number of Streams

JavaScripts

Through the support of JavaScript in the PDF file specification. The attackers know how to utilize the power of a scripting language like JavaScript in performing various attacks. JavaScript is employed to exploit vulnerabilities in PDF JavaScript API and to brim over the PDF reader's memory with malicious content, with the help of a technique called heap spray1ng. An exploit often comprise of a code that first heap sprays the reader's memory with a shellcode, and later calls a vulnerable function, executing the shellcode. Today most of the attacks in PDFs are carried out using JavaScript, as it is easier to inject the executable code and also the code embedded is hard to detect. The observations regarding JavaScript clearly show that the Number of JavaScript is way too high in malicious ones and the non malicious documents barely contain JavaScript embedded in them.

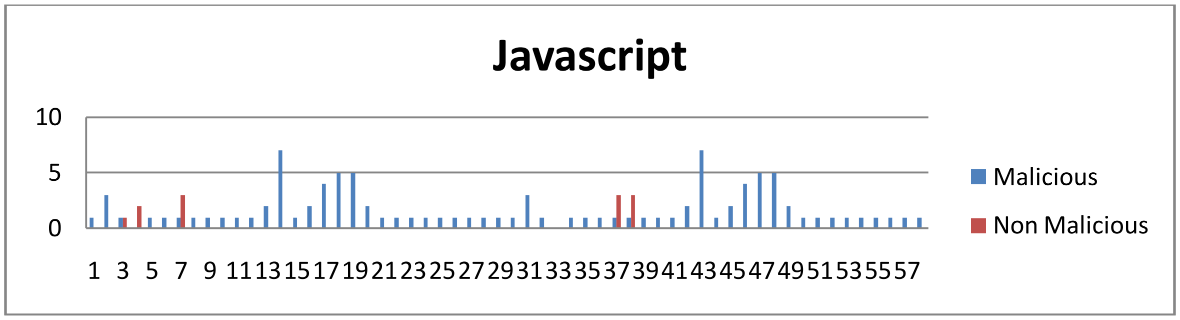


Figure 7: Comparison of number of JavaScript

Object Streams

Object streams are a very useful feature included in the PDF specification which introduces a new type of PDF object. Until they came, PDF objects contained a binary part (which could be compressed) and a text header (which could not). Object Streams allows one to put a lot of PDF objects together inside a single binary stream. The binary stream has a text header, guiding the PDF parser how to locate and extract the PDF objects, but all the PDF objects can be compressed. This makes the PDF smaller in size, potentially more secure and faster to load. But for the analysis of both malicious and non malicious files, this feature does not help too much as they are very few in number, which can be clearly observed in the graph mentioned below.

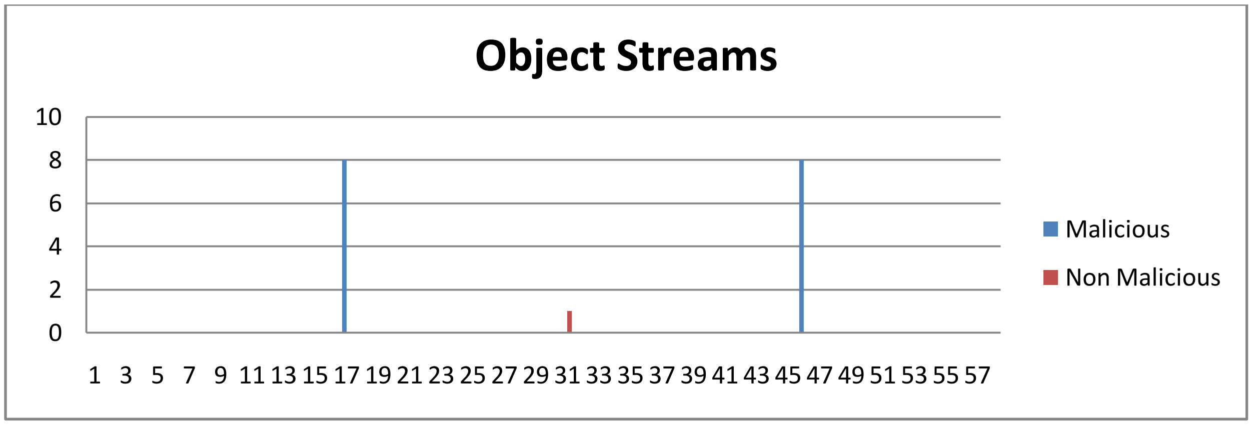


Figure 8: Comparison of number of Object Streams

While these features mentioned above may not be of great use when used individually, but they give a good overview of the whole PDF document structure when used all together. For example, the malicious PDF files and their number of objects or streams are usually smaller, in terms of size, than the non malicious ones. This is acceptable, as the malicious PDF documents do not normally contain text. The smaller is the document size, the smaller is the time taken to infect the victims. The number of *versions* is usually higher than 1 in non malicious document, as a recent version is typically created when a user directly alters or expands a PDF document. Malicious files often possess a higher number of JavaScript objects as compared to non malicious ones. This is because various exploits are implemented by *combining* multiple JavaScript codes in order to create the whole attack code. Conclusively, the *objects and X-ref streams* are often employed to hide the malicious objects inside the document and *compressed objects* may include embedded data, such as scripting code or other EXE or PDF files.

6. Results over weka Classifiers

We employed the pdfid.py from Stevens to start with extraction of various features from the PDF file and then to utilize the machine learning algorithms, we used Weka implementation of various algorithms and also analyzed the documents manually. For that a series of Python scripts, parsed the PDF documents and extracted the features mentioned in earlier sections. From there, we used those scripts to store the output (extracted features) in .CSV format. And later, we analyzed those features for both Malicious and Non Malicious Documents in weka over various classifiers by their true positives and false positives.

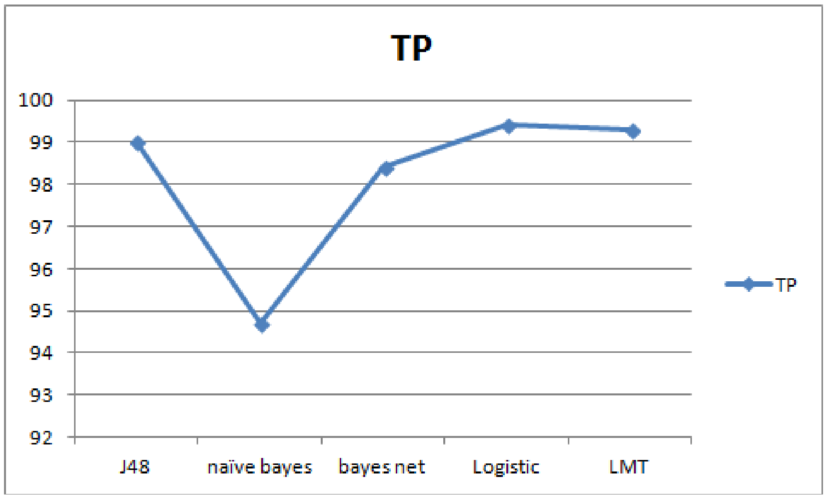


Figure 9: True Positive Rate of Various Classifiers

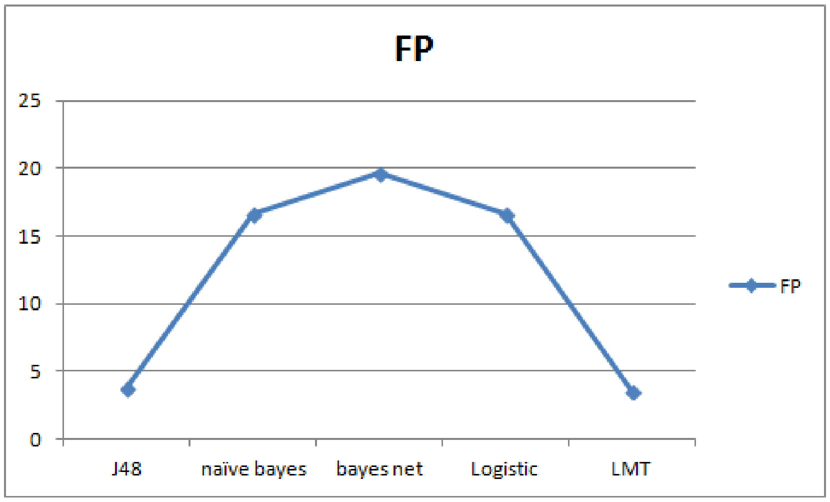


Figure 10: False Positive Rate of Various Classifiers

From the above graphical results, we observed that the true positive (TP) and false positive (FP) for each classifier. If we see the true positives, we found that the logistic classifier provided better results as compared to others and for the false positives LMT performed slightly better than J48 but both were better than Naive Bayes and Bayes Net.

7. Conclusion

Malicious PDF documents have emerged as a serious threat in the past few years. PDFs still compose a very efficient attack vector for the attackers or the adversaries, as the readers available to render them are often vulnerable to zero day attacks. In spite of all the detection mechanisms that have been established during the past years, the research has disclosed that how the PDFs are crafted so that an attacker can easily evade the highly advanced detection systems. These results show that the feature set is an effective method to analyze the malicious PDF documents for classification and analysis. Such work has further applications in identifying malicious PDF documents from the non malicious PDF documents.

8. References

[1] J. Gantz and D. Reinsel, “Big data, bigger digital shadows, and biggest growth in the far east," DC iView: IDC Analyze the Future, 2012.

[2] A. C. Madrigal, “Flash and the PDF: Computing's last great and now endangered monopolies," The Atlantic, Apr. 2012.[Online]. Available: http://www.theatlantic.com/technology/archive/2012/04/ash-and-the-pdf-computings-last-great-and-now-endangered-monopolies/255403/

[3] J.Yonts, “PDF malware overview," SANS, Tech. Rep., Jul. 2010.[Online]. Available:http://www.sans.org/security-resources/malwarefaq/pdf-overview.php

[4] D. Waltermire and K. A. Scarfone, “Sp 800-51: Guide to using vulnerability naming schemes," Recommendations of the National Institute of Standards and Technology, 2011.

[5] D. Danchev. Report: Malicious PDF file comprised 80 percent of all exploits for 2009 ZDNet. 2009 [Online]. Available:http://www.zdnet.com/blog/security/ report-malicious-pdf-file-comprised-80-percent-of-all-exploits-for-2009/5473

[6] Selvaraj, K. & Gutierrez, N. F. The rise of pdf malware. Technical report, Symantec, 2010.

[7] Horst, J., et al. "Information required for dimensional measurement." *37th International Matador Conference, S. Hinduja, and L. Li, eds., Springer, Manchester*. 2012.

[8] Li, W.J., Stolfo, S., Stavrou, A., Androulaki, E., Keromytis, A.D.: A study of malcode bearing documents. In: Proceedings of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. 2007.

[9] Shafiq, M.Z., Khayam, S.A., Farooq, M.: Embedded malware detection using markov n-grams. In: Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. 2008.

[10] Tabish, S.M., Shafiq, M.Z., Farooq, M.: Malware detection using statistical analysis of byte-level file content. In: Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics. 2009.

[11] C. Smutz and A. Stavrou,“Malicious PDF detection using metadata and structural features," Proceedings of the 28th Annual Computer Security Applications Conference, pp. 239-248, Dec. 2012.

[12] C. Vatamanu, D. Gavrilut, and R. Benchea, “A practical approach on clustering malicious PDF documents," Journal in Computer Virology, vol. 8, no. 4, pp. 151-163. 2012.

[13] J. Cross and A. Munson, “Deep PDF parsing to extract features for detecting embedded malware," Sandia National Labs, Albuquerque, New Mexico, Unlimited Release SAND 2011-7982.

[14] Maiorca, D., Corona, I., Giacinto, G.: Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious pdf files detection. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. 2013.

[15] M. Cova, C. Kruegel and G.Vigna.“Detection and analysis of drive-by-download attacks and malicious JavaScript code" Proceedings of the International World Wide Web Conference. 2011.

[16] blog.zeltser.com/post/3235995383/pdf-stream-dumper-malicious-file-analysis Didier Stevens, PDF Tools. "pdfid.py", Available at: h*ttp://blog.didierstevens.com/programs/pdftools/*

[17] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. 2009. The weka data mining software: An update. In *SIGKDD Explorations*, volume 11. 2009.