A Pattern Recognition System for Malicious PDF Files Detection

Davide Maiorca, Giorgio Giacinto, and Igino Corona

Department of Electrical and Electronic Engineering (DIEE), University of Cagliari,
Piazza d'Armi 09123, Cagliari, Italy
tnemesis@tin.it, {giacinto,igino.corona}@diee.unica.it

Abstract. Malicious PDF files have been used to harm computer security during the past two-three years, and modern antivirus are proving to be not completely effective against this kind of threat. In this paper an innovative technique, which combines a feature extractor module strongly related to the structure of PDF files and an effective classifier, is presented. This system has proven to be more effective than other state-of-the-art research tools for malicious PDF detection, as well as than most of antivirus in commerce. Moreover, its flexibility allows adopting it either as a stand-alone tool or as plug-in to improve the performance of an already installed antivirus.

1 Introduction

The ways in which hackers try to violate the security of computer systems have been constantly evolving. Operating systems have become more secure, as security fixes are constantly released, and the possibility of finding Zero-Day Vulnerabilities[1] is reduced. Therefore, third parties applications (such as Adobe Reader, Microsoft Outlook, etc.) and the file formats they read have become the most targeted ones by the attackers.

The PDF file format is nowadays widely used to read documents, and it is common to think that it is safe. However, its security has been harmed during the past years. The applications that are commonly used to open them have been targeted by cyber-criminals, who have been trying to discover bugs or vulnerabilities that might allow them to gain control of the computer systems used to read a PDF file. Moreover, the spectrum of the attacks is widened by the presence of third-party plugins connected to such applications, which often suffer from bugs that, although discovered, are not patched on time. Once these systems have been exploited, they might also be used by cyber-criminal organizations as part of botnets. This problem has been clearly reported by Symantec and IBM in their 2009 and 2010 security reports [2,3].

Attackers have also become smarter and many countermeasures established by software houses such as Adobe are now bypassed. Most of the attacks focus on bypassing the most advanced protections, so the development of a system

[1] Vulnerabilities discovered and not yet patched.

P. Perner (Ed.): MLDM 2012, LNAI 7376, pp. 510–524, 2012.

which can be robust against the widest variety of attacks (including possible new ones) will be crucial to address this threat.

In this paper, we present a new tool for the detection of malicious PDF files, where PDF-specific features are employed to build a statistical classifier through machine learning. This paper presents six sections beyond this one. Section 2 provides a basic description of the PDF file format structure. Section 3 presents a basic approach to the most important attacks that harm PDF files and readers. Section 4 is a list of the previous works and tools about PDF security. Section 5 presents the method we have adopted to develop our tool. Section 6 provides the results and the performance of our tool, as well as a comparison with the most important antivirus on the market and with Wepawet [1] and PJScan [17], a powerful tool academically developed. Section 7 closes the paper with the conclusions.

2 An Overview of PDF Technology

2.1 A Brief History

PDF is the acronym of Portable Document format, and it is a widely used standard to exchange documents. Firstly introduced in 1990, it became available for public domain in 1993; in 1994, Adobe Reader, the software used to read PDF files, became free. Nowadays, PDF is one of the most used formats to read and visualize documents, along with DOC (DOCument) and ODT (OpenDocumenT), respectively used by Microsoft Word and Open Office. There is a good reason for this, as the PDF format is flexible, allows for high typesetting quality with relatively small memory usage, and it is recognized among different software platforms and applications.

2.2 PDF Structure

A PDF file is structured as a sequence of dictionary objects (marked by "<<"and ">>"), logically connected to each other: each object can be followed by a compressed stream of data. Each dictionary object contains simpler types of objects (number, array, names), which provide information about the actions performed by the object itself. The stream of data can contain text, images, codes that are processed using the information provided by the objects in the dictionary. Objects can be pages, fonts, images, embedded code. Figure 1 shows an example of this structure, obtained with the PDF CanOpener, an Acrobat plug-in by WindJack Solutions [4].

We will not describe the details of the objects in the picture, as it is not the purpose of this paper. However, it is possible to visualize the same structure in a textual way (RAW mode), in which the information on the type of actions performed by the object are represented by keywords, which are identified by the tag "/".

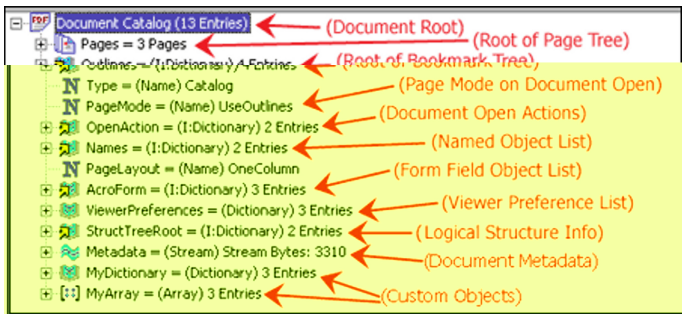

Fig. 1. A graphical representation of a PDF document structure

3 Attacks against PDF Documents

3.1 Attack Types

Due to its flexible object structure, the PDF file format can host Java, Flash code, and a variety of other applications. However, this can bring to a lot of security issues, that may support several types of attacks [5]. In the following we summarize the main security issues.

- Javascript code issue: some malicious Javascript code can be directly injected by the attacker using specific APIs (Application Program Interface) inside a PDF file, in order to exploit a vulnerability of the application that is supposed to read it. An example is reported in the CVE-2009-4324.[2]
- Launch actions: a PDF file may be crafted in order to launch special commands on the operating system, usually after a user confirmation (popup message). A critical example is reported in the CVE-2010-2883.
- Embedded files: a PDF file may contain attached files, which can be extracted and opened by the reader. This may be used to hide malicious executables. See, for instance, the CVE-2010-1240.
- Embedded Flash applications: a PDF file may contain Flash applications (stored as embedded SWF files), as well as malicious ActionScript code. See, for instance, the zero day CVE-2010-3654.

3.2 Evasion Techniques

Attackers often employ obfuscation and encryption techniques in order to bypass IDS and Antivirus. Here is a short description of the most important ones.

[2] CVE stands for Common Vulnerabilities and Exposures and it is a dictionary which classifies the discovered vulnerabilities. It was developed by MITRE, a non-profit organization featuring more than 7000 engineers who work in Applied Systems Engineering and Advanced Technology, and it is considered a vulnerability classification standard.

- GoToEmbedded actions: a PDF file can be embedded inside another PDF file, and a special command can be issued so that Adobe Reader automatically opens the embedded PDF file without notifying the user. This feature may be used to hide a malicious PDF file within a normal PDF file.
- Encryption: a PDF file may be encrypted with a password. However, if an empty password is used, Adobe Reader will open it directly without asking the user.
- Parser "flexibility": it is possible to introduce slight variations on the header of the PDF file in order to bypass too strict antivirus.

3.3 Typical Attack Procedure

Understanding how an attack is typically performed is important to the purposes of our analysis. We will now provide a typical sequence of steps that are adopted when an attack occurs, but we will not go into the details of the exploiting techniques adopted, as it is not the aim of this paper.

1. Opening a malicious PDF file, usually sent by spam. The malicious PDF file could contain SWF, HTML and JS files, Javascript and ActionScript code, and even embedded malicious PDF files. These elements are usually obfuscated in order not to be detected by IDS.
2. Depending on the type code/file embedded, some exploiting techniques are adopted, such as:
   - Buffer Overflow, in which malicious code is inserted in memory areas outside the bounds allowed for the program. [6]
   - Return Oriented Programming (ROP), in which the flow of the program is redirected to a memory area containing the malicious code, using a specific memory address. [7]
   - Heap Spraying (HS), in which the heap area of the memory is filled with multiple copies of an object containing malicious code, in order to bypass some countermeasures against ROP. [8]
   - JIT Spraying (JITS), in which Just In Time (JIT) compilers are used to make writable areas of the memory that should be not writable, and then to inject malicious code into those area. [9]
3. Thanks to the exploit, shellcode embedded inside the PDF file is copied in main memory and executed.
4. The shellcode triggers the download of a "trojan horse"program and automatically executes it, compromising the whole security of the computer system.

4 Related Works on PDF Security

4.1 General PDF Security Research

PDF security is a rather new field of research. Many researchers have been working in order to discover new threats and to produce some tools that can

improve the quality of the PDF analysis. Stevens developed PDFid, a parser that has been used in our experiments [10]: it is able to provide a detailed list of the objects contained in a PDF file, as well as their frequencies (i.e. how many times they appear inside a file). An impressive insight into the vulnerabilities of the PDF files have also been provided by Contagio [11], which also gave us part of the dataset used in our analysis. New tricks and ways of exploiting Adobe vulnerabilities were also described by Cova [12]. Dixon [13] is also helping the scene, providing new tools for malware analysis and a very useful database of malicious files found in the net, along with their characteristics and statistics of the most recurring objects.

4.2 State-of-the-Art Malicious PDF Detection Tools

There are not many academic tools specifically related to PDF security. Most of the available tools detect many types of malware at the same time, and some include PDF as well. We will now provide a brief description of the most important ones.

CWSandbox. This tool has represented an important breakthrough in malware analysis. CWSandbox [14] is a sophisticated platform capable to extract the dynamic behavior of a computer system once a certain (e.g. suspicious) file is opened and executed. Files are executed in a controlled (virtual) environment and a detailed report of the raised operating system events is built. While this tool is not able to tell whether a file is is malicious or not, its reports can be used for manual analysis or even to generate a set of features for automatic classification.

Wepawet. One of the first academic tools specifically designed to detect PDF files is Wepawet [15][1], an online malware detection system that detects malicious URL and PDF files. It extends the approach introduced by CWSandbox by including a features extraction and classification system. It is a machine learning tool which focuses on Javascript attacks: it extracts, deobfuscates and classifies Javascript code within PDF files. To this end, the tool analyses specific commands associated to malicious files, as well as the order in which those commands are executed. The tool employs a Bayesian classifier, which appears to be good for the purposes of the analysis.

Nozzle. It is a specific tool developed to detect Heap Spraying attacks [8]. Although it has not been specifically designed to detect malicious PDF files, it can be a very useful resource, as many PDF files implement HS attacks.

MDScan. This is one of the most recent and advanced tools created, and it was specifically designed to detect malicious Javascript code inside a PDF file [16]. Basically, it implements a hybrid approach: first, it scans the PDF document in order to retrieve Javascript code, i.e. it searches for the objects related to Javascript routines (static part), and then it executes the code using a Javascript interpreter (dynamic part). The main difference from Wepawet is the way the

malware files are classified: MDScan analyzes the part of the memory in which the Javascript routines are written, and heuristics are adopted to determine whether or not the code is malicious.

PJScan. This tool has been recently developed by Laskov and Šrndić, and it extracts the features used for the classification from the Javascript code embedded in the PDF file, using a static N-gram analysis. It then uses a one-class SVM to classify the files. This tool only analyzes files that have embedded Javascript code [17].

Due to their public availability and the possibility of performing massive scans, we have been able to compare the performance of our tool with Wepawet and PJScan.

5 A New PDF Detector

In this paper a new tool called PDF Malware Slayer (PDFMS) is presented. PDFMS is an advanced tool to the detection of malicious PDF files, based on machine learning. This tool is composed by:

- A data retrieval module, which retrieves the files for the training/testing phases.
- A feature extractor module, which determines the type of features used by the classifier.
- The classifier itself.

We will now focus on the methodology adopted to develop the feature extractor module, as well as the guidelines behind the choice of the classifier. The data retrieval module will be analyzed in section 6.1.

5.1 Feature Extraction

This is the most important phase of the project, as an incorrect choice of the features would not make the classifier work well. Malicious code is always contained inside a data stream, which is compressed. However, analyzing data stream as a whole can be quite complex, due to wide variety of PDF objects. Moreover, focusing on a particular kind of object (e.g. Javascript or ActionScript) may allow to detect only a portion of PDF attacks. To overcome this problem, we characterize PDF files according to the set of embedded keywords. It is worth noting that a PDF reader needs to recognize some specific PDF keywords in order to execute actions, opening images, and so on. Thus, the occurrence of each keyword can be useful to understand the high-level behavior of PDF readers when a PDF file is opened, and discriminate between malicious and legitimate PDFs.

Towards this goal, we perform two steps:

1. Let us consider two sets, composed by malicious or legitimate PDFs, respectively. For each of these two sets, we enumerate PDF keywords and, for each keyword, its relative frequency is computed (multiple occurrences of a keyword within a single file are considered only once). Then, for either malicious or benign files, we identify the group of keywords having the highest frequency (i.e. highest centroid) by means of a K-Means clustering with $K = 2$. The *base* feature set is defined by the union of the corresponding two clusters. For each keyword within this set, we add its obfuscated version (if it appears at least once on benign or malicious files) to build the final feature set. For instance, if keyword /JS is within the *base* feature set, and there is a obfuscated version of this keyword, namely /JSoffus, we will include /JSoffus in the final feature set.
2. The feature vector for each PDF file is obtained by computing the frequency of each keyword in the feature set.

Fig. 2 shows the structure of the feature extractor.

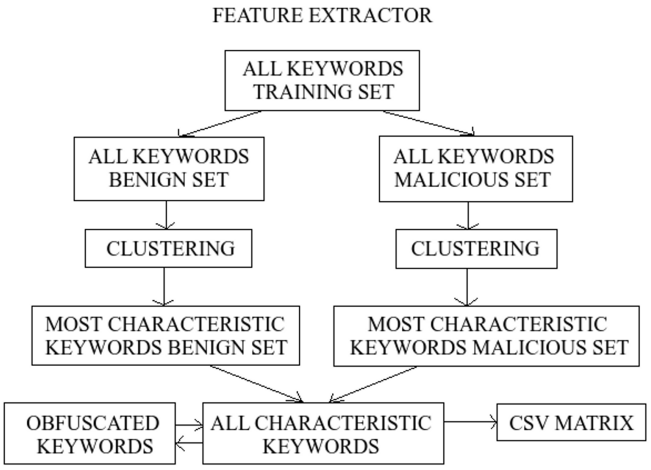

Fig. 2. PDFMS Feature Extractor

5.2 Classification

The type of features chosen for this problem does not provide particular hints for the choice of a classification algorithm. During the experimental phase, different classifiers such as Naive Bayes, SVM and Decision Trees (in particular using Random Forests), have been considered. After having determined which classifier has the best accuracy and stability on the training model, we have performed a comparison between the accuracy of our system on the test set and the one of other systems, both commercial and academic. See the next section for more information.

6 Results

6.1 Data Collection

Basically, both malicious and benign files are used for the training phase. This choice was made because there are not only differences between malicious and benign PDF files, but also common points. Hence, using both classes for training could be a reasonable choice. The data has been retrieved from:

- An interface to the Yahoo search engine [18], which randomly downloaded PDF files using random keys from a dictionary (benign source).
- A huge dataset provided by the Contagio team [11] (benign and malicious files). This choice was made because finding a lot of malicious samples on search engines is a very difficult task, as most of PDF malicious files are sent using spam[3]. Therefore, using the Yahoo Search Engine to retrieve malicious files would not bring good results.

The dataset obtained in this way was divided into a training set and a test set. The division was made randomly, although using two basic criteria:

- The number of the benign and malicious files in the training set must be the same, in order to attain a balanced training set.
- The number of the test set samples must be high, in order to provide reliable results. Hence, we decided to use a number of test samples close to the number of the training ones.

Fig. 3 shows the structure of the data retrieval module.

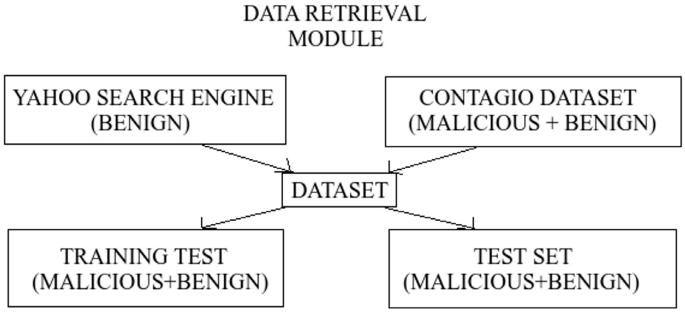

Fig. 3. PDFMS Data Retrieval Module

The 25% of the dataset has been obtained from the Yahoo Search Engine, whilst the other 75% from the Contagio dataset. The total number of files is

[3] With the term SPAM we define advertisements sent through e-mail, blogs or search engines. In particular, an attack performed using SPAM usually requires the user to download a file related to the advertisement and open it (most of times it is an executable one, but also PDF ones).

21146: 11157 malicious files and 9989 benign ones. The training set is composed by 6000 benign plus 6000 malicious files, whilst the remaining samples formed the test set. The malicious files in the training set contain a high variety of attacks (mainly Javascript and ActionScript embedded code), which implement all the techniques described in Section 3.3. The chosen division brought to 12000 files for the training set and 9146 files for the test set, formed by 3989 benign files and 5157 malicious ones.

6.2 Feature Extraction

The number of objects determined from the malicious part of the training set is 10354, and the highest total frequency of a keyword appearing in the malicious set is 5945. Fig. 4 shows the results of the clustering performed on the malicious part. On the y axis the frequency of the object is reported.

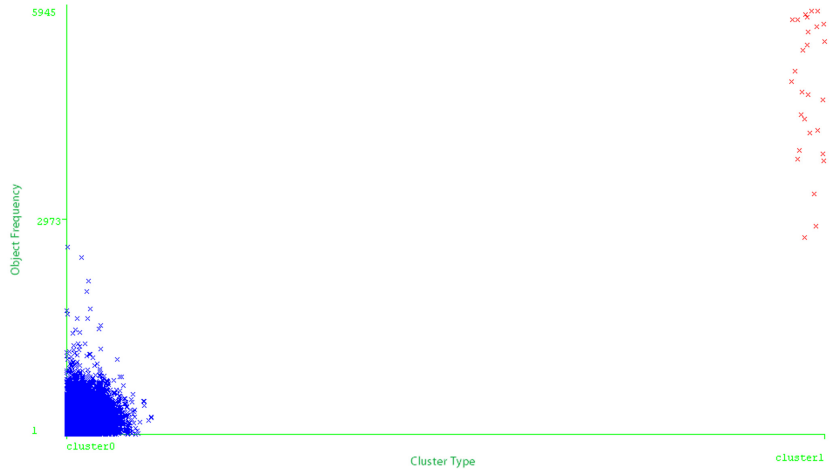

Fig. 4. Malicious training set clustering

As it can be seen, the first 28 objects having higher frequency value (i.e. with the highest occurrence) were considered characteristic by the clusterer.

The number of the objects determined from the benign part of the training set is 650357, and the highest number of occurrences for a single object is equal to 5965. The difference between the malicious part is evident, as genuine files contain a wider variety of objects compared to the malicious ones. Fig. 5 shows the clustering plot for the benign files.

The 156 objects with the highest frequency were considered characteristic by the clusterer (red cluster)[4]. Finally, the objects obtained from the cluster related

[4] One of the last red points seems a bit under the blue one, but this is just because of the jitter, i.e. a random noise introduced to separate the points in order to provide a better visualization of the clusters. With zero jitter the instances are perfectly ordered.

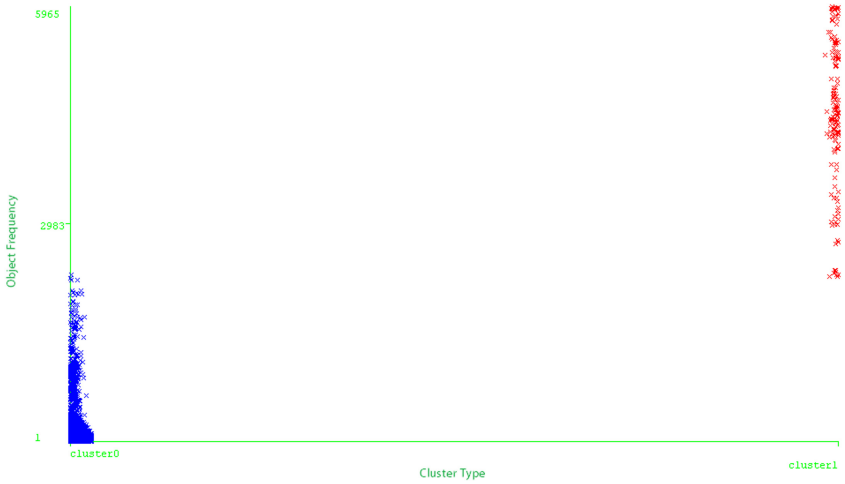

Fig. 5. Benign training set clustering

to the benign files were merged with the ones obtained from the cluster related to the malicious files: this operation returned 168 objects. Considering the presence of the obfuscated instances of the objects considered, the final number of features is 243, as 75 objects out of 168 appear at least once in their obfuscated form.

6.3 Choice of the Classifier

We have performed our tests on Bayesian, SVM, J48 and Random Forests, using a 10-folds Cross Validation repeated 10 times and a split-test (66% training, 34% test) repeated 50 times. Table 1 shows the accuracy (in percentage) attained by different classifiers. The standard deviation is reported between brackets.

Table 1. A comparison of the best classifiers with a 10-folds Cross Validation and a split test

Classifiers	Cross Validation	Split Test (66% training)
Complement Naive Bayes	98.65 (0.32)	98.63 (0.15)
SVM Linear Kernel	99.39 (0.25)	99.29 (0.14)
J48 Decision Tree ($C = 0.25$)	99.59 (0.22)	99.57 (0.14)
Random Forests (18 trees, 165 features)	**99.88 (0.1)**	**99.82 (0.07)**

Random forests provided the highest accuracy. Its accuracy is significantly better than the one provided by other classifiers. This has been proved with a paired T-test with a significance of 0.05.

6.4 Accuracy on the Test Set

After the cross-validation phase, we have analyzed the accuracy of the classifiers on the test set. We have compared the accuracy provided by the proposed PDFMS technique with the 20 most effective antivirus in commerce[5]. The analysis of the test set using these antivirus was made using Virus Total [19], a service which provides the virus scan results from more than 40 antivirus. Table 2 shows the attained results. This table shows also a comparison between PDFMS, Wepawet and PJScan: this comparison is interesting because both tools were developed in an academic environment.

Table 2. A comparison of PDF Malware Slayer accuracy with the other antivirus in commerce

Antivirus	Test Set False Positives	Test Set False Negatives	Total Score
PcTools	0 (0%)	10 (0.19391%)	10
PDF Malware Slayer	**1 (0.02507%)**	**23 (0.446%)**	**24**
GData	33 (0.82728%)	8 (0.15513%)	41
Kaspersky	8 (0.20055%)	35 (0.67869%)	43
Nprotect	6 (0.15041%)	47 (0.91138%)	53
Bitdefender	9 (0.22562%)	49 (0.95017%)	58
Avast5	2 (0.05014%)	59 (1.14408%)	61
Symantec	79 (0.7714%)	31 (1.5319%)	110
Sophos	1 (0.02501%)	134 (2.59841%)	135
ClamAV	26 (0.65179%)	140 (2.71476%)	166
NOD32	4 (0.02501%)	177 (3.43223%)	181
Commtouch	24 (0.60155%)	207 (4.01396%)	231
McAfee-GW-Edition	6 (0.15041%)	259 (5.0223%)	265
Ikarus	1 (0.02507%)	272 (5.27438%)	273
Microsoft	2 (0.05013%)	290 (5.62342%)	292
AVG	32 (0.08022%)	299 (5.79795%)	331
F-Prot	1 (0.02507%)	343 (6.65115%)	344
eTrust-Vet	0 (0%)	385 (7.46558)%)	385
VIPRE	24 (0.60165%)	406 (7.87279%)	430
Fortinet	0 (0%)	530 (10.27773%)	530
Norman	22 (0.55152%)	648 (12.56545%)	670
PJScan	0 (0% out of 25)	483 (11.14958% out of 4332)	483
Wepawet	0 (0% out of 1565)	4664 (90.84534% out of 5134)	4664

Table 3. AND Logic when using PDFMS as a plugin

	PDFMSBenign	PDFMSMalicious
PcToolsBenign	Benign	**Malicious**
PcToolsMalicious	**Malicious**	**Malicious**

[5] Most effective means that have produced the highest accuracy on the test set.

Table 4. Performance improvements using PDFMS as a plugin

Antivirus	Test Set False Positives	Test Set False Negatives	Total Score
PDFMS+PcTools	**1**	**3**	**4**
PcTools	0	10	10
PDF Malware Slayer	1	23	24

Test set False Positives is the number of benign files detected as malicious and Test set False Negatives is the number of malicious files detected as benign; in brackets we provide their percentage values. Total Score is a value given by the sum of false positives and false negatives, and it is used as an index of the total number of errors made by the classifier. As it can be seen, PDFMS performs exceptionally well, since its accuracy is just slightly below the most accurate tool (PCTools antivirus). It performs much better than Wepawet, although our analysis was conditioned by upload errors, probably due to a server overload, and we did not have the possibility to use an offline version of the tool. It also performs better than PJScan, which could not analyze all the test set files because of the internal structure of the tool (see section 4). For these two tools, we have therefore reported only the results related to the files correctly analyzed. To have another interesting idea of the good performance of our PDFMS tool, Fig. 6 shows the ROC chart related to the test set analysis (the Wepawet and PJScan percentage results are related to the correctly analyzed files). On the y axis the percentage of true negatives is reported, whilst on the x axis the percentage of false positive. The more an antivirus stays on the upper left corner of the chart, the more it is considered effective.

Our tool was also designed not only to be used stand-alone, but also as a plug-in for existing antivirus. In particular, we have adopted an AND logic

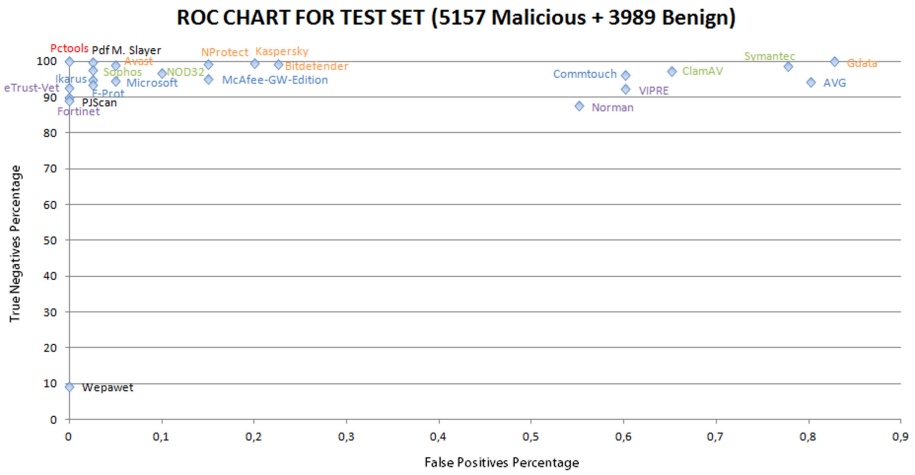

Fig. 6. ROC Chart for test set analysis

approach, in order to improve the malicious files detection rate. We have therefore combined the performance of PDFMS with the ones attained by PC TOOLS, the commercial tool that provided the highest performance.

Table 3 shows the detection mechanism. In other words, classify a sample as benign only if both tools classify it as benign. Table 4 shows the comparison between PDFMS stand-alone, PcTools stand alone and PDFMS used as a plug-in of PcTools. As it can be seen, the malicious detection rate is dramatically improved, making the score index almost perfect. However, there is a trade off, as the false positives score is slightly increased. A zoomed ROC chart (Fig. 7) referred to table 4 better shows this.

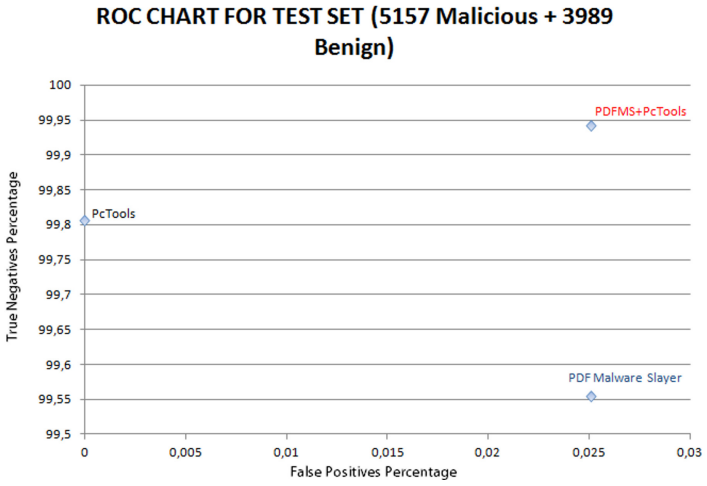

Fig. 7. Zoomed ROC Chart for PDFMS different ways of working

6.5 Weaknesses

Although the system has proved to be very effective, it has some structural weaknesses. Firstly, the same objects that are found in benign files can be also found in malicious ones, meaning that the same object can bring malicious or benign code. What allows PDFMS to correctly establish the maliciousness of the file is the value of the single frequencies of many characteristic objects inside a data stream. Consequently, while PDFMS is able to establish whether or not the PDF file is malicious, it cannot say anything about the type of vulnerability, because it doesn't analyze the code within such a stream. What's more, if an attacker learned how many times certain objects have to appear inside the file in order for it to be considered benign by the tool, it might bypass PDFMS by injecting those specific keywords inside the file. An improvement of the parser is in progress in order to avoid this kind of attack.

7 Conclusions

In this paper, a new system to detect malicious PDF files has been presented. It is a machine learning-based system, strongly related to the internal structure of the file format, easily reproducible, and effective against several types of attacks. It is also possible to use it in different ways according to user needs. The accuracy value shows that the system performs better than the vast majority of commercial antivirus. Moreover, it performs much better than Wepawet, a powerful tool academically developed. In fact, our tool is specialized on the detection of PDF attacks, while Wepawet has been developed to detect a number of threats including malicious PDF files. Thus, specialization appears to be very important to detect this kind of threat. Our tool can also scan any type of PDF file, whilst academic tools such as PJScan can analyze just PDF files carrying Javascript code. The proposed system can be further improved by testing its robustness against new vulnerabilities and improving the parsing process. This tool might also be part of a Multi Classifier System, in which every classifier is specialized in detecting specific threats. As the attacker strategies improve, the challenge for the future is making our security systems more robust against the widest variety of threats, giving them even the possibility to predict new ones.

Acknowledgments. The authors would like to thank the anonymous reviewers for useful insights and suggestions. This research has been carried out within the project "Advanced and secure sharing of multimedia data over social networks in the future Internet" funded by the Regional Administration of Sardinia, Italy (CUP F71J11000690002).

References

1. Wepawet, `http://wepawet.iseclab.org/`
2. IBM : IBM X-Force 2010 Mid-Year Trend and Risk Report (2010)
3. Symantec : Symantec Global Internet Security Threat Report. Trends for 2009 (2010)
4. Parker, T.: Navigating the Internal Structure of a PDF Document, `http://www.planetpdf.com`
5. Decalage: PDF Security Isseues, `http://www.decalage.info`
6. Ogorkiewicz, M., Frej, P.: Analysis of Buffer Overflow Attacks (2004), `http://www.windowsecurity.com`
7. Ramachandran, V.: Buffer Overflow Primer Video Series, `http://www.securitytube.net`
8. Ratanaworabhan, P., Livshits, B., Zorn, B.: NOZZLE: A Defense Against Heap-spraying Code Injection Attacks. In: SSYM 2009 Proceedings of the 18th Conference on USENIX Security Symposium (2009)
9. Bania, P.: JIT Spraying and Mitigations. CoRR (2010)
10. Stevens, D.: PDF tools, `http://blog.didierstevens.com/programs/pdf-tools/`
11. Contagio, `http://contagiodump.blogspot.com/`
12. Cova, M., `http://www.cs.bham.ac.uk/~covam/blog/pdf/`
13. Dixon, B., `http://blog.9bplus.com`

14. Willems, C., Holz, T., Freiling, F.: Toward Automated Dynamic Malware Analysis Using CWSandbox. Journal IEEE Security and Privacy Archive 5(2) (2007)
15. Cova, M., Kruegel, C., Vigna, G.: Detection and Analysis of Drive-by-Downloads Attacks and Malicious Javascript Code. In: Proceedings of International World Wide Web Conference, WWW 2010 (2010)
16. Tzermias, Z., Sykiotakis, G., Polychronakis, M., Markatos, E.P.: Combining Static and Dynamic Analysis for the Detection of Malicious Documents. In: EUROSEC 2011 Proceedings of the Fourth European Workshop on System Security (2011)
17. Laskov, P., Šrndić, N.: Static Detection of Malicious JavaScript-Bearing PDF Documents. In: Annual Computer Security Applications Conference (2011)
18. Yahoo, `http://www.yahoo.com`
19. Virus Total, `http://www.virustotal.com/`