Department of Computer Science
George Mason University Technical Reports

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/ 703-993-1530

Malicious PDF Detection Using Metadata and Structural Features

Charles Smutz
csmutz@gmu.edu

Angelos Stavrou
astavrou@gmu.edu

Technical Report GMU-CS-TR-2012-5

Abstract

Owed to their versatile functionality and size, PDF documents have become a popular avenue for exploitation ranging from broadcasted phishing attacks to targeted attacks. In this paper, we present a method for efficiently detecting malicious documents based on feature extraction of document metadata and structure. We demonstrate that by carefully extracting a wide-range of feature sets we can offer a robust malware classifier. Indeed, using multiple datasets containing an aggregate of 5,000 malicious documents and 100,000 benign ones, our classification rates are well above 99% while maintaining low false negatives 0.01% or less for different classification parameters and scenarios.

Surprisingly, we also discovered that by artificially reducing the influence of the top features we can achieve robust detection even in an adversarial setting where the attacker is aware of the top features and our normality model. Therefore, due to the large number of the extracted features, a defense posture that employs detectors trained on random sets of features, is robust even against mimicry attacks with intimate knowledge of the training set.

1 Introduction

Malicious code is often embedded or packaged within documents that appear legitimate including government forms and bank statements. Thus, malicious documents are also called trojan documents because they carry a malicious payload in a seemingly desirable document which serves as distribution mechanism for the malware. The use of documents as a vehicle for exploitation has become more popular as the exploitation of vulnerabilities in client applications, including document viewers, has increased [7, 8]. In addition, it is easier to entice users to open a malware-bearing document through social engineering. The complexity and structure of modern documents can make detection of the malicious code extremely difficult: the document offers many places where code can be confused for data and many more additional layers of obfuscation compared to a Portable Executable (PE). To make matters worse, client applications, such as document viewers and data containers, are usually the consumers of foreign code and data making them perfect targets for exploitation.

Many recent studies have shown that malicious documents are frequently used in highly socially engineered phishing attacks perpetrated by groups of highly sophisticated, persistent, and targeted attackers whose goal is espionage [16, 2]. The use of malicious documents in targeted attacks, including those perpetrated by a class of attacks called Advanced Persistent Threats (APTs), adds urgency to countering this form of malware delivery. PDF documents have become one of the most popular file formats exploited in targeted attacks [10] and new vulnerabilities continue to be used by targeted attackers [1]. The exact mechanism of delivery and exploitation employed varies widely: in some cases, the document is used to merely to exploit a vulnerability in the reader application with the document itself providing little value in terms of social engineering. For instance, some classes of web based attacks, such as those leveraging cross-site scripting, operate in this way.

In phishing attacks, the document augments the social engineering aspect of the attack. This is the case when a malicious document is attached to a phishing email. In some attacks, the document contains the complete malware payload the attacker wishes to deploy, while in others, the document only has enough code to download additional malware components [19, 23, 9, 25]. Although exploitation of the reader program can result in the viewer program hanging or crashing, potentially alerting the user of a problem, sometimes such faults remain hidden from the end-user because the reader program is used as plug-in in a larger program, such as Internet browsers [11, 5, 17, 3].

In highly targeted attacks, the exploit often involves opening a benign document that is extracted from the trojan document to mask the exploitation and enhance social engineering. Many malicious documents seem to be truly Spartan, in that they contains only malicious content without superfluous metadata or structural elements. However, some malicious documents contain extensive non-malicious metadata items and structural elements seemingly indicative of the malware author either beginning construction with an existing benign document or the author intentionally inserting non-malicious content to avoid detection. Despite the differences, malicious documents are a common vector for exploitation and delivery of malware. This study proposes an alternate mechanism for detecting malicious documents which seeks to leverage some of the restrictions the use of documents enforces to malware delivery.

There exist many approaches for detecting malicious documents. Signature matching is widely employed and is effective for detecting previously identified malware on a broad scale. Indeed, many signature matching systems, such as commodity antivirus scanners [13, 27] provide functionality specific to PDF documents so that they can be decoded to reveal malicious content and exploitation of document specific vulnerabilities. A common alternative to signature matching is dynamic analysis of documents where the behavior of the entire system or the set of programs are observed while the document is opened. Furthermore, some classes of malicious code can be exposed through deviations of file or format specifications. All of the current malicious detection mechanisms provide some benefits, but all have shortcomings. Reliable detection of malicious documents remains an open research topic.

In this paper, we explore the limits of static analysis detection mechanisms that utilize machine learning techniques on document-specific attributes to identify embedded PDF malware. Our approach addresses some of the shortcomings of existing techniques through the use of a broadly applicable mechanism to classify and characterize documents. The proposed method is vulnerability and exploit agnostic and thus seeks to remove the dependence on a priori knowledge of specific malware families and vulnerabilities while remaining computationally and operationally tractable. Moreover, we aim to seek to further classify whether malicious documents are associated with broad based, opportunistic threats or highly targeted attacks.

As part of our analysis, we show that while the use of documents as an exploitation vector can be an enabling mechanism for the attacker, it also provides additional detection opportunities. All of the data closely associated with malicious activity can be used to aid in detection, regardless of whether the data utilized for detection is inherently malicious or not. Indeed, in signature matching systems, signatures are often generated for byte sequences highly specific to known malware families, even if those byte sequences are not malicious in and of themselves. The underlying premise and intuition of our study is that malicious documents do have similarities to other malicious documents; they also have dissimilarities to benign documents, regardless of the specific vulnerability exploited or the specific malware embedded in the document. We posit that features based on document structure and metadata are adequate for reliable document classification given appropriate statistical methods are applied to these features. At a very high level, this is similar to the semantic operations of SPAM detection algorithms that use features of the email, including the textual part of an email to feed a machine learning engine to reliably classify email into SPAM or legitimate.

For the purpose of our analysis, documents are represented by their respective feature vectors extracted from statically processing the document files. These features are constrained to items derived from the document metadata, such as the number of characters in the title, or features derived from the document structure, such as the size of the images in the document. This study uses 10,000 documents from a widely available data set and 100,000 documents collected from a university network. We employed different machine learning techniques and our results show that random forests algorithm performs the best with True Positives (TP) more than 0.99 (99%) while maintaining a False positive rate of 0.0001 (0.01%) or less depending on the scenario, dataset, and thresholds used. The most important finding of our study is the effectiveness of mimicry type attacks appear to be limited. This is because the set of features we extracted provide robust detection as a set even when the top features on our training set are used as part of a mimicry attack by the attacker. By perturbating the training data to mitigate the reliance on features heavily favored by the classifier, evasion becomes much more difficult.

2 Related Work

Malicious documents have been the subject of much research over the years. In terms of static analysis, Li et al. [15] and Tabish et al. [26] demonstrated that detecting malicious Word documents through static analysis using n-grams representation for the document data and course dynamic analysis shows promise but also comes with limitations due to the size of the malcode. While the techniques and file format differ, the end goals are very similar to our work. Signature analysis has also been studied extensively [24, 22] including specifically for PDF vulnerabilities [20].

In terms of dynamic analysis, Munson and Cross [6] use an instrumented reader application and dynamic analysis to extract structural features of PDF documents to be used in a machine learning based classifier. While the high level approach is similar to what is presented

here, they were unable to demonstrate strong detection rates. The approach presented here differs in that static analysis is used to extract a much larger number of features, including metadata, without without seeking to fully decode the document through dynamic analysis. Laskov and Srndic extract, process, and classify javascript embedded in PDFs using support vector machines to achieve relatively mediocre classification rates [14]. In both these studies, an important factor identified was the ability to effectively parse PDF documents which is extremely difficult, especially in the case of malicious documents. The research presented here skirts this issue by focusing on different features and different methods of extracting those features. Tzermias et al. [27] showed that the effectiveness of existing antivirus systems against malicious PDF files is quite modest. To boost the detection, they used the combination of static and dynamic analysis to identify malicious documents with focus on specific vulnerabilities and mixed results due to the need of VM support. Our approach is agnostic to vulnerabilities and does not require any form of execution (i.e. dynamic analysis).

Although different in terms of purpose and target, the use of statistical learning to identify malicious documents can be compared to SPAM detection, a topic on which there is a large corpus of previous work. Documents are similar to email messages in that both are primarily designed for transfer of human readable text and detection is usually concerned with identifying malicious activity very early in the attack life cycle. Statistical methods, such as bayes classification of body text, have been used effectively for years [21]. More recently, progress has been made towards effective network and transport layer identification of SPAM [4, 12]. One could consider the features of network traffic used for SPAM detection to be most closely analogous to the structural features of documents used in this study.

3 Data & Feature Description

3.1 Data Types

This study focuses on classifying malicious documents, specifically Adobe PDF documents. The PDF document type was chosen because of it's ubiquitous use, availability of public data sets, the large number of recent vulnerabilities in the Adobe PDF reader, and the frequent use of PDF documents in targeted attacks.

In our experiments, PDF documents are classified as either benign or malicious with malicious being further split into two categories: opportunistic and targeted. These are abbreviated "ben", "mal", "opp", and "tar" respectively. To be classified as malicious, documents must exploit a software vulnerability and execute malicious code. For the purpose of this study, documents that contain text instructing the reader to perform malicious actions (wire money), that contain hyper links to malicious content where the user must click, etc are considered benign. While difficult to codify precisely and often requiring data outside the document itself, factors such as victim specific social engineering and correlations with other known targeted attacks are used to separate opportunistic from targeted attacks.

3.2 Data Sources

There are two primary data sources used in this study. The first is the widely available Contagio data set [18] designated for signature research and testing. This source of data sets, was selected because it contains a large number of labeled benign and malicious documents, including a relatively large number from targeted attacks. This source provides a few collections of documents. All of the PDF documents from "Collection 1: Email attachments from targeted attacks" were used as targeted malicious documents. The documents from "Collection 4: Web exploit pdf (I think they all are pdf) files" are used as malicious documents. The vast majority of the documents from Collection 4 were attributed to opportunistic threats and were used as such, with a few exceptions. There were 10 identical documents in collection 1 and collection 4: these were considered targeted. Also, an additional 22 documents were identified as targeted through manual inspection and correlation to other targeted attacks. Lastly, "Collection 5: Non-Malicious PDF Collection" was used for benign PDF examples. A total of 10,000 documents were used from this source for the training data set.

Table 1: Data Set Summary

	Training	Testing/Operational
benign (ben)	5,000	99,705
opportunistic (opp)	4,844	286
targeted (tar)	156	9
total	10,000	100,000

The second collection is taken from monitoring of the network of a large university campus. These documents were extracted from HTTP and SMTP traffic. The bulk of this collection was taken from approximately 6 days of capture. Because this data is taken from a real data capture it is termed the "operational" data set. The operational data set required labeling by the researchers to be useful for evaluation. To separate the malicious documents from the benign, a combination of 5 common virus scanners were used. The corpus was scanned with the virus scanners until signature updates ceased adding detections. The virus scanner detections continued to improve until approximately 10 days following the end of the collection. Note that no single virus scanner detected all the malicious documents. All the documents that were flagged by a single scanner were subjected to

additional virus scanning and manual analysis. Two of the twelve documents identified by a lone AV scanner as malicious were found to be benign. All of the malicious documents from the week long collection were considered opportunistic as there was no evidence to support labeling them as targeted. 9 malicious PDFs associated with targeted attacks on the same organization were added to this collection. These targeted PDFs were observed on the campus network over the span of approximately 18 months and were collected in the same manner. The addition of these targeted attacks was necessary to allow the operational data set to be used to evaluate targeted attack detection. A total of 100,000 unique documents were used from the operational data set.

In both data sets, only unique documents were used. Where sampling occurred, the sampling was random. Table 1 summarizes these data sets by displaying the number documents of each class in the two sets.

Note that the training data set includes equal parts benign and malicious documents which is desirable for training. The operational data set's ratio of benign to malicious is intended to mirror a typical operational environment and provide insight to detection rates and false positives in a real environment. The number of targeted documents is undesirably low but is the best that could be obtained given their scarcity. The operational and the training data set are completely independent. The training set was compiled months before the operational data set.

3.3 Examples of Malicious PDFs

Malicious PDF documents present a wide diversity in structure. Here are presented two distinct malicious documents with large differences in metadata and structure, despite using the same exploit: CVE-2009-4324. Some structural elements of these documents, one targeted malicious and the other opportunistic malicious, are represented in Table 2 and Table 3 respectively. All the relevant structural features are shown in Table 15 and Table 16 in Appendix A.

The targeted document is large (4MB) and was delivered via a targeted email. The votes from the classifier are 84.4% malicious and 80% targeted which would be detected with cutoff values exceeding .156 and .20 respectively. The targeted document has many structural elements and attributes, including text content and font objects, that are superfluous to successful exploitation. Indeed, even the content in these unnecessary elements, which would not be seen by the user, are inconsistent with the social engineering used in this attack. Concerning metadata, the targeted document contains PDF ID values, which is normal. However, these values are the same which indicates that this document was not modified by a conventional PDF editor, which should change the ID1 value but leave the ID0 static. Upon successful exploitation of the reader program, the shellcode decrypts and "drops" a malicious windows PE executable and a benign PDF document. The existence of benign document artifacts and the method of embedding the malicious payloads in the targeted document suggest the author constructed the document from an existing benign document or document template. Construction was likely performed without a conforming PDF editor as evidenced by PDFID metadata and invalid streams.

The opportunistic document is very small (2 KB) and was delivered through malicious web traffic. The clas-

Table 2: Example Structure: Targeted

Location	Content	Description
0003E0 0003F0 000400 000410 000420 000430	.endobj..7 0 obj ..<</Type/Font/S ubtype/Type1/Bas eFont/Helvetica/ Encoding/WinAnsi Encoding>>..endo	Font object: Helvetica.
0005A0 0005B0 0005C0 ... 0007D0 0007E0 ... 000AD0	/Action /S /Java Script /JS (..fu nction re(count, ...this.media.ne wPlayer(sgo);}.. ... .>>..endobj..9 0	Malicious javascript object. This javascript exploits CVE-2009-4324.
000FA0 000FB0 000FC0 ... 3FF920 3FF930	</Filter /FlateD ecode/Length 114 688>>stream....l ... ..endstream..end obj..xref..9 4.. (File Extracted and Decrypted): Type: PE32 executable for MS Windows (GUI) Intel 80386 32-bit Name: update.exe Compiled: Wed Dec 29 02:37:00 2010	Object containing purported compressed stream. This stream is not valid compressed data. The majority of the stream contains two encrypted payloads: a PE executable and a PDF document. The PE executable is decrypted, installed on system, and executed. This malware provides remote access trojan capabilities.

Table 3: Example Structure: Opportunistic

000130
000140
000150
000160
000179
000180
...
000690
0006A0
	`j..<</Creator 8 0 R>>..endobj..8 0 obj..<</Length 1299 /Filter /FlateDecode..>>. .stream..x..Y... ... .........bw...en dstream..endobj.`
	
	(Deflated and unescaped to reveal javascript excerpt):
	
	`d + e;}a();a();try {this.media.newPlayer(null);} catch(e) {}a();`
	
	(Further unescaped to reveal shellcode excerpt):
	
	`URLDownloadToFileA.pdfupd.exe.crash.php.http://111.gosdfsdjas.com/l.php?i=16..`
		Reference to, definition of object representing the "Creator" metadata item. This object contains a compressed stream. The compressed stream contains obfuscated javascript and which contains obfuscated shellcode.
		The javascript exploits CVE-2009-4324.
		The shellcode downloads more malicious content from the Internet.

sifier assigned a rating of 100% malicious, 100% opportunistic. This document is minimal in structure. It has only the few structural elements necessary for obfuscation and exploitation. This document has no valid optional metadata. However, an object labeled as the "Creator" metadata item houses the bulk of the malicious content in the document and comprises 70% of the document. Regardless, the "Creator" metadata item is not reported by PDF metadata extraction tools. When exploitation occurs, the necessarily small shellcode pulls additional malicious content from the Internet. Contrasting with the embedding seen in the targeted document, the streams containing the exploit code must be decoded by the vulnerable reader, so these streams are well-formed.

4 Classification Methodology

4.1 Feature Selection

We selected a large number of features to characterize PDF documents. Our main goal was to provide strong classification quality, including the ability to reliably distinguish targeted attacks from opportunistic attacks. In addition, the approach taken here seeks to be resilient to differences in threats and vulnerabilities by focusing on patterns in documents that apply broadly. Therefore most features are derived either from document metadata or document structure. See Appendix B for a complete list of features that we tested our system under.

A secondary goal was to measure the robustness of those features when the attacker was aware that the defender was using them for detection. The features are designed to eliminate reliance on specific strings or byte sequences. For example, when dealing with data that might represent artifacts of specific actors, such as the author metadata item, abstracted features, such as the number of characters in the author field, are used. Similarly, features were intentionally avoided that are tightly related to specific vulnerabilities or malware because including these features could result in strong classification for known attacks while yielding low detection rates for novel attacks. For example, features related directly to jbig2 encoded objects were removed because these features would be strongly tied to a specific vulnerability: CVE-2009-0928.

Features were identified initially through manual review of documents of each class. Then, following an iterative process, feature extraction tools and the classifier were updated to include potentially useful data. Specific features were determined to be valuable through manual inspection or through the improvement in classification accuracy they provided. As the feature selection process continued, misclassified documents were inspected to determine potential features that provide adequate discrimination for correct classification. This iteration occurred until classification rates were very high. The features selected should not be presumed to be exhaustive–no effort was made to identify all useful features. In total, 135 features were chosen for use. The complete list of features and their descriptions is found in Appendix B

As we already mentioned, most features are taken directly from observation of the document metadata or document structure, such as the number of lower case characters in the title or the number of font objects in the document. A few of the features are further refined by transformation of 1 or more elements. For example, one feature is the ratio of the number of pages to the size of the whole document.

4.2 Feature Extraction

We created our own prototypes for extracting features from PDF documents. Regular expressions are applied to the raw document to identify and extract data for further processing, if necessary. Many of the features can be derived from simple string matching reporting solely the location of the matches. Ex. count of Font objects or average length of the stream objects where the length each stream object is measured by the difference between the location of a "stream" marker and and

the next "enstream" marker. Many features required extracting specific data for further normalization or processing such as the dimensions of a box object or the number of lower case characters in the title. This software functions without significant PDF structure parsing or validation which is necessary for success in dealing with malicious or otherwise malformed documents. This "fast and loose" metadata extraction intentionally results in some inaccurate or ambiguous extracted data, but the end features are deterministic. For example, a PDF edited with incremental updates may have extracted features that report an inflated object count. Additionally, instead of trying to use the "correct" value in the case of multiple instances of a metadata item repeated in a document, other features such as the number of times the values differ are used.

The majority of the metadata items are inherently numeric. The other features are all extracted or transformed to make them numeric. There are largely no differences in how binary, discrete, and continuous data is handled, although some of each type exist.

4.3 Classification Techniques

To categorize observed documents, the features are extracted and run through a classifier generated from labeled training data. Random Forest was selected because of its effective classification capability, strong performance, and ease of use. Table 4 compares Random Forest classification performance to that of Support Vector Machine and Naive Bayes using the training data set and the default parameters for each method. These results represent the average of multiple cross validation, classifier training, and classification of observations using the the classifier. All analysis was performed using R, including the randomForest and e1071 packages.

The first step is to produce a classifier using labeled training data. This classifier consists of a set of classification trees. After features are extracted from the document, they are run through the trees, each of which provides a vote in the reported classification. The initial training process is relatively computationally expensive but once the classifier is constructed, categorizing new observations is fast.

The goal of classification here is to not only classify documents as benign or malicious, but also to differentiate opportunistic from targeted attacks. As shown in Figure 1, unclassified documents are fed through a classifier which separates benign documents from malicious documents. Those found to be malicious are fed through a second classifier that differentiates opportunistic from targeted malicious documents. While random forest supports multiple class outcomes, this dual binary classifier arrangement is used because it makes it easier to understand and compare the two classification goals individually. The two binary classifiers are tuned independently and results for each are presented individually.

In order to tune the sensitivity of the classifier, the cutoff value is adjusted. Typically, random forests operates by predicting the class which receives the most votes. Hence, the default cutoff value for a binary classifier is .5. This value can be adjusted, allowing sensitivity of the classifier to be adjusted so that the operator can select a desirable TP/FP ratio. All ROC curves presented here are created by adjusting this cutoff value during prediction.

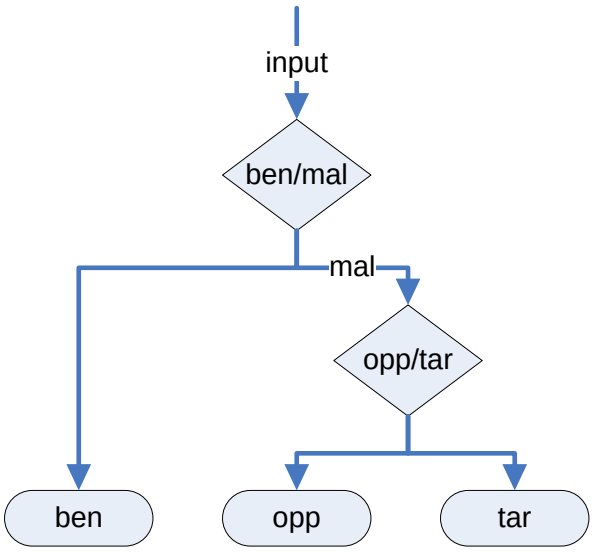

Figure 1: Dual Classifier Arrangement

4.4 Optimizing the Machine Learning Parameters

The Random Forests algorithm has two primary parameters that were tuned for this study. The parameter "ntree" dictates the number of trees to grow in the classifier. In addition, "mtry" controls the number of features sampled at each node in the trees. To properly tune these parameters, multiple values were tested with the classification error being compared. For the results shown in Table 5 and Table 6, the average of 10 independent trials of ten fold cross validation was used on the training data set.

The default values of ntree is 500 and mtry is 11. For the ben/mal classifier, values of 1000 and 33 are used. For the opp/tar classifier, values of 1250 and 22 are used.

5 Performance Evaluation

The classifier was applied to the training data set using 10-fold cross-validation. The results from each fold are averaged to produce a single outcome for the whole set. These resulting ROC curves for the ben/mal and

Table 4: Classifier Performance Comparison

Classifier	Error Rate	Train Time	Classify Time
Naive Bayes	12%	1 sec	54 sec
Random Forest	.18%	52 sec	1 sec
Support Vector Machine	19%	167 sec	27 sec

Table 5: Error Rates for ben/mal (%)

	ntree					
mtry	250	500	750	1000	1250	1500
6	0.224	0.233	0.227	0.234	0.234	0.226
11	0.180	0.176	0.172	0.178	0.172	0.177
17	0.148	0.150	0.152	0.154	0.151	0.156
22	0.144	0.133	0.143	0.134	0.134	0.136
28	0.121	0.126	0.123	0.118	0.117	0.123
33	0.126	0.117	0.116	0.110	0.111	0.117
39	0.130	0.119	0.120	0.123	0.120	0.129
44	0.134	0.134	0.134	0.130	0.128	0.130

Table 6: Error Rates for opp/tar (%)

	ntree					
mtry	250	500	750	1000	1250	1500
6	0.706	0.726	0.714	0.716	0.706	0.712
11	0.678	0.690	0.682	0.682	0.694	0.688
17	0.672	0.672	0.676	0.678	0.668	0.670
22	0.686	0.674	0.672	0.670	0.666	0.670
28	0.696	0.678	0.696	0.678	0.680	0.682
33	0.698	0.678	0.688	0.668	0.680	0.676
39	0.690	0.708	0.688	0.678	0.690	0.676
44	0.690	0.698	0.702	0.704	0.690	0.700

opp/tar classifiers are shown in Figure 2 and Figure 3 respectively.

In addition, Table 7 and Table 8 list select data points from this graph. The cutoff reported in these tables is the minimum percentage of votes as benign that an observations must exceed to be considered the negative class (ben for ben/mal, opp for opp/tar).

Table 7: FP/TP Rates for Training Set (ben/mal)

Cutoff	FP Rate	TP Rate	FP Count	TP Count
0.2	0	0.991	0	4955
0.3	0	0.9938	0	4969
0.4	0.0002	0.9966	1	4983
0.5	0.0006	0.9982	3	4991
0.6	0.0014	0.9988	7	4994
0.7	0.003	0.999	15	4995
0.8	0.0082	0.9992	41	4996

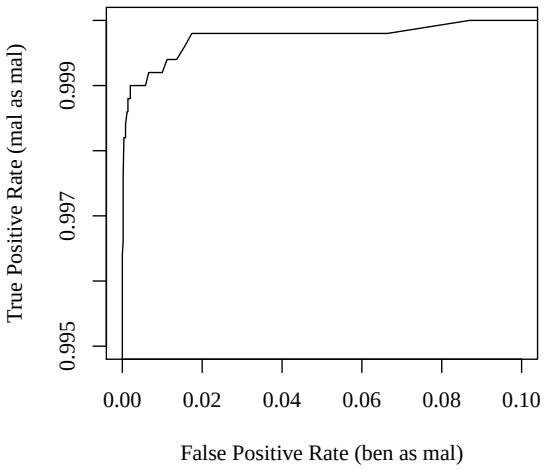

Figure 2: ROC for Training Set(ben/mal)

Table 8: FP/TP Rates for Training Set (opp/tar)

Cutoff	FP Rate	TP Rate	FP Count	TP Count
0.2	0.00082	0.6913	4	108
0.3	0.00144	0.7933	7	124
0.4	0.00248	0.8512	12	133
0.5	0.00351	0.8838	17	138
0.6	0.00578	0.9225	28	144
0.7	0.00825	0.9421	40	147
0.8	0.00949	0.9483	46	148

5.1 Detection Performance

The classifier was applied to the operational data set collected from live network observation. These resulting ROC curves for the ben/mal and opp/tar classifiers are shown in Figure 4 and Figure 5 respectively.

In addition, Table 9 and Table 10 list select data points from this graph. The cutoff reported in these tables is the minimum percentage of votes as benign that an observations must exceed to be considered the negative class (ben for ben/mal, opp for opp/tar).

5.2 Computational Complexity

The document classification process can be divided into three logical steps: feature extraction, classifier training, and classification of new observations. Feature extrac-

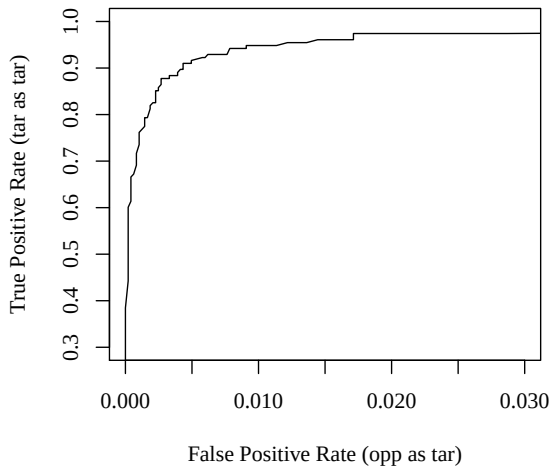

Figure 3: ROC for Training Set (opp/tar)

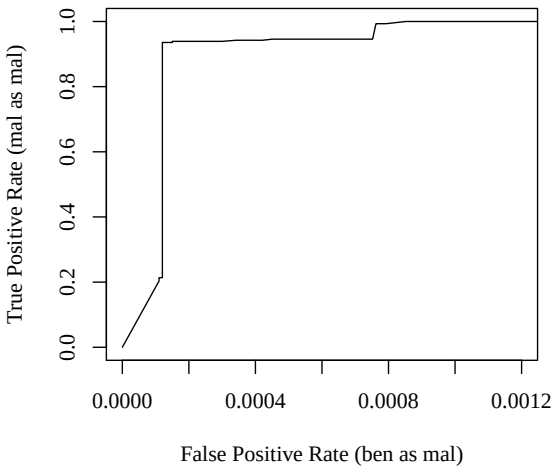

Figure 4: ROC for Operational Set (ben/mal)

Table 9: FP/TP Rates for Operational Set (ben/mal)

Cutoff	FP Rate	TP Rate	FP Count	TP Count
0.2	0.00021	0.9390	21	277
0.25	0.00042	0.9424	42	278
0.3	0.00075	0.9458	75	279
0.35	0.00089	1.0000	89	295
0.4	0.00154	1.0000	154	295
0.45	0.00193	1.0000	192	295

Table 10: FP/TP Rates for Operational Set (opp/tar)

Cutoff	FP Rate	TP Rate	FP Count	TP Count
0.2	0.0000	0.44	0	4
0.25	0.0000	0.56	0	5
0.3	0.0000	0.67	0	6
0.35	0.0000	0.78	0	7
0.4	0.0000	0.89	0	8
0.45	0.0070	1.00	2	9

tion must occur for both training data and new data to be classified. The majority of the processing in feature extraction is dedicated to matching signatures on the documents. This facet was poorly optimized in the implementation used for this study where multiple signatures were applied to the document serially, with each of the 9 signatures requiring another pass through the document. This implementation could be improved by making this signature matching parallel, which would improve performance about an order of magnitude and put performance roughly on par with conventional antivirus scanners.

Training the classifier is the most expensive operation. This training, however, need only occur infrequently. Once the features are extracted from documents to be classified, running these observations through the classifier is extremely fast. Table 11 demonstrates the run times of these operations applied to the training data set, which contains 10,000 documents. The experiments were performed on an Intel Xeon X5550 processor running 2.67GHz CPU. All the applications were executed in single-thread mode.

Similarly, little effort was placed into minimizing use of memory. However, for all operations, memory usage was negligible except for training the classifier, which required about 700 MB of RAM.

6 Adversarial Analysis & Robustness

Strong detection rates for past and current attacks are desirable, but it is important that any detection mechanism demonstrate resistance to intentional evasion. Therefore, the robustness of the selected features under mimicry and evasion attacks is crucial to the actual detection rates that can be achieved in a real-world environment. The detection mechanism presented in this paper is designed to classify documents based on similarity to past documents of the same class. These similarities between documents can arise from a wide spectrum of root causes varying among necessity, convenience, convention, and ambivalence. Presumably some of the attributes of malicious docs are easy to modify and others are more difficult. For example, while use of javascript is often not strictly required for exploiting vulnerabilities in PDF readers, it is often the most practical method for triggering a successful exploit. Hence the features related to the existence of javascript may be hard for attackers to

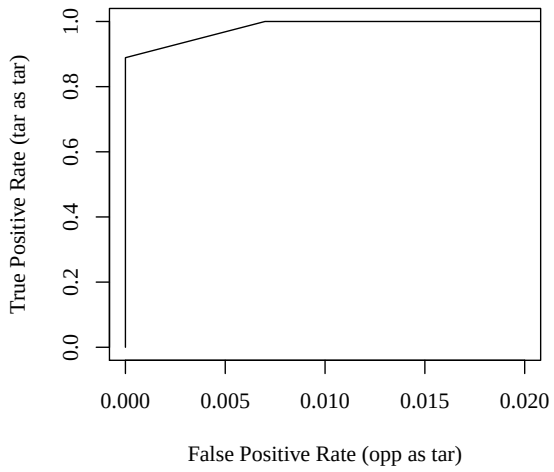

Figure 5: ROC for Operational Set (opp/tar)

Table 11: Run Times on Training Data

Operation	Time
Feature Extraction	12 min
Classifier Training	26 min
Observation Classification	4 sec

modify.

Alternatively, the lack of inclusion of a metadata item, such as the producer field, may be intentional to help prevent detection or the inclusion of the malware generation tool may be an omission on the part of the attacker. Spoofing the producer field may improve social engineering or it may facilitate detection through another method. It is infeasible to fully enumerate or to accurately predict or anticipate all evasion vectors, especially as some of the factors are dependent on the attacker and attack vector. Some constraints on evasion are also caused by the use of this mechanism in parallel to other techniques. However, the use of a training set from a different organization that was was published months before the use of the classifier provides strong indication that at least some resiliency exists in the similarities used as the basis for detection in this study. In the next part of this section, we will provide experimental evidence that support this claim and show the robustness of our approach against attacks that try to evade us.

6.1 Mimicry Attack Effectiveness

The classification and malware detection is based on similarity of extracted features to previous documents. An obvious evasion technique would be to mount a mimicry attack where malicious documents are purposefully modified to "normalize" some of their features and make them similar to benign documents while still retaining the embedded malicious content. If the attacker has knowledge of specific features used in the classifier and their importance, along with a good representation of what the defender considers as normal, the attacker can focus on mimicking the features most important for classification.

To simulate mimicry of document properties, the authors modified the top ranked features of malicious observations and subjected these modified observations to the classifier. For simplicity's sake, the documents themselves are not actually modified, but rather the previously extracted feature sets are modified. Specifically, the mean and standard deviation of the benign observations is calculated and the values for the malicious documents are replaced with random values which fit a normal distribution with the same mean and standard deviation. Note that this method may result in doctored features that are inconsistent or illogical. The six most important features, as ranked by the mean decrease in accuracy measurement, were selected for evasion testing. These features are ranked above the others with some amount of separation, as shown in Figure 6.

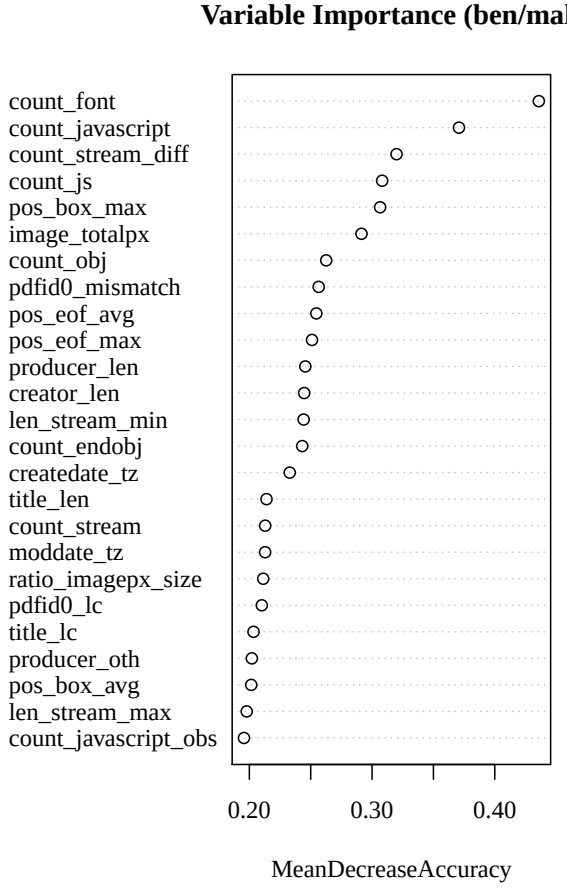

Figure 6: Variable Importance (ben/mal)

By causing the malicious samples to mirror the top six features of the benign, the benign-malicious classifier

error rate can be raised a great degree, as shown in Table 12. The average of the results of 3 independent trials using 10-fold cross validation is presented.

Table 12: Mimicry: Classifier Error Increase

Features Mimicked	Classification Error (%)
None	0.12
count_font	13.61
(+) count_javascript	18.90
(+) count_stream_diff	18.99
(+) count_js	21.35
(+) pos_box_max	23.27
(+) image_totalpx	23.30

By manipulating the most heavily used or distinctive features, it is possible to severely curtail the detection capabilities of the classifier.

6.2 Countering Mimicry

The best reaction to changes in document attributes leading to mis-classification is to retrain the classifier, causing the classifier to adjust how it treats the mimicked features. If retraining the classifier it is not adequate to raise classification rates to an acceptable level, additional features can be discovered and utilized instead. This tactic is reactionary at best and cannot ensure detection of documents that are very dissimilar to historical examples of documents of the same class. To be able to detect intentional evasion, proactive measures must be taken.

An obvious reaction to mimicry attacks on the features heavily employed by the classifier is to remove them altogether and rely on the other features. An important distinction is that variable importance, as reported by random forests, is an indication of the value of the feature as used in the classifier. However, that a feature has a high importance does not necessarily mean that the feature is a useful for classification on it's own nor does it mean that the classifier has to rely heavily on that feature for successful classification. Table 13 shows the increase in classification error as the top features are removed. The average of the results of 3 independent trials using 10-fold cross validation is presented. Note that there are small discrepancies in all Tables. These differences occur due to the variance introduced by use of random values in classification and data perturbation.

Removing the top ranked features has a surprisingly low affect on classification error because so many other useful features are retained. If the attacker is able to only modify a few attributes of malicious documents, and the defender is able to anticipate these, removing features may be an acceptable counter-measure. It is desirable to be able to counter evasion without fully negating the predictive value of variables targeted for evasion. One method of achieving this result is to vary (perturbate) the training set such that the resulting classifier is no longer as susceptible to evasion. The perturbation is performed by artificially modifying the features of a subset of the malicious observations in the training set to increase the variance of these features thus making them less "normal" . The loss of a focal point due to the increased variance reduces the importance of these features without fully eliminating them.

Table 13: Classifier Error with Features Removed

Features Removed	Classification Error (%)
None	.13
count_font	.20
(+) count_javascript	.27
(+) count_stream_diff	.27
(+) count_js	.31
(+) pos_box_max	.32
(+) image_totalpx	.32

Table 14: Classification Error with Training Data Perturbation

% Perturbation	Original Data	Mimicry Data
0	.13	25.68
.05	.15	17.02
.1	.18	11.37
.5	.20	1.63
1	.20	1.17
5	.25	.57
10	.27	.48
50	.34	.16
100	2.14	.12

To test the effectiveness of perturbation, the same method used to simulate evasion is used to modify a subset of the observations in the training set. The top six features of a subset of the malicious observations is set to values taken from a randomly generated normal distribution mirroring the mean and standard deviation of the benign observations. Table 14 shows the results of testing using the perturbation method. The average of the results of 3 independent trials using 10-fold cross validation is presented. The training data is perturbated and the resulting classifier is used both on the remaining unmodified training data and the same training data modified to simulate mimicry evasion. The percentage of the training data perturbated is varied, demonstrating a trade-off between accuracy with historical data and evasion resistance.

7 Future Work

As an extension to the presented results, It would be valuable to determine how well the same detection and classification techniques apply to other documents types. Another potential research avenue would be to determine how well the features used for PDF documents can

be used for grouping malicious documents by malware family and potentially across different file types.

The research presented here focused on identifying and utilizing features sufficient for high fidelity classification. Future areas of research could include investigating the value of other features, determining the optimal set of features, etc. Another option would be to study the features used in this study with other features, such as features derived from content analysis of the document, those derived from transport of the document over the network, recipient oriented features, and attacker oriented features. Lastly, the performance of this mechanism could be systematically compared to other techniques.

8 Conclusion

In this paper, we presented a classification approach for PDF documents that have embedded malicious code. We showed that by carefully extracting a wide-range of feature sets we can create a robust malware detector and classifier that yields very high rates of true positives (TP) while maintaining a low rate of (FP). We evaluated different machine learning techniques and we show that random forest appears to be the most effective. The experimental results that we run using more than 5,000 malicious documents and 100,000 benign ones yield classification rates above 99% while maintaining low false negative rates of 0.01% or less. We can also achieve classification of the malware documents into opportunistic and targeted with varying success depending on the detector objective (malware detection vs separation into classes), dataset, and training parameters.

Furthermore, our approach is robust against evasion and mimicry attacks that target the top classification features. We achieve that by artificially varying the ranges of the top features effectively reducing their influence on the detection. The result is that classification depends more equally on a very large number of features, making evasion much more difficult. Our experiments show that this strategy is still effective in detecting and classifying malware documents allowing for a defender to create bags of random features that he can use to expose malware that attempts to mimic the normality of the documents in the training set.

References

[1] ADOBE INC. Adobe security advisories: APSA11-04 - security advisory for adobe reader and acrobat. http://www.adobe.com/support/security/advisories/apsa11-04.html.

[2] ALPEROVITCH, D., AND (FIRM), M. Revealed operation shady RAT. http://www.mcafee.com/us/resources/white-papers/wp-operation-shady-rat.pdf, 2011.

[3] ANDRIANAKIS, C., SEYMER, P., AND STAVROU, A. Scalable web object inspection and malfease collection. In *Proceedings of the 5th USENIX conference on Hot topics in security* (2010), USENIX Association, pp. 1–16.

[4] BEVERLY, R., AND SOLLINS, K. Exploiting transport-level characteristics of spam. In *Conference on Email and Anti-Spam* (2008).

[5] COVA, M., KRUEGEL, C., AND VIGNA, G. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the 19th international conference on World wide web* (2010), ACM, pp. 281–290.

[6] CROSS, J. S., AND MUNSON, M. A. Deep PDF parsing to extract features for detecting embedded malware. Tech. Rep. SAND2011-7982, Sandia National Laboratories, Sept. 2011.

[7] DHAMANKAR, R., DAUSIN, M., EISENBARTH, M., AND KING, J. The top cyber security risks. http://www.sans.org/top-cyber-security-risks/, Sept. 2009.

[8] DRAKE, J. Exploiting memory corruption vulnerabilities in the java runtime.

[9] EGELE, M., WURZINGER, P., KRUEGEL, C., AND KIRDA, E. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. *Detection of Intrusions and Malware, and Vulnerability Assessment* (2009), 88–106.

[10] F-SECURE. PDF most common file type in targeted attacks - F-Secure weblog : News from the lab. http://www.f-secure.com/weblog/archives/00001676.html.

[11] GRIER, C., KING, S., AND WALLACH, D. How i learned to stop worrying and love plugins. In *In Web 2.0 Security and Privacy* (2009), Citeseer.

[12] KAKAVELAKIS, G., BEVERLY, R., YOUNG, J., LIMONCELLI, T., AND HUGHES, D. Auto-learning of SMTP TCP Transport-Layer features for spam and abusive message detection. In *LISA 2011, 25th Large Installation System Administration Conference* (Boston, MA, USA, Dec. 2011), USENIX Association.

[13] KOJM, T. ClamAV releases. http://lurker.clamav.net/message/20100816.145508.07ae1603.en.html, Aug. 2010.

[14] LASKOV, P., AND RNDI, N. Static detection of malicious JavaScript-bearing PDF documents. In *Proceedings of the 27th Annual Computer Security Applications Conference* (New York, NY, USA, 2011), ACSAC '11, ACM, p. 373382.

[15] LI, W.-J., STOLFO, S. J., STAVROU, A., ANDROULAKI, E., AND KEROMYTIS, A. D. A study of malcode-bearing documents. In *DIMVA* (2007), B. M. Hämmerli and R. Sommer, Eds., vol. 4579 of *Lecture Notes in Computer Science*, Springer, pp. 231–250.

[16] MONITOR., I. W., FOUNDATION., S., AND FOR INTERNATIONAL STUDIES. CITIZEN LAB., M. C. Shadows in the cloud investigating cyber espionage 2.0, 2010.

[17] NIKI, A. *Drive-by download attacks: Effects and detection methods*. PhD thesis, Master's thesis, Royal Holloway University of London, 2009.

[18] PARKOUR, M. contagio: Version 4 april 2011 - 11,355+ malicious documents - archive for signature testing and research. http://contagiodump.blogspot.com/2010/08/malicious-documents-archive-for.html.

[19] RAMILLI, M., AND BISHOP, M. Multi-stage delivery of malware. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on* (2010), IEEE, pp. 91–97.

[20] RAUTIAINEN, S. A look at portable document format vulnerabilities. *Information Security Technical Report 14*, 1 (2009), 30–33.

[21] SAHAMI, M., DUMAIS, S., HECKERMAN, D., AND HORVITZ, E. A bayesian approach to filtering junk E-Mail. *AAAI 98 Workshop on Text Categorization* (July 1998).

[22] SHAFIQ, M., KHAYAM, S., AND FAROOQ, M. Embedded malware detection using markov n-grams. *Detection of Intrusions and Malware, and Vulnerability Assessment* (2008), 88–107.

[23] STEVENS, D. Malicious pdf documents explained. *Security & Privacy, IEEE 9*, 1 (2011), 80–82.

[24] STOLFO, S., WANG, K., AND LI, W. Fileprint analysis for malware detection. *ACM CCS WORM* (2005).

[25] STONE-GROSS, B., COVA, M., KRUEGEL, C., AND VIGNA, G. Peering through the iframe. In *INFOCOM, 2011 Proceedings IEEE* (2011), IEEE, pp. 411–415.

[26] TABISH, S., SHAFIQ, M., AND 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), ACM, pp. 23–31.

[27] TZERMIAS, Z., SYKIOTAKIS, G., POLYCHRONAKIS, M., AND MARKATOS, E. Combining static and dynamic analysis for the detection of malicious documents. In *Proceedings of the Fourth European Workshop on System Security* (2011), ACM, p. 4.

Appendices

A Examples of Malicious PDF Structure

Table 15: Example Structure: Targeted

Location	Content	Description
000000 000020 000040	%PDF-1.5..%.....1 0 obj..<<Pag es 2 0 R /Type/Catalog/OpenActio n 8 0 R >>..endobj..2 0 obj..<</	PDF header and OpenAction object which executes javascript when document is opened
000140 000160	j..4 0 obj..<</Type/Font/BaseFon t/Times-Roman/Subtype/Type1>>..e	Font object: Times Roman.
000180 0001A0 ... 000390	ndobj..5 0 obj.. <</Length 471/Fi lter/FlateDecode >>stream..x.U..n ... .endstream..endobj..6 0 obj<< (Decoded, extracted raw text): Financial Reform Puts Republicans on the Spot (April 26) -- Even as they lost today's Senate vote	Object containing formatted text. The victim never sees this content. This content is not consistent with rest of social engineering used in attack.
0003E0 000400 000420	.endobj..7 0 obj..<</Type/Font/S ubtype/Type1/BaseFont/Helvetica/ Encoding/WinAnsiEncoding>>..endo	Font object: Helvetica.
000500 000520 000540 000560 000580	n..trailer..<</Size 9/Prev 01906 97/Root 1 0 R/ID[<5181383ede9472 7bcb32ac27ded71c68><5181383ede94 727bcb32ac27ded71c68>]>>..startx ref..0..%%EOF..8 0 obj..<</Type	Document trailer and PDF metadata (PDF IDs). These are likely artifacts of an existing benign document that was used in creation of this malicious document.
0005A0 0005C0 ... 0007D0 ... 000AC0	/Action /S /JavaScript /JS (..fu nction re(count, what) {...var v ... ...this.media.newPlayer(sgo);}.. ... ..Func8x9();..)..>>..endobj..9 0	Malicious javascript object. This javascript exploits CVE-2009-4324.
000AE0 000B00 ... 000F80	obj<</Filter /FlateDecode/Lengt h 1062>>stream..iPh4Code..... ... .endstream..endobj..10 0 obj< (Decoded, Extracted, and Disassembled shellcode): jmp short 0x12 pop edx dec edx xor ecx,ecx mov cx,0x40f xor byte [edx+ecx],0x8e loop 0xa jmp short 0x17 call 0x100000002	Object containing purported compressed stream. This stream is not compressed and contains shellcode. The excerpt shows a JMP-CALL-POP sequence followed by an XOR decryption loop that decodes more shellcode.

000FA0 000FC0 ... 3FF920	</Filter /FlateDecode/Length 114 688>>stream....l.............D.. ... ..endstream..endobj..xref..9 4.. (File Extracted and Decrypted using XOR key of 0xFC, location 000FCD to 0079CD): Type: PE32 executable for MS Windows (GUI) Intel 80386 32-bit Name: update.exe Compiled: Wed Dec 29 02:37:00 2010 (File Extracted and Decrypted using XOR Key of 0xFC, location 007A15 to 3FF921): Type: PDF document, version 1.5 CreationDate: D:20110125105603+08'00' Producer: PDFlib 7.0.3 (C++/Win32) PdfID0: D19C9464650960655B0FB612FD9702E0 PdfID1: D19C9464650960655B0FB612FD9702E0 (Decoded, extracted raw text): Rovos rail - Pride of Africa	Object containing purported compressed stream. This stream is not valid compressed data. The majority of the stream contains two encrypted payloads: a PE executable and a PDF document. The PE executable is decrypted, installed on system, and executed. This malware provides remote access trojan capabilities. The PDF document is decrypted and opened for user. This PDF's content is consistent with rest of social engineering in the attack. When considered in isolation, this dropped PDF is benign.
3FF9A0	12>>..startxref ..1092..%%EOF..	End of document.

Table 16: Example Structure: Opportunistic

Location	Content	Description
000000	%PDF-1.0..1 0 obj<</Type/Catalog	Document header.
000120 000140 000160 000180 ... 000690	>>endobj..7 0 obj..<</Creator 8 0 R>>..endobj..8 0 obj..<</Lengt h 1299 /Filter /FlateDecode..>>. .stream..x..Y...(..R.'...k:.'..? ... .bw...endstream..endobj. (Deflated and un-escaped to reveal javascript excerpt): function a(){util.printd('p@1111 1111111111111111111 : yyyy111', new Date());}var h = app.plugIn ... d + e;}a();a();try {this.media.n ewPlayer(null);} catch(e) {}a(); (Further un-escaped to reveal shellcode excerpt): t....URLMON.DLL. URLDownloadToFil eA.pdfupd.exe.crash.php.http://1 11.gosdfsdjas.com/l.php?i=16..	Reference to, definition of object representing the "Creator" metadata item. This object contains a compressed stream. The compressed stream contains obfuscated javascript and which contains obfuscated shellcode. The javascript exploits CVE-2009-4324. The shellcode downloads more malicious content from the Internet.
0006B0 0006D0 ... 000770	.111611 0 obj<</Filter/FlateDeco de/Length 142>>..stream..x...J.* ... a........endstream..endobj..trai (De-obfuscated by removing comments to reveal javascript): var b=this.creator;var a=unescap e(b);eval(unescape(this.creator. replace(/z/igm,'%'));	Compressed javascript object. This obfuscated javascript un-escapes and executes javascript in "Creator" stream above.

000790 0007B0	ler<</Info 7 0 R /Root 1 0 R /Si ze 11>>	End of document (without %EOF footer).

B Feature Descriptions

Name	Description	Name	Description
size	Size of document (bytes)	version	PDF version as extracted from header
count_obj	Count of object markers	count_endobj	Count of end of object markers
count_stream	Count of stream markers	count_endstream	Count of end of stream markers
count_xref	Count of cross reference table markers	count_trailer	Count of trailer markers
count_startxref	Count of cross reference table markers	count_eof	Count of end of file markers
count_page	Count of page markers	count_objstm	Count of object stream markers
count_js	Count of JS object markers	count_javascript	Count of JavaScript object markers
count_action	Count of action (AA and OpenAction) object markers	count_acroform	Count of Acroform object markers
count_font	Count of Font object markers	count_stream_diff	Difference of count_stream and count_endstream
moddate_ts	Modification timestamp (seconds–Unix epoch)	moddate_tz	Modification timezone (seconds–UTC offset)
createdate_ts	Creation timestamp (seconds–Unix epoch)	createdate_tz	Creation timezone (seconds–UTC offset)
createdate_version_ratio	Ratio of creation date to version (days since Jan 1 1993 / version)	moddate_version_ratio	Ratio of modification date to version (days since Jan 1 1993 / version)
createdate_dot	Count of dot characters in creation date	moddate_dot	Count of dot characters in modification date
pdfid0_len	PDFid0: Count of characters	pdfid0_lc	PDFid0: Count of lower case characters
pdfid0_uc	PDFid0: Count of upper case characters	pdfid0_num	PDFid0: Count of numeric characters
pdfid0_oth	PDFid0: Count of other characters	pdfid0_dot	PDFid0: Count of dot characters
pdfid1_len	PDFid1: Count of characters	pdfid1_lc	PDFid1: Count of lower case characters
pdfid1_uc	PDFid1: Count of upper case characters	pdfid1_num	PDFid1: Count of numeric characters
pdfid1_oth	PDFid1: Count of other characters	pdfid1_dot	PDFid1: Count of dot characters
pdfid_mismatch	pdfid0 different from pdfid1 (binary)	title_len	Title: Count of characters
title_lc	Title: Count of lower case characters	title_uc	Title: Count of upper case characters
title_num	Title: Count of numeric characters	title_oth	Title: Count of other characters
title_dot	Title: Count of dot characters	author_len	Author: Count of characters
author_lc	Author: Count of lower case characters	author_uc	Author: Count of upper case characters
author_num	Author: Count of numeric characters	author_oth	Author: Count of other characters
author_dot	Author: Count of dot characters	producer_len	Producer: Count of characters
producer_lc	Producer: Count of lower case characters	producer_uc	Producer: Count of upper case characters
producer_num	Producer: Count of numeric characters	producer_oth	Producer: Count of other characters
producer_dot	Producer: Count of dot characters	creator_len	Creator: Count of characters
creator_lc	Creator: Count of lower case characters	creator_uc	Creator: Count of upper case characters
creator_num	Creator: Count of numeric characters	creator_oth	Creator: Count of other characters
creator_dot	Creator: Count of dot characters	subject_len	Subject: Count of characters
subject_lc	Subject: Count of lower case characters	subject_uc	Subject: Count of upper case characters
subject_num	Subject: Count of numeric characters	subject_oth	Subject: Count of other characters
subject_dot	Subject: Count of dot characters	keywords_len	Keywords: Count of characters
keywords_lc	Keywords: Count of lower case characters	keywords_uc	Keywords: Count of upper case characters
keywords_num	Keywords: Count of numeric characters	keywords_oth	Keywords: Count of other characters
keywords_dot	Keyworks: Count of dot characters	count_page_obs	Count of obfuscated page markers
count_objstm_obs	Count of obfuscated object stream markers	count_js_obs	Count of obfuscated JS object markers
count_javascript_obs	Count of obfuscated JavaScript object markers	count_action_obs	Count of obfuscated action object markers
count_acroform_obs	Count of obfuscated Acroform objects	count_font_obs	Count of obfuscated Font object markers
delta_ts	Difference between creation and modification timestamps	delta_tz	Difference between creation and modification timezones
moddate_mismatch	Count of differences in modification timestamp values	createdate_mismatch	Count of differences in creation timestamp values
pdfid0_mismatch	Count of differences in PDFid0 values	pdfid1_mismatch	Count of differences in PDFid1 values
title_mismatch	Count of differences in title values	author_mismatch	Count of differences in author values
producer_mismatch	Count of differences in producer values	creator_mismatch	Count of differences in creator values
company_mismatch	Count of differences in company values	subject_mismatch	Count of differences in subject values
keywords_mismatch	Count of differences in keywords values	count_box_a4	Count of A4 sized boxes
count_box_letter	Count of US letter sized boxes	count_box_overlap	Count of A4-width, letter-height boxes
count_box_legal	Count of legal sized boxes	count_box_other	Count of other boxes

box_other_only	Boxes are all other sized (binary)	box_nonother_types	Count of page sized (A4, letter, etc) boxes
image_totalpx	Sum of image pixels	count_image_total	Count of image objects
count_image_xsmall	Count of images between 0 and 4096 pixels	count_image_small	Count of images between 4097 and 64000 pixels
count_image_med	Count of images between 64001 and 786432 pixels	count_image_large	Count of images between 786433 and 12582912 pixels
count_image_xlarge	Count of images over 12582912 pixels	image_mismatch	Count of differences in image dimensions
ratio_imagepx_size	Ratio of image_totalpx to size	ratio_size_obj	Ratio of count_obj to size
ratio_size_stream	Ratio count_stream to size	ratio_size_page	Ratio of count_page to size
len_obj_min	Minimum difference between position of obj and next endobj markers	len_obj_max	Maximum difference between position of obj and next endobj markers
len_obj_avg	Average difference between position of obj and next endobj markers	len_stream_min	Minimum difference between position of stream and next endstream markers
len_stream_max	Maximum difference between position of stream and next endstream markers	len_stream_avg	Average difference between position of stream and next endstream markers
pos_eof_min	Normalized position of first EOF marker (% of size)	pos_eof_max	Normalized position of last EOF marker (% of size)
pos_eof_avg	Average of normalized positions of last EOF marker (% of size)	pos_page_min	Normalized position of first page marker (% of size)
pos_page_max	Normalized position of last page marker (% of size)	pos_page_avg	Average normalized positions of page markers (% of size)
pos_acroform_min	Normalized position of first acroform marker (% of size)	pos_acroform_max	Normalized position of last acroform marker (% of size)
pos_acroform_avg	Average normalized positions of page markers (% of size)	pos_box_min	Normalized position of first box marker (% of size)
pos_box_max	Normalized position of last marker (% of size)	pos_box_avg	Average normalized positions of box markers (% of size)
pos_image_min	Normalized position of first image marker (% of size)	pos_image_max	Normalized position of last image marker (% of size)
pos_image_avg	Average normalized positions of image markers (% of size)		