Malicious PDF Detection using Metadata and Structural Features

Charles Smutz
Center for Secure Information Systems
George Mason University, Fairfax, VA 22030
csmutz@gmu.edu

Angelos Stavrou
Center for Secure Information Systems
George Mason University, Fairfax, VA 22030
astavrou@gmu.edu

ABSTRACT

Owed to their versatile functionality and widespread adoption, PDF documents have become a popular avenue for user exploitation ranging from large-scale phishing attacks to targeted attacks. In this paper, we present a framework for robust detection of malicious documents through machine learning. Our approach is based on features extracted from document metadata and structure. Using real-world datasets, we demonstrate the the adequacy of these document properties for malware detection and the durability of these features across new malware variants. Our analysis shows that the Random Forests classification method, an ensemble classifier that randomly selects features for each individual classification tree, yields the best detection rates, even on previously unseen malware.

Indeed, using multiple datasets containing an aggregate of over 5,000 unique malicious documents and over 100,000 benign ones, our classification rates remain well above 99% while maintaining low false positives of 0.2% or less for different classification parameters and experimental scenarios. Moreover, the classifier has the ability to detect documents crafted for targeted attacks and separate them from broadly distributed malicious PDF documents. Remarkably, we also discovered that by artificially reducing the influence of the top features in the classifier, we can still achieve a high rate of detection in an adversarial setting where the attacker is aware of both the top features utilized in the classifier and our normality model. Thus, the classifier is resilient against mimicry attacks even with knowledge of the document features, classification method, and training set.

1. INTRODUCTION

To achieve higher rate of infection through social engineering and, in some cases, to avoid detection, malicious code is often embedded or packaged within documents that appear legitimate including government forms and bank statements. These maliciously crafted 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 number of vulnerabilities in client applications, including document viewers, has increased [7, 8]. Indeed, the risk of infection appears to be higher because attackers entice users to open a malware-bearing document through social engineering. Moreover, 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 demonstrated 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 [23, 2]. The use of trojan PDFs in targeted attacks, including those perpetrated by a class of attacker called the Advanced Persistent Threat (APT), 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 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 attached document augments the social engineering aspect of the attack. 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, 25, 9, 27]. 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 engineer-

ACSAC '12 Dec. 3-7, 2012, Orlando, Florida USA
Copyright 2012 ACM 978-1-4503-1312-4/12/12 ...$15.00.

ing. Many malicious documents seem to be truly Spartan, in that they contain 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. 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 including commodity antivirus scanners [13, 29] 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.

In this paper, we explore the limits of static analysis detection mechanisms that utilize machine learning techniques on document-specific attributes to identify embedded 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 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 operation of SPAM filters that use features from the email and 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. We show that a large number of these features are suitable for high accuracy classification. 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 the Random Forests algorithm performs the best with True Positives (TP) more than 0.99 (99%) while maintaining a False Positive (FP) rate of 0.002 (0.2%) or less depending on the scenario, dataset, and thresholds used. This ensemble classifier is able to classify previously unseen variants. An important finding of our study is the effectiveness of mimicry type attacks can be mitigated. This due to the large set of features we extracted and the ability to construct an ensemble classifier providing robust detection 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 unevenly favored by the classifier, evasion becomes much more difficult.

2. RELATED WORK

Malware bearing documents have been the subject of much research over the years. In terms of static analysis, Li et al. [15] and Tabish et al. [28] 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 and pattern matching has been studied extensively [26, 24] including specifically for PDF vulnerabilities [20, 16].

Addressing detection of malware-laden PDFs using static analysis, Laskov and Šrndić[14] extract, process, and classify javascript embedded in PDFs using support vector machines. However, their approach yields much lower detection rates and they do not provide a robustness analysis for their method. 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. Moreover, the framework presented here differs in that static analysis is used to extract a much larger number of features, including metadata, without seeking to fully decode the document through dynamic analysis. In both of the aforementioned 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 these issues by focusing on different features and different methods of extracting those features.

Furthermore, Tzermias et al. [29] 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 yielding 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

Table 1: Data Set Summary

	Training	Testing/Operational
benign (ben)	5,000	99,703
opportunistic (opp)	4,802	286
targeted (tar)	198	11
total	10,000	100,000

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 [22]. More recently, progress has been made towards effective network and transport layer identification of SPAM [4, 12]. One could consider features such as parameters of network traffic used for SPAM delivery to be analogous to the features of documents used in this study. These features are not measuring inherently malicious attributes, but they often reflect malicious activity and are useful in practice.

3. DATA & FEATURE DESCRIPTION

3.1 Data Types

This study focuses on classifying malicious documents, specifically 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. Targeted attacks are separated from opportunistic ones by factors such as victim specific social engineering and correlations with other known 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 a few additional 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.

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 feed, 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. 11 malicious PDFs associated with targeted attacks on the same organization, but representing multiple victims and attack groups, 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. Sampling, when it occured, 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 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.

4. CLASSIFICATION METHODOLOGY

4.1 Feature Extraction

We implemented our own program for reliably extracting features from PDF documents due to the inability of existing tools to cope with malformed 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 of each stream is measured by the difference between the location of a "stream" marker and and the next "endstream" 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

Table 2: Classifier Performance Comparison

Classifier	Error Rate	Train Time	Classify Time
Naive Bayes	27%	2 sec	95 sec
Random Forest	.19%	92 sec	1 sec
Support Vector Machine	17%	218 sec	33 sec

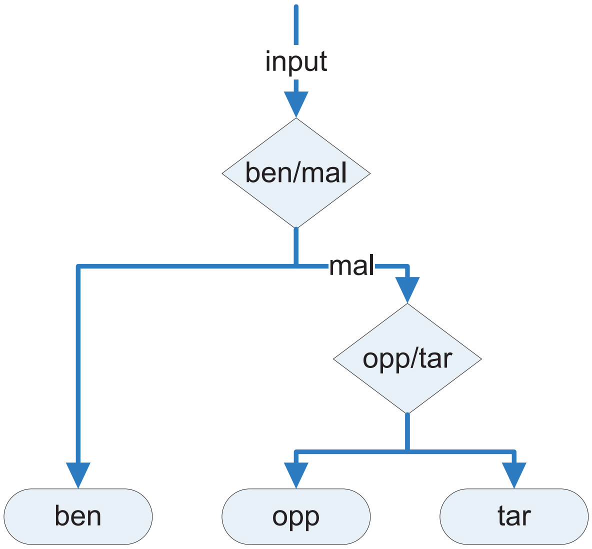

Figure 1: Dual Classifier Arrangement

the title. This software functions without significant PDF structure parsing or validation which is necessary for success in dealing with malformed documents and provides strong performance. 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. Feature extraction and classification works well even on encrypted documents because in PDF documents each object/stream is encrypted individually, leaving structure and metadata to be extracted the same as normal documents.

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.

4.2 Feature Selection

We selected a large number of features to characterize PDF documents. Our aim 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, features are derived either from PDF document metadata or structure.

In our approach, the extracted 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, but which have little general application, because including these features could result in strong classification for known attacks while yielding low detection rates for novel attacks.

The philosophy for feature identification was to generate as many features that parameterize the metadata and structure of the document as possible, without short-sighted regard for usefulness of individual features in discriminating between document classes. The features reflect properties of the metadata, such as the count of the characters in each field; objects/streams, such as the size and count of each; boxes and images, such as the size and location of each; data encoding methods, such as use of each data encoding method; and object types, such as count of encryption objects. In total, 202 features were chosen for use.

It is beyond the scope of this document to explain all 202 features, but some examples will be given. The names of the top features are shown in Figure 7. The count_font, count_javascript, and count_js features are determined by the number of instances of "/Font", "/JavaScript", and "/JS" markers. The count_stream_diff variable is the difference of the instances of "stream" and "endstream" markers. The relative position in the document of the last box marker (box objects are used in layout) is reported as pos_box_max. The sum of all the pixels in all the images in the document is named image_totalpx. producer_len is the number of characters in the producer metadata object and count_obj is the number of instances of the "obj" marker. Each document has a unique document identifier that should never be modified between revisions which is called pdfid0. pdfid0_mismatch reflects the number of unique instances of pdfid0 values in a document (which is usually always 1). These features are identified by either simple strings matches or more complex regular expressions applied to the raw document.

Most features are taken directly from observation of the document metadata or document structure, such as 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.

4.3 Random Forests

To categorize observed documents, the features are extracted and run through a classifier generated from labeled training data. Random Forests was selected because of its effective classification capability, strong performance, and ease of use. Table 2 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 and timing runs. We performed all of our analysis using R [21], including the randomForest and e1071 packages.

The Random Forests classification method gives the result of classification based on the output of many individual classification trees, each of which votes for one of possible classes. Each decision tree is generated from a randomly selected subset of training data. Hence, random forests is an ensemble classifier using bagged training data. Each node in a tree is created by selecting a randomly selected subset of features and determining the best split at each node using the training data for that node. Furthermore, each tree is based on an independent subset of features. Lastly, during classification the votes of each tree determine the result.

The primary tunables for a random forest is the number of variable to try at each node (mtry) and the number of trees to create (ntree). By default, a random forest is constructed using the square root of the number of variables for mtry and 500 for ntree. It is recommended that these values be tuned by trying half the default and double the default in addition to the default. It was determined that 3 times the default is optimal for mtry and double the default is optimal for ntree for this application. These values, 1000 for ntree 43 for mtry, are used throughout this paper.

4.4 Classification Techniques

For classification, 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. These votes are combined to determine the end 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 threshold of the votes from individual trees 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 vote threshold value during prediction.

5. PERFORMANCE EVALUATION

Are structural and metadata features adequate for discriminating malicious documents from benign? If so, how many of these features are needed?

In Figure 2 we depict the classification error decrease as features are added to the benign/malicious classifier. The average and 95% confidence interval of the classification error of multiple randomly selected subsets of features are presented.

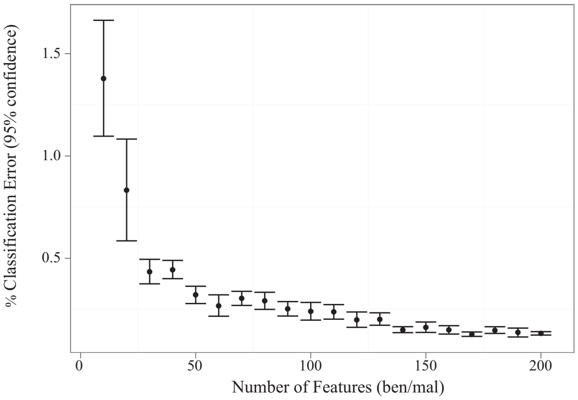

Figure 2: Error Decrease with Feature Count(ben/mal)

5.1 Classification & Detection Performance

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 opp/tar classifiers are displayed in Figure 3 and Figure 4 respectively.

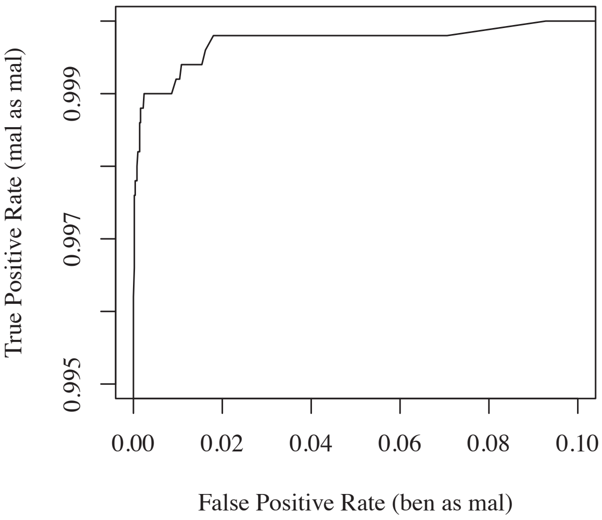

Figure 3: ROC for Training Set(ben/mal)

In addition, Table 3 and Table 4 list select data points from these graphs. The cutoff reported in these tables is the minimum percentage of votes that an observations must exceed to be considered the positive class (mal for ben/mal, tar for opp/tar).

The classifier (trained with the training set) was applied to the operational data set collected from live network observation. In lieu of presenting the ROC graphs of the classifiers applied to the operational data set, Figures 5 and 6 contain

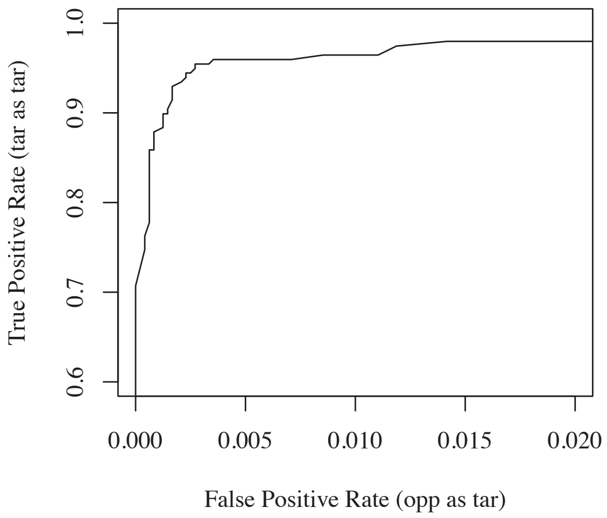

Figure 4: ROC for Training Set (opp/tar)

Table 3: FP/TP Rates: Training Set (ben/mal)

Votes	FP Rate	TP Rate	FP Count	TP Count
0.7	0	0.9942	0	4971
0.6	0	0.9962	0	4981
0.5	0.0008	0.998	4	4990
0.4	0.0014	0.9986	7	4993
0.3	0.0034	0.999	17	4995
0.2	0.0076	0.999	38	4995
0.1	0.0208	0.9998	104	4999

the density plots of the votes for the two classes in each of the binary classifiers.

These plots show the separation between the classes in each classifier, focusing on the ratio of votes that observations of a given class receive for the positive class from the the independent trees in the forest. Perfect classification would result in the negative class being a spike at 0 and the positive class being located at 1. Note that while the operational data set is largely independent of the training set, the classifiers still provide strong discrimination between the classes, even if some trees contribute an incorrect vote. These plots also clearly visualize the ability the operator has to tune the sensitivity of the classifier by adjusting the vote threshold.

In Table 5 and Table 6 we list select data points for the ROC from this data.

5.2 New Variant Detection

To demonstrate the resiliency of structural and metadata features across differing data sets, the results of the off-the-self AV scanners were used to distinguish "variants" of very similar malicious documents. The results of the 5 AV scanners were used to create a variant identifier which is the 5-tuple of the signature name of each scanner. The results of categorization of the samples into variants are shown in Table 7.

This categorization of samples into groups of variants resulted in a significant reduction in the opportunistic samples but a relatively minor reduction in the targeted samples. This is consistent with the wider and more automated distribution of opportunistic samples as well as the more exclusive distribution and higher likelihood to include manual modifications to support AV evasion of targeted samples.

Table 4: FP/TP Rates: Training Set (opp/tar)

Votes	FP Rate	TP Rate	FP Count	TP Count
0.7	0.00125	0.8889	6	176
0.6	0.00167	0.9195	8	182
0.5	0.00271	0.9495	13	188
0.4	0.00333	0.9545	16	189
0.3	0.00437	0.9595	21	190
0.2	0.00583	0.9595	28	190
0.1	0.00854	0.9645	41	191

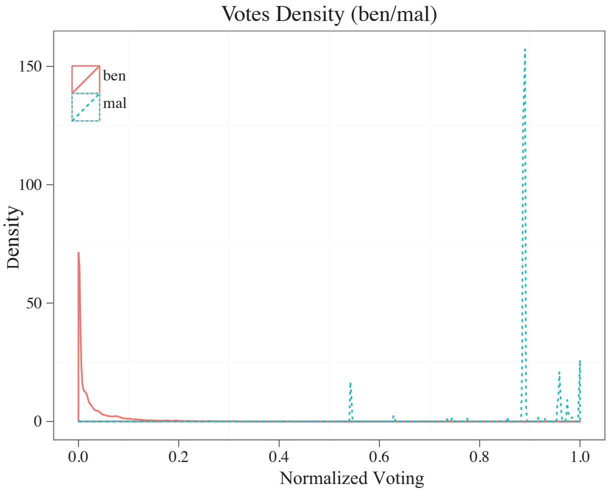

Figure 5: Classification Votes Density (ben/mal)

To the degree possible to discern from the names, the AV signatures are related to the exploit and javascript used in exploitation. There were relatively few signatures that appeared to be related to the actual malware families concealed in the document. Indeed, many of the documents merely contain shellcode which in turn downloads specific malware. Of the two groups of targeted variants in the operational data set, one pair was attributed to the same persistent actor/embedded malware family, while the other pair of documents were from separate actors/embedded malware families. In both cases, the documents had very similar structure and metadata, leading to the conclusion they they were derived from the same document template.

There is a relatively small amount of overlap in the variants from the training and operational data sets. The ability of the classifier to effectively classify new variants that were not included in the training set demonstrates that the features used for classification are durable. Variations in malicious documents that require unique signatures can be detected with a common classifier based on metadata and structure.

5.3 Comparison to PJScan

To demonstrate the effectiveness of the mechanisms presented here and to add further validation to the quality of the data sets used, the results of using PJScan are presented

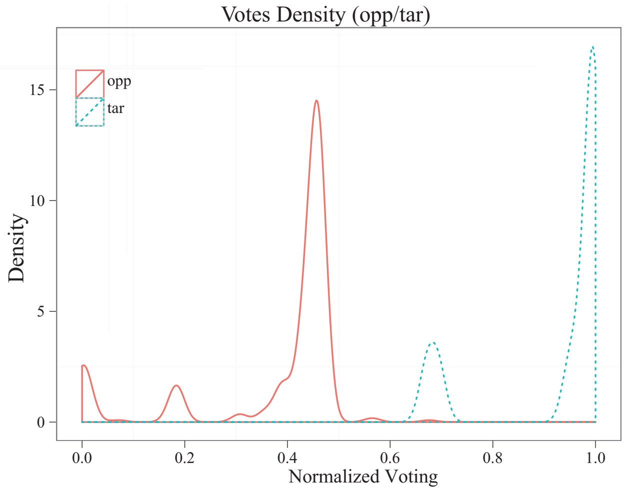

Figure 6: Classification Votes Density (opp/tar)

Table 5: FP/TP Rates: Operational Set (ben/mal)

Votes	FP Rate	TP Rate	FP Count	TP Count
0.8	0.00021	0.9327	21	277
0.7	0.00057	0.9461	57	281
0.6	0.00132	0.9529	132	283
0.5	0.00244	1.0000	243	297

here. PJScan is not designed to separate targeted from opportunistic attacks. Thus, we only used ben/mal classification to separate malicious from benign documents regardless of the type of threat. The classifier was trained on the 5,000 malicious documents in the training set with default parameters. Results are shown in Table 8. PJScan only returns a result for documents from which it can extract javascript. Therefore, in our experiments, we only count the number of documents for which a result is returned. The classification error rate is given for those documents which a result is returned.

PJScan was unable to classify many malicious documents. Manual analysis reveals that many of the malicious documents PJScan can't analyze do have javascript in them but have the javascript in atypical locations. Javascript code inside document metadata sections or inside corrupted document structures are not correctly parsed by PJScan, which is a known limitation. It appears that the newer operational data set has a much higher prevalence of these conditions which prevent successful analysis. PJScan provides decent results for the training set, but when the trained classifier is applied to the operational data set, the quality of classification drops dramatically.

The biggest limitation of PJScan applied to the data sets in this study is the inability to successfully extract the features used for classification. The mechanisms presented here, which utilize simple signature matching without document parsing or decoding, compare favorably in this as they can be more reliably extracted in practice. The mechanisms presented here also compare favorably when considering the adequacy and durability of the classifier when applied to the training data and extrapolated to other, independent data sets.

Table 6: FP/TP Rates: Operational Set (opp/tar)

Votes	FP Rate	TP Rate	FP Count	TP Count
0.8	0.0000	0.82	0	9
0.7	0.0000	0.82	0	9
0.6	0.0035	1.00	1	11
0.5	0.0105	1.00	3	11

Table 7: Variants in Data Sets

	Training	Operational	Overlap
opp samples	4,802	286	-
opp variants	812	31	6
tar samples	198	11	-
tar variants	186	9	1

5.4 Computational Complexity

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

Once the features are extracted from documents to be classified, running these observations through the classifier is extremely fast. Training the classifier is more expensive, but this only needs to occur infrequently. Table 9 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.

Table 9: Run Times on Training Data

Operation	Time
Feature Extraction	14 min
Classifier Training	38 sec
Observation Classification	1 sec

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 1 GB of RAM.

6. ADVERSARIAL ANALYSIS

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

Table 8: PJScan results

Data Set	Class	Classified	Not Classified	Classification Error	Total Detection Rate
Training	ben	5%	95%	1%	-
Training	mal	85%	15%	9%	78%
Operational	ben	3%	97%	1%	-
Operational	mal	17%	83%	36%	3%

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 modify. Alternatively, it may be trivial to spoof or remove metadata such as the producer field. It is infeasible to fully enumerate or to accurately predict or anticipate all the methods used to attempt evasion, 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.

Note that the mere existence of benign elements or lack of specific malicious artifacts is not sufficient to evade detection. Some of the malicious documents evaluated in this study contained large portions of benign content indicative that the malware packager either intentionally added benign elements to a malicious document or that the exploit and malware were added to an existing benign document. Conversely, many malicious documents are devoid of optional metadata. The ability to correctly classify documents despite conflicting alignment or differences between documents demonstrates the value of the machine learner.

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 shows the robustness of our approach on attacks that attempt direct evasion.

6.1 Mimicry Attack Effectiveness

One likely 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 7.

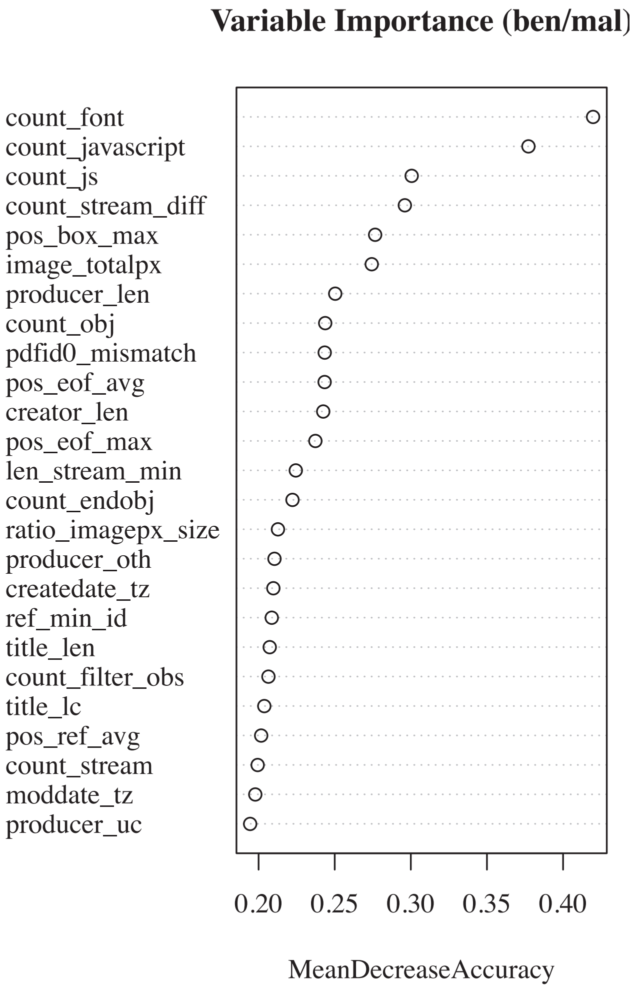

Figure 7: Feature/Variable Importance (ben/mal)

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 10. The average of the results of 5 independent trials using 10-fold cross validation is presented.

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

Table 10: Mimicry: Classifier Error Increase

Features Mimicked	Classification Error (%)
None	0.14
count_font	12.26
(+) count_javascript	17.04
(+) count_stream_diff	20.01
(+) count_js	20.07
(+) pos_box_max	22.18
(+) image_totalpx	22.30

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 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 11 shows the increase in classification error as the top features are removed.

Table 11: Classifier Error with Features Removed

Features Removed	Classification Error (%)
None	.14
count_font	.21
(+) count_javascript	.28
(+) count_stream_diff	.28
(+) count_js	.29
(+) pos_box_max	.29
(+) image_totalpx	.29

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.

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 12 shows the results of testing using the perturbation method. The average of the results of 5 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.

Table 12: Classification Error with Training Data Perturbation

% Perturbation	Original Data	Mimicry Data
0	.14	26.12
.05	.14	14.11
.1	.15	9.19
.5	.15	1.80
1	.16	1.13
5	.21	.69
10	.22	.52
50	.26	.16
100	2.06	.12

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 classifying documents are suited to grouping malicious documents by malware family. Another option would be to combine 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

We presented a classification approach for PDF documents that have embedded malicious code. We showed that by extracting a wide-ranging feature set we can create a robust malware detector and classifier that yields very high rates of true positives (TP) while maintaining a low rate of false positives (FP). We evaluated different machine learning techniques and we show that Random Forests appears to be the most effective.

The experimental results obtained using more than 5,000 malicious documents and 100,000 benign ones yield classification rates above 99% while maintaining low false positive rates of 0.2% or less. We can also achieve classification of the malware documents into opportunistic and targeted categories. We also demonstrate that the classifier is effective at identifying new variants by applying the classifier trained on one data set to a totally independent training set with great success.

Furthermore, our approach is robust against direct 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 to accomplish. Our experiments show that this strategy is still effective in detecting and classifying malware documents allowing for a defender to create a classifier that exposes malware that attempts to mimic the normality of the documents in the training set.

9. 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] D. Alperovitch and M. (Firm). Revealed operation shady RAT. http://www.mcafee.com/us/resources/white-papers/wp-operation-shady-rat.pdf, 2011.

[3] C. Andrianakis, P. Seymer, and A. Stavrou. Scalable web object inspection and malfease collection. In *Proceedings of the 5th USENIX conference on Hot topics in security*, pages 1–16. USENIX Association, 2010.

[4] R. Beverly and K. Sollins. Exploiting transport-level characteristics of spam. In *Conference on Email and Anti-Spam*, 2008.

[5] M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the 19th international conference on World wide web*, pages 281–290. ACM, 2010.

[6] J. S. Cross and M. A. Munson. Deep PDF parsing to extract features for detecting embedded malware. Technical Report SAND2011-7982, Sandia National Laboratories, Sept. 2011.

[7] R. Dhamankar, M. Dausin, M. Eisenbarth, and J. King. The top cyber security risks. http://www.sans.org/top-cyber-security-risks/, Sept. 2009.

[8] J. Drake. Exploiting memory corruption vulnerabilities in the java runtime. 2011.

[9] M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 88–106, 2009.

[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] C. Grier, S. King, and D. Wallach. How i learned to stop worrying and love plugins. In *In Web 2.0 Security and Privacy*. Citeseer, 2009.

[12] G. Kakavelakis, R. Beverly, J. Young, T. Limoncelli, and D. Hughes. 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] T. Kojm. ClamAV releases. http://lurker.clamav.net/message/20100816.145508.07ae1603.en.html, Aug. 2010.

[14] P. Laskov and N. Šrndić. Static detection of malicious javascript-bearing pdf documents. In *Annual Computer Security Applications Conference*, 2011.

[15] W.-J. Li, S. J. Stolfo, A. Stavrou, E. Androulaki, and A. D. Keromytis. A study of malcode-bearing documents. In B. M. Hämmerli and R. Sommer, editors, *DIMVA*, volume 4579 of *Lecture Notes in Computer Science*, pages 231–250. Springer, 2007.

[16] D. Maiorca, G. Giacinto, and I. Corona. A pattern recognition system for malicious pdf files detection. In P. Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, volume 7376 of *Lecture Notes in Computer Science*, pages 510–524. Springer Berlin / Heidelberg, 2012.

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

[18] M. Parkour. 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] M. Ramilli and M. Bishop. Multi-stage delivery of malware. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 91–97. IEEE, 2010.

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

[21] R project. http://www.r-project.org/.

[22] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk E-Mail. *AAAI 98 Workshop on Text Categorization*, July 1998.

[23] Shadows in the cloud: Investigating cyber espionage 2.0. http://shadows-in-the-cloud.net/, 2010.

[24] M. Shafiq, S. Khayam, and M. Farooq. Embedded malware detection using markov n-grams. *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 88–107, 2008.

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

[26] S. Stolfo, K. Wang, and W. Li. Fileprint analysis for malware detection. *ACM CCS WORM*, 2005.

[27] B. Stone-Gross, M. Cova, C. Kruegel, and G. Vigna. Peering through the iframe. In *INFOCOM, 2011 Proceedings IEEE*, pages 411–415. IEEE, 2011.

[28] S. Tabish, M. Shafiq, and M. Farooq. Malware detection using statistical analysis of byte-level file content. In *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, pages 23–31. ACM, 2009.

[29] Z. Tzermias, G. Sykiotakis, M. Polychronakis, and E. Markatos. Combining static and dynamic analysis for the detection of malicious documents. In *Proceedings of the Fourth European Workshop on System Security*, page 4. ACM, 2011.