Practical Evasion of a Learning-Based Classifier: A Case Study

Nedim Šrndić and Pavel Laskov
Department of Cognitive Systems
University of Tübingen
Tübingen, Germany
{nedim.srndic, pavel.laskov}@uni-tuebingen.de

***Abstract*—Learning-based classifiers are increasingly used for detection of various forms of malicious data. However, if they are deployed online, an attacker may attempt to evade them by manipulating the data. Examples of such attacks have been previously studied under the assumption that an attacker has full knowledge about the deployed classifier. In practice, such assumptions rarely hold, especially for systems deployed online. A significant amount of information about a deployed classifier system can be obtained from various sources. In this paper, we experimentally investigate the effectiveness of classifier evasion using a real, deployed system, PDFRATE, as a test case. We develop a taxonomy for practical evasion strategies and adapt known evasion algorithms to implement specific scenarios in our taxonomy. Our experimental results reveal a substantial drop of PDFRATE's classification scores and detection accuracy after it is exposed even to simple attacks. We further study potential defense mechanisms against classifier evasion. Our experiments reveal that the original technique proposed for PDFRATE is only effective if the executed attack exactly matches the anticipated one. In the discussion of the findings of our study, we analyze some potential techniques for increasing robustness of learning-based systems against adversarial manipulation of data.**

I. Introduction

Data analysis methods such as machine learning are increasingly used in security applications. For tasks like malware analysis, deployment of learning methods has become almost imperative. Data-driven analysis enables automatic attribution of seemingly heterogeneous malware samples to a modest number of genuine malware families [1], [2]. Recent work has also witnessed several innovative applications of machine learning for detection of various kinds of security violations, e.g., drive-by-downloads [3], [4], malicious web pages [5], [6], compromised accounts and fake identities in social networks [7], [8], unwanted P2P traffic [9] and many others.

Clearly, deployment of learning methods in any security-critical context requires that they can withstand potential attacks. The security of machine learning methods has been previously discussed from conceptual [10], methodical [11], [12], [13], [14] and practical [15], [16], [17] viewpoints. Despite the growing evidence for susceptibility of learning-based approaches to adversarial data manipulation, this seems to be of little hindrance for their acceptance as a versatile tool for data-intensive security tasks. Typically, the security analysis of proposed learning-based techniques is carried out informally and is occasionally supported by experimental evaluation.

Security assessment of learning-based approaches faces several challenges. The main theoretical hurdle is the lack of formal definitions of security in the context of data analysis. In contrast to privacy, for which several formalisms have been proposed, e.g., privacy-preserving data mining [18] or differential privacy [19], no formal connection to established security objectives is known for machine learning. From the practical perspective, the success of attacks against learning algorithms crucially depends on the amount of knowledge available to an attacker. Most of the previously reported successful attacks assume that the attacker has full knowledge of the learned model [20], [15], [16], [21], [17], [22]. It can, therefore, be argued that reducing the amount of knowledge leaked about the model, as well as a proactive response to potential exploitation of such knowledge should provide adequate protection against adversarial data manipulation.

Still, it remains largely unclear what an attacker may learn about a learning-based method deployed "in the wild" and how this information can be exploited. To investigate this problem, we present the results of a case study we performed on a real learning-based system, PDFRATE[1], an online service for detection of PDF malware [23]. For any submitted PDF file, PDFRATE provides a probabilistic estimate of its maliciousness. Our study addresses the case when an attacker attempts to evade detection by modifying the submitted PDF file so that its malicious functionality remains intact but the probabilistic score returned by PDFRATE is decreased.

We proceed by presenting two classes of evasion strategies suitable for several attack scenarios varying in the amount of knowledge available to the attacker. Since PDFRATE is a research system, its method and technical details are relatively well documented in the original research paper [23] and the accompanying technical report [24]. Based on this information, it is possible to partially reconstruct the features used for creation and evaluation of models, reproduce the training procedures and even independently obtain some of the training data[2]. To systematically explore the attacker's options,

[1]http://pdfrate.com/.

[2]One of the training datasets used by PDFRATE is publicly available for the research community.

we define an orthogonal set of evasion strategies reflecting various degrees of available knowledge, described in detail in Section II. The general idea of our evasion technique is based on insertion of dummy content into PDF files which is ignored by PDF renderers but affects the computation of features used in PDFRATE, as elucidated in Section V-B. Once we can influence a subset of PDFRATE's features, we develop algorithms for constructing attack instances, presented in Section V-C. In the experiments of Section VI, we evaluate the effectiveness of our strategies on a set of 100 malicious files randomly drawn from a dataset known to PDFRATE.

Our results reveal that even with the smallest amount of available information, i.e., an ability to freely modify one sixth and increment another one sixth of the features, our attacks reduce the classification scores of PDFRATE from almost 100% to the median of about 33%. Additional information about the classifier, such as the knowledge of its type (trivial) and possession of the training dataset (somewhat more difficult to obtain), further decreases the median score to about 28%.

We have analyzed the defense strategy suggested and evaluated by the authors of PDFRATE, although we do not know if it is deployed in the online system. The attack scenario of [23] assumes that the attacker instruments a small subset of informative features. It was shown that this attack can be effectively thwarted by including a small portion of the anticipated attack data into the training set. We reconstructed this attack and verified the effectiveness of the original defense strategy. However, such proactive defense turns out to be effective only against the precise "strain" of the attack. Whenever the executed attack does not match the anticipated one, the effect of the proactive defense essentially vanishes, and the detection accuracy falls below 10%.

Our contributions can be summarized as follows:

- We present a general model for practical assessment of security of learning-based detection techniques. This model enables systematic exploration of various kinds of information leaks exploitable by an attacker and is applicable to systems beyond PDFRATE that have a modifiable subset of features.
- We present two evasion attacks that can be staged against a deployed classification model in various scenarios.
- We demonstrate the first automated practical attack against a learning-based classifier deployed "in the wild" performed without knowledge of the learned model and entirely in problem space.
- We provide an open source software framework for all experiments carried out in our study for independent verification and extension of our results.

II. Evasion Attacks against Learning Systems

Any learning-based system which is deployed in a real-world environment and for which there exists a critical amount of economic, political or military interest is certain to attract the attention of individuals or groups striving to gain advantage by manipulating the system in order to influence its decisions. There are numerous examples of such activities. Besides the computer security applications mentioned in the introduction, potential scenarios for such manipulation include adversarial advertisements [25], spam detection [26], [27], recognition of writing style [28], plagiarism detection [29] and many others.

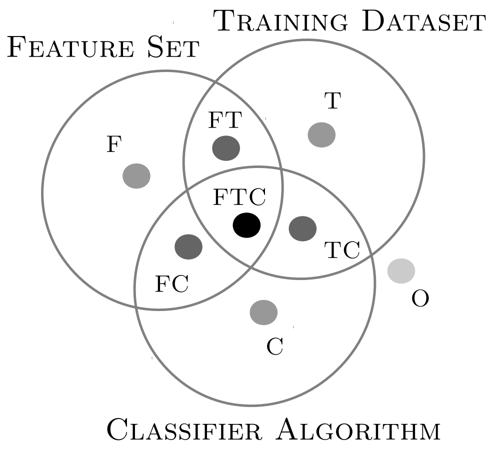

Fig. 1. Taxonomy of evasion scenarios for classifier systems. In every scenario, represented as a point, the knowledge about a given classifier component is high if the scenario point is within that component's circle, otherwise low.

In this work we focus on *classifiers*, a particular kind of learning systems, which classify new data into two or more predefined categories. Classifiers usually make predictions by computing some numeric or probabilistic score and comparing it with a fixed threshold. The goal of an adversary aiming to manipulate a classifier is to *confuse* it into providing a false classification. For binary classification problems, false classifications are called *false positives* and *false negatives*. From an adversarial viewpoint, the more information about a learning-based system is available, the higher the chances become that the system can be successfully gamed.

The essential components comprising every learning-based classifier system are:

- the **set of features** used by the classifier
- the **training dataset** used for classifier training
- the **classification algorithm** with its parameters.

It is, therefore, in the interest of the adversary to maximize their knowledge about the target classifier's components. For example, an adversary A who knows the feature set and training dataset of a certain classifier has a higher chance of evading it than an adversary B who only knows the feature set. In this sense, the two adversaries A and B are operating under different *evasion scenarios*. An evasion scenario is a problem setting for evasion from an adversary's point of view. It describes the classifier system information available to the adversary in a structured way: it outlines whether the adversary has a low or high amount of knowledge about the feature set, training dataset and classification algorithm.

To systematically explore evasion attacks against classifier systems, we propose the taxonomy of evasion scenarios, depicted in Fig. 1, based on the amount of knowledge adversaries possess about the three components of a classifier system.

Our taxonomy comprises 8 evasion scenarios. Their names describe the information available to the adversary. If any of the letters F, T or C, corresponding to the classifier components feature set, training dataset and classifier algorithm, respectively, is present in the name of a scenario, then the level of knowledge about the given classifier component the adversary has in this scenario is high, otherwise low. The scenario named O refers to the case when the adversary has low knowledge about all three classifier components.

In our taxonomy, high knowledge does not necessarily mean complete knowledge, and *vice versa*. There exist no strict criteria for deciding whether the knowledge level about a certain classifier component should be categorized as high or low. We consider the knowledge level high if it can be used to the substantial advantage of the adversary, otherwise low.

Our study is limited to the 4 evasion scenarios in which the level of knowledge about the feature set is high. Without the knowledge of features, the attacker is faced with a major challenge of either deducing them from observation of classification results, or otherwise to attempt to directly measure the sensitivity of a classifier to changes in the original data. We are currently not aware of any techniques for addressing these issues and therefore leave the investigation of scenarios with low feature knowledge for future work.

In the following subsections, we describe high-level algorithms for staging evasion attacks in the 4 scenarios of interest.

A. Scenario F

In scenario F, only the feature set is available to the adversary, to a varying extent. The adversary might be aware of some or all features, mistakenly consider obsolete features as being used, be capable of reading a subset or all features or be able to modify some or all features to a varying degree. Manipulation of a sufficient subset of features is, however, required in order to be able to modify samples and proceed with evasion.

An adversary with no knowledge about the classifier and training dataset may still perform evasion. If he has access to data samples, certified to be benign by the target classifier, he can try to align his malicious examples with known benign examples. This strategy is known as a *mimicry attack*. A particular implementation of this attack for PDFRATE is presented in Section V-C1. In general, mimicry attack is most effective if an attacker can submit probes to the target system during the course of attack, in order to ensure the benign classification of the source examples for the mimicry attack or to choose among multiple benign sources. However, online probing of a target system may be detectable and is therefore less desirable than a fully offline attack, in which only the final result is submitted to the target system.

An adversary that collects a sufficient amount of malicious samples, e.g., those found on the black market, may combine them with a collection of benign samples and thus build a *surrogate dataset*. This dataset can be used to train an off-the-shelf, *surrogate classifier*, which can then be evaded using a special-purpose attack tuned for this particular classifier. The rationale behind the surrogate classifier attack is that the inference of predictive models is based on general statistical properties which are shared among many learning methods. Hence, it is quite likely that one can approximate an unknown classifier with a suitable proxy classifier whose behavior can be controlled by the attacker. The effectiveness of this strategy critically depends on the quality of the data available to the attacker. If surrogate data is a realistic sample of the true distribution of the training data, one can expect the resulting attack to be effective against the original, unknown target classifier. The attack based on the surrogate classifier can be performed offline, with only the final result submitted to the target classifer.

B. Scenario FT

This scenario enables the adversary to take advantage of the knowledge the target classifier's training dataset, in addition to the known features. The dataset may be fully or partially leaked, enabling more accurate decisions in the process of generating a successful attack sample.

Knowledge of the benign training points enables the adversary to generate evasion samples which closely mimic them, using the mimicry attack, thereby increasing the chances of a successful attack in comparison to scenario F. Training a potent classifier on the *original* dataset creates a surrogate classifier that better approximates the target classifier than the on trained in scenario F, again opening up the way to the use of tailored methods for evasion of the surrogate classifier. Knowledge of training data enables the attacker to perform the entire attack offline before submitting the final result.

C. Scenario FC

In Scenario FC, the adversary knows the feature set and some details about the classifier, such as its type, parameters or the specific implementation. An adversary with no information about the training dataset at all and without a surrogate dataset has little advantage of knowing the classifier. With a surrogate dataset they can train a surrogate classifier of the right type, yet the accuracy of this approximation depends on the quality of the gathered data. This attack can also be performed offline, similar to other attacks based on surrogate classifiers.

D. Scenario FTC

The adversary has the best chance of evading the target classifier if he knows the details of all three classifier components. In that case, he can fully reproduce the online classifier in an offline setting, submitting the attack results only when a sufficiently good evading sample has been found. An offline mimicry attack or an offline classifier-specific attack that defeat the offline classifier have a strong probability of defeating the online one as well.

Before presenting the target system PDFRATE and the specific algorithms used to implement the abovementioned general attack scenarios, we give a short overview of the Portable Document Format (PDF) in the following section.

Header	`%PDF-1.4`

Body	`6 0 obj`
	`<</Length 7 0 R`
	`/Filter /FlateDecode>>`
	`...`
	`/Title(An Article)`
	`/Author(NA)`
	`>>endobj`

Cross-reference table	`xref`
	`0 317`
	`0000000000 65535 f`
	`...`
	`0000375296 00000 n`

Trailer	`trailer`
	`...`
	`startxref`
	`377137`
	`%%EOF`

Fig. 2. An example PDF file (detail).

III. The Portable Document Format

The *Portable Document Format (PDF)* is a file format that enables creation of documents that render and print consistently, independent of the underlying environment, and is published as an open standard, ISO 32000-1:2008 [30]. A PDF file, as depicted in Fig. 2, consists of a *header* with the PDF magic number and format version, *body*, a set of PDF objects that comprise the structure and content of the file, the *cross-reference table* (CRT) that indexes the objects in the body and *trailer*, pointing to the CRT. The beginnings of the header, CRT and trailer are denoted by keywords `%PDF`, `xref` and `trailer`, respectively. Objects in the file body are introduced with the keyword `obj` and a pair of integer identifiers. Objects can be of different types, e.g., *numbers*, *strings*, *names* (identifiers), *dictionaries* (sets of key-value pairs where key is always a name object and value can be any object type, including another dictionary), *indirect references* ("pointers" to other objects), *streams* (dictionaries with additional encoded and/or compressed content) etc. ISO 32000-1 prescribes some dictionaries to carry special meaning, such as those whose `Type` is `JavaScript` (containing executable JavaScript code), `Metadata` (with information about the file such as its author, title, creation and modification dates, etc.) or `Page` (describing a single page). A special type of streams, called *object streams*, may contain other objects as part of the encoded and/or compressed stream contents.

The header, body, CRT and trailer constitute the *file structure* of a PDF file, i.e., the content of the file's bytes that can be directly read by software agnostic of the PDF format. The objects in the file body form a graph-like logical structure, called the PDF *document structure*, by means of indirect links to other objects or their direct embedding. The appearance of pages is described by *content streams*. However, we limit our interest at the file structure level because PDFRATE does not parse PDF files, i.e., it only reads their raw bytes.

IV. PDFRate

PDFRATE employs the Random Forest algorithm to classify PDF files into benign or malicious based on their metadata and certain structural features[3]. The following subsections provide an overview of PDFRATE's features, classification algorithm, datasets and adversarial considerations; further details can be obtained from the original paper [23].

A. Features

PDFRATE employs a total of 202 integer, floating point and boolean features. 135 of those were described in [24], the rest remain unknown. A subset of features are shown with their values for one specific file in Fig. 4, Section V-B. The features reflect various properties such as size and version of the file, character counts of PDF metadata items such as author name, creation and modification date, structural properties such as the count of Acrobat forms and their relative positions in the file, etc. All features were manually defined by the authors and selected for best classification performance and robustness against adversaries, respectively. The features are extracted by running a set of regular expressions on raw bytes of the PDF file. By not performing proper PDF parsing, authors of PDFRATE have consciously given preference to speed and simplicity rather than completeness and correctness, as some of the features might lay in encoded and/or compressed object streams, beyond the reach of regular expressions.

The features exhibit significant interdependence. When one feature's value is modified, many others may be affected because they directly or indirectly depend on the targeted feature. For example, by modifying the number of lower-case characters of the `Author` metadata field (`author_lc`), the related feature `author_len` will be affected, but so will less directly related ones such as file size (`size`). A change in size triggers further changes of seemingly completely unrelated features `pos_acroform_*`, that denote the relative file offset of one or more keywords `AcroForm`. Feature interdependence makes the adversarial control of feature values difficult.

B. Datasets

Three datasets were involved in the creation and evaluation of PDFRATE. Three models have been trained on them, which are used separately to assess new data submitted by users.

Two of the three datasets were used in the experimental evaluation of PDFRATE presented in [23]: *Contagio* and *Operational*. The *Contagio* dataset is a collection of malicious and benign PDF files contributed by malware researchers, available for download[4]. Training of PDFRATE was carried out on a subsample of the *Contagio* dataset containing 5,000 benign and 5,000 malicious files[5]. The trained classifier was evaluated on the *Operational* dataset comprising 100,000 PDF files collected "on a large university campus". Presumably, the same dataset was used to train the model currently available as the *George Mason University (GMU)* used by PDFRATE.

[3] PDFRATE's structural features describe physical rather than logical structure, and are not to be confused with the PDF document structure.

[4] The Contagio archives are available at the following URL: http://contagiodump.blogspot.de/2010/08/malicious-documents-archive-for.html.

[5] A list of MD5 sums of those files was published: http://pdfrate.com/contagio_md5_class.csv.

The last dataset, *Community*, was created from files submitted and rated by PDFRATE users and was not used in its original evaluation.

C. Classification Algorithm

PDFRATE employs Random Forest [31], an ensemble learning method comprising a number t_{RF} of independently trained decision trees. In the training step, every tree is learned using CART methodology, but using only a subset of the available training samples. A different subset is generated for every tree by randomly sampling a fixed number of times from the training data, with replacement – a procedure called bootstrap aggregating or bagging. When a new decision node is added to a tree, only a randomly chosen subset of f_{RF} features is considered, where f_{RF} is less than the total number of features. A decision is made by majority voting among all decision trees on a given new data point. Random forests are known for their excellent generalization ability and robustness against data noise. PDFRATE uses the R port of Leo Breiman's and Adele Cutler's original Random Forest implementation, available as the package RANDOMFOREST[6]. t_{RF} and f_{RF} are parameters of RANDOMFOREST called $ntree$ and $mtry$, respectively. The values $ntree = 1000$ and $mtry = 43$ are used by PDFRATE.

All three classifiers deployed by PDFRATE, i.e. the ones trained on the *Contagio*, *GMU*, and *Community* datasets, produce as their result the output of their decision function, i.e., a real value in the interval $[0, 1]$ denoting the percentage of decision that have labeled the submitted file as malicious. There is no threshold given in [23] determining at what percentage should a file be considered malicious. Note that by providing this percentage value instead of a binary decision, PDFRATE reveals much more information about its classification engines than it is necessary for decision-making and thus enables the adversaries to make more informed decisions when developing their evasion methods.

D. Adversarial Considerations

Before describing our attacks, we discuss the properties of PDFRATE crucial for the adversarial setting of our study.

In our evaluation, we are only concerned with the evasion of the *Contagio* classifier. We do not consider the *GMU* and *Community* classifiers because their training datasets were unavailable to us and hence we could not evaluate the full spectrum of attack scenarios defined in Section II. Besides being freely available, the *Contagio* dataset seems to remain static. Periodic retraining, an important security measure, would have complicated the consistent evaluation of effectiveness of our evasion methods, as every classifier update would have rendered previous results outdated. Furthermore, although PDFRATE provides a second level of analysis by classifying malicious files into "targeted" and "opportunistic", our study is limited to evading the initial binary classifier.

From an adversarial perspective, the level of knowledge available to attackers about PDFRATE is high. The availability of its feature definitions facilitates the creation of manipulated samples. Although robust against data noise, the Random Forest classifier was not designed for resilience against adversarial noise. Periodic retraining is also not carried out in the deployed system. These weaknesses make PDFRATE an excellent candidate for our case study. Other, more prominent machine-learning-based malware detectors have features which are either unknown or much more difficult to control.

Despite its weaknesses, adversarial considerations were indeed present in PDFRATE's inital design. The attack model considered in [23] assumes that the adversary knows the means and standard deviations of the 6 most important features, i.e., those on top of the list of variable importance measures of the Random Forest model, for the benign training files. The adversarial model assumes that an attacker can create camouflaged malicious samples in which a subset of top features is set to random values drawn from the normal distributions with the given means and standard deviations characterizing the benign samples. This attack will be referred to as "benign random noise" (BRN). It was shown in [23] that the BRN attack can severely degrade the detection accuracy of the classifier. To counter this attack, a proactive defense strategy was proposed: to modify a subset of malicious data points in the training set in exactly the same way as an attacker would proceed. This simple defense strategy proved to be surprisingly effective.

The BRN attack was implemented synthetically, i.e., by modifying the top 6 features *directly in feature space*. Therefore, it does not address the issue of whether real PDF files can be generated with the required feature vectors. Due to strong feature interdependencies, such an assumption is unrealistic in practice. In our evaluation of the defense mechanisms presented in Section VI, we depart from the feature space and evaluate this attack using real PDF files. Furthermore, we investigate the robustness of the proposed countermeasure against our own mimicry attack.

V. METHODOLOGY

Since our study of evasion scenarios assumes a stealthy attacker, the key elements of our methodology involve reimplementation of the methods deployed by PDFRATE. We first reconstructed a subset of PDFRATE's features using the available public knowledge. The next step was to develop a technique for manipulation of PDF files which affects the selected subset of features. The last step in our methodology was to design attack algorithms for carrying out the generic attack strategies presented in Section II.

The above techniques and methods were implemented in our experimental evasion framework called MIMICUS. The framework consists of a Python module which supports feature extraction, PDF file modification, upload to PDFRATE and score retrieval, training of classifiers and performing attacks against them. MIMICUS is free and open source software, suitable for extension with other attacks and attack targets. It is available for download[7], bundled with all training data (as

[6] http://cran.r-project.org/web/packages/randomForest/index.html

[7] MIMICUS – https://github.com/srndic/mimicus.

feature vectors), classifier models and code required to fully reproduce our experimental results. All attack files used in our experiments can be obtained from the Contagio database.

A. Reimplementation of PDFRATE*'s Features*

Our four evasion scenarios have one common assumption: the adversary knows the features of the attacked system. The level of knowledge about particular features may, however, vary widely. The attacker may not be aware of some features' existence at all. Even for features with known description, the attacker may have partial or no control of their values. Finally, interdependence between features prevents the attacker from arbitrary manipulation of their values.

The knowledge about PDFRATE's features comes from three sources: the original research paper [23], the technical report [24], and the behavior of PDFRATE as deployed online. As stated in [23], a total of 202 features are employed by PDFRATE. However, only 135 of them are described in [24], to a varying extent. This limits the set of features potentially under attacker's control to roughly two thirds of the reported number. Furthermore, it cannot be ruled out that the deployed system does not have a different set of features compared to the reported ones due to a natural progress in development.

As a first step, we reimplemented the extraction of 135 known features by following the general guidelines on feature extraction from [24]. Subsequently, regular expressions were developed for each feature except for `size`, which was read directly. During this process we also examined metadata output produced by PDFRATE for a fixed test suite comprising PDF files with a broad range of values for many features. Values of some features, e.g., counts of `Page` or `obj` keywords, can be accurately deduced from the metadata output. The regular expressions were further refined until consistent behavior was achieved across all test files. Although the reimplementation process required time-consuming expert work, it would be a small hurdle for an incentivized adversary.

Thanks to the availability of the *Contagio* dataset, we were able to verify the correctness of our reimplementation by comparing our classification results on that dataset with those reported in [23]. We, furthermore, verified that despite the discrepancy in the set of implemented features, our local clone of PDFRATE produces similar classification scores as the online system on a benchmark dataset presented in Section VI-A3.

B. Modification of PDFRATE *Feature Values*

The development of the PDF file modification method for our study was guided by the following design goal: once modified, the file in question has to appear indistinguishable from the original to any PDF parser, yet reliably affect PDFRATE's feature extraction. The reason for this is that such a semantics-preserving method can be safely applied to malicious PDF files in our experiments, regardless of the diverse vulnerabilities they may exploit, without the risk of breaking their potentially subtle *modus operandi*.

The feature modification component of MIMICUS can arbitrarily modify values of 35 and increment values of 33

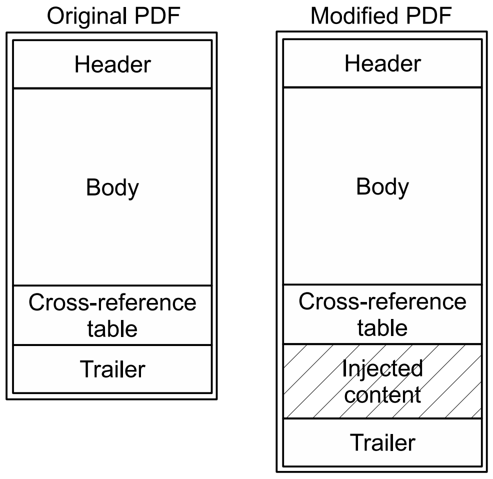

Fig. 3. The PDF modification method takes the original PDF (left) and injects new content between the cross-reference table (CRT) and the trailer. Such a modified file (right) confuses PDFRATE into accepting the newly-injected content as part of the file, while the PDF readers jump from the trailer directly to the CRT, skipping the injected content completely.

features of PDFRATE, as detailed in Appendix A. Modification of further features would have required delicate changes to the structure of PDF files, increasing both the implementation effort and the risk of breaking the malicious functionality.

Our approach to file modification was motivated by the discrepancy between the operation of PDF readers and PDFRATE. This approach was described in [32] as an example of a *semantic gap* in the interpretation of various file formats. PDFRATE evaluates a set of regular expressions over the raw bytes in a PDF file, reading *from the beginning to the end* of the file. In contrast, PDF readers parse PDF files in adherence to the PDF syntax prescribed by ISO 32000-1. A conformant PDF reader reads a file starting from its end. It checks the trailer to find the location of the cross-reference table (CRT) and then *jumps* directly to it in order to locate the objects in the file body. This difference is illustrated in Fig. 3, showing the layout of a PDF file before and after our modifications.

Our solution exploits this semantic gap: as long as the file header, body and CRT are not modified or moved, the trailer can be moved arbitrarily far away from the CRT[8], thereby generating an empty space in the file where arbitrary content can be injected. Such content will be processed by PDFRATE, but PDF readers will *always ignore it*.

The described *content injection approach* leaves behind file modifications which are trivial to detect if one knows what to look for. We believe that it is possible to rewrite the PDF files, modifying the content in-place instead, thereby concealing the modifications alltogether. However, this approach would not come without technical complications. It could potentially affect PDF readers by breaking the rendering of the PDF file and might negatively impact the reliability of the embedded exploit – problems which content injection avoids completely.

Our modification method proceeds by injecting a set of whitespace-separated string patterns into the gap between the

[8] It is only important that the trailer remains at the end of the file.

CRT and the trailer of the target PDF file. The patterns are crafted to make specific PDFRATE regular expressions match them, thereby influencing the extracted feature values. For example, injecting into a file with 5 `obj` keywords the string "`obj obj`" will change its `count_obj` feature value from 5 to 7, as PDFRATE's regular expressions will match them all. As another example, the length of the `Author` metadata field can be "reduced" from 10 to 3 by injecting a new `Author` field with 3 characters, "`/Author(abc)`", as PDFRATE tends to only take into account the content of the last metadata field in the file. By injecting our payload just before the trailer we can ensure that this condition is fulfilled.

Using the described modification method it can be safely assumed that the behavior of PDF readers will not be altered[9], but PDFRATE would be tricked into reading the desired feature values from the modified file. Our experiments have confirmed this behavior for two PDF readers, ADOBE READER and EVINCE. In addition, we have submitted all malicious files involved in our evasion experiments to WEPAWET [3] before and after modification and verified that the exploit effectiveness was not affected for any sample.

As already mentioned, the features of PDFRATE are heavily interdependent, i.e., it is, in general, impossible to perfectly translate data points in feature space into files in problem space. Given a malicious file before the attack, F_B, and a data point P_A generated by the attack algorithm run on F_B, the adversary wants to generate the attack file F_A that optimally defeats the classifier by modifying F_B's feature values to P_A. However, due to feature interdependence, the resulting file $F'_A \neq F_A$, has different features, which may or may not defeat the classifier. Fig. 4 shows a concrete example using the GD-KDE attack, described in Section V-C2. Feature `pos_acroform_min` denotes the relative file offset of the first occurence of the keyword `AcroForm` and is not modifiable by MIMICUS, however, it was *indirectly* influenced by the increase of the total file size. On the other hand, although feature `author_len`, denoting the length of the `Author` metadata field, is directly modifiable, it got the value 11 instead of the desired 0 because other modifiable features, i.e., `author_lc`, `author_num`, `author_oth` and `author_uc`, denoting different character classes in the `Author` field, drove the total character count to 11.

Another important consideration regarding the translation of data points from the feature space into the problem space is that algorithms operating in the feature space may construct data points which are not feasible in the problem space. Examples are the `size` and `version` features to which the attack algorithms attempted to assign negative values. It is neither feasible to enumerate all feature interdependencies and account for their effects *a priori*, nor to identify invalid data points before translation to problem space.

Our approach to dealing with these two limitations is opportunistic: we generate the file from the feature vector

FEATURE	BEFORE	AFTER	FILE
author_lc:	0	2	2
author_len:	0	0	11
author_num:	0	3	3
author_oth:	0	5	5
author_uc:	0	1	1
count_acroform:	1	0	1
count_endobj:	11	918	465
count_endstream:	1	169	85
count_eof:	0	2	2
count_font:	0	86	86
count_image_large:	0	1	1
count_image_small:	0	6	6
count_image_total:	0	0	11
count_image_xsmall:	0	4	4
count_javascript:	3	0	3
count_obj:	14	922	922
count_objstm:	0	28	28
count_page:	0	29	29
count_stream:	1	169	85
count_trailer:	1	0	1
count_xref:	1	0	1
createdate_ts:	-1	7.52e+8	7.52e+8
image_totalpx:	0	0	813898
moddate_ts:	-1	1.0e+09	1.0e+09
pos_acroform_avg:	0.07043	0.07043	0.00716
pos_acroform_min:	0.07043	0.07043	0.00716
pos_acroform_min:	0.07043	0.07043	0.00716
size:	2726	-426760	26782
version:	0	-4	0

Fig. 4. Changes of feature values for a subset of features in an example GD-KDE attack in scenario F. The *BEFORE* column shows the feature values extracted from a malicious candidate file, F_B, with the SHA-1 hash a39cf14b806db14a9e877b665324d203e5a5a666. GD-KDE transformed these values in feature space into data point P_A (*AFTER*). Point P_A was used to modify file F_B in file space and generate the attack file F_A. However, feature interdependence caused the file F'_A to be generated instead, with slightly different feature values (*FILE*).

by translating features one by one, independently from each other and without accounting for the limitations, in the hope that the resulting file's features are not *too far away* from the desired values. Although this approach results in hardly predictable outcome, the resulting files have feature values sufficiently close to the desired ones and are suitable for evasion. As a concrete example, compare columns *AFTER* (desired outcome) and *FILE* (actual outcome) of Fig. 4.

Additional safety mechanisms implemented in MIMICUS prevent feature modification if the desired value is outside of valid bounds specific to the feature and the file, e.g., if there was an attempt to modify `size` to a positive value less than the file already had. The specific lower and upper bounds enforced by our method were collected by enumerating the features of all of the files in the dataset of PDF files available to the adversary, described in Section VI-A, and extracting the minimum and maximum values for each feature. Another reason for preventing feature modification is a feature data type mismatch, e.g., when a data-type-agnostic algorithm wants to set a boolean feature to 7. In the end, the result is a valid PDF file with features close to the desired ones, suitable for evasion.

C. Attack Algorithms

The second major component of MIMICUS are its attack algorithms. Their main goal is to generate PDF files whose

[9] Provided that they do not parse the injected content, but perform the direct jump prescribed by ISO 32000-1 instead.

feature vectors are likely to receive low classification scores. To this end, we have adapted two previously known methods to the specific context of PDFRATE's features.

1) Mimicry Attack: The mimicry attack is well-known in the security literature. Its idea is to transform a malicious sample in such a way that it mimics a chosen benign sample as much as possible, making the resulting mimicry sample harder to detect. This attack is simple to implement, can be applied to any classification algorithm, and does not necessarily depend on a particular learned classifier model. Therefore, it is suitable for evaluation in every evasion scenario. Our implementation takes a malicious file and simply attempts to modify all of its modifiable features at once to take on the values of the features of a chosen mimicry target, a benign file. To increase the effectiveness of a mimicry attack, we repeat it 30 times using different benign targets for every attack file. The resulting 30 files are evaluated using a local classifier, and only the sample which best evades the local classifier is submitted to PDFRATE.

Due to the existence of undisclosed features and technical limitations discussed in Section V-B, it is impossible to generate a file which exactly corresponds to the feature vector resulting from a mimicry attack. It is important that the conversion of a feature vector into a file is performed after the mimicry is complete in the feature space. The latter is technically straightforward: we simply merge a malicious feature vector into a chosen benign one while protecting existing values. Modifying features one at a time while translating them into a file is not a good strategy, as the interdependency between features dominates the transformation and generates a lot of uncontrollable changes. Using a single-step transformation makes such interdependency less prominent.

The generality of the mimicry attack, i.e., its independence of the specific learning algorithm and the underlying dataset, makes it applicable to other learning-based systems which, like PDFRATE, have a known and modifiable subset of features.

An inverse attack, performed by injecting malicious content into a benign PDF file, was described in [33] and demonstrated to be effective against PDFRATE in a small-scale experiment.

*2) Gradient Descent and Kernel Density Estimation (*GD-KDE*) Attack:* The second attack evaluated against PDFRATE is based on a method employing gradient descent and kernel density estimation (hence in this paper we call it GD-KDE) to defeat a classifier with a known, differentiable decision function [22]. It requires the knowledge of a specific learned model and a set of benign samples. Additionally, because it is based on gradient descent, it is only applicable to differentiable classifiers, such as SVM, artificial neural network, etc., and cannot be applied to the Random Forest classifier. Hence, the GD-KDE attack is applicable only to scenarios with differentiable surrogate classifiers (F and FT).

The GD-KDE algorithm proceeds by following the gradient of the weighted sum of the classifier's decision function and the estimated density function of benign examples. The starting point of the gradient descent is the feature vector of the malicious sample. The starting sample is usually correctly classified as malicious; the goal is to move to the area where the classification algorithm classifies points as benign. In order to avoid moving to infeasible areas of the feature space with negative classifications, the algorithm's objective function has the second term, the density of benign examples. This ensures that the final result lies close to the region populated by real benign examples. The density function must be estimated beforehand, using the standard techniques of kernel density estimation [34]. Similarly to the mimicry attack, we run GD-KDE in the feature space to completion before transforming the result into a file.

VI. EXPERIMENTAL EVALUATION

The experiments to be presented in this section assess the effectiveness of evasion techniques presented so far. In our evaluation protocol, we take on the role of an attacker and combine all available means to defeat an up-to-date version of PDFRATE as it is deployed. An attacker has no control over PDFRATE's deployment, hence no guarantees can be provided that the system has not changed between individual experiments. Since our evaluation was carried out against the model trained on a static dataset and took place within one week, it is quite unlikely that any changes in the production system have occurred.

In another set of experiments, we also investigate the impact of our attack on the defensive measures suggested in the original paper [23].

A. Datasets

Three datasets were used in our experiments: two datasets, *Contagio* and *Surrogate*, were intended for training of local classifiers needed for attack implementation, while the *Attack* dataset consisted of malicious files used as starting points for generating attack samples targeting PDFRATE.

1) Contagio dataset: This dataset is an exact copy of the original PDFRATE training dataset, described in Section IV-B. It contains 5,000 benign and 4,999 malicious PDF files[10]. It is reasonable to assume that an adversary knows that this dataset was used for training and obtains access to it.

2) Surrogate dataset: This dataset is designed as a dataset that an adversary without acces to *Contagio* data might have collected to approximate it. Malicious files in the dataset are a random subsample of PDF files uploaded to the online virus-scanning service VirusTotal[11] between the 5th and 22nd of March, 2012. These files are newer than *Contagio* data but were known before PDFRATE was published. Four files in the dataset were found to be present in the *Contagio* dataset and were removed to ensure strict complementarity of data. Benign files in the *Surrogate* dataset are randomly subsampled from the files obtained using keywordless Google web searches for PDF files published between February 5, 2007 and July 25, 2012. The *Surrogate* dataset has the same size and composition as the *Contagio* dataset.

[10] We were unable to locate the malicious file with the MD5 hash 35b621f1065b7c6ebebacb9a785b6d69 in the archives.

[11] VirusTotal – https://www.virustotal.com/.

3) Attack dataset: This dataset contains 100 malicious files that are used as starting points for all attacks. This dataset was deliberately chosen to be small in order to minimize operational impact on PDFRATE. The adversary has access to these files in all scenarios. The files were randomly drawn from the *Contagio* dataset, already known to the classifier and therefore make evasion even more challenging to the attacker. Both PDFRATE and WEPAWET classify all of them as malicious, PDFRATE with a very high score, as seen in Fig. 5, "Baseline". All files are distinct in both the problem and feature space.

B. Classifiers

Depending on whether the attacker knows the exact classification algorithm employed by PDFRATE or not, he might use either the original or an unrelated, surrogate classifier. Our experimental framework MIMICUS models these two cases by deploying the Random Forest classifier in scenarios FC and FTC, where the classifier type is known, and a Support Vector Machine (SVM) classifier in scenarios F and FT.

1) Random Forest: The classifier implementation and parameters are identical to the original classifier of PDFRATE described in Section IV-C.

2) Support Vector Machine: We have chosen the SVM [35] as a surrogate classifier because it delivers high classification performance on many problems, including the discrimination between malicious and benign PDF files using PDFRATE features, and is unrelated to the Random Forest. The SVC implementation of the SCIKIT-LEARN [36] machine learning toolkit version 0.13.1 was used.

The SVM learns by mapping labeled training points into a high- or infinite-dimensional feature space, optionally applying a nonlinear transformation called a *kernel function* to the input feature vectors to make them better separable. It then finds a separating hyperplane in the new space with the largest possible margin, i.e. the distance between the convex hulls of the two classes of points, malicious and benign. The hyperplane vector, represented by a subset of input points, so-called *support vectors*, together with their weights, constitutes an SVM model. When classifying new data, the distance between the hyperplane and the new data point is calculated. This distance, called the *decision function* score, is a real value whose meaning is similar to the classification score of PDFRATE. A binary decision can be made by taking the sign of the decision function score for the new data point. We assign the positive score to the malicious class by convention. To evade an SVM, the adversary needs to modify a malicious data point so that its decision function score changes sign.

Two kernel functions were evaluated: linear and RBF. The linear kernel $k_{linear}(x_1, x_2) = x_1 \cdot x_2$ provides a linear transformation of two input vectors x_1 and x_2, while the RBF kernel $k_{RBF}(x_1, x_2) = exp(-\gamma||x_1 - x_2||^2)$ utilizes the Gaussian radial basis function as a nonlinear transformation of its arguments. For optimal evasion results, a grid search was carried out on the two adversary's datasets, optimizing the SVM parameters C for the linear, C and γ for the RBF kernel. The highest achieved accuracy using 10-fold cross-validation and a 60%:40% training-testing split was 98.7% on the *Contagio* and 99.5% on the *Surrogate* dataset.

TABLE I
REALIZED ATTACK SCENARIOS

Scenario	Classifier	Dataset	Attack(s)
F	SVM	*Surrogate*	Mimicry, GD-KDE
FC	Random Forest	*Surrogate*	Mimicry
FT	SVM	*Contagio*	Mimicry, GD-KDE
FTC	Random Forest	*Contagio*	Mimicry

The parameters found in the grid search are same for both datasets: RBF kernel, $C = 10$, $\gamma = 0.01$. However, the generalization ablility of the two learned models for the two datasets differs greatly. The SVM trained on *Contagio* data achieves an accuracy of 98.5% on *Surrogate* data, but the SVM trained on the *Surrogate* data performs very poorly on *Contagio* data, with an accuracy of 61%. Of course, the adversary in scenarios F and FC, without access to original training data, would be unable to check how well its SVM performs on it. Bad approximation of the training data strongly affects the performance of the GD-KDE attack.

Due to differences in scale among features and as a general rule when using SVMs, *feature standardization* was performed on extracted data points by subtracting the feature mean from the feature value and dividing the result by the feature's standard deviation. Means and standard deviations of all features were calculated on the *Contagio* dataset.

C. Attack Scenarios

The following subsections elucidate how the 4 main attack scenarios described in Section II were implemented in our experiments using the available data and algorithms. A summary of the realized attack scenarios is presented in Table I.

1) Scenario F: In scenario F, all that the adversary knows about PDFRATE is how to read 135 and modify 68 features. Nevertheless, there are two attacks he can perform, depending on the available datasets, as elaborated in Section II-A. The mimicry attack uses randomly sampled benign files from the *Surrogate* dataset, as the *Contagio* dataset is unknown. Similarly, the surrogate classifier is trained on the *Surrogate* dataset for evasion using GD-KDE. The classifier parameters are optimized using grid search on the *Surrogate* dataset. Both attacks were performed offline, without classifier feedback, and their results were uploaded to PDFRATE for evaluation.

2) Scenario FT: In scenario FT, besides the limited knowledge of features, the adversary has a complete knowledge of the training dataset. Therefore, the *Contagio* dataset is used to train a surrogate classifier for the GD-KDE attack in this scenario, and randomly sampled benign files from *Contagio* are used as mimicry targets for the mimicry attack. This time, the surrogate classifier is optimized using grid search on the *Contagio* dataset. Only the final attack results are submitted to PDFRATE.

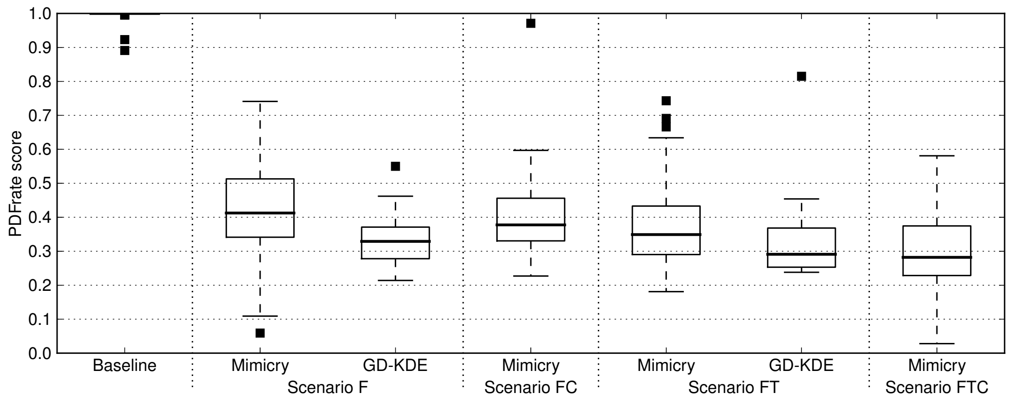

Fig. 5. Populations of PDFRATE scores before ("Baseline") and after each attack (the rest), for all 100 attack samples from the *Attack* dataset. Attacks are grouped by scenario. The boxes extend from the first to the third quartile, with the median value between them (thick line). The whiskers extend to the farthest datum within 1.5 times the interquartile range from the box, while the squares represent the outliers.

3) Scenario FC: Knowledge about the classifier is added to the limited knowledge of features in this scenario. The adversary knows the original classifier, its implementation and parameters. They use the *Surrogate* dataset with the original classifier to produce a surrogate classifier, which they evade offline using mimicry attack, with mimicry targets randomly selected from the *Surrogate* dataset. Results are submitted to PDFRATE for evaluation.

4) Scenario FTC: Given the limited knowledge of features and complete knowledge of the training dataset and classifier, the attacker creates a local clone of PDFRATE and evades it offline. Only the final attack results are submitted online.

D. Results

Before the attack experiments were run, all 100 files in the *Attack* dataset were evaluated by PDFRATE. The results of this evaluation, shown in Fig. 5, "Baseline", provide a baseline with which we compare the attack results. All but 3 files received a 100% malicious classification score.

Our evaluation followed a simple protocol. For every attack, 100 files from the *Attack* dataset were used to generate attack samples. The effectiveness of generated attack files was evaluated by submitting them to PDFRATE and comparing the received classification scores with our baseline. All attack samples were submitted to WEPAWET to verify their maliciousness after modification.

The summary of PDFRATE's scores for attack files is presented in Fig. 5. For each attack, the population of 100 classification scores is represented as a box plot, with the median shown as a thick line, the 25th and 75th percentiles ("interquartile range" or IQR) as a box, scores within 1.5 IQR from the median as "whiskers", and the remaining outliers as single points. Plots are grouped by attack scenario.

The results show that PDFRATE was evaded *in all 4 attack scenarios*. The median score dropped to 28-42% for mimicry and 29-34% for GD-KDE attacks, depending on the scenario. For all attacks except mimicry in scenario F, the 75th percentile of the box plot lies below the 50% mark, implying that 75% of the attacks would be classified as benign if a 50% threshold over classification scores were used for decision making. The significance of these results is further emphasized by the fact that only a third of features were modifiable, and the files used for evaluation were already known to PDFRATE at training and hence more difficult to evade.

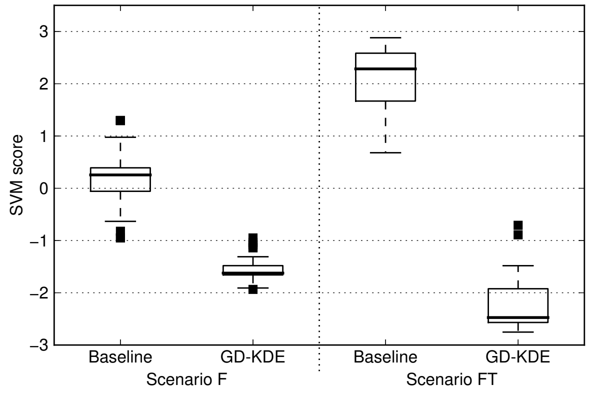

Fig. 6. Populations of SVM decision function values before and after GD-KDE attacks in scenarios F and FT, for all 100 attack samples from the *Attack* dataset. Parameters of the box plot are described in Fig. 5.

Results in scenarios with a surrogate classifier, F and FT, demonstrate the superiority of GD-KDE over mimicry. Furthermore, the mimicry attack in the scenario with the highest amount of knowledge, FTC, only marginally outperforms GD-KDE in scenario FT. Further insights into the behavior of the GD-KDE attack are given in Fig. 6. It shows the values of the decision functions of two SVMs, one trained on the *Surrogate* dataset in scenario F, the other on the *Contagio* dataset in scenario FT, before and after attack. The post-attack SVM scores demonstrate that the GD-KDE attack reliably steers *all* samples far across the decision boundary into the benign region. If the SVM classifier were deployed by PDFRATE, it is very likely that the GD-KDE attack would have attained perfect evasion, driving all scores below zero. Since the

SVM only approximately matches the decision function of a Random Forest, attacks against PDFRATE fall far from being perfect, but still significantly decrease the scores.

By careful observation of Fig. 6 it is evident that over 25% of pre-attack samples in scenario F have a negative decision function value, i.e., are classified as benign by the SVM (but not PDFRATE) *before* attack. This is a consequence of operating under scenario F, where the adversary trains using the *Surrogate* dataset but attacks using samples from the *Contagio* dataset. Because of the poor generalization of SVM models in this case, as elaborated in Section VI-B2, the samples are often misclassified. In scenario FT, where the attacker also trains on the *Contagio* dataset, the baseline SVM scores are strictly positive.

Another important observation based on Fig. 5 is the improvement of attack effectiveness with the increase of adversary's knowledge about the target system. This finding is in agreement with our initial conjecture about the importance of adversary's knowledge for classifier robustness. However, it is curious that the improvement from scenario F to scenario FTC is not as dramatic as one might expect: mimicry improves by around 14% and GD-KDE in scenario F is outperformed by the best overall attack, mimicry in scenario FTC, by a mere 6%. This is an important finding indicating that merely knowing a subset of features might provide the adversary more advantage than previously considered.

The possession of training data is the second most important contribution to the attackers' success, after the knowledge of features. Fig. 7 compares the scores of two local Random Forests with that of PDFRATE on mimicry attacks in scenarios FC, using *Surrogate*, and FTC, using *Contagio* data. It can be seen that on *Surrogate* data, the exact classifier makes an overly optimistic assessment of the attack effectiveness, achieving a median score of about 18%, while the same files get a median score of 37% when submitted to PDFRATE. However, when staged with the *Contagio* dataset, the local estimate of the attack score is almost identical to PDFRATE's (29% and 28%, respectively). This similarity is surprizing taking into account that the local classifier was trained using only a subset of PDFRATE's features and that the training process of Random Forests is heavily randomized.

As a final step in our evaluation, we investigate the impact of attacks on the detection performance of PDFRATE. Recall that the classification score still needs to be compared with some threshold for a binary decision to be made. Earlier, we reported that 75% of the attack points would have fallen under the radar if the threshold were set at the specific value of 0.5. To analyze the detection performance *for all possible thresholds*, the Receiver Operating Characteristic (ROC) curves are presented in Fig. 8 for the baseline and all attacks. The ROC curves were obtained on a mixed data sample containing the same 100 attack samples and all 1051 benign samples from the original Contagio database not in the *Contagio* dataset. It can be clearly seen from this figure that, especially in the lower range of false alarm rates (less than 0.5%), the detection performance of PDFRATE is dramatically decreased by the attacks, and the mimicry attack of the FTC scenario has caused a 7% false positive rate. The relative effectiveness of the attacks with respect to detection performance is similar to their relative effectiveness with respect to classification scores (cf. Fig. 5).

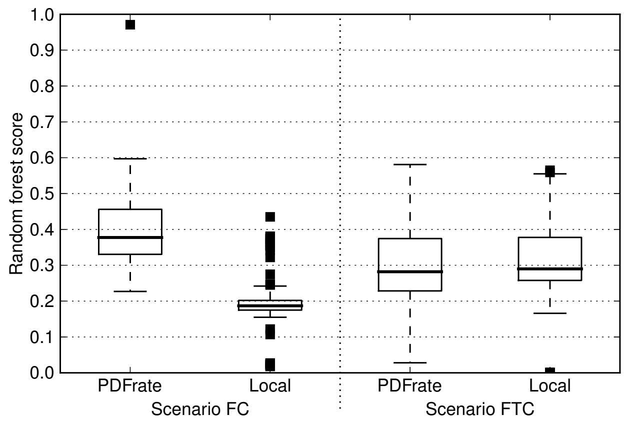

Fig. 7. Populations of Random Forest scores by PDFRATE and two local Random Forests on two mimicry attacks, for all 100 attack samples from the *Attack* dataset. Parameters of the box plot are described in Fig. 5.

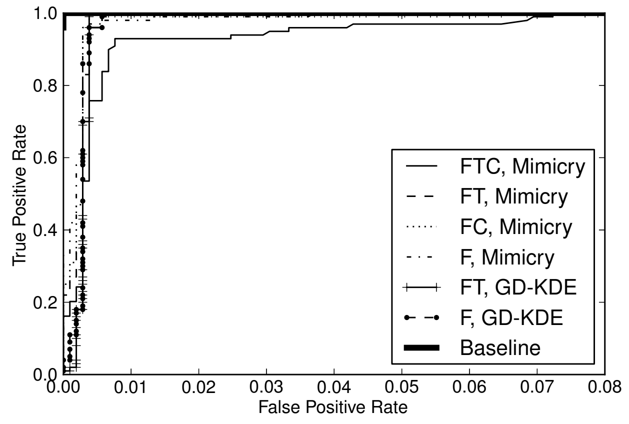

Fig. 8. ROC curves of the baseline and all attacks.

E. Defensive Measures

In our last experiment, we have investigated the robustness of defensive mechanisms proposed in [23] to our evasion technique. To set the baseline, we have reproduced the mimicry attack and the defense technique in exactly the same way as it was proposed by Smutz and Stavrou (cf. Section IV-D), using the Random Forest classifier trained on the *Contagio* dataset. Our classifier ranked the following 10 features[12] as most important, in descending order:

```
count_font          pos_eof_avg        count_endobj
count_js            pos_eof_max        producer_len
count_javascript    len_stream_min
pos_box_max         count_obj
```

[12] We have used 10 instead of 6 top features for mimicry because they were ranked significantly above others.

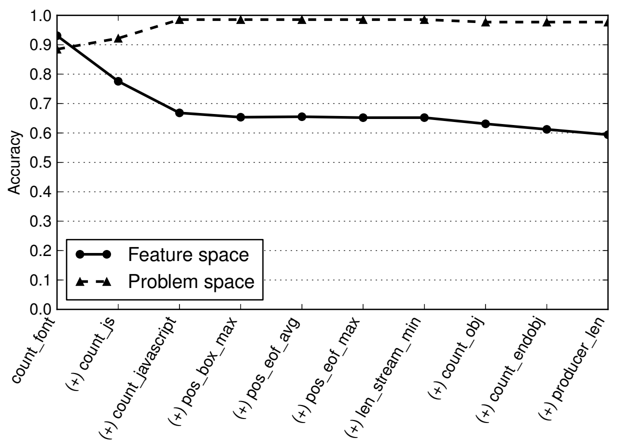

Fig. 9. Results of the BRN attack applied on data points in feature space versus files in problem space. The attack was applied 10 times, starting with only `count_font` and progressively modifying ever more features.

We have then reimplemented the original "benign random noise" (BRN) attack of Smutz and Stavrou in the feature space and extended it to the problem space by generating the target file for the attack's ultimate feature vector. The comparison of the effectiveness of these two attack variants as a function of the number of features modified is shown in Fig. 9. We observe that the behavior of the synthetic variant of the BRN attack (solid line) closely resembles the results reported in [23], with a slightly higher impact on accuracy[13]. However, when applied to files, the BRN attack is ineffective (dashed line). Only the modification of one or two features exhibits some impact on detection accuracy. Attempting to modify further features leads to increasing inadvertent modifications which end up steering the mimicry samples towards the benign class. Furthermore, only 5 of the top 10 features are modifiable. Therefore, the BRN attack in problem space is unpractical and, compared to other attacks presented in Section VI-D, suboptimal.

Finally, we have evaluated the effectiveness of the "vaccination" mechanism proposed by Smutz and Stavrou which modifies a fraction of malicious samples in the training dataset in such a way that they are more similar to *expected* attack samples. Two scenarios are considered: when the defender anticipates the *(1) right* and the *(2) wrong* kind of attack. Effectiveness of the vaccination defense against the BRN attack under both scenarios is compared in Fig. 10. Our experiment confirms the effectiveness of the vaccination defense when the right kind of attack, i.e., BRN, is anticipated (dashed curve). However, the classifier vaccinated with the BRN attack showed no resistance to our mimicry attack from the FTC scenario (dotted curve). Repeating the experiment with the vaccination using our mimicry attack revealed that the resistance to the attack was restored (Fig. 11). Hence it can be concluded that the vaccination mechanism is effective against any *correctly*

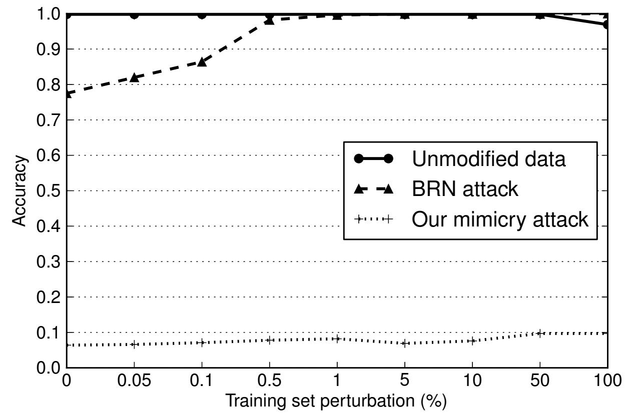

Fig. 10. Performance of the defensive measure proposed in [23]. Averaged results of 5 independent trials, using 10-fold cross-validation.

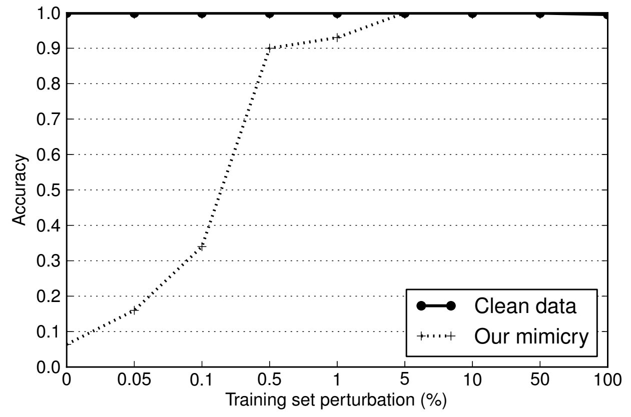

Fig. 11. Effect of the defensive measure proposed in [23] on our mimicry attack. Averaged results of 5 independent trials.

anticipated attack. The latter assumption, however, is rather unrealistic in practice.

VII. Interpretation of Attacks

From the operational perspective, it is crucial to understand which features contribute most to the success of the reported attacks. In general, interpretation of models created by learning techniques is always difficult. Even though Random Forest classifiers provide a ranking of features according to their informativeness, which has been crucial for the design of the BRN attack, this information is only indirectly related to the two types of attacks presented in our paper. Hence, a different analysis technique had to be developed to interpret our attacks.

Our interpretation is based on the binary difference between feature vectors before and after an attack. While it may be tempting to claim that the features with largest change are the most informative, this measure is strongly misleading in our case since the ranges of feature values are vastly different. Even re-scaling the changes to valid value ranges is not suitable since the bounds for specific features can only be determined on the basis of an empirical sample of PDF files

[13] The accuracy scores might vary because of experiment randomization, different Random Forest models or different cutoff values.

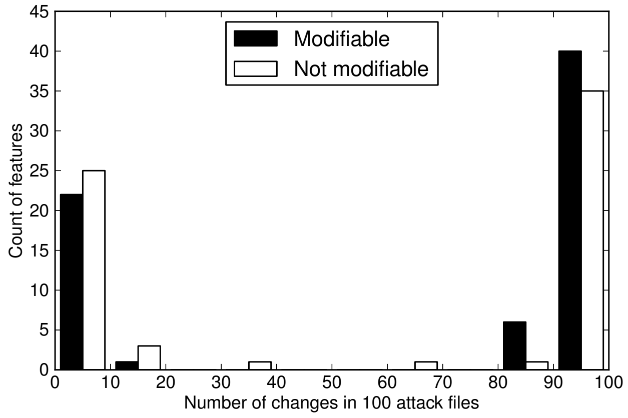

Fig. 12. Distribution of the count of features that were changed across 100 GD-KDE attacks in scenario FT.

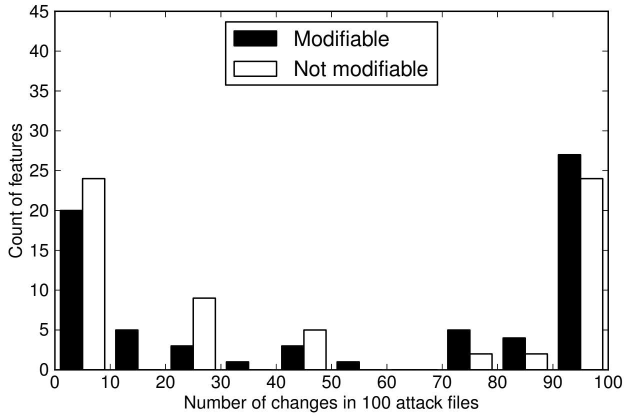

Fig. 13. Distribution of the count of features that were changed across 100 mimicry attacks in scenario FTC.

and are prone to outliers. Furthermore, only one third of the features is directly modifiable in our approach, yet all of them may be indirectly modified as a result of some other changes.

The only conceivable characterization of the mimicry attack is the empirical support of specific features, i.e., the percentage of files for which a given feature was changed by the attack. The histograms of feature support are shown in Figures 12 and 13 for the GD-KDE and mimicry attack, respectively. It can be seen that both attacks perform a significant amount of feature modifications, hence one cannot explain the attacks by a small number of essential features. Between the two attacks, the modifications produced by GD-KDE are more uniform, having a set of 45 features that are changed in almost every attack. The remaining 23 features are rarely modified, most likely due to the opposite direction of change (recall that 35 features are only incrementable in our setup) or due to the infeasibility of the requested change. The changes effected by the mimicry attack exhibit higher variability of support. It is also interesting to observe that direct modifications are accompanied by an almost balanced amount of indirect modifications. This serves as another example for the high interdependency of PDFRATE's features.

A practical way to interpret attacks is to observe concrete changes in feature values produced by the attacks. Although it does not scale to cases with many features and files, this kind of investigation provides deep insight into the *modus operandi* of the attack at hand. Fig. 4 shows how the features of one specific file changed in the GD-KDE attack. Recall that GD-KDE operates by steering malicious data points across the decision boundary into the benign area using gradient descent, and at the same time utilizes kernel density estimation to push them towards seen benign samples. This "benignization" is evident in the provided example. By comparing the *BEFORE* and *AFTER* columns, we see that the attack has added an author (`author*` features), set the creation (`createdate_ts`) and modification (`moddate_ts`) date into recent past, reduced JavaScript occurences (`count_javascript`), added some pages (`count_page`), fonts (`count_font`), images (`count_image*`), etc. – all changes towards the benign class. Some features, e.g., `size` and `version`, were changed to invalid values, possibly due to the influence of the gradient descent component.

VIII. Discussion and Related Work

While the results of our study are only applicable to a single system, our findings suggest several important implications. It was the first attempt to perform a comprehensive practical evaluation of a deployed learning-based system, hence we cannot expect that our results can be exactly reproduced for a large number of similar systems. Still, some key issues revealed by our experiments deserve careful consideration, as they pinpoint some general problems that need to be addressed in the design of future data-driven systems.

The main message of our experiments is that an attacker can significantly decrease the accuracy of a learning-based system if he has sufficient knowledge of its features and methods. The main factor that contributes to this insecurity is the knowledge of features. For PDFRATE we have observed that even the simplest attack from our arsenal, with no further knowledge of the system except for its features, can reduce maliciousness scores for a chosen representative set of malicious samples from 100% to the median of 33%. This was possible despite the fact that roughly one third of the classifier's features was completely unknown to us and another third not modifiable by our tools. Such an impact suggests that even a small amount of knowledge about the features can be exploited for staging evasion attacks. Additional factors such as the knowledge of the training data and the precise type of the algorithm are helpful but not crucial for the attack, as this information can be well approximated by surrogate sources.

The fundamental problem underlying the insecurity of learning-based approaches lies in the design of features. The growing popularity of machine learning in various kinds of information systems – far beyond security – is largely due to its ability to predict, with greater or lesser success, *causes* from *side effects*. It is this generalization ability that makes machine learning algorithms the means of choice for finding solutions to problems shrouded by uncertainty, when one has neither enough understanding of the problem to *design* a solution, nor can figure it out from looking at the raw data. The prevailing approach for designing features for learning algorithms by hand-picking a set of easily computable side effects, or "expert features", obviously has the peril that the attacker may do exactly the same. To protect such methods against evasion, there seems to be no other way than to hide the cricial mass of knowledge about features. Otherwise, as our study shows, there exist principled ways to automatically extend the missing knowledge for staging a successful evasion attack.

Are there alternative solutions that can make learning methods more robust to evasion? One potential solution is to use features that inherently represent, at least to a reasonable degree, the *causes* to be detected. One example of such features can be found in previous work on shellcode detection and classification, e.g., [37], [38], [39], [40], which uses n-grams, or short byte sequences, as basic features. Similar approach has been recently explored for detection of JavaScript malware, with the same techniques applied to sequences of syntactic tokens [41], [42]. The discriminative power of these methods lies in the inherent statistical difference between shellcode and usual packet content, as well as between malicious JavaScript code and benign programs. Hence one can expect such features to be less prone to malicious manipulation than "expert features". If fact, it has been shown that exact evasion of n-gram based features is NP-complete [20], and approximate solutions are widely believed to be difficult in practice.

Another potential solution can be offered by methods attempting to uniformly spread the "discriminative power" across as many features as possible. Some methods of this kind have been recently proposed for learning on problems with potential feature deletion or corruption [43], [44]. At the cost of a significant increase in the complexity of training problems, such methods offer a reasonable protection for limited amount of feature noise, regardless of the type of features. Assuming that the attacker has modification access to a limited number of features, as it was the case in our study, one can expect such methods to deliver a good tradeoff for the cases when no "intrinsic features" can be devised.

Finally, methods based on multiple classifier systems [45] should be mentioned as a potential solution. Evading a number of complementary classifiers can be significantly harder than a single classifier. For example, we were unable to design an optimal attack against a Random Forest classifier using a set of heterogeneous decision functions. Some applications of multiple classifier systems in security and other adversarial scenarios have recently been considered [46], [47].

IX. Conclusions and Future Work

In this paper, we have presented the first empirical security evaluation of a deployed learning-based system. Our study assumed that an attacker has no specific insider information about the system. It demonstrated, however, that enough information can be gathered from various sources and extended with apprximations and automatic inference algorithms in order to stage a successful evasion attack. In our experiments carried out on an established system for detection of PDF malware, PDFRATE, the significant drop in classification scores (from almost 100% to 28-33%) as well as deterioration of detection rates has been observed. We have also observed that simple countermeasures against evasion attacks, such as including a small fraction of attacks in the training data, are only effective if the anticipated attack exactly matches the performed one.

The findings of our study suggest that careful attention should be paid to the design of features and algorithms used in data-driven security techniques. Our future work will attempt to tackle the limitations of learning methods discovered in the presented experiments, especially in understanding of the general properties for construction of attack-resilient features. Our evaluation methods are applicable to other learning-based systems with modifiable features and we intend to extend our methods for security assessment of related systems for detection of malicious JavaScript and PDF files. By publishing our experimental code online we hope to share our experience with other researchers and to facilitate reproducibility of experimental results in security assessment of real systems.

Appendix A
PDFRate Feature Reimplementation

The MIMICUS experimental framework supports reading of 135 PDFRATE features (66%) described in [24]. The remaining 67 of 202 features were not disclosed. Modification of values of the following 68 features (33%) is supported:

- Features whose value can only be incremented (33):

```
count_acroform          count_image_xlarge
count_acroform_obs      count_image_xsmall
count_action            count_javascript
count_action_obs        count_javascript_obs
count_box_a4            count_js
count_box_legal         count_js_obs
count_box_letter        count_obj
count_box_other         count_objstm
count_box_overlap       count_objstm_obs
count_endobj            count_page
count_endstream         count_page_obs
count_eof               count_startxref
count_font              count_stream
count_font_obs          count_trailer
count_image_large       count_xref
count_image_med         size
count_image_small
```

- Features whose value can be both incremented and decremented (35):

```
author_dot       keywords_dot    subject_dot
author_lc        keywords_lc     subject_lc
author_num       keywords_num    subject_num
author_oth       keywords_oth    subject_oth
author_uc        keywords_uc     subject_uc
createdate_ts    moddate_ts      title_dot
createdate_tz    moddate_tz      title_lc
creator_dot      producer_dot    title_num
creator_lc       producer_lc     title_oth
creator_num      producer_num    title_uc
creator_oth      producer_oth    version
creator_uc       producer_uc
```

References

[1] U. Bayer, P. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, behavior-based malware clustering," in *Network and Distributed System Security Symposium (NDSS)*, 2009.

[2] O. Thonnard, "A multicriteria clustering approach to support attack attribution in cyberspace." Ph.D. dissertation, Ecole Doctorale d'Informatique, Télécommunications et Electronique de Paris, 2010.

[3] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious JavaScript code," in *International Conference on World Wide Web (WWW)*, 2010, pp. 281–290.

[4] M. A. Rajab, L. Ballard, N. Lutz, P. Mavrommatis, and N. Provos, "CAMP: Content-agnostic malware protection," in *Network and Distributed System Security Symposium (NDSS)*, 2013.

[5] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: a fast filter for the large-scale detection of malicious web pages," in *International Conference on World Wide Web (WWW)*, 2011, pp. 197–206.

[6] G. Stringhini, C. Kruegel, and G. Vigna, "Shady paths: leveraging surfing crowds to detect malicious web pages," in *ACM Conference on Computer and Communications Security (CCS)*, 2013, pp. 133–144.

[7] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, "COMPA: Detecting compromised accounts on social networks." in *Network and Distributed System Security Symposium (NDSS)*, 2013.

[8] D. M. Freeman, "Using naive bayes to detect spammy names in social networks," in *ACM Workshop on AI and Security (AISec)*, 2013, pp. 3–12.

[9] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: mining for unwanted p2p traffic," in *Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2013, pp. 62–82.

[10] M. Barreno, B. Nelson, A. Joseph, and J. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010.

[11] M. Kearns and M. Li, "Learning in the presence of malicious errors," *SIAM Journal on Computing*, vol. 22, no. 4, pp. 807–837, 1993.

[12] M. Brückner, C. Kanzow, and T. Scheffer, "Static prediction games for adversarial learning problems," *Journal of Machine Learning Research*, pp. 2617–2654, 2012.

[13] M. Kloft and P. Laskov, "Security analysis of online centroid anomaly detection," *Journal of Machine Learning Research*, vol. 13, pp. 3133–3176, 2012.

[14] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE Transactions on Knowledge and Data Engineering*, vol. 99, no. PrePrints, p. 1, 2013.

[15] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee, "Polymorphic blending attacks," in *USENIX Security Symposium*, 2006, pp. 241–256.

[16] R. Perdisci, D. Dagon, W. Lee, P. Fogla, and M. Sharif, "Misleading worm signature generators using deliberate noise injection," in *IEEE Symposium on Security and Privacy*, 2006, pp. 17–31.

[17] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *International Conference on Machine Learning*, 2012.

[18] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *International Cryptology Conference on Advances in Cryptology (CRYPTO)*, 2000, pp. 36–54.

[19] C. Dwork, "Differential privacy: a survey of results," in *International conference on theory and applications of models of computation (TAMC)*, 2008, pp. 1–19.

[20] P. Fogla and W. Lee, "Evading network anomaly detection systems: formal reasoning and practical techniques," in *ACM Conference on Computer and Communications Security*, 2006, pp. 59–68.

[21] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005, pp. 641–647.

[22] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2013. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40994-3_25

[23] C. Smutz and A. Stavrou, "Malicious PDF detection using metadata and structural features," in *Annual Computer Security Applications Conference (ACSAC)*, 2012, pp. 239–248.

[24] ——, "Malicious PDF detection using metadata and structural features," Available at http://cs.gmu.edu, Department of Computer Science, George Mason University, 4400 University Drive MSN 4A5, Fairfax, VA 22030-4444 USA, Tech. Rep. GMU-CS-TR-2012-5, 2012.

[25] D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou, "Detecting adversarial advertisements in the wild," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2011, pp. 274–282.

[26] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," in *USENIX Workshop on Large-scale Exploits and Emergent Threats (LEET)*, 2008, pp. 1–9.

[27] C. Yang, R. C. Harkreader, and G. Gu, "Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers," in *Recent Adances in Intrusion Detection (RAID)*, 2011, pp. 318–337.

[28] M. Brennan, S. Afroz, and R. Greenstadt, "Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity," *ACM Transactions on Information Systems Security*, vol. 15, no. 3, pp. 1–22, 2012.

[29] G. Oberreuter, G. L'Huillier, S. A. Ríos, and J. D. Velásquez, "Outlier-based approaches for intrinsic and external plagiarism detection," in *Knowledge-based and intelligent information and engineering systems*, 2011, pp. 11–20.

[30] "Document management - Portable document format - Part 1: PDF 1.7," http://www.adobe.com/devnet/pdf/pdf_reference.html, 2008.

[31] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[32] S. Jana and V. Shmatikov, "Abusing file processing in malware detectors for fun and profit," in *IEEE Symposium on Security and Privacy*, 2012, pp. 80–94.

[33] D. Maiorca, I. Corona, and G. Giacinto, "Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious pdf files detection," in *SIGSAC Symposium on Information, Computer and Communications Security*, 2013, pp. 119–130.

[34] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[35] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[37] K. Wang, J. Parekh, and S. Stolfo, "Anagram: A content anomaly detector resistant to mimicry attack," in *Recent Adances in Intrusion Detection (RAID)*, 2006, pp. 226–248.

[38] K. Rieck and P. Laskov, "Language models for detection of unknown attacks in network traffic," *Journal in Computer Virology*, vol. 2, pp. 243–256, 2007.

[39] J. Kolter and M. Maloof, "Learning to detect and classify malicious executables in the wild," *Journal of Machine Learning Research*, 2006, to appear.

[40] Z. Shafiq, S. Khayam, and M. Farooq, "Embedded malware detection using markov n-grams," in *Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2008, pp. 88–107.

[41] K. Rieck, T. Krüger, and A. Dewald, "Cujo: Efficient detection and prevention of drive-by-download attacks," in *Annual Computer Security Applications Conference (ACSAC)*, 2010, pp. 31–39.

[42] P. Laskov and N. Šrndić, "Static detection of malicious JavaScript-bearing PDF documents," in *Annual Computer Security Applications Conference (ACSAC)*, 2011, pp. 373–382.

[43] A. Globerson and S. Roweis, "Nightmare at test time: Robust learning by feature deletion," in *International Conference on Machine Learning (ICML)*, 2006, pp. 353–360.

[44] O. Dekel, O. Shamir, and L. Xiao, "Learning to classify with missing and corrupted features," *Machine Learning*, vol. 81, no. 2, pp. 149–178, 2010.

[45] F. Roli, G. Giacinto, and G. Vernazza, "Methods for designing multiple classifier systems," in *Multiple Classifier Systems*, 2001, pp. 78–87.

[46] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "McPAD: A multiple classifier system for accurate payload-based anomaly detection," *Computer Networks*, vol. 53, no. 6, pp. 864–881, 2009.

[47] B. Biggio, G. Fumera, and F. Roli, "Multiple classifier systems for adversarial classification tasks." in *Multiple Classifier Systems*, 2009, pp. 132–141.