Hardening Classifiers against Evasion: the Good, the Bad, and the Ugly

Liang Tong*, Bo Li†, Chen Hajaj*, Chaowei Xiao‡ and Yevgeniy Vorobeychik*
*Vanderbilt University
†University of California, Berkeley
‡University of Michigan, Ann Arbor
*{liang.tong, chen.hajaj, yevgeniy.vorobeychik}@vanderbilt.edu, †crystalboli@berkeley.edu, ‡xiaocw@umich.edu

arXiv:1708.08327v2 [cs.CR] 4 Nov 2017

Abstract*—Machine learning is widely used in security applications, particularly in the form of statistical classification aimed at distinguishing benign from malicious entities. Recent research has shown that such classifiers are often vulnerable to *evasion* attacks, whereby adversaries change behavior to be categorized as benign while preserving malicious functionality. Research into evasion attacks has followed two paradigms: attacks in *problem space*, where the actual malicious instance, such as the PDF file, is modified, and attacks in *feature space*, where the evasion attack is abstracted into directly modifying numerical features corresponding to malicious instances, rather than instances themselves. The feature space abstraction facilitates elegant mathematical modeling and analysis of evasion attacks, and has been the prevalent framework for designing evasion-robust classifiers. However, there exists no prior validation of the effectiveness of feature space threat models in representing real evasion attacks. We make several contributions to address this gap, using PDF malware detection as a case study, with four PDF malware detectors. First, we use iterative retraining to create a baseline for evasion-robust PDF malware detection by using an automated problem space attack generator in the retraining loop. Second, we use this baseline to demonstrate that replacing problem space attacks with feature space attacks may significantly reduce the robustness of the resulting classifier. Third, we demonstrate the existence of *conserved* (or invariant) features, show how these can be leveraged to design evasion-robust classifiers that are nearly as effective as those relying on the problem space attack, and present an approach for automatically identifying conserved features of PDF malware detectors. Finally, we evaluate generalizability of evasion defense through retraining by considering two additional evasion attacks. We show, surprisingly, that feature space retraining with conserved features can be dramatically more robust to the new attacks than classifiers retrained with the problem space model. This suggesting that when we properly account for conserved features, hardening classifiers with abstract feature space models of evasion can yield more generalizable evasion robustness than using specific problem space evasion attacks.

I. Introduction

Machine learning techniques are increasingly deployed in security applications such as network intrusion detection, fraud detection, and malware detection. Most of the traditional malware detection approaches are based on dynamic analysis, which incurs significant computational overhead and latency since such procedures depend on execution of shellcodes [32]. Machine learning—especially statistical classification—offers an effective alternative. The general approach is to extract a set of *features*, or numerical attributes, of entities in question, collect a training data set of labeled examples of malicious and benign instances, and learn a model which categorizes a previously unseen instance, presented in terms of its extracted features, as either malicious or benign. State-of-the-art approaches of this kind enable static detection of malicious entities, such as malware, efficiently and with accuracy often exceeding 99% [32], [31].

Recent research has shown that machine learning approaches, and especially classifier learning, are vulnerable to *evasion attacks* [7], [20], [32], [31], [37], [26]. A fundamental reason for such vulnerabilities is that classification learning algorithms generally assume that the distribution of training and test (or production) data is similar. This assumption is violated in security applications, where malicious entities correspond to attackers who take deliberate action to evade defensive measures. In the context of machine learning, evasion takes the form of modifying the structure of the malicious instances, such as malicious code, to make these look benign to the classifier. The net effect of the evasion attack is a modification of the feature vector corresponding to a malicious instance. However, a central challenge on the road to designing evasion-robust classifiers is an effective model of such malicious transformations in response to a particular classifier.

Two paradigms have emerged for modeling classifier evasion attacks. The first involves attacks in *problem space*, which construct evasions through modifying actual malicious instances [8], [31], [37], [21]. While not guaranteed to exactly emulate behavior of real attackers, these faithfully replicate constraints faced by real attackers in designing evasions, such as the inability to effect arbitrary changes in the feature space, and the requirement that malicious functionality of the instance is preserved (typically evaluated using a sandbox, such as WEPAWET [6] and the Cuckoo sandbox [12]). The second paradigm models attacks as modifications directly in *feature space*, imposing a modification cost typically captured as (weighted) norm difference from the original malicious instance [7], [20], [23], [24], [2], [16]. Feature space models of evasion are abstractions of real attacks, but they greatly facilitate theoretical analysis, such as connections between robustness and regularization [36], as well as principled algorithmic techniques and mathematical optimization methods for both constructing attacks [20], [36], [2] and defending against them [7], [5], [18]. Indeed, essentially all methods proposed for designing evasion-robust classifiers rely on feature space attack models [7], [4], [18], [39], [5], [19].[1]

Two questions therefore naturally arise: (1) how can we develop techniques for evasion-robust classification which rely on problem space evasion models, and (2) how well do feature space evasion models represent actual attacks (if we take problem space attacks to be a reasonable representation thereof)? It turns out that the answer to the first is, in principle, direct: we can leverage iterative classifier retraining schemes proposed in prior literature (e.g., [10], [19]). In such schemes, we iterate between training a classifier and adding adversarial evasions of the latest version of the classifier to data, until convergence (for example, when no new evasions can be found). These methods are agnostic to the specifics of the adversarial evasion model. Moreover, this approach provides us with a natural means to systematically evaluate effectiveness of feature space evasion models, as compared to a strong problem space attack method.

Our first major contribution is a comprehensive evaluation of problem space and feature space evasion attacks in the context of four PDF malware detectors: two that use structure-based features (SL2013 and Hidost), and two that extract features based on PDF content (two variations of PDFRate). Our evaluation of evasion robustness is based on EvadeML, a powerful automated problem space method for PDF malware classifier evasion [37] which uses genetic programming; we also use EvadeML to construct a baseline evasion-robust PDF classifier through adversarial retraining. We demonstrate that EvadeML successfully defeats not only SL2013 and PDFRate, but also the new incarnation of the former, Hidost, designed in part to address vulnerabilities of SL2013. Remarkably, we show that by simply *binarizing* the features of PDFRate, we make it robust against EvadeML *without any additional effort*. On the other hand, we show that after only 10 rounds of iterative retraining with EvadeML serving as the attack model, we can obtain a robust classifier (against EvadeML, on test data) even when we start with the vulnerable SL2013, Hidost, and PDFRate baselines. However, we demonstrate that retraining SL2013 and Hidost using conventional feature space attack models fails to endow the resulting classifier with adequate evasion robustness against EvadeML. In other words, *the widely used feature space evasion models can be inadequate in representing realistic evasion attacks*.

The limitations of feature space evasion models raise a natural follow-up question: is there is minimal modification of such models that preserves their mathematical elegance, but bridges the gap with problem space evasion attacks? Our second major contribution is an affirmative answer to this question. Specifically, we identify a set of *conserved* features which are (essentially) invariant in problem space attacks, likely representing functionally vital regions of the feature space; moreover, we present a novel systematic approach for effectively identifying such features for a given classifier and automated evasion attack method. Conserved features suggest a simple fix to the feature space evasion models: simply constrain that the adversary cannot change features which are conserved. We demonstrate that this fix is extremely effective: retraining with the modified feature space model largely bridges the gap with problem space attacks.

Our final major contribution is an evaluation of classifiers which have been shown to be robust to EvadeML using two alternative evasion attacks: *mimicry* [31], which was one of the first problem space attacks on PDF malware detectors, and MalGAN, an attack based on a generative adversarial network [14]. While all classifiers remain robust to mimicry, we show that those retrained with a problem space model are *vulnerable* to the MalGAN attack, in contrast with feature space retrained classifiers, which remain robust. This demonstrates another important advantage of feature space models modified to use conserved features: their generic nature endows them with greater transferability against alternative attacks than classifiers which are specialized to be robust to a particular problem space attack architecture.

[1]In some domains, such as vision, the distinction between feature space and problem space attacks can be blurred [10], [15]. However, this distinction is important in malware detection; see our discussion in Section III-D.

II. Related Work

A. Problem Space Methods for Classifier Evasion

One of the first problem space evasion attacks on machine learning was devised by Fogla et al. [8], who developed a polymorphic blending attack on anomaly-based intrusion detection systems. Fogla et al. subsequently generalize and systematize the polymorphic blending attack. Šrndic and Lasov [31] present a case study of an evasion attack on a state of the art PDF malware classifier, PDFRate. Their mimicry attack, which primarily adds content to a PDF to make it appear as benign as possible, actually leverages an initial feature-space evasion, which is subsequently modified to effect the actual PDF source. Xu et al. [37] propose EvadeML, a fully problem space attack on PDF malware classifiers which generates evasion instances by using genetic programming to modify PDF source directly, using a sandbox to ensure that malicious functionality is preserved. The fully automated nature of EvadeML makes it a natural candidate for our in-depth exploration of the relationship between problem space and feature space evasion attacks.

In addition to malware evasion attacks, a series of efforts explore evasion in the context of image classification by deep neural networks [10], [15], [26], [17]. These are in a sense both problem and feature space, as actual pixels of the images are modified in the so-called *adversarial examples* which are crafted to effect errors in deep classifiers. Several efforts explored the impact of indirect modification when an adversarial image must be printed, showing that effectiveness of such techniques can nevertheless be preserved [28], [17]. Recently, an approach for printing specifically designed glass frames had been shown to mislead vision-based biometric systems to either mistakenly grant authorization, or enable evasion of face recognition techniques [28].

B. Feature Space Methods for Classifier Evasion

In addition to classifier evasion methods which change the actual malicious instances, a number of techniques have

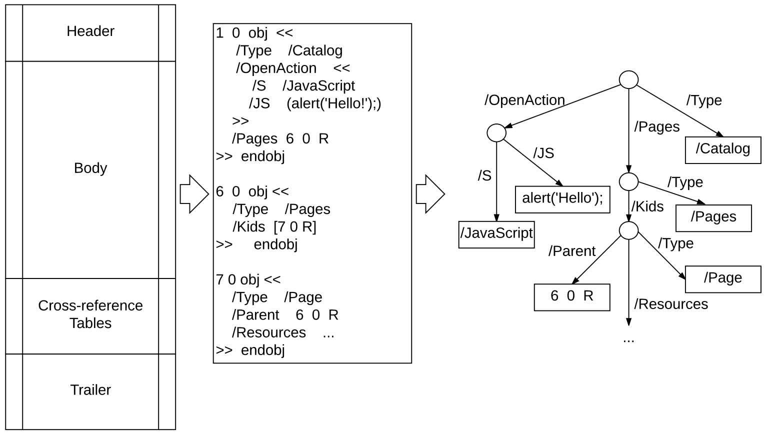

Fig. 1: Various representations of a PDF file: file structure (left), physical layout (middle), and logical structure (right)

sprouted for evasion models acting directly on features [7], [20], [1], [16]. This line of research has become prominent in part for the relative simplicity of modeling attacks in feature spaces, and additionally because it is the primary set of methods informing general-purpose methods for evasion-robust learning. Dalvi et al. [7] present one of the earliest such models as a part of a robust learning approach. An algorithmic investigation of the feature space evasion problem—modeling it formally as minimizing the cost of changing features subject to a constraint that the resulting instance is classified as benign—was initiated by Lowd and Meek [20]. A series of methods follow their framework, but consider more general classes of classifiers, introducing a constraint on the evasion cost, and explicitly trading off the degree to which an instance appears as benign and evasion cost [2], [3], [13], [38], [16], [18], [19], [25], [35].

C. Evasion-Robust Classification

Dalvi et al. [7] presented the first approach for evasion-robust classification, making use of a model in which the attacker aims to transform feature vectors into benign instances in response to the original (non-robust) classifier, and then subsequently devising an approach which is robust to the former attack. A series of approaches formulate robust classification as minimizing maximum loss, where maximization is attributed to the evading attacker aiming to maximize the learner's loss through feature space transformations [34], [39]. All of these effectively assume that the interaction between the learner and attacker is zero sum. A number of alternative methods for designing classifiers robust to evasion relax the somewhat unreasonable assumption that the adversary aims to maximize the defender's loss. Instead, these consider the interaction as a non-zero-sum game, either played simultaneously between the learner and the attacker, or a Stackelberg game, in which the learner is the leader, while the attacker the follower [4], [18], [16], [27]. In the deep learning literature involving adversarial manipulations of images, somewhat ad hoc procedures for retraining the classifier to boost its robustness have been proposed [10], [15], and these have been adapted to other classification models, such as decision tree classifiers [16]. Recently, a systematic iterative retraining procedure had been proposed which leverages general-purpose adversarial evasion models, and offers a theoretical connection to an underlying Stackelberg game played between the learner and the evading adversary [19]. These diverse efforts share one common property: attack models that they leverage use feature space manipulations, which are only a proxy for problem space evasion attacks.

III. Background

This section provides background on PDF document structure, the target PDF malware classifiers, and evasion attacks used to evaluate classifier robustness.

A. PDF Document Structure

The Portable Document Format (PDF) is an open standard format used to present content and layout on different platforms. A PDF file structure consists of four parts: *header*, *body*, *cross-reference table* (CRT), and *trailer*, which are presented on the left-hand side of Figure 1. The header contains information such as the magic number and format version. The body is the most important element of a PDF file, which comprises multiple PDF objects that constitute the content of the file. These objects can be one of the eight basic types: Boolean, Numeric, String, Null, Name, Array, Dictionary, and Stream. They could be referred from other objects via indirect references. There are other types of objects such as JavaScript which contains executable JavaScript code. The CRT indexes objects in the body, while the trailer points to the CRT.

The syntax of the body of a PDF file is shown in the middle of Figure 1. In this example, the PDF body contains three indirect objects referenced by others. The first one is a `catalog` object, which contains two additional entries: `OpenAction` and `Pages`. These two entries are dictionaries. The `OpenAction` entry has two internal entries: `S` and `JS`, which are JavaScript codes to be executed. The `Pages` entry refers to the second object with the type `Pages`. The second object contains an entry "kids" which refers to the third object.

The third one is a `Page` object which refers back to the second object.

The relations between the three objects above could be described as a directed graph to present their logic structure by using edges representing reference relations, and nodes representing different objects as shown on the right-hand-side of Figure 1. As an object could be referred to by its child node, the resulting logic structure is a directed cyclic graph. For example in Figure 1, the second and third objects refer to each other and constitute a directed circle. To eliminate the redundant references, the logic structure could be reduced to a structural tree with the breadth-first search procedure.

B. The Target Classifiers

Several PDF malware classifiers have been proposed [6], [30], [32], [33]. For our study, we selected SL2013 [32], Hidost [33] and PDFRate [30]. SL2013 and its revised version, Hidost, are *structure-based* PDF classifiers, which use the logical structure of a PDF document to construct and extract features used in detecting malicious PDFs. PDFRate, on the other hand, is a *content-based* classifier, which constructs features based on *medadata* and *content* information in the PDF file to distinguish benign and malicious instances. Evasion attacks on both SL2013 and PDFRate classifiers, particularly in the problem space, have been developed in recent literature [32], [31], [37], [33], providing a natural evaluation framework for our purposes.

1) Structure-based classifiers: **SL2013.** SL2013 is a well-documented and open-source machine learning system using Support Vector Machines (SVM) with a radial basis function (RBF) kernel, and was shown to have state-of-the-art performance.

It employs structural properties of PDF files to discriminate between malicious and benign PDFs. Specifically, SL2013 uses the presence of particular *structural paths* as binary features to present PDF files in feature space. A structural path of an object is a sequence of edges in the reduced *logical structure*, starting from the catalog dictionary and ending at this object. Therefore, the structural path reveals the shortest reference path to an object.

SL2013 uses a uniform set of structural paths to classify a PDF file. To get these paths, it first obtains a total of 658,763 benign and malicious PDF files (around 595 GB), then selects only the structural paths that occur in at least 1,000 PDF files. This process reduced the number of features from over 9 million to 6,087. Trained using 5,000 malicious and 5,000 benign PDF files, SL2013 was shown to have 99.8% accuracy on test data and AUC $> 99.9\%$.

Hidost. Hidost is an updated version of SL2013. It inherits all the characteristics of SL2013 and employs *structual path consolidation* (SPC), a technique to consolidate similar features which have the same semantic meaning in a PDF. As the semantically equivalent structural paths are merged, Hidost reduces polymorphic paths and still preserve the semantics of logic structure, so as to improve evasion-robustness of SL2013.

Hidost used a dataset of 407,037 benign and 32,567 malicious PDF files collected over 14 weeks for training, and obtained 99.5% accuracy on test data, with AUC $> 99.5\%$ In our work, we employ the 961 features identified in the experiment for the 14th week, as these correspond to the latest version of Hidost.

2) Content-based classifiers: **PDFRate.** The original PDFRate classifier uses a random forest algorithm, and employs PDF *metadata* and *content* features to categorize benign and malicious PDF files. The metadata features include the size of a file, author name, and creation date, while content-based features properties include position and counts of specific keywords. All features were manually defined by the authors of [30].

PDFRate uses a total of 202 features, but only 135 of these are publicly documented [29]. Consequently, in our work we employ the Mimicus implementation of PDFRate which was shown to be a close approximation [31]. Mimicus trained a surrogate SVM classifier with the documented 135 features and the same dataset with PDFRate, using both the SVM and random forrest classifiers, both performing comparably. We use the SVM implementation in our experiments to enable more direct comparisons with the structure-based classifiers which also use SVM. An important aspect of Mimicus is *feature standardization* on extracted data points performed by subtracting the mean of the feature value and dividing by standard deviation, transforming all features to be real-valued and zero-mean. This surrogate was shown to have $\sim 99\%$ accuracy on the test data.

C. Automated Evasion

1) EvadeML: To evaluate the robustness of a PDF classifier against adversarial evasion attacks, we adopt EvadeML [37], an automated method to craft evasion instances of PDF malware in problem space. EvadeML starts with a malicious PDF which is correctly classified as malicious and aims to produce evasive variants which have the same malicious behavior but are classified as benign. It assumes that no internal information of the target classifier is available to the adversary, such as the set of features, the training dataset, and the classification algorithm. Rather, the adversary has black-box access to the target classifier, and it can repeatedly submit PDF files to get corresponding classification scores. Based on the scores, the adversary can adapt its strategy to craft evasive variants.

EvadeML employs genetic programming (GP) to search the space of possible PDF instances to find ones that evade the classifier while maintaining malicious features. First, an initial population is produced by randomly manipulating a malicious seed. As the seed contains multiple PDF objects, each object is set to be a target and mutated with exogenously specified probability. The mutation is either a deletion, an insertion or a swap operation. A deletion operation deletes a target object from the seed malicious PDF file. As a result, the corresponding structural path is deleted. An insertion operation inserts an object from external benign PDF files (also provided exogenously) after the target object. EvadeML uses 3 most benignly scoring PDF files. A swap operation replaces the entry of the target object with that of another object in the external PDFs.

After the population is initialized, each variant is assessed by the Cuckoo sandbox [12] and the target classifier to evaluate its fitness. The sandbox is used to determine if a variant preserves malicious behavior. It opens and reads the variant PDF in a virtual machine and detects malicious behaviors such as API or network anomalies, by detecting malware signatures. The target classifier (SL2013 in our case) provides a classification score for each variant. If the score is above a threshold, then the variant is classified as malicious. Otherwise, it is classified as a benign PDF. If a variant is classified as benign but displays malicious behavior, or if GP reaches the maximum number of generations, then GP terminates with the variant achieving the best fitness score and the corresponding mutation trace is stored in a pool for future population initialization. Otherwise, a subset of the population is selected for the next generation based on their fitness evaluation. Afterward, the variants selected are randomly manipulated to generate the next generation of the population.

EvadeML was used to evade SL2013 in [37]. The reported results show that it can automatically find evasive variants for all 500 selected malicious test seeds. However, we found a small error in the implementation which caused the reported evasion results to be slightly inflated; we were able to reproduce an approximately 84% evasion rate, as reported below. Throughout, we use 400 malicious seeds as a part of evasion-based training and evaluate on the remaining 100 malicious seed PDFs used by EvadeML.

2) Mimicry: Mimicry assumes that an attacker has full knowledge of the features employed by a target classifier. The mimicry attack then manipulates a malicious PDF file so that it mimics a particular selected benign PDF as much as possible. The implementation of Mimicry is simple and independent of any particular classification model.

Our mimicry attack uses the Mimicus [31] implementation, which was shown to successfully evade the PDFRate classifier. To improve evasion effectiveness, Mimicus chooses 30 different target benign PDF files for each attack file. It then produces one instance in feature space for each target-attack pair by merging the malicious features with the benign ones. The feature space instance is then transformed into a PDF file using a *content injection approach*. The resulting 30 files are evaluated by the target classifier, and only the PDF with the best evasion result is selected, which was submitted to WEPAWET [6] to verify malicious functionality. To make Mimicry consistent with our framework, we employ the Cuckoo sandbox [12] in place of WEPAWET (which was in any case discontinued) to validate maliciousness of the resulting PDF file.

3) MalGAN: MalGAN [14] is a Generative Adversarial Network (GAN) [9] framework to generate malware examples which can evade a black-box malware detector with binary features. It assumes that an attacker has full knowledge of the feature set of the malware detector, but only black-box access to the detector decisions, and it can repeatedly query the classification results of submitted PDF files.

MalGAN comprises three main components: a generator which transforms malware to its adversarial version, a black-box detector which returns detection results, and a substitute detector which has no knowledge of the black-box detector but is used to fit the black-box detector and train the generator. The generator and substitute detector are feed-forward neural networks which work together to evade the black-box attacker.

Before training MalGAN, an attacker collects a dataset with malware and benign instances. Initially, the generator takes malware m and random noise z as inputs. The random noise is applied to produce diverse perturbations on m, resulting in a set of adversarial examples m'. m' is further fed into the black-box detector together with benign examples b. The classification labels obtained provide corresponding gradient information and are used to train the substitute detector and generator. MalGAN is trained with multiple epochs and used to transform the test data into evasive instances. The results of [14] show that MalGAN is able to decrease the *True Positive Rate* on the generated examples from $> 90\%$ to 0%.

D. Problem Space vs. Feature Space Attacks

A distinction that is crucial in this paper is between *problem space* and *feature space* attacks on malware detectors. Feature space attacks assume that features in the feature vector representation of a malicious entity can be modified directly and essentially arbitrarily (modulo domain definition constraints; e.g., binary features can only be flipped). Problem space attacks, in contrast, have two important aspects:

1) they are performed *directly on the malicious entity*, such as the malware source code before said entity is translated into feature space, and
2) after the malicious entity is modified, the attack is verified to still be effective, for example, using a sandbox such as the Cuckoo sandbox in the EvadeML framework above.

To appreciate the importance of the first aspect of problem space attacks, as distinct from those modeled directly through feature modifications, consider a simple example with two features: one which counts the number of Javascript objects, and the second which is just the size of the PDF file. Suppose that fewer Javascript objects, and larger files, are both indicative of benign PDFs according to a classifier. Clearly, however, removing Javascript objects will reduce the size of the PDF—that is, the two features are not independent, and cannot be modified independently in arbitrary ways.

The importance of the second aspect of problem space attacks may seem self-evidence, but it imposes strong, and very complex constraints on the nature of modifications to the malicious entity, such as a PDF file, that can be made, and these are very challenging to reflect through direct feature space modifications. One approach to side-step this issue is to restrict attacks to only modify features which are unlikely to impact malicious functionality, as was done by Grosse et al. [11]. However, strong restrictions on adversarial behavior are inadequate for considering *defense* against adversarial evasion attacks; moreover, one can rarely *guarantee* that seemingly innocuous modifications will not impact malicious functionality, and verification through a sandbox is still a necessary step.[2]

IV. Experimental Methodology

Our main goal is to evaluate the efficacy of classifier evasion models. In particular, we aim to compare the elegant and commonly used feature-space models, which allow an attacker to modify features directly, with attacks that actually modify PDF files and are validated to have preserved malicious functionality.

A. Problem-Space Evasion Model

We use EvadeML as the primary problem space evasion model for the first part of the paper. We set the GP parameters in EvadeML as the same as in the experiments by Xu et al. [37]. The population size in each generation is 48. The maximum number of generations is 20. The mutation rate for each PDF object is 0.1. The fitness threshold of a classifier is 0. We use the same external benign PDF files as Xu et al. [37], for both retraining and robustness evaluation. Subsequently, we evaluate the effectiveness of ostensibly robust classifiers (as evaluated by EvadeML) against two additional evasion attacks, mimicry and MalGAN, described above.

B. Feature-Space Evasion Model

There have been a number of evasion models in feature space proposed in prior literature. All involve casting evasion as an optimization problem essentially trading off two considerations: ensuring that the adversarially modified feature vector is classified as benign, and minimizing the total cost of feature modification, where the latter is commonly measured using an l_p norm difference between the original malicious instance and the modified feature vector [19], [2]. In typical problem space attacks, including EvadeML, a consideration is not merely to move to the benign side of the classifier decision boundary, but to appear as benign as possible. This naturally translates into the following multi-objective optimization in feature space:

$$\underset{x}{\text{minimize}} \quad Q(x) = f(x) + \lambda c(x_M, x), \tag{1}$$

where $f(x)$ is the score of a feature vector x, with the actual classifier (such as SVM) $g(x) = \text{sgn}(f(x))$, x_M is the malicious seed, x an evasion instance, $c(x_M, x)$ the cost of transforming x_M into x, and λ a parameter which determines the relative importance of appearing more benign and feature transformation cost. We use the standard weighted l_p norm distance between x_M and x as the cost function, which is equivalent to weighted l_1 norm: $c(x_M, x) = \sum_i \alpha_i |x_i - x_{M,i}|$, since the features are binary. Below we consider two variations of this cost function: first, using uniform weights, with $\alpha_i = 1$ for all features i, and second, using non-uniform weights (which we term *weighted distance (WD)* below).

As the optimization problem in Equation (1) is non-convex, we use a common local search method, *Coordinate Greedy*, to compute a local optimum. In this method, we optimize one randomly chosen coordinate of the feature vector at a time, until a local optimum is reached. To improve the quality of the resulting solution, we repeat this process from several random starting points.

C. Datasets

The dataset involved in our experiment is from the *Contagio Archive.* [3] We use 5,586 malicious and 4,476 benign PDF files to train SL2013, and another 5,276 malicious and 4,459 benign files as the non-adversarial test dataset. The training and test datasets also contain 500 seeds selected by [37], with 400 in the training data and 100 in the test dataset. These seeds are filtered from 10,980 PDF malware samples and are suitable for evaluation since they are detected with reliable malware signatures by the Cuckoo sandbox [12]. We randomly select 40 seeds from the training data as the retraining seeds and use the 100 seeds in the test data as the test seeds.

D. Iterative Retraining

In order to make a fair and systematic comparison between problem space and feature space evasion models we make use of a variation of an iterative classifier retraining procedure [19], [16]. Variations of this procedure have been used for this purpose in the past, with a systematic version and provable guarantees described by Li et al. [19].

We make use of the latter principled variant of iterative retraining, outlined as follows:

1) Starts with the initial classifier
2) Execute the *evasion model* for each malicious instance in training data to generate a new feature vector
3) Add all new data points to training data (removing any duplicates), and retrain the classifier
4) Terminate after either a fixed number of iterations, or when no new evasions can be added.

We made two modifications to the above approach. First, we used only a set of 40 malicious instances which were seeds to EvadeML to generate evasions, to remain consistent with the prior use of EvadeML, reduce running time, and make the experiment more consistent with realistic settings where a large proportion of malicious data is not directly adapting to the classifier. Nevertheless, as shown below, this set of 40 instances was sufficient to generate a model robust to evasions from held out 100 malicious seed PDFs. The second, and more fundamental, modification is to use EvadeML as the oracle for generating actual malicious PDF instances, from which features are subsequently extracted and added to the training

[2]In some domains, such as vision tasks, the distinction between problem and feature space attacks is blurred. However, it is important in malware detection, where validation of malicious functionality is critical to the attack.

[3]The malicious PDF files are available at http://contagiodump.blogspot.com/2010/08/malicious-documents-archive-for.html, and the benign files are available at http://contagiodump.blogspot.com/2013/03/16800-clean-and-11960-malicious-files.html.

data, rather than a feature space evasion model as in prior work.

We distribute both retraining and adversarial test tasks in two servers (Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz, 18 cores, 71 processors and 64 GB memory running Ubuntu 16.04). For retraining, we assign each server 20 seeds; each seed is processed by EvadeML to produce the adversarial evasion instances. We then add the 40 examples obtained to the training data, retrain the classifier, and then split the seeds into the two servers in next iteration. In the evaluation phase, we assign each server 50 seeds from the 100 test instances, and each seed is further used to evade the classifier by using EvadeML.

E. Evaluation Metrics

Throughout, we evaluate performance in two ways: 1) evaluation of evasion robustness, and 2) traditional evaluation. To evaluate robustness, we compute the proportion of 100 malicious test seed PDFs for which EvadeML successfully evades the classifier; we call this metric *evasion robustness*. Thus, evasion robustness of 0% means that the classifier is successfully evaded in every instance, while evasion robustness of 100% means that evasion fails every time. Our traditional evaluation metric uses test data of malicious and benign PDFs, where no evasions are attempted. On this data, we compute the ROC (receiver operating characteristic) curve and the corresponding AUC (area under the curve).

V. Case Study with a Structure-Based PDF Malware Classifier

Our first case study uses a state-of-the-art PDF malware classifier which engineers features based on PDF *structure*. Indeed, we evaluate two versions of this classifier: an earlier version, which we call *SL2013*, and a more recent version reengineered specifically to be more robust to a class of evasion attacks (specifically, *mimicry* attacks), which we call *Hidost*. The experiments by Xu et al. [37] demonstrate that although SL2013 was designed to be resistant to evasion attacks, it can be successfully evaded. Since Hidost was a recent significant redesign attempting to address its limitations by significantly reducing the feature space, no data exists on its vulnerability to evasion attacks. Below we demonstrate that despite a deliberate effort to harden Hidost, it remains quite vulnerable to evasion attacks (indeed, more so than SL2013). Significantly, we demonstrate that retraining with a problem space evasion attack (PSR henceforth) based on EvadeML significantly hardens both classifiers against evasion. On the other hand, we show that similar hardening through retraining by using a *feature space* evasion model (FSR henceforth) *fails to adequately improve classifier robustness*.

A. Experiments

1) SL2013: In our experiments, we empirically set the *RBF* parameters for training SL2013 as $C = 12$ and $\gamma = 0.0025$.

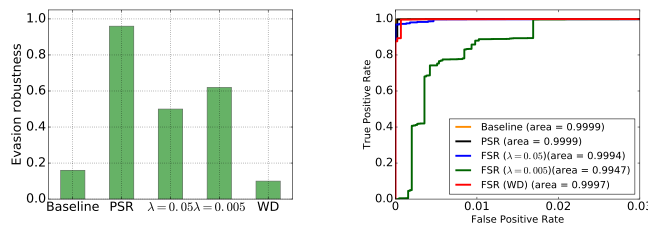

Fig. 2: Evasion robustness (left) and performance on non-adversarial data (right) of problem space retraining (PSR) on SL2013.

a) Problem Space Retraining: We first conduct experiments in which EvadeML is employed to retrain SL2013. Before the target classifier SL2013 was retrained, its robustness was first evaluated by EvadeML. For all the 100 adversarial examples produced by EvadeML, SL2013 could only achieve a 16% evasion robustness. This result provides a baseline with which we compare the robustness after retraining SL2013. We conducted an EvadeML test to evaluate the robustness of SL2013 after each iteration of retraining.

The experiment of retraining SL2013 with EvadeML took approximately six days to execute. The process terminated after 10 iterations as no evasive variants of the 40 retraining seeds could be generated. We observe (Figure 2 (left)) that *the retrained classifier (PSR) obtained by this approach achieves a 96% evasion robustness (that is, EvadeML terminates without successfully finding an evasion in 96% of the instances)*. This provides us with a baseline to evaluate the effectiveness of the feature space evasion model in achieving evasion robustness.

b) Feature Space Retraining: We next conduct experiments to evaluate the effectiveness of the retraining approach to boost evasion robustness which uses a feature space evasion model. As above, we use EvadeML to *evaluate* how evasion robust the resulting classifier is after retraining. We consider the setting with uniform weights, with $\lambda = 0.05$ and $\lambda = 0.005$ in Equation 1, and a natural attack model where feature weights are non-uniform, setting $\lambda = 1$ in this case without loss of generality (since α_is are simply rescaled).

While it is difficult to find a principled means of assigning heterogeneous feature weights based on training data alone, we distinguish high- and low-weight features depending on how frequently the feature appears among malicious and benign instances. If a feature is common in malicious, but rare in benign instances, we assume that changing this feature is associated with a high cost due to the possible modification of the malicious functionality; otherwise, we assume that changing this feature is associated with a low cost. Formally, let n_m and n_b be the counts of malicious and benign instances, respectively, while n_m^i and n_b^i are the counts of appears of feature i in malicious and benign instances. If $\frac{n_m^i}{n_m} \geq \frac{n_b^i}{n_b}$, we set $\alpha_i = 0.5$, and set it to 0.05 otherwise.

The results are summarized in Figure 2 (left). Compared to the SL2013 baseline, retraining with feature space evasion

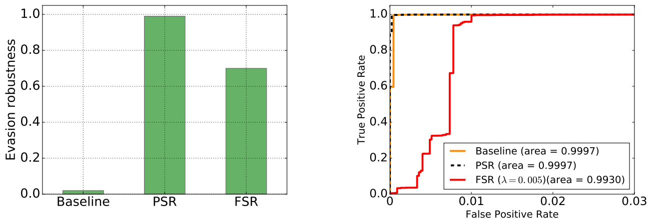

Fig. 3: Evasion robustness (left) and performance on non-adversarial data (right) of the baseline and different retraining approaches for Hidost under EvadeML test.

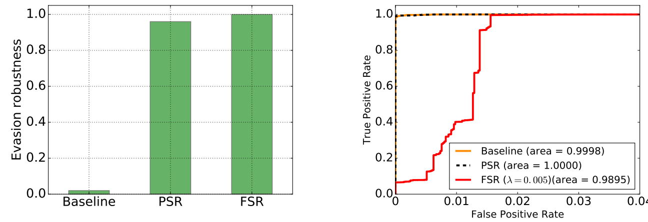

Fig. 4: Evasion robustness (left) and performance on non-adversarial data (right) of the baseline and different retraining approaches for PDFRate-R under EvadeML test.

models (FSR) boosts evasion robustness from 16% to 60%, but only achieves a 10% evasion robustness when weighted distance (denoted as WD in Figures 2 (left) and 2 (right)) is employed. Crucially, *the robustness of the resulting classifier is far below the classifier achieved by PSR.* In other words, *the feature space model is not an adequate representation of real evasion attacks.*

c) Performance on Non-Adversarial Data: Next, we evaluate the performance of the retrained classifiers on non-adversarial test data (i.e., traditional test data validation sans deliberate evasion attacks). The corresponding ROC curves of the classifiers discussed above are presented in Figure 2 (right). PSR achieves a comparable accuracy ($>$ 99.9% AUC) on non-adversarial data with the baseline (original SL2013) classifier. Thus, PSR maintains the performance of SL2013 on non-adversarial instances while significantly improving robustness against evasions in problem space. Evaluating the quality of FSR, we can see that robustness boosting does not much degrade its performance, with AUC remaining above 99%. However, we do see slight degradation for small values of the false positive rates, compared both to SL2013 and the PSR classifier. This is an issue of some concern, as this is the region of most practical import: real systems would necessarily operate at the low false positive rate level.

2) Hidost: We now evaluate the robustness of both the problem space and feature space retraining approach for Hidost, the updated version of SL2013. We use the same setup as in our study of SL2013. We set the retraining parameter $\lambda = 0.005$, and only consider the feature space attack model with uniform weights for each feature.

Evasion robustness of Hidost, as well as improvements achieved by PSR and FSR, are shown in Figure 3 (left). We can observe that although Hidost is designed to be more robust than SL2013, it is actually even more vulnerable to EvadeML evasion attacks, with only a 2% evasion robustness (compared to 15% for SL2013). By retraining with the problem space evasion model, evasion robustness is boosted to 99%, a rather dramatic improvement. In contrast, feature space retraining achieves a 70% evasion robustness, a significant boost over baseline, to be sure, but a far below PSR.

Evaluating these classifiers on non-adversarial test data in terms of ROC curves (Figure 3 (right)), we can observe that PSR achieves comparable accuracy ($>$ 99.9% AUC) with Hidost on non-adversarial data, and even better *True Positive Rate* when *False Positive Rate* is close to zero. On the other hand, FSR achieves $>$ 99.9% AUC, but yields a significant degradation of TPR when FPR$<$ 0.01.

VI. Case Study with a Content-Based PDF Malware Classifier

Our next case study of PSR and FSR concerns another PDF malware classifier which uses features based on PDF file content, rather than logical structure. Aside from the nature of features, another important difference between Hidost and PDFRate is feature types: PDFRate uses features of mixed type, including binary, integer, and real-valued. However, the surrogate representations of PDFRate previously used to evaluate its robustness had normalized the features to be real-valued and zero-mean [31], [37] (henceforth, PDFRate-R). In addition, we construct a *binarized* version of PDFRate (henceforth, PDFRate-B), where each feature is transformed into a binary feature (where needed) by thresholding. Thus, our evaluation enables a direct comparison between binary and real-valued versions of the same classifier, facilitating generalizable insights.

We trained both binarized and real-valued PDFRate on the same dataset as SL2013 and Hidost, and achieved $>$ 99.9% AUC for both classifiers on the test data. In our experiments, we empirically set the SVM *RBF* parameters for training to $C = 10$ and $\gamma = 0.01$.

A. Experiments

We now evaluate the robustness of both the problem space and feature space retraining of the two versions of PDFRate, with the same settings as in our investigation of Hidost. The feature space retraining parameter λ is set to be 0.005, and we only consider feature space attack model with uniform weights for each feature.

1) PDFRate with Real-Valued Features: We begin with the version of PDFRate—PDFRate-R—which has been constructed in previous evaluations and shown comparable in performance to the original implementation [31]. We begin by replicating the EvadeML evasion robustness evaluation of the baseline classifier. As expected, we find the classifier quite vulnerable, with only a 2% evasion robustness.

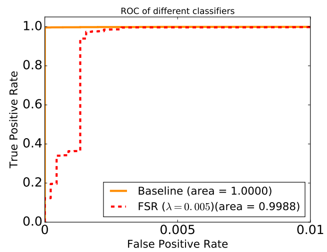

Fig. 5: ROC curves on non-adversarial test data of the baseline and FSR for PDFRate-B

Next, we retrain PDFRate-R with EvadeML for 10 iterations (PSR baseline), and perform feature space retraining using the conventional feature space model above. Our results are quite surprising. As shown in Figure 4 (left), both PSR and FSR achieve high evasion robustness; indeed, FSR is slightly better at 100% robustness, with PSR at 96%.

Comparing PSR and FSR performance on non-adversarial data (Figure 4 (right)), we begin to observe some meaningful degradation for FSR: while PSR remains exceptionally effective (>99.99% AUC), FSR achieves AUC slightly lower than 99%, although most significantly, the degradation is particularly pronounced for low FPR regions (below 0.15). Thus, there remains a gap between PSR and FSR, but this gap now has different nature: both are quite robust to problem space evasion, but FSR degrades baseline performance significantly more than PSR.

2) PDFRate with Binarized Features: One of our greatest surprises is the robustness of the *baseline* binarized PDFRate: despite the fact that the real-valued PDFRate is quite vulnerable, *the same classifier using binary features was 100% robust to EvadeML evasions*. Consequently, the PSR effectively terminates with no retraining at all. Feature space retrained PDFRate-B also exhibits 100% evasion robustness, although it does require a number of iterations to converge.

Considering now the performance of PDFRate-B and FSR on non-adversarial test data (Figure 5), we can make two interesting observations. First, the baseline PDFRate-B is remarkably good even without evasions: in a sense, it appears to hit the sweet spot of adversarial robustness and non-adversarial performance. Second, FSR retrained classifier is competitive in terms of AUC ($\sim$ 99.9%), but is observably worse than the baseline classifier for very low false positive rates.

VII. Evasion-Robust Classification with Conserved Features

So far, we observed that the widely used feature space evasion models may only poorly represent realistic evasion attacks. Next we propose a simple idea for bridging the gap between problem and feature space (where one exists): explicitly accounting for a subset of features which are *conserved* during evasions. More precisely, conserved features are those which are invariant under (are unaffected by) evasion attacks. We develop this idea for binary features; when features are real-valued, it is not clear how we can even defined conserved features, and in any case we observed no appreciable evasion robustness gap between PSR and FSR in that setting.

Next, we present three surprising findings. First, conserved features do exist (for EvadeML), and can be effectively identified. Second, we show that a classifier using *only* the conserved features is (a) completely robust to the EvadeML attack (essentially by construction), and (b) quite effective on test data not involving evasion attacks. We also observe that conserved features cannot be recovered using standard feature reduction, and feature reduction methods do not lead to robust classifiers. The reason is that conservation is connected to evasion attacks, rather than statistical properties of non-evasion data; for example, features which are strongly correlated with malicious behavior are often a consequence of attacker "laziness" (for example, whether a PDF file has an author), and are easy for attackers to modify. Third, we demonstrate that the limitations of feature-space robust classification approaches can be essentially eliminated by incorporating conserved features as attack invariants in the evasion model. Subsequently (in Section VIII) we present a novel automated approach for identifying conserved features in a binary feature space.

A. Conserved Features

In our case study of SL2013, the target classifier employs structural paths as features to discriminate between malicious and benign PDFs. As the shellcode which triggers malicious functionality is embedded in certain PDF objects, those corresponding structural paths should be conserved in each variant crafted from the same malicious seed. In the example of Figure 1, the PDF file has 7 structural paths, among which `/OpenActionJS` and `/OpenActionS` are conserved features, as the JavaScript code is placed in the corresponding objects of these two structural paths. On the other hand, there exist features that are irrelevant to malicious functionality. For example, the structural path `/Type` is unessential to preserve malicious behaviors, and we do not expect it to be conserved.

As discussed by Xu et al. [37], three fundamental operations can be used to craft adversarial examples, which directly modify PDF objects and the corresponding structural paths: insertion, deletion, and swap. The insertion operation does not change malicious functionality as it only inserts an external object after a target object. In contrast, the deletion and swap operations may impact malicious functionality by removing or replacing structural paths corresponding to it. Hence, in our case study of SL2013, conserved features are structural paths that would be neither deleted nor replaced with an external object while preserving malicious functionality.

B. Classifying with Conserved Features

We begin by exploring the effectiveness of using conserved features for classification.

For each classifier in our case study, we use a conserved feature set based on EvadeML and the 40 malicious seeds discussed in Section IV. For SL2013 we identify 8 conserved features (out of >6000), 7 conserved features for Hidost (out

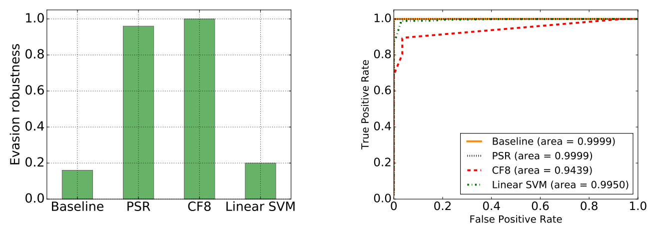

Fig. 6: Classifying with conserved features: comparing evasion robustness (left) and ROC curves (right).

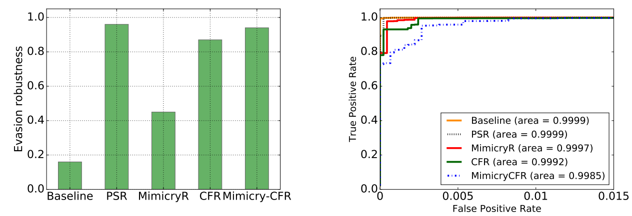

Fig. 7: Evasion robustness (left) and performance on non-adversarial data (right) of the baseline and different retraining methods using conserved features of SL2013.

of $\sim$1000), and 4 for PDFRate-B (out of 135). These are detailed in Appendix A.

The conserved feature sets raise three questions: 1) are they sufficient to make a classifier robust to evasions, 2) do they effectively discriminate between benign and malicious instances, and 3) can they be identified using standard statistical methods (such as sparse regularization)? We explore these for SL2013.

We trained this classifier using *only* the 8 conserved features (*CF8* henceforth), and evaluate both its evasion robustness, and effectiveness on non-evasion test data. To address the third question, we also learn a linear SVM classifier for SL2013 with l_1 regularization where we empirically adjust the SVM parameter C to perform feature reduction until the number of the features is also 8. We use both EvadeML and non-adversarial data set to evaluate these classifiers. The Baseline is, as before, the original SL2013 classifier.

As shown in Figure 6 (left), the classifier using only the 8 conserved features (CF8) is 100% resistant to evasion attacks by EvadeML. In contrast, the linear SVM with sparse (l_1) regularization yielding only 8 most statistically important features is easily evaded. Figure 6 (right) shows performance of the classifiers on non-evasive test data. Surprisingly enough, linear SVM using only 8 features yields better than 99% AUC, approaching the performance of the baseline classifier. More significantly, and surprisingly, even *CF8*, which is robust to evasions, achieves AUC just under 95%!

Nevertheless, the *CF8* classifier can be seen to perform relatively poorly in the region of the ROC curve where the false positive rate is low, which in practice is the most consequential part of it. Next, we consider whether the feature space retraining procedure can be fixed by incorporating conserved features into the evasion model.

C. Feature Space Retraining with Conserved Features

1) Modified Feature-Space Model: As discussed above, the feature space evasion model in Equation (1) may not facilitate sufficient boosting of evasion robustness. Since conserved features represent malicious functionality in feature space, we offer a natural modification of the model in Equation (1), imposing the constraint that conserved features are preserved in evasive instances. We formally capture this in the new optimization problem in Equation (2), where $\mathcal{S}$ is the set of conserved features:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & Q(x) = f(x) + \lambda c(x_M, x), \\ \text{subject to} \quad & x_i = x_{M,i}, \ \forall i \in \mathcal{S}. \end{aligned} \tag{2}$$

Other than this modification, we use the same coordinate greedy algorithm with random restarts as before to compute an adversarial evasion in feature space. We adopt the evasion model in Equation (2) to retrain the target classifier using the retraining procedure from Section IV. We denote the resulting feature space retraining procedure which uses conserved features by *CFR*.

Since we use uniform conserved features for each malicious seed, the malicious functionality of some PDFs may still be removed as a result. To mitigate such degradation, we optionally also generate *mimicry instances* in feature space and add these instances into the training data after the termination of retraining with conserved features. We then retrain the classifier again. In our case, we generate such instances by combining feature vectors of malicious and benign PDFs. For each benign file in the training data, we randomly select a malicious PDF, then copy all the feature of the malicious PDF to the benign one. Therefore, the resulting feature vector has both conserved and non-conserved features, by which its malicious functionality is preserved but it appears more benign compared to the malicious PDF. The resulting classifier is termed *Mimicry-CFR*.

2) Experiments:

a) SL2013: We now evaluate the robustness and effectiveness of the feature space retraining approach, which uses conserved features. We set the parameter $\lambda = 0.005$ as before. The iterative retraining process converges after 220 iterations, producing 4,461 adversarial instances. Afterward, we produce 4,496 mimicry instances and add them to the training data, and then retrain the classifier again.

We first evaluate the evasion robustness of the SL2013 baseline classifier and other classifiers obtained by different retraining. The results are summarized in Figure 7 (left). Observe that *CFR* now significantly improves evasion robustness of the baseline classifier, with evasion robustness rising from 16% to 87%. By adding mimicry instances to the training data and retraining *CFR* again, evasion robustness is further improved to 94%, which is comparable with the problem space retraining approach in Section V. These results demonstrate

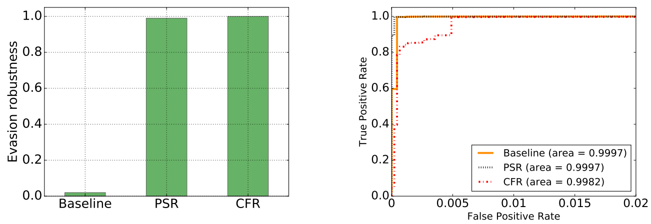

Fig. 8: Evasion robustness (left) and performance on non-adversarial data (right) of the baseline and different retraining methods of Hidost.

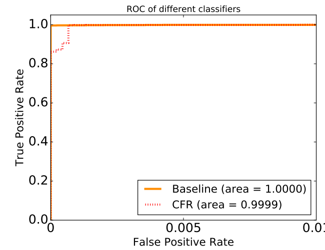

Fig. 9: ROC curves of the baseline and CFR classifier of PDFRate-B on non-adversarial test data

that by leveraging conserved features, the feature space evasion models are now quite effective as a means to boost evasion robustness of SL2013.

In Figure 7 (right) we evaluate the quality of these classifiers on non-adversarial test data in terms of ROC curves. We can observe that all variants of *CFR* classifiers are now comparable in terms of performance on non-adversarial instances to the baseline classifier, as well as the one obtained using problem space evasions, with AUC $\sim 99.9\%$.

b) Hidost: We next conduct experiments to evaluate the robustness of Hidost baseline classifier, PSR and CFR. The results are illustrated in Figure 8 (left). We can observe that by using the modified feature-space model defined in Equation (2), the evasion robustness is boosted from 2% to 100%, which is comparable with PSR and well above conventional FSR.

Evaluating the performance of CFR on non-adversarial test data in terms of ROC curves in Figure 8 (right), we find that the CFR classifier can achieve $\sim 99.9\%$ AUC, and a boost on TPR for FPR< 0.01 compared to FSR.

These results show that, by leveraging conserved features in feature space attack models, the performance on both adversarial and non-adversarial data can be significantly improved.

c) Binarized PDFRate: We now evaluate the robustness of PDFRate-B retrained by the feature space attack model in Equation (2). We observe that the CFR classifier of PDFRate-B achieves a 100% evasion robustness over EvadeML test, just as PSR and FSR counterparts investigated in Section VI-A2. However, a closer look at the ROC curve of CFR shown in Figure 9 demonstrates that CFR achieves far better performance on non-adversarial data, with >99.99% AUC, with improvements particularly stark for small false positive rates.

VIII. Identifying Conserved Features

Having demonstrated the effectiveness of conserved features in bridging the gap between problem space and feature space evasion models, we now describe a systematic automated procedure for identifying these. We first introduce how to identify conserved features of SL2013, and then describe how to generalize the approach to extract conserved features of other classifiers.

The key to identifying the conserved features of a malicious PDF is to discriminate them from non-conserved ones. Since merely applying statistical approaches on training data is insufficient to discriminate between these two classes of features, as demonstrated above, we need a qualitatively different approach which relies on the nature of evasions (as implemented in EvadeML) and the sandbox (which determines whether malicious functionality is preserved) to identify features that are conserved.

We use a modified version of pdfrw [22][4] to parse the objects of PDF file with a logic structure as shown on the right-hand side of Figure 1 and repack objects to produce a new PDF file. We use Cuckoo [12] as the sandbox to evaluate malicious functionality. In the discussion below, we define x_i to be the malicious file, $\mathcal{S}_i$ the conserved feature set of x_i, and $\mathcal{O}_i$ the set of its non-conserved features. Initially, $\mathcal{S}_i = \mathcal{O}_i = \emptyset$.

At the high level, our first step is to sequentially delete each object of a malicious file and eliminate non-conserved features by evaluating the existence of a malware signature in a sandbox for each resulting PDF, which provides a preliminary set of conserved features. Then, we replace the object of each corresponding structural path in the resulting preliminary set with an external benign object and assess the corresponding functionality, which allows us to further prune non-conserved features. Next, we describe these procedures in detail.

A. Structural Path Deletion

In the first step, we filter out non-conserved features by deleting each object and its corresponding structural path, and then checking whether this eliminates malicious functionality (and should therefore be conserved). First, we obtain all the structural paths (objects) by parsing a PDF file. These objects are organized as a tree-topology and are sequentially deleted. Each time an object is removed, we produce a resulting PDF file by repacking the remaining objects. Then, we employ the sandbox to detect malicious functionality of the PDF after the object deletion. If any malware signature is captured, the corresponding structural path of the object is deleted as a non-conserved feature, and added to $\mathcal{O}_i$. On the other hand, if no malware signature is detected, the corresponding feature is added in $\mathcal{S}_i$ as a *possibly* conserved feature.

One important challenge in this process is that features are not necessarily independent. Thus, in addition to identifying $\mathcal{S}_i$ and $\mathcal{O}_i$, we explore *interdependence* between features by

[4]The modified version is available at https://github.com/mzweilin/pdfrw.

deleting objects. As the logic structure of a PDF file is with a tree-topology, the presence of some structural path depends on the presence of other structural paths whose object refers to the object of the prior one. For example in some PDFs, the presence of `/OpenAction/JS` and `/OpenAction/S` depend on `/OpenAction`. If the object `OpenAction` is deleted, so as its structural path `/OpenAction`, then both `/OpenAction/JS` and `/OpenAction/S` are deleted as well. We call `/OpenAction/JS` and `/OpenAction/S` as the dependents of `/OpenAction`. For any feature j of x_i, the set of its dependent features is denoted by $\mathcal{D}_i^j$.

Note that for a given structural path (feature), there could be multiple corresponding PDF objects. In such case, these objects are deleted simultaneously, so as the corresponding feature value is shifted from 1 to 0.

B. Structural Path Replacement

In the second step, we subtract the remaining non-conserved features in the preliminary $\mathcal{S}_i$ and move them to $\mathcal{O}_i$. Similar to the prior step, we first obtain all the structural paths and objects of the malicious PDF file. Then for each object of the PDF that is in $\mathcal{S}_i$, we replace it with an external object from a benign PDF file and produce the resulting PDF, which is further evaluated in the sandbox. If the sandbox detects any malware signature, then the corresponding structural path of the object replaced is moved from $\mathcal{S}_i$ to $\mathcal{O}_i$. Otherwise, the structural path is a conserved feature since both deletion and replacement of the corresponding object removes the malicious functionality of the PDF file. Note that in the case of multiple corresponding and identical objects of a structural path, all of these objects are replaced simultaneously.

After structural path deletion and replacement, for each malicious PDF file x_i, we can get its conserved feature set $\mathcal{S}_i$, non-conserved feature set $\mathcal{O}_i$, and dependent feature set $\mathcal{D}_j$ for any feature $j \in \mathcal{S}_i \cup \mathcal{O}_i$, which could be further leveraged to design evasion-robust classifiers.

C. Obtaining a Uniform Conserved Feature Set

The systematic approach discussed above provides a conserved feature set for each malicious seed to retrain a classifier. Our goal, however, is to identify a single set of conserved features which is *independent* of the specific malicious PDF seed file. We now develop an approach for transforming a collection of $\mathcal{S}_i$, $\mathcal{O}_i$, and $\mathcal{D}_i^j$ for a set of malicious seeds i into a *uniform* set of conserved features.

Obtaining a uniform set of conserved features faces two challenges: 1) minimizing conflicts among different conserved features, as a conserved feature for one malicious instance could be a non-conserved feature for another, and 2) abiding by feature interdependence if a conserved feature should be further eliminated.

To address these challenges, we propose a *Forward Elimination* algorithm to compute the uniform conserved feature set for a set of malicious seeds $\{x_1, x_2, ..., x_n\}$, given the conserved feature sets, non-conserved feature sets and dependent sets for each seed. As Algorithm 1 shows, we first obtain a union of the conserved feature sets. Then, we explore the contradiction of each feature in the union with the others, by comparing the total number of the feature being selected as a non-conserved feature and conserved feature. If the former one is greater than β times the latter one, then this feature, together with its dependents, are eliminated from the union. Otherwise, the feature is added to the uniform feature set. We use β as a parameter to adjust the balance between conserved and non-conserved features. Typically, $\beta > 1$ as we are inclined to preserve malicious functionality associated with a conserved feature, even it could be a non-conserved feature of another PDF file. We set $\beta = 3$ in our experiments.

Algorithm 1 Forward Elimination for uniform conserved feature set.

Input:
The set of conserved features for $x_i (i \in [1, n])$, $\mathcal{S}_i$;
The set of non-conserved features for $x_i (i \in [1, n])$, $\mathcal{O}_i$;
The set of dependent features for $j \in \mathcal{S}_i \cup \mathcal{O}_i$, $\mathcal{D}_i^j$

Output:
The uniform conserved feature set for $\{x_1, x_2, ..., x_n\}$, $\mathcal{S}$;

$\mathcal{S} \leftarrow \bigcup_{i=1}^{n} \mathcal{S}_i$;
$\mathcal{S}' \leftarrow \mathcal{S}$;
$\mathcal{Q} \leftarrow \emptyset$;
$\mathcal{D}^j = \bigcup_{i=1}^{n} \mathcal{D}_i^j$
for each $j \in \mathcal{S}'$ **do**
  if $j \notin \mathcal{Q}$ **then**
    if $\sum_{i=1}^{n} \mathbb{1}_{j \in \mathcal{O}_i} \geq \beta \cdot \sum_{i=1}^{n} \mathbb{1}_{j \in \mathcal{S}_i}$ **then**
      $\mathcal{S} \leftarrow \mathcal{S} \setminus (\{j\} \cup \mathcal{D}^j)$;
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup (\{j\} \cup \mathcal{D}^j)$;
    end if
  end if
end for
return $\mathcal{S}$;

D. Identifying Conserved Features for Other Classifiers

Once we obtain conserved features of SL2013 for each malicious seeds, we can employ these features to identify conserved features for other classifiers using binary features. As our approach relies on the existence of malicious functionality and corresponding features, such a relation is not obvious for real-valued features; we therefore leave the question of how to define and identify conserved features in real space for future work.

a) Hidost: Hidost and SL2013 are similar in nature in such a way that they employ structural paths as features. The only difference is that Hidost consolidates features of SL2013 as described in Section III. Therefore, once the conserved features of SL2013 are identified, we can simply apply the *PDF structural path consolidation rules* described in Srndic and Laskov [33] to transform these features to the corresponding conserved features for Hidost.

b) Binarized PDFRate: We identify the conserved features for PDFRate-B by using the conserved feature set $\mathcal{S}_i$ of each seeds x_i. For each x_i, we generate $|\mathcal{S}_i|$ PDF files, each of which corresponds to the PDF file when an element (structural path) in $\mathcal{S}_i$ is deleted. We then compares PDFRate-B features of these PDFs to the original x_i. If any feature value of x_i is flipped from 1 to 0, then this feature will be added in the conserved feature set of x_i for PDFRate-B. Afterward, we use Algorithm 1 to obtain the uniform conserved feature set. This approach can in fact be used for arbitrary PDF malware detectors over binary features (leveraging conserved structural paths identified using SL2013).

IX. Alternative Evasion Attacks

So far we used EvadeML as the "ground truth" attack. This is reasonable, as EvadeML is a powerful and generic attack which makes few artificial restrictions on the attack space, in contrast to mimicry attacks, which tend to focus on adding benign features to malicious instances. Nevertheless, it is natural to wonder whether classifiers robust to EvadeML remain robust to other classes of evasion attacks.

To answer this question, we consider two additional evasion attacks: a *mimicry* attack [31], which was one of the first problem space attacks on PDF malware detectors, and a MalGAN attack, which uses a generative adversarial network to create novel evasion attacks [14]. The mimicry attack was designed specifically for PDFRate, and cannot be usefully applied to SL2013 or Hidost, whereas the MalGAN attack targets only binary classifiers.

A. Mimicry Attack

We start by considering the mimicry attack. The results, shown in Figure 10, are promising: it appears that classifiers—either PSR or FSR retrained, with and without conserved features—are robust to mimicry (recall that PDFRate-B baseline is equivalent to PSR, as it is already robust to EvadeML). Nevertheless, FSR classifiers do appear to be slightly more

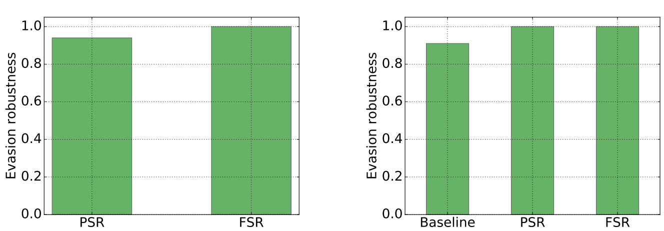

Fig. 10: Robustness to mimicry attack.

robust, but overall it appears that EvadeML is at least as strong an attack as mimicry.

B. MalGAN Attack

Now we consider another attack, MalGAN, on the three types of classifiers over binary feature space we have previously studied: SL2013, Hidost, and PDFRate-B, with PSR and FSR versions that have been shown robust to EvadeML.

The results, shown in Figure 11, are quite surprising. Despite EvadeML being a powerful attack, the PSR approaches

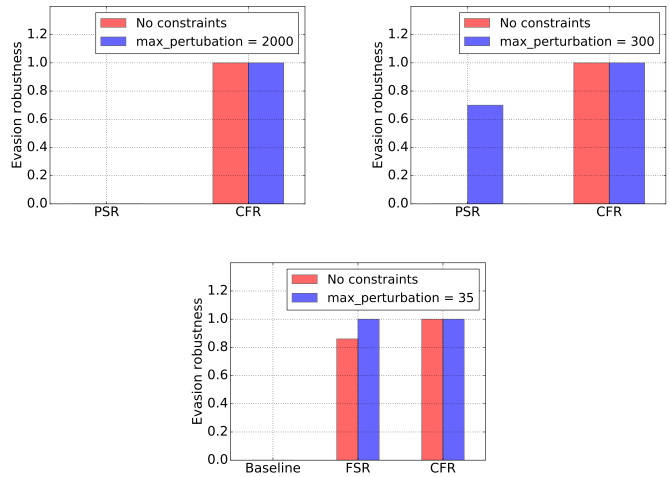

Fig. 11: Robustness to MalGAN attack. SL2013 (top left), Hidost (top right), PDFRate-B (bottom)

which use it for hardening (with resulting classifiers no longer very vulnerable to EvadeML) are *highly* vulnerable to MalGAN, with evasion robustness of 0% in most cases. In contrast, FSR models which use conserved features remain highly robust (100% in all cases). This demonstrates that besides its mathematical elegance, the abstract feature space evasion models, once appropriately corrected to consider conserved features, are potentially more generally robust to evasion attacks, as they are not hardened against a specific evasion architecture.

X. Conclusion

We systematically investigate the consequence of a feature space abstraction of evasion attacks on the ability to harden classifiers to evasion. Our study relies on several recent advances, including EvadeML, an automated, powerful, and generic evasion attack method on PDF malware classifiers which crafts actual malicious PDF instances, and a principled iterative retraining framework which allows for an arbitrary evasion model to be used in incrementally hardening a malware detector. We consider two classes of PDF malware classifiers, one with features based on logical structure, and another based on file content, with two members in each category. Our first finding is that structure-based classifiers hardened with feature space models do not adequately protect against actual evasion attacks. On the other hand, content-based classifiers appear to admit feature-space-based defense without a significant loss in evasion robustness. Moreover, quite surprisingly, a simple binarized version of the content-based PDFRate classifier is already evasion robust against EvadeML. Thus, feature space methods *may* fail, or may not, depending on the specifics of the classifier, although they generally do degrade performance on non-adversarial data. Next, we propose a generic means to bridge the modeling gap exhibited by feature space evasion models, when it does exist: using conserved features. We show that such features do exist, and can be automatically identified based on a problem space attack model. Moreover, we show that clamping down such features in feature space evasion models bridges the gap with real evasion attacks. Finally, we consider two additional attacks: a mimicry attack and one based on a generative adversarial network model (MalGAN). We observe that while the classifiers robust to EvadeML remain robust to mimicry, those retrained using a feature space model with conserved features remain robust to the MalGAN attack, while problem space retraining yields classifiers vulnerable to this attack. Thus, feature space retraining, once we account for conserved features, exhibits greater generalizability in the face of alternative attacks.

APPENDIX A
CONSERVED FEATURE SETS OF DIFFERENT CLASSIFIERS

In our experiments, we identify a subset of the features as the conserved feature set for the classifiers in our case study:

- **SL2013**:
  ```
  /Names
  /Names/JavaScript
  /Names/JavaScript/Names
  /Names/JavaScript/JS
  /OpenAction
  /OpenAction/JS
  /OpenAction/S
  /Pages
  ```
- **Hidost**:
  ```
  /Names
  /Names/JavaScript
  /Names/JavaScript/Names
  /Names/JavaScript/JS
  /OpenAction/JS
  /OpenAction/S
  /Pages
  ```
- **PDFRate-B**:
  ```
  count_box_other
  count_javascript
  count_js
  count_page
  ```

REFERENCES

[1] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *ACM Asia Conference Computer and Communications Security*, 2006, pp. 16–25.

[2] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013, pp. 387–402.

[3] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 4, pp. 984–996, 2014.

[4] M. Brückner and T. Scheffer, "Stackelberg games for adversarial prediction problems," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 547–555.

[5] ——, "Static prediction games for adversarial learning problems," *Journal of Machine Learning Research*, no. 13, pp. 2617–2654, 2012.

[6] 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*, 2010, pp. 281–290.

[7] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 99–108.

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

[9] I. Goodfellow, J. Pouget, M. Mirza, B. Xu, D. Warde, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[11] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," in *European Symposium on Research in Computer Security*, 2017.

[12] C. Guarnieri, A. Tanasi, J. Bremer, and M. Schloesser, "Cuckoo sandbox: A malware analysis system," 2012, http://www.cuckoosandbox.org/.

[13] M. Hardt, N. Megiddo, C. Papadimitriou, and M. Wootters, "Strategic classification," in *ACM Conference on Innovations in Theoretical Computer Science*, 2016, pp. 111–122.

[14] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on gan," 2017, arXiv preprint.

[15] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," in *International Conference on Learning Representations*, 2016.

[16] A. Kantchelian, J. D. Tygar, and A. D. Joseph, "Evasion and hardening of tree ensemble classifiers," in *International Conference on Machine Learning*, 2016, pp. 2387–2396.

[17] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2017.

[18] B. Li and Y. Vorobeychik, "Feature cross-substitution in adversarial classification," in *Neural Information Processing Systems*, 2014, pp. 2087–2095.

[19] B. Li, Y. Vorobeychik, and X. Chen, "A general retraining framework for scalable adversarial classification," 2016, arXiv preprint.

[20] D. Lowd and C. Meek, "Adversarial learning," in *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005, pp. 641–647.

[21] 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 *ACM Asia Conference on Computer and Communications Security*, 2013, pp. 119–130.

[22] P. Maupin, "Pdfrw: A pure python library that reads and writes pdfs," https://github.com/pmaupin/pdfrw, 2017, accessed: 2017-05-18.

[23] B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, S. Lau, S. J. Lee, S. Rao, A. Tran, and J. D. Tygar, "Near-optimal evasion of convex-inducing classifiers," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 549–556.

[24] B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, and J. D. Tygar, "Classifier evasion: Models and open problems," in *Privacy and Security Issues in Data Mining and Machine Learning - International ECML/PKDD Workshop*, 2010, pp. 92–98.

[25] B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. Tygar, "Query strategies for evading convex-inducing classifiers," *Journal of Machine Learning Research*, pp. 1293–1332, 2012.

[26] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," 2016, arxiv preprint.

[27] P. Russu, A. Demontis, B. Biggio, G. Fumera, and F. Roli, "Secure kernel machines against evasion attacks," in *ACM Workshop on Artificial Intelligence and Security*, 2016, pp. 59–69.

[28] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1528–1540.

[29] C. Smutz and A. Stavrou, "Malicious pdf detection using matadata and structural features," Tech. Rep., 2012.

[30] ——, "Malicious pdf detection using matadata structral features," in *Annual Computer Security Applications Conference*, 2012, pp. 239–248.

[31] N. Šrndic and P. Laskov, “Practical evasion of a learning-based classifier: A case study,” in *IEEE Symposium on Security and Privacy*, 2014, pp. 197–211.

[32] N. Srndic and P. Laskov, “Detection of malicious PDF files based on hierarchical document structure,” in *Network and Distributed System Security Symposium*, 2013.

[33] N. Šrndić and P. Laskov, “Hidost: a static machine-learning-based detector of malicious files,” *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 22, 2016.

[34] C. H. Teo, A. Globerson, S. Roweis, and A. J. Smola, “Convex learning with invariances,” in *Neural Information Processing Systems*, 2007.

[35] Y. Vorobeychik and B. Li, “Optimal randomized classification in adversarial settings,” in *International Conference on Autonomous Agents and Multiagent Systems*, 2014, pp. 485–492.

[36] H. Xu, C. Caramanis, and S. Mannor, “Robustness and regularization of support vector machines,” *Journal of Machine Learning Research*, vol. 10, pp. 1485–1510, 2009.

[37] W. Xu, Y. Qi, and D. Evans, “Automatically evading classifiers: A case study on PDF malware classifiers,” in *Network and Distributed System Security Symposium*, 2016.

[38] F. Zhang, P. Chan, B. Biggio, D. Yeung, and F. Roli, “Adversarial feature selection against evasion attacks,” *IEEE Transactions on Cybernetics*, 2015.

[39] Y. Zhou, M. Kantarcioglu, B. M. Thuraisingham, and B. Xi, “Adversarial support vector machine learning,” in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 1059–1067.