A Framework for Validating Models of Evasion Attacks on Machine Learning, with Application to PDF Malware Detection

Liang Tong
Vanderbilt University
liang.tong@vanderbilt.com

Bo Li
University of California, Berkeley
crystalboli@berkeley.edu

Chen Hajaj
Vanderbilt University
chen.hajaj@vanderbilt.edu

Chaowei Xiao
University of Michigan, Ann Arbor
xiaocw@umich.edu

Yevgeniy Vorobeychik
Vanderbilt University
yevgeniy.vorobeychik@vanderbilt.edu

ABSTRACT

Machine learning (ML) techniques are increasingly common in security applications, such as malware and intrusion detection. However, there is increasing evidence that machine learning models are susceptible to *evasion attacks*, in which an adversary makes small changes to the input (such as malware) in order to cause erroneous predictions (for example, to avoid being detected). Evasion attacks on ML fall into two broad categories: 1) those which generate actual malicious instances and demonstrate *both* evasion of ML *and* efficacy of attack (we call these *problem space* attacks), and 2) attacks which directly manipulate features used by ML, abstracting efficacy of attack into a mathematical cost function (we call these *feature space* attacks). Central to our inquiry is the following fundamental question: are *feature space* models of attacks *useful proxies for real attacks*? In the process of answering this question, we make two major contributions: 1) a general methodology for evaluating validity of mathematical models of ML evasion attacks, and 2) an application of this methodology as a systematic hypothesis-driven evaluation of feature space evasion attacks on ML-based PDF malware detectors. Specific to our case study, we find that a) feature space evasion models are in general not adequate in representing real attacks, b) such models can be significantly improved by identifying *conserved* features (features that are invariant in real attacks) whenever these exist, and c) ML hardened using the improved feature space models remains robust to alternative attacks, in contrast to ML hardened using a very powerful class of *problem space* attacks, which does not.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; **Systems security**;

KEYWORDS

Adversarial machine learning, PDF malware detection, classifier evasion, science of secure machine learning

1 INTRODUCTION

Machine learning (ML) has come to be widely used in a broad array of settings, including important security applications such as network intrusion, fraud, and malware detection, as well as other high-stakes settings, such as autonomous driving. A general approach is to extract a set of *features*, or numerical attributes, of entities in question, collect a training data set of labeled examples, and learn a model which labels previously unseen instances, presented in terms of their extracted features. As a concrete example of PDF malware detection, features may correspond to object paths in a PDF, metadata, such as PDF file size, and counts of *Javascript* objects, and labels indicate whether a file is malicious or benign. Success of ML in malware detection is particularly striking, with ML-based static detection of malicious entities commonly exceeding 99% accuracy [31, 32].

Nevertheless, there is increasing evidence that ML-based techniques are susceptible to *evasion attacks*. In a prototypical example of an *evasion attack*, an adversary makes small modifications to malware code so that the resulting malware is categorized as benign by ML, but still successfully executes the malicious payload [9, 21, 32, 37]. A key characteristic of such prototypical attacks is that one evaluates *both* the attacker's success in evading detection *and* preservation of malicious functionality; in our running example of PDF malware detection, the presence of malicious functionality in such an attack is commonly validated using a sandbox, such as WEPAWET [6] and the Cuckoo sandbox [12]. We term such attacks *problem space* attacks, to allude to the fact that an adversary is working directly with the problem instance (such as a malicious PDF). In practice, problem space attacks tend to be carefully engineered for a specific problem setting, somewhat ad hoc, and can be challenging to implement and replicate, especially at scale. Consequently, a natural class of *models* of evasion attacks has emerged, in which the attacker *directly manipulates the features* rather than, say, malware code (from which features are subsequently extracted) [2, 7, 16, 20, 24, 25, 36]. We call these *feature space* attacks, alluding to the fact that the attack directly manipulates features, rather than doing so indirectly through changes to the malicious entity itself.

Clearly, feature space attacks abstract away important aspects of evasion. First, they typically assume that individual features can be arbitrarily modified; in practice, features are often related, and modification of one may restrict the values that others may take. For example, including a benign object in a PDF (as a part of a mimicry attack) will necessarily increase its size, unless other modifications are also made. Even more significantly, malicious functionality is a property of the *entity* (e.g., a malicious PDF), and translates to features in complex ways. As a proxy for these two considerations, feature space models of evasion use a mathematical *cost function*, which penalizes individual feature modifications [2, 5, 7, 18, 20, 36].

The central object of our inquiry is the validity of feature space evasion attack models as proxies for real attacks. In a sense, such models (as many abstract models in security) are trivial to falsify: take two features, number of objects, and PDF size; we cannot increase the former (say, if all objects are empty) without increasing the latter. We argue, however, that such a criterion for model validity is too rigid for security, since the ultimate goal of threat modeling is to improve defense. Consequently, we propose an alternative concept of validation, with *defense* serving as the ultimate judge of efficacy. More precisely, *an attack model is considered valid if defense against this attack is effective against a real (problem space) attack for which the model is a proxy.* This idea is the basis of our first main contribution: a general methodological framework for evaluating validity of mathematical models of ML evasion attacks.

Our second main contribution is an elaboration of our ML attack model validation framework in the context of PDF malware detection, where we validate feature space evasion models with respect to a natural problem space target, *EvadeML* [37], which is arguably the most powerful problem space evasion tool for PDF malware detectors to date. Our first observation is that feature space models may be *inadequate* proxies for actual evasion attacks in this setting. Second, we introduce a novel concept of *conserved* features, or features which are (essentially) invariant in problem space attacks, and present evidence that by constraining these features to remain unmodified, feature space models become an *adequate* proxy for problem space attacks. Crucially, *our approach does not depend on the existence of conserved features*: if no conserved features can be identified, the modified model simply reduces to the conventional feature space attack model. Finally, we exhibit the real power behind abstract models of attacks: their generality. Specifically, we show that defenses designed with respect to feature space attacks remain effective against *several* real evasion attacks. In contrast, problem space attacks are highly specific, and defenses which use a particular problem space threat model can be fragile against another.

2 RELATED WORK

One of the first problem space evasion attacks on machine learning was devised by Fogla et al. [8, 9], who developed a polymorphic blending attack on anomaly-based intrusion detection systems. Šrndic and Lasov [32] present a case study of an evasion attack on a state-of-the-art PDF malware classifier, PDFRate. 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. In addition to classifier evasion methods which change the actual malicious instances, a number of techniques have sprouted for evasion models acting directly on features [1–3, 7, 13, 16, 16, 18–20, 23, 35, 38]. Moreover, a series of efforts explore evasion in the context of image classification by deep neural networks [11, 15, 17, 26, 28].

Dalvi et al. [7] presented the first approach for evasion-robust classification. 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]. A number of alternative methods for designing classifiers consider the interaction as a non-zero-sum game [4, 16, 18, 27]. Finally, a number of retraining procedures have been proposed, both for general adversarial evasion [16, 19], and specifically for deep learning methods for vision [11, 15]. 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.

3 MACHINE LEARNING IN SECURITY

3.1 Learning and Prediction

In the (supervised) machine learning literature, it is common to consider the problem abstractly. We are given a training dataset $\mathcal{D} = \{(x_i, y_i)\}$, where $x_i \in \mathcal{X} \subseteq \mathbb{R}^n$ are numeric feature vectors in some feature space $\mathcal{X}$ and $y_i \in \mathcal{L}$ are labels in a label space $\mathcal{L}$. Each data point (or example) in $\mathcal{D}$ is assumed to be generated i.i.d. according to some unknown distribution $\mathcal{P}$. We are also given a hypothesis (model) space, $\mathcal{H}$, and our goal is to identify (*learn*) a good model $h \in \mathcal{H}$ in the sense that it yields a small expected error on new examples drawn from $\mathcal{P}$. In practice, since $\mathcal{P}$ is unknown, one typically aims to find $h \in \mathcal{H}$ which (approximately) minimizes empirical error on training data $\mathcal{D}$.

In security applications—as in others—one is not given numerical features; instead, we start with a collection of entities, such as executables, along with associated labels (we assume henceforth that these are available, as we focus here on supervised learning problems). We must then *design a collection of feature extractors*, where each feature extractor computes a numerical value of a corresponding feature from an input entity. For example, we extract a "size" feature by computing the size of an executable. Applying feature extractors to each entity in our dataset, and adding associated object labels, then allows us to generate a dataset $\mathcal{D}$ to fit the conventional ML framework.

We will use PDF malware detection as a running example. In this example, the label space is binary: either a PDF file is benign (which we can code as -1), or malicious (which we can code as $+1$). In addition, several prior efforts presented techniques for defining *feature extractors* (commonly known simply as features) for PDF files [31, 32]. Applying such feature extractors to a PDF file dataset transforms this dataset into one comprised of numerical feature vectors and associated binary labels. The goal is to predict whether previously unseen PDFs (simulated by carving out a portion of our dataset as *test data*) are correctly labeled as malicious or benign.

3.2 Evasion Attacks

In an *evasion attack*, abstractly, one is given a model $h(x)$ which returns a label $y = h(x)$ (e.g., malicious or benign) for an arbitrary feature vector $x \in \mathcal{X}$ (e.g., extracted from a PDF file). The attacker additionally starts with an entity e (such as a malicious PDF file), from which we can extract a feature vector $x(e)$. The attacker then transforms e into another entity, e', with an associated feature vector $x'(e')$ so as to accomplish two goals: first, that $h(x'(e'))$ returns an erroneous label (in our running example, labels e' as benign based on its extracted features $x'(e')$), and second, that e' preserves the functionality of the original entity e—which, in our example of PDF malware detection, entails preserving malicious

functionality of e. The evasion attack as just described is presumed to transform the *entity itself*, such as the malicious PDF file, albeit accounting for the effect of such transformation on the extracted features $x'(e')$. We call attacks of this kind *problem space* evasion attacks. In fact, the process by which such problem space evasion attacks can be successfully accomplished is quite non-trivial, and typically warrants independent research contributions (e.g., [32, 37]).

In contrast, it is natural to shortcircuit the complexity involved, and work directly in the *feature space*, as is conventional in the machine learning literature. In this case, the attacker is *modeled* as starting with a malicious feature vector x (*not the malicious entity* e), and *directly modifying the features* to produce another feature vector $x' \in \mathcal{X}$, so as to yield erroneous predictions, i.e., $y' = h(x')$ (for example, being mislabeled as benign). Crucially, since we are no longer appealing to original entities, we must abstract away the notion of preserving (malicious) functionality. This is done through the use of a cost function, $c(x, x')$, whereby the attacker is penalized for greater modifications to the given feature vector x, commonly measured using an l_p norm difference between the original malicious instance and the modified feature vector [2, 19].

4 A FRAMEWORK FOR VALIDATING MODELS OF ML EVASION ATTACKS

The question of model validity is fundamental in scientific endeavor. However, in security in general, and attacks on ML in particular, it is not obvious how model validity can be fairly judged. To illustrate the challenge, consider a common criterion of falsifiability—that is, exhibiting an experiment in which the model fails to match "reality"—as applied to validating feature space evasion models. Srndic and Laskov [32] showed that feature space evasions may not even be implementable in actual PDF! We've already offered some intuition as to why: many features, such as counts of specific object types and size, are related, so that modifying one also modifies another. Even more fundamentally, a single modification to a PDF object—resulting perhaps in modification of only a single feature—may eliminate malicious functionality, whereas adding many objects (or corresponding features) from another benign PDF does not typically impact malicious functionality.

However, attack modeling in security is commonly a means to an end, where the end is better security. If secure ML is our goal, then we would wish to use defense, rather than attacks, as a means to validate attack models. To this end, we start with a conceptual model of defense and attack illustrated in Figure 1. We can view this conceptual model as a Stackelberg game between ML ("defender"), who first chooses a defense θ (in our case, the learned model $h(x)$) and the attacker; indeed, this is a common way to model the adversarial evasion problem in prior literature [4, 18]. The crucial feature of this model is that the attack is treated as a "black box": we can, in principle, design a defense against an arbitrary evasion attack, making no distinction between problem space and feature space attacks.

In practice, of course, we need a means for accomplishing this. To this end, we propose *iterative retraining*, an approach previously proposed for hardening classifiers against evasion attacks [16, 19]. Specifically, we propose to use a principled variant of iterative

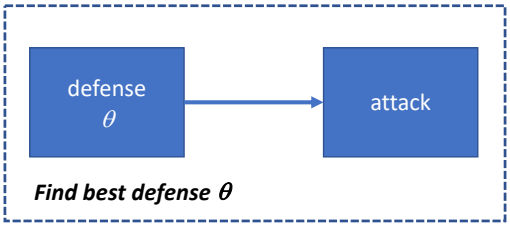

Figure 1: A conceptual model of how an attack (either in problem of feature space) can be used in improving ML security. Let defense be parametrized by θ, and an attack *reacting* to the particular defense θ (e.g., attacker evades the learned ML model $h(x)$). We wish to choose the best defense θ against such a reactive attacker, as captured by our attack model.

retraining with provable guarantees, due to [19], which is outlined as follows:

(1) Start with the initial classifier.
(2) Execute the *evasion attack* 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.

This retraining approach is illustrated in Figure 2.

Putting everything together, we propose the following framework for validating effectiveness of ML evasion models. Choose the ML algorithm (our "defense"); our goal is to improve security of this algorithm (we can also choose among multiple algorithms by directly extending the proposed approach). Next, consider a model of an evasion attack, $\tilde{O}(h)$ (e.g., a feature space attack model), which is a proxy for a "real" (problem space) attack, $O(h)$; note that each attack evades a given ML model h. Finally, let $u(h; \hat{O})$ be a measure of robustness of an ML model h against an attack $\hat{O}$ (which may be any attack), where higher is better (more robust). Now,

(1) Perform iterative retraining, using the model, $\tilde{O}(h)$; let $\tilde{h}$ be the resulting "hardened" ML model;
(2) Perform iterative retraining, using the "real" (problem space) model, $O(h)$; let h^* be the resulting ML model hardened against the real attack;
(3) The degree of validity of $\tilde{O}(h)$ *relative to* $O(h)$ (for which it is a proxy) is $u(h^*; O) - u(\tilde{h}; O)$, that is, the difference in robustness of the ML model hardened against a proxy $\tilde{O}$ compared to the ML model hardened against the real attack, *as evaluated by the real attack*; henceforth, we also call this the *gap* between a model of an attack, and the actual attack.

Thus far, our framework supposes a single "real" attack for which a model is a proxy. In fact, by virtue of models being abstractions of actual attacks, we may have multiple real attacks abstracted into a single feature space evasion model. In this case, validity can be measured by the largest gap in performance against any such real (problem space) attack.

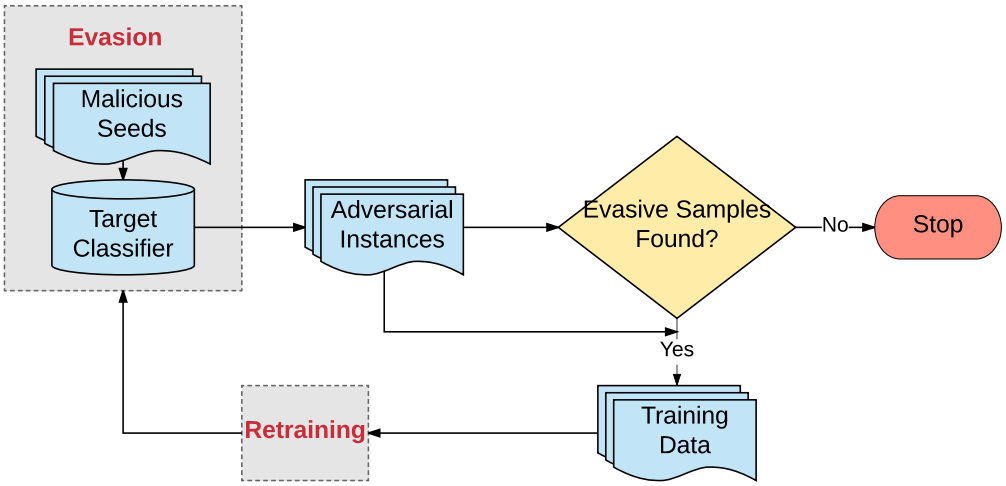

Figure 2: A general framework for retraining a classifier

In the remainder of this paper, we illustrate our framework by applying it in the context of problem and feature space evasion attacks on ML-based PDF malware detectors.

5 BACKGROUND: PDF MALWARE DETECTION AND DETECTOR EVASION

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

5.1 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 3. 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 can be referenced 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 3. 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 can 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 3. As an object can be referred to by its child node, the resulting logic structure is a directed cyclic graph. For example in Figure 3, the second and third objects refer to each other and constitute a directed cycle. To eliminate the redundant references, the logic structure can be reduced to a structural tree with the breadth-first search procedure.

5.2 The Target Classifiers

Several PDF malware classifiers have been proposed [6, 30, 31, 33]. For our study, we selected SL2013 [31] and Hidost [33]. SL2013 and its revised version, Hidost were shown to have AUC > 99.9% in detecting PDF malware. Evasion attacks on SL2013 and Hidost classifiers, particularly in the problem space, have been developed in recent literature [31, 33, 37], providing a natural evaluation framework for our purposes.

SL2013 is an 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 represent PDF files. 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 6,087 most common structural paths among 658,763 PDF files as a uniform set for classification.

Hidost is an updated version of SL2013. It inherits all the characteristics of SL2013 and employs *structural 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

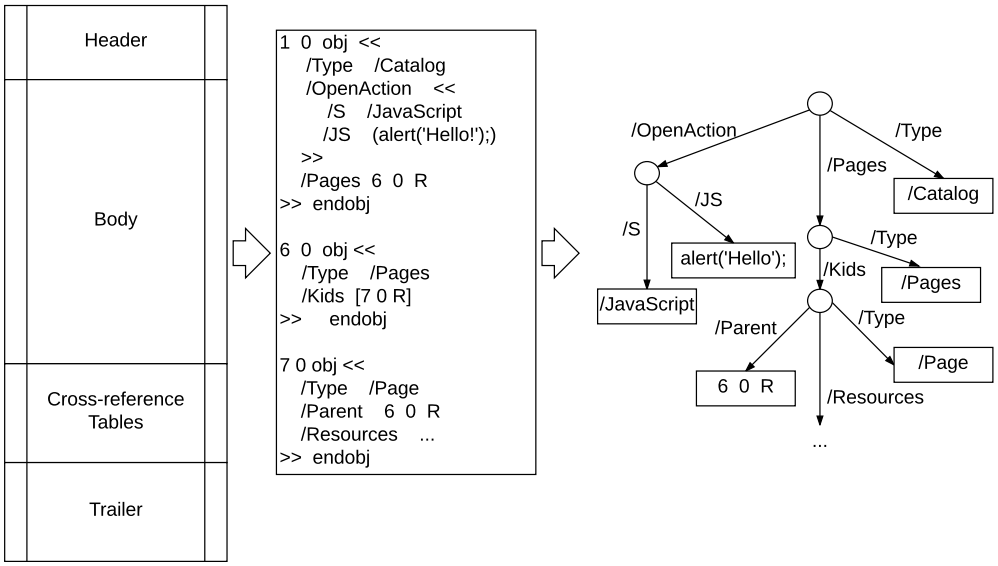

Figure 3: Various representations of the PDF structure: file structure (left), physical layout (middle), and logical structure (right)

preserves the semantics of the logical structure, so as to improve evasion-robustness of SL2013. In our work, we employ the 961 features identified in the latest version of Hidost.

5.3 Automated Evasion

5.3.1 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. There is good reason to believe *a priori* that this is the most potent evasion attack against ML-based PDF malware detectors available: nearly all other attacks are different forms of *mimicry*, attempting to add features/objects from benign files. EvadeML, in contrast, considers both addition and deletion of PDF objects in constructing a successful evasion.

EvadeML assumes that the adversary has black-box access to the classifier and can only get classification scores of PDF files. It 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 PDF repeatedly. The manipulation is either a deletion, an insertion, or a swap operation. A deletion operation deletes a target object from the seed malicious PDF file. An insertion operation inserts an object from external benign PDF files (provided exogenously) after the target object. A swap operation replaces the entry of the target object with that of another object in the external benign 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, such as API or network anomalies. The target classifier 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.

5.3.2 MalGAN. MalGAN [14] is a Generative Adversarial Network (GAN) [10] 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 detector. The results of [14] show that MalGAN is able to decrease the *True Positive Rate* on the generated examples from $> 90\%$ to 0%.

6 EXPERIMENTAL METHODOLOGY

Our main goal is to evaluate the efficacy of PDF malware 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.

6.1 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 an additional evasion attack, MalGAN, described above.

6.2 Feature-Space Evasion Model

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 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_1 norm distance between x_M and x as the cost function: $c(x_M, x) = \sum_i \alpha_i |x_i - x_{M,i}|$. 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 and variables are binary, 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.

6.3 Datasets

The dataset involved in our experiment is from the *Contagio Archive*.[1] We use 5,586 malicious and 4,476 benign PDF files for training, 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.

[1]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.

6.4 Implementation of Iterative Retraining

We made a small modification to the general iterative retraining approach described in Section 4 when it uses EvadeML as the attack $O(h)$. Specifically, 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.

We distribute both retraining and adversarial test tasks on 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 using EvadeML as the attack, 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 between the two servers in the 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.

6.5 Evaluation Metrics

Throughout, we evaluate performance in two ways: 1) evaluation of evasion robustness (which is central to our specific inquiry), and 2) traditional evaluation. To evaluate robustness, we compute the proportion of 100 malicious test seed PDFs for which EvadeML successfully evades the classifier; this is our metric of *evasion robustness*, evaluated with respect to EvadeML. 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).

7 VALIDATION OF FEATURE SPACE EVASION MODELS IN PDF MALWARE DETECTION

Our 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 re-engineered 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 nevertheless remains quite vulnerable to evasion attacks (indeed, more so than SL2013). Throughout these experiments, we use EvadeML as the target "real" (problem space)

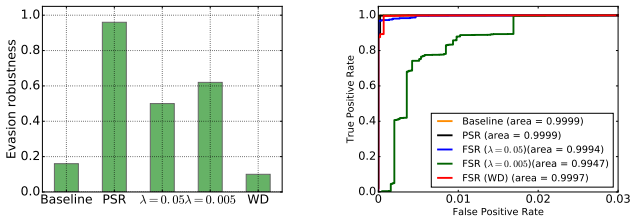

Figure 4: Evasion robustness (left) and performance on non-adversarial data (right) of the baseline and different retraining approaches for SL2013 under EvadeML test.

attack $O(h)$ for which feature space models $\tilde{O}(h)$ are proxies. We use *PSR* to refer to classifiers retrained using problem space $O(h)$, and *FSR* to refer to classifiers retrained with a feature space model $\tilde{O}(h)$.

7.1 Case Study with SL2013

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

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 16% evasion robustness.[2] 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 the termination of retraining. This is the first step in our validation framework, as it will provide a basis of comparison for the abstract feature space model of evasion.

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 4 (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).* More generally, this adds to the body of evidence that retraining classifiers with a specific attack model $O(h)$ succeeds in producing ML h robust to $O(h)$ (of course, not necessarily to other attacks) [11, 19].

Feature Space Retraining. We next conduct experiments to address our first major hypothesis which is implicit in the large body of work using such models in adversarial learning:

Hypothesis 1. *The feature space attack model provides an adequate basis for defense against real attacks.*

We now use our framework for ML evasion model validation in the context of SL2013 PDF malware detection to *reject* this hypothesis by exhibiting an experiment in which the feature space model is significantly inferior to a real attack.

[2]This result is different from the experiments in [37] which shows a 0% evasion robustness. We found a flaw in the implementation of feature extraction in EvadeML which has been reported to the authors and fixed in our experiments.

Specifically, we use EvadeML as our baseline $O(h)$ to *evaluate* evasion robustness of a classifier retrained with a feature space model $\tilde{O}(h)$. We consider the setting with uniform costs of changing each feature (weights), with $\lambda = 0.05$ and $\lambda = 0.005$ in Equation (1), and a natural attack model where feature costs (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 shown in Figure 4 (left). Compared to the SL2013 baseline, retraining with feature space evasion models (FSR) boosts evasion robustness from 16% to 60%, but only achieves a 10% evasion robustness when weighted distance (denoted as WD in Figures 4 (left) and 4 (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 may not be an adequate representation of a real evasion attack for which it is a proxy.* This observation raises a major concern with the extensive use of feature space models of ML evasion in the literature, particularly in hardening classifiers against evasion.

Performance on Non-Adversarial Data. While our focus is on adversarial evasion robustness of ML, a crucial consideration remains effective performance on non-adversarial data. It has previously been argued that retraining procedure does not greatly degrade effectiveness on non-adversarial data [16, 19]. We frame this question as the following hypothesis, for which we provide further (albeit somewhat more mixed) support below.

Hypothesis 2. *Retrained classifiers remain effective on non-adversarial data.*

We now evaluate the performance of our 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 4 (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.

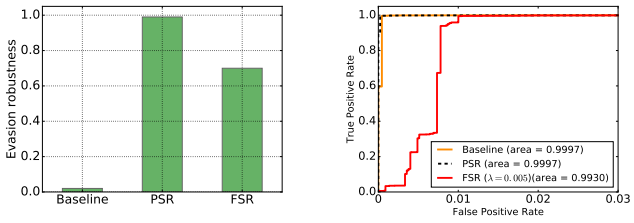

Figure 5: Evasion robustness (left) and performance on non-adversarial data (right) of the baseline and different retraining approaches for Hidost under EvadeML test.

7.2 Case Study with Hidost

Having presented evidence against Hypothesis 1 in the context of SL2013, it is now natural to ask whether the same observation can be generalized to another structure-based PDF malware classifier. We therefore repeat the experiment with Hidost, using 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. It turns out that qualitatively similar results obtain with Hidost.

Evasion robustness of Hidost, as well as improvements achieved by PSR and FSR, are shown in Figure 5 (left). First, 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 16% for SL2013). By retraining with the problem space evasion model, evasion robustness is boosted to 99%, a rather dramatic improvement. Pertaining specifically to Hypothesis 1, feature space retraining achieves a 70% evasion robustness, a significant boost over baseline, to be sure, but far below PSR. Thus, we have additional evidence that the conventionally used feature space models of classifier evasion are not adequate in representing real evasion attacks.

Evaluating these classifiers on non-adversarial test data in terms of ROC curves (Figure 5 (right)) lends further—and similarly qualified—support for Hypothesis 2. Specifically, we can observe that PSR achieves comparable accuracy (> 99.9% AUC) with Hidost on non-adversarial data, and even better *True Positive Rate (TPR)* when *False Positive Rate (FPR)* is close to zero. On the other hand, FSR achieves > 99.9% AUC, but yields some degradation of TPR when FPR< 0.01.

8 REFINING THE FEATURE SPACE MODEL OF ML EVASION

In a scientific process, falsifying a model is only the first step. The natural follow-up question is how to modify or refine a model to make it consistent with experimental data. We now propose a simple refinement to the general feature space model of ML evasion by appealing to the concept of *conserved* features. Specifically, conserved features are those which are invariant under (are unaffected by) evasion attacks. Interestingly, a similar idea of invariant features has been mentioned by Fogla et al. [9], who dismiss it as unlikely to be of practical import (because they are unlikely to exist). In any case, our refinement of the feature space model is to disallow modification of such conserved features, should they be identified. We emphasize that this refinement *does not rely on the existence of conserved features*: if none can be found, we simply obtain the original feature space evasion model.

The refined model of feature space evasion now suggests four hypotheses that we subsequently explore.

Hypothesis 3. *There are feature spaces in which conserved features can be identified.*

Hypothesis 4. *When conserved features exist, ML with only such features is robust to problem space evasion attacks.*

Hypothesis 5. *Conserved features can be recovered using conventional (statistical) regularization methods.*

Hypothesis 6. *The refined feature space model with conserved features is an adequate proxy for problem space attacks.*

Next, we present four surprising findings, which support Hypotheses 3, 4, and 6, and reject Hypothesis 5. First, conserved features can exist (they do for SL2013 and Hidost), and can be effectively identified. Second, we show that a classifier using *only* the conserved features is (a) completely robust to the EvadeML attack (partly by construction), and (b) quite effective on test data not involving evasion attacks. Third, we observe that conserved features cannot be recovered using standard conventional feature reduction, and conventional 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. Fourth, 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. In Appendix A we present a novel automated approach for identifying conserved features in the context of structure-based PDF malware detectors; as we do not claim that such features *always* exist, this suffices to provide support for Hypothesis 3. We leave the question of identifying similar conserved features for other detectors and/or in other domains as an interesting open problem.

8.1 Conserved Features

In our case study of SL2013 and Hidost, the target classifiers employ 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.

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, conserved features are structural

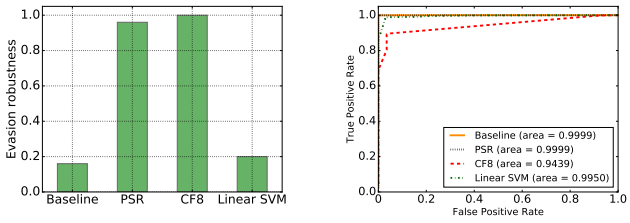

Figure 6: Classifying with conserved features for SL2013: comparing evasion robustness (left) and ROC curves (right).

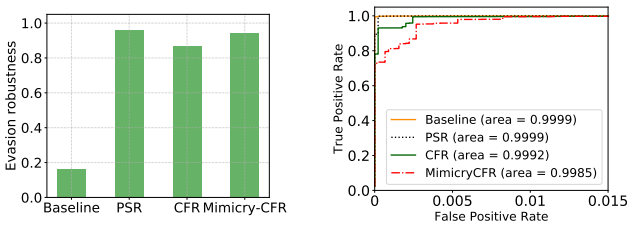

Figure 7: Evasion robustness (left) and performance on non-adversarial data (right) of the baseline and different retraining methods using conserved features of SL2013.

paths that would be neither deleted nor replaced with an external object while preserving malicious functionality.

8.2 Classifying with Conserved Features

We begin by exploring the effectiveness of using conserved features for classification; in Appendix A, we describe a systematic approach for identifying these in the context of structure-based PDF malware detectors (SL2013 and Hidost).

For each classifier in our case study, we use a conserved feature set based on the 40 malicious seeds discussed in Section 6. For SL2013 we identify 8 conserved features (out of >6000) and 7 conserved features for Hidost (out of ~1000), providing evidence for Hypothesis 3

To test Hypothesis 4, we trained the same ML algorithm as used by SL2013, but now using *only* the 8 conserved features (*CF8* henceforth), and evaluated its evasion robustness. To test Hypothesis 5, 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. In addition to evaluating robustness of the *CF8* classifier, we also evaluate its effectiveness on non-adversarial data. 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, supporting Hypothesis 4. In contrast, the linear SVM with sparse (l_1) regularization yielding only 8 most statistically important features is easily evaded, presenting evidence to reject Hypothesis 5.

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 quite 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 address the most important of our hypotheses in this section: that our refined feature space model is now an adequate proxy for "real" (problem space) attacks.

8.3 Validation of the Refined Feature Space Evasion Model with Conserved Features

8.3.1 Modified Feature-Space Model. As discussed above, the feature space evasion model in Equation (1) may not be an adequate representation of a real evasion attack. Since conserved features represent malicious functionality in feature space, we now formalize a natural modification of the model in Equation (1) described at the high level above, imposing the constraint that conserved features are preserved in evasive instances. Specifically, we capture this in the new optimization problem in Equation (2), where $\mathcal{S}$ is the set of identified conserved features (which we allow to be empty as a special case, in which case this formulation becomes equivalent to Equation (1)):

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

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 4. We denote the resulting feature space retraining procedure which uses conserved features by *CFR*.

Since we use uniform conserved features for each malicious seed (see Appendix A for details), 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*.

8.3.2 Experiments.

SL2013. We now evaluate the robustness and effectiveness of the feature space retraining approach, which uses conserved features, providing evidence in support of Hypothesis 6. 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

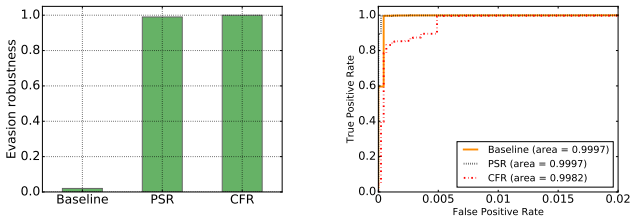

Figure 8: Evasion robustness (left) and performance on non-adversarial data (right) of the baseline, PSR and retraining using conserved features of Hidost.

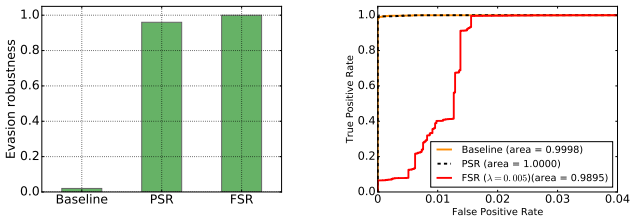

Figure 9: Evasion robustness (left) and performance on non-adversarial data (right) of the baseline and different retraining approaches for PDFRate-R under EvadeML test.

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 7. These results demonstrate 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 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%.

Hidost. We next conduct experiments to evaluate the robustness of Hidost baseline classifier, PSR and CFR, and provide further support for Hypothesis 6. The results are shown 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 presented in Figure 5 (left).

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 shown in Figure 5 (right).

9 FEATURE SPACE ML EVASION MODELS WITH CONTINUOUS FEATURES

Thus far, we provided evidence that our modified feature space model in which we clamp down identified conserved features is an adequate proxy for problem space attacks. Since a special case of our refined model when no conserved features can be identified is the original feature space model, we arrive at another hypothesis (which can be viewed as a special case of Hypothesis 6):

Hypothesis 7. *If conserved features cannot be identified, and the feature space is sufficiently rich, the conventional feature space model is an adequate proxy for problem space attacks.*

To explore this hypothesis, we consider another PDF malware detector which uses continuous features; in this case, conserved features appear unlikely, since an adversary can surely effect a small modification to any feature. For this purpose, we use the PDFRate PDF malware detector. PDFRate uses features based on PDF file content, rather than logical structure [30]. 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 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 [32]. Mimicus trained a surrogate SVM classifier with the documented 135 features and the same dataset with PDFRate, and showed it to perform comparably to the random forest classifier (the surrogate was shown to have $\sim$ 99% accuracy on the test data). 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. We therefore refer to this variant of PDFRate as PDFRate-R (where "R" refers to its standardized *r*eal-valued features).

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 SL2013 and 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.

We begin by replicating the EvadeML evasion robustness evaluation of the baseline classifier. As expected, we find the classifier quite vulnerable, with only 2% evasion robustness.

Next, we retrain PDFRate-R with EvadeML for 10 iterations (PSR baseline), and perform feature space retraining using the conventional feature space model above. As shown in Figure 9 (left), both PSR and FSR achieve high evasion robustness; indeed, FSR is slightly better at 100% robustness, with PSR at 96%. Consequently, our experiment presents evidence in support of Hypothesis 7: when conserved features cannot be meaningfully defined or identified, conventional feature space evasion models indeed perform adequately as proxies for real attacks.

Comparing PSR and FSR performance on non-adversarial data (Figure 9 (right)), we begin to observe some meaningful degradation for FSR: while PSR remains exceptionally effective (>99.99% AUC), FSR achieves AUC slightly lower, but still at a respectable 99% level.

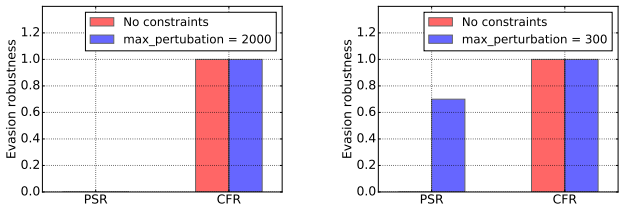

Figure 10: Robustness to MalGAN attack: SL2013 (left) and Hidost (right)

10 EVALUATION WITH AN ALTERNATIVE EVASION ATTACK

So far we used EvadeML as the "real" problem space attack. This is a reasonable starting point, as EvadeML is a powerful and generic attack which makes few artificial restrictions on the attack space. In contrast, most other attacks are forms of *mimicry*, only allowing an adversary to add benign features to malicious instances. Since the true attack space is far broader, relying solely on mimicry to provide our primary evidence would have been inherently suspect.

Nevertheless, abstracting away details inherent in designing a problem space attack opens up the possibility that a feature space attack is a proxy for *multiple* real attacks. It is therefore natural to wonder just how generally efficacious our refined feature space models are against other attacks. Moreover, the apparent generality of EvadeML, which makes essentially arbitrary modifications to PDF files, suggests a related question: is a classifier hardened *using EvadeML* as a threat model still effective against another attack? This question is particularly important because in the absence of a compelling abstract model of ML evasion, we would have relied on an attack such as EvadeML to construct a supposedly hardened classifier. We now frame both these questions as hypotheses to be tested.

Hypothesis 8. *Refined feature space ML evasion models are effective proxies for multiple problem space evasion attacks.*

Hypothesis 9. *A classifier hardened with the EvadeML attack cannot be evaded using other evasion attacks.*

We next provide support for Hypothesis 8, and reject Hypothesis 9. Specifically, we consider an alternative evasion attack: MalGAN attack, which is a form of mimicry (adding features which are common in benign PDFs) and targets binary classifiers (see Section 5 for details). We apply it to the two classifiers over binary feature space we have studied: SL2013 and Hidost, with PSR and CFR versions that have been shown robust to EvadeML.

The results, shown in Figure 10, are quite surprising. Despite EvadeML being a powerful attack, the PSR approaches 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, CFR 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 refined to consider conserved features, possess a rather generic power as a proxy for real attacks. The intuition behind this result is that these mathematical models are not hardened against a *specific* evasion architecture, as is inherent in any problem space attack. This result offers important support for the value of mathematical abstraction in security, *whenever the abstract model is adequately validated.*

11 CONCLUSION

We systematically investigate the consequence of a feature space abstraction of evasion attacks on the ability to harden classifiers to evasion. Our first major contribution is a general framework for validating of abstract models of ML evasion. Our second major contribution is the instantiation of this framework in the context of evasion of PDF malware detection as a hypothesis-driven validation of the efficacy of feature space ML evasion models with respect to real evasion attacks. 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.

In our experimental investigation, we first consider two PDF malware classifiers with features based on logical structure. Our first finding is that structure-based classifiers hardened with feature space models do not adequately protect against actual evasion attacks. Thus, feature space methods *may* fail to represent realistic evasion attacks. In response, we propose a simple refinement of feature space models of ML evasion by using conserved features which are constrained to remain unmodified in the abstract evasion attacks. We show that such features do exist in our setting, and can be automatically identified. Moreover, we show that clamping down such features in feature space evasion models bridges the gap with real evasion attacks, providing evidence for the hypothesis that our refined feature space model of evasion attacks can be an effective proxy for real attacks. Finally, we consider one additional attack based on a generative adversarial network model (MalGAN). We observe that classifiers 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. This observation suggests that abstract mathematical modeling of attacks in security can be a very powerful tool, if the models are adequately validated. Our methodology, in turn, can be viewed as a step towards evidence-based ML threat modeling—that is, more systematic and empirically grounded approaches to security of machine learning, and perhaps security more generally.

APPENDIX
A 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 Hidost.

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][3] to parse the objects of PDF file and repack them 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.

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 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 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.

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.

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.

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 $\mathcal{S}$;

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

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

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.

Identifying Conserved Features for Hidost. Once we obtain conserved features of SL2013 for each malicious seeds, we can employ these features to identify conserved features for Hidost using binary features as our approach relies on the existence of malicious functionality and corresponding features. 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 5. 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.

REFERENCES

[1] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. 2006. Can machine learning be secure?. In *ACM Asia Conference Computer and Communications Security*. 16–25.

[2] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion Attacks against Machine Learning at Test Time. In *European Conference on Machine Learning and Knowledge Discovery in Databases*. 387–402.

[3] Battista Biggio, Giorgio Fumera, and Fabio Roli. 2014. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering* 26, 4 (2014), 984–996.

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

[5] M. Brückner and T. Scheffer. 2012. Static Prediction Games for Adversarial Learning Problems. *Journal of Machine Learning Research* 13 (2012), 2617–2654.

[6] M. Cova, C. Kruegel, and G. Vigna. 2010. Detection and analysis of drive- by-download attacks and malicious JavaScript code. In *International Conference on World Wide Web*. 281–290.

[7] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. 2004. Adversarial Classification. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*. 99–108.

[8] Prahlad Fogla and Wenke Lee. 2006. Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques. In *ACM Conference on Computer and Communications Security*. 59–68.

[9] Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee. 2006. Polymorphic Blending Attacks. In *USENIX Security Symposium*.

[10] I. Goodfellow, J. Pouget, M. Mirza, B. Xu, D. Warde, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative Adversarial Nets. In *Advances in neural information processing systems*. 2672–2680.

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

[12] Claudio Guarnieri, Alessandro Tanasi, Jurriaan Bremer, and Mark Schloesser. 2012. Cuckoo Sandbox: A Malware Analysis System. (2012). http://www.cuckoosandbox.org/.

[13] Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. 2016. Strategic classification. In *ACM Conference on Innovations in Theoretical Computer Science*. 111–122.

[14] W. Hu and Y. Tan. 2017. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. arXiv preprint.

[15] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. 2016. Learning with a Strong Adversary. In *International Conference on Learning Representations*.

[16] Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. 2016. Evasion and Hardening of Tree Ensemble Classifiers. In *International Conference on Machine Learning*. 2387–2396.

[17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. *International Conference on Learning Representations*.

[18] Bo Li and Yevgeniy Vorobeychik. 2014. Feature Cross-Substitution in Adversarial Classification. In *Neural Information Processing Systems*. 2087–2095.

[19] Bo Li and Yevgeniy Vorobeychik. 2018. Evasion-robust classification on binary domains. *ACM Transactions on Knowledge Discovery from Data* (2018). to appear.

[20] Daniel Lowd and Christopher Meek. 2005. Adversarial Learning. In *ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. 641–647.

[21] Davide Maiorca, Igino Corona, and Giorgio Giacinto. 2013. 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*. 119–130.

[22] Patrick Maupin. 2017. PDFRW: A Pure Python library That Reads and Writes PDFs. https://github.com/pmaupin/pdfrw. (2017). Accessed: 2017-05-18.

[23] Blaine Nelson, Benjamin I.P. Rubinstein, Ling Huang, Anthony D. Joseph, Steven J. Lee, Satish Rao, and J.D. Tygar. 2012. Query strategies for evading convex-inducing classifiers. *Journal of Machine Learning Research* (2012), 1293–1332.

[24] Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, Shing-hon Lau, Steven J. Lee, Satish Rao, Anthony Tran, and J. Doug Tygar. 2010. Near-Optimal Evasion of Convex-Inducing Classifiers. In *International Conference on Artificial Intelligence and Statistics*. 549–556.

[25] Blaine Nelson, Benjamin I. P. Rubinstein, Ling Huang, Anthony D. Joseph, and J. D. Tygar. 2010. Classifier Evasion: Models and Open Problems. In *Privacy and Security Issues in Data Mining and Machine Learning - International ECML/PKDD Workshop*. 92–98.

[26] Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. 2016. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. arxiv preprint.

[27] Paolo Russu, Ambra Demontis, Battista Biggio, Giorgio Fumera, and Fabio Roli. 2016. Secure Kernel Machines against Evasion Attacks. In *ACM Workshop on Artificial Intelligence and Security*. 59–69.

[28] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. 2016. 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, 1528–1540.

[29] C. Smutz and A. Stavrou. 2012. *Malicious PDF Detection Using Matadata and Structural Features*. Technical Report.

[30] C. Smutz and A. Stavrou. 2012. Malicious PDF detection using matadata structral features. In *Annual Computer Security Applications Conference*. 239–248.

[31] Nedim Srndic and Pavel Laskov. 2013. Detection of Malicious PDF Files Based on Hierarchical Document Structure. In *Network and Distributed System Security Symposium*.

[32] N. Šrndic and P. Laskov. 2014. Practical Evasion of a Learning-Based Classifier: A Case Study. In *IEEE Symposium on Security and Privacy*. 197–211.

[33] Nedim Šrndić and Pavel Laskov. 2016. Hidost: a static machine-learning-based detector of malicious files. *EURASIP Journal on Information Security* 2016, 1 (2016), 22.

[34] Choon Hai Teo, Amir Globerson, Sam Roweis, and Alexander J. Smola. 2007. Convex learning with invariances. In *Neural Information Processing Systems*.

[35] Yevgeniy Vorobeychik and Bo Li. 2014. Optimal randomized classification in adversarial settings. In *International Conference on Autonomous Agents and Multiagent Systems*. 485–492.

[36] H. Xu, C. Caramanis, and S. Mannor. 2009. Robustness and Regularization of Support Vector Machines. *Journal of Machine Learning Research* 10 (2009), 1485–1510.

[37] Weilin Xu, Yanjun Qi, and David Evans. 2016. Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers. In *Network and Distributed System Security Symposium*.

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

[39] Yan Zhou, Murat Kantarcioglu, Bhavani M. Thuraisingham, and Bowei Xi. 2012. Adversarial support vector machine learning. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1059–1067.