A Novel Adversarial Example Detection Method for Malicious PDFs Using Multiple Mutated Classifiers

Chao Liu[1], *Chenzhe Lou*[1, 2], *Min Yu*[1, 2*], *S.M. Yiu*[3], *K.P. Chow*[3], *Gang Li*[4], *Jianguo Jiang*[1], *Weiqing Huang*[1]
[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]The University of Hong Kong, Hong Kong
[4]School of Information Technology, Deakin University, VIC, Australia
*corresponding author
yumin@iie.ac.cn

***Abstract* — PDF malware remains as a major hacking technique. To distinguish malicious PDFs from massive PDF files poses a challenge to forensic investigation. Machine learning has become a mainstream technology for malicious PDF document detection either to help analysts in a forensic investigation or to prevent a system being attacked. However, adversarial attacks against malicious document classifiers have emerged. Crafted adversarial example based on precision manipulation may be easily misclassified. This poses a major threat to many detectors based on machine learning techniques. Various analysis or detection techniques have been available for specific attacks. The challenge from adversarial attacks is still not yet completely resolved. A major reason is that most of the detection methods are tailor-made for existing adversarial examples only. In this paper, based on an interesting observation that most of these adversarial examples were designed on specific models, we propose a novel approach to generate a group of mutated cross-model classifiers such that adversarial examples cannot pass all classifiers easily. Based on a Prediction Inversion Rate (PIR), we can effectively identify adversarial example from benign documents. Our mutated group of classifiers enhances the power of prediction inconsistency using multiple models and eliminate the effect of transferability (a technique to make the same adversarial example work for multiple models) because of the mutation. Our experiments show that we are better than all existing state-of-the-art detection methods.**

Keywords—Adversarial example, Malicious document detection, Document classifier, Machine learning

I. INTRODUCTION

Deep learning techniques and especially Deep Neural Networks (DNN) have emerged as one of the primary techniques employed extensively in academic communities as well as industries. Their applications have been found in various areas such as malware analysis [1], spam detection [2] and network intrusion detection. DNN shows effective performance on resource-demand task, such as image recognition [3], natural language processing [4], and speech recognition. However, systems that employ machine learning classification have been demonstrated to be vulnerable to adversarial environments with novel evasion attacks [5], i.e., adversarial example [6], which are normal examples imposed on small, human imperceptible changes. This not only disrupts many classification systems [7], but also provides better conditions for many existing attacks, such as APT attacks. Thereby causes greater harm to key security industries [8].

The issue triggered a broad interest in researchers in detecting or defending against adversarial example. Able to detect malicious PDFs from massive PDF files imposes a big challenge to the forensic investigators as well since it is not easy to identify this attacking point (if it is from a PDF malware) based on a huge volume of documents received by the staff of the victim company. In computer vision, methods have been proposed to improve the robustness of the DNN model, such as the adversarial training [7], which includes different adversarial examples to the training set, making the retrained model robust to the corresponding examples. However, they are ineffective against new types of adversarial example that not previously discovered. Masking gradients [9] can also enhance the models to some extent, but new attacks have recently been produced on the basis of the transferability of adversarial example (a technique to make adversarial example pass the classification in more than one model). Alternative approaches [10] mainly detect adversarial example like the malware detection process. Xu et al. [11] proposed to generating a new example using feature squeezing technique and judging the result by comparing the prediction inconsistency. Ma et al. [12] proposed a novel technique to extract DNN invariants and use them to perform runtime detection. Malicious PDFs are most studied in the field of adversarial machine learning [13-20, 25-30]. The defense technique is basically the same as DNN, but the detection cannot be easily transplanted. This is determined by the differences in PDF format, structure, and detection system. Liu et al. [13] proposed a new feature extractor FEPDF to prevent flawed documents from evading the classifier. Smutz et al. [14] proposed the ensemble classifier mutual agreement analysis to identify evasion in malicious PDFs detectors. However, the test of prediction inconsistency between models was not carried out in this work, resulting in the failure to form an effective detection method.

In this paper, we propose a novel and an interesting method to detect PDF adversarial example, as inspired by the basic principles of prediction inconsistency and model transferability. The approach is based on the observation that adversarial example most likely exists around the boundary lines in the

machine learning detector. For example, in the prediction stage (decision tree voting) of an integrated classifier (random forest), different sub-classifiers for adversarial example are more likely to have different predictions [14]. This observation remains valid across different types of machine learning models. So the authentic benign examples will tend to be stable in the prediction of models; but adversarial example may not pass all prediction models. Therefore, our method first builds a model with good performance and takes its data set and various variable parameters as input to generate mutated models since adversarial example may not be robust across different mutated classifiers. The prediction inconsistency of the examples is evaluated by calculating the Prediction Inversion Rate (PIR, basically is the ratio of classifiers that consider the input as malicious), and the adversarial examples with large PIR deviations will become conspicuous. The experimental results show that we can effectively detect adversarial example against the state-of-the-art attacking methods.

We make the following contributions:

- We proposed a detection method that uses a novel idea to generate cross-model mutated classifiers that can identify the malicious PDFs effectively based on one simple, but effective PIR index. Unlike most other detectors, this method focuses on adversarial example and its essential attribute.
- We confirmed that the transferability of the adversarial example will be significantly affected in multiple models, and the prediction inconsistency will be more obvious.
- Our experiments show that our approach is better than the state-of-the-art detection method. In particular, for adversarial examples in the category of EvadeML-H (a specific class of adversarial example), we have an improvement of 45% (AUC values improved from 0.68 to 0.99) compared to the best detection method.

The rest of the paper is organized as follows. We present the necessary knowledge about our study in Section II, and our proposed detection method in Section III. In Section IV, we report the result and evaluation of the experiments. Section V concludes our work.

II. BACKGROUND AND RELATED WORK

This section introduces the related work, including the proposed adversarial attacks and defenses, the basic concepts on Portable Document Format (PDF) and machine learning detection methods.

A. Portable Document Format (PDF)

The structure of a PDF document consists of 4 parts: header, body, cross-reference table (xref), and trailer, as shown in the top of Fig. 1. The header defines the interpreter format version to be used. The body specifies the content of the PDF and contains text blocks, fonts, images, and metadata regarding the file itself. It contains a set of PDF objects that constitute the content of the document. These objects can be one of eight basic types: Booleans, numbers, strings, streams, names, arrays, dictionaries and the null objects. Each object starts with an object number followed by a generation number. The generation number should be incremental if additional changes are made to the object. The xref indexes the objects of body, and the trailer provides methods of finding xref and special objects.

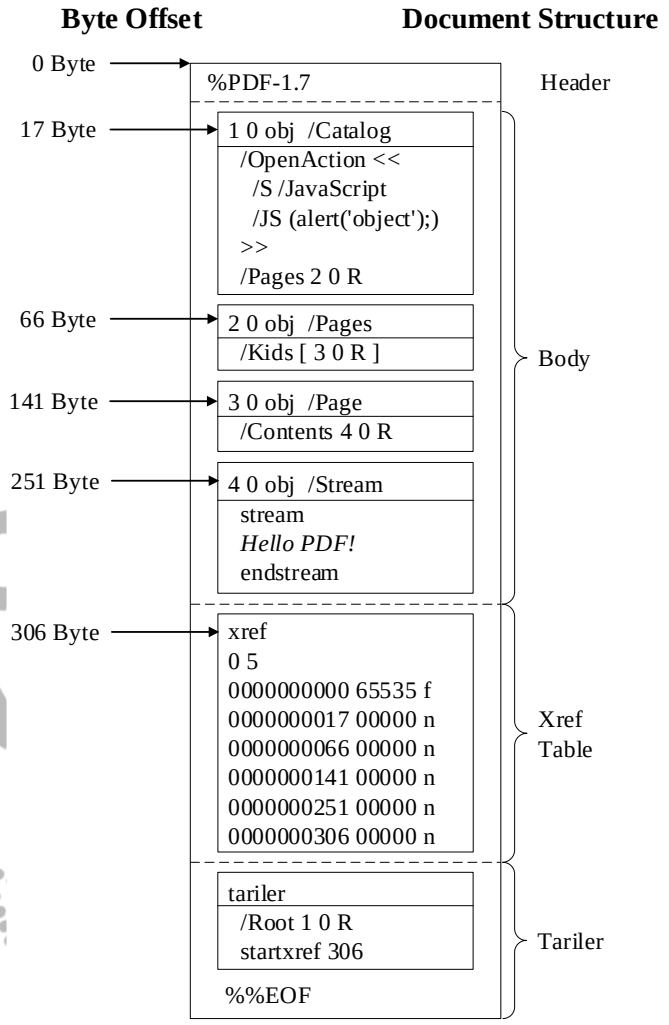

Fig. 1. The structure of a PDF document.

B. Machine Learning For Malicious PDFs Detection

Supervised machine learning has been widely deployed for malicious documents detection. In particular, concerning PDF files, multiple detectors were developed in the last decade that implemented such technology. The primary goal of machine-learning detectors for malicious document detection is to decide whether some unseen PDFs should be labelled as malicious or benign. They can operate by analyzing and classifying information retrieved either from the structure or the content of document. In general, their structure is shown as in Fig. 2, which is composed of 3 main parts [15]: Pre-processing parses PDF and access to information that is crucial for detection. Feature extraction operates on the information by converting it to a normalized vector. Classifier selects the appropriate learning algorithm for training and adjustment, and obtains better parameters to ensure a good prediction. Feature extraction is essential because the quality of features may affect the prediction performance differently.

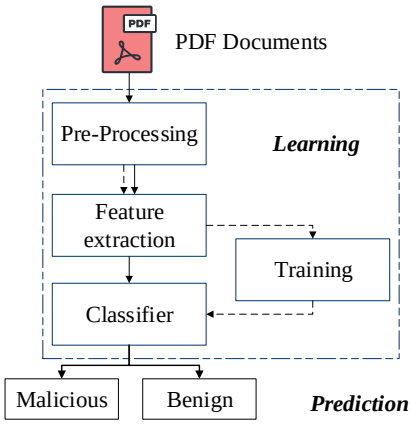

Fig. 2. Machine learning detection process for malicious PDFs.

C. Adversarial Attacks

Adversarial attacks on document detection systems are also called the evasion attacks. They take advantage of knowledge of how the machine learning system operates, and utilize access to the training set or features, to evade detection skillfully. We refer the examples using adversarial attacks as adversarial examples. Similar to image adversarial example, document adversarial example can be generated using two major approaches: content based approach and feature space based approach.

Content based approach. An intuitive way to generate adversarial example is to modify the content of the document, by taking advantage of defects in the official document specifications, or omissions of machine learning system. Flawed documents, mimicry and reverse mimicry are typical attacks.

Among them, flawed documents [13] are the vulnerabilities created by taking advantage of differences between PDF parsers and PDF document specifications. The main manifestation is that the PDF parser accepts flawed documents and tries to find such defects by itself to correct the parsing results. Although PDF document specifications recommend many rules, many PDF readers including the Adobe Reader do not strictly follow those rules. Therefore, an attacker can easily hide malicious code into flawed documents. Most existing feature extractors are unable to extract the inserted malicious code. Flawed documents may destroy the pre-processing component of machine learning system and form a successful evasion.

The mimicry attack [14] is well-known in the security literature. Its idea is to modify existing malicious documents to appear more like benign documents. Similar to the operation of flawed documents, mimicry attacks add additional structural or content data into the document. These additions do not involve adding actual content that is interpreted by a compliant PDF reader, but rather take advantage of weaknesses in universal feature extractors. Implemented tools Mimicus [16] constructs these additions by comparing a malicious document to multiple different benign documents. The feature vectors for the malicious documents are adjusted to mirror the feature vectors for the benign documents. However, the mimicry attack requires knowledge of the feature set used in the model.

Instead of adding content to a malicious document to make it appear benign, reverse mimicry [17] embed malicious content into a benign PDF, and try to modify as little as possible. In order to evade detection, reverse mimicry focus on changing the document structure as little as possible. PDF reader will not display the content associated with the previous document, but the artifacts will be analyzed by the feature extractor of detectors. Mimicry and reverse mimicry escape classifiers by destroying feature extraction components of machine learning systems.

Feature space based approach. The second attack against detectors is based on a method of modifying feature vectors in feature space. This method uses the decision boundary of the detector to iteratively modify the feature vector of the document. Resulting adversarial examples actually evade the classifier component of the machine learning system. Representative attacks include gradient descent attacks and EvadeML attacks.

The gradient descent attacks [18] work by following the gradient of the classifier's decision function and the loss function representing the effect of the prediction results. The starting point of the gradient descent is the feature vector of the malicious example. The goal is to move to the area where the classification algorithm classifies points as benign in feature space. Although starting from a malicious feature vector, directly modifying features can greatly reduce the complexity, but how to limit the modification area to better generate adversarial example is a problem that needs to be continuously solved.

EvadeML attacks [19] uses genetic programming techniques to perform a directed search of the space of possible examples to find ones that evade the classifier while retaining the desired malicious behavior. Compared with gradient descent attacks, EvadeML improves the success rate of generating adversarial example and proves that machine learning methods have defects in the feature space. In fact, EvadeML not only destroys the classifier component of the machine learning system, but also affects the other two components, rendering the entire system unstable. In conclusion, most of the adversarial examples are designed based on the features and details of the classifiers.

D. Defense, Detection and Challenge

One immediate strategy to defend against adversarial example is adversarial training. Using known attack methods, artificially generate different types of adversarial examples and add them to the training set. This will make the retrained model more robust to the corresponding adversarial example. But when faced with a new type of adversarial example that has not been discovered before, it is likely to fail.

Smutz et al. [14] proposed to use the support vector machine (SVM) as the basic classifier to build an ensemble classifier, and then using the ensemble classifier mutual agreement analysis to prevent the adversarial example generated by gradient descent attacks. On one hand, the method defends by discarding uncertain samples, which sacrifices resources and increases the cost of manual analysis. On the other hand, adversarial example has been found that they can be transferred across different models, this property is defined as transferability. This method ignores transitivity, which will pose new challenges to the robustness of the model.

Liu et al. [13] designed a robust feature extractor FEPDF and used it in a machine learning detector. FEPDF mainly studies flawed documents and plays a role in feature extraction

components. FEPDF parses PDF that follow standard document specifications, and scan every suspicious segment in documents. By cleaning PDFs, more realistic features can be obtained, the avoidance of deformed documents is invalidated, and the detector's prediction is more reliable.

Zhou et al. [20] proposed a method for detecting malicious documents using document entropy time series as features. This method is still essentially a machine learning method, but only a variant of the content structure is added from the perspective of the feature space. Therefore, if the upper and lower limits of the entropy sequence are modified, corresponding adversarial example can still be generated.

III. METHODOLOGY

In response to above challenges, this paper proposes a detection method based on one essential attribute of adversarial example. The essential attribute is that existing adversarial examples are usually generated for a certain model, and even if affected by transferability, adversarial examples are likely to fail on another similar model. Our idea is to find these models that invalidate the adversarial example. Our approach uses a set of deforming models to eliminate the effects of transferability. Then we use the prediction inconsistency of each model to validate our method and get the detection results.

This section introduces the generation rules of the detection model group, the calculation method of PIR, and the design of the detection system.

A. Generate detection model group

We take a similar strategy as in the fuzzing [31], which is a method of discovering software vulnerabilities by providing "*unexpected input*" to the target system and monitoring abnormal results. We can think exactly the opposite way. We detect adversarial input by introducing input into "*unexpected target models*" and detecting their results. Therefore, generating a series of detection models to verify their prediction inconsistency is the basis of our method. A related work in [21] has been proposed to introduce a set of mutation operators (changeable parameters for controlling the model) for DNN-based systems at different levels. Thereby generating a series of mutants using the original DNN model as input. Our strategy is similar to them, but with some key differences in the specific implementations. DNN model can be changed quickly by modifying neurons. Machine learning models can also be changed, but mainly rely on some training parameters. Once the training parameters are changed, the model must be retrained and performance will be affected.

There is no consensus on which machine learning algorithms are used in malicious document detectors. In the previous related work, SVM and random forest are the popular models with satisfactory performance. Hence, for the method using random forest, the concept of defense using prediction inconsistency has been proposed. But later work [13] took performance tests of each algorithm model, using the same data set and features. The results show that when the performance of the Pre-processing component and the Feature extraction component is more advanced, the model of the classifier is similar in prediction. Therefore, using this observation, we can extend the prediction inconsistencies to multiple types of models.

Algorithm 1: Detection model group generation

Input: Basic models ***B***, Mutation operators **O**, Original model ***f***.
Output: Detection model group ***F***.

```
Let stop = n; # Set the number of models
Let b = SVM; # b is the basic model for generating
While !stop do
  b = Random(B);
  If b == SVM Then # Taking SVM as an example, the Logistic Regression process is similar
    o = O.SVM; # Take the mutation operator according to model
    kernel = Random(o.kernel);
    C = C0; # C0 is the C value of the original SVM model
    S = Training(C, kernel.gamma); # Generative model
    If Testing(S) >= Testing(f) Then
      Add(F, S);
    stop = stop – 1; Continue;
  If b == DecisionTree # Taking Decision Tree as an example, the remaining three processes are similar
    o = O. DecisionTree;
    max_depth = Random(50, 80) # Set the max_depth below 80
    min_samples_leaf = Random(1, 2 ,3) # Parameters are set by options
    max_leaf_nodes = Random(50, 100)
    D = Training(max_depth, min_samples_leaf, max_leaf_nodes)
    # Leave other parameters of the decision tree as default
    Compare(Testing(D), Testing(f)); # Non-strict comparison
    stop = stop – 1; Continue;
Return F;
```

Our group of detection models will contain mutations from the following 6 basic machine learning models: SVM, k-Nearest Neighbor (kNN), Naive Bayes, Logistic Regression, Decision Tree and Random Forest. For each model, we selected the corresponding parameters as mutation operators, which will be introduced separately below.

SVM. There are two main parameters that can be adjusted when building the SVM model, *C* value and the kernel function. The kernel function is set to four optional values: linear kernel function, polynomial kernel function, RBF kernel function, and Sigmoid kernel function. In addition to the linear kernel function, the other three kernel functions can also adjust the parameter gamma to change the model. The *C* value represents the model's penalty coefficient for the error, and the gamma reflects the distribution of the data after it is mapped to the high-dimensional feature space. When training models, we usually use grid search to find the most suitable *C* value and gamma. So we can slightly modify the obtained parameters to generate new models.

kNN. The adjustable parameters of kNN are the number of neighbouring points (*n_neighbors*) and the distance metric. When the metric uses the Euclidean distance, it can meet the prediction of the KNN model. Therefore, we only use *n_neighbors* as the mutation operator of KNN, and limit it to an integer interval [2, 5].

Naive Bayes. Since the feature sequence is multivariate discrete value in malicious document detection, the naive Bayes Multinomial (NB) with a prior polynomial distribution is used. There are three parameters that can be manipulated: the prior smoothing factor alpha, the Boolean parameter *fit_prior*, and the prior probability *class_prior*. The value of alpha can be adjusted around the default value of 1, and *fit_prior* and *class_prior* are a pair of associated parameters. If *fit_prior* is false then *class_prior* is invalid, otherwise *class_prior* default to m_k / m. Here m is the total number of samples in the training set, and m_k is the number of samples in the training set with k classes.

Logistic Regression. In logistic regression, we generally set the penalty for L_2, which represents the penalty term, so the two remaining parameters can be adjusted: the optimization method solver and the inverse of the regularization coefficient C value. Solver can choose from liblinear, lbfgs and newton-cg. The C value is consistent with that in SVM.

Decision Tree. In the decision tree, combined with the size of our experimental data set and previous experience, we take three parameters as mutation operators: the maximum depth of the decision tree *max_depth*, the minimum number of leaf nodes *min_samples_leaf*, and the maximum number of leaf nodes *max_leaf_nodes*. For *max_depth*, we limit its value to 80 or less, for *min_samples_leaf*, we provide 1, 2, and 3 for selection, and for *max_leaf_nodes*, we limit its value to 100 or less.

Random Forest. Random forest is an integrated tool that builds multiple decision trees and then merges their predictions together to get more accurate and stable predictions. Therefore, the parameters we choose on the random forest are the same as those on the decision tree, except for one unique parameter: *n_estimators*. The *n_estimators* represents the number of decision trees. If *n_estimators* is too small, it is easy to underfit, but *n_estimators* has an increase limit. We adjusted downwards *n_estimators* at the highest performance to generate mutants.

According to those 6 basic models and their corresponding adjustable parameters, a group with a large number of models can be generated, and the process is described in Algorithm 1. With a large number of models having quality classification ability in the group, the transferability of adversarial example can be reduced to a certain extent. This model group will be used for the next step to complete the detection, so we call it the generation of detection model group.

B. Detection method

After the generation of detection model group, the next step is to input the testing examples into the generated models sequentially, and record the prediction results of each model. Then we will calculate the prediction inversion rate (PIR) of examples based on all prediction results, and compare it with the PIR of the benign examples to determine. The principle of detection using PIR is shown in Fig. 3. When a benign example enters the original model, the prediction is benign (0), the prediction reverse if the model in the detection model group classifies the input as malicious (1). And the same change occur when the input is an adversarial example. We can calculate PIR from the number of models that have reversed, and find that the adversarial example is more sensitive to the model. Therefore, we must first ensure that the models used for detection in the model group have good detection performance. Secondly, we need to calculate the PIR of benign examples.

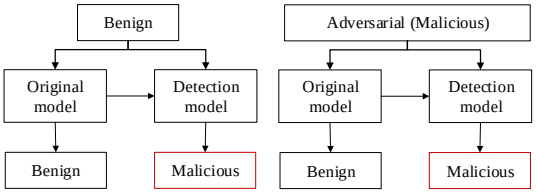

Fig. 3. The principle of detection using PIR

The process of generating a model is actually a process of training the model, and the trained model has a performance test for the test set. We select those models with a similar performance as the original model to form the final model group F. This is because mutations with poor performance do not reflect the sensitivity of adversarial example to decision boundaries.

Algorithm 2: Detection method

Input: Input example $\boldsymbol{x}$, Original model $\boldsymbol{f}$, Detection model group $\boldsymbol{F}$, Adjustment factor $\boldsymbol{m}$.
Output: Prediction $\boldsymbol{p}$.

```
Let stop = F.Len(); # Set the number of detection models used
Let p = 0; # Used to count prediction inversions
If x.label = 0 Then
    # The process is consistent, in practice need to calculate first
    Benign example PIR calculation();
While !stop do
    fi = Random(F);
    If fi(x) = ¬f(x) Then
    p = p + 1;
    stop = stop – 1;
P(x) = |{fi | fi ∈ F and fi(x) = ¬f(x)}| / |F|;
If P(x) > m·Pb Then
    Return Malicious;
Else Return Benign;
```

In the detection method (Algorithm 2), we set inputs as input example x, original model f, detection model group F and adjustment factor m. If the input example x was labeled, it is regarded as a benign example of PIR calculation in the first stage. In this stage, we will continue to input benign examples into each model, calculate the final average PIR according to the formula and record it as P_b. Then input the example to be tested, observe the original model and the detection model to see if the prediction results have reversed. Finally, calculate the PIR and compare it with the threshold $m \cdot P_b$ to get the detection result.

C. Detection System Design

In this work, the detection system is also based on a machine learning method. The overall system design is shown in Fig. 4.

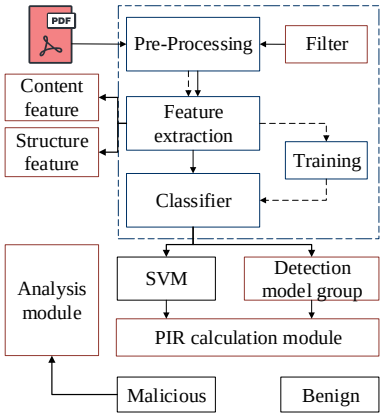

Fig. 4. The overall design of the detection system.

In the Pre-processing stage of machine learning methods, example filtering of flawed documents and mimicry attack is added. Because the main operation is to parse the PDFs, and the result will affect the subsequent performance. In the subsequent Feature extraction part, we mainly use two types of effective feature sets: content-based features and structure-based features, to ensure that the detection results are reliable. When generating

the final classifier, we first need to train an original model and ensure satisfactory performance; then generate a series of models for calculating PIR based on the detection model group generation algorithm.

After all models are generated, a PIR calculation module will be set up, which is important to detecting input examples. Finally, the results obtained by the PIR calculation module will be output. The system will determine the example with a larger PIR as an adversarial example according to the threshold, that is, malicious, and move it into the verification analysis module to collect it.

IV. Experiments and analysis

In this section, we first calculate the average PIRs, including benign examples and adversarial examples. Then, based on the experimental setting, the results of our system are compared with other detectors to evaluate the effectiveness of the method.

A. Dataset

Our experiments used two datasets: the data set *S* for training and testing, and attack data sets *M* for generating adversarial example. Since our experiments need to use different attacks to generate corresponding adversarial examples, the malicious PDFs in *M* must be truly malicious. Therefore, we continuously select the collected malicious PDFs to form a data set containing 500 examples.

We collected a total of 11,200 malicious and 10,500 benign PDFs. The malicious PDFs are mainly from Contagio archive [22], Virus Share [23] and Virus Total [24], a small number of documents were captured by us on the Internet. Benign PDFs come from Contagio archive, Google search and research institutions or companies that frequently use PDFs, including announcements and operation manuals.

B. Experimental setting

Machine. A host machine (Intel Core i7-6300 CPU @ 3.40GHz and 16GB of physical memory running 64-bit Windows 10 Desktop), an auxiliary machine (Intel Core i5-7300HQ CPU @ 2.50GHz and 16GB of physical memory running 64-bit Ubuntu 16.04 Server). The auxiliary machine deploys mainly a sandbox verification system..

Detector. We selected four state-of-the-art detectors, PJScan [25], PDFRate [26] and Hidost [27], for comparison. Among them, PDFRate and Hidost are the latest technologies with high-precision detectors. They are often used as target detectors for adversarial attacks against PDF.

Attacks. We use the mimicry attacks, gradient descent attacks (Mimicus) and EvadeML to generate adversarial example, where EvadeML can be divided into EvadeML-P for PDFRate and EvadeML-H for Hidost according to different target detectors.

C. Evaluation Indicators

The most commonly used evaluation indicators in malware detection are Accuracy, Recall and AUC(Area Under Curve). Accuracy refers to the ratio of detection results that are true positives in malicious. Recall is the percentage of correctly identified malicious examples. The AUC is equal to the probability that a randomly selected positive example ranks higher than a randomly selected negative example.

TABLE I. Confusion Matrix

	Detected as malicious	Detected as Benign
Malicious	TP	FN
Benign	FP	TN

Generally, the confusion matrix is used in malicious document detection to calculate the above four indicators, as shown in Table I. TP indicates the number of malicious documents that were correctly detected, FP indicates the number of normal documents that were misidentified as malicious, TN indicates the number of normal documents that were correctly detected, and FN indicates the number of malicious documents that were identified as harmless. The calculation method of each evaluation index is as follows:

$$\text{Accuracy} = TP / (TP + FP)$$

$$\text{Recall} = TP / (TP + FN)$$

$$\text{AUC} = (\sum_m k - n_m(n_m + 1) / 2) / (n_m \times n_b)$$

D. Results and analysis

In the classifier training phase, we use different machine learning algorithms to obtain the original model and the detection model group. Therefore we have to test the performance of the original system and ensure that it is optimal. The experiment uses a 10-fold cross validation on training. The original data is divided into 10 equal subsets. Each subset is used as a validation set, and the remaining 9 subsets are used as the training set. The average indicators of *K* models are used as the overall performance indicator of the original model. Table II lists the indicators of the final original model. It can be seen that the detection performance of the model is very high in the case of using integrated features.

TABLE II. Training results of original model

	Accuracy	Recall	AUC
SVM	0.9963	0.9972	0.9981

At the same time, we also generate detection model groups during the training phase, and we need to choose between them. For SVM and logistic regression methods, we can get more mutants, and SVM is the method used in the original model. So, we generate the optimal model, based on its parameters and used its performance as the benchmark, and select the model with the appropriate performance. For the other four methods, training and verification are performed directly. The final detection model group contains 700 models.

In order to verify that there is a boundary between adversarial and benign examples, we also need to calculate the PIR of adversarial examples generated by various attacks. Compare this to the PIR of a benign example. We need to generate corresponding adversarial examples on the attack dataset based on various attacks (the attack dataset contains 500 examples, but each attack generates no more than 500 adversarial examples). Then our experiment has the following settings when calculating the PIR: for each input example, 500 models are randomly selected in the detection model group each time, and the average PIR is calculated 10 times. Finally, the mean value of all examples of each type is calculated and recorded in Table III.

TABLE III. PIR average

Benign	Mimicry	Mimicus	EvadeML-P	EvadeML-H
3.22±0.23	45.15±3.21	36.25±2.78	32.56±2.62	14.88±1.64

It can be seen from the table that the PIR of a benign example is significantly less than the PIR of any adversarial example, which is the key to our detection method. Machine learning algorithms try to distribute benign examples far away from the decision plane. Although adversarial example is generated based on the example near the decision plane, this distance still exists in the changed model group. The PIR generated based on this distance becomes an effective means to detect the essential attributes of the adversarial example. Finally, the comparison experiment was conducted with other detectors. The AUC results and the average detection time are listed in Table IV.

TABLE IV. COMPARISON OF DETECTION RESULTS

	PJScan	PDFRate	Hidost	Our system
Original	0.8853	0.9626	0.9912	0.9981
Flawed documents	0.6624	0.9223	0.991	0.9981
Mimicry	0.6236	0.7492	0.989	0.998
Mimicus	0.5844	0.6483	0.9751	0.9978
EvadeML-P	0.4263	0.6398	0.9682	0.9975
EvadeML-H	0.4935	0.6627	0.6834	0.9974
Average detection time / s	0.622	0.734	0.875	1.242

Compare the original performance of each detector (where our system uses the indicators of the original model) and the performance after being attacked by adversarial example. It can be clearly seen that our system can effectively detect adversarial examples generated by flawed documents, mimicry attacks, gradient attacks, and genetic algorithms. There are three main reasons: First, we used comprehensive features when training the model, content-based features and structure-based features. This ensures that the model maintains the accuracy under ordinary conditions. Second, in the Pre-processing and Feature extraction stages of the original model, we added the state-of-the-art parsing method and pruning algorithm. These methods invalidate some flawed documents and mimicry attacks. Most importantly, PIR points out the essential flaws of the adversarial example. That is, the effective range of transferability is limited, and human modification has distance fluctuations. This flaw makes it easy to detect specific adversarial example (in fact, it is the examples that are dangerous). In addition, our system is slightly higher in average detection time than other detectors. The main reason is the existence of non-optimal models, although the detection model group in the detection part can use parallel computing.

V. CONCLUSION

In this paper, we propose a method for detecting PDF adversarial example based on the transferability and the corresponding prediction inconsistency. This method generates a certain number of models by making minor changes to the classification model. On one hand, the influence of transferability is eliminated in quantity; on the other hand, the prediction results of each model reflect the nature of the adversarial example. The experimental results show that the adversarial example is sensitive to the model and there is a clear boundary with the benign example. Our detection system can find this feature and detect the adversarial example.

This method can be applied to other detectors based on machine learning methods, such as detection of malicious Android applications and malicious traffic. There are some possible future extensions to the detection method proposed in this paper. For example, we can further consider stronger attacks, and more transferable adversarial examples. In addition, it is also interesting to analyze those captured unseen adversarial examples and get some more useful knowledge to strengthen our defense.

REFERENCES

[1] J. Jang, D. Brumley, and S. Venkataraman, "BitShred: feature hashing malware for scalable triage and semantic analysis," in Proceedings of the 18th ACM conference on Computer and communications security, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 309–320.

[2] G. Kakavelakis, R. Beverly, and J. Young, "Auto-learning of SMTP TCP Transport-Layer Features for Spam and Abusive Message Detection," in 25th Large Installation System Administration Conference. Boston, LISA, 2011.

[3] He K, Zhang X, Ren S, et al. "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[4] D. Andor , et al. "Globally normalized transition-based neural networks," in Proceedings of the 2016 Association for Computational Linguistics, (ACL), 2016, pp. 2442–2452.

[5] M. Davide, B. Biggio, and G. Giacinto, "Towards adversarial malware detection: Lessons learned from PDF-based attacks," ACM Computing Surveys (CSUR), 52.4 (2019): 1-36.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, "Intriguing properties of neural networks," in ICLR, 2014.

[7] I. Goodfellow, J. Shlens, C. Szegedy, "Explaining and harnessing adversarial examples," in ICLR, 2015.

[8] M. Bojarski, et al. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, 2016.

[9] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," arXiv:1412.5068, 2014.

[10] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," arXiv:1702.04267, 2017.

[11] W. Xu, D. Evans, and Y. Qi, "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks," in Proceedings of the 2018 Network and Distributed Systems Security Symposium, NDSS, 2018.

[12] S. Ma, Y. Liu, et al. "NIC: Detecting Adversarial Samples with Neural Network Invariant Checking," NDSS, 2019.

[13] M. Li, Y. Liu, et al. "FEPDF: a robust feature extractor for malicious PDF detection," In 2017 IEEE Trustcom/BigDataSE/ICESS, 2017, pp. 218–224.

[14] C. Smutz, A. Stavrou, "When a Tree Falls: Using Diversity in Ensemble Classifiers to Identify Evasion in Malware Detectors," NDSS, 2016.

[15] M. Davide, et al, "Digital Investigation of PDF Files: Unveiling Traces of Embedded Malware," IEEE Security Privacy, 17.1(2019): 63–71.

[16] N. Šrndić, P. Laskov, "Practical evasion of a learning-based classifier: A case study," in IEEE Symp. Security and Privacy, SP '14, 2014, pp. 197–211.

[17] M. Davide, et al, "A structural and content-based approach for a precise and robust detection of malicious PDF files," 2015 International Conference on Information Systems Security and Privacy (ICISSP), 2015, pp. 27–36.

[18] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, F. Roli, "Evasion attacks against machine learning at test time," ECML PKDD, Part III, Vol. 8190 of LNCS, Springer, 2013, 387–402.

[19] W. Xu, Y. Qi, D. Evans, "Automatically evading classifiers," in Annual Network & Distr. Sys. Sec. Symp. (NDSS), The Internet Society, 2016.

[20] L. Liu, et al. "Capturing the symptoms of malicious code in electronic documents by file's entropy signal combined with machine learning," Applied Soft Computing 82 (2019): 105598.

[21] L. Ma, F. Zhang, et al. "Deepmutation: Mutation testing of deep learning systems," in 2018 IEEE 29th International Symposium on Software Reliability Engineering, ISSRE, 2018.

[22] S. Chenette, (2009). *Malicious Documents Archive for Signature Testing and Research - Contagio Malware Dump* [Online]. Available: http://contagiodump.blogspot.de/2010/08/malicious-documents-archive-for.html.

[23] J. M. Roberts, (2011). *Virus Share* [Online]. Available: https://virusshare.com.

[24] V. Total, (2012). *Virustotal-free online virus, malware and url scanner* [Online]. Available: https://www. virustotal. com.

[25] Laskov, Pavel, and Nedim Šrndić, "Static detection of malicious JavaScript-bearing PDF documents," Proceedings of the 27th annual computer security applications conference (ACSAC '11), 2011, pp. 373–382.

[26] C. Smutz, A. Stavrou, "Malicious PDF Detection Using Metadata and Structural Features," in Proceedings of the 28th annual computer security applications conference, 2012, pp. 239–248.

[27] N. Šrndić, P. Laskov, "Hidost: a static machine-learning-based detector of malicious files," EURASIP Journal on Information Security, 1, 2016, 22.

[28] N. Šrndić, P. Laskov, "Detection of malicious pdf files based on hierarchical document structure," in Proceedings of the 20th Annual Network & Distributed System Security Symposium, San Diego, 2013, pp. 1–16.

[29] H. Dang, Y. Huang, E. Chang, "Evading classifiers by morphing in the dark," in ACM CCS '17, ACM, 2017, pp. 119–133.

[30] C. Smutz, A. Stavrou, "Malicious PDF Detection Using Metadata and Structural Features," in Proceedings of the 28th annual computer security applications conference, 2012, pp. 239–248.

[31] V. J. Manès, H. Han, et al. "Fuzzing: Art, science, and engineering," arXiv:1812.00140, 2018.