Received: 16 February 2021 | Revised: 26 March 2021 | Accepted: 12 April 2021

DOI: 10.1002/int.22451

RESEARCH ARTICLE

WILEY

Boosting training for PDF malware classifier via active learning

Yuanzhang Li[1] | Xinxin Wang[1] | Zhiwei Shi[2] | Ruyun Zhang[3] | Jingfeng Xue[1] | Zhi Wang[4]

[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China

[2]Office of Science and Technology, China Information Technology Security Evaluation Center, Beijing, China

[3]Intelligent Network Research Institute, Zhejiang Lab, Hangzhou, China

[4]College of Cyber Science, Nankai University, Tianjin, China

Correspondence

Jingfeng Xue, School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China.
Email: xuejf@bit.edu.cn

Zhi Wang, College of Cyber Science, Nankai University, Tianjin, China.
Email: zwang@nankai.edu.cn

Funding information

National Natural Science Foundation of China, Grant/Award Numbers: 62072037, U1936218; Zhejiang Lab, Grant/Award Number: 2020LE0AB02

Abstract

Machine learning algorithms are widely used for cybersecurity applications, include spam, malware detection. In these applications, the machine learning model has to face attack by adversarial samples. Therefore, how to train a robust machine learning model with small samples is a very hot research problem. portable document format (PDF) is a widely used file format, and often utilized as a vehicle for malicious behavior. There have been various PDF malware detectors based on machine learning. However, the labeling of large-scale data samples is time-consuming and laborious. This paper aims to reduce the size of training set while maintain the performance of detection. We propose a novel PDF malware detection method, using active learning to boost training. Particularly, we first make clear the meaning of uncertain samples in this paper, and theoretically explain the effectiveness of these uncertain samples for malware detection. Second, we present an active-learning based malware detection model, using mutual agreement analysis to choose the uncertain sample as the data augmentation. The detector is retrained according to the ground truth of the uncertain samples rather than the whole test samples in the previous epoch, which can not only improve the detection performance, but also reduce the training time consumption of the detector. We conduct 10 epochs of retraining experiments for comparison, using the uncertain samples and the whole test samples from the previous epoch respectively as training set augmentation. The experimental results show that our active-learning based model can achieve the same performance as the traditional model in the tenth epoch of retraining, while the former only needs to use one thirtieth of the latter's training samples.

KEYWORDS
active learning, machine learning, malware detection, PDF

1 | INTRODUCTION

With the continuous development of artificial intelligence, machine learning algorithms are widely used for cybersecurity applications, the Internet has entered the era of big data and cloud services, which has brought great changes to our daily social economy, education, medical treatment, finance, and so on. However, the progress of technology is a double-edged sword, which brings convenience to our life, and is also accompanied by more and more network information security problems.[1] These security problems are one of the factors limiting the prosperity of the mobile communications and have penetrated into plenty of research fields, including cloud services, and their applications.[2] Due to the existence of large-scale security problems, various research fields have to pay attention to the possible troubles, and the solutions[3-5] or technologies[6-8] for different fields have been emerging in the past period of time. Malware,[9] as the carrier of vulnerability exploitation, has always been the focus of information security research. PDF file format is widely used in commercial office due to its high efficiency, stability, and interactivity. With the development of nonexecutable file attack technology and attack methods like advanced persistent threat (APT),[10] the security of PDF has been seriously threatened, and malicious PDF files constitute the most studied infection vectors in adversarial environments.[11,12]

In recent years, machine learning technology[13] has become increasingly mature, and researchers have developed many machine learning-based systems to detect various attack types related to PDF file. Malicious PDF file detection technology can be divided into detection method based on static analysis, detection method based on dynamic analysis and detection method based on both dynamic and static analysis.[14] Most state-of-the-art works have demonstrated that machine learning-based PDF detectors could achieve high accuracy with extremely low FPR.

However, in most cases, training a great detector requires a large number of samples, whose ground truths are usually labeled by expert in specific field. Labeling large amounts of data is a time-consuming and laborious process. Additionally, even with these vast labeled samples, training a model with them will also take a long time. We can take full backup and differential backup in information storage as an example.[15,16] The former requires full backup of the whole system, causing a large amount of redundant in the storage carrier. In contrast, differential backup only backs up data that has changed since the last full backup. The latter not only reduces the backup time, but also saves the storage carrier space. It is a great strategy to introduce this idea into the detector training process. Appropriate active learning methods can effectively learn the difference information between different time periods. Making good use of the difference information, the detector will maintain its excellent performance and decrease resource consumption. Additionally, how to choose the difference information between various period is also a critical element to active learning.

To summarize, in this study we provide the following main contributions:

- We analyze and make use of uncertain samples to research their advantage in malware detection.
- We train a PDF malware detector, which utilizes active learning based on mutual agreement analysis to select a few uncertain samples and augment the training set to improve the classifier constantly.
- We conduct an experimental evaluation of the detector above, performed on Contagio data set. In the evaluation, the detector using active learning makes using less training samples and obtaining better classifier performance possible.

The structure of this paper is as follows. We will mainly elaborate the PDF malicious code detection domestic and foreign present situation in Section 2. Section 3 is the background, mainly introduces the structure of PDF file format, PDF malware and the Hidost classifier. Section 4 is the approach, which introduces the scheme design and the key technologies used by the detector. Section 5 is the experimental evaluation. The classifier model before and after active learning is compared. Section 6 is the summary and future work, mainly summarizes the experimental process, as well as raises the prospect of future research work.

2 | RELATED WORK

PDF embedded with malware have been the subject of several studies over the years. The following is the research status of this field in the past decade, which will be introduced separately from the detection technology categories for malicious PDF files. According to the features used to detect, PDF malware detection can be divided into detection technology based on static analysis, detection technology based on dynamic analysis as well as detection technology based on both dynamic and static analysis.

The detection technology based on static analysis does not need to execute the JavaScript program, but only needs to scan the file statically, so the processing speed is fast and the computer resource required is less. Shafiq et al.[17] was the first one who detected malicious files through combining machine learning technique with Markov *n*-gram static model, and adopted entropy rate for byte level features extraction in 2008. It could locate the malicious codes and compared with the previous technology, the true-positive rate (TPR) was significantly improved. Shortly afterwards, Tabish et al.[18] presented a malware detection method based on decision tree in 2009, in which feature extraction was also at byte level, and they used standard data mining algorithms to analyze the content of each block. It was based on nonsignature and could detect zero-day malware accurately. In 2011, Laskov and Šrndic[19] developed PJScan, a malicious PDF document detection technology, which was a JS-based static analysis method and could achieve 85% of the detection accuracy of all antivirus engines at that time. But compared with the most advanced dynamic models, PJScan was more prone to false-positives. During 2012, Maiorca et al.[20] proposed Slayer, which utilized pattern recognition method to extract keyword features in PDF files, and classified these files by means of random forest algorithm. It was not limited to malicious JavaScript code, but can be used to detect any type of PDF file attack. Almost at the same time, Smutz and Stavrou[21] presented PDFRate, a malicious PDF detector based on metadata and structure features. This method could get the best results even against unknown malware, with a classification rate of 99% and a false-positives rate of less than 0.2%. It could also effectively resist mimicry attacks. In 2013, Šrndić and Laskov[22] constructed a detector called SL2013, which was based on the structured path in PDF files and used decision tree as well as support vector machine (SVM) as learning algorithm. Though its ability to cope with unknown security threats was quite strong, its robustness was relatively weak. In 2015, Maiorca[23] proposed Slyer NEO, an automatic detection model, which extracted features from both structure and content to prevent evasion based on structure detection. Its accuracy was obviously better than other systems using static analysis technologies, but it was too sensitive to the training set. In 2016, Smutz and Stavrou[24] implemented PDFRate-New and designed mutual agreement analysis to improve the robustness of the detector, which used the voting status of subclassifiers in an ensemble classifier to judge whether the prediction is reliable. During the same year, Šrndić and Laskov[25] presented Hidost, an extension of their previous work SL2013, which worked effectively on a variety of file formats. Compared with SL2013, Hidost was more robust, because it constantly retrained itself with new data set to adapted to new malware features.

The detection technology based on dynamic analysis needs to execute the program in the PDF files, using sandbox or other tools for dynamic analysis. When the program is running, special data is constructed by dynamic analysis tools to trigger potential errors and analyze the results, so the calculation resources occupy a large amount. In 2010, Cova et al.[26] proposed Wepawet, an online malware detector, and used a bayesian classifier to detect malicious JavaScript code. Combining anomaly detection with dynamic simulation, it could automatically detect abnormal JS code. In 2011, Snow et al.[27] considered it unnecessary to use software-based simulation technology, and proposed a new framework SHEELOS, to quickly and accurately detect code injection attacks. They used 10 precisely designed heuristic features to get a normal JavaScript model through training. Once the JavaScript code is very different from this model, it is judged as malware.

The detection technology based on static and dynamic analysis combines the above two technologies, so it has their advantages, but the model will become more complex. In 2011, Tzermias et al.[28] developed an independent malicious document scanner MDScan, which combines static document analysis and dynamic code execution to detect previously unknown PDF threats. The evaluation shows that even if it has been confused, MDScan can find a large number of malicious PDF files. Corona et al.[29] implemented a lightweight JavaScript malware detection model Lux0R in 2014. It detected malicious JavaScript code through Application Programming Interface (API) references, and achieved excellent performance in terms of detection accuracy, throughput and generalization.

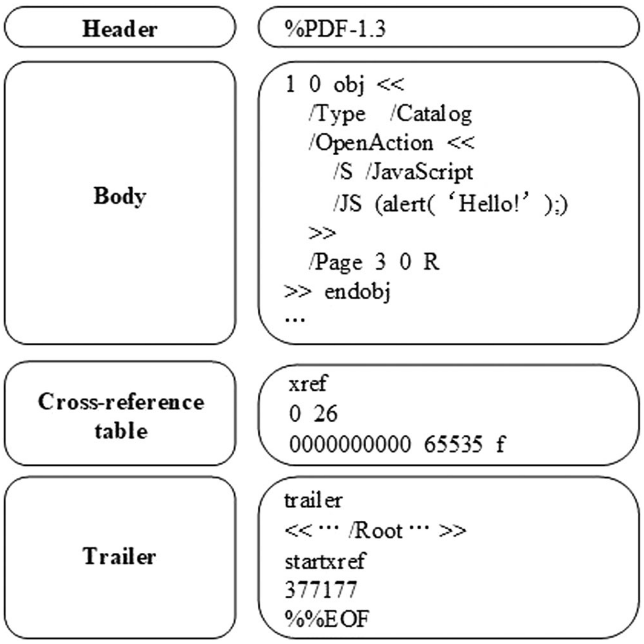
Because of the simplicity of detection technology based on static analysis, it was utilized by most of the current PDF malicious code detection methods, such as Hidost detectors, which can still achieve good performance. In the case of guaranteed performance, there are also many research on the robustness of the detector,[30] which have employed various methods, such as mutual information,[31] expert domain knowledge,[32] and information from cuckoo sandbox.[33] Even against the adversarial attack,[34] there are many studies[35] to improve the robustness of the detector. However, training a high-performance detector generally requires large-scale data set, and the cost of labeling these samples is still a problem. Thus, malicious PDF detection has become a necessary research task in information security field.

3 | BACKGROUND

3.1 | PDF file structure

PDF, short for Portable Document Format, is a type of document format that supports consistent rendering and printing, and is independent of the underlying environment. Physically, a PDF file is constructed with four components: a header, a body, a cross-reference table, and a tail[36,37] (Figure 1).

Logically, PDF is a structural document format and a PDF file consists plenty of modules called object. Each object is numbered so that it can be referenced by other objects. These objects could appear in the PDF document in any order, and the only benefit of the objects appearing in order is to increase readability of the file. The information of objects is stored in the cross-reference table. The trailer of the file indicates the object number of the root object and the location of the cross-reference table, where catalog objects can be found by querying the cross-reference table. This catalog object is the root object of the entire document,



```
%PDF-1.3

1 0 obj <<
  /Type  /Catalog
  /OpenAction <<
    /S /JavaScript
    /JS (alert( ‘Hello!’ );)
  >>
  /Page 3 0 R
>> endobj
...

xref
0 26
0000000000 65535 f

trailer
<< ··· /Root ··· >>
startxref
377177
%%EOF
```

FIGURE 1 Structure of PDF file.[36] In the figure, four components of the PDF file are clear, including header, body, cross-reference table and trailer. Header: to store the version number of the PDF file. Body: the main part of a PDF file, consists of multiple objects that define the operations to be performed by the file. Cross-reference table: the address index table of the indirect object, which can be queried for random access to the indirect object. Trailer: to store the address of the cross-reference table and specify the object number of the file body root object

containing the outline of the PDF document and the reference of page group object. The outline object refers to the bookmark tree of the PDF file, while the page group object contains the number of pages for the file, as well as the object number for each page object.

The parsing process for a PDF file[38] is as follows. First, the reader parses the trailer to get the root object in the PDF structure. And then it combines the information contained in the cross-reference table with the reference numbers found in each object to traverse and render each object in the PDF file body.

3.2 | PDF malware

PDF malware[39] exploits the vulnerabilities in the PDF reader to commit malicious behaviors. Most attacks that are executed with documentation are done in the form of scripting code to execute malicious code. There are three main types of malicious attacks.

JavaScript-based malware is a common way to exploit PDF document vulnerabilities. For example, some PDFs may use specific API calls in Adobe to implement heap spray attacks. In the file structure, this kind of malicious code is usually introduced through the object /javasript or /JS. Usually, in a malicious PDF, an attacker will often refer to an object containing such an object multiple time. In addition, the attack code can be contained in one object or scattered in multiple objects of the file to achieve the purpose of confusion. The typical vulnerabilities that JavaScript-based malware usually exploits include API-based overflow, use-after-free, malformed data, and so on.

Another way to exploit adobe reader vulnerability is ActionScript-based malware, triggering the vulnerability of flash component by embedding malicious shockwave flash file and ActionScript code. Usually, this kind of attack is implemented in combination with JavaScript, which triggers the vulnerability through the execution of ActionScript and exploits the vulnerability through the execution of JavaScript. The types of vulnerabilities exploited by ActionScript-based attacks include memory corruption, bytecode verification, corrupted file loading, and so on.

There is also a less common way to exploit vulnerabilities—file embedding. It often relies on external file types, such as .bmp, .tiff, .exe, and even other PDF files. This exploit is triggered when the reader attempts to parse such malicious files. However, file embedding is often used as a way to hide attacks rather than implement them.

3.3 | Hidost classifier

In this section, a previous work, Hidost classifier, will be introduced, which exists as the basis of this study. Hidost classifiers extract the structure of PDF files as classification features, which are parsed by the third-party tool poppler.[40] Although many features are not malicious in themselves, different combinations of each feature will show differences between benign and malicious files. In the paper, the authors use random forest and SVM as classification models. Experimental data show that their models have achieved excellent performance in 14 consecutive time period, with TPR close to 0.997 and false positive rate (FPR) less than 0.06. In the experiment, we also use the third-party tool poppler to extract structural features just like Hidost. We take the structural path that occurs more than 300 times in Contagio PDF file data set as the feature set in our experiment, including 438 structural path features in total.

4 | PROPOSED METHOD

4.1 | Uncertain samples

As we all known, SVM[41] is suitable for small sample classification. The reason is that SVM is only determined by the support vector, so the sample size has little influence on the final support vector. Before the formal experiment, we will verify the SVM classifier for small sample set classification. The main idea of SVM can be summarized into two points: First, it is aimed at the linear separable case. For the linear inseparable case, the samples that cannot be linearly separated by the low-dimensional input space will be transformed into high-dimensional feature space through the nonlinear mapping. In this way, it is possible to use the high-dimensional feature space to make linear analysis on the nonlinear features of those samples. Second, based on the theory of structural risk minimization, it constructs the optimal hyperplane in the feature space, which makes learner get the global optimization, and the expectation of the whole sample space satisfies a certain upper bound with a probability.

In this section, we will briefly introduce the working principle of SVM. As shown in the Figure 2, there are two types of data samples, namely "" and "×". Our goal is how to determine a suitable decision boundary to construct a classifier. For the unknown data samples, its classification can be predicted according to the position relationship between this point and the decision boundary. Generally, we prefer to choose an optimal decision boundary, maximizing the sum of the distances between the boundary and the nearest data points of the two types of data.

Assuming the data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_N, y_N)\}$. $\boldsymbol{x}_i \in \mathbb{R}^n, y_i \in \{+1, -1\}$, $i = 1, 2, ..., N$. for the hyperplane $W\boldsymbol{x} + b = 0$ that we need to solve, the geometric margin from the point $(\boldsymbol{x}_i, y_i)$ to the hyperplane is $\gamma_i = y_i\left(\frac{W}{\|W\|} \cdot \boldsymbol{x}_i + \frac{b}{\|w\|}\right)$. The minimum geometric margin from the training set to the hyperplane is $\gamma = \min_{i=1,2,\ ...,\ N}\gamma_i$, which is the so-called distance between the support vectors and the hyperplane, and our target can be abstracted into a formula $t = \max_{W,b}\gamma$. In the Figure 2, the red solid line is our target hyperplane, but before

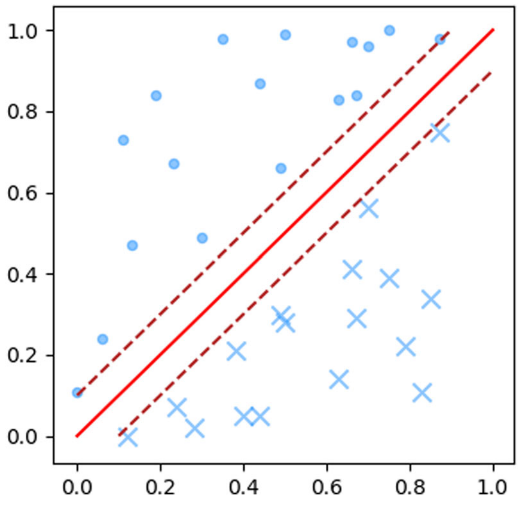


FIGURE 2 SVM decision boundary. It's an example for explaining the working principle of SVM. SVM, support vector machine [Color figure can be viewed at wileyonlinelibrary.com]

we find it, we first need to find the boundaries belonging to the "·" and "×" sets, respectively, as shown by the two dotted lines. We assume that they are $W\boldsymbol{x} + b = c$ and $W\boldsymbol{x} + b = -c$, respectively, and the distance between the two lines and the final classification line is equal. We only need to make W equal to W/c and b equal to b/c, then we can simplify the expressions of the two dotted lines to $W\boldsymbol{x} + b = 1$ and $W\boldsymbol{x} + b = -1$, while the expression of our target hyperplane remains unchanged. The distance between the two lines and the final classification line is $\frac{1}{\|W\|}$. At this point, our target expression can be changed from to $t = \max\limits_{W,b}\frac{1}{\|W\|}$.

In the adversary environment, there is a work[42] to achieve better defense effect by analyzing the decision boundary of the adversary samples. For our binary classification problem, we can introduce uncertain samples to optimize the decision boundary of the classifier. As shown in the left figure of Figure 3, suppose that the red solid line is the real decision boundary, and the decision boundary after training with training set is the black solid line. Obviously, there is a gap between the two. Now we consider such a case, if we take the samples of error classification of the classifier as uncertain samples, such as the red dot in the right figure, using the boundary of the left figure for classification will inevitably result in misclassification. If we use these uncertain samples to augment the training set, we can optimize the classifier to get the decision boundary of the black solid line in the right figure. Obviously, it will be closer to the real boundary than the previous one. To some extent, those uncertain samples contain rich information which is helpful for classification and may optimize decision boundary as support vectors. In the Section 5.3 and 5.4, experiments will be carried out to verify the effectiveness of uncertain samples and their advantages in retraining.

4.2 | Active learning

In the traditional supervised-learning based classification methods, the larger the training set size, the better the classification effect. However, in most real-world applications, the labeling of large-scale data set is quite difficult, generally requiring manually marking by experts in this

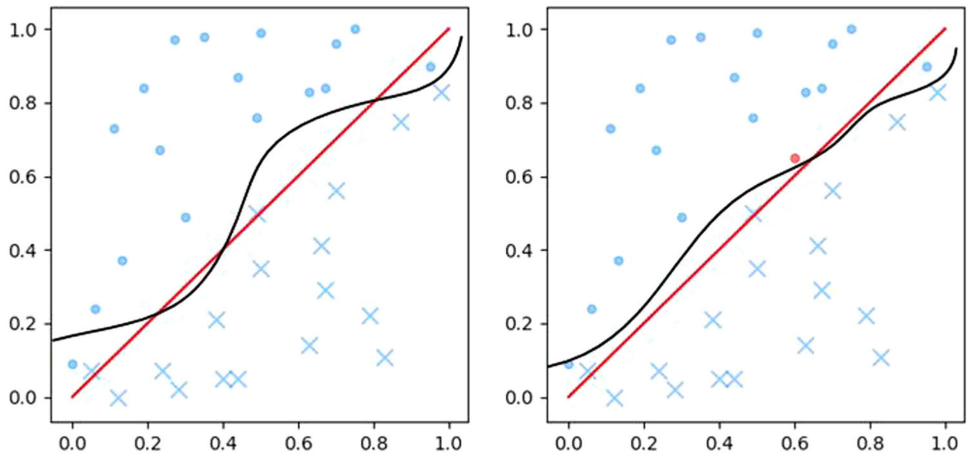


FIGURE 3 Decision boundary optimization. In the figures above, red solid lines are the real decision boundary, while black solid lines are the decision boundary after training. As shown in the two figures, adding the red dot may optimize the decision boundary of a classifier [Color figure can be viewed at wileyonlinelibrary.com]

professional field, which is a time-consuming and laborious task. Besides, the larger training data set will cause the longer time consumption of training detection model. The proposal of active learning method makes it possible to use smaller-scale training data set and obtain classifiers with better performance at the same time. A few uncertain samples with the richest information are picked and labeled, then the classifier is retrained with these samples to improve the accuracy gradually.[43,44]

Figure 4 is the design of our active learning method in this paper, which continuously detects new malwares through keeping updating the classifier. First, we transform unknown PDF files into a feature vector form by preprocessing and feature extraction process. Second, we take those feature vectors as the inputs of classifier and gets the predicted value for each unknown file. If the file can be classified as benign or malicious accurately, we do not need to do anything extra. If the judgment of classifier is uncertain, we need to set ground truth labels of these samples, deposit them into the training data set, and retrain classifier with them to improve the performance of our classifier.

The framework is divided into two phases: the training stage and the test/update stage.

- Training stage: We use the initial training set, including benign samples and malicious samples, to train the classifier. The initial performance of our model is evaluated by plenty of unknown PDF files tested for the first time.
- Detection/update stage: The classifier will provide a prediction for each unknown file, while the active learning method will obtain the information value contained in these files according to their uncertainty. The framework will pick a few uncertain sample files based on the information value and querying the ground truth label of the corresponding samples, after which these information-rich files will be used to augment the training set. The updated classifier can be achieved by retraining with the augmented training set, and will be used for the next epoch of evaluation.

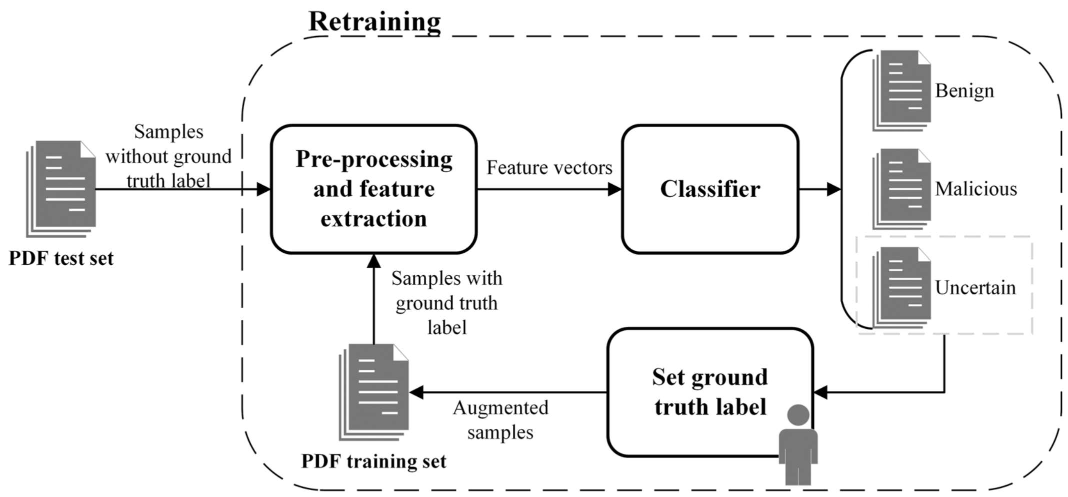


FIGURE 4 Active learning model. In the retraining period, we do not get the ground truth of all test samples, but only deal with those uncertain samples, which will greatly reduce the number of samples to be labeled

As described above, only picking the information-richest samples to retrain the classification model, active learning method can not only effectively improve the detection performance, but also limit the scale of training set and control the training time consumption to a certain extent.

For our method, we have to explain two problem. The first one is the standard of uncertainty. The uncertain files must include rich information that could be used to improve the detector performance. The means to pick uncertain files will be detailed later in this section. The second problem is how to obtain ground truth labels of the uncertain samples. All the experiment samples (including training and test data set) in this paper are labeled in advance, thus we can just query to obtain their ground truth labels. However, obtaining ground truth labels in the real world needs to consult security experts and manually label it by them.

4.2.1 | Classifier

The classifier used in this paper is improved based on Hidost model, whose preprocessing adopts a structure-based static analysis and a third-party tool poppler.[40] In addition, the features are mainly extracted from the structure information of PDF files, and the learning algorithm adopts the random forest algorithm.

4.2.2 | Mutual agreement analysis

In the implementation of PDFRate-new model,[34] mutual agreement analysis was proposed. In addition to benign and malicious label, mutual agreement analysis adds another label to the classifier output: uncertain (Table 1). The score of our classifier is split into four parts instead of two parts as usual. Most votes think that the sample was benign if the score is between 0% and 25%. Similarly, in the 75%–100% area, most votes agree that the sample was malicious. Differently, if the score is between 25% and 75%, meaning that there is a substantial divergence among the subclassifiers, and the output label will be uncertain.

To describe the definition of mutual agreement analysis more accurately, a metric is introduced to quantify the consistency of individual votes of the ensemble classifier:

$$A = |v - 0.5|*2 \tag{1}$$

where A is the mutual agreement rate of the ensemble classifier, and v is the voting score of the subclassifiers. Clearly, we can obtain Figure 5 according to the above metric. More specifically, when the mutual agreement rate is 50%, the result output is consistent with Table 1.

TABLE 1 Classifier outputs

Score	Output label		Evasion type
[0, 25]	Benign		Strong evasion
(25, 50)	Uncertain[a]	(Benign)	Weak evasion
[50, 75)		(Malicious)	
[75, 100]	Malicious		No evasion

[a]In addition to the general benign and malicious labels, an uncertain label is added to describe the test samples that the classifier classifies uncertainly.

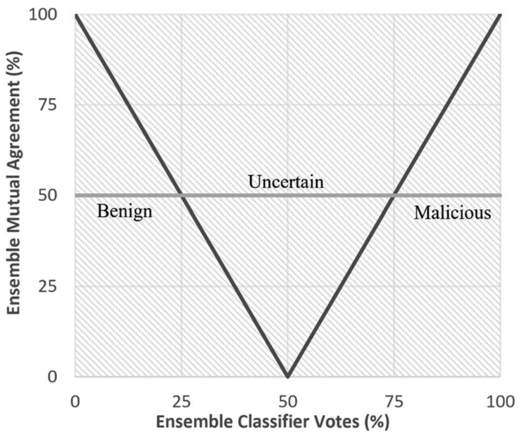


FIGURE 5 Mutual agreement analysis.[24] The coverage of uncertain samples is changed by controlling the ensemble mutual agreement. In the figure, the mutual agreement is set as 50%, so the coverage of uncertain samples is between 25% and 75%

Algorithm 1 Active Learning Algorithm

```
Input: Labeled training set L, unlabeled test set $U_i$(i $\in$ [1, 10]), ensemble mutual agreement $\tau \in [0, 1]$
Output: The prediction $p_{i,j} \in \{0, 1\}$ for the jth file in jth test set
1: retraining the model for 10 epochs with 10 test sets
2: for $i = 1$ to 10
3:   $\theta$ = train($L$)
4:   Initializing the augmented set of uncertain samples
5:   $A = \varnothing$
6:   for $j = 1$ to test set size do
7:     get the score $S_{i,j}$ of $U_{i,j}$, the jth file in the ith epoch
8:     $s_{i,j}$ = prediction ($\theta$, $U_{i,j}$)
9:     if $s_{i,j} > (1 + \tau)/2$ then
10:       $p_{i,j} = 1$
11:     else
12:       if $s_{i,j} < (1 - \tau)/2$ then
13:         $p_{i,j} = 0$
14:       else
15:         set the ground truth label of $U_{i,j}$
16:         add $U_{i,j}$ to set $A$
17:       end if
18:     end if
19:   end for
20:   $L = L \cup A$
21: end for
22: return
```

We left two problems in the design of our active learning method, one of which is the standard of uncertainty. Here, we apply mutual agreement analysis mentioned above to active learning to solve the selection problem of uncertain samples. We can choose a threshold according to Figure 5 to control the coverage of uncertain samples in our experiments. The smaller the threshold value, the fewer the uncertain samples selected. But the information contained in each sample will be more valuable, because when we set the threshold value to 0, the prediction of uncertain samples is equivalent to blind classification. On the other hand, the larger the threshold value, the more the uncertain samples we choose, and the more the training time is consumed. At the extreme, when the threshold is set to 100%, it is equivalent to putting all the test samples into the next epoch of retraining, rendering active learning meaningless.

Algorithm 1 describes the active learning algorithm. First of all, we need to set the mutual agreement rate τ and determine the initial training set $\boldsymbol{L}$. In each epoch of evaluation, we obtain the prediction result $S_{i,j}$ of every sample $U_{i,j}$. If $S_{i,j}$ is greater than $(1+\tau)/2$, the sample will be classified as malicious; if $S_{i,j}$ is less than $(1-\tau)/2$, the sample will be classified as benign; and if $S_{i,j}$ is between $(1+\tau)/2$ and $(1-\tau)/2$, the sample will be classified as uncertain, and it will be added to the uncertain sample set $\boldsymbol{A}$ to augment the training set $\boldsymbol{L}$.

5 | EXPERIMENT

5.1 | Data set

All PDF samples used in this paper are from the Contagio* data set, including 9000 benign samples and 10,982 malicious samples. The reason for choosing this data set is that it is available and contains a large number of labeled benign and malicious samples. A total of 9000 benign samples and 9000 malicious samples from this data source were used in the experiment. For the obtained samples, we use popper to analyze the structure path, and obtain the feature vector of each file to facilitate the input of the classifier. We take the structural path that occurs more than 300 times in PDF file data set as the feature set in our experiment, including 438 structural path features in total. The processed data set form is $\mathrm{D}=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$. $x_i\in\mathbb{R}^n, y_i\in\{+1,-1\}, i=1,2,\ldots,N,$ where $\boldsymbol{x}_i$ represents the feature vector of the ith PDF file, $\boldsymbol{y}_i$ represents the ground truth of the ith PDF file, and N is the size of this data set.

5.2 | Small sample classification

At the beginning of the experiment, we tested the performance of SVM classifier for small sample set. The number of training samples increased by 20 from 20 to 200, and the number of test samples remained at 1000. Performance measures included accuracy rate, TPR, FPR, and AUROC. It can be seen from the experimental results (Figure 6) that, although the performance of SVM classifier improves slightly with the increase of the number of samples, the overall trend is very gentle, which shows that SVM classifier is indeed suitable for small sample classification.

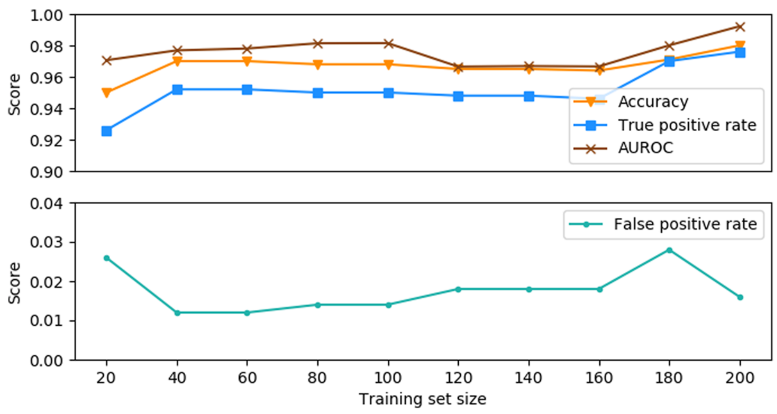


FIGURE 6 Support vector machine for small sample classification [Color figure can be viewed at wileyonlinelibrary.com]

5.3 | Uncertain sample classification

5.3.1 | Data set processing

To verify the effectiveness of uncertain samples mentioned above for classification, we used Contagio data set to do some experiments. Before the experiment, we used the VirusTotal** platform to relabel these samples. Taking VirusTotal as an ensemble classifier composed of multiple classifiers, we used API interface to write Python script to upload samples in data set for sample labeling. The format of label is $\{y, t, p, \text{path}\}$, where y is the classification result of VirusTotal, t is the number of subclassifiers that can be used to classify the current PDF, p is the number of subclassifiers that classify the file as positive, and the path is the local path of PDF file.

After uploading VirusTotal, we found that all uncertain samples are malicious files, in other words, all benign files could be correctly classified on all classifiers of the platform. By changing the threshold p, we could get the number of uncertain samples under the threshold, that is to say, the number of samples that only no more than p classifiers could detect its maliciousness. In the Figure 7, we show the trend of uncertainty samples with thresholds from 30 to 45. The data shows that with the increase of threshold, the number of uncertain samples is increasing rapidly, which indicates that for most samples, most of the subclassifiers can still classify correctly, while the proportion of uncertain samples is relatively small. Thus, if we use these samples to argument the training set, the sample size will not increase dramatically.

5.3.2 | Evaluation

For the processed data set, we trained SVM model for comparative analysis. Three groups of experiments were conducted with threshold p of 35, 40, and 45, respectively. In each group of comparative experiments, 100 uncertain malicious samples were mixed with 100 random benign samples to train a classifier, and the remaining uncertain samples were used as the test set; 100 random malicious samples were mixed with 100 random benign samples to train another classifier, and the test set was the same as the former. The experimental results are

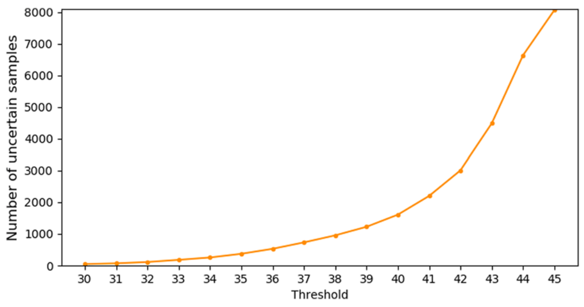


FIGURE 7 Size of uncertain samples for different threshold [Color figure can be viewed at wileyonlinelibrary.com]

shown in the Table 2. With TP as the number of malicious samples correctly classified as malicious, TPR can reflect the ability of detector to correctly detect malicious codes. The data shows that the accuracy of the classifier trained with uncertain samples is better than that trained with random samples. From the results, we can draw a conclusion that the uncertain samples do have a positive effect on the correct classification of the classifier.

5.4 | Active learning

5.4.1 | Data set processing

After the verification of Section 5.3, the uncertain samples can contribute to malware detection, and the number of samples is very small, which is very suitable for the augmented samples of training set in active learning. In this section, we will experimentally boost training with uncertain samples. For data set processing, 354 samples were randomly selected as the original training set, including 176 malicious samples and 178 benign samples. Each epoch, about 1400 samples were used as the test set (Table 3).

We apply active learning to the retraining process and only select a small number of test set samples as the augmentation of the previous training set, instead of adding all the evaluated test set samples to the next epoch. When it comes to selecting the test set samples, mutual agreement analysis was used. We set the threshold value to 50%, and obtain the coverage of uncertain

TABLE 2 Comparison between uncertain and random sample

		Measures	
Threshold	Training set	Accuracy	TPR
35	Random samples	0.956	0.956
	Uncertain samples	0.975	0.975
40	Random samples	0.934	0.934
	Uncertain samples	0.965	0.965
45	Random samples	0.885	0.885
	Uncertain samples	0.908	0.908

TABLE 3 Experimental data sets

	Size	Benign samples	Malicious samples
Original training set	354	176	178
Test set (×10)	About 1400	About 700	About 700

samples (see Figure 5), that is, if the evaluation scores of a sample is between 0.25 and 0.75, it will be classified as uncertain. However, if the performance of the model is poor, the size of uncertain samples will become large, even close to the size of the whole test set. Using large-scale samples to augment the training set will make significance of active learning method lost. Therefore, the number of samples augmented to the training set in each epoch should be controlled, and we set it to 20 in our experiment to limit the time consumption of model retraining.

5.4.2 | Evaluation

When we apply active learning method to Hidost model, set the mutual agreement rate to 50%, and set the maximum number of uncertain samples selected in each epoch to 20, we will get the result shown in Figure 8. In the picture, **_all* means to use the whole test set of the previous epoch to augment the training set, while **_uncertain* means that we only use a part of information-rich uncertain samples to augment the training set. There was no significant difference in the four measures before and after using active learning method. This shows that

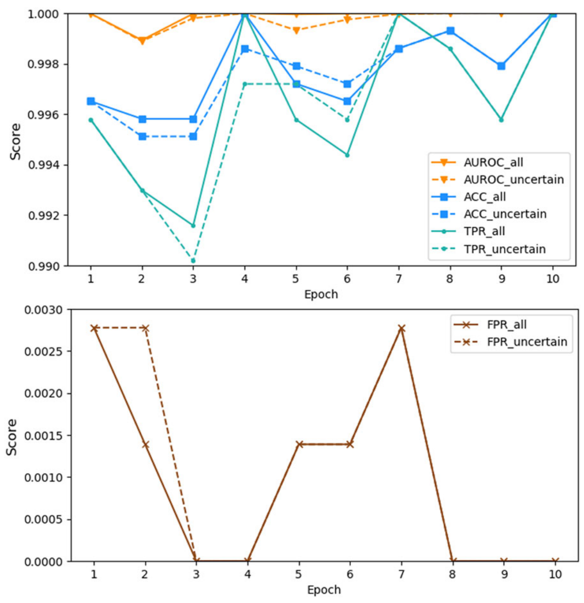


FIGURE 8 The performance of Hidost models with/without active learning. The *x*-axis represents the epochs of retraining, which are 10 discrete numbers, while the *y*-axis represents the measure scores. The broken lines in the figure can reflect the performance improvement trend of the detectors during retraining [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 4 Comparison of Hidost models with/without active learning

		Original Hidost model		Hidost model based on active learning		
Epoch	Test set	Training set	Time (s)	Training set	Time (s)	Training set reduction
1	1433	354	1.06924414635	354	1.16369819641	0
2	1433	1787	1.27057909966	362	1.22491407394	1425
3	1433	3220	1.15115809441	373	1.30330085754	2847
4	1433	4653	1.28022480011	381	1.20180821419	4272
5	1433	6086	1.39241123199	390	1.20856690407	5696
6	1433	7519	1.70291876793	395	1.33792090416	7124
7	1433	8952	1.63864588737	399	1.20936203003	8553
8	1433	10,385	1.52327895164	404	1.19747900963	9981
9	1433	11,818	1.53993701935	408	1.2542309761	11,410
10	1421	13,251	1.7291829586	411	1.19509601593	12,840

a small number of uncertain samples basically contain all the differential information useful for classification in each epoch.

Table 4 shows the comparison of the size of training set and time consumption per epoch between the original Hidost model and the Hidost model based on active learning method. On the one hand, the size of training sets varies greatly. The original Hidost model uses 13,251 samples in the 10th epoch, while the Hidost model based on active learning method only uses 411 samples, which is 1/30 of the former. Compared with the first epoch, the Hidost model based on active learning method only increases 57 samples in the 10th epoch, less than 1/6 of the initial training set. On the other hand, the time consumption of model based on active learning method is also significantly shortened. We can compare the time consumption of detectors more intuitively in Figure 9.

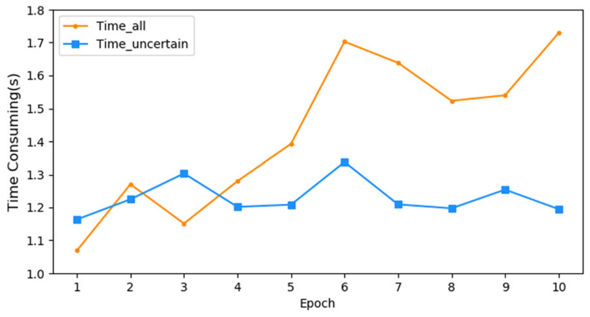


FIGURE 9 Advantages of uncertain samples for active learning. The meaning of *x*-axis is the same as that of Figure 8, while *y*-axis represents the time consumption of the detectors during each epoch of retraining. It can be seen from the broken lines in the figure that the time consumption of the model training with the whole test samples is on the rise, while the time consumption of the model only using uncertain samples keeps stable at a lower value [Color figure can be viewed at wileyonlinelibrary.com]

The use of uncertain samples for active learning greatly reduces the training set size used, and greatly shortens the time consumption in each retraining epoch. At the same time, the performance of the Hidost model based on active learning method is basically the same as that of the original Hidost model. This is of great significance to the malware detection model in the real world. Malware always tries new ways to avoid detection, and we must periodically retrain our model to adapt to the new malicious features. Generally speaking, the larger the training set, the better the performance. Nonetheless, it's a time-consuming work to add all the previous test sets to the training set, and will lead to a substantial increase in training time, which has no practical significance. The use of active learning method makes it possible to use fewer training samples and obtain better classification performance. We only select a few uncertain samples containing almost all the useful information of the test set for data augmentation in each retraining epoch. Therefore, it not only does not cause large-scale training set and lose practical significance, but also can improve the performance of detector to maintain the value of retraining.

6 | CONCLUSION AND FUTURE WORK

In this paper, the significance of uncertain samples is verified, and an active learning detection model with those uncertain samples is implemented. In each evaluation epoch, a small number of information-rich test set samples, so-called uncertain samples, are picked to augment the training set, and the performance of classifier is improved gradually. We can greatly reduce the training time consumption compared with the traditional retraining models. This paper implements an active learning model based on mutual agreement analysis. We apply active learning method to Hidost model, and use mutual agreement analysis as the selection criterion of uncertain samples, making it possible to use smaller training set and obtain better classification performance.

Since our classifier is implemented on the basis of Hidost, which is a detection model based on static analysis, our active learning method may not be suitable for detecting PDF malware that need dynamic analysis. To remove this limitation, we still need to add dynamic analysis technology to our method to apply to a wider range of PDF malware. PDF malware detection method based on traditional machine learning has been quite mature. With the development of deep learning,[45] there is no doubt that deep learning algorithm will be widely used in malware detection in the future. There is still a lot of work to be done in the future research with the emergence and improvement of various new technologies. Attempting to utilize different deep learning models and explore deep learning algorithms suitable for PDF malware detection will be one of our future research tasks. In addition, we will also consider applying the active learning method to the deep learning model, using mutual agreement analysis to pick the uncertain samples near the decision boundary according to the detection results. And a small number of uncertain samples are used to augment the training set to improve the performance of the deep learning model and control the time consumption to a certain extent.

ACKNOWLEDGMENT

This study is supported by Zhejiang Lab (No. 2020LE0AB02) and National Natural Science Foundation of China (No. 62072037 and U1936218).

CONFLICT OF INTERESTS

The authors declare that there are no conflict of interests.

ENDNOTES

*http://contagiodump.blogspot.it

**https://www.virustotal.com

REFERENCES

1. Wang W, Wang X, Feng DW, Liu JQ, Han Z, Zhang XL. Exploring permission-induced risk in android applications for malicious application detection. *IEEE Trans Inform Forensics Security*. 2017;9(11): 1869-1882.
2. Zhang QK, Wang XM, Yuan JL, Liu L, Wang RF, Huang H, Li YZ. A hierarchical group key agreement protocol using orientable attributes for cloud computing. *Inform Sci*. 2019;480:55-69.
3. Wang W, Shang YY, He YZ, Li YD, Liu JQ. BotMark: automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. *Inform Sci*. 2020;511:284-296.
4. Meng WZ, Li WJ, Kwok LF. EFM: enhancing the performance of signature-based network intrusion detection systems using enhanced filter mechanism. *Comput Security*. 2014;43:189-204.
5. Yuan YL, Huo LW, Wang ZX, Hogrefe D. Secure APIT localization scheme against sybil attacks in distributed wireless sensor networks. *IEEE Access*. 2018;6:27629-27636.
6. Li YZ, Hu JJ, Wu ZZ, Liu C, Peng FF, Zhang Y. Research on QoS service composition based on coevolutionary genetic algorithm. *Soft Comput*. 2018;22(23):7865-7874.
7. Guan ZT, Liu XY, Wu LF, Wu J, Xu RZ, Zhang JH, Li YZ. Cross-lingual multi-keyword rank search with semantic extension over encrypted data. *Inform Sci*. 2020;514:523-540.
8. Zhang QK, Gan Y, Liu L, Wang XM, Luo XY, Li YZ. An authenticated asymmetric group key agreement based on attribute encryption. *J Network Comput Appl*. 2018;123:1-10.
9. Namanya AP, Cullen AJ, Awan IU, Diss JP. The world of malware: An overview. In: *2018 IEEE 6th International Conference on Future Internet of Things and Cloud*(*FiCloud*); 2018:420-427.
10. Chakkaravarthy SS, Sangeetha D, Vaidehi V. A survey on malware analysis and mitigation techniques. *Comput Sci Rev*. 2019;32:1-23.
11. Biggio B, Corona I, Maiorca D, Nelson B, Laskov P, Giacinto G, Roli F. Evasion attacks against machine learning at test time. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*; 2013:387-402.
12. Xu WL, Qi YJ, Evans D. Automatically evading classifiers. In: *Proceedings of the 2016 Network and Distributed Systems Symposium*; 2016.
13. Li WJ, Meng WZ, Tan ZY, Xiang Y. Design of multi-view based email classification for IoT systems via semi-supervised learning. *J Network Comput Appl*. 2019;128:56-63.
14. Maiorca D, Biggio B, Giacinto G. Towards adversarial malware detection: Lessons learned from PDF-based attacks. *ACM Comput Surveys*. 2019;52(4):1-36.
15. Chervenak AL, Vellanki V, Kurmas Z. Protecting file systems: a survey of backup techniques. In: *Joint NASA and IEEE Mass Storage Conference*; 1998.
16. Ofek Y, Cakeljic Z, Gagne M. Apparatus and method for differential backup and restoration of data in a computer storage system. 2002.
17. Shafiq MZ, Khayam SA, Farooq M. Embedded malware detection using markov n-grams. In: *International Conference on Detection of Intrusions and Malware and Vulnerability Assessment*; 2008:88-107.
18. Tabish SM, Shafiq MZ, Farooq M. Malware detection using statistical analysis of byte-level file content. In: *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*; 2009:23-31.
19. Laskov P, Šrndić N. Static detection of malicious JavaScript-bearing PDF documents. In: *Proceedings of the 27th Annual Computer Security Applications Conference*; 2011:373-382.
20. Maiorca D, Giacinto G, Corona I. A pattern recognition system for malicious pdf files detection. In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*; 2012:510-524.
21. Smutz C, Stavrou A. Malicious PDF detection using metadata and structural features. In: *Proceedings of the 28th Annual Computer Security Applications Conference*; 2012:239-248.
22. Šrndić N, Laskov P. Detection of malicious PDF files based on hierarchical document structure. In: *The 20th Annual Network & Distributed System Security Symposium*; 2013:24-27.

23. Maiorca D, Ariu D, Corona I, Giacinto G. A structural and content-based approach for a precise and robust detection of malicious PDF files. In: *2015 International Conference on Information Systems Security and Privacy (ICISSP)*; 2015:27-36.
24. Smutz C, Stavrou A. When a tree falls: using diversity in ensemble classifiers to identify evasion in malware detectors. In: *23rd Annual Network and Distributed System Security Symposium*; 2016.
25. Šrndić N, Laskov P. Hidost: a static machine-learning-based detector of malicious files. *EURASIP J Inform Security*. 2016;2016(1):22.
26. Cova M, Kruegel C, Vigna G. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In: *Proceedings of the 19th International Conference on World Wide Web*; 2010:281-290.
27. Snow KZ, Krishnan S, Monrose F, Provos N. SHELLOS: enabling fast detection and forensic analysis of code injection attacks. In: *USENIX Security Symposium*; 2011:183-200.
28. Tzermias Z, Sykiotakis G, Polychronakis M, Markatos E. Combining static and dynamic analysis for the detection of malicious documents. In: *Proceedings of the Fourth European Workshop on System Security*; 2011:1-6.
29. Corona I, Maiorca D, Ariu D, Giacinto G. Lux0r: detection of malicious pdf-embedded javascript code through discriminant analysis of API references. *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*; 2014:47-57.
30. Chen YZ, Wang SQ, She DD, Jana S. On training robust PDF malware classifiers. arXiv; 2019.
31. Grosse K, Papernot N, Manoharan P, Backes M, McDaniel P. Adversarial perturbations against deep neural networks for malware classification. arXiv; 2016.
32. Íncer RÍ, Theodorides M, Afroz S, Wagner D. Adversarially robust malware detection using monotonic classification. In: *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics*; 2018: 54-63.
33. Tong L, Li B, Zhang N, Hajaj C, Xiao CW, Vorobeychik Y. Improving robustness of ML classifiers against realizable evasion attacks using conserved features. In: *Proceedings of the 28th USENIX Security Symposium*; 2019:285-302.
34. Dey S, Kumar A, Sawarkar M, Singh PK, Nandi S. EvadePDF: towards evading machine learning based PDF malware classifiers. In: *International Conference on Security & Privacy*; 2019:140-150.
35. Kang AR, Jeong YS, Kim SL, Woo JY. Malicious PDF detection model against adversarial attack built from benign PDF containing javaScript. *Appl Sci*. 2019;9(22):4764.
36. Adobe. PDF Reference. Adobe Portable Document Format Version 1.7; 2006.
37. Adobe. Adobe Supplement to ISO 32000; 2008.
38. Gao LC, Tang Z, Lin XF, Liu Y, Qiu RH, Wang YT. Structure extraction from PDF-based book documents. In: *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*; 2011:11-20.
39. Stevens D. Malicious PDF documents explained. *IEEE Security Privacy*. 2011;9(1):80-82.
40. FreeDesktop.org. Poppler; 2018.
41. Vapnik VN, Lerner AY. Recognition of patterns with help of generalized portraits. *Avtomat. i Telemekh*. 1963;24(6):774-780.
42. He W, Li B, Song D. Decision boundary analysis of adversarial examples. In: *ICLR*; 2018.
43. Nissim N, Cohen A, Elovici Y. ALDOCX: detection of unknown malicious microsoft office documents using designated active learning methods based on new structural feature extraction methodology. *IEEE Trans Inform Forensics Security*. 2016;12(3):631-646.
44. Pimentel T, Monteiro M, Veloso A, Ziviani N. Deep active learning for anomaly detection. In: *Proceedings of the International Joint Conference on Neural Networks*; 2020.
45. Schmidhuber J. Deep learning in neural networks: an overview. *Neural Networks*. 2015;61:85-117.

How to cite this article: Li Y, Wang X, Shi Z, Zhang R, Xue J, Wang Z. Boosting training for PDF malware classifier via active learning. *Int J Intell Syst*. 2022;37:2803-2821. https://doi.org/10.1002/int.22451