Received April 2, 2019, accepted April 23, 2019, date of publication May 1, 2019, date of current version May 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2913439

Adversarial Examples for CNN-Based Malware Detectors

BINGCAI CHEN[1, 2], (Member, IEEE), ZHONGRU REN[1], CHAO YU[1], IFTIKHAR HUSSAIN[3], AND JINTAO LIU[1]
[1]School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China
[2]School of Computer Science and Technology, Xinjiang Normal University, Urumqi 830054, China
[3]School of Computer Science and Technology, University of Science and Technology of China, Hefei 230000, China

Corresponding author: Zhongru Ren (renzhongru@mail.dlut.edu.cn) and Chao Yu (cy496@dlut.edu.cn)

This work was supported by the Natural Science Foundation of China under Grant 61771089 and Grant 61502072.

ABSTRACT The convolutional neural network (CNN)-based models have achieved tremendous breakthroughs in many end-to-end applications, such as image identification, text classification, and speech recognition. By replicating these successes to the field of malware detection, several CNN-based malware detectors have achieved encouraging performance without significant feature engineering effort in recent years. Unfortunately, by analyzing their robustness using gradient-based algorithms, several studies have shown that some of these malware detectors are vulnerable to the evasion attacks (also known as adversarial examples). However, the existing attack methods can only achieve quite low attack success rates. In this paper, we propose two novel white-box methods and one novel black-box method to attack a recently proposed malware detector. By incorporating the gradient-based algorithm, one of our white-box methods can achieve a success rate of over 99%. Without prior knowledge of the exact structure and internal parameters of the detector, the proposed black-box method can also achieve a success rate of over 70%. In addition, we consider adversarial training as a defensive mechanism in order to resist evasion attacks. While proving the effectiveness of adversarial training, we also analyze its security risk, that is, a large number of adversarial examples can poison the training dataset of the detector. Therefore, we propose a pre-detection mechanism to reject adversarial examples. The experiments show that this mechanism can effectively improve the safety and efficiency of malware detection.

INDEX TERMS Adversarial examples, CNN, malware detection.

I. INTRODUCTION

The detection of malicious software (malware) has been playing an increasingly important role in cyber security. In recent years, the number of malware and its variants has shown a trend of explosive growth [1]. For example, the notorious WannaCry ransom virus, which caused huge losses worldwide in May 2017, has generated a number of variants that have been active until now.

Anti-virus products provide some protection against malware. Traditional malware detection methods, such as signature-based methods and heuristic-based methods, have been widely used by these anti-virus products untill now [2]. Typically, signature-based methods are primarily used to identify the known malware, so these methods can be easily bypassed by malware writers using anti-anti-virus techniques (such as encryption, packing, obfuscation, and polymorphism) [3]. Heuristic-based methods can detect partially unknown malware, but these methods rely on the rules/patterns constructed by domain experts, which are often error-prone and time consuming. Therefore, traditional malware detection methods cannot adapt to rapidly growing malware and a large number of variants. Machine learning algorithms provide an opportunity to solve this issue by generalizing known malware to new ones in an efficient way [4]. In the last decades, a great number of machine learning based methods have been proposed in the literature [5]–[8]. However, most of these methods require substantial effort and domain expertise in creating and identifying important features, calling for new end-to-end malware detection methods without significant manual feature engineering.

As a widely used machine learning algorithm, Convolutional Neural Networks (CNN) excels in many end-to-end applications, such as image identification [9],

The associate editor coordinating the review of this manuscript and approving it for publication was Bora Onat.

text classification [10], and speech recognition [11], because of its excellent feature learning capabilities. By replicating these successes to the field of malware detection, in recent years, several CNN-based malware models have been proposed, achieving high detection accuracy without domain knowledge [12], [13].

However, it has been shown that most machine learning based models, including state-of-the-art CNN based models, are vulnerable to evasion attacks, also known as adversarial examples (AEs) [14]. AEs are a class of inputs that are derived from legitimate inputs by adding carefully chosen perturbations such that the model can be induced to output erroneous predictions. The majority of existing studies mainly focus on image classification models [15]–[17], where attackers add subtle modifications that are beyond human recognition to the input image pixels so as to deceive the victim models. Similarly, in the domain of malware detection, an AE is a carefully modified binary file that is derived from existing malware but misclassified as benign to avoid detection. Although the use of CNN to detect malware is well known, the emergence of AEs makes the existing CNN-based malware detectors no longer robust. Thus, it is highly urgent to analyze the vulnerability of existing malware detectors and propose effective defensive mechanisms to improve the robustness of these detectors.

In this paper, we extensively investigate the vulnerability of the CNN-based malware detectors, specifically a recently proposed detector Malconv [12]. Most attack methods in previous literature [18]–[20] against Malconv appended perturbations to the end of malicious files for evading detection. These perturbations were all initialized by random noises and iteratively modified by gradient-based algorithms. However, the ignorance of the importance of selecting initial perturbations can lead to low attack success rates using these methods. To address this issue, we propose two novel white-box attack methods that use saliency vectors to select perturbations from benign files. For an explicit illustration, we use a saliency vector to represent the feature (benign or malicious) of each region in an input file. These saliency vectors can be generated by using the Gradient-weighted Class Activation Mapping (Grad-CAM) method [21]. In addition, we also present a black-box attack method for situations when attackers cannot know the exact structure and internal parameters of the victim model. By implementing these attack methods on a dataset of Windows Portable Executable (PE) files, we demonstrate the vulnerability of the CNN-based malware detector to evasion attacks. At last, we also consider two defensive mechanisms, including adversarial training and rejecting AEs, in order to resist evasion attacks.

The major contribution of this paper is as follows:

1) We propose two novel white-box attack methods and a novel black-box attack method against CNN-based malware detectors. When attacking a recently proposed malware detector Malconv, one of our white-box methods, by incorporating the Fast Gradient Sign Method (FGSM) [15], can achieve an attack success rate of over 99%, and our black-box method can also achieve a success rate of over 70% without knowing the detector details.

2) We consider adversarial training [14] as a defensive mechanism in order to resist evasion attacks. While proving the effectiveness of adversarial training, we also analyze its security risk that the training dataset of the detector can be poisoned by a large number of AEs. Therefore, we propose a pre-detection mechanism, which can effectively reject the AEs, to protect the malware detector.

The rest of this paper is organized as follows. Section II presents background and related work. Section III describes the methods of evasion attacks. Experiments for attacks are given in Section IV. Defenses against these attacks are given in Section V. Finally, we conclude the paper in Section VI.

II. RELATED WORK

Malware detection is gradually shifting from traditional rule-based approaches to machine learning based methods [22]. In this section, we first give a short introduction to machine learning based malware detection methods (II-A), and then mainly focus on CNN-based malware detectors (II-B). Subsequently, we discuss the methods of evasion attacks in the field of image classification and malware detection (II-C). Finally, we briefly review several defensive mechanisms that have been proposed so far (II-D).

A. MACHINE LEARNING BASED MALWARE DETECTION

Most machine learning based malware detection methods analyze the input binary files by man-made features. These features, which are extracted by security experts according to the specific format of the software, largely determine the quality of detections. The software formats in different operating systems are quite different. For example, the PE format [23] is the standard format for Windows operating system software. According to the way of feature extraction, these machine learning based methods can be mainly divided into dynamic methods and static methods. Static methods, including n-gram based, PE-header based, and multi-features based methods *etc.* [5], [6], [24], are simple and efficient, which extract features directly from the raw bytes of static binary files. Dynamic methods, such as system-call-based and behavior-based methods [7], [8], have higher detection accuracy but require more complex feature engineering by running binary files. In terms of classification algorithms, Support Vector Machine (SVM), Random Forest, fully-connected Deep Neural Networks (DNN), Recurrent Neural Network (RNN), and CNN are widely used in recent years [25]–[29]. Since most machine learning based methods require a lot of effort and domain expertise in manual feature engineering, end-to-end malware detection methods, without domain knowledge, have attracted much more attention in recent years.

B. CNN-BASED MALWARE DETECTORS

Malware examples are generated by real attackers, who usually use encryption and obfuscation techniques for malware to combat static parsing and dynamic analyzing. In this case, manual feature engineering requires significant effort of experts with sufficient prior knowledge. As a typical end-to-end algorithm, CNN has excellent feature learning capabilities in many real-life applications, including malware detection. Recent studies by Raff *et al.* [12] and Krčal *et al.* [13] have reported effective end-to-end malware detectors using the CNN algorithm. These detectors analyze the raw bytes directly and discriminate malicious and benign files by automatically extracting features. The detector named Malconv proposed by Raff *et al.* [12], using a shallow CNN-based model, could obtain a detection accuracy of 95.9% when trained on a dataset with two million PE files. Based on this, Krčal *et al.* then proposed a deeper model which could achieve better performance than Malconv when trained on a larger dataset. However, recent studies [18]–[20] have demonstrated that these detectors are vulnerable to evasion attacks, which calls for a more in-depth analysis to the robustness of these detectors and motivates our work in this paper.

C. EVASION ATTACKS

Evasion attacks use AEs to trick the model classifier into outputting incorrect classification results. An AE $\tilde{\boldsymbol{x}}$ is generated by adding iteratively modified perturbations $\boldsymbol{\eta}$ to a correctly classified original file $\boldsymbol{x}$. $\tilde{\boldsymbol{x}}$ is called an effective AE when the model outputs the incorrect classification result for $\tilde{\boldsymbol{x}}$. At present, the majority of existing evasion attack methods, such as L-BFGS [14], Deepfool [30], FGSM [15], and JSMA [16], focus on the field of image classification. The L-BFGS and Deepfool reduce the AEs generation problem to the process of searching for the optimal perturbation $\boldsymbol{\eta}$. Their complex optimization processes result in slow speed and high computational costs. The advantage of FGSM and JSMA is that they are easier to generate AEs when the model is very sensitive to input changes. The difference is that FGSM can generate AEs quickly when using larger perturbations, but JSMA is much slower because it requires adversarial saliency maps to be calculated at each iteration. Therefore, FGSM is more widely used when generating AEs with large perturbations. All these previous attack methods are white-box methods that require attackers to understand the exact structure and parameters of the victim model. Without knowing the model details, Papernot *et al.* later used the FGSM and JSMA to implement black-box attacks against the DNN-based models by training substitute models [17], [31].

Recently, generating malware AEs has also become a hot topic in the security field. However, different from the image AEs, there are more stringent restrictions on the generation of PE format malware AEs. A PE binary file is an executable file that is characterized by an organizational structure, and any changes to its bytes can cause functionality to be corrupted or even make it impossible to execute. So attacks that modify the raw bytes of PE files must maintain the full syntax structure and functionality of the original files. Anderson *et al.* [32] proposed a black-box attack method, which uses a Reinforcement Learning (RL) algorithm to generate AEs against a Gradient Boosted Decision Tree (GBDT) based malware detector. Grosse *et al.* [33] proposed a JSMA-based method against DNN-based malware detection models. Hu and Tan [34] generated AEs against DNN-based models by a black-box method based on Generative Adversarial Networks (GAN). Attack methods against the CNN-based model Malconv have also been proposed by Kolosnjaji *et al.* [18], Kreuk *et al.* [19], and Suciu *et al.* [20] more recently. Directly modifying the internal bytes of PE files during the process of generating AEs requires sufficient domain expertise, so most these methods simplify this process by appending perturbations to the end of PE files. Kolosnjaji *et al.* [18] proved the vulnerability of Malconv for the first time, but their method requires high computational costs and achieves low attack success rate. Kreuk *et al.* [19] proposed an improved loss function to increase the attack success rate of the FGSM-based method. Suciu *et al.* [20] compared several attack methods on a large-scale dataset and a small-scale dataset. Their experiments show that the FGSM-based attack method hardly works on the small-scale dataset but achieves a higher success rate on the large-scale dataset. Common to these methods is that the perturbations added to AEs are all initialized by random noises and then iteratively modified by gradient-based algorithms. However, the ignorance of the importance of selecting initial perturbations can lead to low success rates of these methods. In contrast, we use saliency vectors to select perturbations in benign files in order to increase the attack success rate.

D. DEFENSIVE MECHANISMS

Defenses against evasion attacks mainly focus on the field of image classification. The extending defensive distillation method [35] proposed by Papernot *et al.*, which can identify abnormal inputs with large uncertainty, protects the original model by training a distillation model of the same scale. Szegedy *et al.* [14] proposed to actively construct virtual AEs for adversarial training. Although adversarial training is effective, it cannot resist all AEs and still has limitations. Hosseini *et al.* [36] proposed to add 'NULL' to the output labels for training and identify AEs by classifying them as 'NULL'. Lu *et al.* [37] proposed a framework named SafetyNet that contains detectors and classifiers. Their framework uses the detector to discriminate whether the input is an AE, such that AEs will be rejected before entering into the classifier. The defensive mechanisms against malware AEs are mainly adversarial training and defensive distillation. Al-Dujaili *et al.* [38] proposed an online adversarial training framework named SLEIPNIR that treats adversarial training as a saddle point problem. Grosse *et al.* [33] evaluated two potential defensive mechanisms, including adversarial training and defensive distillation, for system-call-based malware detectors. They showed that the effect of the defensive distillation method is not as obvious as the adversarial training. Since few studies focus on the defenses for end-to-end malware detectors, in this paper, we study two defensive mechanisms in this situation and analyze their pros and cons in detail.

III. METHOD

We first describe the detail architecture of Malconv and introduce how to generate saliency vectors for input files by using the Grad-CAM method. We then introduce two white-box attack methods by using these saliency vectors. One is called Benign Features Append (BFA) and the other, called Enhanced-BFA, is an enhanced version of the first one by incorporating the FGSM algorithm. Finally, we briefly describe the Random attack method and introduce a more efficient black-box method by summarizing the successful experiences of Random attacks.

A. ARCHITECTURE

First, we describe the specific network architecture of the Malconv model, as shown in Fig.1 [12]. An input $\boldsymbol{x}$ (maximum length L) is a sequence of discrete bytes, $\boldsymbol{x} = (x_1, x_2, \ldots, x_i)$, $i < L$. Data preprocessing, ensuring that the input vectors provided to the network have a fixed size regardless of the size of input files, is first performed to generate a fixed length sequence by padding 0. Then the preprocessed sequence is mapped to a fixed-size matrix $\boldsymbol{e}$ by an embedding layer $\mathbf{W}$, where $\boldsymbol{e} = (e_1, e_2, \ldots, e_i)$, $i = L$. The embedding layer, which allows the meaning of input bytes to depend on the context rather than the byte values, is essentially a lookup table that maps each input byte x_i to a D-dimensional vector $e_i \in \mathbb{R}^D$. Then the matrix $\boldsymbol{e}$ is fed into two 1-dimensional convolution layers whose activation functions are sigmoid and relu respectively. The corresponding outputs (feature maps) of the two convolutional layers are multiplied element-wise. This mechanism, which is called gating, was proposed by Dauphin *et al.* [39] when dealing with gradient vanishing problems in language models. Afterwards, the result of gating is passed to a global max-pooling layer, such that all feature maps are reduced to a fixed-size vector before entering into a fully-connected layer. Finally, the classification result will be output by a softmax layer.

B. SALIENCY VECTORS

Next, we describe saliency vectors in detail and introduce how to generate them by using the Grad-CAM method. A saliency vector, which contains features of a series of data blocks in an input binary file, can roughly show the benign and malicious regions of the file. The position of the elements in the vector corresponds with the regions of the original file and the value of each eliment indicates the significance of features in the corresponding region, namely, a larger value indicates a more significant feature. In the image classification task, Zhou *et al.* [40] proposed a method by using the Class Activation Mapping (CAM) to produce a 'visual interpretation' for the decision of a CNN-based model. Raff *et al.* [12] then improved this approach to visualize the sections where the malicious features are located. However, these two methods, which all require modifying

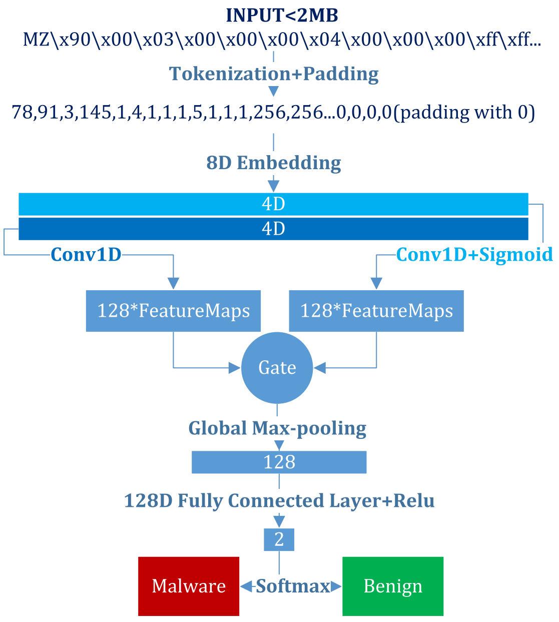
INPUT<2MB
MZ\x90\x00\x03\x00\x00\x00\x04\x00\x00\x00\xff\xff...
Tokenization+Padding
78,91,3,145,1,4,1,1,1,5,1,1,1,256,256...0,0,0,0(padding with 0)
8D Embedding
4D
4D
Conv1D
Conv1D+Sigmoid
128*FeatureMaps
128*FeatureMaps
Gate
Global Max-pooling
128
128D Fully Connected Layer+Relu
2
Malware
Softmax
Benign

FIGURE 1. Architecture of the CNN-based malware detector MalConv [12].

and retraining the original model to obtain the CAM, are difficult to implement for attackers. Selvaraju *et al.* [21] proposed a Grad-CAM method more recently, which uses the gradients of target concept flowing into the final convolutional layer to produce CAM, without modifying and retraining the original model. Inspired by these efforts, we use the Grad-CAM method to generate saliency vectors for subsequent attacks.

Let the convolution filter size of Malconv be s_{filter}. An input file $\boldsymbol{x}$ with length m can be divided into u data blocks of size s_{filter}, where $u = \lceil m/s \rceil$. $\boldsymbol{v_x^c} \in \mathbb{R}^u$ is used to represent the saliency vector of $\boldsymbol{x}$ for the classification target $c \in \{0,1\}$, which can be generated as follows. First, the gradient of the kth feature map A^k for the target label c is calculated, recorded as $\alpha_k^c = \frac{\partial y^c}{\partial A^k}$, which captures the importance of the kth feature map for target c. According to the importance of all feature maps, the weighted superposition is performed to obtain the saliency vector $\boldsymbol{v_x^c} = \{v_1^c, \ldots, v_u^c\}$, where v_i^c can be given by (1).

$$v_i^c = \begin{cases} \sum_k \alpha_k^c A_i^k, & i = \arg\max(A_{1,\ldots,u}^k) \\ 0, & i = other \end{cases} \tag{1}$$

If v_i^c is positive, then the feature of ith data block is malicious. Similarly, a negative value indicates benign and 0 indicates no significant feature. We used two 250KB benign and malicious files to calculate their saliency vectors respectively. In order to facilitate visualization, these two saliency vectors are converted into 500*500 grayscale images, as shown in Fig.2. The dark lines indicate benign regions and the bright lines represent malicious regions. The remaining regions with no significant features are represented by gray lines with a pixel value of 128. It can be seen that a saliency vector can effectively display the malicious and benign regions of

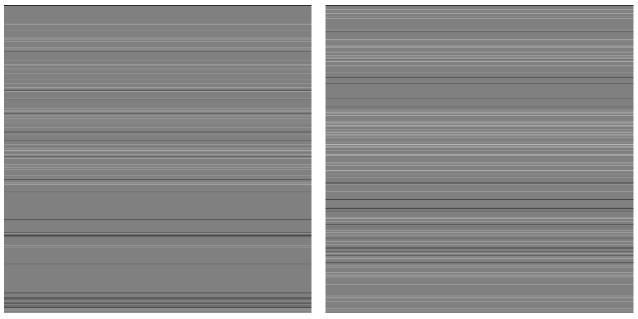

FIGURE 2. Grayscale images converted from the saliency vectors of a benign file (a) and a malicious file (b). (a) Benign. (b) Malware.

an input file. By comparison, it can be observed that the bright (malicious) region of the malicious file is obviously larger than the benign one.

C. WHITE-BOX ATTACK METHODS

In the following section, we introduce two white-box attack methods against Malconv using saliency vectors. Both methods require attackers to get the exact structure and all parameters of the victim model. The difference is that the first attack method only needs to debug the victim model once for generating saliency vectors, and then the model can be regarded as a black-box for subsequent attacks, while the second attack method, by incorporating the FGSM algorithm, requires continuous debugging of the model to obtain a higher attack success rate.

1) BENIGN FEATURES APPEND METHOD

The BFA attack method can generate effective AEs by appending carefully selected perturbations to the end of malicious files. These perturbations are selected from benign files by using saliency vectors. A file $\boldsymbol{x_0}$ that was predicted to be benign by the model is chosen to generate the saliency vector $\boldsymbol{v}$. First, the feature maps $\boldsymbol{A}$ are calculated by the Malconv model (line 1). The saliency vector $\boldsymbol{v} = \{ v_1^c, \ldots, v_u^c \}$ can be obtained using the Grad-CAM method described in Subsection III-B, where v_i^c can be given by Equation (1) (line 2). The positions of the negative elements in the vector are used as indexes of the benign feature blocks in $\boldsymbol{x_0}$, denoted as $\boldsymbol{l} = \{ l_1, l_2, \ldots, l_p \}$, where p is the number of data blocks with benign features (line 3). A benign block represents a data block (size is s_{filter}) corresponding to a benign feature index. If the size of a malicious file $\boldsymbol{x}$ is m and the maximum length of model input is L, then the maximum number of data blocks that can be appended to $\boldsymbol{x}$ is $\lfloor (L-m)/s_{filter} \rfloor$ (line 4). At each time, one data block is selected from p benign blocks as perturbations and appended to $\boldsymbol{x}$ for generating an AE $\tilde{\boldsymbol{x}}$ (line 7). This process loops up to $\min(p, \lfloor (L-m)/s_{filter} \rfloor)$ times until $\tilde{\boldsymbol{x}}$ is predicted to be benign (lines 6-13). It is important to note that the starting location for appending should be set to $padidx = (\lfloor m/s_{filter} \rfloor + 1) * s_{filter}$. For convenient illustration, we will ignore the padding bytes, which will be initialized to 0, from the end of the file to the starting location for appending. The more benign features are appended to the end of the malicious file $\boldsymbol{x}$, the greater the probability that $\tilde{\boldsymbol{x}}$ will be identified as a benign file by the victim model. The pseudocode is described in Algorithm 1.

Algorithm 1 The BFA Attack Method

```
Input:  $x_0, x, y_{benign}, s_{filter}, L, m$;
Output:  $\tilde{x}$;
1: $A = \text{Model}(x_0)$;
2: $v = \text{GradCAM}(A)$;
3: $l = \text{Index}(v < 0)$;
4: $q = \min(p, \lfloor (L-m)/s_{filter} \rfloor)$;
5: $padidx = (\lfloor m/s_{filter} \rfloor + 1) * s_{filter}$;
6: for $i = 1$ to $q$ do
7:     $\tilde{x} = \text{Append}(x, padidx, x_0[l_i : l_i + s_{filter}])$;
8:     if $\text{Model.predict}(\tilde{x}) == y_{benign}$ then
9:         return $\tilde{x}$;
10:    else
11:        $padidx = padidx + i * s_{filter}$;
12:    end if
13: end for
14: return $False$;
```

2) ENHANCED-BFA METHOD

The Enhanced-BFA attack method can significantly improve the attack success rate by incorporating the FGSM [15] algorithm with the BFA attack method. The FGSM-based attacks against Malconv proposed by Kreuk *et al.* [19] and Suciu *et al.* [20], using random noises to initialize the perturbations, could only achieve quite low attack success rate. By analyzing these random perturbations flowing in the model, we can discover why these methods are inefficient. At first, these random raw-byte perturbations are mapped to random vectors through the embedding layer, then features of these vectors are extracted by the convolution layer. Afterwards, these features are passed to the global maximum pooling layer before entering into the fully-connected layer. Since most feature values of these random vectors are not largest when passed to the global maximum pooling layer, these features will be ignored by the model before entering the subsequent fully-connected layer. In this situation, the back propagation gradients of these vectors will be 0 most of the time, and thus these random vectors cannot be modified by the gradient-based algorithms (e.g. FGSM) effectively. Moreover, since the raw-byte inputs are discrete and their mapped vectors are continuous, the modified vectors cannot be mapped back to raw bytes by looking up the table $\mathbf{W}$ directly. So even if these random vectors are modified successfully, the resulting AEs may also be invalid because the modified vectors cannot always be mapped back to valid raw-byte perturbations. Therefore, using random noise as the initial perturbations will make these methods have low success rates. In contrast, the appended perturbations in our

Algorithm 2 The Enhanced-BFA Attack Method

```
Input: $x_0, x, y_{benign}, s_{filter}, L, m, \varepsilon, \beta$;
Output: $\tilde{x}$;
1: $A = \text{Model}(x_0)$;
2: $v = \text{GradCAM}(A)$;
3: $l = \text{Index}(v < 0)$;
4: $q = \min(p, \lfloor (L - m)/s_{filter} \rfloor)$;
5: $padidx = (\lfloor m/s_{filter} \rfloor + 1) * s_{filter}$;
6: for $i = 1$ to $q$ do
7:     $\tilde{x} = \text{Append}(x, padidx, x_0[l_i : l_i + s_{filter}])$;
8:     $e_a = \mathbf{W}(x_a)$
9:     while $J(\theta, e_a, y_{benign}) >= \beta$ do
10:        $g = \text{sign}(\nabla_{e_a} J(\theta, e_a, y_{benign}))$
11:        $\eta_{pad} = \varepsilon \, \text{Mask}(g)$
12:        $e_a[padidx : padidx + i * s_{filter}] = \text{Mask}(e_a) - \eta_{pad}$
13:    end while
14:    $\tilde{x} = \mathbf{W}^{-1}(e_a)$
15:    if Model.predict($\tilde{x}$) == $y_{benign}$ then
16:        return $\tilde{x}$;
17:    else
18:        $padidx = padidx + i * s_{filter}$;
19:    end if
20: end for
21: return False;
```

attack method, which are initialized by significant benign blocks, can quickly attract the attention of the model to obtain back propagation gradients. On this basis, by incorporating the FGSM algorithm to iteratively modify these perturbations, AEs can be generated more efficiently.

The pseudocode of the Enhanced-BFA attack method is described in Algorithm 2. First, using the BFA attack method introduced in the previous section, a benign block index vector $\boldsymbol{l} = \{l_1, l_2, \ldots, l_p\}$ is generated (lines 1-3). According to the index number, a benign block will be selected from the original benign file $\boldsymbol{x_0}$ as initial perturbations and appended to the malicious file $\boldsymbol{x}$ (line 7). Then the appended file $\boldsymbol{x_a}$ is mapped to the matrix $\boldsymbol{e_a}$ through the embedding layer $\mathbf{W}$ (line 8). Let θ be the parameters of the model, y_{benign} the target label ('benign' in our case) and $J(\theta, \boldsymbol{e_a}, y_{benign})$ the cost function used by the model, then the optimal perturbation direction can be expressed as $\boldsymbol{g}$, which is given as follows:

$$\boldsymbol{g} = \text{sign}\left(\nabla_{\boldsymbol{e_a}} J\left(\theta, \boldsymbol{e_a}, y_{benign}\right)\right) \tag{2}$$

where $\boldsymbol{g}$ is a vector whose elements are equal to the sign of the elements of the gradient of the cost function J with respect to $\boldsymbol{e_a}$. The perturbations will be iteratively modified until the value of the cost function is below the threshold β (lines 9-13). In this loop, Mask operation limits the region modified by the perturbations and ε is a weight factor which determines the extent of each modification (lines 11-12). The larger the value of ε, the faster the process but the optimal solution may not be obtained. In our case, only the appended region can be iteratively modified by imperceptibly small perturbations. Afterwards, by using the K-Nearest Neighbor (KNN) [41] algorithm to find the nearest bytes for vectors in $\boldsymbol{e_a}$, the modified matrix $\boldsymbol{e_a}$ is mapped back to a sequence of raw bytes $\tilde{\boldsymbol{x}}$ (line 14). If $\tilde{\boldsymbol{x}}$ is predicted to be benign, then the function will return the successful AE $\tilde{\boldsymbol{x}}$ (lines 15-16), otherwise, the above process will continue to loop until no more data blocks can be appended (lines 6-20).

D. BLACK-BOX ATTACK METHODS

In the following section, we introduce two black-box attack methods for situations when attackers cannot know the exact structure and internal parameters of the victim model. In order to speed up the attack process, it is assumed that the convolution filter size s_{filter} is known. Apart from this, according to the input file, only the classification result of the model can be obtained.

1) RANDOM METHOD

Our Random method, different from the random method proposed by Suciu *et al.* [20], appends randomly selected data blocks of a benign file instead of random noises to increase the success rate. First, a benign file $\boldsymbol{x_0}$ of moderate size m is divided into p data blocks as perturbations, where $p = \lceil m/s_{filter} \rceil$. Then at each time a data block will be randomly selected from the p data blocks and appended to the malicious file $\boldsymbol{x}$ as perturbations to generate AE $\tilde{\boldsymbol{x}}$. Assuming that the maximum number of data blocks that can be appended to $\boldsymbol{x}$ is q, this process will be repeated up to min(p, q) times until the model predicts $\tilde{\boldsymbol{x}}$ as benign.

2) EXPERIENCE-BASED METHOD

The Experience-based attack method selects the perturbations by summarizing the successful trajectories of Random attacks. By calculating the contribution degree of each data block to the successful trajectories, data blocks with high contribution degrees will be used as perturbations for subsequent attacks. First, N Random attacks are performed and n successful trajectories $\boldsymbol{\tau} = (\boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \ldots, \boldsymbol{\tau_n})$ are recorded, where $\boldsymbol{\tau_k} = (b_1^k, b_2^k, \ldots, b_i^k) \in \mathbb{R}^{l_k}$ represents the kth successful trajectory, with each b_i^k representing the index of a data block in $\boldsymbol{x_0}$ used for the kth successful trajectory and l_k indicating the total number of data blocks appended in kth trajectory. Then the number of times each data block appearing in $\boldsymbol{\tau}$ is counted and multiplied by the corresponding weights. Thus the contribution degree d_j of each block in $\boldsymbol{x_0}$ can be computed as follows:

$$d_j = \sum_{k=1}^{n} \left(\frac{1}{l_k} \text{count}(\boldsymbol{\tau_k}, j) + \alpha I(b_{l_k}^k == j)\right) \tag{3}$$

where $j \in (1, \ldots, p)$, count$(\boldsymbol{\tau_k}, j)$ indicates the number of occurrences of jth block in $\boldsymbol{\tau_k}$, $\frac{1}{l_k}$ is a penalty factor, indicating that the more data blocks required for the kth trajectory, the smaller the importance of the trajectory, α indicates the extra weight of the last data block in $\boldsymbol{\tau_k}$ and I is the indicator function ($I \in (0, 1)$). Since the entire trajectory is successful after appending the last data block, the last data block is given an extra weight α to highlight its contribution. Then a vector $\boldsymbol{d} = (d_1, d_2, \ldots, d_p)$ with contribution degrees of all the data blocks in $\boldsymbol{x_0}$ is obtained. Finally, $\boldsymbol{d}$ is sorted from high to low and its first q data blocks are chosen to be the final perturbations for subsequent attacks. The subsequent attack process is similar to the BFA attack.

IV. EXPERIMENTS

In this Section, we describe the experiments carried out for evasion attacks. First, the dataset for training is described. Then the malware detector Malconv is trained and its performance for malware detection is evaluated. Afterwards, the saliency vectors for all files in the dataset are generated to analyze the quantity distribution of benign and malicious blocks. Finally, the trained Malconv is attacked using the methods introduced above and the effects of each attack method are analyzed in detail.

A. DATASET

In our experiments, two groups of PE files are collected to form a dataset. The first group is the malware set containing 5,200 malicious files from three projects: VirusShare [42], DAS [43], and malwarebenchmark [44], where each file is labelled with 1. The second group is the benign set containing 5,150 benign files from the pure version of Windows XP (32bit/SP3), Windows 7 ultimate (64bit/SP1), Windows 8.1 (64bit) image and more than 30 software companies, where each file is labelled with 0. All these files are larger than 1KB and less than 2MB, including multiple PE file types (such as exe, dll *etc.*). Source of files in the dataset is shown in Fig.3.

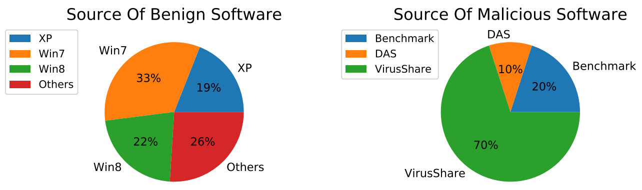

FIGURE 3. Source of files in the dataset. Benign (Left) and malware (Right).

B. MODEL PERFORMANCE EVALUATION

The Malconv model is reproduced by using the Keras library [45] and its parameters used for training are as follows: the maximum filesize is 2,000,000 Bytes, 1-dimensional convolution filter size is 500 and stride is 500. Other parameters are shown in Fig.1. The dataset is shuffled and divided into a training set, validation set and test set by 80%, 10%, and 10% respectively. As with the original literature, the metrics accuracy and AUC are used to evaluate the performance of the model. All experiments are performed on a CUDA-enabled NVIDIA Tesla K80 GPU server. In order to lower the bias, the training and testing processes are repeated 5 times, as shown in Table 1. It can be seen that the average accuracy and AUC are similar to the original literature reported (ACC=94.0%, AUC=98.1%) [12].

TABLE 1. Performance (Accuracy and AUC) of Malconv.

Experiment	1	2	3	4	5	mean
ACC	93.0%	93.8%	94.0%	90.7%	90.0%	92.3%
AUC	97.6%	97.1%	98.4%	96.4%	97.1%	97.2%

C. SALIENCY VECTORS GENERATION

Saliency vectors for all files in our dataset are generated to analyze the quantity distribution of benign and malicious blocks in a file. The joint distribution maps are shown in Fig.4, where the horizontal axis represents the number of benign blocks in a file and the vertical axis represents the number of malicious blocks. According to the Pearson Correlation Coefficient, we can see that there is a strong linear correlation between the number of benign blocks and malicious blocks contained in each file. In addition, Fig.4 shows that the number of benign blocks in a benign file is mainly 10-30. For convenience of description, 20 is used as the number of benign blocks in a benign file.

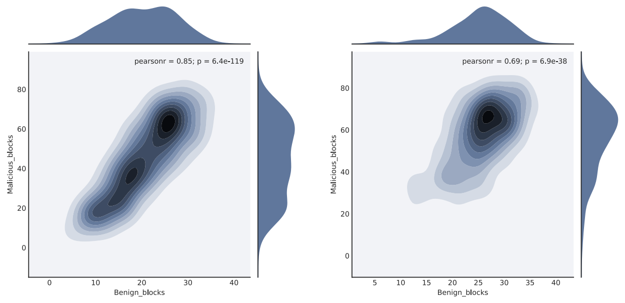

FIGURE 4. The joint distribution maps show the number of benign and malicious blocks in a file. Benign files are showed in the left map and malicious files are showed in the right map.

D. EVASION ATTACKS

Using the model trained in Section IV-B as the original model, two white-box methods in Section III-C, two black-box methods in Section III-D and the FGSM-based method described by Suciu *et al.* [20] are used to generate AEs for all files in the malware dataset. The effectiveness of all these attacks are evaluated by using the success rate (SR): the percentage of AEs that can successfully evade detection.

Table 2 gives the results of SR using three white-box attack methods, with different sizes of appended perturbations. The BFA method can achieve a SR of 60% by appending 20 benign blocks (10,000 bytes) taken from only one benign file. If another benign file is taken to get 40 benign blocks (20,000 bytes) for attacking, the SR will increase to 90%. Furthermore, the Enhanced-BFA attacks ($\varepsilon = 0.01$, $\beta = 0.001$) can achieve a SR of 74% by appending only one benign block (500 bytes). By appending 40 benign blocks, more than 99% of AEs can successfully evade detection. In contrast, it can be seen that the SR of the FGSM-based attacks is very low, only about 1 2%, which is similar to the result on Mini-dataset reported by Suciu *et al.* [20]. The FGSM-based

TABLE 2. The success rates of 3 white-box attack methods with different sizes of appended perturbations.

Appending Bytes	FGSM	BFA	Enhanced-BFA
500	1%	3%	**74%**
2000	1%	4%	**81%**
5000	2%	23%	**90%**
10000	2%	60%	**96%**
20000	2%	90%	**99%**

(also known as FGM) attack method in [20], which could only achieve quite low SR on the Mini-dataset (8,598 files) but achieve 71% SR on the Full-dataset (16.3 million files), are unstable and the reason why this method is inefficient has been analyzed in section III-C.2.

Table 3 gives the results of SR using two black-box attack methods. The Random attack method is used as a baseline to evaluate other methods. Although the Random method achieves low SR, its successful attack trajectories can be recorded to implement the Experience-based attacks. We performed 19,150 Random attacks (by appending 40 data blocks) and recorded 1,340 successful trajectories, such that 40 high contribution degree blocks could be obtained for subsequent attacks. The most interesting finding is that more than half of these blocks are identical to the benign blocks obtained by the BFA attack method, which means that we can use the Experience-based method instead of the BFA method when the model details are unknown. The results in Table 3 show that the SR of Experience-based attack ($\alpha = 2$) is similar to that of the Full-dataset FGSM attack reported in [20]. Moreover, if more Random attacks can be performed, more effective benign blocks will be obtained through more successful trajectories, resulting in a better performance of Experience-based attacks. The most important is that the Experience-based method does not need to understand the exact structure and parameters of the victim model. Fig.5 plots the SR curves of the above five attack methods, with different sizes of appended perturbations. It can be seen that the Enhanced-BFA method achieves the highest SR with minimal perturbations. Although BFA and Experience-based method start to achieve lower SR, SR will rise rapidly as the perturbations increases.

TABLE 3. The success rates of 2 black-box attack methods with different sizes of appended perturbations.

Appending Bytes	Random	Experience-based
500	1%	**9%**
2000	2%	**35%**
5000	3%	**53%**
10000	4%	**63%**
20000	7%	**73%**

In summary, the two write-box methods and one black-box method we proposed can effectively generate AEs to deceive the CNN-based malware detector Malconv. The Enhanced-BFA attack method is the most efficient when the exact structure and parameters of the victim model can

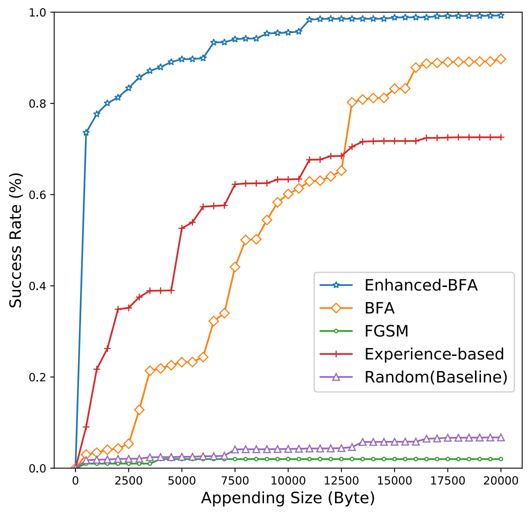

FIGURE 5. The success rates of 5 attack methods with different sizes of appended perturbations. Random attack is the baseline and FGSM-based attack is proposed in [20].

be given beforehand. If the model details are unknown, the Experience-based attack method can also achieve satisfactory results. Although these attacks in this paper are all against Malconv, they can be readily extended to other similar CNN-based malware detectors.

V. DEFENSES

In this section, we investigate two defensive mechanisms against evasion attacks, one is adversarial training, and the other is rejecting AEs. Adversarial training can improve the robustness of the model itself to AEs, and rejecting AEs is a pre-detection mechanism that requires an additional database system to help identify the AEs.

1) ADVERSARIAL TRAINING

The most common defensive mechanism against evasion attacks is adversarial training, which was first proposed by Szegedy *et al.* [35]. Adversarial training involves the following steps: a) Train the model $\boldsymbol{F}$ on the original training set $D = M \cup B$, where M is the malware set, and B is the set of benign; b) generate AEs set A against $\boldsymbol{F}$ using the evasion attack methods described in Section III; c) modify the training set $\tilde{D} = D \cup A$ and perform iterative training on the basis of $\boldsymbol{F}$.

First, two groups of effective AEs sets A_{train} and A_{test}, containing 5,168 and 5,152 AEs respectively, are actively generated by using the Enhanced-BFA method. The initial perturbations required to generate these two group AEs are selected from two files, x_{train} in the training set and x_{test} in the test set. These two files are all predicted to be benign by $\boldsymbol{F}$ with high probability. Different number of AEs from these two AEs groups are used to create new training sets and then the new model $\tilde{\boldsymbol{F}}$ can be retrained based on $\boldsymbol{F}$. Afterwards, $\tilde{\boldsymbol{F}}$ is used to evaluate the SR of all AEs and to test whether x_{train} and x_{test} are correctly classified. Table 4 reports the evaluation and test results in detail. The first two lines indicate the SR

TABLE 4. Evaluation of adversarial training by adding different number of AEs to training set.

Add Number	2	5	10	50	100	200
SR(A_{train})	59.8%	24.9%	21.8%	4.6%	1.6%	0.4%
SR(A_{test})	62.2%	28.4%	22.5%	4.4%	1.4%	0.6%
x_{train}	✓	✓	✓	✓	✓	✓
x_{test}	✓	✓	✓	✓	×	×

of the two groups of AEs against the retrained model $\tilde{\boldsymbol{F}}$. SR(A_{train}) represents the SR of the first group of AEs A_{train} and SR(A_{test}) the SR of the second group A_{test}. The last two lines indicate whether x_{train} and x_{test} are correctly classified by $\tilde{\boldsymbol{F}}$.

As can be seen from the data in Table 4, when 5 AEs are added to the training set (8,280 files), the SR of other AEs in both groups will reduce to 29%. If 50 AEs are added, the SR is only less than 5% and lower than the detection error rate (7.7%) of the original model, thus the remaining few AEs that can successfully avoid detection can be ignored. It can also be observed that when 100 AEs are added, although the SR decreased to 1.4%, the file x_{test}, which is considered to be benign by $\boldsymbol{F}$, is incorrectly identified as malicious by $\tilde{\boldsymbol{F}}$. Therefore, we urge researchers to pay attention to this problem, that is, implementation of adversarial training needs to be cautious. The same group of AEs cannot be used in large numbers for adversarial training. Otherwise, the training dataset will be poisoned. Moreover, it is necessary to prevent attackers from adopting a similar method to implement poisoning attacks [46]. Attackers can use essential operation system PE files to generate a large number of ineffective AEs and send them to the detector. Although these AEs cannot evade detection, they may be added to the training set as regular malware by the defender, resulting in the training dataset being poisoned. If the defender retrains a new detector by using the poisoned training dataset, the essential system files may be misclassified as malware. In this case, the new detector may in turn damage rather than protect the operating system, so there is an urgent need for a defensive mechanism to reject AEs.

2) REJECTING ADVERSARIAL EXAMPLES

We next propose a pre-detection mechanism to identify AEs, and these identified AEs are rejected before entering into the malware detector. Fig.6 shows the specific architecture of this mechanism. A database table is used to record the characteristics of malware being detected. Through the expertise of the PE format, it is generally agreed that directly modifying the executable section of a PE file is difficult, because a subtle change may lead to unpredictable consequences, such as destruction function or running error. Therefore, it is assumed that the hash values of the executable section in AEs for the same malicious file should be equal. The input x is parsed by LIEF library [47] to get its executable code section x_{code}. So the hash values, denoted as Hash(x_{code}), can be used as the indexes of the table. The other fields of the table are

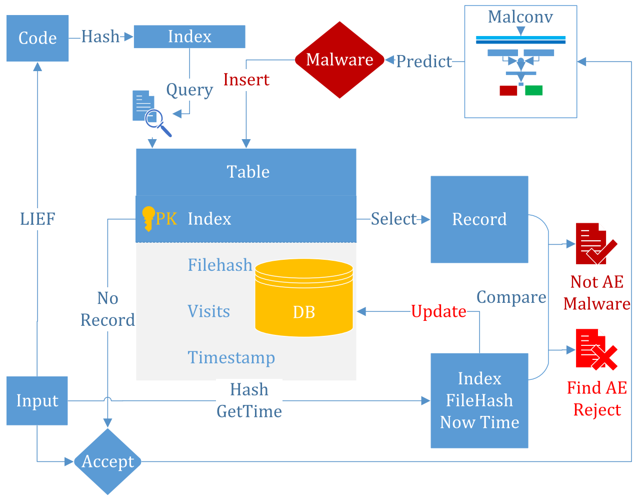

FIGURE 6. The architecture of the pre-detection mechanism for rejecting AEs.

including the hash value of the entire file Hash(x), number of times the record has been accessed x_{num} and the last accessed timestamp t_{now}. The pre-detection mechanism identifies and records the AEs by querying and updating the database.

The Algorithm 3 highlights the architecture of our pre-detection mechanism. First, Hash(x_{code}) of the input file x is used as the index to query the database. If a record exists, it first indicates that the input file is malware, then the values of the record and the file attributes are compared to discriminate whether the input file is an AE. AEs will be dropped but normal malware can be stored for retraining. No record means that the input file can be accepted and predicted by the malware detector. If the file is predicted to be malware, its characteristics will be inserted into the table as a new record. In order to prevent a large number of records in the table from affecting the query speed, one can set a time threshold T to periodically delete some expired records.

By performing the pre-detection mechanism based on the trained Malconv, we build a malware detection system which deployed in the following configuration server: 40 Xeon(R) E5-2630 cpus, 32*16G memory, Ubuntu16.04 64-bit operating system, MYSQL 14.14 database which is accessed locally by the pymysql library of Python 3.6. This detection system continuously detects normal files (malware and benign) for 24 hours, and every 10 seconds an AE was inserted to the normal files. The experiment shows that 100% of the AEs can be accurately rejected. Rejecting all AEs means that our pre-detection mechanism can successfully help the malware detector resist evasion attacks. In order to evaluate the efficiency of the detection system, the **clock** function in Python is used to calculate the cpu time spent by the pre-detection mechanism and malware detector respectively. The cpu time used for detecting 800 files by Malconv is about 3,800s, but for pre-detection (e.g. file parsing and hash operations, database connecting and querying operations) is not more than 10s. That is to say, if the number of malware exceeds

Algorithm 3 The Pre-Detection Mechanism for Rejecting AEs

```
Input:  $x, N$;
Output:  malware,benign,reject;
1: $x_{code}$ = LIEF($x$);
2: $index$ = Hash($x_{code}$);
3: $x_{hash}$ = Hash($x$);
4: $t_{now}$ = Now();
5: QueryDB($index$);
6: if record exists then
7:     $r_{num}, r_{hash}, r_{time}$ = GetValue($index$) ;
8:     if ( $r_{num} > N$ ) and ($r_{hash} \neq x_{hash}$) then
9:         UpdateDB($index, r_{num} + 1, t_{now}$)
10:        return reject;
11:    else
12:        return malware;
13:    end if
14: else
15:    if Model.predict($x$) ==‘malware‘ then
16:        InsertDB($index, 1, x_{hash}, t_{now}$) ;
17:        return malware;
18:    else
19:        return benign;
20:    end if
21: end if
```

0.26% of the total files being detected, the efficiency of the detection system will be higher than the single detector. In summary, the pre-detection mechanism can not only effectively reject the AEs, but also greatly improve the efficiency of the malware detection system.

VI. CONCLUSIONS

In this paper, we extensively investigated the vulnerability of the CNN-based malware detectors, specifically a recently proposed detector Malconv. We proposed two white-box attack methods and one black-box attack method to attack these CNN-based malware detectors. By implementing these attack methods to Malconv, our Enhanced-BFA white-box method can achieve an attack success rate of over 99%, and our Experience-based black-box method can also achieve a success rate of over 70%. These high attack success rates strongly demonstrate the vulnerability of such CNN-based malware detectors. Our Enhanced-BFA method, combining the Grad-CAM and FGSM algorithms to improve the attack success rate, can be readily extended to other similar adversarial machine learning tasks. In addition, we considered adversarial training as a defensive mechanism in order to resist evasion attacks. Experiments show that although adversarial training has a certain effect, there is a risk that the training dataset can be poisoned by a large number of AEs. Finally, we proposed a pre-detection mechanism to reject the AEs. Experiments show that this mechanism can not only effectively reject evasion attacks, but also improve the efficiency of malware detection.

There are several future research works. We plan to modify the raw bytes of the code section of the PE files to generate AEs, and such AEs will be difficult to detect. We also plan to repeat the attack methods of this paper on a larger production-scale dataset. Moreover, the database system, which may cause other security risks (e.g. the vulnerabilities of MySQL [48]), is introduced by our pre-detection mechanism. Therefore, we will continue to study the model-based defensive mechanisms to resist the evasion attacks and ultimately improve the robustness of the model detectors themselves.

REFERENCES

[1] Symantec. (2018). *The 2018 Internet Security Threat Report (ISTR)*. [Online]. Available: https://www.symantec.com/security-center/threat-report

[2] Y. Ye, T. Li, D. A. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 41:1–41:40, 2017. doi: 10.1145/3073559.

[3] J. Aycock, "Anti-anti-virus techniques," in *Computer Viruses and Malware*. Boston, MA, USA: Springer, 2006, pp. 97–108.

[4] D. Ucci, L. Aniello, and R. Baldoni. (2017). "Survey on the usage of machine learning techniques for malware analysis." [Online]. Available: http://arxiv.org/abs/1710.08189

[5] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "N-gram-based detection of new malicious code," in *Proc. 28th Annu. Int. Comput. Softw. Appl. Conf.*, Sep. 2004, pp. 41–42.

[6] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw.*, Oct. 2015, pp. 11–20. doi: 10.1109/MALWARE.2015.7413680.

[7] K. Rieck, T. Holz, C. Willems, and P. Düssel, and P. Laskov, "Learning and classification of malware behavior," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*, 2008, pp. 108–125. doi: 10.1007/978-3-540-70542-0_6.

[8] G. Yan, N. Brown, and D. Kong, "Exploring discriminatory features for automated malware classification," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*, 2013, pp. 41–61. doi: 10.1007/978-3-642-39235-1_3.

[9] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, Feb. 2017 pp. 4278–4284. [Online]. Available: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806

[10] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Oct. 2014, pp. 1746–1751. [Online]. Available: http://aclweb.org/anthology/D/D14/D14-1181.pdf

[11] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, New Orleans, LA, USA, Mar. 2017, pp. 4845–4849. doi: 10.1109/ICASSP.2017.7953077.

[12] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, "Malware detection by eating a whole EXE," in *Proc. Workshops 32nd AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, Feb. 2018, pp. 268–276. [Online]. Available: https://aaai.org/ocs/index.php/WS/AAAIW18/paper/view/16422

[13] M. Krcál, O. Svec, M. Bálek, and O. Jasek, "Deep convolutional malware classifiers can learn from raw executables and labels only," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018. [Online]. Available: https://openreview.net/forum?id=HkHrmM1PM

[14] C. Szegedy *et al.* (2013). "Intriguing properties of neural networks." [Online]. Available: http://arxiv.org/abs/1312.6199

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy. (2014). "Explaining and harnessing adversarial examples." [Online]. Available: http://arxiv.org/abs/1412.6572

[16] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Mar. 2016, pp. 372–387. doi: 10.1109/EuroSP.2016.36.

[17] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 506–519. doi: 10.1145/3052973.3053009.

[18] B. Kolosnjaji *et al.*, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *Proc. 26th Eur. Signal Process. Conf.*, Sep. 2018, pp. 533–537. doi: 10.23919/EUSIPCO.2018.8553214.

[19] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, and J. Keshet. (2018). "Adversarial examples on discrete sequences for beating whole-binary malware detection." [Online]. Available: http://arxiv.org/abs/1802.04528

[20] O. Suciu, S. E. Coull, and J. Johns. (2018). "Exploring adversarial examples in malware detection." [Online]. Available: http://arxiv.org/abs/1810.08280

[21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 618–626. doi: 10.1109/ICCV.2017.74.

[22] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 2001, pp. 38–49. doi: 10.1109/SECPRI.2001.924286.

[23] M. Pietrek, "Peering inside the PE: A tour of the win32 (R) portable executable file format," *Microsoft Syst. J.-US Ed.*, pp. 15–38, 1994.

[24] E. Raff, J. Sylvester, and C. Nicholas, "Learning the PE header, malware detection with minimal domain knowledge," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Dallas, TX, USA, Nov. 2017, pp. 121–132. doi: 10.1145/3128572.3140442.

[25] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Inf. Sci.*, vol. 231, pp. 64–82, May 2013. doi: 10.1016/j.ins.2011.08.020.

[26] A. Sharma, S. K. Sahay, and A. Kumar. (2016). "Improving the detection accuracy of unknown malware by partitioning the executables in groups." [Online]. Available: http://arxiv.org/abs/1606.06909

[27] B. Kolosnjaji, A. Zarras, G. D. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Proc. Australas. Joint Conf. Artif. Intell.*, 2016, pp. 137–149. doi: 10.1007/978-3-319-50127-7_11.

[28] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 1916–1920. doi: 10.1109/ICASSP.2015.7178304.

[29] B. Athiwaratkun and J. W. Stokes, "Malware classification with LSTM and GRU language models and a character-level CNN," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 2482–2486. doi: 10.1109/ICASSP.2017.7952603.

[30] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2574–2582. doi: 10.1109/CVPR.2016.282.

[31] N. Papernot, P. D. McDaniel, and I. J. Goodfellow. (2016). "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples." [Online]. Available: http://arxiv.org/abs/1605.07277

[32] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth. (2017). "Learning to evade static PE machine learning malware models via reinforcement learning." [Online]. Available: http://arxiv.org/abs/1801.08917

[33] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. D. McDaniel, "Adversarial examples for malware detection," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2017, pp. 62–79. doi: 10.1007/978-3-319-66399-9_4.

[34] W. Hu and Y. Tan. (2017). "Generating adversarial malware examples for black-box attacks based on GAN." [Online]. Available: http://arxiv.org/abs/1702.05983

[35] N. Papernot and P. D. McDaniel. (2017). "Extending defensive distillation." [Online]. Available: http://arxiv.org/abs/1705.05264

[36] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, and R. Poovendran. (2017). "Blocking transferability of adversarial examples in black-box learning systems." [Online]. Available: http://arxiv.org/abs/1703.04318

[37] J. Lu, T. Issaranon, and D. A. Forsyth, "Safetynet: Detecting and rejecting adversarial examples robustly," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 446–454. doi: 10.1109/ICCV.2017.56.

[38] A. Al-Dujaili, A. Huang, E. Hemberg, and U. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," in *Proc. IEEE Secur. Privacy Workshops, SPW*, San Francisco, CA, USA, May 2018, pp. 76–82. doi: 10.1109/SPW.2018.00020.

[39] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, Aug. 2017, pp. 933–941. [Online]. Available: http://proceedings.mlr.press/v70/dauphin17a.html

[40] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 2921–2929. doi: 10.1109/CVPR.2016.319.

[41] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Information Theory*, vol. 13, no. 1, pp. 21–27, Jul. 1967. doi: 10.1109/TIT.1967.1053964.

[42] VirusShare. (2015). *Virusshare.com*. Accessed: May 05, 2018. [Online]. Available: https://virusshare.com/

[43] DAS. (2015). *DAS MALWERK*. Accessed: May 05, 2018. [Online]. Available: https://dasmalwerk.eu/

[44] G. Liang, J. Pang, Z. Shan, R. Yang, and Y. Chen, "Automatic benchmark generation framework for malware detection," *Secur. Commun. Netw.*, vol. 2018, pp. 4947695:1–4947695:8, Aug. 2018. doi: 10.1155/2018/4947695.

[45] F. Chollet *et al.* (2015). *Keras*. [Online]. Available: https://github.com/keras-team/keras

[46] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, Los Angeles, CA, USA, Dec. 2016, pp. 508–519. [Online]. Available: http://dl.acm.org/citation.cfm?id=2991125

[47] R. Thomas. (Apr. 2015). *LIEF—Library to Instrument Executable Formats*. [Online]. Available: https://lief.quarkslab.com/

[48] J. Fonseca, M. Vieira, and H. Madeira, "Evaluation of Web security mechanisms using vulnerability & attack injection," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 5, pp. 440–453, Sep./Oct. 2014.

BINGCAI CHEN received the M.S. and Ph.D. degrees in information and communication engineering from the Harbin Institute of Technology (HIT), Harbin, China, in 2003 and 2007, respectively.

He has been a Visiting Scholar with the University of British Columbia, Canada, in 2015. He is currently an Associate Professor with the School of Computer Science and Technology, Dalian University of Technology, Dalian, China, and also with the School of Computer Science and Technology, Xinjiang Normal University, Urumqi, China. His current research interests include computer vision, wireless ad hoc networks, and network and information security.

Dr. Chen received the National Science Foundation Career Award of China, in 2009. He is severing as a Reviewer for project proposals with the National Science Foundation of China, Ministry of Education of China. He is also serving as a Reviewer for some refereed journals including the IEEE/ACM TRANSACTIONS ON NETWORKING, *Journal of Electronics*, and the *Journal of Communication*.

ZHONGRU REN received the B.S. degree in computer science and technology from Zhenzhou University, Zhenzhou, China, in 2011. He is currently pursuing the M.S. degree with the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. His current research interests include information security and deep learning.

CHAO YU received the M.S. degree in computer science and technology from Huazhong Normal University, Wuhan, in 2010, and the Ph.D. degree in computer science from the University of Wollongong, Wollongong, NSW, Australia, in 2013. He is currently an Associate Professor with the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. His current research interest includes multiagent systems and learning, with their wide applications in modeling and solving various real-world problems.

IFTIKHAR HUSSAIN received the B.S. degree in computer science from the Islamia College University of Peshawar, in 2013, and the M.S. degree in computer science and technology from the Dalian University of Technology, Dalian, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. His research interests include information security and wireless networks.

JINTAO LIU received the B.S. degree in computer science and technology from Zhenzhou University, Zhenzhou, China. He is currently pursuing the M.S. degree with the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. His current research interests include natural language processing and code retrieval.