Transferability of Adversarial Examples in Machine Learning-based Malware Detection

Yang Hu*, Ning Wang*, Yimin Chen†, Wenjing Lou*, Y. Thomas Hou*
*Virginia Polytechnic Institute and State University, VA, USA,
†University of Massachusetts Lowell, MA, USA
Email: *{yanghu, ning18, wjlou, thou}@vt.edu, †ian_chen@uml.edu

***Abstract*—Machine Learning (ML) has been increasingly applied to malware detection in recent years. Adversarial example (AE) attack, a well-known attack against ML working across different mediums, is effective in evading or misleading ML-based malware detection systems. Such an attack could be made more effective if the generated AEs are transferable so that the AEs can evade different types of malware detection models. To better understand AE transferability in the malware domain, in this paper, we study AE transferability enhancement techniques and how they impact AE generation and Android malware detection. Firstly, we adapt the current image-based AE transferability enhancement techniques (i.e., ensemble sample (ES) and ensemble model (EM)) to malware. In the adapted ES and EM methods, we maintain malware functionality and executability while adding perturbations. Further, we develop a new transfer-based AE generation method, BATE, using a novel *feature evenness* metric. The idea is to spread perturbations more evenly among perturbed features by incorporating an evenness score in the objective function. We compare our proposed methods with EM and ES on a real Android dataset. The extensive evaluations demonstrate the effectiveness of our method in increasing the upper bound of AE transferability. We also confirm the effectiveness of our evenness-score-based method by showing quantitative correlations between AE transferability and feature evenness score.**

***Index Terms*—malware detection, AE attack, transferability**

I. Introduction

Recent years have seen rapid advancements in machine learning (ML) in both theory and deployment. A variety of applications, such as object detection and natural language processing, have achieved remarkable success, due to the use of machine learning technologies. ML is also widely used in the security domain such as network intrusion detection and malware detection. Compared with the traditional signature-based detection methods, ML-based detection systems not only exhibit high detection accuracy but also show the capability of detecting unseen and/or zero-day attacks.

With the advancement of ML mechanisms, ML-based detection systems have outperformed their predecessors and become the mainstream for many security applications. However, they suffer from the same vulnerabilities that most ML models do. Adversarial examples (AE) attack is one of the most devastating attacks on ML models because of their imperceptible nature and misleading capability. As shown in [1], an ML model will misclassify a testing image to the attacker's targeted class by adding small (even negligible) perturbations to it. As for malware, an AE attacker is able to mislead a well-trained malware detector to miss detecting malware by slightly modifying some features of a piece of malware. AEs can be extremely powerful as an AE crafted from one ML model can also mislead another ML model to make the same misclassification. The power that an AE generated to evade one model can evade other models of different architectures is called *AE transferability*.

With transferability, an attacker is able to generate AEs using a *substitute model*, apply them to the *victim model*, and achieve a high attack success rate without the exact knowledge of the victim model. This is the so-called *"black-box transfer attack"* [2], [3]. The AEs generated from one ML model, e.g., the substitute model, are also effective for another model, e.g., the victim model, provided they are somehow related. One example of the substitute model and the victim model is that the substitute model and the victim model are obtained from the same training set with different training algorithms or settings. A higher transferability indicates that a high portion of generated AEs from the substitute model is effective for the victim model. The feasibility of the black-box transfer attack greatly enhances the attacker's capability, making the AE attack a practical one on real-world ML models.

AE transferability is significant for malware research since it is directly related to the attack success rate. Compared with the image classification tasks, AEs in the malware domain exhibit their distinct characteristics. Firstly, as a software intellectual property, a malware detection model is usually a black box. The model architecture and parameters of the malware detector are unknown to the third party (thus including potential AE attackers). It is challenging for an attacker to generate effective AEs without knowing this critical information. One way to overcome such a challenge is to design an approximate feature extractor and a substitute model and then generate AEs based on the substitute model. To improve the AE attack success rate (i.e., the probability that an AE evades the detection), attackers demand that the generated AE be highly transferable. Secondly, malware detection usually utilizes an ensemble of models for performance enhancement in practice. When facing such ensemble-based detection methods, AEs with high transferability would be preferred as these AEs are effective across different detection models.

Therefore, it is essential for us to understand AE transferability in the malware detection domain well. While AE attack has been extensively studied in the computer vision domain, AE transferability is relatively under-researched in malware detection. In this work, we investigate AE attacks in malware detection with a particular focus on improving AE transferability. To start with, we make efforts to adapt and evaluate the ensemble sample (ES) method (a popular technique for enhancing transferability) to malware detection models because most previous techniques from the computer vision domain, ES included, cannot be directly applied to malware due to the non-continuous nature of malware. Next, we proceed to design a new adversarial example generation method, *BATE*, to boost AE Transferability.

The intuition of *BATE* is that if the victim model's prediction is dominated by a small number of perturbed features, the AE generated on the victim model will not succeed in fooling other models that do not rely heavily on these features. We observe that current AE generation methods perturb some features more heavily than other features. This is one factor that negatively impacts AE transferability and thus negatively impacts the AE detection capability. To address this problem, we perturb a larger set of features and make all features have a similar importance for the model's prediction. To this end, we incorporate an evenness score to the AE generation objective to generate AEs with perturbations spread across features more evenly. We demonstrate that even perturbation is effective in improving AE transferability.

"Know your enemy" is always the first step to building an effective defense. We believe a good understanding of the attack strategies will provide valuable insights into designing effective defense mechanisms. We summarize our contributions as follows.

- We adapt the ES and EM methods to malware detection from the computer vision domain. Specifically, we define a perturbation method for malware that performs constrained manipulation on modifiable features to maintain malware's executability and functionality.
- We identify a new direction and propose a new method to improve AE transferability. Our mechanism, namely *BATE*, generates a perturbation that evenly modifies the modifiable features. Intuitively, if adversarial perturbation favors partial features, some models that do not rely heavily on these features will not be fooled. A more even perturbation of features would be more robust to different models.
- We perform extensive evaluations to study AE transferability. For EM methods, we evaluate the impact of the ensemble models on AE transferability. We observe that the optimal EM method highly depends on the victim model. Our observation shed light on how to customize the ensemble model when facing different victim models. Further, we demonstrate that *BATE* can improve the upper bound of AE transferability among malware detection models. A correlation between AE transferability and feature evenness score is also studied to confirm our intuition in utilizing feature evenness.

II. Related Works

AE attacks have been well-explored in the computer vision domain in recent years. We will focus on AE transferability on ML-based malware detection systems.

A. AE attacks on the malware detection system

Most existing AE attacks on malware detection are designed for white-box malware classifiers. In [4], Grosse et al. were the first to launch AE attack on the static malware detectors. It attacks different DNN Android malware classifiers using the gradient perturbation method (FGSM) [5]. In their work, the perturbation could only be added in the Manifest file to not interfere with other features as much as possible to retain malware's semantics. The attack success rate ranges from 50% to 84% in their work. Many follow-up works on the same topic adopt a similar approach but use a slightly different objective function. Biggio et al. [6] try to evade the malicious PDF detection system using a gradient-based approach by adding new features. Based on the analysis of API calls, Rosenberg et al. [7] proposed a black-box attack against ML-based malware detectors on Windows operating system. The attack algorithm added non-operational system calls (extracted from harmless software) iteratively to the binary code. In [8], Hu et al. implemented the grey-box AE attack on the RNN-based malware detectors by inserting some irrelevant APIs into the original API call sequences. Recently, Anderson et al. [9] used reinforcement learning to evade the malware detection system by choosing from a predefined list of transformations that preserve semantics. Several gradient-based AE attacks [10], [11] are designed typically for the MalCov arthitecures [12] for PE files. Except for the gradient information, in [13], Xu et al. utilized the heuristics algorithms to find the optimal manipulation of PDFs while maintaining necessary syntax.

Most of the aforementioned research is based on the white-box adversary model, where the attacker is assumed to have access to the victim model. The methods in [8], [14] proposed the grey-box AE attack framework without requiring white-box access. These methods still need to query the targeted victim models frequently to adjust the perturbation in rounds. In a black-box scenario, the attacker is assumed to know no information about the victim model. An attacker with black-box access to the model needs to build a substitute model and generate AEs based on the substitute model. The attack success rate will highly depend on AE transferability.

B. Transferable AE attacks

The transferability of AEs was first discovered by Goodfellow et al. in [5]. They had the observation that the AEs would have higher transferability in the case that the intersecting or overlapping decision boundaries (for classification tasks) are often learned by two models. Later, in [2], [3] researchers found that even with very different model structures as SVMs, logistic regression models, or neural networks, AEs successfully transfer among these different ML models. Current prevailing methods to improve AE transferability can be split into two

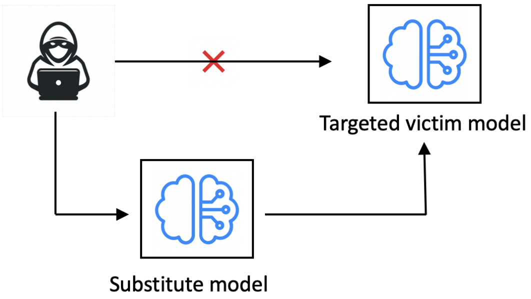

Fig. 1. transfer-based AE attacks.

genres. One is inspired by the data augmentation techniques. Works in [15]–[17] enhance the generated AE transferability by ensembling multiple samples together. Instead of only using the original images to generate AEs, they apply data augmentation to the input images in iterations as the input to generate the AE using the base FGSM [5] algorithms. Another genre increases the AE transferability by utilizing the comprehensive gradients information from ensembles of models technique [18]–[20]. In these works, the authors concluded that the ensembled model would provide a lower-variance model since it achieved a smoother and more stable decision boundary.

Extensive works have recognized the importance of the transferability property of AE, but there is still a lack of effort in the malware domain. Compared with computer vision data, malware perturbation is more challenging because of the need to maintain functionality and execution.

C. Transferable AE attacks on malware classifiers

A few recent works [14], [21], [22] explored the success rate of the transfer-based AE attacks on malware classifiers, and these works show much lower transferability compared with that in the CV domain. The transferable AE attacks on malware are an under-studied problem.

In this paper, we study how existing transferability enhancement techniques [17], [19] would impact malware AE transferability. We will demonstrate the limitations of current transferability enhancement techniques on malware classifiers and propose a new AE generation method to improve AE transferability. We hope our work on transferable AE attacks in the malware domain could advance the understanding of effective AE attack strategies and provide useful inputs to promote a more robust malware detection design.

III. System Model and Adversarial Model

A. System Model

In this work, we aim to generate AEs with higher transferability, which can evade different detection models in the grey-box scenario. We assume that the attacker has no information about the victim model except the feature representation X and the training data D of the targeted victim model. The attacker first builds a substitute model $f_s(x)$ using the feature representation X and the training data D. Attackers also have a set of testing samples that have been classified as malware by the victim model. Then AEs would be generated on this substitute model $f_s(x)$. In such a case, the victim model is a grey box for the attacker. Then the generated AEs are evaluated on the victim model again to see if they can successfully evade the victim model. Note that the attacker's goal is to improve the probability that AEs generated from the substitute model could also successfully evade the targeted victim model. Figure 1 illustrates the process of a transfer-based AE attack. Attackers train the substitute and generate AE from it for attacking the targeted victim model.

TABLE I
Categorization of attack capability.

Attack Category	Attacker's Information	Defense Difficulty
White-box attacks	$D, X, \hat{f_v}, \hat{w}$	Low
Grey-box attacks	X and a subset of D	Medium
Black-box attacks	None	Difficult

B. Adversary Model

Attackers' Goal is to increase the transferability of AE attacks on malware classifiers. The transferability is defined as the probability that AEs crafted from one white-box malware detection model (e.g., a substitute model to which the attacker has full access) can successfully evade/mislead the unknown victim model. A higher probability corresponds to a 'higher' transferability, which indicates that the AEs are more powerful.

Attackers' Capability refers to the amount of information the attacker knows about the victim model. From an attacker's viewpoint, there are four types of information about the victim model—the training dataset D, the feature representation of a sample X, the model architecture $\hat{f_v}$, and the model parameters $\hat{w}$. Depending on the amount of information the attacker has, we can divide attacks into three categories: white-box attacks, grey-box attacks, and black-box attacks. A white-box attacker knows all the information about the targeted victim model, i.e., $\{D, X, \hat{f_v}, \hat{w}\}$, a black-box attacker knows nothing about the victim model, and a grey-box attacker knows a subset of $\{D, X, \hat{f_v}, \hat{w}\}$. We also illustrate the differences in Table I. In this work, we adopt a grey-box attack in which the attacker knows X and a subset of D while she does not know $\hat{f_v}$ and $\hat{w}$. The grey-box assumption is realistic since an attacker should be able to obtain some training samples and X while much more challenging to obtain $\hat{f_v}$ and $\hat{w}$.

Attackers' Strategy. Transferable AE attacks under the grey-box adversary model consist of two steps. In Step One, the attacker builds a substitute model for the victim model using any resources she has under her disposal, including X and a subset of D. The substitute model is expected to make similar decisions as the victim model. Note that the substitute model is a white-box model to the attacker as she builds it by herself. In Step Two, given a malicious clean sample X, the attacker generates the corresponding AE X' for the substitute model, i.e., X' is a successful AE on the substitute model. In Step Three, the attacker applies X' to the victim model to see if X' works on the victim model or not. The higher the probability that X' works on the victim model, the higher the transferability of our attack.

IV. *BATE* Design

A. Intuition

ML models, particularly deep learning models, tend to be overfitted because of iterative training [23], which makes a trained model very sensitive to a small portion of 'important' features. In other words, small perturbations over these features could significantly change the output layer, thus making a trained model vulnerable to AE attacks. In [23]–[25], the authors showed that ML models are more robust against sparse attacks if the feature importance for model prediction is distributed more evenly across the features. Let $I(x_i)$ denote the importance of $x_i, i = 1, 2, \cdots, n_d$ (x_i is the i-th features in x). If $variance(I(x))$ is smaller, the corresponding ML model is likely to be more robust. On the contrary, assuming that different ML models have different sets of important features for their predictions (even under the same feature presentation X), it is intuitive that if the victim model's prediction is dominated by a small number of perturbed features, the AE generated on the victim model will not succeed in fooling other models that do not rely heavily on these features. In *BATE*'s design, we aim to perturb a larger set of features and make all features of similar importance for the model's prediction. We call the scenario where crafted AEs are only effective in the model used for AE generation 'overfitting'. We can avoid the generated AE from overfitting to a specific model by making all features with even contributions to the final prediction.

B. Evenness Score

The uniqueness of our transferable attack is that we aim to generate AEs with high evenness scores such that they are more likely to achieve high transferability. Our definition of evenness score can be easily illustrated in two steps. Recall that we denote an ML model and its input feature vector by $y = f(x)$ and x, respectively.

1) **Step One**, we use explainable ML techniques [24], [26] to compute the importance of each feature in x on the prediction result (i.e., $f(x)$). As a result, we obtain the contribution vector I of which I_i corresponds to the importance of x_i on $f(x)$.
2) **Step Two**, we compute an *Evenness Score*, r, using the following equation.

$$r = \frac{||I||_1}{n_d ||I||_\infty}. \tag{1}$$

Here $I = \{I_i\}_{i \in [1,n_d]}, \sum_{i=1}^{n_d} I_i = 1, n_d$ is the number of features in x, and $||I||_1$ and $||I||_\infty$ represent the L-1 norm and L-∞ norm of vector I.

Intuitively, we can boost AE transferability by minimizing the variance of $\{I(x'_1), I(x'_2), ..., I(x'_{n_d})\}$ of the generated x's, which makes x's become more likely to succeed in attacking the targeted victim model $f_v(\cdot)$.

C. Objective Function

Our objective is to find AE x' from x so that x' is classified to a targeted class t while its evenness score $r(x')$ is maximized. We formulate the objective function to generate AEs as:

$$\begin{aligned} \underset{\delta}{\text{maximize}} \quad & r(x+\delta), \\ \text{subject to} \quad & f_s(x+\delta) = t, \\ & x+\delta \in F^{n_d}. \end{aligned} \tag{2}$$

where $r(\cdot)$ is the evenness score function, x is the feature representation of an input malware, δ is the adversarial perturbations for x, $f_s(\cdot)$ is the *substitute* model from the attacker, t is the targeted class, and F^{n_d} denotes all valid input feature vectors. The crafted AE x' is $x' = x + \delta$. An attacker typically aims to generate x' which is still malware but classified by $f_s(\cdot)$ as a benign application. We denote malware with label 1, then the target class $t = 0$.

Due to the nonlinear property of $f_s(x + \delta) = t$, Formula 2 is not solvable. We follow the approach in [27] and replace $f_s(x + \delta) = t$ by $g(x + \delta) \leq 0$ where $g(x)$ is defined as:

$$g(x) = max\{0.5 - f_s(x), 0\}. \tag{3}$$

Then the formulation of our optimization problem in Formula 2 is stated as follows:

$$\begin{aligned} \underset{\delta}{\text{minimize}} \quad & -r(x+\delta), \\ \text{subject to} \quad & g(x+\delta) \leq 0, \\ & x+\delta \in F^{n_d}. \end{aligned} \tag{4}$$

Moreover, we further modify the objective function as follows by Lagrangian relaxation:

$$\begin{aligned} \underset{\delta}{\text{minimize}} \quad & -r(x+\delta) + cg(x+\delta), \\ \text{subject to} \quad & x+\delta \in F^{n_d}, \end{aligned} \tag{5}$$

where $c \leq 0$ is a penalty constant and a hyper-parameter. Different c can lead to quite different δ thus x'. In practice, we conduct a grid search to find a better c. Since we remove the constraint in Formula 4, the minimum of objective in formula 4 will be smaller. Then we add a penal term in Formula 5 to penalize the original objective function. There exists a c such that the optimal solution to the objective in Formula 5 is equal to the optimal solution of the objective in Formula 4 with the constraints.

D. Solving the Objective Function

Here we focus on solving Equation 5 covering how to find c and handle the constraints.

Finding the constant c. As mentioned above, we use a grid search to find a proper c for Formula 5. Specifically, we start from 0.01 to 1000. For each c, we then use stochastic gradient descent (SGD) to obtain δ_p, i.e., the optimal δ. Note that it is possible that we cannot obtain δ_p for some x when solving Equation 5. If c exceeds the threshold (i.e., 1000) and still cannot meet the requirements, we will abort the search and fail to generate an AE for the x.

Box constraint. As introduced in Section V-A1, x_i for an Android apk corresponds to whether or not a specific permission or API call is declared in the corresponding *AndroidManifest.xml* file. As a result, x_i is either 0 (i.e., not declared) or 1 (i.e., declared). So we replace the constraint in formula 5 as:

$$x + \delta \in \{0, 1\}^{n_d}, \tag{6}$$

which is called as "box constrait". Following [27], we set another form constrait in Equation 5:

$$\delta_i = \frac{1}{2}(\tanh(w_i) + 1) - x_i, \tag{7}$$

where w_i is the replacement variable for the ith dimension of optimization variable δ_i. Since $0 < \frac{1}{2}(tanh(w_i) + 1) < 1$, the new modified vector $\delta_i + x_i$ would always fall into valid range. Instead of optimizing δ directly, we modify the input vector x on variable w of function $tanh$ and valid the box constraints automatically.

L_∞ and L_1 adjustment. Based on the optimization formulation and evenness score function defined above, the final objective is:

$$\underset{\delta}{\text{minimize}} \quad cg(x + \delta) - \frac{||I||_1}{n_d \cdot ||I||_\infty}. \tag{8}$$

We normalize the $||I||_1$ each iteration to further simplify the optimization formulation. We observe that directly solving Formula 8 does not generate effective AEs. One possible reason is that it is hard to realize the gradient search on component $||I||_\infty$. The infinity norm $||I||_\infty$ is solely related to the dimension with the largest value, so it penalizes only one dimension at one time. It is possible that the value of I_j increases when we penalize I_i. Thus, a few dimensions may change back and forth without any optimization progress in gradient descent. We aim to obtain the gradient descent direction toward the decrease of all dimensions with a "large value". As a remedy, we modify Formula 8 as follows.

$$\underset{\delta}{\text{minimize}} \quad cg(x + \delta) + \frac{\sum_i [I_i - \lambda]}{n_d}. \tag{9}$$

where λ is a hyper-parameter used to penalize all $I_i \geq \lambda$ to avoid only penalizing one dimension at one iteration. We also introduce a decay factor, $\gamma \leq 1$, to λ, i.e., $\lambda_{k+1} = \lambda_k^\gamma$ so that the algorithm could optimize I_i gradually and smoothly. k denotes the k-th iteration round.

We summarize how to generate an AE in our transfer attack and illustrate it in Algorithm 1.

V. Performance Evaluation

A. Experimental Setup

1) Dataset and Data Processing: We use Drebin dataset, one of the most popular datasets for evaluating malware classifiers [28], for evaluation. The dataset includes SHA256 values of 129,013 android applications, of which 123,453 are benign, and 5,560 are malicious. Based on the given SHA256 value, we collect the APK files from the APK markets, including VirusTotal for the malware APKs and GooglePlay store or AppChina for benign APKs. We follow the framework of [28] to statically analyze Android applications.

Algorithm 1 The Proposed Algorithm

```
Input: training dataset $D_{train}$, validation dataset $D_v$, testing
    set $D_{test}$, maximum threshold $c_{max}$, minimum threshold
    $c_{min}$, penalizer $\lambda$, decay factor $\gamma$, malware example $\hat{x}$ to
    be modified
Output: AE $\hat{x}'$
$f_s(\cdot) \leftarrow train\_model(D_{train}, D_v, D_{test})$ # Train the substitute model
$\hat{x}' \leftarrow \hat{x}$
$c \leftarrow c_{min}$
while $c < c_{max}$ do
    $r \leftarrow contribution\_compute(f_s(\cdot), \hat{x}')$ #Compute the evenness score
    $\delta \leftarrow gradient\_descent(f_s(\cdot), \hat{x}', \lambda, r)$ # Optimization step based on Equation 9
    $\hat{x}' \leftarrow \hat{x} + \delta$
    if $f_s(\hat{x}') == 0$ then
        break # Successfully evade the substitute model
    end if
    $\lambda \leftarrow \lambda^\gamma$
end while
return $\hat{x}'$
```

TABLE II
Overview of the Drebin feature sets.

manifest		dexcode	
S_1	Hardware components	S_5	Restricted API calls
S_2	Requested permissions	S_6	Used permission
S_3	Application components	S_7	Suspicious API calls
S_4	Filtered intents	S_8	Network addresses

The dataset contains $545,333$ features, each is represented by a binary value that indicates whether the feature is present in an application. As shown in Table II, Drebin constructs 8 feature classes based on manifest and dexcode files. Subsets S_1 to S_4 are extracted from AndroidManifest.xml and S_5 to S_8 are extracted from Class.dex file. An APK file $z \in Z$ then will be mapped in to a feature vector $x \in X$ through a mapping function $Z \rightarrow X$, where $x = (x^1, ..., x^{n_d})^T \in X = \{0, 1\}^{n_d}$. Each dimension of x indicates whether this feature exists in the APK file z. We further select 10,000 ($n_d = 10,000$) features out of $545,333$ features with high frequencies as the input vector for ML models.

2) Victim and Substitute Model Training: According to the adversarial model in Sec. III-B, we train two models for each experiment: the victim model and the substitute model. Note that the attacker's knowledge is formed as the quadruple $\{D, X, \hat{f}, \hat{w}\}$. For the training set D, We train the victim model and the substitute model using the full training set of Drebin. We assume the attacker and the victim use the same feature representation X.

For each experiment, we choose one victim model and one substitute model from {Support Vector Machine(SVM), Logistics Regression(LR), Ridge Regression(RR), Neural Network(NN), Ensemble Model(EM)}. For each substitute model and the victim model, we end the training process when the validation accuracy no longer improves.

B. Evaluation Metric

We use the transferability of AEs as the major metric. Attackers generate AEs on the substitute model and examine these generated AEs on the victim model. It is evaluated by the transferability:

$$T_{SR} = \frac{N_v}{N_s}, \tag{10}$$

where N_s is the number of generated AEs that successfully evade the substitute model and N_v is the number of generated AEs evade the attacker's substitute model $f_s(\cdot)$ and also successfully evade the targeted victim model $f_v(\cdot)$.

C. Benchmarks

We use three algorithms as the benchmarks, including original FGSM [1] and FGSM attack with two transferability enhance techniques, i.e., ensemble-sample (ES) and ensemble-model (EM). We briefly introduce them here.

1) FGSM: Fast Gradient Sign Attack attack(FGSM) [1] performs only one gradient update in the direction of the gradient sign at each pixel. The perturbation form is:

$$x' = x + \alpha sign(\nabla_x J_\theta(x, y)),$$

where α is the magnitude of the perturbation and J_θ is the cost function for the ML model.

2) Ensemble Sample: ES is basically from the data augmentation idea. ES ensembles multiple samples together at first or in the middle process of generating AE to enhance the transferability of generated AE. Dong et al. [17] propose an ES method, namely Translation-Invariant Attack. ES method generates AEs by utilizing a set of translated samples from the original sample:

$$\begin{aligned} \underset{\delta}{\text{argmin}} \quad & \sum_{i,j} w_{i,j} J_\theta(T_{i,j}(x+\delta), y^*), \\ \text{subject to} \quad & D(x, x+\delta) < \varepsilon \\ & x+\delta \in [0,1]^{n_d}, \end{aligned}$$

where $T_{ij}(x)$ is a translation operation that shifts the image x along two dimensions of i and j pixels respectively. ES can generate AEs that are less sensitive to the discriminative regions of the attacker's substitute model. In malware, we adapt it by randomly flipping a specific number of 0 to 1 in the feature space as the translation operation.

3) Ensemble Model: EM [19] is another major technique in the computer vision domain to improve AE transferability. The idea is to integrate multiple models to generate transferable AEs. Given a set of models $\{f_n(x)\}$ and a set of weights $\{a_n\}$, an attacker first compute an ensemble model $F(x) = \sum_{i=1}^{n} a_i f_i(x)$ which is the weighted average of the set of models. The attacker then generates AEs from $F(x)$. The EM method solves the following problem:

$$\begin{aligned} \underset{\delta}{\text{argmin}} \quad & -log((\sum_{i=1}^{n} a_i f_i(x+\delta)) \cdot \mathbf{1}_{y^*}) + \lambda D(x, x+\delta), \\ \text{subject to} \quad & x+\delta \in [0,1]^{n_d}, \end{aligned}$$

where a_i is the ensemble weight, $\sum_{i=1}^{n} a_i = 1$, and y^* is the target label which the attacker want to mislead.

D. Adapt ES and EM to Malware Data

An attacker generates an AE $x+\delta$ based on the original malware input vector x to evade the victim model. The generated AE must maintain the original application's integrity and malicious functionality. Therefore, we redefine the manipulation δ instead of using the default $x + \delta \in [0, 1]$.

We consider feature addition strategy (i.e., '0' to '1' strategy). In the feature addition strategy, attackers can flip the value in the input vector from '0' to '1', meaning injecting features in the original APK file, such as adding more permission requests in AndroidManifest.xml or another system call in Class.dex file. The '0' to '1' change is a much safer manipulation compared with the '1' to '0' change. Generally, adding more features to the original sample at AndroidManifest.xml would not affect its functionality(e.g., more permission requests). The '0' to '1' change at the Class.dex file means adding some dead code that has never been called or executed, which is also safe.

Further, following the work in [4] we pre-define a modifiable feature set. And modifications on features in this set would not harm the functionality and the executability. We gradually increase the percentage of modifiable features in the modification restriction set ϵ from 0.01 to 0.50 and observe the performance.

E. Evaluation Results

We examine the effectiveness of transfer-based AE attacks on malware in different aspects, including the substitute models, the size of the modifiable features set, and the transferability enhancement techniques ES and EM. Finally, we investigate the proposed methodology. We give the experimental results and analysis as follows:

1) Evaluate FGSM in Malware Classifiers: We first train the victim model (i.e., a neural network (NN) with three hidden layers and 100 neurons at each layer, denoted as NN(3,100)). We train multiple substitute models to simulate a grey-box scenario, including SVM, LR, RR, and NN. The substitute NN is with two layers and 100 neurons at each layer, denoted as NN(2,100).

Table III shows the experimental results of the FGSM attack. The attack transferability increases as the modification restriction get looser (ϵ gets higher), indicating AEs can successfully evade the malware detection system with a higher probability when attackers are given permission to modify more features. And the transferability stops increasing after ϵ hits a certain level.

The transferability is also impacted by substitute models. As Table III shows, NN(2,100) achieves the fastest-growing transferability performance while the other three ML models (SVM,

TABLE III
AE TRANSFERABILITY OF FGSM.

Different Substitute models	Victim model(NN(3,100))								
	$\epsilon = 0.01$	$\epsilon = 0.02$	$\epsilon = 0.03$	$\epsilon = 0.04$	$\epsilon = 0.05$	$\epsilon = 0.06$	$\epsilon = 0.10$	$\epsilon = 0.20$	$\epsilon = 0.50$
SVM	4%	27%	73%	91%	91%	91%	91%	91%	91%
LR	5%	46%	84%	96%	96%	96%	96%	96%	96%
RR	5%	46%	84%	96%	96%	96%	96%	96%	96%
NN(2,100)	27%	68%	93%	94%	94%	94%	94%	94%	94%

TABLE IV
AE TRANSFERABILITY FGSM WITH ES ENHANCEMENT.

Different Substitute models	Victim model(NN(3,100))								
	$\epsilon = 0.01$	$\epsilon = 0.02$	$\epsilon = 0.03$	$\epsilon = 0.04$	$\epsilon = 0.05$	$\epsilon = 0.06$	$\epsilon = 0.10$	$\epsilon = 0.20$	$\epsilon = 0.50$
SVM	4%	28%	74%	92%	92%	92%	92%	92%	92%
LR	5%	46%	84%	96%	96%	96%	96%	96%	96%
RR	5%	46%	84%	96%	96%	96%	96%	96%	96%
NN(2,100)	28%	68%	93%	94%	94%	94%	94%	94%	94%

TABLE V
AE TRANSFERABILITY OF FGSM WITH EM ENHANCEMENT.

Different Victim models	Corresponding attacker's EM	Modification restrictions					
		$\epsilon = 0.01$	$\epsilon = 0.02$	$\epsilon = 0.03$	$\epsilon = 0.04$	$\epsilon = 0.05$	$\epsilon = 0.06$
NN	SVM-LR-RR	6%	44%	83%	95%	96%	96%
	SVM-NN-LR-RR	0%	1%	6%	17%	55%	55%
SVM	NN-LR-RR	8%	9%	16%	16%	17%	17%
	SVM-NN-LR-RR	8%	9%	16%	17%	19%	19%
LR	SVM-NN-RR	6%	9%	13%	21%	24%	24%
	SVM-NN-LR-RR	1%	5%	23%	30%	39%	40%
RR	SVM-NN-LR	3%	5%	11%	11%	12%	12%
	SVM-NN-LR-RR	2%	3%	16%	21%	30%	30%

TABLE VI
AE TRANSFERABILITY OF *BATE*

Different Substitute models	Victim model(NN(3,100))								
	$\epsilon = 0.01$	$\epsilon = 0.02$	$\epsilon = 0.03$	$\epsilon = 0.04$	$\epsilon = 0.05$	$\epsilon = 0.06$	$\epsilon = 0.10$	$\epsilon = 0.20$	$\epsilon = 0.50$
SVM	0%	24%	47%	68%	76%	81%	90%	95%	95%
LR	1%	33%	58%	75%	88%	92%	97%	97%	98%
RR	1%	33%	58%	75%	88%	92%	97%	97%	98%
NN(2,100)	14%	43%	72%	89%	95%	96%	98%	98%	98%

LR, and RR) grow much slower to achieve their maximum transferability value. Using NN(2,100) as the substitute model gets much better transferability in the lower ϵ range (0.01 to 0.03 in this experiment). This is due to the very similar structure between the substitute model NN(2,100) and the victim model NN(3,100). Note that we tried several NNs with a different number of layers and neurons at each layer and found they have a similar experimental trend as NN(2,100). So here, we only show NN(2,100) in the results.

We also validate the previous criterion [29] that lower-complexity models (with stronger regularization) provide better substitute models. As the ϵ increases, SVM LR and RR also achieve higher than 90% transferability. SVM is designed to generate more complex decision boundaries [30] than LR and RR. LR and RR consistently outperform SVM in all ϵ settings.

2) Evaluate ES in Malware Classifier: Table IV shows the results of the FGSM method enhanced by ES technique. Compared with Table III, we can see that ES only has minor improvements for a few settings, such as SVM($\epsilon = 0.02$ to $\epsilon = 0.50$) and NN(2,200)($\epsilon = 0.01$). ES does not improve the AE transferability significantly.

3) Evaluate EM in Malware Classifier: We designed multiple victim models and substitute models to examine how the two settings impact the AE transferability of EM methods.

We use NN(3,100) as the victim model and SVM-LR-RR as the ensemble substitute model. As shown in Table V, the SVM-LR-RR achieves as high as 96% transferability when $\epsilon = 0.05$, which slightly surpasses baseline FGSM and ES FGSM at the same perturbation level (i.e., the same ϵ). When we ensemble one more model NN(3,100) (the same structure as the victim model) and get SVM-NN-LR-RR, the transferability gets even lower than SVM-LR-RR in every ϵ setting. The reason can be that the substitute model becomes more complicated after adding NN. Generating AE on a complicated model but attacking a rather simple model would result in lower transferability. It is also consistent with our previous argument in baseline FGSM experimental results.

We have interesting findings by using various victim models. When using SVM as the victim model, NN-LR-RR achieves only 17% transferability at $\epsilon = 0.06$. It is because the substitute model is complicated (i.e., NN is included). For the LR and RR victim models, we get similar results. When we ensemble one more model the same as the victim model and get SVM-NN-LR-RR, we can see the upper bound of transferability increased to 19%. We get a similar increase for the LR victim and RR victim models. However, ensembling NN into SVM-LR-RR will harm the transferability. This is because NN increases the complexity of SVM-LR-RR. Therefore, ensembling one more model into the substitute model would be beneficial for enhancing AE transferability if the added model does not increase the complexity of the current substitute ensemble models.

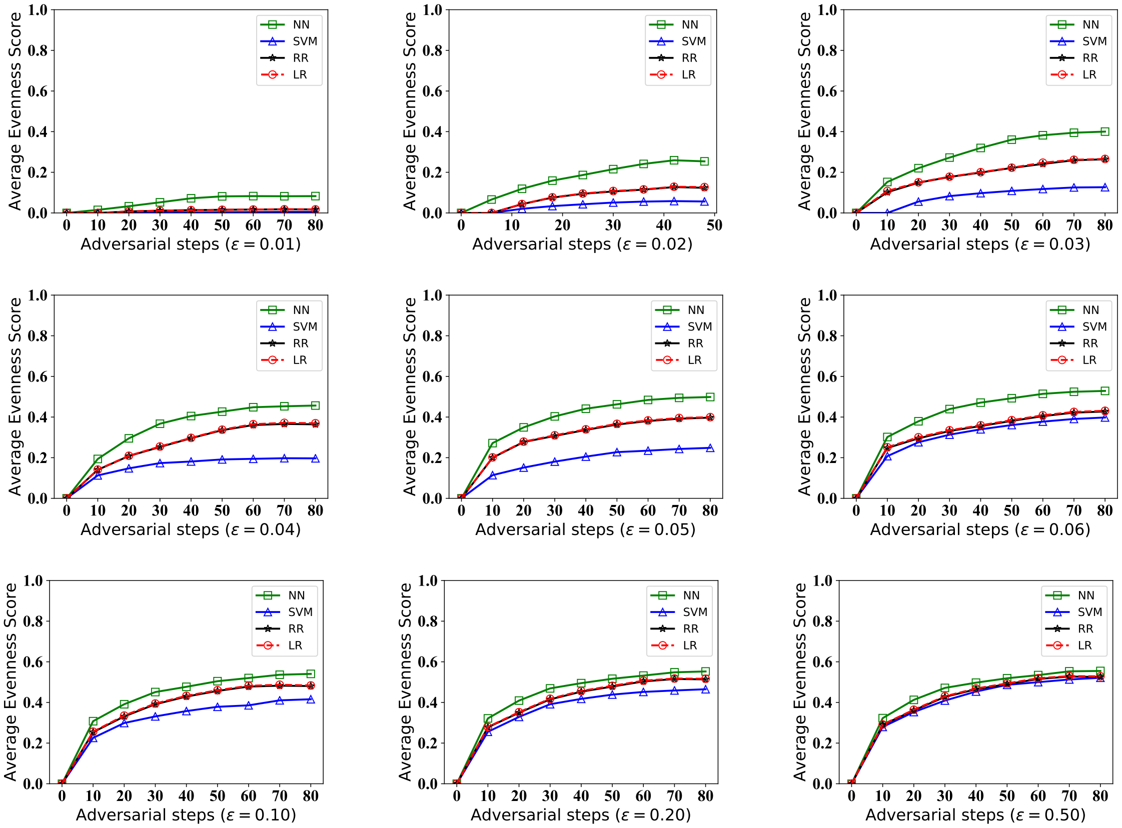

Fig. 2. Evenness scores as the adversarial step increases.

Based on the results above, we conclude that the transferability performance of EM is unstable. With a small ϵ, simpler models would provide higher transferability. At larger ϵ settings, transferability is highly relevant to the model complexity level between the victim model f_v and the substitute model f_s. For the attacker, a simpler substitute model (Or ensemble of very simple models) would be a safer choice.

4) Evaluate BATE in Malware Classifier: In this experiment, we use the same settings as the FGSM and ES experiments. Table VI shows transferability of *BATE*. The transferability increases with ϵ and exceeds the maximum value of the previous baseline methods though the transferability at small ϵ is relatively lower than baselines. Specifically, when the ϵ is up to certain values, the transferability will reach the peak, which excels FGSM and EM baseline by 2% to 4%. The more modifiable features, the higher evenness score our method could optimize, thus the higher transferability results the attacker could get.

Is the evenness score optimized?

To further validate *BATE* works and whether the evenness score gets optimized, we examine the average evenness score change as the adversarial steps increase in the gradient descent process. We plot the average evenness score over 100 testing samples when proceeding with optimization in different ϵ settings from 0.01 to 0.5 in Figure 2. When the ϵ is in small values (e.g., $\epsilon = 0.01$), average evenness scores remain constant as the adversarial steps increase since the optimization space is limited. As the ϵ gets large, the substitute model NN has the fastest growth rate in evenness score. The LR is slightly higher than RR, while the SVM is the slowest one. This phenomenon is because the NN has more nonlinear relationships in model space and could get overfitted more easily. Therefore, it has more potential to be optimized even in small ϵ. Also, since LR has no regularization penalty as in RR, it will get overfitted more easily than RR.

Even though the model non-linearity affects evenness score optimization space, it would be a minor factor if the ϵ value is large enough. From Figure 2, we can observe that the evenness scores are very close in all models when $\epsilon = 0.50$.

What is the relationship between Evenness Score and Transferability?

Here we investigate the relationship between the Evenness Score and the transferability to examine our intuition further. To investigate the statistical significance of evenness score to AE transferability, we compute the associated correlation values with three metrics, including Pearson value, Spearman Rank value, and Kendall's Tau value. The correlation is shown in Table VII. We observed that the hypothesis that the evenness score and the transferability have no correlation failed. We obtained the test value from the transferability and the average evenness score of 100 testing samples in different ϵ settings (0.01 to 0.5). All the p-values are smaller than 10^{-5}, which confirms 99% statistical significance.

TABLE VII
CORRELATION BETWEEN TRANSFERABILITY RATE AND EVENNESS SCORE. EACH RESULT IS OBTAINED FROM 100 TESTING SAMPLES.

		coefficients value	p-value
SVM	Pearson	0.68	1e-5
	Spearman Rank	0.65	1e-5
	Kendall's Tau	0.48	1e-5
LR	Pearson	0.75	1e-5
	Spearman Rank	0.72	1e-5
	Kendall's Tau	0.51	1e-5
RR	Pearson	0.75	1e-5
	Spearman Rank	0.72	1e-5
	Kendall's Tau	0.51	1e-5
NN(2,200)	Pearson	0.82	1e-5
	Spearman Rank	0.94	1e-5
	Kendall's Tau	0.81	1e-5

VI. Conclusion and Future Work

We have studied how transferability enhancement techniques ES and EM perform in the malware detection domain. We propose a novel formulation for transfer-based AE attacks based on the contribution of each feature toward the prediction result. Comprehensive experiments show that ES and EM have no apparent improvement in transferability, while our proposed methodology performs better in large modifiable feature sets due to the larger evenness score optimization space. From the attacker's viewpoint, simpler models provide a better substitute for transfer-based AE attacks. We hope this paper will inspire more research into the context of adversarial malware detection.

VII. Acknowledgment

This work was supported in part by the Office of Naval Research under grant N00014-19-1-2621, the US National Science Foundation under grants CNS-1837519 and CNS-1916902, the Army Research Office under grant W911NF-20-1-0141, and the Virginia Commonwealth Cyber Initiative (CCI).

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *ICLR'15*, 2015.

[2] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[3] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[4] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," *arXiv preprint arXiv:1606.04435*, 2016.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[6] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[7] I. Rosenberg, A. Shabtai, L. Rokach, and Y. Elovici, "Generic black-box end-to-end attack against state of the art api call based malware classifiers," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 490–510.

[8] W. Hu and Y. Tan, "Black-box attacks against rnn based malware detection algorithms," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[9] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth, "Learning to evade static pe machine learning malware models via reinforcement learning," *arXiv preprint arXiv:1801.08917*, 2018.

[10] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *2018 26th European signal processing conference (EUSIPCO)*. IEEE, 2018, pp. 533–537.

[11] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, and J. Keshet, "Deceiving end-to-end deep learning malware detectors using adversarial examples," *arXiv preprint arXiv:1802.04528*, 2018.

[12] H. S. Anderson and P. Roth, "Ember: an open dataset for training static pe malware machine learning models," *arXiv preprint arXiv:1804.04637*, 2018.

[13] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proceedings of the 2016 network and distributed systems symposium*, vol. 10, 2016.

[14] W. Song, X. Li, S. Afroz, D. Garg, D. Kuznetsov, and H. Yin, "Mab-malware: A reinforcement learning framework for attacking static malware classifiers," *arXiv preprint arXiv:2003.03100*, 2020.

[15] W. Wu, Y. Su, M. R. Lyu, and I. King, "Improving the transferability of adversarial samples with adversarial transformations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9024–9033.

[16] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2730–2739.

[17] Y. Dong, T. Pang, H. Su, and J. Zhu, "Evading defenses to transferable adversarial examples by translation-invariant attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4312–4321.

[18] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193.

[19] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[20] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," *arXiv preprint arXiv:1704.03453*, 2017.

[21] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren, "Android hiv: A study of repackaging malware for evading machine-learning detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 987–1001, 2019.

[22] O. Suciu, S. E. Coull, and J. Johns, "Exploring adversarial examples in malware detection," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 8–14.

[23] A. Kołcz and C. H. Teo, "Feature weighting for improved classifier robustness," in *CEAS'09: sixth conference on email and anti-spam*. Citeseer, 2009.

[24] M. Melis, M. Scalas, A. Demontis, D. Maiorca, B. Biggio, G. Giacinto, and F. Roli, "Do gradient-based explanations tell anything about adversarial robustness to android malware?" *International Journal of Machine Learning and Cybernetics*, vol. 13, no. 1, pp. 217–232, 2022.

[25] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! a case study on android malware detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 4, pp. 711–724, 2017.

[26] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *arXiv preprint arXiv:1605.01713*, 2016.

[27] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[28] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in *Ndss*, vol. 14, 2014, pp. 23–26.

[29] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, and F. Roli, "Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks," in *28th USENIX security symposium (USENIX security 19)*, 2019, pp. 321–338.

[30] N. Pochet and J. Suykens, "Support vector machines versus logistic regression: improving prospective performance in clinical decision-making," *Ultrasound in Obstetrics and Gynecology: The Official Journal of the International Society of Ultrasound in Obstetrics and Gynecology*, vol. 27, no. 6, pp. 607–608, 2006.