Response to all reviewers' comments

Manuscript ID: 1933

Title: Automated Malware Assembly Line: Uniting Piggybacking and Adversarial Example in Android Malware Generation

---

We would like to express our gratitude to the reviewers for their time spent on the review of our manuscript. We have addressed all the comments raised by the reviewers, and will revise our manuscript accordingly.

For convenience, we provide a jumping index for a smoother reviewing experience. Please kindly click the interested section and navigate to the corresponding response, if it would be helpful. In addition, to save the reviewers' time, **we have highlighted the most important comments with a light red box**.

Reviewer A Reviewer B

I Response to Reviewer: A

We would like to express our sincere gratitude to reviewer A for providing invaluable comments that are helpful for improving our work. We have provided a detailed response to each of the reviewer's comments.

Q1: [Detailed explanation] Algorithm 1 could be improved with more detailed explanations of the selection of the poorest perturbation pk in the fine-tuning phase.

Response:

Thank you for your comment. In the fine-tuning phase, the algorithm will modify the perturbation p based on the query-reply results with the target model to enhance the universality of p. Specifically, we will iterate through all perturbations in p to identify the poorest perturbation p_k that, when removed, results in the best universality of p:

$$p_k{}^* = \underset{p_k \in p}{\operatorname{argmin}} E_1\left(p - p_k\right) - E_1\left(p\right),$$

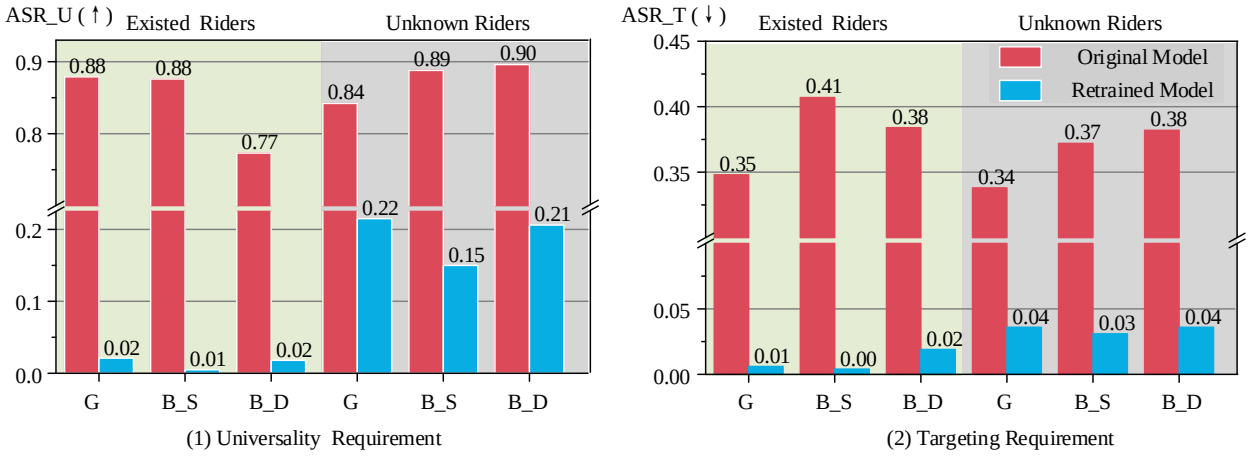

Fig. 1. The attack performance on retrained model.

where $E_1 = E_{x_b \in B}[F(x_b + r + p)]$ represents the expected value of the adversarial piggybacked malware classified by the model F. The adversarial piggybacked malware is generated by the malicious rider r and perturbation p applied to benign samples b. It is noteworthy that a smaller value of this expectation indicates that the classification model is more likely to classify the adversarial piggybacked malware as benign.

Then we delete ${p_k}^*$ from perturbation$p$: $p = p - {p_k}^*$. We will repeat the above process until the maximum allowable number of iterations is reached.

Q2:[Potential defense] Discuss potential defense mechanisms and their effectiveness against the proposed attack.

Response:

We want to express our gratitude for the reviewer's comment. In our original manuscript, we have provided potential defense methods for the app market and Android users in the Discussion section. Additionally, according to the reviewer's comment we will discuss another adversarial example defense method in the revised manuscript. This newly added defense method is based on the assumption that the app market can utilize similarity analysis to identify a limited number of adversarial piggybacked malware and thus can enhance the performance of its classification model through adversarial retraining. In this context, we consider a rather extreme scenario where users acquire 100 instances of malicious adversarial piggybacked malware for each malicious rider and incorporate them into the training set for model retraining.

The attack results are illustrated in Fig. 1. Subfigures (1) and (2) in the figure represent the

attack success rates ASR_U and ASR_T on matched and unmatched riders [1], respectively, where a higher value for the former is preferable and a lower value for the latter is desirable. In each subfigure, the red bars correspond to the attack results on the original model, while the blue bars represent the attack results on the adversarially retrained model. The bars with a green background denote the attack results on existing riders, and those with a gray background reflects the results on unknown riders [2]. The experimental results demonstrate that adversarial retraining can mitigate the attack effectiveness of adversarial piggybacked malware, reducing the attack success rate from 85.9% to 10.5%. However, this defense strategy has two significant drawbacks. The first is the difficulty of obtaining a substantial amount of adversarial piggybacked malware for training in real-world scenarios. The second is that the model, strengthened by adversarial retraining, tends to overfit to the adversarial piggybacked malware present in the training set. As shown in Subgraph (1), the defense effectiveness of the model against adversarial piggybacked malware generated from Unknown Riders is 17.6% lower than that against malware derived from Existed Riders. In the future, we will explore more robust malware detection models.

Q3: [Ethical implications] Discussion of the ethical implications of the research.

Response:

Thank you for the comment. In fact, we have already addressed the Ethical Concerns in Section VII of our manuscript. Our ethics statement is as follows: "Our work falls within the realm of offensive research. The main goal of our work is to make the academic and industrial communities pay more attention to adversarial example attacks against Android malware detection systems. Our work demonstrates that adversaries can generate a large number of evasive malware with hook technology. In the sections of Potential Defense and Appendix, we have deliberated on potential defense mechanisms against our attack method. We ensure that the relevant technology will only be utilized for academic research purposes."

It may be because our ethics statement is in the subsection, making it not very easy to find. In order to make it more prominent, we plan to include it in a separate section in the revised manuscript.

Q4: [Formal proofs] Present formal proofs to demonstrate the effectiveness of the proposed method (since the proposed approach mostly relies on experimental evaluation).

[1]To assist the reviewers in recalling the definitions, we present the following clarifications: Matched rider refers to the rider that requires the attacker to employ the perturbation p to conceal malicious activities. In contrast, unmatched riders denote other riders that are not designed by the attacker. The perturbation p will be ineffective on unmatched riders.

[2]For the developers of malware detection models, there exist certain malware composed of previously identified riders in their dataset, which are referred to as Existed Riders. Conversely, malware that does not include any malicious riders present in the detection model's training set is termed Unknown Riders.

Response: Thank you for the comments. The convergence proof of the algorithm for solving the min-max problem has been extensively discussed in several studies [1][2][3]. To facilitate the reviewers' understanding, we have provided an outline of the proof process and detailed steps as follows:

Our proof strategy is as follows: 1) First, due to the discrete nature of variables in the field of Android malware detection, which presents significant challenges for the proof process, we will approach the convergence proof under the assumption of continuous variables. 2) Next, we will establish Lemma 1, which analyzes the Basic Iterate Relations derived during the algorithm's iterative process. 3) Subsequently, we will introduce Assumption 1, which posits that the gradients during the solving process are bounded. 4) Following this, we will present Lemma 2, demonstrating the relationship between the iterate averages of the obtained solutions and all feasible solutions within the solution space. 5) Finally, we will substitute the saddle point (i.e., the optimal solution) into Lemma 2 to derive the convergence of the solution obtained by the algorithm. The detailed proof is shown as follows.

For the sake of convenience, we denote the loss as $\mathcal{L}$, i.e., $\mathcal{L} = -E_{x_b \in B}(F(x_b + (r+n) + p))$. Our goal is to solve the following saddle point problem:

$$\min_{p} \max_{n} \mathcal{L}(p, n), \tag{1}$$

If r and p are continuous values, we can solve the aforementioned problem using the following gradient descent ascent (GDA) algorithm.

$$\begin{aligned} p_{k+1} &= \mathcal{P}_p \left[p_k - \alpha \mathcal{L}_r \left(p_k, n_k \right) \right], \quad \text{for } k = 0, 1, \ldots \\ n_{k+1} &= \mathcal{P}_n \left[n_k + \alpha \mathcal{L}_n \left(p_k, n_k \right) \right], \quad \text{for } k = 0, 1, \ldots \end{aligned} \tag{2}$$

where P_P and P_N denote the projections onto the sets P and N, respectively. The vectors $p_0 \in P$ and $n_0 \in N$ are the initial iterates, and the scalar $\alpha > 0$ is a constant step size. The vectors $L_p(p_k, n_k)$ and $L_n(p_k, n_k)$ represent the subgradients of L at (p_k, n_k) with respect to p and n, respectively.

However, in our work, r and p are discrete. Therefore, we use $\frac{E_2(n+c,p)-E_2(n,p)}{\text{len}(c)}$ and $\frac{E_2(n,p+c)-E_2(n,p)}{\text{len}(c)}$ to replace $L_p(p_k, n_k)$ and $L_n(p_k, n_k)$, respectively. For convenience, we will only analyze the convergence properties under the condition of continuous variables.

Under general assumptions, we consider $\mathcal{L}$ to be a convex-concave function. Specifically, $\mathcal{L}(\cdot, n)$ is convex for every $n \in \mathbb{N}$. $\mathcal{L}(\cdot, n)$ is concave for every $p \in \mathbb{P}$.

Following the above assumptions and Equa. 2, we start with a Lemma.

Lemma 1. Let the sequences p_k and n_k be generated by the subgradient Equa. 2. We then have:

(a). For any $p \in P$ and for all $k \geq 0$,

$$\|p_{k+1}-p\|^2 \leq \|p_k-p\|^2 - 2\alpha\left(\mathcal{L}\left(p_k,n_k\right)-\mathcal{L}\left(p,n_k\right)\right)+\alpha^2\left\|\mathcal{L}_p\left(p_k,n_k\right)\right\|^2 \tag{3}$$

(b). For any $n \in N$ and for all $k \geq 0$,

$$\|n_{k+1}-n\|^2 \leq \|n_k-n\|^2 + 2\alpha\left(\mathcal{L}\left(p_k,n_k\right)-\mathcal{L}\left(p_k,n\right)\right)+\alpha^2\left\|\mathcal{L}_n\left(p_k,n_k\right)\right\|^2 \tag{4}$$

Proofs (a) and (b) follow a similar proof process; here, we will only provide the proof outline for (a). Based on the nonexpansive property of the projection operation and Equation 2, for any $p \in P$ and all $k \geq 0$,

$$\begin{aligned}\|p_{k+1}-p\|^2 &= \left\|\mathcal{P}_P\left[p_k-\alpha\mathcal{L}_p\left(p_k,n_k\right)\right]-p\right\|^2 \\ &\leq \left\|p_k-\alpha\mathcal{L}_p\left(p_k,n_k\right)-p\right\|^2 \\ &= \|p_k-p\|^2 - 2\alpha\mathcal{L}_p'\left(p_k,n_k\right)\left(p_k-p\right)+\alpha^2\left\|\mathcal{L}_p\left(p_k,n_k\right)\right\|^2\end{aligned} \tag{5}$$

Since the function $\mathcal{L}(p,n)$ is convex in p for each $n \in N$, we have that, for any p,

$$-\mathcal{L}_p'\left(x_p,n_k\right)\left(p_k-p\right) \leq -\left(\mathcal{L}\left(p_k,n_k\right)-\mathcal{L}\left(p,n_k\right)\right). \tag{6}$$

Hence, for any $p \in P$ and all $k \geq 0$,

$$\|p_{k+1}-p\|^2 \leq \|p_k-p\|^2 - 2\alpha\left(\mathcal{L}\left(p_k,n_k\right)-\mathcal{L}\left(p,n_k\right)\right)+\alpha^2\left\|\mathcal{L}_p\left(p_k,n_k\right)\right\|^2. \tag{7}$$

Assumption 1. The $\mathcal{L}_p\left(p_k,n_k\right)$ and $\mathcal{L}_n\left(p_k,n_k\right)$ used in the method defined by Equa. 2 are uniformly bounded, i.e., there is a constant $L>0$ such that

$$\left\|\mathcal{L}_p\left(p_k,n_k\right)\right\| \leq L, \quad \left\|\mathcal{L}_n\left(p_k,n_k\right)\right\| \leq L, \quad \text{for all } k \geq 0. \tag{8}$$

Under the preceding assumption and analyses, we have the second lemma.

Lemma 2. Let the sequences p_k and x_k be generated by the Equa. 2. $\hat{p}_k$ and $\hat{n}_k$ are the iterate averages given by:

$$\hat{p}_k = \frac{1}{k}\sum_{i=0}^{k-1} p_i, \quad \hat{n}_k = \frac{1}{k}\sum_{i=0}^{k-1} n_i \tag{9}$$

We then have, for all $k \geq 1$,

$$\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p_i,n_i\right)-\mathcal{L}\left(p,\hat{n}_k\right)\leq\frac{\left\|p_0-p\right\|^2}{2\alpha k}+\frac{\alpha L^2}{2},\quad\text{for any } p\in P \tag{10}$$

$$-\frac{\left\|n_0-n\right\|^2}{2\alpha k}-\frac{\alpha L^2}{2}\leq\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p_i,n_i\right)-\mathcal{L}\left(\hat{p}_k,n\right),\quad\text{for any } n\in N. \tag{11}$$

Since the proof processes for Equation 10 and Equation 11 are similar, we will only provide the proof for Equation 10 here. By using Lemma 1 and the boundedness of the $\mathcal{L}_p\left(p_i,n_i\right)$, we have that, for any $p\in P$ and $i\geq 0$

$$\left\|p_{i+1}-p\right\|^2\leq\left\|p_i-p\right\|^2-2\alpha\left(\mathcal{L}\left(p_i,n_i\right)-\mathcal{L}\left(p,n_i\right)\right)+\alpha^2L^2. \tag{12}$$

Therefore,

$$\mathcal{L}\left(p_i,n_i\right)-\mathcal{L}\left(p,n_i\right)\leq\frac{1}{2\alpha}\left(\left\|p_i-p\right\|^2-\left\|p_{i+1}-p\right\|^2\right)+\frac{\alpha L^2}{2} \tag{13}$$

By adding these relations over $i=0,\ldots,k-1$, we obtain, for any $p\in P$ and $k\geq 1$,

$$\sum_{i=0}^{k-1}\left(\mathcal{L}\left(p_i,n_i\right)-\mathcal{L}\left(p,n_i\right)\right)\leq\frac{1}{2\alpha}\left(\left\|p_0-p\right\|^2-\left\|p_k-p\right\|^2\right)+\frac{k\alpha L^2}{2} \tag{14}$$

implying that

$$\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p_i,n_i\right)-\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p,n_i\right)\leq\frac{\left\|p_0-p\right\|^2}{2\alpha k}+\frac{\alpha L^2}{2} \tag{15}$$

Since the function $\mathcal{L}(p,n)$ is concave in n for any fixed $p\in P$, there holds

$$\mathcal{L}\left(p,\hat{n}_k\right)\geq\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p,n_i\right),\quad\text{with } p\in P \text{ and } \hat{n}_k=\frac{1}{k}\sum_{i=0}^{k-1}n_i. \tag{16}$$

Combining the preceding two relations, we obtain that, for any $p\in P$ and $k\geq 1$,

$$\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p_i,n_i\right)-\mathcal{L}\left(p,\hat{n}_k\right)\leq\frac{\left\|p_0-p\right\|^2}{2\alpha k}+\frac{\alpha L^2}{2} \tag{17}$$

According to the above analyses, we can have following conclusion.

Conclusion 1. Let $(p^*,n^*)\in P\times N$ be a saddle point of $\mathcal{L}(p,n)$. We have:

$$-\frac{\|n_0-n^*\|^2}{2\alpha k}-\frac{\alpha L^2}{2}\leq\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p_i,n_i\right)-\mathcal{L}\left(p^*,n^*\right)\leq\frac{\|p_0-p^*\|^2}{2\alpha k}+\frac{\alpha L^2}{2} \tag{18}$$

The proof is given as follows. Based on Lemma 2, by letting $p=p^*$ and $n=n^*$, we have for all $k\geq 1$:

$$\begin{aligned}&\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p_i,n_i\right)-\mathcal{L}\left(p^*,\hat{n}_k\right)\leq\frac{\|p_0-p^*\|^2}{2\alpha k}+\frac{\alpha L^2}{2}\\&-\frac{\|n_0-n^*\|^2}{2\alpha k}-\frac{\alpha L^2}{2}\leq\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p_i,n_i\right)-\mathcal{L}\left(\hat{p}_k,n^*\right)\end{aligned} \tag{19}$$

By the saddle-point relation, we have

$$\mathcal{L}\left(p^*,\hat{n}_k\right)\leq\mathcal{L}\left(p^*,n^*\right)\leq\mathcal{L}\left(\hat{p}_k,n^*\right) \tag{20}$$

Combining the preceding relations, we obtain that, for all $k\geq 1$:

$$-\frac{\|n_0-n^*\|^2}{2\alpha k}-\frac{\alpha L^2}{2}\leq\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p_i,n_i\right)-\mathcal{L}\left(p^*,n^*\right)\leq\frac{\|p_0-p^*\|^2}{2\alpha k}+\frac{\alpha L^2}{2}. \tag{21}$$

These results show that averaged function values $\frac{1}{k}\sum_{i=0}^{k-1}\mathcal{L}\left(p_i,n_i\right)$ converges to the saddle point value within an error level of $\frac{\alpha L^2}{2}$.

[1] A Single-Loop Smoothed Gradient Descent-Ascent Algorithm for Nonconvex-Concave Min-Max Problems. NeurIPS 2020.

[2] On Gradient Descent Ascent for Nonconvex-Concave Minimax Problems. ICML 2020.

[3] Subgradient Methods for Saddle-Point Problems. J. Optimization Theory and Applications. 2009.

Q5: [Improve writing] Proofread writing so it does not hinder readability

Response: Thank you very much for the comment. We will thoroughly revise the wording and structure of our manuscript to improve readability.

II Response to Reviewer: B

Thank you very much for the positive comments. We will further improve our work according to your comments.

Q1: Explain or resolve the logic problem in the min-max problem designed for the targeting requirement.

Response:

We apologize for any misunderstanding caused by our inappropriate phrasing. The min-max problem is introduced to meet the targeting requirement, i.e., the perturbation is designed only for the adversary's own malicious rider. Solving this min-max problem involves first identifying the rider (i.e., r + n*) most susceptible to perturbation p and then optimizing p to ensure that it cannot attack this rider (r + n*) successfully.

In the minimization phase, we identify the rider (r + n*) that is most easily attacked. In the original manuscript, we stated, '... find an optimal $n*$, where the resulting rider $r + n*$ is very prone to being misclassified as benign.' What we intended to convey is that we aim to 'find the rider $r + n*$ that is most susceptible to perturbation by p and is more likely evade detection.'

In the maximization phase, we optimize the perturbation p to ensure that it cannot be successfully applied to the most vulnerable rider (r + n*). Therefore, it is reasonable to believe that the perturbation p is unlikely to be effective for other riders, fulfilling the targeting requirement.

Q2: Modify or design other attack methods suitable for this scenario and compare them with the method proposed in this paper.

Response:

We want to express our sincere gratitude to the reviewer for this valuable feedback. According to this feedback, we selected four attack algorithms for comparative analysis.

First, to validate the effectiveness of our adversarial perturbations, we designed a Random-Noise attack algorithm. This approach involves randomly generating an adversarial perturbation of the same size as the original input, with the goal of attack effectiveness evaluation.

Then, we selected two attack algorithms (i.e., HIV-CW and HIV-JSMA) proposed in the Android HIV framework [1] that specifically target Android malware detection systems. Since both algorithms generate adversarial perturbations for individual APKs, we focused on the adversarial perturbations associated with a particular malicious rider that results in piggybacked malware. We applied these perturbations to other instances of piggybacked malware generated by the same malicious rider to test whether the universality requirement is met. Furthermore, we tested the targeting requirement by applying the perturbations to piggybacked malware produced by different malicious riders.

Finally, we selected a universal attack method, MalPatch [2], which generates a universal perturbation for a specific instance of piggybacked malware associated with one malicious rider. We

then evaluated the targeting requirement by applying this universal perturbation to piggybacked malware produced by other malicious riders.

[1] Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection. IEEE Transactions on Information Forensics and Security 2020.

[2] MalPatch: Evading DNN-Based Malware Detection With Adversarial Patches. IEEE Transactions on Information Forensics and Security 2024

The experimental results on different features and different attack scenarios are presented in TABLE I. The results indicate that existing attack methods often fail to simultaneously satisfy the universality requirement and the targeting requirement (i.e., achieving a high U value and a low T value). In contrast, our approach effectively meets both requirements, resulting in superior attack performance.

TABLE I

COMPARISION WITH SOTA MALWARE ADVERSARIAL ATTACK METHODS

Drebin								
METHODS	W		B_S		G		B_D	
	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)
RN	0.044	0.019	0.041	0.029	0.026	0.019	0.040	0.030
HIV-JSMA	0.456	0.381	0.197	0.259	0.191	0.261	0.170	0.224
HIV-CW	0.599	0.525	0.156	0.265	0.269	0.321	0.126	0.216
Malpatch	0.891	0.629	0.144	0.159	0.242	0.239	0.103	0.114
Ours	1.000	0.439	0.880	0.393	0.862	0.344	0.821	0.383

FD-VAE								
METHODS	W		B_S		G		B_D	
	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)
RN	0.860	0.874	0.572	0.569	0.792	0.784	0.774	0.747
HIV-JSMA	0.315	0.225	0.123	0.117	0.091	0.077	0.145	0.134
HIV-CW	0.560	0.467	0.298	0.281	0.359	0.329	0.329	0.326
Malpatch	0.836	0.583	0.349	0.304	0.413	0.381	0.419	0.381
Ours	1.000	0.484	0.800	0.302	0.836	0.305	0.810	0.308

FD-VAE-E1								
METHODS	W		B_S		G		B_D	
	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)
RN	0.228	0.221	0.650	0.638	0.354	0.358	0.430	0.439
HIV-JSMA	0.277	0.286	0.096	0.133	0.102	0.159	0.062	0.132
HIV-CW	0.518	0.400	0.248	0.289	0.220	0.251	0.221	0.277
Malpatch	0.828	0.549	0.355	0.339	0.299	0.306	0.288	0.321
Ours	1.000	0.288	0.902	0.362	0.862	0.271	0.872	0.328

FD-VAE-E2								
METHODS	W		B_S		G		B_D	
	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)
RN	0.157	0.120	0.122	0.079	0.137	0.092	0.143	0.104
HIV-JSMA	0.323	0.293	0.097	0.176	0.087	0.159	0.124	0.186
HIV-CW	0.521	0.453	0.246	0.228	0.333	0.294	0.271	0.269
Malpatch	0.958	0.662	0.295	0.314	0.350	0.354	0.375	0.387
Ours	1.000	0.384	0.776	0.390	0.812	0.443	0.847	0.400

MaMaDroid								
METHODS	W		B_S		G		B_D	
	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)
RN	0.275	0.242	0.053	0.039	0.229	0.201	0.165	0.143
HIV-JSMA	0.457	0.387	0.437	0.393	0.434	0.375	0.419	0.356
HIV-CW	0.850	0.829	0.826	0.818	0.808	0.810	0.831	0.829
Malpatch	0.908	0.881	0.867	0.856	0.856	0.846	0.868	0.850
Ours	0.984	0.368	0.840	0.350	0.857	0.311	0.864	0.312

APIGraph								
METHODS	W		B_S		G		B_D	
	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)
RN	0.360	0.295	0.295	0.231	0.314	0.248	0.251	0.192
HIV-JSMA	0.678	0.591	0.615	0.518	0.647	0.560	0.598	0.531
HIV-CW	0.957	0.948	0.931	0.918	0.948	0.940	0.937	0.920
Malpatch	0.959	0.957	0.927	0.930	0.951	0.955	0.944	0.936
Ours	0.998	0.500	0.793	0.407	0.851	0.446	0.836	0.413

Q3: Increase the number of distinct malicious riders.

Response:

We sincerely appreciate the reviewer's feedback. In response to this comment, we have expanded the distinct malicious riders from 65 to 265. Our expansion methods primarily involve two approaches:

1. We merged existing malicious riders in pairs. We discovered scenarios where two malicious riders coexist within one piggybacked malware. Inspired by this observation, we aimed to expand the dataset by combining existing malicious riders in pairs. To ensure that the paired riders could coexist, we first selected 25 non-overlapping malicious riders (if two riders share the same file name, merging them would result in the overwriting of critical code). Subsequently, we constructed $\frac{25\times24}{2} = 325$ new malicious riders. Finally, we filtered out riders with identical features using a feature comparison method, resulting in 168 new distinct malicious riders. We refer to these riders as MR (merge riders).

2. We relaxed the extraction criteria for malicious riders. In the original manuscript, we stipulated that the malicious payload must be connected to the benign carrier through a single HOOK function call. Here, we have relaxed this constraint to allow for the presence of multiple HOOK functions, resulting in the identification of 83 malicious payloads. Subsequently, we applied a feature filtering method, which yielded 32 distinct malicious riders. We refer to these riders as MHR (multi-hook riders).

Due to time constraints, we have only validated our approach using the FD-VAE-V1 features. If the reviewer allows us to make revisions, we assure you that we will include results for all features. The experimental results are presented in TABLE II, where the first column indicates four different attack scenarios, while the second and third columns show results for the 65 distinct malicious riders used in the original manuscript. The fourth and fifth columns present the results of our algorithm on the newly added merge rider and multi-hook rider. The experimental findings demonstrate that our algorithm also performs well against the newly introduced malicious riders.

TABLE II
THE ATTACK PERFORMANCE ON NEW MALICIOUS RIDERS

Scenarios	ER		UR		MKR		MR	
	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)	U(↑)	T(↓)
W	1.000	0.253	1.000	0.227	1.000	0.156	0.970	0.248
B_S	0.890	0.312	0.895	0.263	0.811	0.185	0.754	0.279
G	0.841	0.216	0.907	0.208	0.882	0.187	0.780	0.237
B_D	0.855	0.250	0.908	0.267	0.905	0.165	0.784	0.244

Q4: Describe the machine learning models used.

Response: We sincerely appreciate the reviewer's comments. We utilize a Deep Neural Network (DNN) classification model, and relevant descriptions can be found in Appendix A. We will also include the appropriate citations in the main text.

For convenience, we have extracted the relevant descriptions here.'We utilize a DNN as the classification model, with the target model consisting of a 4-layer neural network in a gray-box scenario. For the Drebin feature, each layer comprises 10,000, 32, 32, and 1 neurons respectively. For the FD-VAE feature, each layer consists of 1,000, 1,000, 1,000, and 1 neurons. For the FD-VAE-E1 and FD-VAE-E2 features, they have 1,200, 1,200, 1,200, and 1 neurons respectively. The hidden layers employ ReLU as their activation function, while the output layer uses Sigmoid. For the MaMaDroiD and APIGraph features, each layer has 300, 300, 300, and 1 neurons. In the B_S and B_D scenarios, an additional hidden layer is either removed or added.'

Q5: Explain the motivation for dividing the black-box scenario into B_D and B_S.

Response: We really appreciate this comment. B_D and B_S represent the use of deeper and shallower DNNs, respectively, in black-box attack scenarios. The inclusion of these two scenarios aims to demonstrate that adversarial perturbations generated on a local substitute model can transfer to classification models with different architectures. We will provide additional descriptions in the main text.

Q6: How the authors' approach differ from the example provided in Pierazzi et al. (cited as [41] by the authors).

Response:

The work of Pierazzi et al. [41] was the first to explore the constraints imposed by adversarial examples in the domain of malware detection, marking an early breakthrough in this field. We compare our work with [41] on three aspects of Goal, Attack Method, and Attack Constraints.

1) In terms of Goal, Pierazzi et al. focus solely on perturbing a single piece of malware to generate adversarial malware. In contrast, our approach investigates the generation of thousands of adversarial malware instances in a short time frame. Specifically, while Pierazzi et al. can generate only one adversarial malware per attack, our algorithm explores a more prevalent method of malware generation—piggybacking. By hooking a malicious rider and perturbing it onto any benign carrier, we can produce thousands of adversarial malware instances in a single operation. Therefore, our research poses a new threat.

2) In terms of attack methods, Pierazzi et al. first obtained benign code snippets that can be used to perturb malware, subsequently using optimization algorithms to select suitable segments for injection. In contrast, our approach employs optimization algorithms to identify the functions

that need to be inserted, followed by utilizing a large model to generate the corresponding code snippets.

3) In terms of attack constraints, Pierazzi et al. proposed constraints for the problem space of attacks across all domains, while our attack constraints build upon and modify their framework. Specifically, Pierazzi et al. introduced four attack constraints:

Available transformations: The perturbations made by the attacker must meet practical requirements. In our work, we specify this constraint to ensure the normal operation of the program (i.e., Program Normal Execution Constraint).

Preserved semantics: The semantics of the samples must remain unchanged before and after perturbation. We abstract this point in our work to mean that the functionality of the malicious code segment does not change before and after perturbation (i.e., Functional Consistency Constraint).

Plausibility: The added perturbations must appear realistic upon manual inspection. In our work, we address this requirement based on the concept of Naturalness of the language (i.e., Naturalness Constraint).

Robustness to preprocessing This requirement emphasizes that the applied perturbations should not be easily filtered out, such as dead code, which can be readily eliminated by program analysis tools. However, our algorithm inherently satisfies this requirement, as it necessitates hooking both the malicious code and the perturbation code onto benign carriers. Consequently, we did not explicitly discuss this requirement in our manuscript.

Finally, we introduce an additional constraint, the flexibility constraint, which ensures that the attacker can freely select perturbations from the feature space.