GRASP: Hard-Label Black-Box Malware Evasion with Higher Success, Fewer Queries, and Smaller Perturbations

Yutong Liu[1] , **Jianting Ning**[1*] , **Qi Feng**[1] , **Yanjun Zhang**[2] , **Yujin Huang**[3] , **Leo Yu Zhang**[2]
[1]School of Cyber Science and Engineering, Wuhan University, China
[2]Griffith University, Australia
[3]The University of Melbourne, Australia
liuyutong619@whu.edu.cn, jtning88@gmail.com, fengqi.whu@whu.edu.cn,
yanjun.zhang@griffith.edu.au, jinx.huang@unimelb.edu.au, leo.zhang@griffith.edu.au

Abstract

Machine learning (ML)-based malware detectors are widely deployed but remain vulnerable to adversarial attacks. However, under hard-label black-box access, existing adversarial attacks on Windows Portable Executable (PE) malware are often query-inefficient and incur large file-size inflation. A common paradigm is to predefine a set of semantics-preserving atomic perturbations and search for evasive combinations under only binary feedback. Among these atomic perturbations, (i) those with highly combinatorial search spaces are difficult to explore effectively under hard-label feedback, leaving their potential untapped, and (ii) those relying on transplanting benign fragments are often laden with evasion-irrelevant bytes and exhibit highly variable adversarial utility. We propose Gradient-seeded Reinforcement Learning And Stealthy Pruning (GRASP), a three-stage framework that tackles these challenges. First, we decouple perturbations with highly combinatorial search spaces from the query-based search and instead apply a gradient-seeded warm-up that uses Gumbel-Softmax relaxation to enable gradient-based updates over the discrete space. This yields a strong warm start that improves evasion and reduces queries in later stages. Second, Reinforcement Learning (RL)-based refinement is accelerated by a perturbation library that filters, caches, and reuses compact high-utility patterns, reducing wasted queries on low-utility benign fragments. Third, a perturbation minimization stage removes redundant bytes while preserving evasion, reducing size inflation and feeding compact patterns back to the library. Experiments show that GRASP outperforms baselines, achieving higher attack success with fewer queries and smaller file-size inflation. We additionally demonstrate its practical effectiveness against commercial Antivirus engines.

*Corresponding author. J. Ning is also with the Zhejiang Key Laboratory of Digital Fashion and Data Governance, Zhejiang Sci-Tech University and with the Faculty of Data Science, City University of Macau, Macau.

1 Introduction

The rapid evolution of malware variants poses a persistent security threat. Among these, Windows Portable Executable (PE) binaries remain one of the most prevalent vectors [Aonzo *et al.*, 2023]. In practice, however, the rapid mutation of PE variants has rendered traditional signature-based detection increasingly inadequate [Cheng *et al.*, 2021]. To address this challenge, ML-based malware detectors have been widely adopted in both industry and academia [Gibert *et al.*, 2020]. By learning discriminative patterns from massive datasets and complex feature spaces, these detectors demonstrate strong generalization capability against previously unseen threats.

However, ML-based malware detectors are known to be vulnerable to adversarial examples [Goodfellow *et al.*, 2015; Szegedy *et al.*, 2014; Eykholt *et al.*, 2018; Huang *et al.*, 2025]. This concept was first extensively studied in computer vision, where imperceptible input perturbations can induce misclassification; in the malware domain, adversarial perturbations typically take the form of discrete, functionality-preserving transformations on the PE structure. Depending on the attacker's knowledge, existing methods are commonly categorized into **white-box** and **black-box** settings: white-box attacks assume access to the model architecture and parameters, leveraging gradient information to guide perturbation generation [Kreuk *et al.*, 2018; Demetrio *et al.*, 2019; Kolosnjaji *et al.*, 2018], whereas black-box attacks rely only on query access to the deployed detector. Since commercial antivirus engines are typically deployed as closed, proprietary services, the black-box setting is generally more realistic in practice. Within black-box attacks, some studies assume real-valued confidence scores (score-based feedback) [He *et al.*, 2023]; however, most real-world services expose only a binary verdict, yielding a hard-label oracle. Motivated by this practical deployment setting, we focus on hard-label black-box attacks. Under this setting, prior work has explored a variety of search and optimization strategies—such as genetic algorithms and reinforcement-learning-based frameworks—to discover evasive perturbations without breaking malicious functionality [Demetrio *et al.*, 2021; Song *et al.*, 2022; He *et al.*, 2024; Li *et al.*, 2025; Tian *et al.*, 2024].

Despite recent progress, existing hard-label black-box attacks remain challenging in practice. Specifically, current black-box attacks typically predefine a heterogeneous set

of atomic perturbations and perform decision-based search. This paradigm faces two critical challenges: First, among predefined perturbations, a subset of fixed-region discrete perturbations (e.g., PE header metadata modifications) induces a vast combinatorial search space that is difficult to explore under hard-label feedback. Consequently, existing methods often fail to exploit such perturbations effectively and still waste queries on unproductive random exploration, reducing attack efficiency and leaving their evasion potential largely untapped. Second, among predefined perturbations, another widely used class constructs perturbations by transplanting benign byte fragments (e.g., injecting benign fragments into newly added sections). However, these fragments frequently contain substantial evasion-irrelevant bytes and exhibit highly variable adversarial utility, resulting in an increased number of queries needed to reach evasion, while also inflating the resulting file size, undermining real-world deployability (e.g., delivery or upload size limits).

To address the above challenges, we propose **G**radient-seeded **R**einforcement Learning **A**nd **S**tealthy **P**runing (GRASP), a three-stage framework for hard-label black-box PE malware evasion designed to improve query efficiency and mitigate file-size inflation. GRASP first decouples highly combinatorial fixed-region discrete perturbations from the RL action space and conducts a gradient-seeded, query-free warm-up that applies a Gumbel-Softmax relaxation to enable differentiable optimization over this discrete search space. This yields a strong initialization that pushes candidates toward the target detector's decision boundary, increasing the evasion rate and reducing hard-label queries per successful evasion in later stages. It then refines candidates via reinforcement learning (RL) to drive them across the target detector's decision boundary. This process is assisted by a perturbation library that ingests minimized perturbations and reuses previously verified high-utility perturbation patterns to further enhance query efficiency. Finally, GRASP applies perturbation minimization to prune redundant bytes while preserving evasiveness, improving deployability and stealthiness in practical settings.

Our contributions are summarized as follows:

- We propose a **query-free gradient-seeded warm-up** to unlock the potential of highly combinatorial discrete perturbations. By employing Gumbel-Softmax relaxation, we enable gradient-based optimization over discrete spaces, yielding a strong initialization that improves evasion success and reduces hard-label queries per success, while incurring no query cost and no size overhead during warm-up.
- We propose **perturbation minimization** to prune redundant bytes while preserving evasiveness, reducing file-size inflation and improving practical stealth; the resulting compact, evasion-relevant byte patterns are stored in a **perturbation library** for reuse, improving the query efficiency of RL-based refinement under hard-label feedback.
- We conduct comprehensive experiments showing that GRASP consistently outperforms strong baselines, achieving higher evasion success with fewer queries and lower perturbation rate. We additionally evaluate GRASP on VirusTotal, demonstrating its practical effectiveness against multiple commercial detection engines.

Our source code and datasets are publicly available.[1]

2 Related Work

From the perspective of deployment scenarios, existing methods can be broadly categorized into two classes: white-box and black-box methods.

White-box methods. Early adversarial malware attacks relied on gradient-based optimization with white-box access. [Kolosnjaji *et al.*, 2018; Kreuk *et al.*, 2018] append bytes to PE files and optimize them via gradient updates in the embedding space, followed by a projection/quantization step back to valid bytes. [Demetrio *et al.*, 2019] further uses Integrated Gradients to localize sensitive header regions and perturbs only a small set of DOS-header bytes. However, real-world AV engines are usually accessible only via queries, providing neither model internals nor scores and often only a hard label; hence the shift toward hard-label black-box evasion.

Black-box methods. Under the hard-label black-box setting, early works such as [Demetrio *et al.*, 2021] formulated evasion as a constrained optimization problem and used genetic algorithms to inject benign content into file padding or new sections. By modeling malware mutation as a series of functionality-preserving atomic perturbations, subsequent research adopted RL to optimize action sequences: [Song *et al.*, 2022] adopts a multi-armed bandit to trade off exploration and exploitation, while [He *et al.*, 2024; Tian *et al.*, 2024] further enlarge the action space with more diverse functionality-preserving perturbations. To improve deployability, recent work has begun to explicitly reduce the file-size growth induced by adversarial perturbations. [Li *et al.*, 2025] employed a roulette-wheel selection strategy to iteratively choose injection locations and benign content until evasion was achieved, which can introduce substantial redundant bytes. It then prunes redundancy with query-intensive Particle Swarm Optimization (PSO) and binary search.

3 Threat Model

Adversary Goals. Given a malicious PE file x, the adversary aims to craft a perturbation δ to generate an adversarial example x_{adv}—denoted in this work as adversarial malware. This adversarial malware is considered successful only if it: (1) misleads the detector f into classifying the malware as benign; (2) preserves the original malicious behavior. Additionally, the adversary seeks to minimize the introduced perturbation to improve practical deployability. Formally, the adversary's goals can be formulated as:

$$\min_{\delta} \frac{\text{size}(x_{adv}) - \text{size}(x)}{\text{size}(x)}, \quad \text{s.t.} \quad \begin{cases} f(x_{adv}) \neq f(x) \\ \mathcal{F}(x_{adv}) = \mathcal{F}(x) \\ q \leq Q \end{cases}, \tag{1}$$

[1] https://github.com/liuyutong619/GRASP

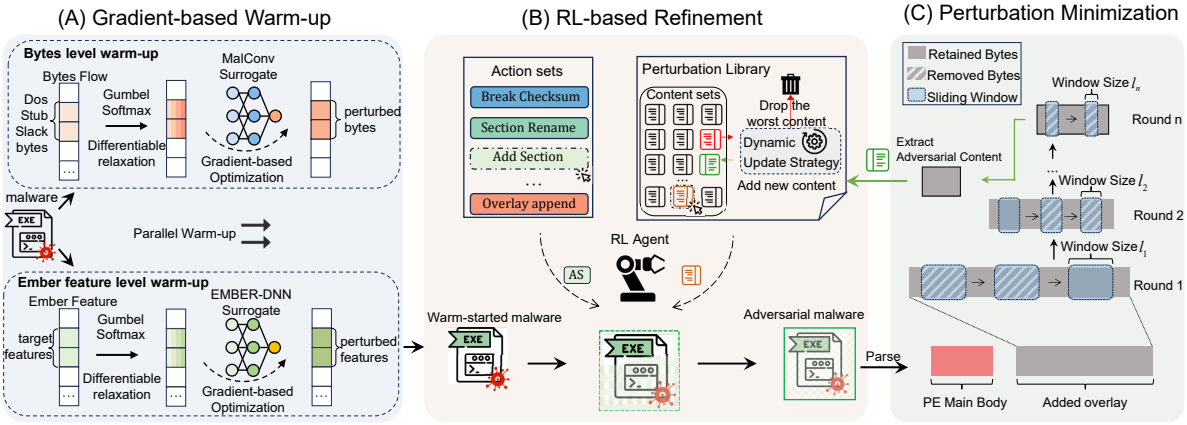


Figure 1: The overall framework of GRASP.

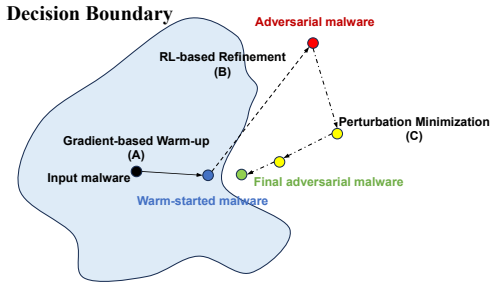


Figure 2: Optimization trajectory of GRASP in decision space.

Cat.	ID-Name	Description
Grad.-based	**SA**: Slack-Append	Modify bytes in section slack space.
	DA: Dos-Append	Modify bytes in PE DOS stub.
	VM: Ver-Modify	Modify image/linker/OS versions.
	TM: Time-Modify	Modify *TimeStamp* field.
RL-based	**OA**: Overlay-App	Inject bytes at the end of the PE file.
	AS: Add-Section	Adds a new section containing bytes.
	SR: Sect-Rename	Rename section to benign-like name.
	RC: Rm-Cert	Zeroes out security dir (certificate).
	RD: Rm-Debug	Zeroes out debug information.
	BC: Brk-ChkSum	Zeroes out *CheckSum* in Opt Header.

Table 1: Perturbation actions used in our framework.

where $f : \mathcal{X} \to \{0, 1\}$ is the target malware detector with hard-label feedback, and $f(x) = 1$ (resp. 0) indicates malware (resp. benign), $\mathcal{F}(\cdot)$ represents the functionality verification function. q denotes the current number of queries, constrained by the maximum budget Q. The objective function minimizes the relative file size increase. Our perturbation design follows the problem-space constraints identified by Pierazzi et al. [Pierazzi *et al.*, 2020]: all actions preserve semantics (verified via sandbox, Sec. 5.5), use only available PE transformations (Table 1), survive diverse preprocessing pipelines (validated on 70+ VirusTotal engines, Sec. 5.4), and maintain plausibility by minimizing file-size inflation (Sec. 4.4).

Adversary Knowledge. We consider a realistic black-box setting. The adversary has no access to the target model's parameters, gradients, or training data. The adversary can only query the target detector with a sample and receive a hard-label prediction (malicious or benign) within a query budget Q. However, the adversary is permitted to utilize public datasets and resources to train local surrogate models.

4 Methodology

4.1 Framework Overview

To generate high-stealth adversarial malware efficiently, we propose **G**radient-seeded **R**einforcement Learning **A**nd **S**tealthy **P**runing (GRASP). As illustrated in Fig. 1, GRASP comprises three cascaded modules, which correspond to the distinct phases of the optimization trajectory in the decision space shown in Fig. 2: (A) **Gradient-based Warm-up:** To overcome the non-differentiable nature of discrete features, this module utilizes Gumbel-Softmax relaxation to leverage gradient information, pushing the malware to the vicinity of the decision boundary. (B) **RL-based Refinement:** This module employs reinforcement learning to fine-tune the warm-started malware, utilizing a dynamically maintained Perturbation Library to guide the sample to successfully cross the decision boundary. (C) **Perturbation Minimization:** A sliding-window strategy is applied to eliminate redundant added bytes, thereby generating final adversarial malware with both high evasiveness and low perturbation rates.

4.2 Gradient-Based Warm-Up

The Gradient-based Warm-up phase specifically targets "Gradient-Based Actions" in Table 1, i.e., high-dimensional discrete perturbations confined to predefined PE locations, including header-field edits (TM/VM) and N-byte injection into fixed regions (SA/DA). Prior black-box RL studies ([Song *et al.*, 2022], [Tian *et al.*, 2024], [He *et al.*, 2024]) often treat such perturbations as discrete atomic actions, which yield a very large combinatorial search space: header fields span wide integer domains, while byte injections induce a 256^N combinatorial space. As a result, only a small fraction of actions produces effective adversarial perturbations. Un-

der a limited query budget and hard-label feedback, the agent observes mainly zero/low-reward signals, making policy collapsing to near-random exploration.

To mitigate this, leveraging gradient information from surrogate models to guide the search direction is a promising solution. Nevertheless, a key barrier exists: the features in Table 1 are intrinsically discrete. This non-differentiability obstructs the gradient flow in the computational graph, preventing the adversarial loss from being backpropagated through the embedding layer to the input, standard gradient-based attacks (e.g., FGSM [Goodfellow *et al.*, 2015]) are not directly applicable without a differentiable relaxation.

As illustrated in Fig. 3, we introduce the Gumbel-Softmax technique [Jang *et al.*, 2017] to construct a differentiable optimization path. Formally, let $F \in \mathbb{Z}^L$ denote the model input feature vector and $\mathcal{I}$ the set of perturbable indices. For each $i \in \mathcal{I}$, we maintain a learnable proxy distribution $\theta_i \in \mathbb{R}^V$ over V candidate values for feature F_i, initialized to approximate the one-hot encoding of the original value. We then apply the Gumbel-Softmax reparameterization to θ_i to produce a differentiable weight vector w_i, compute the expected embedding $e_i = w_i^\top D_i$ via a pre-stored dictionary, and backpropagate the adversarial loss to update θ_i; final perturbed values are sampled from the converged θ_i. Specifically, $w_{i,k}$ is computed as:

$$w_{i,k} = \frac{\exp((\theta_{i,k} + g_{i,k})/\tau)}{\sum_{j=1}^{V} \exp((\theta_{i,j} + g_{i,j})/\tau)}, \tag{2}$$

where $k \in \{1, \cdots, V\}$ indexes the V candidate discrete values for feature F_i, τ denotes the temperature parameter, and $g_{i,k} \sim \text{Gumbel}(0, 1)$ represents Gumbel noise. To feed w_i into the surrogate model, we construct a set of dictionaries $D = \{D_i\}_{i \in \mathcal{I}}$, where $D_i \in \mathbb{R}^{V \times H}$ pre-stores the embedding vectors corresponding to all V candidate discrete values at position i (H denotes the embedding dimension); for positions not in $\mathcal{I}$, embeddings are fixed to the original values.

To handle the heterogeneous action types, we instantiate two parallel warm-up strategies:

Bytes-level Warm-up. Targeting **SA** and **DA** which involve raw byte manipulations, we employ MalConv and AvastNet [Raff *et al.*, 2018; Krcál *et al.*, 2018] as the surrogate model. The input $F \in \mathbb{Z}^L$ is the PE file byte stream, so $V = 256$ (corresponding to byte values 0-255). We set $D_i \equiv D_{\text{shared}}$ to be the byte embedding matrix (with $H = 8$) from the surrogate, which is shared across all byte positions; thus for each perturbable byte position $i \in \mathcal{I}$, $e_i = w_i^\top \cdot D_{shared}$ is the expected embedding vector under the relaxed distribution w_i.

Feature-level Warm-up. Targeting **VM** and **TM** which involve numerical modifications, we construct an EMBER-feature ([Anderson and Roth, 2018]) based DNN as the surrogate model. Since these fields span wide integer ranges, we discretize each target feature F_i by uniformly selecting V values from $[\min_i, \max_i]$ to form D_i. In this case, $H = 1$ and $D_i \in \mathbb{R}^{V \times 1}$ contains the candidate scalar values. Consequently, the $e_i = w_i^\top D_i$ is the weighted expectation of F_i, serving directly as the scalar input to the DNN.

4.3 RL-Based Refinement

Although the gradient-based warm-up can efficiently optimize a differentiable surrogate objective using a local surrogate model, the resulting updates are inevitably affected by the surrogate–target mismatch in our hard-label black-box setting [Liu *et al.*, 2017]. Consequently, warm-started samples are often moved into the vicinity of the target decision boundary but may still fail to cross it. To address this gap, we introduce an RL-based refinement stage that operates directly on the target's feedback and further improves evasion for the remaining samples.

We adopt the Multi-Armed Bandit (MAB) framework proposed by [Song *et al.*, 2022] as our optimization engine. MAB treats the process as a stateless arm-selection problem, and is query-efficient under hard-label feedback. Specifically, we model the search space as a set of arms, where each arm represents an action-content tuple $\langle a, c \rangle$. Here, a belongs to the candidate action set $\mathcal{A}$ (defined as "RL-Based Actions" in Table 1), and c denotes the specific content associated with that action. The agent operates with two types of arms to balance exploration and exploitation:

- Generic arms (Exploration): For a chosen action a, the content c is sampled on-the-fly enabling the agent to explore new perturbation patterns.
- Specific arms (Exploitation): The agent selects a registered action-content tuple $\langle a, c \rangle$, where the content c is a fixed and verified entry stored in our Perturbation Library $\mathcal{L}$ for reuse.

MAB tracks each arm with a Beta distribution $\text{Beta}(\alpha, \beta)$, where α and β represent the accumulated counts of successful evasions and failures, respectively. These statistical parameters (α, β) not only guide the current refinement search through Thompson sampling (i.e., sampling a score from each $\text{Beta}(\alpha, \beta)$ posterior and selecting the arm with the highest score), but also simultaneously serve as the key metric for maintaining the Perturbation Library.

Perturbation Library

The efficacy of byte-injection actions (e.g., OA and SA) largely depends on the utility of the injected byte content. Injecting benign-derived bytes with low utility can dilute the adversarial effect and unnecessarily inflate the file size. Existing methods often extract byte sequences directly from benign executables; however, this raw byte content typically contains substantial redundancy that does not contribute to evasion, as reflected by the considerable size reduction achieved by our Perturbation Minimization module (Sec. 4.4). Consequently, the minimized perturbation yields higher-utility, more compact byte content by retaining only the evasion-critical byte segments. Motivated by this, we construct the Perturbation Library to cache and reuse such high-utility content across samples, thereby improving the query efficiency of the refinement stage.

In the RL refinement stage, the library supplies verified, high-utility content for the RL agent, enabling content reuse across samples and improving query efficiency.

However, the library cannot grow indefinitely: adding more content entries expands the arm space and reduces

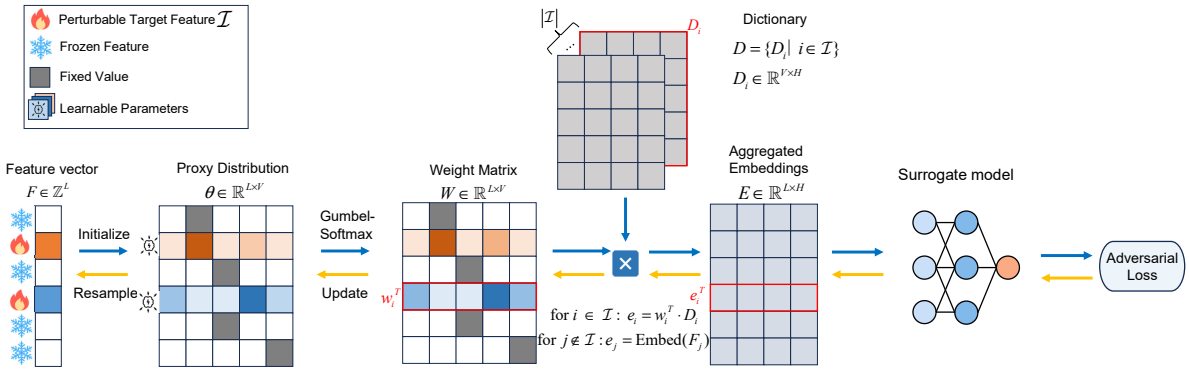


Figure 3: Illustration of the Gradient-based Warm-up Mechanism via Gumbel-Softmax Relaxation.

query efficiency under a fixed budget. We thus impose a capacity limit M and adopt a dynamic maintenance strategy.

We leverage the Beta posterior parameters maintained by the RL agent to assess the quality of stored content. At initialization, we seed the perturbation library with 50 content entries extracted from benign executables. For registered entries, we aggregate statistics across all actions paired with the same content by summing their counts (i.e., $\alpha_c = \sum \alpha_{a,c}$ and $\beta_c = \sum \beta_{a,c}$ over all registered $\langle$a,c$\rangle$), and compute a utility score $s_c = \alpha_c/\beta_c$. When the library reaches its capacity (M=150) and a new content c_{new} arrives, we preferentially evict an entry that has not yet been registered as any MAB arm (i.e., one of the initial benign-derived contents that has never been selected for perturbation), thereby replacing unverified seeds with proven adversarial content; otherwise, we evict the registered entry with the lowest utility score, i.e., $c_{\text{remove}} = \arg\min_{c \in \mathcal{L}} s_c$.

4.4 Perturbation Minimization

RL-based refinement may rely on repeatedly appending bytes to the file end (overlay) to achieve evasion, which can cause substantial file-size inflation and undermine practicality (e.g., causing transmission overheads and compromising stealthiness). Given an already successful adversarial malware, our goal is to prune redundant overlay bytes while preserving evasion under a local query budget.

In practice, the overlay appended by RL may contain redundancy that is not concentrated in a single contiguous block, but scattered across multiple disjoint segments. Unlike prior minimization methods that typically prune bytes within a fixed contiguous region, such as the PSO-based MiniMal [Li *et al.*, 2025] and the GA-based GAMMA [Demetrio *et al.*, 2021], our coarse-to-fine sliding-window deletion can remove multiple disjoint redundant segments.

Starting from a coarse window size, we slide the window from left to right and tentatively delete the bytes inside the window, if the resulting candidate still satisfies evasion, we accept the deletion, otherwise we keep the bytes. We then progressively decrease the window size to refine the solution from coarse to fine, and terminate early when the budget is exhausted or the window becomes too small for cost-effective gains. Finally, we insert the minimized overlay into the perturbation library to improve reusability across samples.

5 Experiments

5.1 Experimental Setup

Detectors. To ensure a fair comparison with state-of-the-art (SOTA) black-box attacks, we evaluate GRASP against two representative ML-based malware detectors that are widely adopted as standard benchmark targets in recent evasion studies [Song *et al.*, 2022; He *et al.*, 2024]. Additionally, we evaluate GRASP on VirusTotal in a real-world setting (Sec. 5.4).

- **EMBER.** A feature-based PE malware classifier built on the EMBER feature set and trained with LightGBM [Anderson and Roth, 2018].
- **MalConv.** A byte-based PE malware classifier that takes raw executable bytes as input and adopts a convolutional architecture [Raff *et al.*, 2018].

We use the pretrained models from MLSEC 2019 [Endgame, Inc., 2019] and query both detectors under hard-label access.
Datasets. We use different datasets for training local surrogate models and for evaluating attacks on the target detectors.

To support the gradient-based warm-up stage, we train a local surrogate 3-layer DNN model on the Ember feature space using Sorel20M [Harang and Rudd, 2020]. Since Sorel20M only provides pre-extracted EMBER features rather than raw PE binaries, for byte-based surrogate models that require raw executable bytes, we train them on the Dike dataset [Iosif, 2021]. Note that these surrogate training datasets differ from the data used to train the target detectors [Endgame, Inc., 2019], which better reflects a realistic black-box setting. For evaluation, we adopt VirusShare as the malware corpus and use its most recent package, VirusShare-499, collected in Aug. 2025. We filter the package by retaining only binaries that are classified as malicious by both target detectors to ensure that every sample is initially detected as malware. This results in 3921 test samples. We use a fixed random subset of 1024 samples for Secs. 5.2 and 5.3, and 99 samples for Sec. 5.4 due to query-cost constraints.
Evaluation Metrics. We adopt three metrics to evaluate GRASP: Attack Success Rate (ASR), Average Queries

Method	EMBER			MalConv		
	ASR(%)↑	Q_{used}↓	P(%)↓	ASR(%)↑	Q_{used}↓	P(%)↓
MAB	82.32	87.51	93.53	85.16	60.67	78.46
GAMMA-section	39.94	234.18	1582.17	77.54	234.21	1124.39
GAMMA-overlay	19.82	234.30	751.46	87.50	234.34	971.13
MalwareTotal	21.58	203.02	**12.79**	-	-	-
MiniMal	74.12	198.24	65.78	92.28	234.57	12.13
GRASP	**93.65**	**59.43**	35.44	**95.02**	**47.41**	**8.39**

Table 2: Comparison of different attacks' performance against ML-based malware detectors.

Method	EMBER			MalConv		
	ASR(%)↑	Q_{used}↓	P(%)↓	ASR(%)↑	Q_{used}↓	P(%)↓
GRASP	**93.65**	59.43	**35.44**	**95.02**	47.41	**8.39**
w/o Warm-up	89.36	68.00	47.00	93.26	59.86	14.61
w/o library	86.91	76.93	37.63	87.01	65.37	15.48
w/o Minimization	88.09	**48.46**	107.71	88.57	**43.41**	73.86

Table 3: Ablation study of GRASP components. "w/o" denotes "without". The best results are highlighted in **bold**.

(Q_{used}), Perturbation Rate (P). ASR is defined as the percentage of malware samples that successfully evade the target detector. Average Queries (Q_{used}) denotes the mean number of queries consumed across *all* samples, capturing the overall query cost. Perturbation Rate (P) measures the relative file-size increase, which is averaged exclusively across the *successful* adversarial examples.

Baselines. To assess the performance of GRASP, we benchmark it against five SOTA methods: GAMMA-section [Demetrio *et al.*, 2021], GAMMA-overlay [Demetrio *et al.*, 2021], MAB [Song *et al.*, 2022], MalwareTotal [He *et al.*, 2024], and MiniMal [Li *et al.*, 2025]. These baselines encompass a broad spectrum of optimization strategies, ranging from reinforcement learning to heuristic algorithms, representing the current SOTA in adversarial malware generation.

Implementation Detail. We implement GRASP in Python, leveraging the pefile and LIEF libraries for PE file parsing and manipulation. All experiments were conducted on an Ubuntu 20.04.5 LTS server with an Intel Xeon Platinum 8470Q CPU, an NVIDIA RTX 5090 GPU (32GB), and 90 GB RAM.

5.2 Comparison Between Different Methods

We compare GRASP with five representative baselines covering both RL-based and heuristic-based attacks on ML-based detectors. For all baselines, we use their official open-source implementations. For MalwareTotal, we directly use the released pre-trained RL agent for evaluation (available for EMBER only). For a fair comparison, we set the maximum query budget to 250 per sample for all methods, and all methods are restricted to use the same benign section contents extracted from benign PE files. For GRASP, we cap perturbation minimization to at most 30 queries.

Table 2 summarizes the overall experimental results. GRASP achieves the best ASR on both ML-based malware detectors while reducing the number of queries and file-size inflation, highlighting its favorable ASR–query–size trade-off. RL-based baselines such as MAB and MalwareTotal are not well aligned with fixed-region, high-dimensional discrete perturbations; under hard-label feedback, exploring these combinatorial actions is query-inefficient, wasting queries and degrading the ASR–query trade-off. In contrast, while GAMMA and MiniMal aim to reduce perturbation rate, they fundamentally rely on transplanting raw benign byte fragments. These fragments are often laden with evasion-irrelevant redundancy, resulting in highly variable adversarial utility that limits their overall ASR. Furthermore, mitigating this inherent redundancy necessitates additional queries to reduce file-size inflation (P); GAMMA consumes excessive queries yet fails to effectively reduce P, while MiniMal requires query-intensive optimization to prune the excess bytes.

GRASP overcomes these limitations by decoupling fixed-region perturbations into a gradient-based warm-up, reusing verified high-utility patterns via the perturbation library, and pruning redundant bytes through sliding-window minimization, thereby achieving a superior balance among ASR, query cost, and file size.

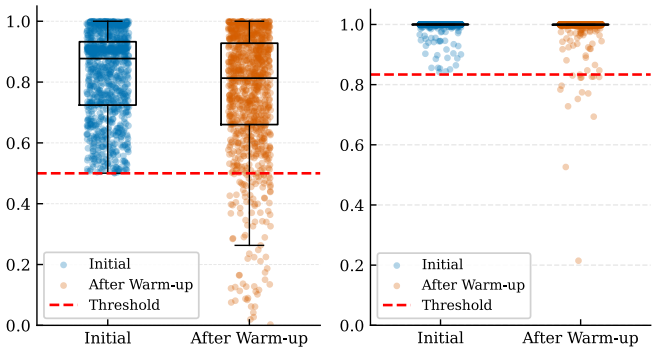
5.3 Ablation Study

We conduct an ablation study by removing each core component of GRASP in turn, with results reported in Table 3. Removing gradient-based warm-up decreases ASR while increasing both Q_{used} and the perturbation rate P, indicating that warm-up provides a stronger starting point for the subsequent refinement, thereby reducing the cost of achieving evasion. Removing the perturbation library markedly degrades ASR, Q_{used} and P, indicating that reusing verified, high-utility perturbation patterns improves the hit rate of the refinement stage. Finally, removing perturbation minimization causes P to surge while Q_{used} decreases, showing that minimization trades a modest additional query budget for substantial compression of the injected bytes. Moreover, minimization also strengthens the library by supplying compact, reusable high-utility patterns; without it, library becomes less effective and the overall trade-off degrades.

Beyond the quantitative results reported above, we provide a fine-grained analysis of the gradient-seeded warm-up and the perturbation library in the following sections.

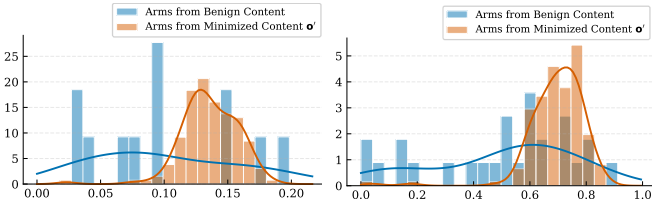
Effect of the Gradient-Based Warm-Up. To validate the effect of the Gradient-Based Warm-up stage, we visualize the distribution of malicious scores before and after the Warm-up stage in Fig. 4, the warm-up stage consistently shifts the malicious-score distributions toward the decision threshold, with some samples already crossing it. This suggests that Gradient-Based Warm-up can move binaries close to the decision boundary prior to querying the target model, which facilitates the subsequent refinement stage in driving samples across the boundary.

Effect of the Perturbation Library. To validate the effect of the Perturbation Library, we plot the distributions of utility scores (α/β) for arms derived from (i) raw benign content and (ii) minimized content in Fig. 5, where the latter corresponds to the library-specific arms. A larger utility score indicates that the corresponding arm is more effective. As shown in Fig. 5, arms built from minimized content exhibit a distribution that is more concentrated and consistently shifted toward



(a) MalConv (b) Ember

Figure 4: Impact of the Gradient-based Warm-up on malicious scores for (a) MalConv and (b) EMBER. The red dashed line denotes the decision threshold.



(a) MalConv (b) Ember

Figure 5: Utility score (α/β) distributions of perturbation arms derived from benign content vs. minimized content $\mathbf{o}'$ for (a) MalConv and (b) EMBER.

higher utility, whereas arms derived from benign content are more dispersed and skewed toward lower scores. This pattern suggests that the Perturbation Library provides higher-quality and more reliable perturbation contents: minimized-content arms tend to deliver stable utility across samples, while benign-content arms are generally weaker and only occasionally yield strong arms with high variance.

5.4 Performance on Real-World AVs

To validate the practical effectiveness of GRASP, we evaluate it against VirusTotal (VT) [VirusTotal, 2026], an online multi-engine scanning service that aggregates verdicts from 70+ antivirus products. We curate a test set of 99 malware samples from VirusShare-499, where each sample is initially flagged as malicious by more than 50 VT engines. Since VT returns a malicious score (i.e., the detection count) rather than a binary label, we adapt the RL-based refinement to this setting: after each action, we query VT and assign a positive reward if the detection count decreases. We allocate 15 queries per sample for RL refinement and an additional 5 queries for perturbation minimization.

Fig. 6 reports the VT malicious score distribution across stages. Importantly, the Gradient-based Warm-up stage uses no VT feedback. It reduces the mean VT score from 64.8 to 56.8, indicating strong transferability of gradient-guided perturbations to unseen commercial engines. With 15 VT queries for RL refinement, the mean score further decreases to 39.7, and 23 out of 99 samples achieve zero detection on

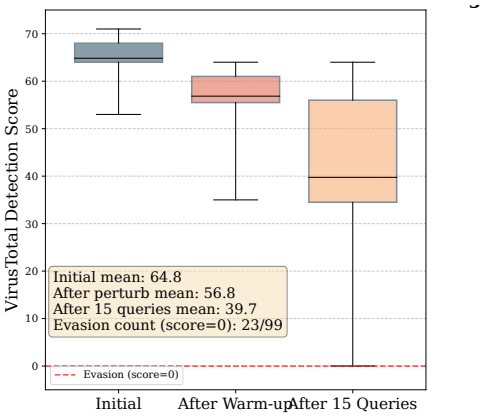


Figure 6: VT malicious Score Distribution at Different Stages.

VT at the time of querying. We additionally analyze the zero-detection cases and observe that they are frequently associated with OA and SA actions that reuse high-utility byte content retrieved from the **Perturbation Library**, suggesting that cached perturbation content can be beneficial in the real-world setting. Finally, with 5 additional VT queries, perturbation minimization reduces the mean perturbation rate from 67.03% to 19.00% while preserving the achieved VT-evasion status, yielding stealthier adversarial binaries under strict query budgets.

5.5 Functionality Preservation

We assess whether perturbations preserve malicious functionality using the Threatbook Cloud Sandbox.[2] We randomly sample 50 malware files and generate perturbed variants at different stages of our framework: (1) after the Gradient-based Warm-up, (2) after the RL-based Refinement, (3) after the Perturbation Minimization and (4) variants obtained by applying each action in Table 1. Each variant is paired with its corresponding original sample for comparison. For each original and its variant, we extract the sandbox-reported signatures from the analysis report. Following the functionality validation method used in [He *et al.*, 2024], we consider a perturbed variant to be functionality-preserving if the signatures returned for the perturbed sample include all signatures observed for the corresponding original sample. Under this criterion, all evaluated variants preserve functionality.

6 Conclusion

We introduced GRASP, a three-stage framework for hard-label black-box PE malware evasion that integrates a gradient-based warm-up, an RL-based refinement with a perturbation library, and a perturbation minimization module, achieving a superior trade-off among attack success, query cost, and perturbation size. The primary limitation of the current warm-up lies in the surrogate-target gap, which may limit its effectiveness when the two architectures diverge significantly. Future work includes extending the gradient-based warm-up to optimize a broader range of perturbations and developing transferability-enhancement techniques to better bridge surrogate and target detectors under distribution shifts.

[2]https://threatbook.io/

Ethical Statement

The methodology described in this paper is designed to uncover limitations in existing malware detectors and to help develop defenses. We acknowledge that adversarial malware generation can be dual-use. We conducted all of our experiments in a secure and isolated sand-box environment to avoid infecting external systems with any malicious code. All datasets (VirusShare, SOREL-20M, Dike) are publicly available for research purposes.

Acknowledgments

Qi Feng is supported by National Natural Science Foundation of China (NSFC Grant No. 62572355). Jianting Ning is supported by the Program of Zhejiang Key Laboratory of Digital Fashion and Data Governance (No. 2024E10049).

References

[Anderson and Roth, 2018] Hyrum S. Anderson and Phil Roth. EMBER: an open dataset for training static PE malware machine learning models. *CoRR*, abs/1804.04637, 2018.

[Aonzo *et al.*, 2023] Simone Aonzo, Yufei Han, Alessandro Mantovani, and Davide Balzarotti. Humans vs. machines in malware classification. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 1145–1162. USENIX Association, 2023.

[Cheng *et al.*, 2021] Binlin Cheng, Jiang Ming, Erika A. Leal, Haotian Zhang, Jianming Fu, Guojun Peng, and Jean-Yves Marion. Obfuscation-resilient executable payload extraction from packed malware. In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 3451–3468. USENIX Association, 2021.

[Demetrio *et al.*, 2019] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Fabio Roli, and Alessandro Armando. Explaining vulnerabilities of deep learning to adversarial malware binaries. In Pierpaolo Degano and Roberto Zunino, editors, *Proceedings of the Third Italian Conference on Cyber Security, Pisa, Italy, February 13-15, 2019*, volume 2315 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[Demetrio *et al.*, 2021] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Fabio Roli, and Alessandro Armando. Functionality-preserving black-box optimization of adversarial windows malware. *IEEE Trans. Inf. Forensics Secur.*, 16:3469–3478, 2021.

[Endgame, Inc., 2019] Endgame, Inc. Machine learning security evasion competition source code. https://github.com/endgameinc/malware_evasion_competition, 2019. Accessed: 2026-01-10.

[Eykholt *et al.*, 2018] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1625–1634. Computer Vision Foundation / IEEE Computer Society, 2018.

[Gibert *et al.*, 2020] Daniel Gibert, Carles Mateu, and Jordi Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *J. Netw. Comput. Appl.*, 153:102526, 2020.

[Goodfellow *et al.*, 2015] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Harang and Rudd, 2020] Richard E. Harang and Ethan M. Rudd. SOREL-20M: A large scale benchmark dataset for malicious PE detection. *CoRR*, abs/2012.07634, 2020.

[He *et al.*, 2023] Ping He, Yifan Xia, Xuhong Zhang, and Shouling Ji. Efficient query-based attack against ml-based android malware detection under zero knowledge setting. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 90–104. ACM, 2023.

[He *et al.*, 2024] Shuai He, Cai Fu, Hong Hu, Jiahe Chen, Jianqiang Lv, and Shuai Jiang. Malwaretotal: Multi-faceted and sequence-aware bypass tactics against static malware detection. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024*, pages 172:1–172:12. ACM, 2024.

[Huang *et al.*, 2025] Yujin Huang, Zhi Zhang, Qingchuan Zhao, Xingliang Yuan, and Chunyang Chen. Themis: Towards practical intellectual property protection for post-deployment on-device deep learning models. In *34th USENIX security symposium (USENIX Security 25)*, pages 7311–7330, 2025.

[Iosif, 2021] George-Andrei Iosif. Dikedataset: Dataset with labeled benign and malicious files. https://github.com/iosifache/DikeDataset, 2021. Accessed: 2026-01-10.

[Jang *et al.*, 2017] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[Kolosnjaji *et al.*, 2018] Bojan Kolosnjaji, Ambra Demontis, Battista Biggio, Davide Maiorca, Giorgio Giacinto, Claudia Eckert, and Fabio Roli. Adversarial malware binaries: Evading deep learning for malware detection in executables. In *26th European Signal Processing Conference, EUSIPCO 2018, Roma, Italy, September 3-7, 2018*, pages 533–537. IEEE, 2018.

[Krcál *et al.*, 2018] Marek Krcál, Ondrej Svec, Martin Bálek, and Otakar Jasek. Deep convolutional malware classifiers can learn from raw executables and labels only. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018.

[Kreuk *et al.*, 2018] Felix Kreuk, Assi Barak, Shir Aviv-Reuven, Moran Baruch, Benny Pinkas, and Joseph Keshet. Adversarial examples on discrete sequences for beating whole-binary malware detection. *CoRR*, abs/1802.04528, 2018.

[Li *et al.*, 2025] Chengyi Li, Zhiyuan Jiang, Yongjun Wang, Tian Xia, Yayuan Zhang, and Yuhang Mao. Minimal: Hard-label adversarial attack against static malware detection with minimal perturbation. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 5589–5597. ijcai.org, 2025.

[Liu *et al.*, 2017] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[Pierazzi *et al.*, 2020] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ML attacks in the problem space. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 1332–1349. IEEE, 2020.

[Raff *et al.*, 2018] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles K. Nicholas. Malware detection by eating a whole EXE. In *The Workshops of the The Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, volume WS-18 of *AAAI Technical Report*, pages 268–276. AAAI Press, 2018.

[Song *et al.*, 2022] Wei Song, Xuezixiang Li, Sadia Afroz, Deepali Garg, Dmitry Kuznetsov, and Heng Yin. Mab-malware: A reinforcement learning framework for black-box generation of adversarial malware. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, pages 990–1003. ACM, 2022.

[Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[Tian *et al.*, 2024] Buwei Tian, Junyong Jiang, Zichen He, Xin Yuan, Lu Dong, and Changyin Sun. Functionality-verification attack framework based on reinforcement learning against static malware detectors. *IEEE Trans. Inf. Forensics Secur.*, 19:8500–8514, 2024.

[VirusTotal, 2026] VirusTotal. VirusTotal. https://www.virustotal.com/, 2026. Accessed: 2026-01-10.