LLM Guided Adversarial Feature-to-Executable Malware Generation

Saeyeon Hong[1], Hyejin Woo[2], Md Mahmuduzzaman Kamol[3], Se Eun Oh[1], Mohammad Saidur Rahman[3]
[1]Dept. of Computer Science and Engineering, Ewha Womans University, Korea
[2]Dept. of AI Cyber Security, Korea University, Korea
[3]Dept. of Computer Science, University of Texas at El Paso, USA
{saeyeonhong, seoh}@ewha.ac.kr, whk5139@korea.ac.kr, mkamol@miners.utep.edu, msrahman3@utep.edu

***Abstract*—Feature-space adversarial malware is only a practical threat if perturbed features can be realized as executable binaries. We propose an LLM-guided feature-to-executable pipeline that (i) uses SHAP to produce target adversarial feature vectors and (ii) prompts LLMs to synthesize overlay and section modifications that preserve PE executability. On 10,000 VirusShare samples against the EMBER detector, our SHAP attacks achieve 87.2% Attack Success Rate (ASR) (vs. 71.4% for MAB-malware). All 100 tested binaries executed in sandboxing; however, when we re-extract features from the synthesized binaries and re-evaluate the detector, the ASR falls to 55.1% due to the feature-to-binary reconstruction gap.**

I. Introduction

VirusTotal [7] reports processing approximately 1.8 million samples every day. As the underlying data distribution continuously evolves, machine learning-based malware detectors inevitably suffer from concept drift, leading to degraded performance over time [3]. Beyond distributional changes, the threat posed by newly generated *adversarial malware* is particularly concerning, as it can actively evade learning-based detectors. To remain effective in such environments, malware detection systems must not only be robust to drift but also resilient against adversarial manipulation.

Unlike adversarial examples in the image domain, adversarial malware must satisfy an additional but crucial constraint: *executability*. An adversarial malware sample is only a real threat if the perturbed binary can still execute while preserving its malicious functionality. Without executability, adversarial perturbations remain abstract feature-space manipulations with no operational impact. This distinction underscores executability as the defining criterion for whether adversarial malware constitutes a genuine security risk.

Prior studies on adversarial malware [1] have primarily applied perturbations in feature space. While these methods achieve high attack success rates (ASR) – misclassifying malware as benign – their practical threat remains questionable because the adversarial feature vectors cannot be realized as functioning executables. More recently, MAB-malware [5] employs reinforcement learning (RL) to generate perturbations. Although RL-based methods report improved performance, their attack strategy is opaque: the learned policy does not explicitly guarantee evasive behavior or executability for unseen samples until the policy is tested, leaving reliability uncertain.

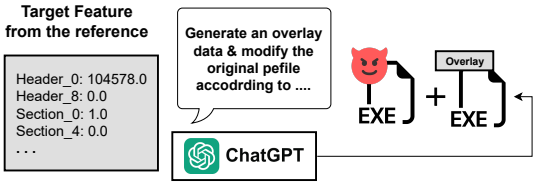

Fig. 1. Overview of Feature-to-Executable

In this work, we propose an LLM-guided feature-to-executable adversarial malware generation framework that closes the gap between feature perturbations and executable malware. Our approach (i) leverages Shapley Additive Explanations (SHAP) to identify high-impact features and craft stronger adversarial perturbations than prior methods, and (ii) employs large language models (LLMs) to translate adversarial features into fully executable binaries while preserving their functionality. By enforcing executability as a core design principle, our method provides a more realistic and consequential evaluation of adversarial threats against malware detection systems.

II. Methodology

Our method perturbs feature representations to induce benign classification and then translates them into executable binaries while preserving malicious behavior.

A. Generating adversarial features with SHAP.

We employ SHAP [4] to decompose each EMBER [2] sample (2,381 features) into additive contributions, where positive values indicate evidence toward the malicious class and negative values favor the benign class.

To flip a prediction, we adopt a greedy perturbation strategy (Algorithm 1): at each step, the feature with the largest positive contribution is replaced with a benign candidate value from a reference set that minimizes its SHAP score. The process repeats until the total contribution sum becomes negative, yielding a benign classification.

Algorithm 1: SHAP-Guided Greedy Perturbation

Input: Sample x, SHAP values $\{\phi_i(x)\}$, reference set R
Output: Perturbed sample x'
$x' \leftarrow x$;
while $\sum_i \phi_i(x') \geq 0$ **do**
  $j \leftarrow \arg\max_i \phi_i(x')$;
  `// largest malicious contribution`
  $v^\star \leftarrow \arg\min_{v \in R_j} \phi_j(x'_{[j]\leftarrow v})$;
  $x'_{[j]} \leftarrow v^\star$;
  recompute $\{\phi_i(x')\}$;
return x'

Algorithm 2: LLM-Guided Feature-to-Executable

Input: Perturbed features x', original PE b
Output: Executable $b^\star$
$b^\star \leftarrow b$;
prompt LLM to synthesize sections from x' and apply to $b^\star$;
prompt LLM with Δ_{byte} to generate overlay code;
apply overlay O_{byte} to $b^\star$;
return $b^\star$

B. Feature-to-Executable Generation with Prompt.

Prompt Example

Prompt. *Please generate code that constructs the overlay O_{byte} so that the executable meets the following condition: feature j : target value $v^\star$.*

We leverage LLMs to translate perturbed feature vectors into valid PE binaries (Algorithm 2). For byte-histogram features, the LLM is prompted to generate overlay code that adjusts byte counts without corrupting the file structure. For section-level features, which often carry large SHAP contributions, the LLM synthesizes benign-looking payloads (e.g., ZIP, SQLite, BMP) with valid headers and alignment. This process enforces executability while approximating the adversarial target features.

III. Evaluation

We evaluate our framework on 10,000 randomly selected malware samples from the VirusShare dataset [6]. The EMBER model [2] is adopted as the target detector. We compare our method against MAB-malware, a reinforcement-learning based perturbation approach. Evaluation focuses on three aspects: ASR, executability, and robustness of feature-to-executable translation.

Our method achieves a substantially higher ASR than MAB-malware (87.2% vs. 71.39%), demonstrating the effectiveness of SHAP-guided perturbations. To verify that adversarial binaries remain valid, we randomly selected 100 generated samples and submitted them to the Any.Run sandbox for dynamic analysis. All samples executed successfully, confirming that our framework preserves core malicious functionality while evading detection.

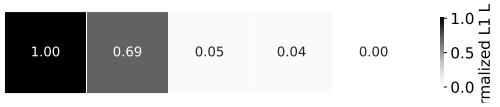

Fig. 2. Top5 L1 Loss between Target Features and Generated Executables

Although executability is preserved, we observe a discrepancy between the adversarial features designed in feature space and those re-extracted from the crafted executables. While adversarial features achieved an ASR of 87.2%, the ASR measured on the corresponding binaries dropped to 55.14%. As shown in Figure 2, most bytes are reconstructed with minimal error, but some target values deviate significantly when the required perturbation is too large to realize without excessive overhead. These reconstruction gaps directly reduce ASR when perturbations are translated into executable form.

Overall, our evaluation demonstrates that SHAP-guided perturbations improve ASR over prior work and preserve executability, but also highlights a fundamental challenge: large feature gaps cannot always be faithfully reconstructed without compromising binary integrity. This trade-off underscores the importance of aligning feature-level perturbations with feasible executable-level modifications in order to assess adversarial threats realistically.

IV. Conclusion and Future work

We introduced an LLM-guided framework for feature-to-executable adversarial malware generation. Our approach combines SHAP-based feature selection with LLM-driven synthesis, achieving higher ASR than prior work while preserving executability. As future work, we aim to improve feature-to-binary fidelity and exploring methods to predict original feature representations from hashed features.

References

[1] Abdullah Al-Dujaili, Alex Huang, Erik Hemberg, and Una-May O'Reilly. Adversarial deep learning for robust detection of binary encoded malware. In *IEEE Security and Privacy Workshops (SPW)*, 2018.

[2] Hyrum S Anderson and Phil Roth. Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637*, 2018.

[3] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. Transcending transcend: Revisiting malware classification in the presence of concept drift. In *IEEE Symposium on Security and Privacy (S&P)*, 2022.

[4] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 1 2020.

[5] Wei Song, Xuezixiang Li, Sadia Afroz, Deepali Garg, Dmitry Kuznetsov, and Heng Yin. Mab-malware: A reinforcement learning framework for blackbox generation of adversarial malware. In *Proceedings of the ACM on Asia conference on computer and communications security (AsiaCCS)*, 2022.

[6] VirusShare. Virusshare, 2025.

[7] VirusTotal. Virustotal: Analyse suspicious files, domains, ips and urls to detect malware and other breaches, automatically share them with the security community., 2025.