From Large Language Models to Adversarial Malware: How far are we

Shuai He
optimus_he@hust.edu.cn
Huazhong Science and technology University
Wuhan, Hubei, China

Hao Yan
u202112208@hust.edu.cn
Huazhong Science and technology University
Wuhan, Hubei, China

Wenke Li
winkli1@hust.edu.cn
Huazhong Science and technology University
Wuhan, Hubei, China

Sheng Hong
hongsheng@hust.edu.cn
Huazhong Science and technology University
Wuhan, Hubei, China

Xiaowei Guo
d202280650@hust.edu.cn
Huazhong Science and technology University
Wuhan, Hubei, China

Xiaofan Liu
xiaofanliu@hust.edu.cn
Huazhong Science and technology University
Wuhan, Hubei, China

Cai Fu*
fucai@hust.edu.cn
Huazhong Science and technology University
Wuhan, Hubei, China

Abstract

Large Language Models (LLMs) have achieved notable progress in fields including natural language processing, cyber threat detection, and automated penetration testing, increasingly being applied in practical settings. However, the rapid advancement of these models has also led to their potential misuse, posing new challenges to cyberspace security. Security incidents have already been reported in areas such as phishing attacks and disinformation campaigns. Nevertheless, the progress and potential impact of LLMs in generating adversarial malware remain underexplored. This study systematically investigates the evasion capability of adversarial malware generated by LLMs. By integrating chain of thought into a Markov process and designing prompt based state transition functions and reward mechanisms, this research evaluates the effectiveness and efficiency against mainstream static detection methods on a dataset comprising over 2,000 real-world malware samples. Experimental results demonstrate an average evasion rate of 89.92% across 12 commercial antivirus engines on VirusTotal. The findings reveal that individuals with minimal technical expertise and basic natural language skills can generate malware that evades static detection, which underscores potential vulnerabilities in current cyberspace defense and detection systems regarding adversarial malware countermeasures.

CCS Concepts

• **Security and privacy → Malware and its mitigation**.

Keywords

large language models, adversarial malware, static detection

*Corresponding author.

ACM Reference Format:
Shuai He, Hao Yan, Wenke Li, Sheng Hong, Xiaowei Guo, Xiaofan Liu, and Cai Fu. 2025. From Large Language Models to Adversarial Malware: How far are we. In *34th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA Companion '25), June 25–28, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3713081.3732928

1 Introduction

Anti-malware is the front-line of cyberspace security, where mutated malware detection has always been a long-term challenge for anti-virus software. According to *SonicWall Cyber Threat Report* [1], malware attacked worldwide users over 2.8 billion times during the first half of 2022, of which 36.5% are mutated malware. Regardless of the specific tactics employed in malicious campaigns, they typically rely on malware to infiltrate systems and achieve objectives such as data exfiltration or system disruption.

As the number of malicious software increased in the last years to hundreds of thousands of new malware samples daily [1], security technologies can no longer rely solely on virus signatures, instead, they have increasingly adopted heuristics and machine learning techniques for over a decade [2]. However, relying exclusively on machine learning poses significant risks, as machine learning models are inherently vulnerable to adversarial attacks, where indistinguishable perturbations can lead to incorrect predictions [3]. Recent research has demonstrated the vulnerability of neural network-based malware detectors to both white-box [4] and black-box adversarial attacks [5]. The challenge of generating adversarial malware lies in balancing efficiency, stealth, and adaptability. Existing adversarial attack methods often rely on gradient-based or genetic optimization techniques [6], which require a large number of queries to generate effective adversarial samples. This inefficiency conflicts with the rapid growth and scalable nature of real - world malware scenarios. Additionally, while adversarial training has been proposed as an effective defense mechanism, it suffers from limitations in generalization. As highlighted by Keane et al. [7], retraining a model on a single adversarial attack does not ensure robustness against other attack variants. Therefore, novel adversarial attacks need to be proposed to facilitate the data generation

[1]https://www.sonicwall.com/medialibrary/en/white-paper/mid-year-2022-cyber-threat-report.pdf

for adversarial training. Another emerging challenge lies in the rapid development of LLMs, whose potential impact on adversarial malware remains underexplored. As these models grow in complexity and capability, they could introduce new vulnerabilities or opportunities for adversarial manipulation, yet their implications for cybersecurity have not been thoroughly investigated.

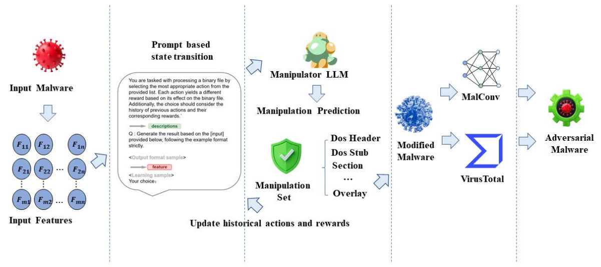
To mitigate the above challenges, this paper proposes a novel method for generating adversarial samples against static malware detection leveraging LLMs. By modeling the problem of generating adversarial samples for static malware detection as a Markov process and utilizing LLMs to guide its optimization, our approach efficiently generates adversarial samples in black-box scenarios without any model training. To ensure the functional integrity of the original files, we introduce 7 independent manipulations based on the characteristics of the PE file structure. By combining these manipulations and setting the optimization objective to the shortest evasion sequence, our method achieves an evasion rate of 98.62% against the MalConv detector, requiring only 8,808 queries and a relative file size increase of 0.46. Futhermore, we conduct evaluation on real-world integrated malware detector VirusTotal, the adversarial malware samples can evade 12 vendors with 89.92% average attack success rate, demonstrating potential threats to anti-virus vendors. The core contributions of this paper are as follows:

- This paper introduces an end-to-end adversarial attack framework leveraging LLMs to simulate a Markov process, which facilitates the efficient generation of adversarial malware samples while maintaining their original functionality. By removing the dependency on model training, which is a necessary step in existing methods, our approach presents a more efficient adversarial sample generation framework, offering an alternative to traditional paradigms.
- Our approach demonstrates competitive evasion rates against static malware detector with significantly fewer queries and smaller file size compared to existing methods.
- The findings highlight the potential risks posed by LLM-assisted adversarial malware generation in real-world malware detection systems, underscoring the need for robust defensive mechanisms.

2 Related Work

Previous approaches design attacks from the perspective of adversarial attacks [8], which can be divided into two categories:

White-box attack. Those attacks require prior knowledge of the target model such as model gradient and model architecture. [9] proposes three distinct attacks: *Full DOS*, *Extend*, and *Shift*. The *Full DOS* attack manipulates 58 bytes in the DOS header and the entire DOS stub. The *Extend* attack enlarges the DOS header and injects adversarial content, while the *Shift* attack injects content before the first section. Meanwhile, [10] focuses on manipulating the names of each section in the Optional Header. Additionally, [11] manipulates 58 bytes in the DOS header (from MZ to the offset to the PE header), guided by the integrated gradients method. In a different approach, [12] enhanced padding manipulations by padding in both the section unused space and overlay, replacing the optimizer with an iterative variant of the fast gradient sign method (FGSM) [3]. This method was originally proposed for image adversarial attacks and



You are tasked with processing a binary file by selecting the most appropriate action from the provided list. Each action yields a different reward based on its effect on the binary file. Additionally, the choice should consider the history of previous actions and their corresponding rewards.'

Q : Generate the result based on the [input] provided below, following the example format strictly.

Your choice:

Figure 1: Schematic diagram of LLMs-based adversarial malware generation.

has been adapted for malware manipulation. Finally, [13] conducted an experiment on MalConv, which pads embedding values at the overlay of the PE file guided by the positive direction of the gradient. This approach achieved a 60% attack success rate with an average of 10,000 bytes modified per sample.

Black-box attack. In this scenario, attacks can only acquire input and output of the target model, thus generative methods are often used. [5] proposes a generative adversarial network that utilizes the the feature extracted from API calls to generate adversarial malware. However this work did not provide a method to convert the adversarial feature vectors back to real-world binaries. [14] minimizes the malware score using the same manipulations as those in [10]. But [14] employs a genetic algorithm to optimize the chain of manipulations and utilizes a sandbox to discard malware whose functionality is broken. Similarly, Song et al. [15] also use a packer and a set of manipulations, but they rely on a multi-armed bandit (MAB) as the optimizer. Their experiments require 20 servers to conduct the manipulations. In a different approach, [16] propose two individual attacks. One attack conducts manipulations through a set of operations in the section and a packer. The other attack combines *Full DOS*, *Extend* and *Shift*, with the manipulation optimization relying on a genetic algorithm.

Our objective is to explore the capability of LLMs in generating adversarial malware, aiming at the advancing sequential optimization capability of LLMs. Initially, we model the evasion to a Markov decision process (MDP). Subsequently, we integrate the binary file manipulations to a condensed manipulation set, demonstrating its effectiveness in generating adversarial malware samples.

3 METHODOLOGY

Overview. We focus on generating adversarial malware efficiently. Since the defense models develop rapidly, training target-exclusive adversarial malware generator can be demanding on computational resource. Hence, to mitigate the challenge, in this work, we investigate the capabilities of LLMs in generating adversarial malware, which is composed of two parts: (i) file space manipulations, which determine which part of malware bytes will be modified. (ii) Prompt based state transition, which utilizes Markov-based context to predict the next manipulation. The workflow the proposed method is shown in Fig. 1.

3.1 File Space Manipulation

To embed malicious behaviors within different manipulations. According to the characteristic PE file, 7 independent manipulations are introduced: (i) Apply random values to 58 modifiable bytes within the DOS Header. (ii) Extending the DOS-stub with randomly generated bytes. (iii) Inject random bytes into unused space at the end of each section. (vi) Create a section after the last section and inject bytes that are extracted from benign binary's section. (v) Create a section after the last section and inject random bytes. (vi) Padding bytes that are extracted from benign binaries to the overlay. (vii) Padding random bytes to the file overlay.

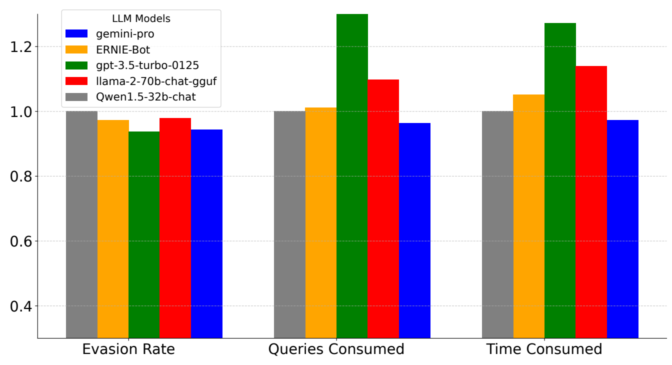


Figure 2: Comparison of different LLMs on evasion performance.

3.2 Prompt based state transition

To balance output precision and informational completeness, we design a four-tiered state transition strategy in the prompt:

Base Task Layer. The foundational task is defined as follows: "Your task is processing a binary file by selecting the most suitable action from the provided list. Each action will be associated with a reward, determined by its impact of the file and execution sequence. Furthermore, the selection of an action should consider the history of prior actions and their respective rewards." Then we explain the specific meaning of each action, as detailed in Section 3.1.

Constraint Layer. Preliminary experiments indicate that unconstrained settings often lead to redundant outputs. We observed that the "output-only-action-index" instruction significantly enhances the validity of the generated results. To mitigate ambiguous outputs while ensuring the model delivers concise and meaningful information, we designed the prompt as follows: "Only output the name of the selected action." Additionally, we incorporated the instruction "Do not give any extra information or reasons," based on insights from our preliminary analysis.

Information Fusion Layer. Contains a two-dimensional feature integration. The first one is structured metadata, which is extracted from PE headers (COFF/Machine Characteristics), optional headers (Subsystem/DLL flags), and section attributes. The other is dynamic context, which contains a sliding window (size=2) of historical file features, actions and rewards, the reward corresponds to the difference between the initial malicious score and the score observed during the mutation process.

Contrastive Context Layer. This component extracts triplets composed of file features, actions, and rewards from historical evasion samples, which are used to simulate few-shot learning scenarios and enhance the effectiveness of large language models in predicting actions. These evasion samples are generated by an LLM through a prompt scheme without a contrastive context layer. To minimize token consumption during LLM interactions, this section adopts a modular prompt design, where action and reward lists are directly updated in each interaction, rather than relying on the LLM's memory function to gradually expand these lists.

To mitigate the impact of LLM hallucinations on the decision making process, we set the *max_tokens* parameter to 1. This configuration compels LLMs to solely predict the next action. Full details are publicly accessible in our open-source repository[2].

4 EXPERIMENTS

4.1 Experimental Setup

Dataset. We collected experimental data from a real-world malware dataset VirusShare[3]. Specifically, we extracted over 10,000 malware samples from the VirusShare_001 collection. We employed MalConv [13] to screen and retain only those samples identified as malicious. Subsequently, we randomly selected 2,036 malware samples as the experimental dataset as outlined in [17].

Experimental environment. Our implementation using PyTorch 1.8.2 and the stable-baselines3 library. The PEfile library was used to modify the bytes of PE files, with the corresponding Python version being 3.6.2. Additionally, we utilized the ToucanStrike library to reproduce related adversarial attacks. Experiments were conducted on a Windows server equipped with a 2.9 GHz Intel Xeon Gold 6326R CPU, 2 × 80GB NVIDIA A100 Tensor Core GPU GPUs, and 256GB of memory.

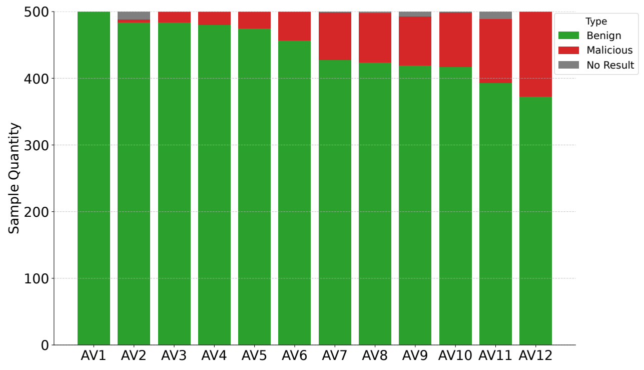


Figure 3: Comparison of different LLMs on evasion rate.

Evaluation metrics. We employed four widely-used metrics: Evasion rate (ER), number of queries, evasion cost, and time consumption. Evasion rate refers to the efficiency of the method in

[2]https://github.com/Optimus-He/From-Large-Language-Models-to-Adversarial-Malware-How-far-are-we
[3]https://virusshare.com

generating adversarial malware, which is calculated by the proportion of malware in the dataset that evades detection. The number of queries indicates how many times the target model is queried during the generation of adversarial malware. This is a crucial metric in practice, as excessive queries may be blocked by real-world antivirus software. The evasion cost α is the average increase in size of the modified malware relative to the original size, showing how many bytes are injected during the generation process.

4.2 Internal Parameter Analysis

To systematically evaluate the impact of adversarial perturbation parameters on evasion effectiveness, we first conduct parameter sensitivity analysis. As shown in Table 1 and 2, we focus on two critical hyperparameters: the evasion cost α and maximum query budget. We randomly selected 200 malware samples from the dataset, repeated each experiment three times, and report the mean results.

Table 1: Performance of the proposed method under varying query limit.

Maximum-query	ER(%)	Queries	Time
1	43.50	457	52mins20s
2	76.62	642	67min39s
3	94.03	663	58mins1s
4	94.53	635	66mins41s
5	97.01	493	62mins33s
10	**99.00**	682	70mins25s
15	**99.00**	714	76mins50s

To explore the impact of different α values on the experimental results, we gradually increased α from 0.1 and monitored the evasion rate. The findings indicate that our method achieves the optimal evasion rate when α reaches 2, after which the evasion rate stabilizes. Consequently, we set α to 2 for subsequent experiments. Similarly, in the maximum query experiment, we observed that the evasion rate peaks when the maximum query count reaches 10.

To investigate the impact of LLMs on evasion performance, we conduct cross-model comparative experiments. As illustrated in Fig. 2, using the top-performing Qwen-1.5-32B as the benchmark, we normalize three core evaluation metrics (evasion rate, query consumed and time consumed) into relative performance. Result reveals Qwen achieves highest evasion rate while maintaining less query and time consumed, which justifies our selection of Qwen as the foundational generator for subsequent experiments.

Table 2: Performance of the proposed method under varying evasion cost α.

Maximum-α	ER(%)	Queries	Time
0.1	79.00	540	61mins16s
0.5	85.00	576	58min27s
0.8	93.00	638	63mins10s
1.0	93.00	616	60mins48s
1.5	95.00	632	67mins25s
2.0	**96.50**	655	69mins33s
2.5	**96.50**	673	72mins33s

Table 3: Comparisons with related methods against MalConv malware detectors.

Methods	ER(%)	Queries	Evasion Cost	Functionality
This work	98.62	**8808**	0.46	**100%**
Full DOS	41.29	38737	0.15	98.50%
Partial DOS	50.26	41907	**0.12**	27.07%
Extend	88.00	34666	3.81	72.00%
Shift	99.22	30789	3.78	97.00%
Padding	83.24	30251	3.73	100%
Header Fields	48.58	38984	0.27	100%
GAMMA Section	94.15	32295	0.50	95.00%
GAMMA Padding	88.85	28489	0.33	99.00%
MAB	82.00	22203	3.56	92.00%
Aimed	56.40	73624	0.76	100%

4.3 Effectiveness Analysis

To comprehensively evaluate the method efficacy, we conduct comparative experiments with 10 state-of-the-art adversarial attacks using 2,036 malware samples. All methods are reproduced through the benchmark from [18]. As shown in Table 3, our method, with a 98.62% evasion rate, ranks second. However, the *Shift* method can not preserve the functionality of input samples (verified via sandbox executability checks). Although *Shift* evades 14 more samples, it requires 21,981 more queries, which is critical in practice, as too many queries may be blocked by real-world antivirus software.

To assess the real-world threat potency of our method against commercial antivirus systems, we conduct a validation experiment on VirusTotal. 500 adversarial samples that successfully evaded MalConv detector are randomly selected and submitted to VirusTotal for large-scale scanning. Based on vendors' technical whitepapers, we identify 12 antivirus products explicitly employing ML-based static analysis (excluding solutions relying on dynamic behavior detection), which are anonymized as AV1-AV12. Fig. 3 presents the detection results of the 12 antivirus vendors. Against AV1, misclassified all adversarial samples as benign (100% evasion rate), the vendor while the best-performing AV12 still showed 80.36% false negative rate after excluding 11 timeout cases. Cross-vendor analysis reveals an average evasion rate of 89.92%. The findings not only validate the practical effectiveness of our method, but also expose systemic vulnerabilities in commercial detection systems when facing LLM-powered adversarial attacks.

5 Conclusion and Future Work

We investigate the capabilities of LLMs in generating adversarial malware samples. Through introducing 7 functionality-preserving manipulates and modeling the evasion optimization as a Markov process, we show this method achieves a 98.62% evasion rate against MalConv and evades 12 real-world antivirus engines on VirusTotal with 89.92% evasion rate. However, in the current implementation it does not work equally well on different malware detectors.

5.1 Future Work

A promising research direction is constructing robust detection methods for adversarial samples. An approach we are exploring involves applying ensemble learning methods. This framework aims to enhance detection accuracy by integrating classification insights across malware and adversarial sample categories, thereby improving the system's robustness against sophisticated attacks.

References

[1] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. 2017. A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 1–40.

[2] Deqiang Li and Qianmu Li. 2020. Adversarial deep ensemble: Evasion attacks and defenses for malware detection. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3886–3900.

[3] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9* (2015).

[4] Felix Kreuk, Assi Barak, Shir Aviv-Reuven, Moran Baruch, Benny Pinkas, and Joseph Keshet. 2018. Adversarial examples on discrete sequences for beating whole-binary malware detection. *arXiv preprint arXiv:1802.04528* (2018), 490–510.

[5] Weiwei Hu and Ying Tan. 2017. Generating adversarial malware examples for black-box attacks based on GAN. *arXiv preprint arXiv:1702.05983* (2017).

[6] Octavian Suciu, Scott E Coull, and Jeffrey Johns. 2019. Exploring adversarial examples in malware detection. In *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 8–14.

[7] Lucas Keane, Pai Samruddhi, Lin Weiran, Bauer Lujo, Reiter Michael K., and Sharif Mahmood. 2023. Adversarial Training for Raw-Binary Malware Classifiers. In *32th USENIX Security Symposium (USENIX Security 23)*.

[8] Shuai He, Cai Fu, Hong Hu, Jiahe Chen, Jianqiang Lv, and Shuai Jiang. 2024. MalwareTotal: Multi-Faceted and Sequence-Aware Bypass Tactics Against Static Malware Detection. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. 2123–2134. doi:10.1145/3597503.3639141

[9] Luca Demetrio, Scott E Coull, Battista Biggio, Giovanni Lagorio, Alessandro Armando, and Fabio Roli. 2021. Adversarial exemples: A survey and experimental evaluation of practical attacks on machine learning for windows malware detection. *ACM Transactions on Privacy and Security (TOPS)* 24, 4 (2021), 1–31.

[10] Hyrum S Anderson, Anant Kharkar, Bobby Filar, and Phil Roth. 2017. Evading machine learning malware detection. *black Hat* 2017 (2017).

[11] Luca Demetrio, Battista Biggio, Lagorio Giovanni, Fabio Roli, Armando Alessandro, et al. 2019. Explaining vulnerabilities of deep learning to adversarial malware binaries. In *CEUR WORKSHOP PROCEEDINGS*, Vol. 2315.

[12] Adeilson Antonio da Silva and Mauricio Pamplona Segundo. 2022. On deceiving malware classification with section injection. *preprint arXiv:2208.06092* (2022).

[13] Bojan Kolosnjaji, Apostolis Zarras, George Webster, and Claudia Eckert. 2016. Deep learning for classification of malware system call sequences. In *Australasian joint conference on artificial intelligence*. Springer, 137–149.

[14] Raphael Labaca Castro, Sebastian Franz, and Gabi Dreo Rodosek. 2021. AIMED-RL: Exploring Adversarial Malware Examples with Reinforcement Learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 37–52.

[15] Wei Song, Xuezixiang Li, Sadia Afroz, Deepali Garg, Dmitry Kuznetsov, and Heng Yin. 2022. MAB-Malware: A Reinforcement Learning Framework for Blackbox Generation of Adversarial Malware. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. 990–1003.

[16] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Fabio Roli, and Alessandro Armando. 2021. Functionality-preserving black-box optimization of adversarial windows malware. *IEEE Transactions on Information Forensics and Security* 16 (2021), 3469–3478.

[17] Javier Yuste, Eduardo G Pardo, and Juan Tapiador. 2022. Optimization of code caves in malware binaries to evade machine learning detectors. *Computers & Security* 116 (2022), 102643.

[18] L. Demetrio. 2022. *Toucanstrike*. https://github.com/pralab/toucanstrike