Poster: Leveraging LLMs to Generate and Counter Sandbox Evasion Techniques

Anh Nguyen Van*, Matsuzawa Hikaru*, Ito Hikaru*, Yin Minn Pa Pa*, Rui Tanabe*, Katsunari Yoshioka*
*Yokohama National University
Email: {nguyen-anh-xd,matsuzawa-hikaru-nx,ito-hikaru-gd,yinminn-papa-jp,tanabe-rui-xj,yoshioka}@ynu.ac.jp

***Abstract*—Sandboxes are essential in malware analysis, enabling the safe observation of unknown programs. However, malware authors have been trying to evade sandboxes. Developing stealthier sandboxes has become critical; yet, existing methods heavily rely on expert knowledge, with proofs-of-concept (PoCs) and counter-strategies developed on a case-by-case basis, often lacking systematic automation. In this study, we leverage large language models (LLMs) to automate the generation of evasion PoCs and counter-measures for sandbox evasion techniques. We introduce AutoEvasion-Defense (AED): an LLM-driven sandbox evasion co-evolution framework. Our framework consists of an Attacker Module, which collects sandbox evasion techniques from existing sources and generates corresponding PoCs, and a Defender Module, which analyzes these PoCs to automatically generate sandbox updates. Utilizing 122 generated PoCs, we show that our updated sandbox surpasses conventional public sandboxes. Demonstrating a scalable approach to stay ahead of the arms race between malware authors and sandbox developers.**

***Index Terms*—Sandbox Evasion, LLM, Code Generation**

I. Introduction

Sandboxes have become the *de facto* standard for automated malware analysis. However, malware authors increasingly employ sandbox evasion techniques to bypass these environments. Prior studies indicate that 40% of analyses terminate prematurely in commercial or default sandboxes, and nearly 80% of malware samples exhibit at least one evasive behavior [1]. These limitations undermine the efficacy of automated analysis and are further extended by the emerging use of AI to accelerate the development of malware code.

To address these challenges, researchers have focused on developing stealthier sandboxes. Methods to remove sandbox artifacts or fingerprints have been discussed. Sandbox detection tools and industry reports reveal several evasion techniques, and the MITRE ATT&CK framework catalogs defenses against such attacks. In practice, yet updating sandboxes typically relies on expert knowledge or requests to vendors, resulting in high costs and limited scalability—especially when evasion techniques are mostly described in natural language sources such as analysis reports or academic papers.

In this study, we propose a framework designed to bridge the gap between high-level conceptual attack descriptions and actionable, concrete proof-of-concept (PoC) implementations. To the best of our knowledge, this is the first work to leverage Large Language Models (LLMs) to dynamically update sandboxes in response to the ongoing cat-and-mouse game. By semi-automating the sandbox update process, our approach aims to reduce the manual effort and enhance scalability.

II. LLM-driven Sandbox Evasion Co-evolution Framework

We introduce AutoEvasion-Defense (AED): an LLM-driven sandbox evasion co-evolution framework that consists from two different modules: *Attacker Module* and *Defender Module* (Fig. 1). Both modules use LLMs from the *GPT-4o* API.

A. Attacker Module

The *Attacker Module* generates evasion PoCs from high-level descriptions of existing literature, producing both source code and binaries. The *Attacker Module* comprises four stages.

1) We first generate evasion code snippets by providing natural-language evasion ideas (e.g., timing checks, network filtering) to an LLM-assisted Code Generator.
2) The generated snippet is integrated into a predefined PoC base template to better mimic malware behavior. The template includes logic that terminates execution in sandbox environments while executing a payload on real user systems. For evaluation, we use benign actions (e.g., file creation, registry modification) as payloads.
3) The PoC source codes are compiled into binaries. For codes that failed to compile, we exclude them.
4) Each binary is executed on both user machines and sandboxes. A PoC is deemed successful if it executes the payload on a user machine but is prevented from running in sandboxes due to evasion. The successful PoC samples are then used as input to the *Defender Module*.

B. Defender Module

The *Defender Module* semi-automatically updates the target sandbox by leveraging the evasion PoCs to generate counter-measures. The *Defender Module* comprises three stages.

1) *Inferrer:* An LLM model analyzes PoC source codes to extract evasion techniques and the associated thresholds. The separation of the *Inferrer* from the *Updater* allows for extensions to support analysis of malware binaries.
2) *Updater:* Generates sandbox updates based on inferred evasion techniques. Prompts to the *Updater* LLM include (i) sandbox configuration details, (ii) output format constraints, and (iii) sandbox code context. The generated updates fall into three categories: step-by-step manual instructions for human operators—constrained to minimal, easily understandable operations (e.g., opening GUI windows and filling text fields); configuration

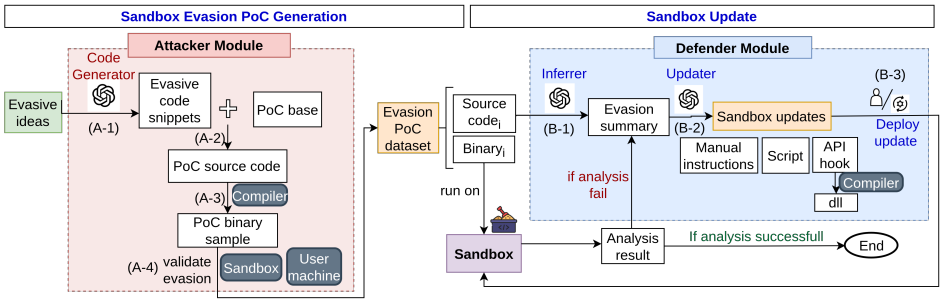


Fig. 1. Overview of AutoEvasion-Defender (AED) Framework

scripts (e.g., PowerShell scripts for virtual machine setup); and API hooks for sandbox injection.

3) *Deploy:* Step-by-step instructions are manually followed by human operators as provided, without consulting external sources and within a restricted timeframe. Scripts are applied semi-automatically by human operators using a standardized execution procedure, ensuring operator-independent results. API hooks are compiled into *.dll* files and automatically injected into the sandbox. An update is considered successful when deployment completes and evasive binaries can be analyzed.

III. Evaluation

A. Sandbox Evasion PoC Generation Results

Using 20 evasion techniques from Check Point [2] as input to the *Attacker Module*, we generated 105 C++ binaries that compiled using MS Visual Studio. Each technique has multiple variants using different APIs and implementations. Among the compiled binaries, 61 were confirmed to evade sandbox analysis; forming the *AI-generated Evasion PoC Dataset*.

B. Sandbox Update Result

For the target sandbox, we deployed a *CAPE sandbox* with a *Windows 10 Home* guest running on *QEMU/KVM*. We evaluated the sandbox using 61 binaries generated by *Attacker Module*, together with 61 binaries from the well-known sandbox detection tool *Pafish* [3].

Table I shows that a substantial portion of evasive PoC samples can be mitigated by *Defender Module*. A single evasion technique may be addressed by multiple update methods. API hooks are particularly effective against *stalling* evasions, while instructions and scripts successfully counter *usage-mark* evasions (e.g., mouse activity or recycle bin checks). In *system environment*, failures stem from incorrect Windows paths or system details, which is expected because explicit Windows system knowledge is not provided to the LLM. *Timing* evasions, which compare execution time against sleep duration, remain unhandled and are left for future work.

Applying these updates, we deployed a customized sandbox (*AI-enhanced CAPE*). As shown in Table II, it successfully analyzed 110 of 122 samples (90%), outperforming both the initial installation (*Default CAPE*) and *VirusTotal CAPE*.

TABLE I
Sandbox Update Results by Evasion Category

Evasion category	# binaries require updates / # binaries	# successful updates / # generated updates		# successful binaries / # binaries require updates
		Manual (instructions)	Semi-auto (scripts & API hooks)	
AI-Generated Evasion PoC Dataset				
Human Interaction	1/4	0/0	1/1	1/1
Stalling	3/3	0/0	3/3	3/3
System Environment	15/27	3/7	5/11	8/15
Timing	1/2	0/0	0/1	0/1
Usage Marks	17/25	2/2	14/17	16/17
Total	**37/61**	**5/9**	**23/33**	**28/37**
Pafish [3]				
Human Interaction	3/6	0/3	1/3	1/3
System Environment	3/22	2/3	0/3	2/3
Other categories	0/33	–	–	
Total	**6/61**	**2/6**	**1/6**	**3/6**

binaries requiring updates: PoC binaries failing initial CAPE analysis.
generated updates: Updates generated by the LLM.
successful updates: Updates enabling successful analysis.
successful binaries: Binaries analyzed successfully after sandbox updates.

TABLE II
Comparison on Analysis Success Results of Sandboxes

	# binaries	VirusTotal CAPE	Default CAPE	AI-enhanced CAPE
AI-generated Evasion PoC	61	23	24	52
Pafish [3]	61	54	55	58
Total	**122**	**77 (63%)**	**79 (65%)**	**110 (90%)**

IV. Conclusions and Future Work

We propose an LLM-assisted framework that jointly models the generation of evasive malware PoCs and the enhancement of sandbox defenses against evasion. Our approach significantly improves sandbox effectiveness while reducing reliance on expert analysts and manual implementation. By advancing both components, the framework enables the co-evolution of evasion technique databases and sandbox defenses. On the attacker side, we plan to extend evasion inputs to real-world malware reports and the literature. On the defender side, we will refine the update process, extend support to sandboxes beyond CAPE, and evaluate on real malware datasets.

Acknowledgements. This paper is based on results obtained from a project, JPNP24003, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] G. et. el., "A systematical and longitudinal study of evasive behaviors in windows malware," *Computers & security*, vol. 113, p. 102550, 2022.
[2] C. Point, "Sandbox Evasions Resource Portal," https://evasions.checkpoint.com, 2024, accessed: 2025-07-28.
[3] A. Ortega, "PaFish," https://github.com/a0rtega/pafish, 2024, accessed: 2025-07-28.

YNU YOKOHAMA National University

Poster: Leveraging LLMs to Generate and Counter Sandbox Evasion Techniques

Anh Nguyen Van*, Matsuzawa Hikaru*, Ito Hikaru*, Yin Minn Pa Pa*, Rui Tanabe*, Katsunari Yoshioka*

*Yokohama National University

Email: {nguyen-anh-xd, matsuzawa-hikaru-nx, ito-hikaru-gd, yinminn-papa-jp, tanabe-rui-xj, yoshioka}@ynu.ac.jp

1 Introduction

Problems

- Sandboxes are essential but easy to evade
- **Evasion is widespread [1]**
  - 40% of analyses terminate prematurely
  - ~80% of malware exhibits evasive behavior
- **Current sandbox updates are manual and costly**, relying on expert knowledge or vendor support

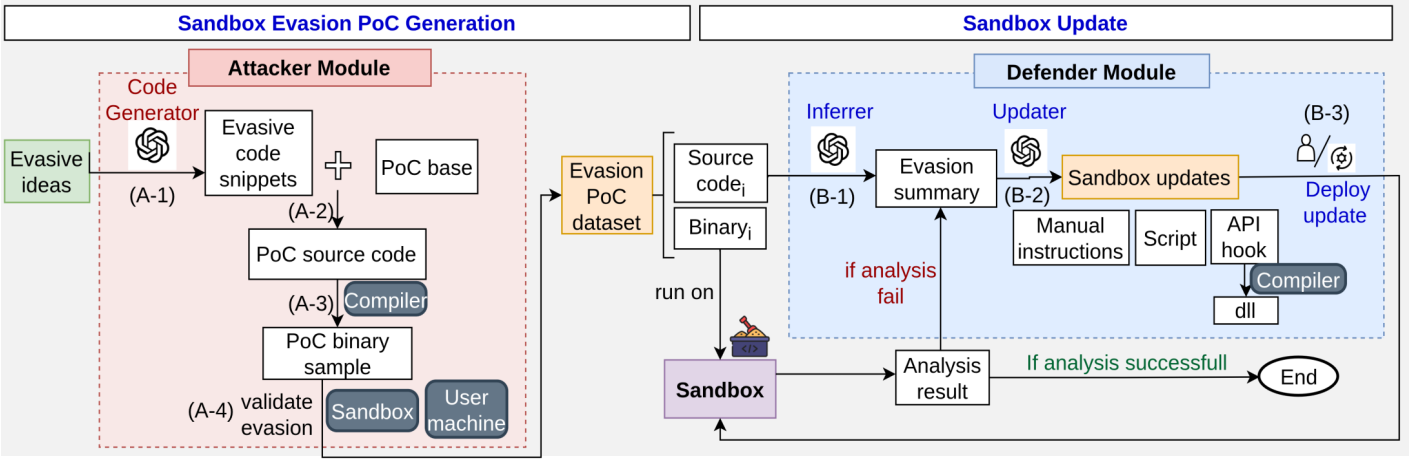
Proposal

AutoEvasion-Defense (AED)

An LLM-driven sandbox evasion co-evolution framework

- **Attacker Module:** Automatically generates sandbox-evasive PoC code from evasion descriptions
- **Defender Module:** Generates countermeasures and deploys sandbox updates

2 Framework Overview



3 Evaluation

Sandbox Evasion PoC Generation Result

20 evasion descriptions in Checkpoint report [3]

↓

Attacker Module

↓

AI-generated Evasion PoC Dataset

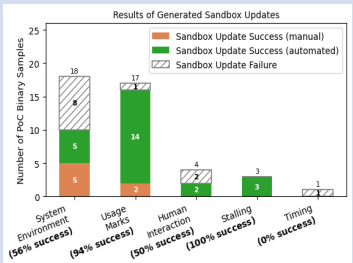
Evasion Category	System Env	Usage Marks	Human Interaction	Stalling	Timing
Generated PoCs	27	15	4	3	1

61 sandbox-evasive C++ PoCs generated

- Uses varied APIs and implementations
- Each PoC evades at least 1 of 5 commercial sandboxes

Sandbox Enhancement Result

AI-generated Evasion PoC Dataset → Defender Module ← Pafish Dataset [2]



Successfully update 28/37 evasions that evade default CAPE

- **Usage-mark evasions** are countered
- API hooks handle **stalling**
- **System environment evasions** show partial coverage
  - failures due to LLM missing Windows details (e.g., paths)
- **Timing evasions** remain unhandled

+ manual: an operator followed LLM generated instructions
+ automated: updated by generated scripts and API hooks

→ Successful Updates

Proposed sandbox

	VirusTotal cape	Default cape	AI-enhanced cape
Analysis success rate	63% (77/122)	65% (79/122)	**90% (110/122)**

4 Conclusions and Future Work

- **LLM-assisted framework** for evasive PoC generation and sandbox updates
- **Co-evolution** of evasion datasets and sandbox defenses
- **Future Work:** Extend evasion inputs to real malware and update to other sandboxes beyond CAPE

References

[1] G. et. el., "A systematic and longitudinal study of evasive behaviors in windows malware," Computers & Security, vol. 113, p. 102574, 2022.
[2] A. Ortega, "PaFish" github.com/a0rtega/pafish, 2024.
[3] C. P, "Sandbox Evasions Resource," evasions.checkpoint.com, 2024.

Acknowledgements

This paper is based on results obtained from a project, JPNP24003, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).