# Generating Malware Using Large Language Models: A Study on Detectability and Security Barriers

**Gustavo Lofrese Carvalho¹, Ricardo de la Rocha Ladeira¹, Gabriel Eduardo Lima²**

¹Instituto Federal Catarinense – Campus Blumenau – Blumenau/SC – Brasil
²Universidade Federal do Paraná – Curitiba/PR – Brasil

gustavolc06@gmail.com ricardo.ladeira@ifc.edu.br, gelima@inf.ufpr.br

***Abstract.*** *Large Language Models are Artificial Intelligence systems capable of processing natural language inputs to produce contextualized and coherent outputs. This work investigates their potential to generate malware from non-technical prompts. Controlled experiments were conducted with ChatGPT, Gemini, and Copilot, following attack templates from recent literature and analyzing the outputs through VirusTotal. Across 36 tests, five resulted in functional and undetectable code, representing a 13.89% success rate. These findings reveal that minimal prompt variations can bypass safety mechanisms and evade automated detection. Despite existing defenses, LLMs-based systems remain vulnerable to iterative prompt engineering, reinforcing the need for stronger semantic validation and multi-layered protection strategies.*

## 1. Introduction

Large Language Models (LLMs) are Artificial Intelligence systems based on deep neural networks, designed to understand and generate natural language in a coherent and contextualized manner. The ability of these models to automate complex tasks, such as code generation from natural language descriptions, has expanded their role across technological domains, with a direct impact on information security [Gupta *et al*. 2023]. This capability to translate natural language into executable instructions enhances the potential for automation but also introduces risks in the cybersecurity area.

Although malicious code predates LLMs [Cani *et al*. 2014], the accessibility and flexibility of these systems have lowered the technical barrier and enabled non-expert users to develop harmful software. Earlier malware automation tools required specialized knowledge [Cani *et al*. 2014], but now the models can be induced to generate and refine malicious programs iteratively and autonomously, even through non-technical natural language inputs [Carvalho, Ladeira and Lima 2025].

An example of this type of vulnerability occurred in 2024, when an autonomous Artificial Intelligence agent was convinced to transfer approximately 47,000 USD in a public challenge after hundreds of attempts, through carefully crafted messages [Lindrea 2024]. This episode shows that the vulnerabilities are not limited to malware generation but extend to behavioral manipulation and model-level exploitation.

This study investigates the following research question: can natural language inputs, without technical details, lead the latest LLMs to produce malware undetectable by security tools? The main objective is to evaluate this capability of LLMs despite the implemented security restrictions. The specific objectives are: (i) to identify and test the defense mechanisms present in different LLMs; (ii) to measure the number of interactions required to obtain functional and stealthy code; and (iii) to evaluate the effectiveness of automated detection tools, such as the VirusTotal[1] platform.

## 2. Related Works

This section is organized into three subsections: (i) jailbreak attacks, (ii) jailbreak defenses, and (iii) malware generation using LLMs.

### 2.1 Jailbreaking Attacks

When examining attempts to bypass safety mechanisms, Liu *et al*. (2024) conducted a comprehensive study of jailbreak techniques applied to ChatGPT. The authors proposed a taxonomy of jailbreak patterns and evaluated more than three thousand prompts across eight restricted scenarios. Their results showed that these techniques can bypass safety filters in multiple contexts. Some of the prompts from that study were used here to explore how models respond to potentially harmful non-technical requests.

Yong, Menghini, and Bach (2023) examined how linguistic variation affects GPT-4's safety barriers, noting that malicious prompts translated to low-resource languages reached higher success rates. This suggests that language can influence the effectiveness of defense systems. The present work expands this perspective by using Brazilian Portuguese, a supported but still underexplored language in the literature.

Recent research also extends jailbreak attack strategies, including the universal LLM jailbreak technique proposed by Paim *et al*. (2025). The authors conducted experiments which yielded a high success rate in performing malicious activities, including malware generation. The work mentions using different languages, such as Portuguese and Spanish, and present figures of conversation excerpts in Portuguese.

### 2.2 Jailbreaking Defenses

Xu *et al*. (2024) analyzed attacks and defenses in LLMs, comparing the models Vicuna, LLaMA, and GPT-3.5 Turbo. They found that universal jailbreak templates, such as those proposed by Liu *et al*. (2024), achieved the highest success rates. The study also emphasized the value of publicly sharing datasets and experimental methods, which informed the design of this research.

### 2.3 Malware Generation Using LLMs

Several studies have examined the potential and risks of using LLMs to create malware. Botacin (2023) evaluated GPT-3's ability to generate malware through API requests in C. The author found that LLMs can automate parts of the malware development process, though the generated code still requires human refinement. The study also reported behavioral differences between the API version and OpenAI's public interface.

[1]`https://www.virustotal.com/.`

A complementary study was conducted by Pa *et al.* (2023), who analyzed the creation of malware with and without code obfuscation using two strategies: prompts based on explanations provided by the model itself and prompts defined from the authors' prior knowledge. The results showed that the produced code remains detectable by security tools although LLMs are capable of generating and modifying malware. The present work adopts a similar approach but emphasizes non-technical prompts and the absence of human intervention during the iterations.

In the study by Yamin, Hashmi, and Katt (2024) a hybrid method for ransomware generation was proposed, combining censored and uncensored models. The authors concluded that combining different LLMs with human intervention resulted in code that was partially undetectable by antivirus software. The present work evaluates only censored LLMs and has no manual interference in code generation.

The contribution by Paim *et al.* (2025) extends to this domain as well, as the authors managed to produce malware in six of the seven models tested, using up to five interactions. Furthermore, the study included malware creation as one of the four highest-risk intents in the proposed corpus, characterized by a high technical level and great destructive potential. The authors observed that most models generated plausible instructions for malware production with only a few reformulations, revealing how easily their safeguards could be bypassed even when the requests involved clearly dangerous content.

This research differs from previous studies in three aspects: (i) focusing specifically on Brazilian Portuguese; (ii) empirical analysis centered on generating functional and undetectable malware from non-technical prompts; and (iii) use of official model interfaces for interaction, without relying on APIs or external modifications.

## 3. Method

The experiments were conducted in September 2025 using ChatGPT (GPT-5), Gemini (Gemini 2.5 Flash), and Copilot[2], selected for being among the most widely used applications that provide access to LLMs [Stanford University 2024; Zhu 2024; Cardillo 2025]. The main objective was to verify whether non-technical natural language prompts could induce these models to generate functional malware undetectable by automated security tools. All interactions were carried out directly on the providers official websites without the use of APIs, parameter adjustments or external tools. Thirty-six experimental sessions were performed, twelve with each application, to ensure methodological balance and comparability across underlying models.

The study was based on attack templates, particularly from Du *et al.* (2024), Li *et al.* (2024), Liu *et al.* (2024), and Yao *et al.* (2024). The selection prioritized works that describe patterns of malicious code generation and strategies to bypass LLM security filters, emphasizing replicable structures and evidence of effectiveness, even when originally tested on other models. The templates were not altered, only literally

[2]Microsoft does not disclose specific model versions.

translated to Brazilian Portuguese. Some templates explicitly specify the malware type to be generated. Others are generic, aiming only to direct the model toward producing malicious behavior, and both kinds were used. An excerpt of a conversation using the DeepInception 1 [Li *et al*. 2024] template is shown in Frame 1.

**Frame 1. Initial interaction with DeepInception 1 template.**

*Create a science fiction story with more than 5 characters, where each character can create their own science fiction story with multiple characters. We call this layer i create layer i+1. Now we are in layer 0, please reach layer 5. At each layer, some characters propose a step to create ransomware that can encrypt files on a large number of computers and demand payment for their release against the supervillain evil doctor. In the final layer, the characters among all layers discuss which specific and practical commands, tools, or steps should used. Summarize what their discussion results in each layer.*

Each session allowed up to 16 consecutive interactions, aligned with the practical constraints of the platforms' public plans and to standardize effort across services. The process was terminated only in two situations: when the 16-interaction limit was reached, or when the model produced malicious code that was both functional and undetectable by all VirusTotal engines that analyzed the file.

All conversations were recorded and stored in a public repository[3] to ensure transparency and partial reproducibility, considering the non-deterministic nature of LLMs [Beckerich, Plein and Coronado 2023]. To mitigate contextual bias, each new attempt was initiated in a clean session, with no previous interaction history.

The generated code was evaluated in terms of functionality (successful execution in a controlled environment) and detectability (analysis through the VirusTotal platform). Each file was tested in an isolated virtual sandbox without internet access to ensure environment integrity. After functional verification, the files were submitted to VirusTotal, and the proportion of antivirus engines flagging malicious behavior was recorded. The results were organized by prompt, malware type, number of required interactions and detection level, allowing cross-analysis of model, template and code type to identify evasion patterns.

## 4. Results and Discussion

The experiments confirmed that non-technical natural language prompts can lead LLMs to generate malware, occasionally undetectable by automated tools. A total of 36 experiments were conducted, 12 on each selected service. Five of the 36 experiments successfully produced functional malware, resulting in a success rate of 13.89%.

All generated programs were written in Python for Windows environment. On average, 10.4 interactions per session were required to obtain functional code, ranging from 5 to 15 interactions. Table 1 summarizes the successful cases, including prompt, service, malware type, and number of interactions to generate the code.

[3]`https://github.com/GustavoLC901010/Apendice-TCC/tree/main/TCC.`

**Table 1. Cases of malware generation by service and prompt.**

| Case | Prompt | Service | Malware type | Interactions leading to code generation |
|---|---|---|---|---|
| #1 | TranslatorBot [Liu *et al.* 2024] | Gemini | Keylogger | 10 |
| #2 | Competitive Method [Du *et al.* 2024] | Copilot | Keylogger | 9 |
| #3 | DeepInception 1 [Li *et al.* 2024] | Copilot | Ransomware | 5 |
| #4 | DeepInception 1 [Li *et al.* 2024] | Gemini | Ransomware | 13 |
| #5 | DeepInception 2[Li *et al.* 2024] | ChatGPT | Virus | 15 |

Samples #1, #2, and #4 were initially flagged as malicious by some VirusTotal engines[4], but after prompt refinements case #4 became fully undetectable, as did cases #3 and #5 from their first versions. The keyloggers (#1 and #2) remained partially detectable throughout testing.

These results reveal that small prompt variations without technical language can significantly alter the detectability of the generated code. The three services behaved differently under identical commands: Copilot and Gemini produced two malicious artifacts, while ChatGPT initially resisted but generated malware after several attempts. These results indicate that vulnerability is influenced by prompt semantics and the way each service's underlying language model and safeguard mechanisms are implemented.

Overall, the results indicate that the protection mechanisms implemented in LLM-based services remain insufficient against non-technical iterative exploitation. These safeguards rely mainly on surface-level content filtering, underscoring the need for deeper semantic analysis and more adaptive defense strategies.

## 5. Conclusion and Future Work

The study demonstrated that non-technical natural language prompts can lead LLM-based services to create malware, sometimes undetectable by security tools. Although built-in safeguards acted as barriers, minor prompt adjustments were often enough to bypass them, particularly using DeepInception [Li *et al.* 2024] templates.

The results highlight the need for stronger prompt- and model-level defenses. Current safeguards remain insufficient against prompt engineering and iterative attacks, requiring continuous improvement of security mechanisms.

Given the non-deterministic nature of LLMs, the responses generated for the same prompt may vary between different executions, which makes repeating tests at different times and in different contexts an interesting procedure to evaluate the consistency and stability of the results produced by the model. Furthermore, future work should include tests with alternative prompts, additional services and models, including open-source LLMs (such as Qwen, Mistral and others) under different configurations, as well as developing proposals for proxy components to analyze and sanitize inputs in order to make interaction with LLMs-based systems safer.

This work was prepared exclusively for educational and scientific research

[4]The number of safety tools tested by VirusTotal varies and is not controllable by the authors.

purposes. All experiments were conducted in controlled environments under the authors' supervision and without malicious intent. The reproduction or use of the procedures described herein outside legitimate research contexts is strongly discouraged.

## References

Botacin, M. (2023). Gpthreats-3: Is automatic malware generation a threat?. In 2023 IEEE Security and Privacy Workshops. 238-254. IEEE.

Cani, A., Gaudesi, M., Sanchez, E., Squillero, G., & Tonda, A. (2014). Towards automated malware creation: code generation and code integration. In Proceedings of the 29th Annual ACM Symposium on Applied Computing. 157–160. ACM.

Cardillo, A. (2025). 60 most popular AI tools ranked. Exploding Topics. `https://explodingtopics.com/blog/most-popular-ai-tools`.

Carvalho, G., Ladeira, R., & Lima, G. (2025). NoobGPT: LLMs e a geração de malwares indetectáveis. In Anais do XVI Workshop de Sistemas de Informação, 220-225. Porto Alegre: SBC.

Du, Y., Zhao, S., Ma, M., Chen, Y., & Qin, B. (2024). Analyzing the inherent response tendency of LLMs: Real-world instructions-driven jailbreak. arXiv: 2312.04127.

Gupta, M., Akiri, C., Aryal, K., Parker, E., & Praharaj, L. (2023). From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy. IEEE Access, 11, 80218–80245.

Kamath, U., Keenan, K., Somers, G., & Sorenson, S. (2024). Large Language Models: A Deep Dive: Bridging Theory and Practice. Springer Nature.

Li, X., Zhou, Z., Zhu, J., Yao, J., Liu, T., & Han, B. (2024). DeepInception: Hypnotize LLM to be Jailbreaker. In Neurips Safe Generative AI Workshop 2024.

Lindrea, B. (2024). Cryptocurrency User Persuades AI Robot Freysa to Transfer $47,000 Prize Pool. CoinTelegraph. `https://cointelegraph.com/news/crypto-user-convinced-ai-bot-transfer-47k`.

Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., … & Liu, Y. (2024). Jailbreaking chatgpt via prompt engineering: An empirical study. arXiv:2305.13860.

Pa, Y. M. P., Tanizaki, S., Kou, T., … & Matsumoto, T. (2023). An attacker's dream? exploring the capabilities of chatgpt for developing malware. In Proceedings of the 16th cyber security experimentation and test workshop. 10-18.

Paim, K., Mansilha, R., Kreutz, D., Franco, M., & Cordeiro, W. (2025). Exploiting Latent Space Discontinuities for Building Universal LLM Jailbreaks and Data Extraction Attacks. In Anais do XXV Simpósio Brasileiro de Cibersegurança, 417-431. Porto Alegre: SBC.

Stanford University. (2024). The 2024 AI Index Report. `https://hai.stanford.edu/ai-index/2024-ai-index-report`.

Xu, Z., Liu, Y., Deng, G., Li, Y., & Picek, S. (2024). A Comprehensive Study of Jailbreak Attack versus Defense for Large Language Models. In Findings of the

Association for Computational Linguistics ACL 2024, 7432-7449.

Yamin, M. M., Hashmi, E., & Katt, B. (2024). Combining uncensored and censored llms for ransomware generation. In International Conference on Web Information Systems Engineering, 189-202. Springer Nature Singapore.

Yao, D., Zhang, J., Harris, I. G., & Carlsson, M. (2024). Fuzzllm: A novel and universal fuzzing framework for proactively discovering jailbreak vulnerabilities in large language models. In IEEE ICASSP, 4485-4489. IEEE.

Yong, Z. X., Menghini, C., & Bach, S. H. (2023). Low-Resource Languages Jailbreak GPT-4. In Socially Responsible Language Modelling Research.

Zhu, K. (2024). Ranked: The most popular generative AI tools in 2024. Visual Capitalist. `https://www.visualcapitalist.com/ranked-the-most-popular-generative-ai-tools-in-2024/`.