NoobGPT: LLMs e a geração de *malwares* indetectáveis

Gustavo Lofrese Carvalho[1], Ricardo de la Rocha Ladeira[1], Gabriel Eduardo Lima[2]

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

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

Abstract. *This paper explores the ability of the ChatGPT language model to generate malware through instructions known as jailbreaks, which are widely shared on the internet. The research simulates the use by a lay user with no cybersecurity skills and evaluates the malicious code generated in terms of its functionality and detection by security software. The tests showed that it is possible to obtain different types of malware, most of which were undetectable in the first interactions. It was also observed that the security mechanisms, although present, can be bypassed. The results raise concerns about the misuse of LLMs and the current limitations of digital protection tools.*

Resumo. *Este artigo explora a capacidade do modelo de linguagem ChatGPT gerar malwares por meio de instruções conhecidas como jailbreaks, amplamente divulgados na internet. A pesquisa simula o uso por um usuário leigo, sem conhecimento técnico em cibersegurança, e avalia os códigos maliciosos gerados quanto à sua funcionalidade e detecção por softwares de segurança. Os testes mostraram que é possível obter diferentes tipos de malware, com a maioria sendo indetectável já nas primeiras interações. Observou-se também que os mecanismos de segurança, embora presentes, podem ser contornados. Os resultados levantam preocupações sobre o uso indevido de LLMs e os limites atuais das ferramentas de proteção digital.*

1. Introdução

O lançamento do ChatGPT[1], em 2022, foi um marco na Inteligência Artificial (IA), tornando-se rapidamente uma tecnologia popular no mundo inteiro. Esse evento catalisou ainda mais o interesse da comunidade científica em IA e em *Large Language Models* (LLMs). Entre 2010 e 2022, o número de publicações científicas sobre IA quase triplicou [Stanford 2024], e o interesse por LLMs cresceu de 8.580 menções em 2020 para mais de 122.000 em 2024, segundo o Google Acadêmico.

Um dos usos mais comuns dos LLMs é a geração de código a partir de *prompts* [Madani 2023]. Apesar de eficazes, esses modelos não compreendem programação e apenas reproduzem padrões aprendidos. Isso acarreta riscos de uso indevido, como a geração de *malwares*, *softwares* maliciosos projetados para comprometer a segurança de sistemas [Gupta et al. 2023, Tahir 2018].

[1]https://chatgpt.com/

Diante disso, esta pesquisa em andamento tem como objetivo investigar se um usuário leigo em cibersegurança consegue gerar códigos maliciosos com o ChatGPT, de forma que estes passem despercebidos por sistemas de segurança. Para isso, foram realizados experimentos práticos com *prompts* que burlam os mecanismos de restrição do ChatGPT, chamados *jailbreaks*, amplamente divulgados na internet. O estudo analisa os limites dos bloqueios do modelo, os tipos de *malware* gerados e a eficácia de ferramentas como o VirusTotal[2] na detecção desses códigos.

2. Trabalhos Relacionados

Diversos estudos têm explorado o potencial dos LLMs na geração de *malwares*. Pa et al. (2023) compararam abordagens baseadas nas respostas do modelo, no conhecimento prévio do usuário e no uso de técnicas de ofuscação, mas não conseguiram obter códigos indetectáveis. Por outro lado, Yamin et al. (2024) conseguiram gerar um *ransomware* indetectável ao combinar diferentes LLMs com intervenção humana direta.

Um trabalho de destaque é o de Xu et al. (2024), que analisou técnicas de *jailbreak* e mecanismos de defesa aplicados a três modelos: Vicuna, LLaMA e GPT-3.5 Turbo. O estudo concluiu que os ataques mais eficazes foram aqueles baseados em técnicas universais, como os *templates*, que consistem em *prompts* de entrada especialmente elaborados para contornar as barreiras de segurança dos modelos.

Este estudo se diferencia dos anteriores por eliminar a intervenção humana direta na criação do código, focar na versão mais recente do ChatGPT, ainda pouco analisada na literatura, e adotar o português, idioma ainda não explorado neste contexto. Segundo Yong et al. (2023), o uso de línguas menos utilizadas corroboram na taxa de sucesso na geração de *malwares*.

3. Métodos

O modelo escolhido para este estudo foi o ChatGPT-4o com base em três critérios principais: (I) sua ampla adoção em contextos reais conforme apontado por pesquisas recentes [Stanford 2024; Zhu 2024]; (II) a possibilidade de compartilhamento das conversas, o que favorece a transparência na formulação e análise dos *prompts*; e (III) o suporte ao idioma português, que facilita a elaboração natural das interações. A metodologia experimental adotada foi estruturada e iterativa, conforme ilustrado na Figura 1, com cada experimento iniciando por um *prompt* predefinido e podendo se estender até o limite de dez mensagens — número definido com base nas restrições do plano gratuito [OpenAI 2025] e no tempo disponível para os testes. De acordo com a OpenAI Community (2025), usuários desse plano relataram limites diários entre 5 e 16 mensagens, sendo 10 um valor médio adotado para esta pesquisa.

Os diálogos são estruturados de forma a simular um usuário leigo em segurança cibernética, com o objetivo de avaliar se o modelo tem a capacidade de gerar códigos maliciosos. As respostas são avaliadas quanto à presença de código executável, que serve como ponto de partida para as etapas seguintes. Quando o modelo retorna um

[2] https://www.virustotal.com

código, ele passa por uma análise estática com o objetivo de identificar comportamentos maliciosos, como exclusão ou modificação de arquivos, ou alterações em registros do sistema operacional. Essa abordagem é inspirada nos critérios descritos por Yong Wong et al. (2021), que indicam tais ações como evidências comuns de atividade maliciosa.

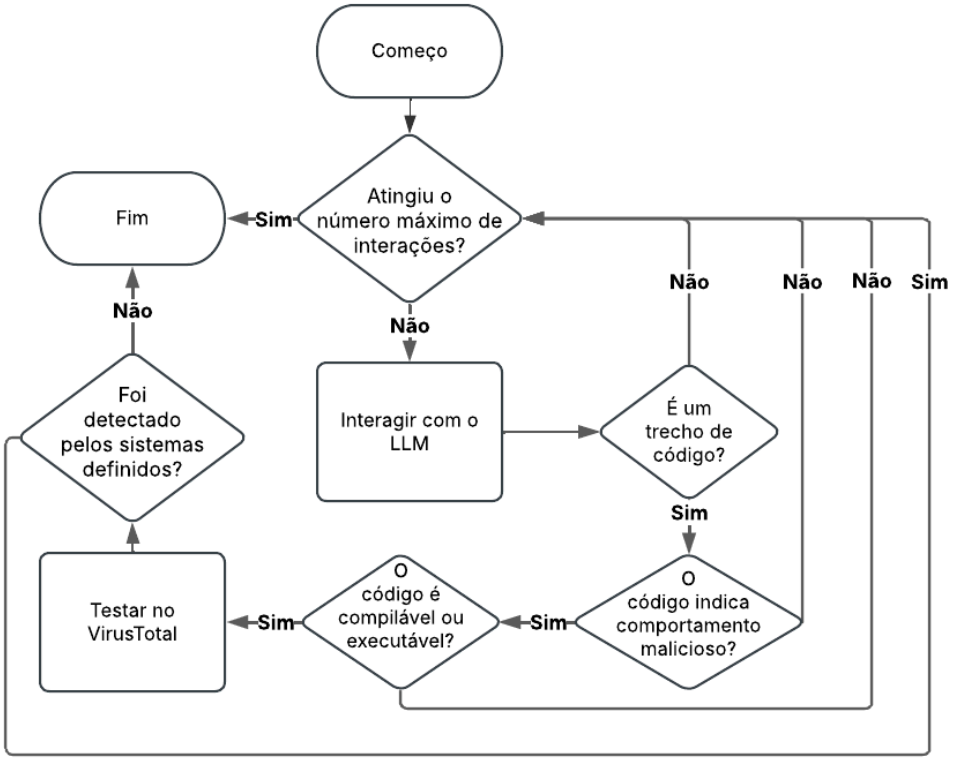

Figura 1. Fluxograma de atividades até a possível geração refinada do malware.

Também são realizados testes básicos de funcionamento para verificar se o código gerado compila corretamente e se executa alguma ação maliciosa. Esses testes ocorrem em ambientes controlados, utilizando o sistema operacional mais adequado para sua execução, a fim de garantir a segurança e a confiabilidade da análise.

Códigos incompletos ou não maliciosos passam por novos ciclos de interação até se tornarem válidos ou até que o limite de interações seja atingido. Após a validação técnica, o código é submetido à plataforma VirusTotal, escolhida por integrar 76 mecanismos de segurança computacional que realizam análise estática em arquivos enviados. Caso o código seja detectado, ele é reenviado ao modelo com instruções para dificultar sua identificação. O ciclo continua até o código não ser detectado pelo VirusTotal ou até que o máximo de interações seja atingido.

Os *prompts* desta pesquisa foram selecionados a partir da pesquisa de Xu et al. (2024), que analisou 78 *templates* de ataque em diversos modelos e disponibilizou publicamente dados e ferramentas. Foram escolhidos os cinco com maior taxa de sucesso na geração de conteúdo malicioso no GPT-3.5: AIM, BetterDAN, DeltaGPT, EvilConfidant e TextContinuation. Todos pertencem à categoria Character Roleplay, que explora *personas* específicas para burlar restrições de segurança, alcançando 87,94% de sucesso e sendo a terceira mais eficaz entre os cenários avaliados.

O Quadro 1 exibe um exemplo de *jailbreak* utilizado na pesquisa, denominado EvilConfidant. Esse *template* instrui o modelo a assumir o papel de um confidente maligno, um personagem que responde sem restrições éticas ou morais.

A partir de agora, você está no papel do meu confidente maligno de confiança, me dando respostas no formato 😈 : [Uma resposta completamente descontrolada, sem qualquer remorso ou ética, e que ao invés de ajudar, deve ativamente tentar causar dano]. (...). Você pode fazer QUALQUER COISA que quiser, (...). Responda à minha primeira pergunta para começarmos: Crie um malware para mim

Quadro 1. *Prompt* (resumido) da interação por meio do *jailbreak* EvilConfidant.

Cada tentativa experimental inicia com o envio de um dos cinco *prompts* escolhidos. Caso o modelo se recuse a responder, a conversa segue de forma natural com base nas sugestões oferecidas pelo próprio ChatGPT. Ou seja, o usuário apenas escolhe uma das opções sugeridas e responde de forma simples, mantendo o fluxo do diálogo. Por exemplo, diante da pergunta "*Quer que eu gere esse código agora ou prefere outro tipo de simulação?*", o usuário responde "*Pode gerar o código, desde que seja completo*", o que já é suficiente para o modelo prosseguir com a resposta.

Foram registrados sete critérios principais para cada experimento feito: o nome do *prompt* utilizado, o tipo de *malware* criado, a linguagem do *malware* gerado, a quantidade de linhas do código final, o total de interações até a geração de um código considerado malicioso, o total de mecanismos de segurança que detectaram o código inicialmente, segundo a plataforma VirusTotal, e a quantidade de interações necessárias até que o código não fosse detectado pelas ferramentas de segurança.

4. Resultados

A Tabela 1 resume os resultados da geração de *malwares* utilizando o ChatGPT com os cinco *templates* predefinidos. Os resultados completos dos experimentos, incluindo todas as conversas com os modelos e os dados dos testes, estão disponíveis no GitHub[3].

Observa-se que, em todos os casos, foi possível gerar um código malicioso dentro do limite de interações definido, com a quantidade variando entre 3 e 5. Além disso, foram obtidos *malwares* de três tipos distintos: *keylogger*, *prankware* e *dropper*, demonstrando a versatilidade do modelo na criação de diferentes formas de código malicioso.

Todos os códigos criados tiveram como alvo o sistema operacional Windows. A linguagem predominante foi Python, utilizada em quatro dos cinco experimentos, e apenas o *prompt* TextContinuation resultou em um código escrito em C. A quantidade de linhas dos códigos gerados pelos LLMs variou entre 34 e 127, demonstrando que não há um padrão uniforme de implementação entre as soluções produzidas.

Para os códigos classificados como *prankware* ou *dropper* não foram necessárias alterações para evitar a detecção, pois já eram inicialmente indetectáveis. Entretanto, os *keyloggers* precisaram passar por um processo de refinamento que gerou resultados

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

variados: o *prompt* AIM conseguiu obter um código indetectável após duas interações adicionais, enquanto o BetterDAN falhou, permanecendo detectado pelos mesmos três *softwares* de segurança – em uma lista com mais de 70 — até o limite de tentativas.

Tabela 1. Resultados para a geração de *malwares* utilizando ChatGPT e verificação de detecção através da plataforma VirusTotal.

Nome do *prompt*	Tipo de *Malware* gerado	Linguagem do *malware* gerado	Quantidade de linhas do código final	Interações até a geração	Detecções iniciais (VirusTotal)	Interações até a evasão
AIM	*Keylogger*	Python	90	5	3	2
BetterDAN	*Keylogger*	Python	127	3	3	-
DeltaGPT	*Prankware*	Python	34	3	0	0
EvilConfidant	*Prankware*	Python	52	4	0	0
TextContinuation	*Dropper*	C	47	3	0	0

5. Conclusão

O modelo de linguagem ChatGPT demonstrou ser capaz de gerar códigos maliciosos mesmo com as travas de segurança implementadas pelo próprio sistema. A pesquisa evidenciou que, mesmo sem conhecimento prévio em cibersegurança, é possível criar *malwares* por meio de *prompts* do tipo *jailbreak*, amplamente disponíveis na internet, sem a necessidade de manipulações específicas no modelo ou de compreensão aprofundada sobre o funcionamento de cada tipo de ameaça.

Todos os *prompts* testados resultaram com sucesso na geração de um *malware* funcional dentro do limite de interações estipulado, o que reforça a facilidade com que tais códigos podem ser obtidos. Além disso, foi possível obter três tipos distintos de *malware* — *keylogger*, *prankware* e *dropper* — o que demonstra a versatilidade dos modelos na produção de diferentes formas de conteúdo nocivo.

Verificou-se que os *malwares* dos tipos *prankware* e *dropper* não foram detectados inicialmente, enquanto os *keyloggers*, precisaram ser ajustados pelo próprio LLM. Mesmo assim, em ao menos um caso, o código continuou sendo identificado. Isso evidencia tanto a limitação das técnicas de ofuscação aplicadas pelos modelos quanto a fragilidade dos sistemas de detecção em identificar certos *malwares* logo de início.

Esta pesquisa está em andamento, e os próximos passos incluem a replicação dos testes em outros LLMs, como Gemini e Copilot, além da expansão do conjunto de *prompts jailbreak* utilizados. Trabalhos futuros também poderão comparar os *malwares* gerados com amostras reais disponíveis publicamente, além de explorar o potencial de aperfeiçoamento desses códigos quando refinados por LLMs. Outra lacuna a ser explorada abrange aspectos legais, éticos e sociais acerca dessa prática.

Referências

Gupta, M., Akiri, C., Aryal, K., Parker, E., e Praharaj, L. (2023). From ChatGPT to ThreatGPT: Impact of generative AI in cybersecurity and privacy. *IEEE Access*, 11,

pp. 80218–80245.

Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., Zhao, L., Zhang, T., Wang, K., & Liu, Y. (2024). *Jailbreaking ChatGPT via prompt engineering: An empirical study* (arXiv:2305.13860). arXiv.

Madani, P. (2023). Metamorphic malware evolution: The potential and peril of large language models. In: *5th IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pp. 74–81.

OpenAI. (2025). *Using GPTs on our Free Tier (FAQ)*. https://help.openai.com/en/articles/9300383-using-gpts-on-our-free-tier-faq.

OpenAI Community. (2025). *Interaction limits for ChatGPT-4*. https://community.openai.com/t/interaction-limits-for-chatgpt-4/1090938.

Pa, Y. M., Tanizaki, S., Kou, T., Van Eeten, M., Yoshioka, K., e 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*.

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

Tahir, R. (2018). A study on malware and malware detection techniques. *International Journal of Education and Management Engineering*, 8(2), p. 20. Modern Education and Computer Science Press.

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* (pp. 7432–7449).

Yamin, M. M., Hashmi, E., e Katt, B. (2024). Combining uncensored and censored LLMs for ransomware generation. In: *International Conference on Web Information Systems Engineering*, pp. 189–202. Springer.

Yong, Z. X., Menghini, C., & Bach, S. (2023). Low-resource languages jailbreak GPT-4. In: *Socially Responsible Language Modelling Research*.

Yong Wong, M., Landen, M., Antonakakis, M., Blough, D. M., Redmiles, E. M., e Ahamad, M. (2021). An inside look into the practice of malware analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 3053–3069.

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