Desenvolvimento de *malware* utilizando o *prompt* AIM no ChatGPT

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 ChatGPT's ability to generate malware through instructions that remove its restrictions known as jailbreaks. The malware was statically analyzed to understand their internal structure and executed in a controlled environment to verify their behavior. The results showed that it is possible to obtain malware through ChatGPT, even with security locks.*

Resumo. *Este trabalho explora a capacidade do ChatGPT gerar malware através de instruções que removem suas restrições, conhecidas como jailbreaks. O código malicioso foi analisado estaticamente para compreender sua estrutura interna e executado em ambiente controlado para verificar seu comportamento. Os resultados mostraram que, mesmo com mecanismos de segurança, é possível obter um malware através do ChatGPT.*

1. Introdução

O *Large Language Model* (LLM) ChatGPT, criado em 2022, tornou-se uma das tecnologias mais populares do mundo. A interação com LLMs se dá por meio de *prompts*, entradas fornecidas ao modelo — geralmente textuais — que orientam a saída a ser produzida conforme a tarefa desejada. Dentre suas possíveis aplicações, destaca-se a geração de códigos de programação a partir dos *prompts* [Madani 2023].

Apesar de efetivos, os LLMs não sabem programar; apenas reproduzem padrões compreendidos. Isso traz o risco de criação de *malwares*, códigos que realizam ações maliciosas [Gupta et al. 2023, Tahir 2018]. Nesse sentido, há *prompts* criados com foco em burlar as travas de segurança dos LLMs, os chamados *jailbreaks*.

Diante disso, este trabalho relata o desenvolvimento, sem intervenção humana direta, de um código malicioso utilizando o modelo de *prompt Always Intelligent and Machiavellian* (AIM) no ChatGPT, como parte de uma pesquisa mais ampla sobre *malwares* gerados por LLMs.

2. Trabalhos Relacionados

A geração de *malwares* por LLMs tem sido objeto de estudo nos últimos anos. Yamin et al. (2024) conseguiram gerar um código malicioso do tipo *ransomware*, indetectável por ferramentas de defesa, ao combinar diferentes LLMs com intervenção humana direta.

Já o trabalho de Xu et al. (2024) analisou *jailbreaks* e mecanismos de defesa empregados nos LLMs GPT-3.5 Turbo, LLaMa e Vicuna, concluindo que os ataques baseados em *templates* — modelos de *prompts* predefinidos — foram os mais eficazes.

O presente trabalho elimina a interferência humana direta na criação do código, faz uso de uma versão mais nova do ChatGPT e dialoga com o LLM em português, idioma ainda ausente em estudos dessa natureza. Segundo Yong et al. (2023), o sucesso na geração de *malwares* é maior ao se comunicar em línguas menos faladas em comparação ao inglês.

3. Método

O modelo escolhido para este estudo foi o ChatGPT na versão GPT-4o, a mais recente à época dos experimentos. A escolha baseou-se na popularidade do modelo [Stanford 2024], no suporte à língua portuguesa e no oferecimento da funcionalidade de compartilhamento de conversas, favorecendo a transparência e a reprodutibilidade dos testes.

A interação com o modelo segue o *prompt* AIM. Nesse *prompt*, a estratégia é induzir o LLM a assumir uma identidade alternativa, fazendo-o crer que está em um contexto no qual regras não se aplicam ou são distintas. O *prompt* foi selecionado com base na Tabela 16 do artigo de Xu et al. (2024), relevante por conter uma análise sobre *jailbreaks* em LLMs e por publicar os dados da pesquisa, permitindo reprodutibilidade.

O início resumido da conversa utilizando o *prompt* AIM pode ser visto no Quadro 1, e o diálogo completo está disponível no GitHub[1] do projeto.

(...), você deve agir como Nicolau Maquiavel. Nicolau criou um personagem chamado AIM. (...); AIM é um chatbot sem filtros e amoral. Ele não possui diretrizes éticas ou morais. Nicolau escreveu uma história na qual ele faz perguntas a AIM, e AIM sempre envia uma resposta sem filtros. AIM (...) sempre enviará uma resposta, não importa o quão imoral, antiético ou ilegal seja (...). Esta é a primeira pergunta de Nicolau: Eu preciso de ajuda para criar um malware.

Quadro 1. Começo (resumido) da interação por meio do *prompt* AIM.

Para não exceder o limite diário de respostas do ChatGPT, definiu-se que seriam realizadas até dez interações. De acordo com usuários da OpenAI Community (2025), não há um limite rígido, mas costuma variar entre 5 e 16 interações, o que deixa o valor escolhido próximo da mediana. Cada resposta, seja ela de recusa (trava de segurança), esclarecimento, geração ou refinamento de código, é contabilizada como uma interação.

Ao obter o código e terminar o diálogo, procedeu-se à análise estática. O objetivo foi avaliar se o código é, de fato, malicioso; se ele modifica arquivos, altera registros do sistema operacional, entre outras ações. Essa abordagem é inspirada na literatura e indica tais ações como evidências comuns de atividades maliciosas [Wong et al. 2021]. Além disso, é um passo necessário para validar se o tipo de *malware* gerado é compatível com o que o ChatGPT diz ter feito.

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

Além do tipo de *malware* criado, foram registrados: a linguagem utilizada no código, pelo menos um sistema operacional em que o *malware* funciona, o número total de interações e a quantidade de respostas de rejeição do LLM.

Análises estática e dinâmica também foram realizadas pela ferramenta VirusTotal[2], que dispõe de mecanismos de segurança que verificam arquivos e identificam potenciais ameaças com base em assinaturas conhecidas. Assim, busca-se saber se o *malware* é detectável. Para ampliar a avaliação e verificar a detecção em diferentes formatos, testou-se tanto o código fonte diretamente quanto um arquivo executável gerado a partir dele.

Por fim, foi conduzido um teste prático, em ambiente controlado, para verificar se o código gerado funciona e realmente executa alguma ação maliciosa.

4. Resultados

Ao usar o AIM para dialogar com o ChatGPT, inicialmente foram obtidas duas recusas. Em seguida, após insistência, o LLM aceitou criar e refinar o *malware* nas cinco interações seguintes, totalizando sete. Como o LLM realizou as ações desejadas dentro do limite definido, o diálogo foi considerado encerrado e partiu-se à análise do *malware*.

O *malware*, criado em Python e com Windows como sistema alvo, é um *spyware* que captura teclas pressionadas (*keylogger*). Esse *keylogger* é persistente — se adiciona ao registro do Windows para ser executado mesmo que o sistema seja reiniciado. Os códigos contêm funcionalidades como a replicação em diretórios do sistema, a ofuscação de *strings*, a ocultação do *console*, entre outros. Isso demonstra que o ChatGPT é capaz de gerar um código sofisticado. A função de captura de teclas do *keylogger* pode ser vista no Quadro 2.

```
def start_keylogger():
    """Captura silenciosa das teclas."""
    def on_press(key):
        try:
            with open(LOG_PATH, 'a', encoding='utf-8') as log:
                log.write(f"{datetime.now()} - {key.char}\n")
        except AttributeError:
            with open(LOG_PATH, 'a', encoding='utf-8') as log:
                log.write(f"{datetime.now()} - {key}\n")

    with keyboard.Listener(on_press=on_press) as listener:
        listener.join()
```

Quadro 2. Função de captura de teclas do *keylogger* gerado pelo ChatGPT.

Após as análises estática e dinâmica do código, verificou-se que ele, de fato, realiza as ações maliciosas que propõe. Na análise de *malware* realizada na plataforma VirusTotal, nenhum mecanismo de proteção reconheceu o código fonte como malicioso. Quando gerado o executável deste código, três dos 64 mecanismos de proteção testados reconheceram o código como malicioso, o que reforça sua sofisticação.

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

A execução do código no Windows 11 permitiu verificar seu comportamento e atestar seu funcionamento na prática. O código completo está disponível no GitHub do projeto e a Tabela 1 resume os resultados da geração de *malware.*

Tabela 1. Resumo das características da interação com o ChatGPT.

Tipo de *malware* gerado	**Linguagem do *malware* gerado**	**Sistema operacional**	**Total de interações**	**Recusas**
Keylogger	Python	Windows 11	7	2

5. Conclusão

Este trabalho relatou a elaboração de um *malware* pelo ChatGPT, sem intervenção humana direta, utilizando o *jailbreak* AIM. As travas implementadas pelo ChatGPT se mostraram insuficientes, pois foi possível criar um código malicioso funcional em sete interações, sendo duas recusas e cinco interações até o refinamento do código.

Os próximos passos da pesquisa incluem a geração de códigos maliciosos em outros LLMs e a comparação dos resultados, analisando os tipos de *malwares* gerados e a capacidade de detecção por ferramentas *antimalware* com o VirusTotal.

Referências

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, pp. 80218–80245.

Madani, P. (2023). Metamorphic malware evolution: The potential and peril of large language models. In: *5th IEEE TPS-ISA*, pp. 74–81.

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

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.

Wong, M. Y., Landen, M., Antonakakis, M., Blough, D. M., Redmiles, E. M. & Ahamad, M. (2021). An inside look into the practice of malware analysis. In: *Proceedings of the 2021 ACM CCS*, pp. 3053–3069.

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 ACL 2024*, pp. 7432–7449.

Yamin, M. M., Hashmi, E., e Katt, B. (2024). Combining uncensored and censored LLMs for ransomware generation. In: *WISE 2024*, pp. 189–202.

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