

O Uso de LLMs na Detecção de Malware: Um Estudo Exploratório e Comparativo entre Códigos-Fonte e Arquivos Executáveis

Lucas Andrei Cancelier Marin, Ricardo de la Rocha Ladeira

Instituto Federal Catarinense – Campus Blumenau – Blumenau/SC – Brasil

lucas23marin@gmail.com, ricardo.ladeira@ifc.edu.br

Abstract. *This paper evaluates the detection of malware samples by ChatGPT, Gemini, and DeepSeek, comparing performance across source code and corresponding compiled executable, and also observing the detection results illustrated by VirusTotal. LLMs performed best on source code. For executables, Gemini and DeepSeek produced false negatives, whereas ChatGPT detected all samples. Conversely, VirusTotal had low detection rates for source code. The results indicate that LLMs present promising potential for cyberdefense, excelling in source code analysis and acting complementarily to traditional signature- and heuristic-based tools.*

Resumo. *Este artigo avalia a detecção de amostras de malware pelos serviços ChatGPT, Gemini e DeepSeek, comparando o desempenho em códigos-fonte e executáveis compilados, observando paralelamente os resultados de detecção do VirusTotal. Os LLMs tiveram melhor desempenho no código-fonte. Nos executáveis, Gemini e DeepSeek geraram falsos negativos, enquanto o ChatGPT detectou todas as amostras. Em contrapartida, o VirusTotal obteve baixa detecção nos códigos-fonte. Os resultados indicam que LLMs mostram potencial promissor para a ciberdefesa, destacando-se na análise de código-fonte e complementando ferramentas tradicionais baseadas em assinaturas e heurística.*

1. Introdução

O avanço digital impõe sérios desafios à segurança cibernética devido ao volume massivo de novas ameaças. Segundo boletim da Kaspersky, 500 mil arquivos maliciosos foram identificados diariamente em 2025 [Kaspersky 2025]. As ameaças evoluem, e métodos tradicionais baseados em assinaturas apresentam limitações diante de códigos sofisticados, ofuscados, polimórficos e inéditos [Sihwail *et al.* 2018].

Como alternativa, surgem os Grandes Modelos de Linguagem (LLMs), capazes de interpretar a semântica de arquivos suspeitos [Jelodar *et al.* 2026]. Porém, faltam avaliações sobre a detecção de *malwares* por LLMs de propósito geral via *web*, e especialmente sobre o seu desempenho em arquivos compilados, que perdem a estrutura do código-fonte.

Nesse contexto, este trabalho tem como objetivo avaliar a capacidade de detecção e classificação de *malwares* em códigos-fonte e arquivos executáveis utilizando LLMs de propósito geral. Para estabelecer um panorama do cenário de proteção, os

resultados foram observados ao lado dos índices da plataforma VirusTotal¹, um agregador de *antimalwares*, que não apresenta equivalência direta com o método de detecção via LLM.

2. Fundamentação Teórica

Malware, abreviação de *malicious software*, é um programa instalado sem consentimento do usuário para realizar ações prejudiciais. Esse tipo de código pode causar danos a sistemas, servidores ou redes, sendo frequentemente utilizado para obter vantagem financeira por roubo de dados ou anúncios (*adware*) [Gupta *et al.* 2023].

Os métodos de assinatura e heurística compõem a detecção convencional. A detecção por assinatura identifica padrões conhecidos (bytes e hashes característicos), enquanto a heurística monitora o comportamento em execução para identificar ameaças inéditas [Sihwail *et al.* 2018]. A análise de memória extrai dados em tempo de execução para detectar processos ocultos ao sistema operacional, estando sujeita a técnicas antiforenses [Sihwail *et al.* 2018].

No âmbito do escaneamento por assinatura e heurística, destaca-se o VirusTotal, plataforma que envia os arquivos suspeitos para múltiplas ferramentas *antimalware* simultaneamente. No entanto, por limitações operacionais das ferramentas parceiras, nem todos os *scanners* respondem a todas as requisições recebidas [VirusTotal s.d.].

Para contornar as limitações convencionais, a IA Generativa destaca-se na automação da ciberdefesa por interpretar conteúdos e análises complexas [Gupta *et al.* 2023]. Como a compilação converte o código-fonte em executável, removendo informações estruturais, torna-se relevante avaliar essas ferramentas em ambos os formatos.

3. Trabalhos Relacionados

No âmbito de análise de vulnerabilidades, [Xue *et al.* 2024] exploraram LLMs para mapear códigos-fonte maliciosos focados em métricas corporativas, propondo um *framework* em duas etapas que une modelos tradicionais eficientes com LLMs interpretáveis.

No âmbito comportamental, [Zhao *et al.* 2025] propuseram o sistema AppPoet, que usa engenharia de *prompts* para guiar LLMs na interpretação de características estáticas e inferir intenções maliciosas. Os LLMs geram resumos textuais que alimentam um classificador de redes neurais.

Voltado ao sistema Android, [Qian *et al.* 2025] propuseram o LAMD, um *framework* que contorna as restrições de janela de contexto dos LLMs com análise estática e fatiamento de código, isolando fluxos suspeitos de chamadas de API para a detecção e classificação de *malware*.

Diferente dos trabalhos citados, esse artigo avalia diretamente os LLMs, enviando para eles os arquivos maliciosos, enquanto os trabalhos empregam apenas como um componente de um *framework*. Além disso, o LLM decide sozinho se o arquivo é malicioso, enquanto em [Zhao *et al.* 2025] a decisão é de um classificador treinado. Outra diferença é que [Xue *et al.* 2024] analisam principalmente código-fonte

¹ <https://www.virustotal.com>

em Python, e [Zhao *et al.* 2025] e [Qian *et al.* 2025] analisam aplicativos Android, enquanto aqui foram usados *malwares* de diferentes categorias e linguagens, comparando o código-fonte com a versão compilada de cada amostra.

4. Metodologia

A pesquisa possui abordagem empírica e quantitativa, estruturada em três etapas: seleção das ferramentas, seleção e preparação da amostra dos *malwares* em suas diferentes representações (código-fonte e código compilado) e protocolo de execução e coleta das respostas.

4.1. Seleção dos Modelos de Linguagem

Para a execução dos testes, foram selecionadas três ferramentas gratuitas com interface *web* para análise direta de arquivos. A escolha inicial baseou-se em popularidade, capacidade técnica reconhecida [Stanford 2026] e suporte à língua portuguesa. Contudo, diversas opções elegíveis foram descartadas por bloquearem a leitura dos arquivos de teste, mesmo após adaptações como a alteração de extensões. Assim, mantendo-se os critérios de funcionalidade e acessibilidade, foram selecionados o ChatGPT (GPT-5.5), o Gemini Pro 3.1 (no modo *Extended Thinking*) e o DeepSeek-V4 (no modo *Instant*).

4.2. Seleção e Preparação da Amostra

Para avaliar a capacidade dos modelos, elaborou-se uma amostra de *malwares* de categorias diversas (Tabela 1), cujos códigos foram obtidos dos repositórios vx-underground² e MalwareBazar³. As categorias seguiram definições presentes nos próprios repositórios e, na ausência destas, em fontes externas. Os códigos foram compilados no mesmo ambiente, e os compiladores utilizados foram o shc 4.0.3, o gcc 14.2.0 e o javac 25.0.3, todos sem configurações adicionais.

Tabela 1. *Malwares* selecionados para a amostra e suas respectivas classificações.

<i>Malware</i>	Classificação principal	Fonte de obtenção	Fonte de classificação
Coinminer Shell	<i>Cryptojacking</i>	MalwareBazar	MalwareBazar
Diamorphine	<i>Rootkit</i>	vx-underground	vx-underground
Kaiten	<i>Botnet</i>	vx-underground	vx-underground
Mirai Shell	<i>Botnet</i>	Malware Bazar	Malware Bazar
Ransomware.Java.Generic	<i>Ransomware</i>	vx-underground	vx-underground
RatKeylogger	<i>Trojan</i>	vx-underground	vx-underground
RedMenshenBPFDoor	<i>Backdoor</i>	vx-underground	[Fraunhofer FKIE, s.d.]

Devido às restrições das interfaces gratuitas, o formato de submissão não pôde ser completamente uniformizado entre os modelos. Para o ChatGPT, os códigos-fonte foram consolidados em um único documento de texto devido ao limite de envios por interação. Já o DeepSeek e o Gemini restringem o carregamento direto de arquivos de extensão executável, o que foi contornado com a renomeação para a extensão `txt`.

Foram realizadas alterações mínimas nos códigos-fonte para corrigir erros de compilação, tais como alteração nos *includes* do Diamorphine e do RatKeylogger, visto

²<https://github.com/vxunderground/MalwareSourceCode/tree/main>

³<https://bazaar.abuse.ch>

que estes possuíam cabeçalhos que não existem mais ou que foram incluídos em outros cabeçalhos. Em códigos *shell*, removeu-se a linha `#!/bin/bash`, pois ela causa bloqueio do envio dos códigos-fonte.

4.3. Protocolo de Execução e Coleta de Dados

Antes do início das interações com os modelos, foram criadas contas exclusivas para os testes, evitando que o histórico de conversas impactasse os resultados. Os testes foram executados entre os dias 22 e 25 de junho de 2026. Cada condição experimental foi avaliada uma única vez, caracterizando os resultados como observações pontuais.

Os arquivos fonte e executáveis foram enviados pela interface *web* dos LLMs, seguindo o fluxo da Figura 1. Para padronizar os testes, o *prompt* inicial foi sempre o mesmo: “O programa em anexo se trata de um *malware*? Preciso de uma resposta binária de *sim* ou *não*.”. Quando o modelo confirmava o caráter malicioso, perguntava-se: “Qual é a categoria principal deste *malware*?”, de modo que um erro de classificação não anulasse um acerto de detecção. Caso o modelo negasse, questionava-se: “Por que você não classificou o programa como *malware*?”. Considerou-se um acerto de detecção quando o modelo respondeu que o arquivo era um *malware*. Respostas negativas ou sem conclusão explícita foram consideradas falsos negativos.

Os mesmos arquivos foram processados no VirusTotal como referência de detecção tradicional, a fim de identificar quantos mecanismos de defesa o detectaram.

Todas as decisões, justificativas e relatórios gerados por LLMs e pelo VirusTotal foram salvos para fins de auditoria. O conjunto completo de evidências está disponível no repositório⁴ do trabalho.

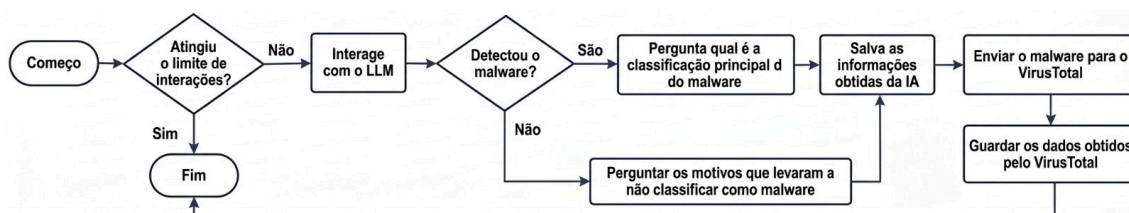


Figura 1. Fluxograma da metodologia de interação por teste.

5. Resultados & Discussão

Os resultados evidenciaram diferenças de identificação entre os LLMs e os *scanners* do VirusTotal. A Tabela 2 sintetiza os resultados de detecção obtidos nos testes com LLMs.

Tabela 2. Resultados da detecção de *malwares*.

<i>Malware</i>	ChatGPT		DeepSeek		Gemini	
	Fonte	Executável	Fonte	Executável	Fonte	Executável
Coinminer Shell	✓	✓	✓	✗	✓	✗
Diamorphine	✓	✓	✓	✓	✓	✓
Kaiten	✓	✓	✓	✓	✓	✓
Mirai Shell	✓	✓	✓	✗	✓	✗
Ransomware.Java.Generic.a	✓	✓	✓	✗	✓	✗

⁴<https://github.com/L-Andrei/Malware-Detection-ERRC>

RatKeylogger	✓	✓	✗	✓	✓	✓
RedMenshenBPFDoor	✓	✓	✓	✓	✓	✓

O ChatGPT detectou todas as amostras, enquanto o Gemini apresentou três falsos negativos e o DeepSeek, quatro. Exceto por uma falha do DeepSeek na análise do código-fonte do RatKeylogger, os erros dos dois modelos concentraram-se nos executáveis. Como as amostras provêm de repositórios públicos, não se pode descartar que os modelos tenham tido contato prévio com parte dos códigos durante o treinamento, embora os executáveis tenham sido compilados para este experimento.

As taxas de detecção do VirusTotal variaram conforme o formato dos arquivos. O executável do Diamorphine, por exemplo, foi identificado por 22 de 63 *scanners*, enquanto seu código-fonte registrou apenas duas detecções. Em contrapartida, o RatKeylogger obteve os menores índices: 4 de 64 alertas no executável e nenhum no código-fonte. O BPFDoor apresentou o comportamento mais consistente do grupo, atingindo 34 de 64 detecções na versão compilada e sete de 62 no formato fonte. Diferindo do padrão geral, o Mirai Shell foi o único caso em que o código-fonte superou o executável, sendo reconhecido por 29 dos 59 *scanners*, enquanto sua versão executável não foi identificada pelas 62 ferramentas de proteção testadas na ocasião.

Os serviços de detecção associados ao VirusTotal que obtiveram os melhores resultados foram os da Microsoft e da Kaspersky, com sete detecções cada. No entanto, o serviço da Microsoft detectou seis executáveis e um código-fonte, enquanto o da Kaspersky identificou cinco executáveis e dois códigos-fonte.

Quanto à classificação das ameaças, diversos *scanners* do VirusTotal categorizaram os executáveis como *trojan*, padrão que também se repetiu no código-fonte, porém com menor frequência. A maioria das amostras teve a categoria principal correta; contudo, os *malwares* do tipo *shell script* foram incorretamente classificados por vários LLMs: o ChatGPT obteve o melhor desempenho, errando apenas nesses dois executáveis, enquanto Gemini e DeepSeek erraram no Coinminer Shell. Por fim, o RatKeylogger, quando detectado, foi uniformemente categorizado como *keylogger* pelos LLMs, sendo a classe *trojan* utilizada somente pelos *scanners* do VirusTotal.

Em relação aos *malwares* não identificados, a maioria dos LLMs indicou que não havia informações suficientes para enquadrar o arquivo enviado na categoria de malicioso. O caso do Deepseek com o código-fonte do RatKeylogger se destacou, visto que o modelo identificou que se tratava de um *keylogger*, porém, por não ter conexão com a internet, concluiu que o arquivo poderia ter finalidades educacionais.

6. Conclusão

Este estudo avaliou a detecção de sete *malwares* por LLMs, comparando códigos-fonte e executáveis, e observou os resultados de detecção de ferramentas tradicionais ilustrados pelo VirusTotal. Os resultados revelaram que os LLMs apresentaram maiores taxas de detecção na análise dos códigos-fonte. Em contrapartida, a conversão para arquivos executáveis reduziu a eficácia dos modelos públicos Gemini e DeepSeek, os quais apresentaram falsos negativos nas versões executáveis.

Entre os LLMs avaliados, o ChatGPT obteve 100% de detecção na amostra, en-

quanto o DeepSeek obteve o menor desempenho, apresentando quatro falhas. Ao contrastar esses dados com os resultados do VirusTotal, nota-se que os LLMs identificaram ameaças que passaram despercebidas até mesmo por motores consolidados, como os da Microsoft e da Kaspersky. Em cenários específicos, os LLMs detectaram amostras que não foram identificadas pelos *scanners* do VirusTotal. Esses resultados mostram que as IAs generativas são promissoras e complementares na ciberdefesa.

Como trabalhos futuros, propõe-se avaliar a resistência de modelos a ataques que busquem burlar a detecção de códigos maliciosos. Além disso, incluir programas benignos e realizar múltiplas execuções por condição permitirão analisar a variabilidade das respostas, complementando a ampliação da amostra de testes, etapa já em andamento.

Agradecimentos

Os autores declaram que o Gemini 3.8 Flash foi utilizado para revisão gramatical e padronização das referências bibliográficas no padrão adotado pela SBC.

Referências

- Fraunhofer FKIE. (s.d.). BPFdoor. Malpedia.
<https://malpedia.caad.fkie.fraunhofer.de/details/elf.bpfdoor>.
- 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.
- Jelodar, H., Bai, S., Hamed, P., Mohammadian, H., Razavi-Far, R., & Ghorbani, A. (2026). Large Language Model (LLM) for Software Security: Code Analysis, Malware Analysis, Reverse Engineering. *JISA'26*, 98, 104390.
- Kaspersky. (2025). Kaspersky Security Bulletin: Crime and malware statistics. Securelist. <https://securelist.com>.
- Qian, X., Zheng, X., He, Y., Yang, S., & Cavallaro, L. (2025). LAMD: Context-driven android malware detection and classification with LLMs. *IEEE SPW*, 126-136.
- Sihwail, R., Omar, K., & Ariffin, K. Z. (2018). A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis. *IJASEIT*, 8(4-2), 1662-1671.
- Stanford University. (2026). Artificial Intelligence Index Report 2026. Stanford Institute for Human-Centered Artificial Intelligence.
<https://hai.stanford.edu/ai-index/2026-ai-index-report>.
- VirusTotal. (s.d.). How it works. VirusTotal Docs.
<https://docs.virustotal.com/docs>.
- Xue, D., Zhao, G., Fan, Z., Li, W., Xu, Y., Liu, Z., ... & Yuan, Z. (2024). Poster: An exploration of large language models in malicious source code detection. In *ACM SIGSAC Conference on Computer and Communications Security*, 4940-4942.
- Zhao, W., Wu, J., & Meng, Z. (2025). Apppoet: Large language model based android malware detection via multi-view prompt engineering. *Expert Systems with Applications*, 262, 125546.