Análise Comparativa do Desempenho de LLMs e Ferramentas de Segurança na Detecção de Vulnerabilidades em Contratos Inteligentes

Eduardo Vinícius Audero da Costa, Carol Anely Miranda Guzman, Oscar Bruno Maciel de Abreu, Jeffson Celeiro Sousa, Alexandre Braga

[1] CPQD – Centro de Pesquisa e Desenvolvimento em Telecomunicações
eduardo.viniciusac@gmail.com, carolanely.academico@gmail.com, oscarbma@gmail.com, jcsousa@cpqd.com.br, ambraga@cpqd.com.br

Abstract. *The use of smart contracts in decentralized applications highlights the need for automated mechanisms to detect security vulnerabilities. Static analysis tools, such as Slither and Mythril, are widely used, but present limitations, including false positives and difficulty in capturing complex semantic patterns. Large Language Models (LLMs) show potential for contextual code analysis. This work presents an experimental comparative evaluation between static analysis tools and open-source LLMs in the analysis of smart contracts written in Solidity, indicating opportunities for complementarity toward more effective security audits.*

Resumo. *O uso de contratos inteligentes em aplicações descentralizadas evidencia a necessidade de mecanismos automatizados para a detecção de vulnerabilidades de segurança. Ferramentas de análise estática, como Slither e Mythril, são amplamente utilizadas, mas apresentam limitações, como falsos positivos e dificuldade em capturar padrões semânticos complexos. Modelos de linguagem de grande porte (LLMs) têm potencial para análise contextual de código. Este trabalho apresenta uma avaliação experimental comparativa entre ferramentas estáticas e LLMs open source na análise de contratos inteligentes em Solidity, indicando oportunidades de complementaridade para auditorias de segurança mais eficazes.*

1. Introdução

A adoção de contratos inteligentes em plataformas blockchain aumentou significativamente a exposição a falhas de segurança, com impacto financeiro direto. Esses contratos automatizam regras de negócio e transferências de valor sem intermediários, o que traz eficiência, mas também novos riscos. Como são imutáveis após a implantação, qualquer erro de desenvolvimento pode ser explorado de forma irreversível, gerando perdas expressivas e afetando a confiança nos ecossistemas que sustentam plataformas como o Ethereum.[Atzei et al. 2017].

Para mitigar esses riscos, ferramentas de análise estática, como Slither e Mythril, são amplamente utilizadas em auditorias automatizadas de contratos inteligentes. Essas ferramentas permitem análise rápida e reprodutível, por meio de detectores baseados em padrões de vulnerabilidade e da exploração de fluxos de execução, facilitando a identificação de falhas conhecidas. Apesar de sua ampla adoção, estudos empíricos

apontam limitações recorrentes, especialmente na detecção de falhas que dependem de contexto semântico mais complexo, além da geração de um elevado volume de alertas, o que aumenta o esforço de triagem manual. [Durieux et al. 2020].

Paralelamente, avanços recentes em modelos de linguagem de grande porte (Large Language Models – LLMs) motivaram sua investigação em tarefas de análise de código e segurança de software. Por realizarem inferência contextual sobre o código, esses modelos levantam a hipótese de comportamento distinto na detecção de vulnerabilidades, particularmente em cenários associados à lógica de negócio e à interação entre múltiplos trechos de código.

Apesar do crescimento dessa linha de pesquisa, ainda são escassas avaliações comparativas controladas que posicionem ferramentas de análise estática e LLMs sob condições experimentais equivalentes.

Diante desse cenário, este trabalho realiza uma avaliação experimental comparativa entre ferramentas tradicionais e LLMs open source na detecção de vulnerabilidades em contratos inteligentes escritos em Solidity. O objetivo é comparar diretamente o desempenho de Slither, Mythril e LLMs por meio de métricas comuns de detecção, analisando diferenças de cobertura e volume de alertas gerados. Aspectos operacionais específicos dos LLMs são avaliados de forma complementar, apenas para caracterizar sua viabilidade prática, sem desviar do foco central da comparação.

As principais contribuições deste trabalho são: (i) uma avaliação empírica direta e sistemática entre ferramentas de análise estática e LLMs sob métricas comuns e condições controladas; (ii) uma análise quantitativa do impacto dessas abordagens sobre o volume de alertas e o comportamento de super-sinalização; e (iii) evidências experimentais que contribuem para o entendimento do papel atual dos LLMs em relação às ferramentas tradicionais no contexto de auditorias de segurança de contratos inteligentes.

2. Fundamentação Teórica e Trabalhos Relacionados

2.1. Contratos Inteligentes e Vulnerabilidades

Em contratos inteligentes implantados em blockchain pública, falhas de implementação não podem ser corrigidas após a implantação, o que amplia o impacto de vulnerabilidades exploráveis e reforça a necessidade de mecanismos automatizados de detecção. Relatórios de incidentes Web3 indicam que, apenas em 2024, as perdas superaram US$ 1,4 bilhão em 149 casos envolvendo contratos inteligentes e protocolos DeFi, enquanto o total hackeado entre 2011 e 2025 soma US$ 26 bilhões [OWASP Foundation 2025, SolidityScan 2024].

Além do impacto econômico direto, a expansão dos contratos inteligentes para domínios críticos, como a saúde digital e a telemedicina, introduz riscos regulatórios severos. Nesses cenários governamentais ou institucionais, a exploração de vulnerabilidades pode resultar no vazamento de dados médicos sensíveis, acarretando violações diretas a regulamentações de privacidade (como a LGPD) e comprometendo irreversivelmente a confiança do paciente na infraestrutura.

A caracterização das falhas mais críticas também evoluiu nos últimos anos. Classificações recentes, como o *OWASP Smart Contract Top 10 (2025)*, mostram deslocamento do foco de erros aritméticos para vulnerabilidades associadas *a controle de acesso, erros de lógica de negócio, manipulação de oráculos, chamadas externas não verificadas e ataques*

com flash loans, mantendo reentrância e erros inteiros como categorias ainda relevantes. Dados de incidentes indicam que falhas de controle de acesso e de lógica concentram hoje o maior impacto financeiro agregado. [OWASP Foundation 2025, Kacherginsky 2024].

Essas vulnerabilidades decorrem de interações complexas entre estado e fluxo de execução, limitando detectores baseados em regras e motivando a comparação com abordagens baseadas em inferência contextual, como LLMs. [Durieux et al. 2020].

2.2. Ferramentas de Análise Estática

Ferramentas de análise estática são amplamente empregadas na auditoria de contratos inteligentes por permitirem a identificação automatizada de padrões de vulnerabilidade sem necessidade de execução do contrato.

Neste trabalho foram selecionadas as ferramentas **Slither** e **Mythril** como representantes de duas estratégias técnicas complementares e amplamente validadas na literatura. O *Slither* opera sobre o código-fonte Solidity, utilizando uma representação intermediária e um conjunto extensivo de detectores baseados em regras e análise de dependências, com foco em alta cobertura e baixo tempo de execução [Trail of Bits 2024]. O *Mythril*, por sua vez, atua principalmente sobre bytecode da Ethereum Virtual Machine e emprega execução simbólica e resolução de restrições para explorar caminhos de execução potencialmente exploráveis, sendo particularmente utilizado para detectar vulnerabilidades dependentes de estado e sequência de chamadas [Mueller 2024].

A escolha dessas ferramentas como baseline experimental se justifica por sua ampla adoção, natureza open source e uso recorrente como referência em benchmarks, permitindo comparação direta com abordagens baseadas em aprendizado de máquina e LLMs. [Durieux et al. 2020, Ince et al. 2025].

2.3. LLMs para Análise de Código e Segurança

Modelos de linguagem de grande porte (LLMs) vêm sendo explorados em tarefas de análise de código, incluindo identificação de vulnerabilidades. Diferentemente de ferramentas estáticas, operam por inferência contextual, permitindo capturar padrões semânticos associados à lógica de negócio e ao fluxo de execução.

Estudos empíricos recentes mostram resultados heterogêneos. Avaliações comparativas indicam que LLMs podem superar ferramentas tradicionais em classes específicas de vulnerabilidades, especialmente falhas de lógica e dependências contextuais, alcançando maior recall em certos cenários. Trabalhos mais recentes também destacam avanços com técnicas como *chain-of-thought prompting e self-consistency*, que melhoram a capacidade de raciocínio e análise estrutural do código. [Li et al. 2026]. Por outro lado, análises sistemáticas também demonstram que ferramentas estáticas ainda apresentam desempenho superior e maior consistência global em diversas categorias críticas, não sendo ainda substituíveis por abordagens baseadas exclusivamente em LLMs. [Ince et al. 2025].

Além disso, o desempenho de LLMs mostra alta sensibilidade ao protocolo experimental, à estratégia de prompt e às métricas utilizadas. A natureza probabilística desses modelos também introduz riscos de falsos positivos plausíveis (“alucinações”), o que exige validação adicional dos resultados. Por esse motivo, a literatura recente converge para a necessidade de avaliações controladas, com métricas comuns e condições equivalentes

de teste, posicionando LLMs como candidatos a complemento, e não substituição, das ferramentas estáticas.

3. Metodologia Experimental

3.1. Visão Geral da Metodologia

A metodologia consiste em uma avaliação experimental comparativa entre ferramentas de análise estática e LLMs para detecção de vulnerabilidades em contratos inteligentes. Todas as abordagens foram executadas de forma independente sobre o mesmo conjunto de contratos sanitizado, sem acesso a rótulos durante a análise. A validação foi realizada posteriormente por comparação com um conjunto de referência rotulado, garantindo execução cega, comparabilidade e reprodutibilidade.

No contexto de aplicações descentralizadas para serviços essenciais, as categorias listadas na Tabela 1 assumem criticidade ímpar. Uma falha de 'Access Control', por exemplo, poderia permitir que atores maliciosos ou não autorizados emitissem, alterassem ou revogassem credenciais verificáveis de saúde, corrompendo a integridade do histórico do paciente e invalidando regras de negócio estabelecidas.

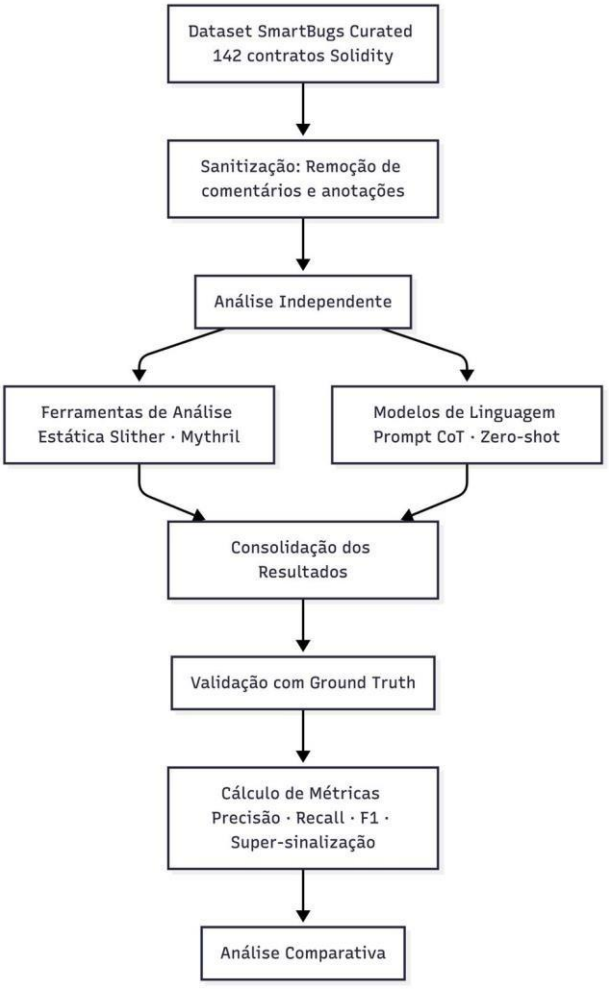

Figura 1. Fluxo Experimental da Metodologia Proposta

3.2. Dataset Utilizado

Foi utilizado o dataset *SmartBugs Curated* [Durieux et al. 2020], amplamente adotado na literatura como benchmark para avaliação de ferramentas de segurança. O conjunto é composto por 142 contratos previamente auditados e rotulados em múltiplas categorias de vulnerabilidade, incluindo reentrância, falhas de controle de acesso, erros aritméticos e negação de serviço. Para a comparação quantitativa central entre todas as abordagens, foi adotado o subconjunto comum de 100 contratos para os quais houve saída válida e comparável entre Slither, Mythril e os LLMs analisados, preservando uniformidade metodológica nas métricas reportadas.

Para evitar vazamento de informação contextual, foi realizada a sanitização manual dos contratos, removendo comentários e anotações que indicavam explicitamente a existência ou a localização das vulnerabilidades. Essa etapa é particularmente relevante para a avaliação de modelos de linguagem, que utilizam todo o conteúdo textual fornecido como entrada. A *versão sanitizada do dataset* foi utilizada como entrada comum para todas as abordagens avaliadas, enquanto a *versão original rotulada* foi mantida isolada e empregada exclusivamente na etapa de validação. Após a execução, as saídas dos LLMs foram normalizadas para categorias do SWC Registry, permitindo comparação com o conjunto de referência e com as ferramentas estáticas.

Ressalta-se que o dataset utilizado é composto exclusivamente por contratos vulneráveis, implicando a ausência de exemplos negativos (contratos seguros). Essa característica impede a estimação de métricas dependentes de verdadeiros negativos e falsos positivos, como precisão e especificidade, aspecto discutido na Seção 5 como ameaça à validade dos resultados.

Tabela 1. Distribuição das Vulnerabilidades no Dataset Smart Bugs Curated

Categoria	Descrição Breve
Reentrancy	Chamadas externas repetidas antes da atualização de estado.
Access Control	Falhas na restrição de quem pode executar funções críticas.
Arithmetic	Overflows e underflows de inteiros.
Unchecked Low Level Calls	Falha ao verificar o retorno de chamadas externas.
Denial of Service	Lógica que pode travar o contrato ou exceder o gas limit.
Bad Randomness	Uso de fontes previsíveis para aleatoriedade.
Time Manipulation	Dependência insegura de timestamps do bloco.
Short Addresses	Ataques relacionados à codificação de endereços (ERC20).
Front Running	Exploração da ordem de transações para vantagem indevida.
Other	Outras vulnerabilidades lógicas diversas.

3.3. Ferramentas e Modelos Avaliados

A avaliação contempla duas classes de abordagens: ferramentas consolidadas de análise estática e modelos de linguagem de grande porte. Foram selecionados *Slither* e *Mythril* como baseline devido à sua ampla adoção e uso recorrente na literatura. [Durieux et al. 2020, Torres et al. 2021]

Slither realiza análise estrutural sobre o código-fonte em Solidity por meio de detectores baseados em regras, apresentando alta eficiência de execução e boa cobertura de padrões conhecidos de vulnerabilidade[Feist et al. 2019], enquanto Mythril, opera

predominantemente sobre bytecode e emprega execução simbólica para explorar caminhos de execução potencialmente vulneráveis, sendo referência open source nesse paradigma de análise.

No caso dos modelos de linguagem, foram avaliados principalmente *LLMs open source*, incluindo *GPT-OSS-120B*, *Qwen-32B*, *Llama-Maverick-17B* e *Llama-3.3-8B*. Os modelos foram utilizados sem ajuste fino específico para o dataset e sem estratégias few-shot, refletindo um cenário de uso genérico. Foi adotado um prompt padronizado baseado em Chain-of-Thought, mantido constante entre os modelos, com o objetivo de reduzir variações metodológicas e favorecer comparabilidade entre arquiteturas.

A priorização de modelos abertos permite maior controle de versão, transparência de configuração e repetição independente dos experimentos, favorecendo a reprodutibilidade e a validação externa dos resultados, alinhando-se a recomendações recentes para avaliações comparativas em segurança. [Ince et al. 2025]

3.4. Métricas de Avaliação

A avaliação foi estruturada para permitir a comparação direta de desempenho entre ferramentas de análise estática (Slither e Mythril) e modelos de linguagem de grande porte (LLMs). O foco principal do estudo é a comparação de desempenho de detecção por meio de métricas comuns, derivadas de um conjunto de referência rotulado, garantindo equivalência de critério entre abordagens determinísticas e baseadas em inferência.

Métricas específicas de LLMs são incluídas apenas como *análise operacional complementar*, com o objetivo de caracterizar viabilidade prática, custo de execução e perfil de comportamento dos modelos.

Dessa forma, as métricas adotadas foram organizadas em dois grupos: (i) métricas comparáveis centrais de detecção; e (ii) métricas operacionais e comportamentais específicas de LLMs.

3.4.1. Métricas Comparáveis Centrais (Baseadas em Ground Truth)

A avaliação quantitativa principal é realizada por contrato, utilizando o dataset rotulado como referência de validação (*ground truth*), a partir do qual é construída a matriz de confusão de cada abordagem.

São considerados os seguintes eventos:

- *Verdadeiro Positivo (TP)*: contrato vulnerável corretamente sinalizado;
- *Verdadeiro Negativo (TN)*: contrato seguro corretamente classificado;
- *Falso Positivo (FP)*: sinalização de vulnerabilidade não correspondente ao rótulo de referência disponível, sendo tratado neste estudo como proxy operacional de sobre-detecção, no contexto de um dataset composto exclusivamente por contratos vulneráveis;
- *Falso Negativo (FN)*: contrato vulnerável não detectado.

A partir desses valores, são calculadas as métricas centrais de desempenho:

- *Precisão (Precision)* = TP / (TP + FP);

- *Recall* = TP / (TP + FN);
- *F1-score* = média harmônica entre precisão e recall.

Devido às características do dataset utilizado, composto exclusivamente por contratos vulneráveis, métricas dependentes de verdadeiros negativos não podem ser estimadas de forma estatisticamente completa. Nesse contexto, métricas baseadas em falsos positivos são interpretadas de forma operacional, especialmente no caso dos LLMs, como proxies de super-sinalização ou alucinação, definidas como a geração de diagnósticos plausíveis não alinhados ao rótulo de referência por contrato. Essa definição pode penalizar detecções não rotuladas no dataset, sendo considerada uma limitação experimental discutida na Seção 5.

Adicionalmente, a avaliação é realizada no nível de contrato, o que pode não capturar completamente a natureza multi-label das vulnerabilidades.

Considera-se detecção correta quando ao menos uma vulnerabilidade identificada corresponde a uma das categorias presentes no rótulo de referência do contrato. Essas métricas constituem a base principal das tabelas comparativas de desempenho apresentadas na Seção 4.

3.4.2. Métricas Operacionais e Comportamentais de LLM

Além das métricas centrais, são coletadas métricas comportamentais que caracterizam o perfil de detecção e o custo de triagem associado às saídas geradas:

- *Número de contratos classificados como vulneráveis*
- *Número médio de alertas por contrato*
- *Distribuição por categoria de vulnerabilidade detectada*
- *Grau de concordância entre métodos (interseção de contratos sinalizados)*

Essas medidas permitem analisar seletividade vs. super-sinalização, aspecto diretamente relacionado ao esforço de validação manual em auditorias.

Como análise complementar, são registradas métricas operacionais específicas dos LLMs, incluindo:

- *execução e robustez*: taxa de contratos concluídos com sucesso e interrupções por limite de contexto;
- *custo computacional*: tokens de prompt, tokens de completion, tokens totais e médios por contrato;
- *desempenho temporal*: latência total por lote, latência média por contrato e throughput;
- *estabilidade de resposta*: variação de explicações e indícios de degradação de consistência em execuções longas.

Os LLMs foram executados em lotes padronizados de contratos com prompt baseado em Chain-of-Thought, e os valores operacionais registrados são utilizados exclusivamente para análise de viabilidade prática e robustez de execução.

3.5. Pipeline Experimental e Protocolo de Execução

Os *contratos do dataset sanitizado* foram submetidos individualmente a cada abordagem avaliada, garantindo uniformidade de entrada e independência de execução. As ferramentas de análise estática foram executadas com configurações padrão recomendadas em sua documentação oficial. No caso dos LLMs, cada contrato foi analisado isoladamente com prompt padronizado, sem adaptações específicas e sem reexecuções condicionais. Em seguida, a resposta textual foi convertida em uma flag binária de detecção e em uma categoria principal de vulnerabilidade, permitindo a consolidação das métricas comparativas.

Esse procedimento também reduz risco de viés de confirmação: os modelos não receberam rótulos prévios, exemplos do próprio dataset nem indicação de qual vulnerabilidade deveria ser encontrada em cada contrato. A comparação com o ground truth foi realizada apenas após o término das execuções, durante a etapa de agregação dos resultados.

Após a execução, os resultados foram agregados por contrato e comparados com o conjunto de referência rotulado, permitindo o cálculo das métricas definidas na Seção 3.4. Todo o processo de agregação e avaliação foi implementado de forma determinística, assegurando rastreabilidade e reprodutibilidade experimental.

4. Resultados Experimentais

Esta seção apresenta os resultados da avaliação experimental comparativa. Os resultados são organizados de acordo com as métricas centrais de comparação definidas na Seção 3.4, seguidos por análises estratificadas por categoria de vulnerabilidade e por métricas operacionais complementares dos LLMs.

4.1. Desempenho Comparativo

A Tabela 2 apresenta os resultados centrais de desempenho comparativo entre as ferramentas de análise estática *Slither* e *Mythril* e os modelos de linguagem, utilizando métricas comuns baseadas no conjunto de referência rotulado. São reportadas as métricas de *Precisão, Recall e F1-score*, bem como a taxa de falsos positivos interpretada operacionalmente como proxy de super-sinalização, além do número de contratos sinalizados e do volume médio de alertas por contrato.

As ferramentas estáticas Slither e Mythril obtiveram os seguintes resultados globais: o **Mythril** obteve *Precisão igual a 1,00, Recall de 0,91 e F1-score de 0,95*, enquanto o **Slither** apresentou *Precisão igual a 1,00, Recall de 0,90 e F1-score de 0,94*, respectivamente. Esses valores servem como linha de base determinística para comparação com os modelos de linguagem, uma vez que ambas ferramentas operam com detectores especializados e regras explícitas de análise.

Entre os modelos de linguagem avaliados, observa-se variação significativa de desempenho entre arquiteturas. O modelo Qwen apresentou o maior valor de Recall entre os LLMs avaliados (≈ 80 na amostra analisada), acompanhado de baixa taxa de super-sinalização (~ 1%). O modelo Llama-Maverick apresentou desempenho próximo (≈ 74% de Recall), enquanto o Llama-3.3-8B apresentou valores inferiores de cobertura (≈ 31% de Recall).

O modelo GPT-OSS-120B apresentou desempenho intermediário em termos de detecção (≈ 49% de Recall considerando apenas execuções concluídas), porém com

taxa relevante de falhas operacionais por limite de contexto (≈ 28%), o que afeta sua comparabilidade direta. Por esse motivo, seus indicadores são reportados apenas para os casos concluídos, sendo essa limitação analisada na Seção 4.4.

De forma geral, os resultados indicam diferenças estruturais entre abordagens determinísticas e baseadas em inferência contextual. As ferramentas estáticas tendem a gerar *maior volume de alertas por contrato*, enquanto os LLMs apresentaram comportamento mais seletivo, o que implica menor carga de triagem manual, porém potencial variação de cobertura entre modelos. Essa diferença de perfil é analisada de forma estratificada por categoria de vulnerabilidade na Seção 4.2.

Tabela 2. Desempenho Comparativo de Detecção

Abordagem	Precisão	Recall	F1-score	FPR*
Mythril	1,00	0,91	0,95	–
Slither	1,00	0,90	0,94	–
Qwen (Reasoning)	–	0,80	–	1%
Llama-Maverick	–	0,74	–	1%
Llama-3.3-8B	–	0,31	–	1%
GPT-OSS	–	0,49	–	1%

*FPR interpretada operacionalmente como proxy de super-sinalização.

4.2. Cobertura por Vulnerabilidade

A Tabela 3 apresenta a cobertura de detecção por categoria de vulnerabilidade, medida em termos de Recall por classe. Essa análise complementa os resultados globais ao evidenciar diferenças de especialização entre ferramentas de análise estática e modelos de linguagem.

As ferramentas estáticas demonstram alta consistência na identificação de categorias clássicas baseadas em padrões estruturais, como *Reentrancy*, *Arithmetic* e *Access Control*, mantendo elevados valores de recall. Esse comportamento reflete a eficácia de detectores especializados e é consistente com o caráter determinístico dessas ferramentas.

Entre os modelos de linguagem, observa-se maior variação de cobertura entre categorias. Modelos com maior capacidade de raciocínio, como Qwen e Llama-Maverick, apresentam melhor desempenho relativo em vulnerabilidades associadas à lógica de negócio e ao fluxo de execução, enquanto modelos menores, como o Llama-3.3-8B, mostram queda de recall em cenários que exigem maior encadeamento lógico.

A análise dos LLMs também revelou heterogeneidade na forma de descrição das falhas. Uma mesma vulnerabilidade pode ser expressa por diferentes formulações, exigindo normalização semântica para consolidação dos resultados, etapa desnecessária nas ferramentas estáticas, cujas saídas já são estruturadas.

Quando corretos, os LLMs tendem a fornecer descrições mais explicativas e contextualizadas; quando incorretos, os erros geralmente aparecem como confusão entre classes semanticamente próximas ou generalização de riscos plausíveis não alinhados ao rótulo de referência.

Para o modelo GPT-OSS-120B, a análise considera apenas execuções concluídas com sucesso, sendo seus resultados interpretados também à luz do comportamento operacional (Seção 4.3).

Casos de divergência entre abordagens foram observados, com vulnerabilidades identificadas exclusivamente por ferramentas estáticas ou por modelos de linguagem, evidenciando diferenças entre detecção baseada em padrões estruturais e inferência contextual.

De forma geral, o desempenho varia entre categorias: ferramentas estáticas apresentam cobertura mais estável em vulnerabilidades recorrentes, enquanto LLMs mostram maior sensibilidade em cenários dependentes de contexto, ainda que com maior variabilidade entre modelos.

A análise por categoria possui caráter predominantemente qualitativo, dado o tamanho da amostra e a distribuição desigual das vulnerabilidades. Ainda assim, os padrões observados são consistentes com o comportamento esperado das abordagens e com a literatura.

Tabela 3. Cobertura por vulnerabilidades

Modelo	% Concluídos	Limite de Contexto	Interrupções
GPT-OSS-120B	72%	sim	não
Llama-3.3-8B	100%	não observado	não
Llama-Maverick	100%	não observado	não
Qwen	100%	não observado	não

4.3. Eficiência Operacional LLM

A eficiência operacional dos modelos de linguagem foi avaliada por meio de métricas de custo computacional e desempenho temporal, incluindo consumo médio de tokens por contrato, latência média de inferência e taxa de conclusão. Essas métricas são apresentadas como análise complementar de viabilidade prática.

A Tabela 4 apresenta os indicadores operacionais obtidos em execuções padronizadas com prompt baseado em Chain-of-Thought, evidenciando variação significativa entre os modelos quanto ao custo de geração e ao tempo de resposta.

O modelo *Qwen (Reasoning)* apresentou o maior consumo médio de tokens por contrato (2.609 tokens) e a maior latência média (13,24 s), refletindo maior complexidade e extensão das respostas geradas. Em contrapartida, o modelo *Llama-Maverick* apresentou um perfil mais equilibrado, com consumo médio de 1.405 tokens por contrato e latência média de 9,15 s, configurando um ponto intermediário entre custo e desempenho temporal.

O modelo *Llama-3.3-8B* apresentou consumo médio de 1.646 tokens por contrato e latência média de 9,09 s, porém com eficácia de detecção significativamente inferior, conforme discutido na Seção 4.1. Já o modelo *GPT-OSS* teve sua avaliação operacional limitada pela elevada taxa de falhas de execução (28%), o que inviabilizou a mensuração consistente de métricas de custo e latência.

De modo geral, observa-se uma relação consistente entre o volume de tokens gerados e a latência média de execução, indicando que respostas mais extensas tendem a implicar maior custo temporal. Esse comportamento evidencia um trade-off entre profundidade de análise e eficiência operacional entre os modelos avaliados.

Esses resultados são apresentados com o objetivo de caracterizar o comportamento de execução dos LLMs e sua viabilidade prática, sendo analisados em conjunto com os indicadores de robustez discutidos na Seção 4.4.

Tabela 4. Eficiência Operacional dos LLMs

Modelo	Recall	Tokens/Contrato	Latência Média	Obs
Qwen (Reasoning)	80%	2.609	13,24s	Melhor eficácia
Llama-Maverick	74%	1.405	9,15s	Trade-off balanceado
Llama-3.3-8B	31%	1.646	9,09s	Baixa eficácia
GPT-OSS	49%	N/A	11,78s	28% de falha

4.4. Robustez de Execução

A robustez de execução dos LLMs foi avaliada considerando estabilidade de processamento, taxa de conclusão e falhas associadas a limites operacionais. Foram registrados o percentual de contratos concluídos, interrupções por limite de contexto e falhas de processamento.

Os modelos da família Llama (3.3-8B e Maverick-17B) apresentaram taxa de conclusão de 100% nos lotes executados, sem interrupções por limite de contexto ou falhas de geração. O modelo GPT-OSS-120B apresentou maior incidência de interrupções (~28% dos contratos), associadas a limitações de contexto e encerramento antecipado da geração.

Também foram observadas variações na consistência das respostas em execuções longas para alguns modelos, caracterizadas por redução progressiva no nível de detalhamento após múltiplas inferências sequenciais. Neste estudo, esse comportamento é tratado como um indicativo exploratório de degradação de consistência, sendo considerado apenas como observação qualitativa complementar.

Esses resultados indicam que, além da capacidade de detecção, a estabilidade de execução e a tolerância a limites de contexto são fatores relevantes para a aplicação prática de LLMs em ambientes de auditoria automatizada.

5. Discussão e Ameaças à Validade

Os resultados experimentais apresentados neste trabalho demonstram que ferramentas tradicionais de análise estática e LLMs apresentam *perfis de detecção distintos e complementares* na identificação de vulnerabilidades em contratos inteligentes. Em vez de indicar superioridade absoluta de uma abordagem sobre outra, a avaliação evidencia diferenças estruturais nos mecanismos de sinalização, com implicações práticas diretas para auditorias de segurança em ambientes blockchain.

As ferramentas de análise estática, representadas por Slither e Mythril, mantêm comportamento mais abrangente e sistemático, com maior volume de alertas gerados por contrato, oferecendo ampla cobertura de padrões bem formalizados, ainda que com maior esforço de triagem manual. Tais resultados reforçam observações da literatura, que apontam alta sensibilidade dessas ferramentas para classes estruturais de vulnerabilidade, acompanhada por super-sinalização operacional [Durieux et al. 2020, Alves and Henriques 2025].

Por outro lado, os LLMs avaliados apresentaram comportamento mais seletivo, com menor número de contratos sinalizados e menor volume médio de alertas. Modelos com

maior capacidade de raciocínio contextual, como Qwen (Reasoning) e Llama-Maverick, alcançaram maior cobertura relativa *entre os modelos de linguagem avaliados*, particularmente em cenários associados à lógica de negócio e ao fluxo de execução. Em contraste, modelos menores apresentaram limitações evidentes de cobertura, indicando dificuldade em lidar com encadeamentos semânticos mais complexos. Esses resultados destacam a heterogeneidade do desempenho de LLMs em tarefas de segurança de código, fortemente dependente da arquitetura, do protocolo experimental e do tipo de vulnerabilidade analisada [Ince et al. 2025, Li et al. 2025].

A análise por categoria de vulnerabilidade reforça que *nenhuma das abordagens avaliadas apresenta desempenho uniforme em todas as classes de falha*. Ferramentas estáticas mantêm maior estabilidade em vulnerabilidades associadas a padrões recorrentes e formalizáveis, enquanto LLMs demonstram maior sensibilidade relativa em cenários dependentes de contexto semântico e interação entre múltiplos trechos de código.

As métricas operacionais evidenciam que custo computacional, latência e robustez variam significativamente entre modelos de linguagem. Embora não sejam o foco principal da comparação, esses fatores impactam diretamente a viabilidade prática dos LLMs em pipelines de auditoria. Em especial, instabilidades como falhas por limite de contexto indicam que a confiabilidade operacional é tão relevante quanto a capacidade de detecção em cenários reais.

De forma consolidada, os resultados posicionam os LLMs *como ferramentas complementares às soluções tradicionais de análise estática*, e não como substitutos diretos, reforçando seu papel na análise semântica e contextual de vulnerabilidades.

Este estudo está sujeito a limitações que devem ser consideradas na interpretação dos resultados. A principal ameaça à validade decorre da composição do dataset utilizado. O SmartBugs Curated é composto exclusivamente por contratos vulneráveis, o que inviabiliza a estimação estatística completa de métricas dependentes de verdadeiros negativos. Em função disso, a taxa de falsos positivos foi interpretada de forma operacional, como proxy de super-sinalização, sendo utilizada apenas para comparação relativa de comportamento entre abordagens.

Outra limitação está relacionada ao tamanho reduzido da amostra avaliada. Embora o SmartBugs Curated seja amplamente adotado como benchmark na literatura, o recorte quantitativo comparável deste estudo concentrou-se em 100 contratos comuns entre todas as abordagens, o que restringe análises estatísticas mais refinadas e pode amplificar variações específicas de determinados contratos ou categorias de vulnerabilidade.

No caso dos LLMs, o desempenho observado é sensível à estratégia de prompt e à ausência de ajuste fino específico para o domínio. Ainda que um protocolo padronizado tenha sido adotado para garantir comparabilidade, estratégias alternativas poderiam impactar os resultados. Ademais, a natureza probabilística dos LLMs pode introduzir variabilidade entre execuções, mesmo sob condições controladas.

Por fim, limitações operacionais, como falhas por limite de contexto e interrupções de execução observadas em alguns modelos, afetam a comparabilidade direta e a aplicabilidade prática dos resultados. Essas limitações foram explicitamente registradas e consideradas na análise, reforçando a necessidade de avaliações futuras em ambientes com maior estabilidade operacional.

Apesar dessas restrições, o protocolo experimental adotado assegura execução cega, métricas comuns e condições equivalentes de avaliação, permitindo comparações consistentes entre abordagens e posicionando adequadamente os resultados no contexto da literatura atual.

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma avaliação experimental comparativa entre ferramentas tradicionais de análise estática e modelos de linguagem de grande porte na detecção de vulnerabilidades em contratos inteligentes escritos em Solidity. Os resultados indicam diferenças consistentes não apenas entre essas duas classes de abordagens, mas também entre as diferentes arquiteturas de LLMs avaliadas.

Ferramentas estáticas, como Slither e Mythril, mantêm comportamento estável e abrangente na identificação de padrões conhecidos, consolidando seu papel como base em auditorias automatizadas, embora com maior volume de alertas e esforço de validação manual. No caso dos modelos de linguagem, os resultados evidenciam desempenho heterogêneo entre as arquiteturas. Modelos com maior capacidade de raciocínio e geração estruturada de respostas apresentaram maior cobertura relativa entre os LLMs avaliados, enquanto modelos menores demonstraram limitações significativas de detecção. Esses achados indicam que o desempenho dos LLMs não é uniforme e depende fortemente da arquitetura e do protocolo experimental adotado.

De forma agregada, os resultados indicam que abordagens distintas atendem a etapas diferentes do processo de auditoria, reforçando a necessidade de estratégias combinadas. Ferramentas estáticas são mais adequadas para varredura ampla e sistemática, enquanto LLMs agregam valor na análise contextual e na interpretação de achados, especialmente em cenários que envolvem interações complexas entre trechos de código.

Essa complementaridade sugere que pipelines híbridos, combinando ferramentas determinísticas e modelos de linguagem, representam uma direção promissora para auditorias de contratos inteligentes mais eficazes.

Como trabalhos futuros, destacam-se: (i) a ampliação da avaliação para datasets balanceados, de maior escala, com ênfase em contratos de aplicações institucionais reais, como sistemas de Identidade Digital Descentralizada (IDD); (ii) a inclusão de outras ferramentas e arquiteturas de LLMs; (iii) a investigação sistemática de pipelines híbridos, explorando o uso integrado de LLMs com ferramentas tradicionais; e (iv) a padronização automática das saídas dos LLMs para categorias de vulnerabilidade comparáveis.

Esses resultados reforçam que o avanço na auditoria de contratos inteligentes não reside na substituição de abordagens existentes, mas na integração de técnicas complementares capazes de explorar simultaneamente padrões estruturais e inferência contextual.

7. Agradecimentos

Este trabalho foi desenvolvido no âmbito do Programa Residência Tecnológica, coordenado pela Softex, com apoio do Ministério da Ciência, Tecnologia e Inovações (MCTI) e financiamento por meio de recursos da Lei nº 8.248, de 23 de outubro de 1991, conforme o Regulamento 002/2025. no âmbito do PPI SOFTEX (coordenado pela Softex e publicado como Residência em TIC 11, DOU 01245.011733/2022-83).

As atividades contaram com o apoio institucional da Pontifícia Universidade Católica do Paraná (PUCPR) e do Centro de Pesquisa e Desenvolvimento em

Telecomunicações (CPQD).

Os autores agradecem o apoio dos recursos financeiros da FUNTTEL, administrados pela FINEP, especificamente no âmbito do projeto "5G Saúde - Segurança, Privacidade, Inclusão e Qualidade em Telemedicina no Contexto da Web 3.0", sob o Acordo nº 01.23.0468.00, Referência 0844/23.

Referências

Alves, R. S. R. R. and Henriques, M. A. (2025). Regressão da eficácia de analisadores de vulnerabilidades em contratos inteligentes de block-chains. In *Anais do Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)*.

Atzei, N., Bartoletti, M., and Cimoli, T. (2017). A survey of attacks on Ethereum smart contracts (SoK). In *Principles of Security and Trust (POST)*, volume 10204, pages 164–186.

Durieux, T., Ferreira, J. F., Abreu, R., and Cruz, P. (2020). Empi-rical review of automated analysis tools on 47,587 Ethereum smart contracts. ar-Xiv:1910.10601.

Feist, J., Grieco, G., and Groce, A. (2019). Slither: A static analysis framework for smart contracts. IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB).

Ince, P., Yu, J., Liu, J. K., and Du, X. (2025). Utilização de modelos gene-rativos de linguagem ampla na detecção de vulnerabilidades em contratos inteligentes. arXiv:2504.04685v1 [cs.CR].

Kacherginsky, P. (2024). Top DeFi attack vectors – 2024. Relatório técnico sobre vetores de ataque em protocolos DeFi. Publicado pelo autor.

Li, X. et al. (2026). Automatic contrastive chain-of-thought prompting. *Expert Systems with Applications*.

Li, Z., Li, X., Li, W., and Wang, X. (2025). SCALM: Detecting bad practices in smart contracts through LLMs. *arXiv*.

Mueller, B. (2024). Mythril: Security analysis tool for Ethereum smart contracts. GitHub repository.

OWASP Foundation (2025). OWASP smart contract top 10. `https://owasp.org/www-project-smart-contract-top-10/`.

SolidityScan (2024). Web3HackHub: Blockchain security incidents and losses database. SolidityScan Research.

Torres, C. F. et al. (2021). Smart contract security: a practitioners' perspective. arXiv.

Trail of Bits (2024). Slither: A static analysis framework for smart contracts. GitHub repository.