SolRAG: Detecção de Vulnerabilidades em Contratos Inteligentes Solidity com Geração Aumentada via Recuperação

Luís H. S. de Carvalho[1], Josué N. Campos[1], Arastoo Zibaeirad[2], Marco Vieira[2] e José A. M. Nacif[1]

[1]Universidade Federal de Viçosa (UFV)
Florestal, MG – Brasil

[2]University of North Carolina at Charlotte
Charlotte, NC – USA

{luis.h.carvalho,josue.campos,jnacif}@ufv.br

{azibaeira,marco.vieira}@charlotte.edu

Abstract. *Detecting vulnerabilities in smart contracts is essential for the security of blockchain-based applications. Traditional static analysis tools often generate large numbers of false positives, increasing the cost of manual auditing. In this work, we propose SolRAG, a Retrieval-Augmented Generation (RAG) approach for vulnerability detection in Solidity smart contracts. The method performs vulnerability-oriented analysis and incorporates semantically similar vulnerable examples into the model context. Experiments on a benchmark dataset containing 4,270 queries show that the proposed approach reduces the false positive rate from 38.1% to 2.8% while increasing the proportion of true negatives from 58.1% to 93.9%. Additionally, the system correctly identified 37 vulnerable contracts compared to 32 detected by the baseline method.*

Resumo. *A detecção de vulnerabilidades em contratos inteligentes é essencial para a segurança de aplicações baseadas em blockchain. Ferramentas tradicionais de análise estática frequentemente produzem grande quantidade de falsos positivos, aumentando o custo da auditoria manual. Neste trabalho, nós propomos SolRAG, uma abordagem baseada em Retrieval-Augmented Generation (RAG) para análise de vulnerabilidades em contratos inteligentes escritos em Solidity. O método realiza análise orientada por vulnerabilidade e utiliza exemplos recuperados por similaridade semântica para compor o contexto do modelo. Experimentos em um benchmark com 4270 consultas mostram que a abordagem reduz a taxa de falsos positivos de 38,1% para 2,8% e aumenta a proporção de verdadeiros negativos de 58,1% para 93,9%. Além disso, o sistema identificou corretamente 37 contratos vulneráveis, em comparação com 32 do método de referência.*

1. Introdução

Contratos inteligentes tornaram-se um dos principais componentes das aplicações descentralizadas baseadas em blockchain, especialmente em plataformas como Ethereum

[Oliva et al. 2020]. Esses contratos permitem a execução automática e imutável de regras de negócio, eliminando intermediários e aumentando a transparência das transações. Nos últimos anos, o uso dessas aplicações tem crescido significativamente, ampliando também a superfície de ataques associados a falhas de segurança. A própria imutabilidade que garante confiabilidade também intensifica o impacto de vulnerabilidades, uma vez que contratos implantados não podem ser facilmente corrigidos. Ainda que abordagens como contratos com paradigma *upgradable* permitam atualizações via proxy, isso pode introduzir complexidade adicional de modo que a detecção preventiva de falhas permaneça essencial [Khan et al. 2021]. Como consequência, falhas podem resultar em perdas financeiras irreversíveis, interrupções de serviço e exploração maliciosa em larga escala [Hewa et al. 2021]. Nesse contexto, a detecção preventiva de vulnerabilidades é fundamental para garantir a segurança e a confiabilidade das aplicações descentralizadas.

Apesar dos avanços na área, a detecção automática de vulnerabilidades em contratos inteligentes ainda apresenta desafios significativos. Ferramentas tradicionais baseiam-se predominantemente em análise estática, execução simbólica e regras pré-definidas para identificar padrões conhecidos de falhas [Kushwaha et al. 2022]. Embora essas abordagens sejam eficazes em cenários específicos, frequentemente apresentam limitações de generalização para novos padrões de vulnerabilidade e dificuldades de escalabilidade [Qian et al. 2022].

Nos últimos anos, Modelos de Linguagem de Grande Escala (*Large Language Models* – LLMs) têm demonstrado avanços expressivos na compreensão e análise de código-fonte [Nam et al. 2024]. Esses modelos são capazes de capturar relações semânticas complexas e realizar inferências sobre estruturas lógicas, o que sugere potencial para aplicação na auditoria automatizada de contratos inteligentes [Boi et al. 2024]. No entanto, o uso direto de LLMs para detecção de vulnerabilidades ainda apresenta limitações importantes. Entre os principais desafios estão a geração de alucinações, inconsistência entre execuções e alta sensibilidade à formulação do *prompt*. Além disso, a ausência de contexto específico do domínio pode levar a variações significativas de desempenho entre diferentes categorias de vulnerabilidades [He et al. 2024].

Uma abordagem promissora para mitigar essas limitações é o uso de Geração Aumentada via Recuperação (*Retrieval-Augmented Generation* - RAG). A técnica RAG incorpora informações externas relevantes ao processo de inferência do modelo. Ao recuperar exemplos semanticamente similares e integrá-los ao contexto de entrada, sistemas baseados em RAG podem direcionar o raciocínio do modelo, reduzir ambiguidades e aumentar a fundamentação das respostas [Lewis et al. 2020]. Entretanto, a aplicação de RAG na análise de segurança de contratos inteligentes ainda é emergente na literatura, especialmente em abordagens que busquem controlar explicitamente o processo de inferência e reduzir a incidência de falsos positivos.

Nesse sentido, observam-se lacunas importantes nas abordagens existentes, incluindo a alta taxa de falsos positivos em métodos automatizados [Hu et al. 2023, David et al. 2023], a dificuldade de generalização para diferentes categorias de vulnerabilidade [Xiao et al. 2025], a ausência de controle estruturado sobre o processo de inferência dos modelos [Hu et al. 2023] e o uso predominante de estratégias de detecção simultânea de múltiplas vulnerabilidades, o que pode introduzir ambiguidades na análise [Bani-Hani et al. 2024].

Neste trabalho, propomos SolRAG, uma abordagem baseada em RAG para análise de segurança de contratos inteligentes escritos em Solidity. Em relação a abordagens RAG existentes, o método proposto adota uma estratégia de análise orientada por vulnerabilidade, avaliando uma categoria de falha por vez, em vez de realizar detecção multi-rótulo em uma única execução. Para cada vulnerabilidade alvo, o sistema recupera exemplos previamente rotulados como vulneráveis a partir de uma base de conhecimento curada e os utiliza como contexto semântico para guiar a inferência do modelo.

A abordagem proposta combina um mecanismo de recuperação vetorial com LLMs, permitindo incorporar exemplos semanticamente relevantes diretamente no processo de análise. Além disso, o sistema utiliza uma saída estruturada, possibilitando a extração automática de métricas e garantindo maior consistência experimental. Essa estratégia busca reduzir ambiguidades, limitar a superdetecção de vulnerabilidades e aumentar a estabilidade do processo de auditoria automatizada.

As principais contribuições deste trabalho são:

- Propor uma abordagem baseada em RAG para detecção de vulnerabilidades em contratos inteligentes escritos em Solidity, combinando recuperação semântica com análise orientada por vulnerabilidade e saída binária estruturada;
- Introduzir uma estratégia de análise orientada por vulnerabilidade, na qual cada execução avalia apenas uma categoria de falha por vez, reduzindo ambiguidades na inferência;
- Construir e disponibilizar uma base de conhecimento curada com aproximadamente 1.200 contratos inteligentes vulneráveis, cobrindo diversas categorias de falhas;
- Avaliar a abordagem em um benchmark com 4.270 consultas, demonstrando redução significativa na taxa de falsos positivos (de 38,1% para 2,8%), aumento na proporção de verdadeiros negativos (de 58,1% para 93,9%) e identificação correta de 37 contratos vulneráveis, contra 32 do método de referência.

O restante deste artigo está organizado da seguinte maneira. A Seção 2 apresenta o referencial teórico e trabalhos relacionados que embasam o trabalho. A Seção 3 descreve a metodologia proposta, incluindo a arquitetura e as estratégias adotadas. Em seguida, a Seção 4 apresenta a configuração experimental, detalhando os *datasets* e as métricas de avaliação utilizadas. A Seção 5 apresenta e analisa os resultados obtidos, e a Seção 6 discute as implicações e limitações do estudo. Por fim, a Seção 7 apresenta as conclusões e possíveis direções para trabalhos futuros.

2. Trabalhos Relacionados

LLMs têm impulsionado novas abordagens para análise automatizada de código, incluindo a detecção de vulnerabilidades em contratos inteligentes. Diferentemente de técnicas tradicionais de análise ou modelos de aprendizado de máquina, LLMs são capazes de capturar relações semânticas complexas presentes no código-fonte, permitindo interpretar padrões de vulnerabilidade de forma contextualizada. Esses modelos são treinados em grandes volumes de dados textuais e de código, o que lhes permitem realizar tarefas como compreensão de código, geração de correções e identificação de possíveis falhas. Em particular, LLMs podem analisar trechos de código de contratos inteligentes e identificar padrões associados a vulnerabilidades conhecidas, além de fornecer explicações e recomendações [He et al. 2024].

Modelos como GPT-4 e variantes open-source têm demonstrado capacidade de compreender código-fonte e identificar padrões complexos de vulnerabilidade. Trabalhos como GPTLens [Hu et al. 2023] exploram o uso de *prompting* estruturado (elaboração de instruções direcionadas ao modelo) para realizar auditorias automatizadas utilizando modelos de linguagem, empregando um processo em múltiplas etapas para gerar e validar hipóteses de vulnerabilidade.

Outros estudos investigam a integração entre LLMs e mecanismos de recuperação de conhecimento externo por meio de RAG. Em [Yu 2024] é proposto um sistema que combina GPT-4 com um repositório vetorial contendo contratos vulneráveis previamente conhecidos, permitindo que o modelo utilize exemplos recuperados durante a análise para melhorar a detecção de falhas. Nesse sistema, *embeddings* (representações vetoriais dos trechos de código que capturam a similaridade semântica entre eles) são gerados a partir de contratos vulneráveis e armazenados em um banco vetorial, sendo posteriormente recuperados durante a inferência para auxiliar o modelo na identificação de vulnerabilidades.

De forma semelhante, o framework RAG-SmartVuln integra modelos de linguagem ajustados por *fine-tuning* com um mecanismo de recuperação baseado em um grafo de conhecimento de vulnerabilidades derivado do SWC Registry [Nhu et al. 2025]. Esse sistema utiliza relações semânticas entre comportamentos de código, causas de vulnerabilidades e possíveis correções para enriquecer o contexto fornecido ao modelo durante a análise. Os resultados demonstram que a combinação entre RAG e modelos *fine-tuned* pode melhorar significativamente métricas como precisão e F1-Score em *benchmarks* como SolidiFI e SmartBugs.

Em [David et al. 2023] é apresentada uma avaliação abrangente do uso de LLMs para auditorias de segurança em contratos inteligentes, analisando os modelos GPT-4-32k e Claude-v1.3-100k em um conjunto de 52 contratos DeFi comprometidos, abrangendo 38 categorias de vulnerabilidade. Os resultados indicam que ambos os modelos em conjunto, identificam corretamente o tipo de vulnerabilidade em 40% dos casos, porém com alta taxa de falsos positivos, limitando a aplicação prática dessas ferramentas sem supervisão humana.

Tabela 1. Comparação com abordagens existentes para detecção de vulnerabilidades em contratos inteligentes.

Critérios	**SolRAG**	**GPTLens**	**RAG-LLM**	**RAG-SmartVuln**
Uso de RAG	✓	×	✓	✓
Dataset curado de contratos vulneráveis	✓	△	△	△
Análise específica por vulnerabilidade	✓	×	△	△
Saída binária estruturada	✓	△	✓	△

✓ Totalmente abordado △ Parcialmente abordado × Não abordado

Em comparação com trabalhos anteriores, o SolRAG se diferencia por combinar três aspectos ainda não consolidados em conjunto na literatura: o uso de uma base curada de contratos auditados como conhecimento RAG, a análise orientada por vulnerabilidade específica, na qual cada categoria de falha é avaliada individualmente, reduzindo ambi-

guidades e uma saída binária estruturada que facilita a avaliação sistemática. A Tabela 1 resume essa comparação.

3. Metodologia

3.1. Visão Geral do Sistema

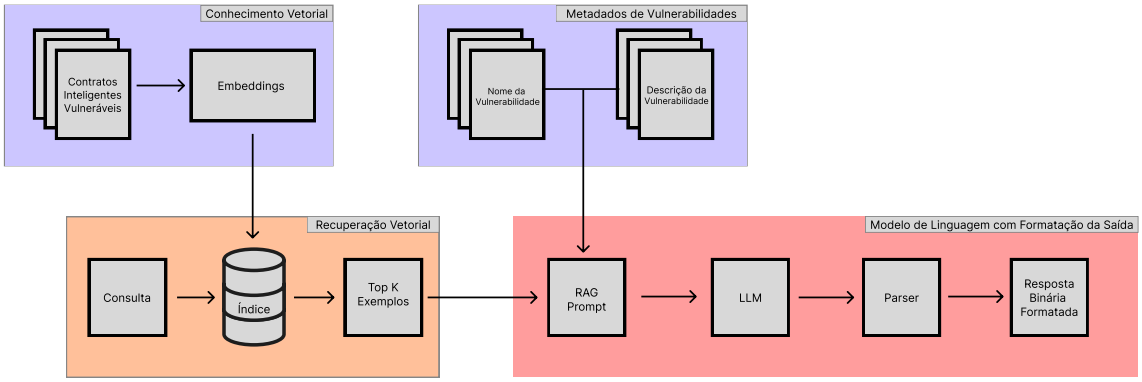

Figura 1. Arquitetura do sistema proposto baseado em RAG.

A abordagem proposta consiste em uma arquitetura modular, baseada em RAG para detecção de vulnerabilidades em contratos escritos em Solidity. A abordagem é orientada por vulnerabilidade específica, ou seja, cada contrato é avaliado separadamente uma categoria por vez, visando reduzir ambiguidade e alucinação.

A arquitetura é composta por três componentes principais: base de conhecimento vetorial, mecanismo de recuperação e modelo de linguagem com formatação da saída gerada. A orquestração é realizada por um grafo de estados implementado com LangGraph, permitindo controle das etapas de recuperação, construção de contexto, inferência e avaliação das respostas. A Figura 1 exemplifica de maneira gráfica a topologia de toda a abordagem proposta.

Inicialmente o sistema recebe como entrada o código fonte de um contrato e a definição textual da vulnerabilidade. Após isso, realiza-se a recuperação de exemplos semanticamente similares previamente rotulados como vulneráveis àquela categoria específica. Esses exemplos são armazenados em uma base vetorial construída a partir de *embeddings* do código-fonte, permitindo busca por similaridade semântica.

Os exemplos recuperados são então incorporados ao prompt, enviado ao modelo de linguagem juntamente com a descrição formal da vulnerabilidade e o código do contrato sob análise. Essa etapa visa fornecer contexto adicional que auxilie o modelo a identificar padrões relevantes, sem assumir automaticamente a presença da falha.

Após a inferência do modelo, a saída é processada por um mecanismo de *parsing* (uma interpretação e extração estruturada de informações a partir do texto gerado), garantindo que a resposta siga um formato padronizado. Esse controle permite extrair automaticamente a decisão de classificação (presença ou ausência da vulnerabilidade) e registrar justificativas associadas, assegurando reprodutibilidade e consistência experimental.

3.2. Estratégia de Análise por Vulnerabilidade

A detecção de vulnerabilidades em contratos inteligentes frequentemente envolve múltiplas categorias de falhas, cada uma caracterizada por padrões de código, condições

de execução e implicações de segurança distintas. Em muitas abordagens automatizadas, diferentes vulnerabilidades são analisadas simultaneamente, tratando o problema como uma tarefa de classificação multi-rótulo [Bani-Hani et al. 2024]. Embora esse tipo de abordagem permita avaliar várias categorias em uma única execução, ele também pode introduzir ambiguidade na análise, dificultando a distinção entre padrões específicos de falhas e aumentando a ocorrência de falsos positivos.

Neste trabalho foi adotada uma estratégia alternativa baseada na análise orientada por vulnerabilidade, na qual cada execução do SolRAG considera apenas uma categoria de vulnerabilidade por vez. Nesse cenário, o modelo recebe como contexto a definição textual da vulnerabilidade alvo e exemplos de código associados a essa categoria, direcionando a análise exclusivamente para a identificação daquele tipo específico de falha. Essa estratégia reduz o espaço de busca da inferência, permitindo que o modelo concentre sua atenção em padrões semânticos e estruturais relevantes para a vulnerabilidade analisada.

Essa abordagem facilita a construção de prompts mais especializados e o uso de mecanismos de recuperação contextual direcionados, como no caso da integração com técnicas de RAG. Ao recuperar exemplos previamente rotulados da mesma categoria de vulnerabilidade, o sistema fornece ao modelo referências concretas que auxiliam na identificação de padrões semelhantes no contrato sob análise.

3.3. Construção do Contexto RAG

A construção do contexto na abordagem proposta segue o paradigma no qual informações externas são recuperadas e incorporadas ao prompt enviado ao modelo de linguagem, como proposto pela técnia RAG [Lewis et al. 2020]. O objetivo desse mecanismo é fornecer exemplos relevantes que auxiliem o modelo na identificação de padrões associados à vulnerabilidade analisada. Para isso, o fluxo de execução é organizado por meio de um grafo de processamento implementado com LangGraph, no qual cada etapa da pipeline é representada como um nó responsável por uma operação específica dentro do processo de análise.

Inicialmente, o SolRAG realiza a etapa de recuperação vetorial, na qual o código do contrato a ser analisado é utilizado como consulta em um banco de dados vetorial, uma base de dados especializada no armazenamento e recuperação eficiente de representações vetoriais de texto. Após isso, o mecanismo de busca baseado em similaridade semântica retorna os documentos mais relevantes armazenados no índice vetorial. No presente trabalho, são recuperados até três exemplos de código previamente indexados, cada um associado a metadados que indicam o tipo de vulnerabilidade presente no exemplo original. Esses documentos são tratados como referências de padrões vulneráveis que podem auxiliar o modelo na identificação de comportamentos semelhantes no contrato analisado.

Após a recuperação, os documentos selecionados são processados para a geração do contexto utilizado pelo modelo de linguagem. Cada documento recuperado é transformado em um bloco de texto contendo a identificação do exemplo, o tipo de vulnerabilidade associado e um trecho representativo do código fonte. Esses blocos são então concatenados para formar o contexto final, que será incorporado ao prompt enviado ao modelo. Caso nenhum documento relevante seja encontrado durante a etapa de recuperação, o sistema continua a análise sem exemplos auxiliares, indicando explicitamente a ausência de contexto recuperado.

Após a inferência do modelo, a resposta é parseada como JSON estruturado, com fallback por expressão regular em caso de falha, e a decisão binária é derivada programaticamente verificando se algum issue retornado corresponde à vulnerabilidade-alvo: caso positivo, o contrato é classificado como vulnerável; caso contrário, como não vulnerável.

Dessa forma, o modelo de linguagem recebe simultaneamente a especificação da vulnerabilidade, exemplos de padrões vulneráveis previamente observados e o código do contrato a ser analisado. Essa combinação permite que o modelo utilize os exemplos recuperados como referência para reconhecer padrões semelhantes, sem assumir automaticamente que o contrato analisado apresenta a mesma vulnerabilidade.

4. Configuração Experimental

4.1. Dataset

A construção do conjunto de dados utilizado neste trabalho foi baseada em benchmarks amplamente utilizados na literatura de segurança de contratos inteligentes. Como ponto de partida, utilizou-se o estudo de revisão sistemática conduzido por [Iuliano and Di Nucci 2026], que apresenta um levantamento atualizado de ferramentas, vulnerabilidades e benchmarks empregados na análise de segurança de contratos inteligentes. Esse trabalho identifica e organiza diversos conjuntos de dados utilizados em pesquisas acadêmicas, servindo como referência para a seleção de datasets com contratos previamente auditados e vulnerabilidades rotuladas.

A partir dos benchmarks listados nesse estudo, foi realizada uma análise dos datasets disponíveis. Entre os 72 benchmarks identificados na revisão, 46 puderam ser acessados diretamente, sendo que quatro deles continham conjuntos de contratos inteligentes com vulnerabilidades verificadas manualmente por auditores ou pesquisadores. Esses datasets foram priorizados neste trabalho por fornecerem rótulos de vulnerabilidade com maior confiabilidade, uma vez que os contratos foram auditados ou validados manualmente.

Os conjuntos de dados coletados apresentam diferentes origens e metodologias de construção. Em particular, três categorias principais foram identificadas: (i) contratos auditados por empresas especializadas em segurança de blockchain, (ii) contratos com vulnerabilidades injetadas artificialmente e posteriormente verificadas manualmente, e (iii) contratos coletados de repositórios públicos e posteriormente auditados por especialistas. Essa diversidade de fontes contribui para ampliar a variedade de padrões de vulnerabilidade presentes no dataset e reduzir o risco de viés associado a um único tipo de coleta de dados.

A partir da integração desses datasets, foi construída a base de conhecimento utilizada no mecanismo de recuperação semântica do SolRAG, contendo aproximadamente 1200 contratos inteligentes vulneráveis escritos em Solidity. As vulnerabilidades presentes nesses contratos abrangem diversas categorias amplamente discutidas na literatura, incluindo falhas de controle de acesso, *reentrancy*, *integer overflow/underflow*, *transaction order dependency* e *unchecked low-level calls*. Para adequar esse conjunto ao modelo de análise adotado, foi realizada uma etapa de expansão em que cada função vulnerável identificada dentro de um contrato foi extraída individualmente por meio de expressões regulares e tratada como um item independente, cada um contendo uma função específica

juntamente com o rótulo da vulnerabilidade associada, aumentando a granularidade da análise e reduzindo ruídos no prompt enviado ao modelo.

Os 52 contratos de avaliação de [David et al. 2023], alguns com múltiplas vulnerabilidades, foram expandidos para 70 instâncias, cada uma associada a uma única vulnerabilidade alvo e avaliados contra 61 categorias, totalizando $70 \times 61 = 4.270$ consultas.

4.2. Métricas de Avaliação

As métricas coletadas são verdadeiros positivos (TP), verdadeiros negativos (TN), falsos positivos (FP) e falsos negativos (FN), onde um TP corresponde à identificação correta de uma vulnerabilidade presente, TN à classificação correta de sua ausência, FP à detecção incorreta de uma vulnerabilidade inexistente e FN à falha em identificar uma vulnerabilidade existente.

4.3. Configuração dos Experimentos

Os experimentos foram conduzidos utilizando o modelo Gemma 3 na variante de 27 bilhões de parâmetros (gemma3:27b) com janela de contexto de 128000 tokens, executado localmente por meio da plataforma Ollama. A escolha desse modelo se deu com base em experimentos preliminares conduzidos com modelos alternativos, incluindo CodeLlama, DeepSeek e DeepSeek Coder, nos quais o Gemma 3 27B obteve os melhores resultados para a tarefa de análise de contratos inteligentes, desempenho consistente com sua capacidade de processamento de código evidenciada pelo relatório técnico do modelo [Gemma 2025]. Além disso, a escolha foi motivada pela capacidade de execução local sem dependência de APIs externas, pela janela de contexto de 128.000 tokens, que permite incluir o contexto gerado pelo RAG, e por ser um modelo de fácil acesso, o que favorece a reprodutibilidade dos experimentos.

O índice vetorial foi construído a partir dos trechos de código Solidity extraídos da base de conhecimento, representados por meio do modelo de embedding *all-MiniLM-L6-v2*. Os vetores gerados foram indexados em um banco vetorial ChromaDB persistido em disco, com cada documento contendo o código-fonte e metadados associados, como o tipo de vulnerabilidade e sua descrição.

A recuperação de exemplos no componente de RAG foi realizada por meio de um mecanismo de busca vetorial, no qual representações vetoriais dos trechos de código são utilizadas para identificar exemplos semanticamente semelhantes ao contrato analisado. Durante a etapa de recuperação, foram selecionados os três exemplos mais relevantes do índice vetorial para compor o contexto fornecido ao modelo de linguagem (top-k = 3). Esse valor foi definido com base em experimentos preliminares: com top-k = 1, o contexto reduzido resultou em 26 verdadeiros positivos, enquanto top-k = 5, ao gerar contexto excessivo no prompt além de tempos maiores de inferência, reduziu os verdadeiros positivos para 19.

A execução dos experimentos foi realizada em um ambiente local com sistema operacional Ubuntu 24.04 (64 bits), equipado com um processador Intel(R) Core(TM) i9-14900K e 128 GB de memória RAM. Para acelerar a inferência do modelo de linguagem, foi utilizada uma GPU NVIDIA A40, permitindo a execução eficiente do modelo de grande porte durante as etapas de análise dos contratos inteligentes.

5. Resultados

5.1. Desempenho Global

Para isolar a contribuição do mecanismo RAG dos demais fatores, como o modelo base utilizado, foi conduzido um experimento adicional executando o gemma3:27b sem recuperação de exemplos, mantendo a mesma estratégia de análise orientada por vulnerabilidade e o mesmo conjunto de avaliação. A Tabela 2 apresenta a comparação.

Tabela 2. Comparação de desempenho entre as configurações avaliadas.

Método	TP	FP	TN	FN	Total de Consultas
Gemma sem RAG	25	542	3.514	189	4.270
SolRAG	37	121	4.009	103	4.270
Δ (%)	+48,0%	−77,7%	+14,1%	−45,5%	–

A Tabela 2 apresenta a comparação entre o SolRAG e o Gemma 3 27B executado sem o mecanismo RAG, mantendo as demais configurações idênticas. O SolRAG obteve 37 verdadeiros positivos contra 25 do modelo sem RAG, redução de 77,7% nos falsos positivos (de 542 para 121) e aumento de 14,1% nos verdadeiros negativos (de 3.514 para 4.009), indicando que o mecanismo de recuperação de contexto contribui de forma independente para a melhoria da precisão da análise, além da capacidade do modelo base.

Tabela 3. Comparação de desempenho no dataset de [David et al. 2023].

	TP	FP	TN	FN	Total de Consultas
[David et al. 2023]	32	740	1128	41	1941
SolRAG	37	121	4009	103	4270
Δ (%)	+15.6%	-92.6%	+61.5%	+14.2%	–

A Tabela 3 apresenta a comparação entre os resultados obtidos pelo método proposto e aqueles reportados em [David et al. 2023] utilizando o mesmo conjunto de avaliação. Como o método proposto realiza um número maior de consultas, as métricas de FP, TN e FN são analisadas proporcionalmente ao total de consultas, enquanto o número de verdadeiros positivos é comparado diretamente, uma vez que cada contrato possui no máximo uma vulnerabilidade alvo.

Observa-se uma redução significativa na taxa de falsos positivos, que passa de 38,1% no trabalho de Gervais para 2,8% no método proposto, representando uma redução relativa de 92,6%. De forma semelhante, a proporção de verdadeiros negativos aumenta de 58,1% para 93,9%, correspondendo a um aumento relativo de 61,5%. Esses resultados indicam uma melhora substancial na capacidade do sistema de identificar corretamente contratos que não apresentam a vulnerabilidade analisada.

Em relação à detecção de vulnerabilidades reais, o número de verdadeiros positivos aumenta de 32 para 37, representando um ganho de 15,6%. Como cada contrato pode contribuir com no máximo um verdadeiro positivo, esse aumento corresponde diretamente ao número adicional de contratos vulneráveis corretamente identificados pelo método proposto.

Em relação aos falsos negativos, observa-se um aumento proporcional de 2,1% para 2,4%. Esse comportamento está associado ao uso de exemplos recuperados via similaridade semântica como contexto para a inferência, o que eleva o limite de decisão do modelo ao exigir uma correspondência maior entre os padrões observados no contrato analisado e as referências de vulnerabilidade. Como consequência, embora a taxa de falsos positivos seja significativamente reduzida, alguns casos de vulnerabilidade podem deixar de ser identificados.

5.2. Desempenho por Vulnerabilidade

A Tabela 4 apresenta os resultados obtidos pelo método proposto para cada categoria de vulnerabilidade presente no conjunto de avaliação. Para cada vulnerabilidade são reportados o número de contratos que possuem aquela falha, bem como os valores de verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN) observados durante os experimentos.

Tabela 4. Desempenho do método proposto por categoria de vulnerabilidade.

Vulnerabilidade	Contratos	TP	FP	TN	FN	Precisão	Recall	F1	Spec.	MCC
absence of code logic or sanity check	18	9	10	38	13	0,474	0,409	0,439	0,792	0,210
deployment mistake	2	0	4	63	3	0,000	0,000	0,000	0,940	-0,052
direct call to untrusted contract	1	1	10	58	1	0,091	0,500	0,154	0,853	0,162
fake tokens	1	0	1	68	1	0,000	0,000	0,000	0,986	-0,014
flash liquidity borrow, purchase, mint	5	2	2	61	5	0,500	0,286	0,364	0,968	0,328
governance attack	3	2	3	62	3	0,400	0,400	0,400	0,954	0,354
insider trade or other activities	1	0	4	65	1	0,000	0,000	0,000	0,942	-0,030
locked or frozen tokens	1	0	2	66	2	0,000	0,000	0,000	0,971	-0,029
on-chain oracle manipulation	8	6	1	58	5	0,857	0,545	0,667	0,983	0,641
other coding mistakes	1	1	41	28	0	0,024	1,000	0,047	0,406	0,098
other fake contracts	1	0	2	67	1	0,000	0,000	0,000	0,971	-0,021
other unfair or unsafe defi interaction	1	0	3	65	2	0,000	0,000	0,000	0,956	-0,036
other unsafe defi protocol dependency	6	3	1	61	5	0,750	0,375	0,500	0,984	0,492
token standard incompatibility	4	2	0	65	3	1,000	0,400	0,571	1,000	0,618
unfair liquidity providing	1	1	3	66	0	0,250	1,000	0,400	0,957	0,489
unfair slippage protection	2	1	0	67	2	1,000	0,333	0,500	1,000	0,569
unsafe call to phantom function	1	0	2	65	3	0,000	0,000	0,000	0,970	-0,036
visibility errors, incl. unrestricted action	5	3	4	60	3	0,429	0,500	0,462	0,938	0,408
access control	1	0	38	30	2	0,000	0,000	0,000	0,441	-0,187
arithmetic	1	1	38	30	1	0,026	0,500	0,049	0,441	-0,020
reentrancy	6	5	1	58	6	0,833	0,455	0,588	0,983	0,569
Média macro						**0,316**	**0,319**	**0,245**	**0,878**	**0,215**

A Tabela 4 reporta métricas calculadas por categoria: precisão, recall, F1-score, especificidade (*specificity*) e coeficiente de correlação de Matthews (MCC). As métricas foram computadas individualmente para cada categoria e agregadas por média macro, tratando cada par contrato-vulnerabilidade como uma instância de classificação binária independente. Essa escolha é necessária dado que o dataset possui forte assimetria entre positivos e negativos: cada contrato possui no máximo uma vulnerabilidade alvo, enquanto o número de consultas negativas é proporcional ao total de categorias avaliadas. Calcular métricas globais de forma agregada inflacionaria artificialmente a acurácia e a especificidade, comprometendo a comparabilidade com a literatura.

De modo geral, observa-se que o desempenho do sistema varia entre as diferentes categorias de vulnerabilidade, refletindo a diversidade estrutural das falhas presentes em contratos inteligentes. Categorias como *on-chain oracle manipulation*, *reentrancy* e *token*

standard incompatibility obtiveram boas métricas em F1-score e MCC, indicando poder discriminativo consistente, e apresentaram maior número de detecções corretas, indicando que o modelo conseguiu identificar padrões recorrentes associados a essas categorias. Por outro lado, algumas vulnerabilidades com baixa ocorrência no dataset apresentaram maior dificuldade de detecção, resultando em maior número de falsos negativos. Esse comportamento é esperado em cenários com poucos exemplos disponíveis, uma vez que o modelo possui menos referências semânticas para identificar padrões característicos dessas falhas.

Também é possível observar que determinadas categorias apresentam número elevado de falsos positivos, como no caso de *other coding mistakes*, *access control* e *arithmetic*. Esse comportamento pode estar relacionado ao fato de que essas categorias abrangem um espectro mais amplo de padrões de código, cujas construções sintáticas e semânticas frequentemente se assemelham às de contratos seguros, dificultando a distinção pelo modelo entre código vulnerável e código legítimo. Como direção para mitigar esse comportamento, a inclusão de exemplos negativos (como trechos de código seguro associados a cada categoria) na base de conhecimento do mecanismo RAG pode auxiliar o modelo a estabelecer uma fronteira mais precisa entre padrões vulneráveis e implementações legítimas, reduzindo a ocorrência de falsos positivos nessas categorias.

A especificidade macro de 0,878 confirma a eficácia da proposta em evitar alarmes falsos na maioria das categorias. No geral, a média macro de F1 igual a 0,245 e de MCC igual a 0,215 devem ser interpretadas tendo em vista a presença de categorias com poucos ou nenhum exemplo positivo corretamente detectado, que prejudicam fortemente as médias agregadas. Apesar dessas variações entre categorias, os resultados demonstram que o método proposto é capaz de identificar corretamente vulnerabilidades em diferentes contextos de contratos inteligentes. A análise por vulnerabilidade também permite observar os tipos de falha que são facilmente identificados pelo sistema, bem como aqueles que ainda apresentam desafios para detecção automática.

5.3. Análise de Erros

A análise dos erros observados durante os experimentos permite identificar padrões recorrentes nas decisões incorretas do sistema, fornecendo indícios sobre as limitações atuais do método e possíveis direções para melhorias futuras. Nesta análise, são considerados principalmente os casos de falsos positivos (FP) e falsos negativos (FN), que representam respectivamente detecções incorretas de vulnerabilidades e falhas na identificação de vulnerabilidades reais.

Uma parcela relevante dos falsos positivos está associada a categorias de vulnerabilidade mais amplas, como *other coding mistakes*, *access control*. Essas categorias frequentemente incluem padrões de código que podem aparecer tanto em implementações vulneráveis quanto em implementações legítimas, dificultando a distinção clara entre comportamento correto e incorreto. Como resultado, o modelo tende a interpretar determinadas construções como potenciais indicadores de vulnerabilidade, mesmo quando não há evidência suficiente para confirmar a falha.

Outro fator observado está relacionado à similaridade estrutural entre diferentes padrões de código. Em alguns casos, contratos que utilizam chamadas externas, manipulação de variáveis de estado ou operações matemáticas complexas podem apre-

sentar estruturas semelhantes às encontradas em exemplos vulneráveis. Quando esses padrões aparecem fora do contexto específico que caracteriza a vulnerabilidade, o modelo pode gerar classificações incorretas.

No caso dos falsos negativos, observa-se que muitas ocorrências estão associadas a vulnerabilidades com menor número de exemplos disponíveis na base de recuperação. Como o método proposto depende da recuperação de exemplos semanticamente semelhantes para construir o contexto de análise, categorias com poucos exemplos podem fornecer referências insuficientes para orientar corretamente o modelo. Esse comportamento foi observado especialmente em vulnerabilidades com baixa frequência no dataset, nas quais o número de instâncias positivas é limitado.

6. Discussão

Os resultados obtidos neste trabalho indicam que a integração de modelos de linguagem com mecanismos de recuperação de contexto pode contribuir significativamente para a análise automatizada de vulnerabilidades em contratos inteligentes. Em particular, a combinação de uma estratégia de análise orientada por vulnerabilidade com o uso de RAG mostrou-se eficaz na redução de falsos positivos (38,1% para 2,8%), um problema recorrente em ferramentas de detecção automática de falhas de segurança.

Uma diferença relevante em relação a abordagens anteriores está relacionada à evolução das capacidades dos modelos de linguagem utilizados. Neste trabalho, foi possível utilizar modelos com janelas de contexto ampliadas, chegando a até 128K tokens, permitindo incluir simultaneamente o código analisado, a descrição da vulnerabilidade e múltiplos exemplos recuperados pela base RAG. Essa ampliação possibilita fornecer ao modelo um contexto mais rico, potencialmente melhorando a capacidade de raciocínio sobre padrões de vulnerabilidade presentes no código.

Outro avanço em relação a trabalhos anteriores está na estruturação da saída do modelo. Em vez de depender exclusivamente de respostas textuais livres, foi adotado um formato estruturado de saída com decisão binária explícita sobre a presença ou ausência da vulnerabilidade analisada. Essa estratégia facilita o processamento automático dos resultados, reduz ambiguidades na interpretação das respostas e permite coletar métricas experimentais de forma mais confiável e reprodutível.

Embora o experimento com o Gemma 3 27B sem RAG permita isolar a contribuição do mecanismo de recuperação de contexto, o presente trabalho não avalia de forma independente os demais componentes da arquitetura proposta. Em particular, não foram conduzidos experimentos que isolem o impacto da estratégia de análise orientada por vulnerabilidade nem da adoção de saída binária estruturada, de modo que não é possível determinar com precisão em que medida cada um desses componentes contribui individualmente para os resultados observados.

Apesar desses avanços, observa-se que a base RAG não cobre completamente toda a diversidade de vulnerabilidades do conjunto de avaliação, de modo que categorias com poucos ou nenhum exemplo disponível podem receber referências insuficientes durante a recuperação, levando a classificações incorretas ou maior incidência de falsos negativos.

Uma limitação adicional diz respeito à possível sobreposição semântica entre a base de conhecimento utilizada pelo mecanismo RAG e o conjunto de avaliação. Embora

tenha sido verificada a ausência de contratos idênticos entre os dois conjuntos, não foi conduzida uma análise quantitativa da similaridade semântica entre eles, de modo que contratos estruturalmente semelhantes aos exemplos da base RAG podem ser recuperados com maior facilidade, o que poderia inflar os resultados observados, sendo a avaliação desse impacto uma direção relevante para trabalhos futuros.

7. Conclusão e Trabalhos Futuros

Este trabalho apresentou o SolRAG, uma abordagem baseada em RAG que combina LLMs com recuperação semântica e análise orientada por vulnerabilidade, para detecção de vulnerabilidades em contratos inteligentes. Os experimentos demonstraram redução significativa na taxa de falsos positivos (38,1% para 2,8%) e aumento nos verdadeiros negativos (58,1% para 93,9%) em comparação com [David et al. 2023], com o mecanismo RAG contribuindo de forma independente para esses ganhos além da capacidade do modelo base.

Entre as principais limitações, destaca-se a ausência de um estudo de ablação completo que isole a contribuição individual da estratégia de análise orientada por vulnerabilidade e da saída estruturada. A inclusão de exemplos negativos na base de conhecimento constitui uma direção promissora para reduzir falsos positivos em vulnerabilidades com padrões de código mais amplos. Como trabalhos futuros, a inclusão de comparações com ferramentas tradicionais de análise estática como *Slither* e *Mythril* permite comparar abordagens baseadas em LLM em relação ao estado da arte. Além disso, a avaliação de múltiplos modelos de linguagem é necessária para avaliar a generalidade da abordagem. A substituição do gemma3:27b por modelos como DeepSeek, Qwen e GPT permite analisar se os ganhos observados dependem de características específicas do modelo utilizado ou se são reprodutíveis em diferentes arquiteturas. Por fim, análises granulares do código, capazes de identificar não apenas a presença de uma vulnerabilidade, mas também sua localização exata no contrato, constituem uma direção relevante para tornar a abordagem mais precisa e interpretável para desenvolvedores e auditores.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001. Fapemig e CNPq também apoiaram a execução deste trabalho.

Referências

Bani-Hani, R. M., Shatnawi, A. S., and Al-Yahya, L. (2024). Vulnerability detection and classification of ethereum smart contracts using deep learning. *Future Internet*, 16(9):321.

Boi, B., Esposito, C., and Lee, S. (2024). Smart contract vulnerability detection: The role of large language model (llm). *ACM SIGAPP applied computing review*, 24(2):19–29.

David, I., Zhou, L., Qin, K., Song, D., Cavallaro, L., and Gervais, A. (2023). Do you still need a manual smart contract audit? *arXiv preprint arXiv:2306.12338*.

Gemma (2025). Gemma 3 technical report. Technical report, Google DeepMind.

He, Z., Li, Z., Yang, S., Ye, H., Qiao, A., Zhang, X., Luo, X., and Chen, T. (2024). Large language models for blockchain security: A systematic literature review. *arXiv preprint arXiv:2403.14280*.

Hewa, T., Ylianttila, M., and Liyanage, M. (2021). Survey on blockchain based smart contracts: Applications, opportunities and challenges. *Journal of network and computer applications*, 177:102857.

Hu, S., Huang, T., İlhan, F., Tekin, S. F., and Liu, L. (2023). Large language model-powered smart contract vulnerability detection: New perspectives. In *2023 5th IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 297–306. IEEE.

Iuliano, G. and Di Nucci, D. (2026). Smart contract vulnerabilities, tools, and benchmarks: An updated systematic literature review. *Journal of Systems and Software*, page 112788.

Khan, S. N., Loukil, F., Ghedira-Guegan, C., Benkhelifa, E., and Bani-Hani, A. (2021). Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-peer Networking and Applications*, 14(5):2901–2925.

Kushwaha, S. S., Joshi, S., Singh, D., Kaur, M., and Lee, H.-N. (2022). Ethereum smart contract analysis tools: A systematic review. *Ieee Access*, 10:57037–57062.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Nam, D., Macvean, A., Hellendoorn, V., Vasilescu, B., and Myers, B. (2024). Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13.

Nhu, N. D. Q., Van, T. H., Trung, D. M., Duy, P. T., et al. (2025). Rag-smartvuln: Enhancing smart contract vulnerability detection via retrieval-augmented llms. In *2025 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, pages 1–6. IEEE.

Oliva, G. A., Hassan, A. E., and Jiang, Z. M. (2020). An exploratory study of smart contracts in the ethereum blockchain platform. *Empirical Software Engineering*, 25(3):1864–1904.

Qian, P., Liu, Z., He, Q., Huang, B., Tian, D., and Wang, X. (2022). Smart contract vulnerability detection technique: A survey. *arXiv preprint arXiv:2209.05872*.

Xiao, Z., Wang, Q., Pearce, H., and Chen, S. (2025). Logic meets magic: Llms cracking smart contract vulnerabilities. In *2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–3. IEEE.

Yu, J. (2024). Retrieval augmented generation integrated large language models in smart contract vulnerability detection. *arXiv preprint arXiv:2407.14838*.