Detecção de Oportunidades de Sanduíche MEV via Grafo de Fluxo de Controle de Contratos Inteligentes

Josué N. Campos[1], Aline C. S. Silva[1], Cleidimar L. dos Passos[1], Glauber D. Gonçalves[2], Alex B. Vieira[3] e José Augusto M. Nacif[1]

[1] Universidade Federal de Viçosa (UFV) – Florestal, MG – Brasil
[2] Universidade Federal do Piauí (UFPI) – Teresina, PI – Brasil
[3]Universidade Federal de Juiz de Fora (UFJF) – Juiz de Fora, MG – Brasil

{josue.campos,aline.c.silva,cleidimar.passos,jnacif}@ufv.br

alex.borges@ufjf.edu.br,ggoncalves@ufpi.edu.br

Resumo. *As Finanças Descentralizadas (DeFi) têm transformado o ecossistema financeiro ao permitir negociações sem intermediários, mas também introduz novos vetores de ataque. Estratégias do tipo Sanduíche MEV exploram configurações inseguras de derrapagem de preço em trocas descentralizadas, gerando perdas financeiras significativas aos usuários. Enquanto as mitigações atuais focam predominantemente na camada de rede, há uma lacuna em ferramentas preventivas de auditoria de código. Este trabalho propõe uma abordagem de análise estática baseada em Gráfico de Fluxo de Controle para detectar a ausência de proteções de limite de preço em contratos inteligentes. Nossa metodologia rastreia o código de contratos inteligentes Solidity da rede Ethereum, visando identificar parâmetros inseguros ou não validados pelo usuário. Resultados preliminares demonstram que o detector alcançou uma acurácia de 78,9% em protocolos DeFi reais, contribuindo para a segurança preventiva em redes Blockchain.*

1. Introdução

A ascensão das Finanças Descentralizadas (DeFi) transformou o paradigma de transações financeiras globais, permitindo a troca de ativos sem intermediários por meio de *Automated Market Makers* (AMMs), como Uniswap e PancakeSwap. Contudo, a transparência inerente à fila de transações pendentes (*mempool*) e a mecânica de ordenação de transações deram origem ao fenômeno da Máxima Extração de Valor em Blocos (MEV) [Ferreira Torres et al.]. O MEV refere-se ao lucro que construtores ou validadores de blocos podem extrair ao reordenar, incluir ou impedir transações dentro de um bloco. Dentre as estratégias de extração de MEV, o Sanduíche MEV destaca-se como uma das estratégias proeminentes na rede blockchain Ethereum, na qual um agente (denominado de Pesquisador MEV) observa uma transação pendente de troca de ativos e insere suas próprias transações imediatamente antes e depois dela, manipulando o preço do ativo para obter lucro às custas do usuário original [Zhou et al. 2021].

Embora o MEV seja frequentemente discutido sob a perspectiva da camada de consenso e infraestrutura de rede, a raiz da vulnerabilidade ao Sanduíche MEV reside, predominantemente, na camada de aplicação: o código do contrato inteligente. Especificamente, o sanduíche pode ocorrer quando desenvolvedores negligenciam a definição

correta de parâmetros de proteção contra derrapagem de preço (*slippage*), como o argumento quantidade mínima em roteadores de AMMs [Weintraub et al. 2022]. Ferramentas de análise estática convencionais muitas vezes falham em detectar essa vulnerabilidade lógica, pois focam em padrões sintáticos simples ou erros de reentrada, ignorando o fluxo de dados que determina se o parâmetro de proteção é, de fato, seguro ou se foi definido arbitrariamente, expondo o contrato à manipulação de preço pela dependência da ordem de transações [Chaliasos et al. 2024].

Nesse sentido, a persistência dessa atividade MEV resulta em perdas financeiras significativas e compromete a integridade do ecossistema DeFi. Segundo o EigenPhi[1], uma das principais plataformas de monitoramento de atividades MEV na rede Ethereum, somente em 30 dias os Pesquisadores MEV atingem lucros de em média USD 64.000 com o Sanduíche. Trabalhos recentes, como DeFort [Xie et al. 2024] e DeFiTainter [Kong et al.], propõem soluções robustas baseadas em execução simbólica complexa ou geração automática de simulações. No entanto, tais abordagens atuam de forma reativa ou em estágios avançados de auditoria. Existe, portanto, uma lacuna para ferramentas preventivas e integráveis ao ciclo de desenvolvimento do contrato inteligente, capazes de identificar se um contrato é suscetível ao Sanduíche MEV através da análise estrutural do código antes da implantação na rede blockchain.

O presente trabalho propõe uma abordagem de análise estática para a detecção automática de suscetibilidade a Sanduíche MEV em contratos inteligentes Solidity. O objetivo principal é desenvolver um detector que utiliza o Grafo de Fluxo de Controle (CFG) do contrato para rastrear a origem dos parâmetros de derrapagem de preço em chamadas de troca de ativos (*swaps*). A metodologia diferencia-se por verificar não apenas o valor do argumento, mas seu fluxo de dados, determinando se a proteção de preço é configurável pelo usuário ou estática/inexistente, reduzindo a taxa de falsos negativos comuns em auditores genéricos e a falta de alertas de suscetibilidade à atividades MEV durante a auditoria automática.

Para validar a abordagem proposta, foi empregado o uso da principal ferramenta de auditoria automática da literatura Slither [Feist et al. 2019] para gerar e percorrer o CFG, pois com ela é possível criar detectores customizados e integrá-la ao fluxo de desenvolvimento do contrato. O detector foi avaliado contra um dataset curado de 19 contratos inteligentes coletados da rede principal da blockchain Ethereum, contendo tanto instâncias comprovadamente independentes quanto dependentes da derrapagem de preço. Os resultados demonstram que o detector alcançou uma acurácia de 78,9%, superando analisadores estáticos que negligenciam a identificação de oportunidades de Sanduíche MEV. Além disso, o tempo médio de análise por contrato foi de 4,88 segundos, evidenciando a viabilidade da técnica para uso contínuo no ciclo de desenvolvimento do contrato.

O restante do artigo está organizado da seguinte maneira: a Seção 2 aborda os principais conceitos relacionados ao trabalho. A Seção 3 elenca os trabalhos relacionados. Na Seção 4 são apresentadas as metodologias para geração dos resultados, discutidos na Seção 5. Por fim, na Seção 6 são discutidas algumas limitações da abordagem proposta e na Seção 7 são apresentadas as conclusões e trabalhos futuros.

[1] `https://eigenphi.io/`

2. Fundamentação Teórica

Nesta seção são apresentados os principais conceitos relacionados ao trabalho: Contratos Inteligentes, Sanduíche MEV e Grafo de Fluxo de Controle.

2.1. Contratos Inteligentes

Contratos Inteligentes são códigos escritos em uma linguagem de programação Turing-completa que permitem automatizar protocolos contratuais de maneira digital [Chu et al. 2023]. Por meio desses códigos, interações financeiras como troca, empréstimo, compra e venda de ativos digitais podem ser realizadas em redes Blockchain. Os contratos inteligentes, uma vez implantados na rede Blockchain, tornam-se imutáveis *i.e.*, não podem ter o código-fonte alterado e qualquer funcionalidade é executada automaticamente desde que as condições necessárias sejam atingidas [Campos and et al. 2024].

Apesar desta natureza imutável introduzida pelas redes Blockchain, padrões de desenvolvimento de contratos inteligentes como *Proxy* permitem que os contratos sejam atualizados à nível da aplicação que o utiliza, *e.g.* uma nova versão do código do contrato pode ser implantada na rede e apenas o endereço ser atualizado no *proxy* que encapsula a sua utilização [Bodell III et al. 2023]. Nesse sentido, os contratos inteligentes possuem um ciclo de vida que, em etapas como desenvolvimento, execução e atualização, podem ser introduzidas vulnerabilidades de segurança que comprometem a validade do acordo codificado.

2.2. Sanduíche MEV

O Sanduíche MEV é uma estratégia de maximização de lucro que ocorre em redes Blockchain [Zhou et al. 2021]. Diferentemente de estratégias convencionais, o Sanduíche caracteriza-se por necessitar de uma ou mais transações-alvo de uma ou mais vítimas para obtenção de lucro.

A estratégia de Sanduíche MEV baseia-se na seguinte metodologia: A pessoa que realiza o sanduíche, denominada de Pesquisador MEV, identifica uma ou mais transações-alvo em potencial que realizam a operação de troca de ativos digitais. Geralmente, essas transações movimentam uma grande quantidade de um ativo e passam pela fila de transações pendentes da rede Blockchain. O Pesquisador MEV encapsula a transação da vítima em outras transações de maneira que, as transações anteriores à da vítima realizam a mesma troca do ativo digital identificado na transação-alvo e as transações posteriores realizam a operação de venda desses ativos trocados [Fontinele et al. 2024]. Com base nessa configuração, da ordem das transações no bloco e da taxa de derrapagem definida pela vítima *i.e.*, a porcentagem do valor do ativo que a vítima permite que ele varie para a sua transação ser efetivada, o Pesquisador MEV pode obter lucros consideráveis ao manipular o preço do ativo digital [Campos et al. 2025b].

2.3. Grafo de Fluxo de Controle

O Grafo de Fluxo de Controle (CFG) é uma estrutura de dados que representa todos os caminhos possíveis de execução de um programa de computador [Contro et al. 2021]. Cada nó representa um bloco básico de código, representados por sequências de instruções. As arestas representam operações de fluxo de controle, como declarações de *if/else*, laços ou *jumps* que direcionam para outros blocos básicos.

A estrutura de um CFG permite encontrar caminhos de execuções vulneráveis e mapear padrões de vulnerabilidades de segurança. Em relação aos contratos inteligentes escritos na linguagem de programação Solidity, uma das principais ferramentas da literatura é a Slither [Feist et al. 2019]. A Slither é uma ferramenta de análise estática de contratos que emprega o uso de CFGs para encontrar padrões de vulnerabilidades embutidos na ferramenta, bem como permite a criação de detectores customizados para mapear diferentes padrões de execução do código [Campos et al. 2025a].

3. Trabalhos Relacionados

Ao longo dos últimos anos diversas ferramentas de detecção automática de vulnerabilidades foram propostas na literatura, conforme estudado em [Ivanov et al. 2023].

Os autores em [Kong et al.] propõem a ferramenta DeFiTainter com o objetivo de encontrar vulnerabilidades genéricas de manipulação de preço. A ferramenta identifica automaticamente variáveis que afetam o saldo de ativos *e.g* parâmetros de transferências ou atualizações de saldos. Ao marcar as variáveis de entrada que podem ser controladas pelo usuário, a ferramenta verifica se uma dessas variáveis consegue influenciar um componente crítico do código sem passar por uma verificação de segurança. Ao final, a ferramenta transforma o código em equações matemáticas para verificar se é logicamente possível que o atacante controle o preço.

Já em [Xie et al. 2024] os autores apresentam a ferramenta DeFort, capaz de detectar automaticamente variáveis que agem como preço dentro do contrato inteligente e realizar ataques de testes para atestar manipulações de preço. A ferramenta filtra funções suspeitas que interagem com protocolos de finanças descentralizadas, detecta automaticamente quais variáveis agem como preço no contrato e tenta executar o ataque utilizando técnicas de Fuzzing com tentativa e erro inteligente guiado pelo objetivo de maximizar o lucro. Toda transação é executada em uma cópia da blockchain para provar que o dinheiro é roubado de verdade.

Em relação ao Sanduíche MEV, os trabalhos da literatura focam na detecção *on-chain i.e.*, detectando a atividade MEV com base nas transações que ocorrem na rede Blockchain em tempo-real. Trabalhos como em [Wu et al. 2025, Kong 2025] utilizam os recursos de Redes Neurais para detectar padrões nas transações que ocorrem nos blocos da blockchain, combinando heurísticas que definem como o Sanduíche MEV acontece e a ordem das transações. Apesar de serem abordagens com identificação em tempo-real da atividade MEV, os autores não levam em consideração como o Sanduíche pode ser evitado em tempo de desenvolvimento dos contratos inteligentes.

Enquanto ferramentas como DeFiTainter e DeFort focam na geração complexa de testes de vulnerabilidades, nossa abordagem oferece retorno imediato sobre a possibilidade do contrato inteligente estar vulnerável a um Sanduíche MEV, permitindo a correção antes mesmo do contrato ser implantado na rede de testes.

4. Metodologia

A metodologia proposta baseia-se na análise estática do código-fonte de contratos inteligentes para identificar padrões estruturais suscetíveis ao Sanduíche MEV. A abordagem difere das técnicas tradicionais de correspondência de padrões (expressões regulares ou nomes de variáveis conhecidos) ao focar na semântica operacional das instruções.

4.1. Conjunto de Dados

O conjunto de dados utilizado neste trabalho é composto por 19 contratos inteligentes implantados na rede Blockchain principal da Ethereum, identificados como alvos da estratégia do tipo Sanduíche MEV. A seleção das amostras fundamentou-se em relatórios de exploração gerados pela plataforma EigenPhi[2], uma ferramenta especializada em detecção de atividades MEV.

Tabela 1. Relação de Protocolos/Corretoras utilizadas pelas vítimas coletadas de acordo com o EigenPhi e Etherscan.

Protocolo/DEX	Contrato	Qtde. Transações Realizadas
Uniswap	UniswapV2Router02	289.194.515
Uniswap	UniversalRouter	50.245.383
1inch	AggregationRouterV5	47.819.772
Uniswap	UniswapV3Pool	21.473.188
Li.Fi	ChainflipFacet	17.436.281
1inch	AggregationRouterV6	15.682.288
Uniswap	SwapRouter02	10.826.966
Uniswap	NonfungiblePositionManager	10.826.865
0x Protocol	AllowanceHolder	9.689.065
OKX	DexRouterV105	5.965.986
Binance	Diamond	5.861.156
KyberSwap	MetaAggregationRouterV2	5.243.896
OKX	DexRouterV107	3.842.084
Transit	TransitSwapRouterV5	3.268.626
OKX	DexRouterV106	2.813.270
-	FastMCTP	59.254
InstaDapp	InstaDefault	41.271
-	StakingPool	1.733
Safe	GnosisSafe	1.558

Conforme a Tabela 1, foram extraídos os 10 contratos com maior volume de exploração registrados no mês de dezembro de 2025, seguidos por outros 9 contratos que sofreram exploração similares ao longo de outros meses do mesmo ano. Para a obtenção do código-fonte, utilizou-se a ferramenta *Blockscan IDE*, integrada ao explorador de blocos Etherscan.io[3]. Através deste método, foi possível extrair a estrutura completa de arquivos de cada alvo, incluindo o contrato principal, interfaces, bibliotecas e dependências internas da linguagem Solidity. Cada contrato analisado está associado a um projeto de protocolo ou corretora que foi utilizado pela vítima. Houve 2 registros de contratos que não estão associados a uma corretora ou protocolo, indicando a possibilidade de serem contratos criados e/ou utilizados pela própria vítima.

Cada entrada do conjunto de dados corresponde a uma Corretora Descentralizada (DEX) ou protocolo que sofreu exploração via Sanduíche MEV. Como muitos desses são projetos complexos (roteadores, agregadores, piscina de ativos, etc.), o código-fonte

[2]`https://eigenphi.io/`
[3]`https://etherscan.io/`

completo foi coletado, resultando em pastas com múltiplos arquivos *.sol* (em certos casos, ultrapassando 50 arquivos, incluindo dependências, interfaces e implementações). Os dados coletados foram organizados em diretórios individuais, seguindo a convenção de nomenclatura baseada no nome da pasta raiz do contrato (referindo ao nome do projeto/protocolo) e o seu respectivo endereço hexadecimal na rede Ethereum. Em alguns projetos, os protocolos utilizam o paradigma de Proxy para encapsular a execução do contrato, dessa maneira filtramos nos códigos coletados apenas os contratos inteligentes que continham a lógica de execução do protocolo, descartando os contratos de Proxy que apenas delegavam as chamadas de funções. A estrutura resultante preserva a hierarquia original dos arquivos (*.sol*), mas com remapeamento de importações de bibliotecas e padronização de versão do compilador Solidity, garantindo que as lógicas de execução de cada contrato estejam disponíveis para análise.

4.2. Detecção de Sanduíche MEV

O algoritmo desenvolvido (Algoritmo 1) opera sobre a Representação Intermediária (SlithIR) da ferramenta Slither [Feist et al. 2019]. O princípio fundamental da detecção é a identificação de um padrão para o Pesquisador MEV marcar o contrato como vítima, caracterizado pela convergência de três componentes: uma fonte ativa que altera o estado do contrato ou consulta um saldo pós-execução, entrada do usuário com parâmetros controlados externamente e um predicado de desigualdade que valida a fonte ativa contra a entrada do usuário. Este padrão caracteriza que a vítima realiza uma operação de troca de ativos no contrato com um indicativo que a taxa de derrapagem (*slippage*) vem de uma fonte externa que pode estar desprotegida, abrindo oportunidade para a atividade de Sanduíche MEV acontecer. Nesse sentido, a implementação do detector foi separada em três etapas distintas de análise: Mapeamento de Fontes, Filtragem de Operações e Análise de Dependência de Dados.

$$S_{active} = \left\{ i \in \mathcal{I} \;\middle|\; \begin{array}{l} (Opc(i) \in \{\texttt{CALL}, \texttt{DELEGATECALL}\} \wedge \neg Static(i)) \\ \vee\, Opc(i) = \texttt{BALANCE} \end{array} \right\} \quad (1)$$

A primeira etapa (Equação 1) consiste em iterar sobre o Grafo de Fluxo de Controle (CFG) para identificar instruções ($\mathcal{I}$) que representam o resultado de uma troca de ativos (*swap*). Para reduzir a taxa de falsos positivos, aplicamos uma filtragem baseada no tipo de *Opcode* da instrução (i). São consideradas fontes válidas as instruções `CALL` e `DELEGATECALL` (seja em alto nível ou em Assembly), pois representam a execução efetiva de uma transação externa e, portanto, geram mudança de estado. Instruções estáticas são explicitamente descartadas ($\neg IsStatic(i)$). Esta decisão metodológica baseia-se na premissa de que sanduíches exploram a volatilidade imediata causada por uma transação de escrita, enquanto chamadas estáticas (leitura) são, por definição, imunes a alterações de estado na mesma transação.

Além disso, devido à natureza não tipada das variáveis em blocos Assembly do Solidity, o algoritmo adota a heurística de que variáveis escritas por um nó Assembly contendo Opcodes de chamada (*call*) são tratadas como inteiros sem sinal implícitos e marcadas como fontes de risco, garantindo a detecção em implementações otimizadas de protocolos. Adicionalmente, o algoritmo reconhece o padrão de design que o contrato

Algoritmo 1 Detector de Oportunidade de Sanduíche MEV.

Require: $\mathcal{C}$ {Um Contrato Inteligente Solidity.}
$\mathcal{X} \leftarrow \emptyset$ {Conjunto de vulnerabilidades encontradas.}
for $f \in C$ **do**
  $S \leftarrow \emptyset$
  for $n \in f$ **do**
    if Tipo(n) = `ASSEMBLY` **then**
      if `CALL` $\in$ Instrucoes(n) **and** `STATICCALL` $\notin$ Instrucoes(n) **then**
        $S \leftarrow S \cup$ Variaveis(n)
      end if
    else
      for Instrucao $i \in n$ **do**
        if (i = `CALL` **and** $i \neq$ `STATICCALL`) **or** i = `BALANCE` **then**
          if Tipo(LValue(i)) = `INT` **then**
            $S \leftarrow S \cup \{$LValue(i)$\}$
          end if
        end if
      end for
    end if
  end for
  for $n \in f$ **do**
    if Tipo(n) == `IF` **and** Inequacao(n) **then**
      $V \leftarrow$ Variaveis(n)
      if $V \in$ Params(f) **and** $V \in S$ **then**
        $\mathcal{X} \leftarrow \mathcal{X} \cup \{(f, n,$ "Oportunidade encontrada"$)\}$
      end if
    end if
  end for
end for
return $\mathcal{X}$

não verifica o retorno da função, mas sim a diferença de saldo do ativo. Instruções de verificação de saldo dominadas por chamadas de execução também são tratadas como fontes ativas ($Opc(i) =$ `BALANCE`).

$$\mathcal{N} = \{n \in \texttt{CFG}_{nodes} \mid Type(n) = \texttt{IF} \wedge Op(n) \in \{<, >, \leq, \geq\}\} \tag{2}$$

Na segunda etapa (Equação 2), diferentemente de detectores genéricos que sinalizam qualquer uso de variáveis contaminadas, nós restringimos a detecção em predicados de desigualdade. O algoritmo analisa as instruções condicionais e examina as operações binárias associadas. Apenas os operadores de desigualdade ($<$, $>$, $\leq$, $\geq$) são considerados válidos. Tal propriedade leva em consideração que verificações de derrapagem de preço são inerentemente verificações de "valor mínimo aceitável" e esta restrição remove possíveis falsos positivos comuns gerados por verificações de sucesso booleano (*require(success)*) ou verificações de integridade de dados (*hash* == *expected*), que utilizam

operadores de igualdade.

$$\begin{aligned} \Phi_p &= \{p \in \mathcal{V} \mid p \in Params(f_{public})\} \\ \Phi_f &= \{v \in \mathcal{V} \mid \exists i \in S, v \in Def(i)\} \end{aligned} \tag{3}$$

A etapa final correlaciona as fontes ativas com os predicados através da análise de dependência de dados. O algoritmo verifica a intersecção de dois conjuntos de entradas (Equação 3): variáveis derivadas direta ou indiretamente dos parâmetros da função pública (Φ_p) e variáveis definidas nas fontes ativas identificadas na primeira etapa (Φ_f), como variáveis de instruções de chamadas externas ou atualizações de saldo ($Def(i)$). A propagação de dados através do grafo é verdadeira se uma variável é dependente por qualquer elemento do conjunto Φ (Equação 4). Uma vulnerabilidade é confirmada se, e somente se, uma instrução condicional de desigualdade depender simultaneamente destes dois conjuntos de entradas (Equação 5). Para garantir a robustez contra diferentes padrões de design de contratos, o algoritmo considera retornos de chamadas internas e de bibliotecas como fontes potenciais, permitindo a detecção de vulnerabilidades ocultas em funções auxiliares e importações.

$$\Delta(v, \Phi) \iff \exists \phi \in \Phi \mid \phi \hookrightarrow v \tag{4}$$

$$IsVulnerable(n) \iff n \in \mathcal{N} \wedge (\exists v_p \in \Delta(v, \Phi_p)) \wedge (\exists v_f \in \Delta(v, \Phi_f)) \tag{5}$$

5. Resultados

A Figura 1 exibe um fragmento crítico do Grafo de Fluxo de Controle (CFG) gerado pela análise estática do Slither com o detector aplicado, ilustrando explicitamente o mecanismo de validação da derrapagem de preço na função analisada. O nó de decisão condicional (IF 26) avalia o predicado de desigualdade `returnAmount < minReturn`, criando uma bifurcação no fluxo de execução onde o ramo "Verdadeiro" culmina no nó de exceção (EXPRESSION 27), o qual invoca a instrução *revert*. No contexto da metodologia de detecção desenvolvida, este segmento representa a cláusula que compara o valor de saída de uma operação externa (a variável *returnAmount*, identificada pela execução anterior de uma troca) contra um parâmetro controlado pelo usuário (*minReturn*), confirmando que a segurança da transação depende inteiramente da precisão desse parâmetro de entrada fornecido externamente.

Para calcular as métricas de classificação do funcionamento do detector nos contratos analisados, utilizamos a Classificação por Arquivo, *i.e.* consideramos como Verdadeiro Positivo (TP) quando um contrato inteligente possui a oportunidade de Sanduíche MEV e o detector a encontra. Os Verdadeiros Negativos (TN) são calculados como sendo os contratos que não possuem a oportunidade de Sanduíche MEV explícita e o detector não a encontra. Já os Falsos Negativos (FN) são contabilizados quando o detector falha em reportar a oportunidade no contrato analisado e os Falsos Positivos (FP) são contratos seguros que foram incorretamente sinalizados.

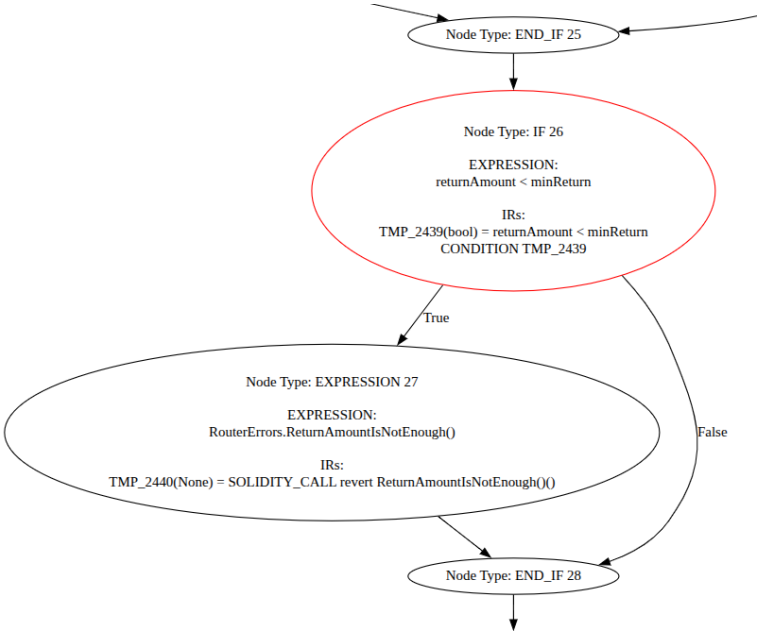

Figura 1. Recorte do CFG gerado pelo Slither com o nó de indicativo de oportunidade de Sanduíche MEV.

Nesse sentido, a avaliação experimental do detector proposto demonstrou eficácia na identificação de padrões de oportunidades de Sanduíche MEV em contratos de infraestrutura de roteamento, conforme a Tabela 2. O detector identificou corretamente os padrões em todos os contratos de roteamento analisados ($TP = 9$), incluindo o AggregationRouterV5, AggregationRouterV6 e múltiplas versões do DexRouter (V105, V106 e V107). Esta consistência na detecção através de diferentes implementações e versões sugere que a abordagem de analisar o fluxo de dados pelo CFG é robusta, sendo capaz de capturar a semântica fundamental da verificação de derrapagem de preço, independentemente de variações sintáticas no código-fonte. Mesmo em contratos com lógicas complexas, como o FastMCTP e MetaAggregationRouterV2, o detector foi capaz de rastrear o fluxo de dados através de múltiplas funções. No UniversalRouter e no UniswapV3Pool, o detector identificou com sucesso a oportunidade de Sanduíche MEV, mesmo sendo contratos que utilizam padrões de verificação de derrapagem de preço pós-execução explícita, típicos de agregadores e protocolos V3. A capacidade de sinalizar a oportunidade de sanduíche em diferentes casos, desde roteadores diretos até agregadores, reforça que o detector cobre as implementações de troca de ativos presentes na amostra, minimizando o risco de não detecção em cenários reais de Sanduíche MEV.

Tabela 2. Métricas para o detector de oportunidades de Sanduíche MEV.

Métrica	**Fórmula**	**Valor**
Acurácia	$(TP + TN)/N$	78,95%
Recall	$TP/(TP + FN)$	75,00%
Taxa de Falsos Negativos	$FN/(TP + FN)$	25,00%
Taxa de Falsos Positivos	$FP/(TN + FP)$	14,29%

Adicionalmente, o detector soube interpretar resultados negativos consistente-

mente ($TN = 6$) para contratos utilitários e de governança que, por definição, não executam trocas de ativos, tais como AllowanceHolder, Diamond, ChainflipFacet, GnosisSafe, InstaDefault e StakingPool. Esta discriminação evidencia a robustez da estratégia baseada em fluxo de dados para isolar lógicas suscetíveis ao Sanduíche MEV. Apesar disso, estes contratos podem conter lógicas que ultrapassam a análise do fluxo de dados definida no detector, pois são contratos utilizados por vítimas de Sanduíche MEV. Uma justificativa para isso, é que os Pesquisadores MEV podem ter utilizado estratégias que vão além de analisar somente a derrapagem de preço dos ativos digitais para realizar a atividade.

Sobre os falsos negativos ($FN = 3$), foi possível observar casos nos contratos UniswapV2Router02, SwapRouter02 e TransitSwapRouterV5. Apesar dos contratos terem sido utilizados por vítimas de Sanduíche MEV, este resultado sugere uma limitação do detector na forma de tratar fontes estáticas. Por exemplo, o Uniswap V2 utilizado nos contratos UniswapV2Router02 e TransitSwapRouterV5 frequentemente valida a derrapagem de preço comparando a entrada do usuário contra valores calculados previamente via bibliotecas, em vez de verificar diretamente o retorno da chamada de execução. Como o detector ignora fontes estáticas e funções puras, isso acaba negligenciando padrões onde a segurança é garantida por pré-cálculo matemático determinístico em vez de validação dinâmica pós-troca de ativo.

No entanto, o detector também gerou falsos positivos semânticos ($FP = 1$) na função *sweepToken* do contrato NonfungiblePositionManager. Mesmo que esta função realize comparações de desigualdade sobre saldos, o que caracteriza uma fonte ativa do detector, o objetivo semântico é a limpeza de ativos, e não uma troca financeira. Isso demonstra que, embora a detecção estrutural seja precisa em identificar o padrão de código, ela carece de contexto sobre a regra de negócio do contrato para distinguir entre uma operação com possibilidade de ser vulnerável e uma manutenção de saldo. Além disso, assim como nos verdadeiros negativos, este contrato pode ter sido utilizado por uma vítima que sofreu o sanduíche a partir de outras estratégias do Pesquisador MEV que ultrapassam a verificação de derrapagem de preço.

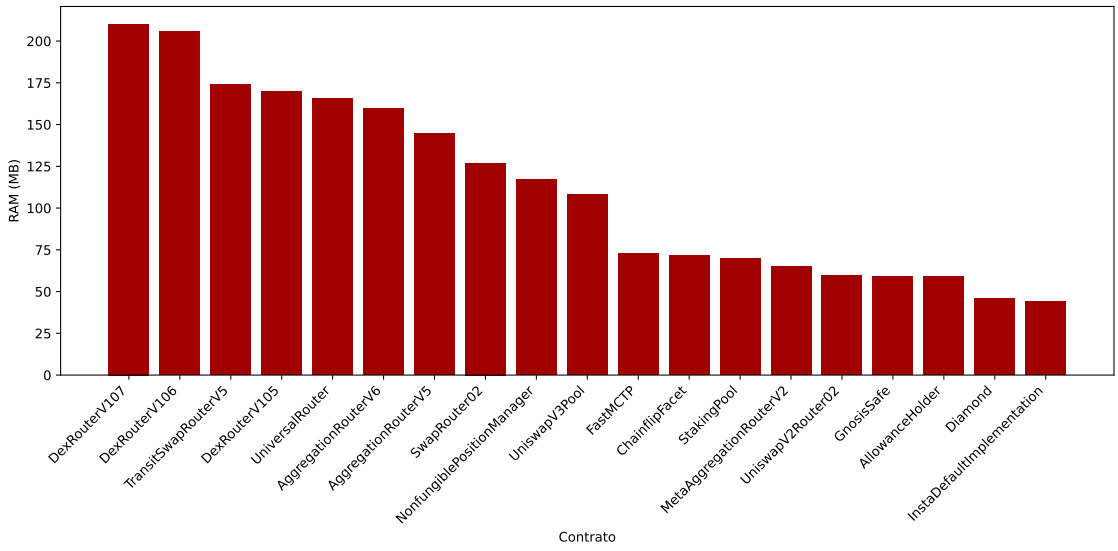

Figura 2. Consumo de memória RAM gasto pela ferramenta por contrato.

Em relação ao consumo de memória RAM, de acordo com a Figura 2 a ferramenta aloca uma quantidade similar de memória de acordo com a complexidade do contrato e quantidade de bibliotecas e interfaces. Os contratos de roteadores da DEX OKX se destacam pelo maior consumo de RAM devido ao número de bibliotecas e interfaces utilizadas, fazendo com que a ferramenta aloque mais recursos para buscar oportunidades de Sanduíche MEV.

Conforme a Figura 3, o contrato AggregationRouterV6 destaca-se com um tempo de execução de $28,76\,\text{s}$, situando-se a quase 6σ (seis desvios padrões) da média amostral. Por outro lado, apesar deste pico temporal, seu consumo de RAM ($159,46\,\text{MB}$) permaneceu dentro do quartil superior, mas não de forma isolada. Este desacoplamento sugere que para contratos de roteamento complexos, o gargalo migra do armazenamento de estados em memória para o processamento lógico intensivo, abrindo oportunidades para otimizações na forma de percorrer o CFG, a fim de processar de forma mais eficiente a elevada profundidade de chamadas, uma característica estrutural em agregadores de liquidez.

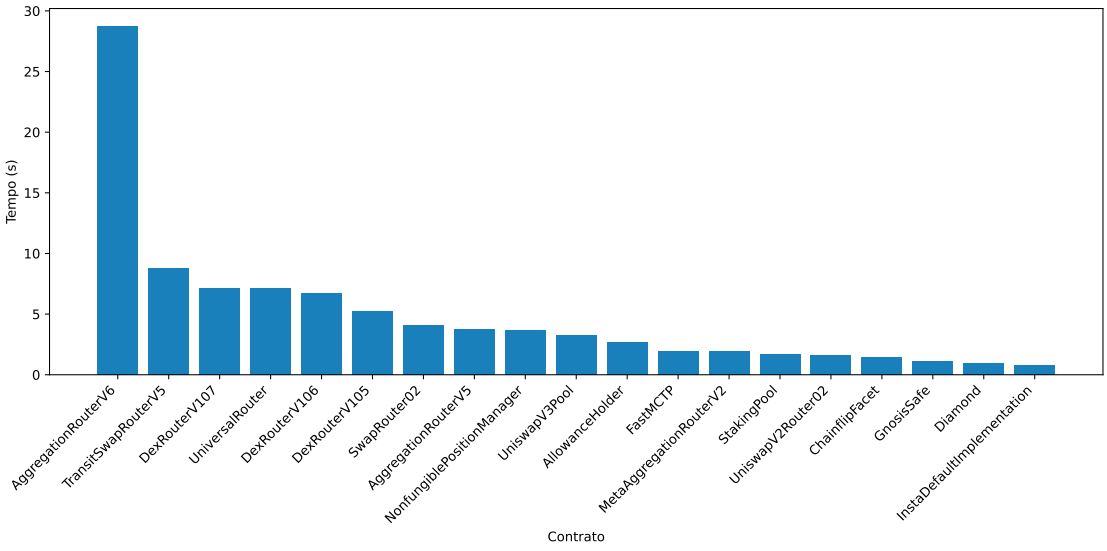

Figura 3. Tempo de execução gasto pela ferramenta por contrato.

O detector implementado demonstrou confiabilidade técnica durante a rodada de testes, apresentando uma taxa de erro de 0% em todos os 19 casos. Todos os contratos foram processados sem exceções de tempo de execução. A métrica de tempo de uso de CPU (s) reportou consistentemente valores nulos. Isso aponta que a execução do detector é governada majoritariamente por operações de I/O e gerenciamento de memória em vez de ciclos aritméticos.

O desempenho computacional apresentou uma variância significativa, com um tempo médio de execução de $4,88\,\text{s}$ ($\sigma = 6,27\,\text{s}$) e um consumo médio de memória RAM de $112,04\,\text{MB}$ ($\sigma = 56,05\,\text{MB}$) de acordo com a Tabela 3. O elevado desvio padrão em relação à média, particularmente no tempo de execução, indica uma carga de processamento altamente variável. Isso sugere que a complexidade estrutural dos contratos ou a profundidade atingida ao percorrer o CFG no algoritmo de análise divergem consideravelmente entre as amostras, exigindo que a ferramenta se adapte dinamicamente.

Tabela 3. Distribuição da detecção de oportunidades de Sanduíche MEV.

	Tempo (s)	**RAM (MB)**
Média	4,88	112,04
DP.	6,27	56,05
Mín.	0,82	44,17
25%	1,62	62,29
50%	3,24	107,89
75%	5,99	162,60
Máx.	28,76	210,21

6. Limitações

Apesar dos resultados preliminares promissores, o estudo apresenta limitações metodológicas que devem ser consideradas. Primeiramente, a significância estatística da pesquisa é restringida pelo tamanho reduzido do conjunto de dados de validação, composto por apenas 19 contratos inteligentes, o que limita a capacidade de generalização da acurácia reportada de 78,9%. Além disso, a ausência de avaliação cruzada com ferramentas do estado-da-arte restringe a comprovação das vantagens da abordagem apresentada em relação aos métodos já existentes na literatura. Apesar das restrições metodológicas, vale ressaltar que o desempenho em identificar a oportunidade da vulnerabilidade no conjunto de dados apresentado atesta o potencial prático para mitigar riscos antes da implantação dos contratos na rede, preenchendo uma lacuna arquitetural na segurança de aplicações DeFi.

7. Conclusão e Trabalhos Futuros

O presente trabalho propõe uma abordagem de análise estática para a detecção automática de suscetibilidade a Sanduíche MEV em contratos inteligentes Solidity. O detector implementado foi avaliado contra um dataset curado de 19 contratos inteligentes coletados da rede principal da blockchain Ethereum. Os resultados demonstram que o detector alcançou uma acurácia de 78,9%. Além disso, o tempo médio de análise por contrato foi de 4,88 segundos e consumo médio de RAM de 112,04 MB, evidenciando a viabilidade da técnica para uso contínuo no ciclo de desenvolvimento do contrato.

Com base nos resultados, o detector demonstra eficácia para detectar implementações complexas (envolvendo Assembly e bibliotecas internas), mas também para ignorar lógicas de segurança não relacionadas a troca de ativos. A abordagem estrutural pela análise de fluxo de dados ao percorrer o CFG elimina a dependência de nomes de variáveis, tornando o detector universalmente aplicável a qualquer protocolo desenvolvido em Solidity. Com uma taxa de falsos positivos baixa (14,29%), o detector demonstra alta confiabilidade. Entretanto, a taxa de falsos negativos de 25% sugere que a decisão de ignorar chamadas estáticas negligencia roteadores clássicos (V2) que validam a derrapagem de preço deterministicamente.

Como trabalhos futuros, esperamos melhorar o detector para supressão dos falsos negativos e falsos positivos, bem como para captar construções em padrões de contratos atualizáveis (*proxy upgradeable*) e chamadas delegadas (*delegatecall*), que têm o potencial de ofuscar o fluxo de dados real, restringindo o rastreamento preciso das variáveis e

a modelagem do CFG. Além disso, esperamos estudar métricas de classificação baseadas na localização da linha do alerta no código-fonte para melhorar a capacidade de detecção. Observamos que o detector pode acusar oportunidades de Sanduíche MEV particularmente em bibliotecas de cálculo aritmético e gestão de comissões. Este fenômeno ocorre porque tais bibliotecas frequentemente processam dados numéricos derivados de fontes ativas (como saldos) em operações de desigualdade, satisfazendo tecnicamente a condição estrutural do detector, embora não representem uma ameaça direta. Nesse sentido, embora a detecção implementada seja restrita, ela pode carecer de contexto para diferenciar oportunidades de Sanduíche MEV de operações matemáticas de arredondamento ou cálculo de taxas.

Para mitigar a limitação do conjunto de dados, é recomendável que trabalhos futuros ampliem o número de contratos inteligentes englobando protocolos menos convencionais e estendendo a análise para outras redes, como Binance Chain e Polygon, a fim de validar a generalização da abordagem baseada em Grafos de Fluxo de Controle (CFG). Em relação à limitação de avaliação cruzada, trabalhos futuros podem realizar comparações com ferramentas do estado-da-arte, como DeFort [Xie et al. 2024] e DeFiTainter [Kong et al.] para comprovar as vantagens da abordagem proposta em relação aos métodos já existentes na literatura.

Todos os artefatos referentes ao trabalho podem ser consultados no GitHub[4].

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001. Fapemig (Projeto #BIS-00699-25) e CNPq também apoiaram a execução deste trabalho.

Referências

Bodell III, W. E., Meisami, S., and Duan, Y. (2023). Proxy hunting: Understanding and characterizing proxy-based upgradeable smart contracts in blockchains. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1829–1846.

Campos, J. N., de Carvalho, L. H., Oliveira, I. R., Silva, A. C., Falcao, I. G., da Silva, M. J., Gonçalves, G. D., Vieira, A. B., and Nacif, J. A. (2025a). Análise das ferramentas de detecçao de vulnerabilidades para contratos inteligentes de blockchains evm. In *Workshop em Blockchain: Teoria, Tecnologias e Aplicações (WBlockchain)*, pages 126–139. SBC.

Campos, J. N. and et al. (2024). Finanças descentralizadas em redes blockchain: Perspectivas sobre pesquisa e inovação em aplicações, interoperabilidade e segurança. In *Jornada de Atualização em Informática 2024*, volume 44 of *Congresso da Sociedade Brasileira de Computação*, pages 7–56. SBC, Porto Alegre, 43nd edition.

Campos, J. N., Oliveira, I. R., Fontinele, A., Gonçalves, G. D., Vieira, A. B., and Nacif, J. A. M. (2025b). Impacto do paradigma de separação proponente-construtor em ataques sanduíche na rede ethereum. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 756–769. SBC.

[4]`https://github.com/lesc-ufv/static-sandwich-mev-detector`

Chaliasos, S., Charalambous, M. A., Zhou, L., Galanopoulou, R., Gervais, A., Mitropoulos, D., and Livshits, B. (2024). Smart contract and defi security tools: Do they meet the needs of practitioners? In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13.

Chu, H., Zhang, P., Dong, H., Xiao, Y., Ji, S., and Li, W. (2023). A survey on smart contract vulnerabilities: Data sources, detection and repair. *Information and Software Technology*, 159:107221.

Contro, F., Crosara, M., Ceccato, M., and Dalla Preda, M. (2021). Ethersolve: Computing an accurate control-flow graph from ethereum bytecode. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, pages 127–137. IEEE.

Feist, J., Grieco, G., and Groce, A. (2019). Slither: a static analysis framework for smart contracts. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pages 8–15. IEEE.

Ferreira Torres, C., Mamuti, A., Weintraub, B., Nita-Rotaru, C., and Shinde, S. Rolling in the shadows: Analyzing the extraction of mev across layer-2 rollups. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*.

Fontinele, A., Campos, J. N., Oliveira, I. R., Gonçalves, G. D., Nacif, J. A., Vieira, A. B., and Soares, A. C. (2024). Análise de ataques sanduíche sob as transações da blockchain ethereum. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 728–741. SBC.

Ivanov, N., Li, C., Yan, Q., Sun, Z., Cao, Z., and Luo, X. (2023). Security threat mitigation for smart contracts: A comprehensive survey. *ACM Computing Surveys*, 55(14s):1–37.

Kong, J. (2025). Sadt: Sandwich attack detection for transactions on decentralized exchanges. In *Ubiquitous Security: 4th International Conference, UbiSec 2024, Changsha, China, December 29–31, 2024, Revised Selected Papers*, volume 2469, page 145. Springer Nature.

Kong, Q., Chen, J., Wang, Y., Jiang, Z., and Zheng, Z. Defitainter: Detecting price manipulation vulnerabilities in defi protocols. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*.

Weintraub, B., Torres, C. F., Nita-Rotaru, C., and State, R. (2022). A flash (bot) in the pan: measuring maximal extractable value in private pools. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 458–471.

Wu, C., Chen, L., Wang, K., Han, W., and Chai, H. (2025). Sandwatch: Towards detecting sandwich attacks in ethereum using a dual-task graph neural network. *ACM Transactions on the Web*.

Xie, M., Hu, M., Kong, Z., Zhang, C., Feng, Y., Wang, H., Xue, Y., Zhang, H., Liu, Y., and Liu, Y. (2024). Defort: Automatic detection and analysis of price manipulation attacks in defi applications. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 402–414.

Zhou, L., Qin, K., Torres, C. F., Le, D. V., and Gervais, A. (2021). High-frequency trading on decentralized on-chain exchanges. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 428–445. IEEE.