Análise de Bugs em Contratos Inteligentes de Blockchains Compatíveis com Ethereum Virtual Machine: Janeiro a Setembro de 2023

Ana Julia Bittencourt Fogaça

[1]Universidade do Sul de Santa Catarina (UNISUL)
Tubarão - SC - Brasil
anajuliabit@gmail.com

1. Abstract

2. Resumo

Software é criado por seres humanos e, consequentemente, está sujeito a bugs. Isso não é diferente para aplicações que funcionam em blockchains que suportam a Ethereum Virtual Machine (EVM). Entretanto, em contraste com software convencional, as aplicações descentralizadas representam alvos particularmente lucrativos para hackers, dada a natureza transparente e de código aberto da blockchain. Neste artigo, analisamos 145 bugs descobertos durante auditorias públicas em vários projetos desenvolvidos em Solidity, operando na Ethereum ou em blockchains compatíveis com a EVM.

3. Introdução

Introduzida em 2008 pelo whitepaper do Bitcoin [14], a tecnologia blockchain é reconhecida como um vetor de transformação em múltiplas indústrias [20]. Suas características de segurança robusta, transparência e rastreabilidade, juntamente com a natureza de código aberto, contribuem para sua adoção em operações críticas de negócios. Até 2023, o uso mais evidente da blockchain, o mercado de criptomoedas, atingiu um valor de mercado superior a um trilhão de dólares [6]. A aplicabilidade da blockchain vai além das criptomoedas, influenciando áreas como finanças, gerenciamento do, identidade, saúde e governança eleitoral [2].

A inovação do Ethereum, lançada pelo seu *whitepaper* em 2014, representou um marco na evolução da blockchain. Ao contrário do Bitcoin, concebido como uma moeda eletrônica *peer-to-peer* [14], o Ethereum apresentou o conceito inovador de "contratos inteligentes"[8]. Esta funcionalidade ampliou o escopo da blockchain, permitindo sua aplicação em novos domínios. A plataforma Ethereum se destaca pela hospedagem de uma máquina virtual que executa códigos em linguagens *Turing-complete*. No entanto, sendo os contratos inteligentes criados por humanos, eles não estão isentos de falhas. Em um ambiente de código aberto, típico de blockchains como Ethereum, essas falhas podem se tornar alvos atrativos para hackers. Apenas no primeiro trimestre de 2023, ataques à rede Ethereum resultaram no roubo de 320 milhões de dólares [10]. Para minimizar tais riscos, é prática comum realizar auditorias nos contratos inteligentes antes de seu lançamento em ambiente de produção. As auditorias podem ser privadas, conduzidas por empresas especializadas, ou públicas, através de plataformas como Code4rena [4] e Sherlock [16], onde participantes diversos podem identificar vulnerabilidades, sendo recompensados individualmente ou em grupo pela descoberta de bugs.

Desde 2020, projetos de blockchain que negligenciaram o processo de auditoria sofreram comprometimentos financeiros da ordem de 3.69 bilhões de dólares, enquanto que projetos auditados reportaram perdas de 1.3 bilhões[5], sugerindo que, embora as auditorias reduzam a probabilidade de ataques bem-sucedidos, ainda há desafios na detecção antecipada de vulnerabilidades. Com a demanda por contratos inteligentes crescendo e uma projeção de aumento anual de 82,2% de 2023 a 2030 [17], é crucial entender e classificar as vulnerabilidades emergentes. Neste estudo, examinamos uma amostra de 145 bugs identificados em 31 competições de auditoria públicas realizadas entre janeiro e setembro de 2023 em duas plataformas de renome, Sherlock [16] e Code4rena [4]. Nosso objetivo é elucidar questões fundamentais como a complexidade na detecção de diferentes tipos de bugs, a incidência de categorias específicas de bugs em variados protocolos e a correlação entre vulnerabilidades exploradas por hackers e aquelas identificadas em competições de auditoria.

O artigo está divivido em..

4. Revisão bibliográfica

4.1. Ethereum Blockchain

Ethereum, conforme delineado no whitepaper por Vitalik Buterin et al., é uma plataforma descentralizada que permite a construção de aplicações financeiras em cima de uma infraestrutura de blockchain[8]. A blockchain é um sistema de registro distribuído e imutável que mantém um registro contínuo de transações ou dados em blocos ligados por criptografia. Esse design assegura a integridade e a veracidade dos dados, resistindo a alterações retroativas.

4.2. Contratos inteligentes

Contratos inteligentes são programas que rodam na blockchain do Ethereum, permitindo que as partes cumpram acordos sem a necessidade de um intermediário. Uma vez implantados, os contratos inteligentes não podem ser alterados, o que exige que o código seja verificado para potenciais vulnerabilidades antes do lançamento. Eles são fundamentais para a finança descentralizada e têm bilhões de dólares em valor atrelados a eles[11].

4.3. Ethereum Virtual Machine

A Ethereum Virtual Machine (EVM) é o ambiente de execução para contratos inteligentes na Ethereum. Funciona como uma camada global que pode executar código de contrato inteligente em um contexto descentralizado[22]. Isso possibilita que os desenvolvedores criem aplicações que funcionam exatamente conforme programadas, sem qualquer possibilidade de fraude ou interferência de terceiros. A EVM é isolada, significando que o código executado dentro dela não tem acesso ao sistema de arquivos da rede, a outros contratos inteligentes, ou a qualquer recurso externo. Esse isolamento garante um alto nível de segurança no ecossistema Ethereum.

4.4. Solidity

Solidity é uma linguagem de programação de alto nível para a implementação de contratos inteligentes e é fortemente tipada, suporta herança, bibliotecas e tipos de usuário complexos[18]. Projetada para se alinhar com a EVM, Solidity facilita o desenvolvimento de

contratos inteligentes através de uma sintaxe semelhante a JavaScript, tornando-a acessível a um amplo espectro de programadores. Solidity, apesar de ser uma linguagem de alto nível com características robustas, não está isenta de vulnerabilidades. Muitas delas decorrem de uma desconexão entre a semântica da linguagem e a intuição dos programadores, principalmente porque Solidity implementa características de linguagens conhecidas, como JavaScript, de maneiras peculiares. Além disso, a linguagem carece de construções específicas para lidar com aspectos do domínio de blockchain, como a imprevisibilidade na ordem ou no atraso das etapas de computação registradas publicamente na blockchain. Isso ressalta a importância de uma compreensão aprofundada de Solidity ao desenvolver contratos inteligentes, para mitigar o risco de vulnerabilidades de segurança.

4.5. Finanças descentralizadas

Finanças Descentralizadas (DeFi) representam uma infraestrutura financeira alternativa construída sobre a tecnologia blockchain, utilizando contratos inteligentes para replicar serviços financeiros existentes de forma mais aberta, interoperável e transparente[15]. DeFi é uma evolução tecnológica emergente que vem ganhando destaque juntamente com FinTech, RegTech, criptomoedas e ativos digitais, embora seu significado completo, implicações legais e consequências políticas ainda sejam pouco compreendidos. Este movimento revolucionário visa criar um sistema financeiro baseado apenas em código, sem intermediários, e viu um crescimento de ativos bloqueados de $4 bilhões para $104 bilhões em três anos[13]. Ainda é uma área complexa que exige uma compreensão rigorosa de suas nuances tanto por praticantes quanto por pesquisadores de sistemas de informação[9]. Como um novo setor de tecnologia financeira, DeFi tem o potencial de remodelar a estrutura da finança moderna e criar um novo cenário para empreendedorismo e inovação, prometendo e enfrentando desafios e limites[3].

4.6. Python

5. Metodologia e perguntas da pesquisa

5.1. Categoria dos protocolos

Os protocolos investigados neste estudo são dedicados ao setor de Finanças Descentralizadas (DeFi), abarcando exclusivamente as seguintes subcategorias conforme a classificação proposta por DefiLlama [7]:

- Derivativos: Protocolos que disponibilizam ferramentas para operações financeiras alavancadas, possibilitando que os usuários façam previsões e especulações acerca de valores futuros de ativos, amplificando suas projeções de lucro ou prejuízo.
- Yield Farming: Protocolos que incentivam a prática de staking ou fornecimento de liquidez por parte dos usuários, oferecendo recompensas por tais atividades.
- Agregadores de Yield: Protocolos que otimizam os rendimentos por meio da integração de diversas estratégias de *yield farming*.
- Opções: Protocolos que ofertam o direito, embora não a obrigação, de adquirir um ativo por um valor preestabelecido em um momento futuro.
- DAOs (Organizações Autônomas Descentralizadas): Entidades organizacionais inovadoras que operam sem centralização, com decisões sendo tomadas de forma coletiva pelos membros.

- Launchpads: Protocolos desenvolvidos para lançar novos projetos e criptoativos no mercado.
- Índices: Protocolos que rastreiam ou replicam a performance de uma série de ativos interligados.
- DEXs (Trocas Descentralizadas): Protocolos que permitem a troca de criptoativos de forma descentralizada.
- RWAs (Ativos do Mundo Real): Protocolos relacionados à tokenização de ativos físicos, como imóveis.
- Stablecoins: Criptomoedas com valor atrelado a moedas fiduciárias ou outros ativos, buscando manter sua estabilidade por intermédio de mecanismos descentralizados.
- Gestores de Liquidez: Protocolos que gerenciam posições de liquidez em formadores de mercado automatizados com liquidez concentrada.
- Empréstimos: Protocolos que permitem o empréstimo e a tomada de empréstimos de diversos ativos.

5.2. Classificação dos Bugs

A classificação das vulnerabilidades dos protocolos analisados neste trabalho segue uma taxonomia híbrida, combinando os modelos propostos por Zhang et al. [23] e as tags de Solodit [19], detalhada da seguinte forma:

- O: Fora do Escopo
  - Inacessibilidade ao código-fonte do projeto.
  - Bugs manifestados em componentes off-chain.
  - Contratos inteligentes desenvolvidos em linguagens distintas
- C01: Manipulação do Mempool / Vulnerabilidades de Front-Running
  - Ataques do tipo sandwich #TODO
  - Exploração baseado em *Flash loans* #TODO
- C02: Ataque de Reentrada Vulnerabilidades de reentrância, resultantes de chamadas externas realizadas antes da conclusão de atualizações de estado internas, possibilitando a um adversário explorar o estado inconsistente.
- C03: Atualizações de Estado Errôneas. Ausência ou incorreção na atualização de estado, tal como uma atualização que não deveria ser efetuada.
- C04: Configuração *Hardcoded* Inserção de valores ou parâmetros estáticos diretamente no código do contrato inteligente, o que pode representar um risco de segurança se houver necessidade de flexibilidade.
- C05: Escalada de Privilégios e Problemas de Controle de Acesso.
  - Chamada de funções privilegiadas sem restrições adequadas.
  - Fundos de usuários que podem ser imobilizados por falhas ou ausência de código de retirada.
- C06: Matemática Incorreta / Contabilidade Errônea. Erros de cálculo decorrentes de implementações matemáticas falhas, conduzindo a resultados imprecisos, incluindo:
  - Ordem incorreta de operações.
  - Retorno de valores inesperados.
  - Utilização de números incorretos para cálculos.
  - Erros de contabilidade.

  - Underflow/overflow.
- C07: Lógica de Negócios Quebrada. Defeitos na lógica de negócios ou protocolos que, mesmo alinhados à intenção do desenvolvedor, são inerentemente falhos.
  - Invocações de funções inesperadas ou omissas
  - Condições anômalas de ambiente ou contrato
  - Argumentos de função impróprios
- C08: Bugs Específicos da Implementação do Contrato. Bugs que não se enquadram claramente em outras categorias.
- C09: Falta de Proteção Contra Replay de Assinatura
  - Nonce ausente #TODO
  - Colisão de hash.
- C10: Verificação Ausente. Omissão crítica de condições ou validações essenciais no código.
- C11: DoS (Negação de Serviço). Vulnerabilidades que permitem a um atacante comprometer a resposta ou eficiência do contrato. Esta categoria inclui casos que não são bem descritos por outra classe e onde a consequência primária é o encerramento do contrato ou ineficiência operacional.
- C12: Validação de Dados Falhas na verificação ou saneamento de entradas, particularmente daquelas oriundas de fontes externas.
- C13: Correspondência de Lista Branca/Lista Negra. Gerenciamento inadequado de endereços baseado em listas predefinidas.
- C14: Arrays. Vulnerabilidades associadas ao manuseio inadequado de arrays, incluindo:
  - Leituras/escritas fora dos limites
  - Problemas na exclusão
  - Questões relacionadas ao redimensionamento de arrays

5.3. Coleta de dados

No período de janeiro a setembro de 2023, nossa análise focou em 31 das competições de auditoria pública realizadas nas plataformas Code4rena [4] e Sherlock [21], nas quais foram identificados 145 bugs de alta severidade. Com duração média de sete dias, estes eventos visaram a identificação de falhas críticas em contratos inteligentes antes de seu lançamento final. A Tabela 1 detalha as competições examinadas, incluindo o número de participantes, o prêmio total, a categoria do projeto, a quantidade de vulnerabilidades de alta severidade encontradas e o total de linhas de código abrangidas em cada competição. O valor total das recompensas distribuídas nesses eventos superou os dois milhões de dólares, com uma média de 150 participantes por evento. Após a fase de submissão, juízes especializados em auditoria de contratos inteligentes, selecionados pela comunidade, avaliaram a severidade dos bugs. Os bugs classificados como de alta severidade representam riscos significativos, incluindo a possibilidade de roubo ou perda de ativos digitais [12]. Todos os bugs, suas descrições e a classificação realizada estão disponíveis no nosso dataset [1]

5.4. Perguntas da pesquisa

Nos focamos nas seguintes perguntas de pesquisa:

- Q1: Que tipo de vulnerabilidade é mais difícil de ser encontrada por auditores?

Tabela 1. Visão Geral das Competições Avaliadas. 'HRF' (High Risk Findings) representa bugs de alta severidade. 'nSLOC' indica o número total de linhas de código associadas a cada competição.

Plataforma	Categoria	Competição	Prêmio	HRF	nSLOC	Par
Code4rena	DAO	Arbitrum security council election	90500	1	2184	
Code4rena	DAO	Llama	60500	2	2096	
Code4rena	Stablecoin	Lybra finance	60500	8	1762	
Code4rena	Dexes	Maia DAO ecosystem	300500	35	10997	
Code4rena	Yield	PoolTogether	121650	9	3324	
Code4rena	Yield	PoolTogether v5: part deux	42000	2	1001	
Sherlock	Lending	Ajna update	85600	6	5659	
Sherlock	Yield Agreggator	Blueberry	72500	10		
Sherlock	Yield Agreggator	Blueberry Update #3	23600	5	3633	
Sherlock	Opções	Bond options	23600	2	885	
Sherlock	Empréstimos	Cooler update	17000	4	512	
Sherlock	Dexes	GFX labs	20400	2	716	
Sherlock	Derivativos	GMX	200000	5	10571	
Sherlock	Lending	Iron bank	67400	1	2241	
Sherlock	Derivativos	Perennial	122000	1	4063	
Sherlock	Derivativos	Perennial v2	125200	6	2494	
Sherlock	Derivativos	Symmetrical	91000	8	3553	
Sherlock	Derivativos	Symmetrical Update	27600	2	3921	
Sherlock	Launchpad	Tokensoft	21400	1	769	
Sherlock	Stablecoin	Unitas protocol	36400	1	1433	
Code4rena	RWA	Ondo finance	60500	1	4365	
Sherlock	Índices	Index coop	130600	2	4383	
Sherlock	Stablecoin	USSD	18200	3	402	
Sherlock	RWA	Dinari	16000	1	575	
Sherlock	Dexes	RealWagmi	33200	5	1080	
Code4rena	DAO	Nouns DAO	100000	1	9098	
Sherlock	Dexes	DODO v3	57800	5	2079	
Sherlock	Derivativos	Hubble Exchange	60000	3	1945	
Code4rena	Stablecoin	Angle Protocol	52500	3	2276	
Code4rena	Gestores de liquidez	Arrakis	81400	2	2801	
Sherlock	Dexes	Unstoppable	36400	8	2035	
TOTALS			2255950	145	3095.1	1

- Q2: Que categoria de protocolo apresenta mais presença de bugs?
- Q3: Os auditores frequentemente perdem tipos específicos de bugs que são posteriormente explorados?
- Q4: Qual é o impacto financeiro médio de diferentes tipos de vulnerabilidades?
- Q5: Como a complexidade do contrato inteligente afeta a probabilidade de encontrar bugs?
- Q6: Qual a relação entre categoria de bugs e os diferente tipos de protocolos?

6. Análise e Discussão dos Resultados

6.1. Análise de dados coletados

A identificação de vulnerabilidades em contratos inteligentes é uma tarefa que exige extrema atenção aos detalhes e uma mentalidade orientada para a descoberta de falhas, similar à de um potencial atacante. Nesse cenário, os auditores enfrentam desafios significativos na condução de suas investigações, principalmente porque muitos bugs ainda passam despercebidos [5]. Para aprofundar nosso entendimento desses desafios, realizamos uma análise exploratória do , utilizando as bibliotecas Python Panda e Seaborn. O script empregado nessa análise está documentado em [**bittencourtsolidityccommonvulnerabilities2023**], e todos os gráficos nesta seção foram gerados a partir dele.

As principais descobertas da nossa análise incluem:

1. A detecção de bugs por um número limitado de auditores indica uma complexidade elevada na sua identificação. Como ilustrado na Figura 1, entre as diversas categorias de vulnerabilidades, aquelas associadas a Falhas Específicas na Implementação de Contratos (C08), seguidas pelas Vulnerabilidades de Reentrância (C02) e a Ausência de Verificações (C10), emergem como as mais desafiadoras de serem detectadas. Em contraste, as categorias como Problemas de Controle de Acesso e Escalada de Privilégios (C05), Arrays (C14), e Manipulação de Mempool / Vulnerabilidades de Front-Running (C01) são relativamente mais fáceis de serem identificadas.

1. C06 - Erros de Cálculo (Matemática Incorreta/Contabilidade Errônea): De acordo com a Figura 2, podemos notar que esta categoria é a mais prevalente, indicando que falhas em lógica matemática, como underflows/overflows ou sequências errôneas de operações, são desafios comuns no setor de Finanças Descentralizadas (DeFi). Isso é compreensível, considerando que o DeFi opera intensamente com cálculos matemáticos complexos. Esta observação implica que numerosos contratos inteligentes são vulneráveis devido a implementações falhas de operações matemáticas, potencialmente resultando em perdas financeiras substanciais.

1. C07 - Lógica de Negócios Quebrada: A segunda classificação mais comum conforme mostra a Figura 2, mostrando que muitos contratos têm falhas inerentes na lógica de negócios, que poderiam ser prevenidas com uma análise e teste mais detalhados das regras de negócio e cenários de contrato.
2. C03 - Atualizações de Estado Errôneas: A terceira classificação em frequência sugere que uma quantidade considerável de contratos tem falhas na lógica que gerencia o estado do contrato, o que pode levar a consequências imprevistas e possíveis vulnerabilidades de segurança.

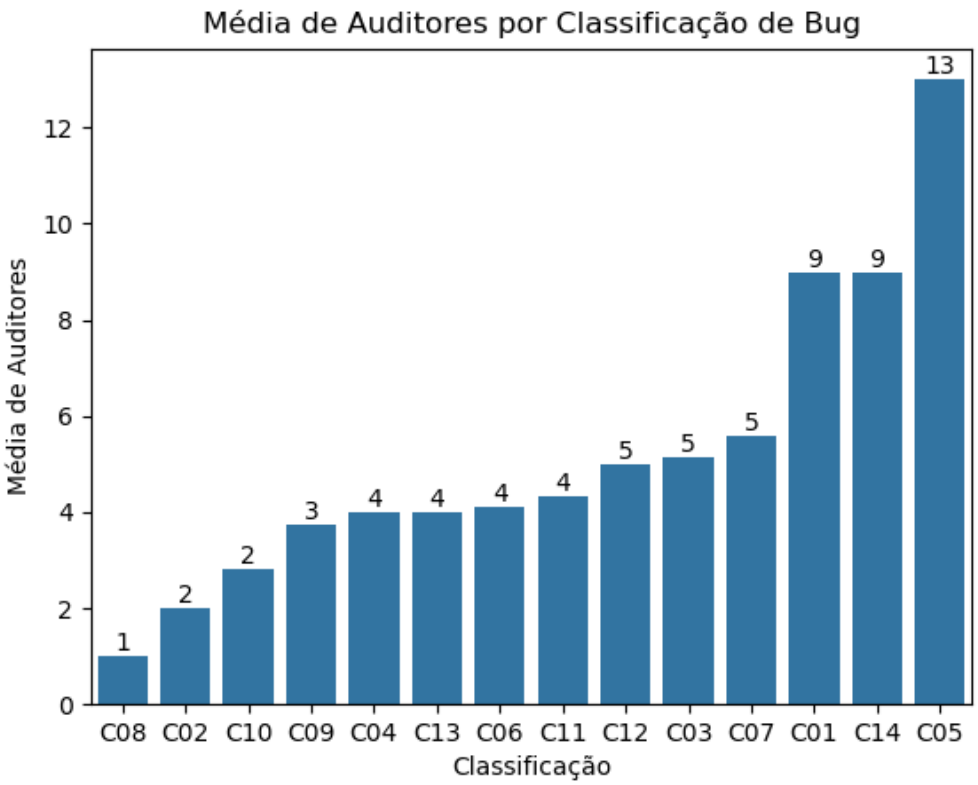

Figura 1. Média de auditores por classificação

3. Classes Menos Frequentes (C02, C08, C13): Vulnerabilidades como Reentrância (C02) e Falhas Específicas na Implementação de Contratos (C08) são menos frequentes, o que pode ser interpretado como um sinal de que são mais difíceis de detectar, o que condiz com a nossa descoberta 1 de que C02 e C08 são as categorias mais difíceis de serem identificadas. A classe C13, que inclui vulnerabilidades como o uso de funções de hash inseguras, também é menos frequente, o que pode indicar que os desenvolvedores estão mais conscientes dessas vulnerabilidades e as evitam.
4. C01 - Manipulação de Mempool/Vulnerabilidades de Front-Running: Apesar de menos comuns, estas vulnerabilidades podem ter impactos devastadores, permitindo ataques de 'sandwich' ou exploração baseada em empréstimos instantâneos ('flash loans'), destacando a necessidade de estratégias de mitigação robustas.

Vulnerabilidades de Implementação Específica (C08) e Proteção Contra Replay (C09): São menos comuns, o que pode sugerir que são específicas de certos contratos ou que a conscientização e ferramentas de detecção para essas vulnerabilidades são eficazes.

C14 - Arrays: Menos frequente, mas ainda assim significativa, essa categoria indica problemas no manuseio de arrays, como leituras/escritas fora dos limites, que são bugs mais técnicos e possivelmente menos explorados em ataques reais.

Classes dominantes de vulnerabilidades, como atualizações de estado incorretas (C03), problemas de controle de acesso e escalada de privilégios (C05), e erros de cálculo

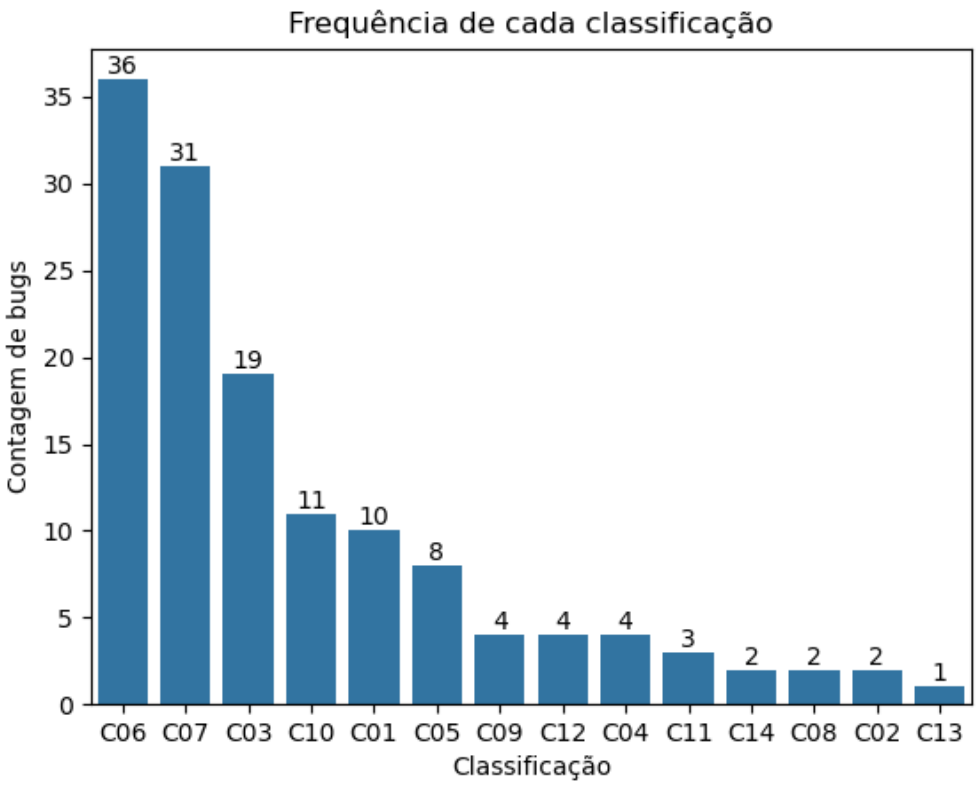

Figura 2. Frequncia de bugs por classificação

(C06), são recorrentes em várias categorias. Esses padrões indicam a prevalência desses tipos de vulnerabilidades.

Algumas classes de vulnerabilidades são menos frequentes, sugerindo especificidade para certos tipos de protocolos ou maior dificuldade na detecção, como é o caso da falta de proteção contra replay de assinaturas (C09), ausência de verificações (C10) e bugs específicos de implementação de contrato (C08).

As diferentes categorias de protocolo apresentam distintos perfis de vulnerabilidade. Por exemplo, protocolos Dexes apresentaram o maior número de bugs identificados.

Há uma correlação complexa entre as categorias de vulnerabilidade e os diferentes tipos de protocolos, com variações significativas nas classificações mais frequentes entre as categorias."

- Bugs cujo foram identificados por uma quantidade menor de auditores, indicam uma maior dificuldade na identificação dos mesmos. Dos vários tipos de vulnerabilidades, aqueles relacionados a bugs específicos da implementação do contrato (C8), seguidos por Ataque de Reentrada Vulnerabilidades de reentrância (C02) e Verificação Ausente (C10), são particularmente difíceis de identificar. Inversamente, C05, C14 e C01 são os mais fáceis de identificar.
- Atualizações de estado errôneas (C03), escalada de privilégios e problemas de controle de acesso (C05) e cálculos errados (C06) são as classes dominantes de vulnerabilidades em várias categorias. Isso sugere que esses tipos de vulnerabili-

dade são comuns em geral. Especificamente, atualizações de estado errôneas são encontradas em oito categorias, enquanto escalada de privilégios e problemas de controle de acesso (C05) e cálculos errados (C06) aparecem em sete categorias.

- Certas classes de vulnerabilidades aparecem com menos frequência, indicando que podem ser específicas para certos tipos de protocolo ou mais desafiadoras de encontrar. Por exemplo, classes como falta de proteção contra replay de assinatura (C09), checagem ausente (C10) e bugs específicos da implementação de contrato (C08) são escassos.
- Diferentes categorias de protocolo têm diferentes perfis de vulnerabilidade. Protocolos Dexes tiveram o maior número de bugs identificados, com 47 classificados, seguidos por Derivativos com 25 e Agregador e Stablecoin com 15 cada.
- Cálculos errados (C06) é a classificação mais frequente em todas as categorias, com 36 bugs classificados dessa forma. Lógica de negócios quebrada (C07) é a segunda classificação mais frequente, com 31 bugs identificados. Atualizações de estado errôneas (C03) e checagem ausente (C10) seguem com 19 e 11 bugs classificados, respectivamente. Finalmente, 10 bugs foram classificados como Manipulação de Mempool / Vulnerabilidades de Front-Running (C01).
- A relação entre categorias de vulnerabilidade e diferentes tipos de protocolo é complexa. Cálculos errados (C06) é a classificação mais frequente encontrada em Derivativos, Agregador de Rendimentos e Stablecoins, enquanto Lógica de negócios quebrada (C07) é a classificação mais frequente encontrada em Dexes e Rendimentos. Além disso, arrays (C14) e cálculos errados (C06) são o segundo tipo mais comum de bugs encontrados em Dexes.

6.2. Perguntas de pesquisa

1. Por que Bugs Específicos da Implementação do Contrato (C08) São Mais Desafiadores de Identificar? Esta pergunta visa explorar os fatores subjacentes que tornam bugs específicos, particularmente C08 e C02, mais difíceis de serem identificados por auditores.
2. O Que Torna Certas Vulnerabilidades Mais Fáceis de Detectar? Focando em vulnerabilidades como C05, C14 e C01, esta pergunta busca entender as características que as tornam mais aparentes para os auditores.
3. Vulnerabilidades Comuns Entre Categorias de Protocolo Indicam Problemas Sistêmicos? Investigar por que atualizações de estado errôneas (C03), escalada de privilégios (C05) e cálculos errados (C06) são comuns em várias categorias pode revelar problemas sistêmicos mais amplos no design de contratos inteligentes.
4. Quais Fatores Contribuem para a Raridade de Certas Classes de Vulnerabilidade? Olhando para vulnerabilidades menos frequentes como C09, C10 e C08, esta pergunta investiga por que essas classes são menos comuns, explorando se são específicas do protocolo ou inerentemente difíceis de detectar.
5. Como Diferentes Categorias de Protocolo Influenciam o Perfil de Vulnerabilidades? Esta pergunta visa entender a relação entre categorias de protocolo (como Dexes, Derivativos

6.3. Resultados

6.3.1. Q1: Que tipo de vulnerabilidade é mais difícil de ser encontrada por auditores?

Em nossa análise sobre a dificuldade de detecção de vulnerabilidades por auditores em contratos inteligentes, identificamos que as vulnerabilidades com uma média menor de auditores envolvidos na detecção tendem a ser mais desafiadoras. Especificamente, as categorias Específicos da Implementação do Contrato (C08) e Ataque de Reentrada (C02) emergem como as mais complexas, com médias de apenas 1 e 2 auditores, respectivamente. Isso indica que essas vulnerabilidades podem ser mais sutis ou requerer uma expertise mais especializada, o que as torna menos detectáveis em auditorias.

Contrastando, categorias como Escalada de Privilégios e Problemas de Controle de Acesso (C05), com média de 13 auditores, Manipulação do Mempool / Vulnerabilidades de Front-Running (C01) e Arrays (C14), ambas com média de 9 auditores, Lógica de Negócios Quebrada (C07), com média de aproximadamente 5.58 auditores, e Atualizações de Estado Errôneas (C03), com média de aproximadamente 5.16 auditores, demonstram ser mais facilmente detectáveis. Essas categorias possivelmente são mais evidentes ou exigem menos especialização técnica para sua identificação, culminando em uma frequência maior de detecção em auditorias. Esta análise sublinha a necessidade de uma compreensão aprofundada das diferentes classes de vulnerabilidades em contratos inteligentes e enfatiza o papel crucial dos auditores especializados em sua identificação e mitigação.

- Q2: Que categoria de protocolo apresenta mais presença de bugs?
- Q3: Os auditores frequentemente perdem tipos específicos de bugs que são posteriormente explorados?
- Q4: Qual é o impacto financeiro médio de diferentes tipos de vulnerabilidades?
- Q5: Como a complexidade do contrato inteligente afeta a probabilidade de encontrar bugs?
- Q6: Qual a relação entre categoria de bugs e os diferente tipos de protocolos?

7. Considerações finais

Referências

[1] Ana Julia Bittencourt. *Solidity Common Vulnerabilities*. Out. de 2023. URL: https://github.com/anajuliabit/solidity-common-vulnerabilities (acesso em 12/11/2023).

[2] *Blockchain Adoptions in the Maritime Industry: A Conceptual Framework*. URL: https://www.tandfonline.com/doi/epdf/10.1080/03088839.2020.1825855?needAccess=true (acesso em 06/10/2023).

[3] Yan Chen e Cristiano Bellavitis. "Blockchain Disruption and Decentralized Finance: The Rise of Decentralized Business Models". Em: *Journal of Business Venturing Insights* 13 (jun. de 2020), e00151. ISSN: 2352-6734. DOI: 10.1016/j.jbvi.2019.e00151. URL: https://www.sciencedirect.com/science/article/pii/S2352673419300824 (acesso em 09/11/2023).

[4] *Code4rena | Keeping High Severity Bugs out of Production*. URL: https://code4rena.com/ (acesso em 01/11/2023).

[5] *Competitive vs Private Audits - Pros and Cons | The Full Comparison*. URL: https://www.cyfrin.io/blog/competitive-vs-private-audits-comparison (acesso em 09/11/2023).

[6] *Cryptocurrency Statistics 2023*. Set. de 2023. URL: https://www.forbes.com/advisor/au/investing/cryptocurrency/cryptocurrency-statistics/ (acesso em 27/10/2023).

[7] *DefiLlama*. URL: https://defillama.com/categories (acesso em 06/11/2023).

[8] *Ethereum Whitepaper*. URL: https://ethereum.org (acesso em 02/10/2023).

[9] Vincent Gramlich et al. "A Multivocal Literature Review of Decentralized Finance: Current Knowledge and Future Research Avenues". Em: *Electronic Markets* 33.1 (abr. de 2023), p. 11. ISSN: 1422-8890. DOI: 10.1007/s12525-023-00637-4. URL: https://doi.org/10.1007/s12525-023-00637-4 (acesso em 09/11/2023).

[10] *Here's How Much Was Lost to Crypto Hacks and Exploits in Q1 2023 | Bitcoin Insider*. URL: https://www.bitcoininsider.org/article/211488/heres-how-much-was-lost-crypto-hacks-and-exploits-q1-2023 (acesso em 01/10/2023).

[11] *JCP | Free Full-Text | The State of Ethereum Smart Contracts Security: Vulnerabilities, Countermeasures, and Tool Support*. URL: https://www.mdpi.com/2624-800X/2/2/19 (acesso em 02/11/2023).

[12] *Judging Criteria*. URL: https://docs.code4rena.com/awarding/judging-criteria%5C#estimating-risk (acesso em 02/11/2023).

[13] Eva Meyer, Isabell M. Welpe e Philipp G. Sandner. *Decentralized Finance—A Systematic Literature Review and Research Directions*. SSRN Scholarly Paper. Rochester, NY, 2022. DOI: 10.2139/ssrn.4016497. URL: https://papers.ssrn.com/abstract=4016497 (acesso em 09/11/2023).

[14] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". Em: ().

[15] Fabian Schär. *Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets*. DOI: 10.20955/r.103.153-74. URL: https://research.stlouisfed.org/publications/review/2021/02/05/decentralized-finance-on-blockchain-and-smart-contract-based-financial-markets (acesso em 09/11/2023).

[16] *Sherlock*. URL: https://www.sherlock.xyz/ (acesso em 01/11/2023).

[17] *Smart Contracts Market Size, Share, & Trends [2023 Report]*. URL: https://www.grandviewresearch.com/industry-analysis/smart-contracts-market-report (acesso em 02/10/2023).

[18] *Solidity — Solidity 0.8.22 Documentation*. URL: https://docs.soliditylang.org/en/v0.8.22/ (acesso em 02/11/2023).

[19] *Solodit_content/Report_tags.Md at Main · Solodit/Solodit_content*. URL: https://github.com/solodit/solodit%5C_content/blob/main/report%5C_tags.md (acesso em 06/11/2023).

[20] *Technology Tipping Points and Societal Impact*. URL: https://www3.weforum.org/docs/WEF%5C_GAC15%5C_Technological%5C_Tipping%5C_Points%5C_report%5C_2015.pdf (acesso em 06/10/2023).

[21] Johnny Time. *The Most Interesting Web3 Security Interview with Peter Kacherginsky*. Out. de 2023. URL: https://medium.com/@JohnnyTime/the-most-interesting-web3-security-interview-with-peter-kacherginsky-cc03b0a30930 (acesso em 13/10/2023).

[22] Dr Gavin Wood. "ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER". Em: ().

[23] Zhuo Zhang et al. "Demystifying Exploitable Bugs in Smart Contracts". Em: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. Melbourne, Australia: IEEE, mai. de 2023, pp. 615–627. ISBN: 978-1-66545-701-9. DOI: 10.1109/ICSE48619.2023.00061. URL: https://ieeexplore.ieee.org/document/10172700/ (acesso em 16/10/2023).