# Uma análise classificatória em bugs encontrados em contratos inteligentes escritos em Solidity entre janeiro e 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

## 3. Introdução

Introduzida pela primeira vez em 2008 através do whitepaper do Bitcoin[**nakamotoBitcoinPeertoPeerElectronic**], a tecnologia blockchain é reconhecida como uma megatendência em computação com o potencial para transformar diversas indústrias[5]. Suas características únicas de segurança, transparência e rastreabilidade têm incentivado uma ampla variedade de setores a adotá-la para remodelar suas operações essenciais. Até 2023, o valor de mercado das criptomoedas, o caso de uso mais popular da blockchain até o momento, ultrapassou um trilhão de dólares[**CryptocurrencyStatistics20232023**]. A aplicabilidade da blockchain vai além das criptomoedas, abrangendo áreas como finanças, gerenciamento de identidade, saúde e governança eleitoral[1].

A publicação do whitepaper do Ethereum em 2014 marcou um avanço significativo na evolução da blockchain. Diferente do Bitcoin, que foi projetado inicialmente como uma versão p2p de dinheiro eletrônico[**nakamotoBitcoinPeertoPeerElectronic**], o Ethereum introduziu a noção revolucionária de contratos inteligentes[2]. Essa funcionalidade expandiu o alcance da tecnologia blockchain para novos setores. A inovação do Ethereum reside em sua capacidade de suportar uma máquina virtual que pode executar códigos em linguagens de programação *Turing-complete*. No entanto, como qualquer software, contratos inteligentes são desenvolvidos por humanos e, portanto, estão sujeitos a erros. Em um ambiente de código aberto que é inerente à blockchain, essas vulnerabilidades se tornam alvos lucrativos para hackers. Somente no primeiro trimestre de 2023, o Ethereum perdeu 320 milhões de dólares devido a ataques cibernéticos[3]. Para mitigar esses riscos, plataformas blockchain frequentemente oferecem recompensas financeiras em troca de vulnerabilidades encontradas através de competições realizadas em plataformas.

Com a crescente demanda por contratos inteligentes e uma expectativa de aumento anual de 82,2% de 2023 a 2030[4], torna-se fundamental compreender e categorizar as vulnerabilidades recentes. Neste artigo, analisamos um conjunto de dados de 154 bugs extraídos de 31 competições realizadas entre janeiro a setembro de 2023, através de duas plataformas de alta reputação,[**Sherlock**] e[**Code4renaKeepingHigh**]. Nosso estudo busca esclarecer questões críticas, como a dificuldade de detecção de diferentes tipos de bugs, a prevalência de certas categorias de bugs em distintos protocolos, e a

relação entre as recentes vulnerabilidades exploradas por hackers e as vulnerabilidades encontradas nas competições.

## 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[2]. 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[**JCPFreeFullText**].

### 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[**woodETHEREUMSECUREDECENTRALISED**]. 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[**SolidparticleSolidity22**]. 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 [**SolidityVulnerabilities**]. Isso ressalta a importância de uma compreensão aprofundada de Solidity ao desenvolver contratos inteligentes, para mitigar o risco de vulnerabilidades de segurança.

### 4.5. Protocolos

1. **TODO** explain DEFI

### 4.6. ERCs

## 5. Metodologia e perguntas da pesquisa

### 5.1. Coleta de dados

Foram coletados 145 bugs encontrados em 31 competições que aconteceram entre Janeiro a Setembro de 2023 nas plataformas Code4rena[**Code4renaKeepingHigh**] e [**Sherlock**]. Todos foram classificados com severidade alta - ativos podem ser roubados, perdidos ou compromissados[**JudgingCriteria**]. Geralmente as competições duram em torno de 7 dias e o objetivo é capturar bugs antes de realizar o deploy oficial. As somas das premiações das 31 competições analisadas ultrapassam dois milhões de dólares, e em média aproximadamente participam 150 participantes por competição.

| Plataforma | Categoria | Competição | Prêmio | HRF | nSLOC |
|---|---|---|---|---|---|
| Code4rena | DAO | Arbitrum security council election system | 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 | Options | Bond options | 23600 | 2 | 885 |
| Sherlock | Lending | Cooler update | 17000 | 4 | 512 |
| Sherlock | Dexes | GFX labs | 20400 | 2 | 716 |
| Sherlock | Derivatives | GMX | 200000 | 5 | 10571 |
| Sherlock | Lending | Iron bank | 67400 | 1 | 2241 |
| Sherlock | Derivatives | Perennial | 122000 | 1 | 4063 |
| Sherlock | Derivatives | Perennial v2 | 125200 | 6 | 2494 |
| Sherlock | Derivatives | Symmetrical | 91000 | 8 | 3553 |
| Sherlock | Derivatives | 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 | Indexes | 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 | Derivatives | Hubble Exchange | 60000 | 3 | 1945 |
| Code4rena | Stablecoin | Angle Protocol | 52500 | 3 | 2276 |
| Code4rena | Liquidity manager | Arrakis | 81400 | 2 | 2801 |
| Sherlock | Dexes | Unstoppable | 36400 | 8 | 2035 |
| TOTALS | | | 2255950 | 145 | 3095.1 |

### 5.2. Categoria dos protocolos

As competições referem a diferente protocolos do setor de Finanças Descentralizadas que podem ser classificados em:

- Derivativos: Protocolos para apostas com alavancagem, permitindo que os usuários especulem sobre preços futuros de ativos com a possibilidade de ampliar seus ganhos (ou perdas).
- *Yield*: Protocolos que recompensam os usuários por fazer *"stake"* ou fornecer liquidez em suas plataformas.
- Agregadores de Yield: Protocolos que combinam e otimizam o rendimento de diferentes fontes de yield.
- Opções: Protocolos que oferecem o direito de comprar um ativo por um preço fixo em uma data futura.

- DAO (Organização Autônoma Descentralizada): Estruturas legais emergentes sem um corpo governante central, onde as decisões são tomadas de forma coletiva pelos membros.
- Launchpad: Protocolos que lançam novos projetos e criptomoedas no mercado.
- Índices: Protocolos que rastreiam ou criam o desempenho de um grupo de ativos relacionados.
- Dexes (Trocas Descentralizadas): Protocolos que permitem a troca ou negociação de criptomoedas.
- RWA (Ativos do Mundo Real): Protocolos envolvendo a tokenização de ativos do mundo real, como imóveis.
- Stablecoin: Moedas estáveis atreladas ao dólar ou outras moedas fiduciárias através de mecanismos descentralizados.
- Gestores de Liquidez: Protocolos que gerenciam posições de liquidez em formadores de mercado automatizados com liquidez concentrada.
- Empréstimos: Protocolos que possibilitam aos usuários emprestar e tomar emprestado ativos diversos.

### 5.3. Perguntas da pesquisa

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

### 5.4. Classificação dos bugs

- O: Out-of-scope
  - We cannot access the source code of the project.
  - Bugs that occur in off-chain components
  - Smart contracts are written in another language
- C01: Mempool Manipulation / Front-Running Vulnerabilities, (e.g sandwich attacks, flash-loan exploits)
- C02: Reentry attack Reentrancy vulnerabilities happen when external contract calls are made before internal state updates, allowing an adversary to recursively call back into the contract, exploiting the inconsistent state.
- C03: Erroneous state updates. Missing state update, or incorrect state updates, e.g., a state update that should not be there.
- C04: Hardcoded Setting - refers to the practice of embedding fixed values or parameters directly into the source code of a smart contract. This can pose a security risk if the setting needs to be dynamic or adaptable.
- C05: Privilege escalation and access control issues.
  - Privileged functions can be called by anyone or at any time.
  - User funds can get locked due to missing/wrong withdraw code

- C06: Wrong Math / Erroneous accounting. Wrong Math refers to a potential issue where mathematical operations within a smart contract are implemented incorrectly, leading to inaccurate calculations.
  - Incorrect calculating order.
  - Returning an unexpected value that deviates from the expected semantics specified for the contract.
  - Calculations performed with incorrect numbers (e.g., x = a + b ==> x = a + c, incorrect precisions).
  - Other accounting errors (e.g., x = a + b ==> x = a - b).
  - Underflow/overflow
- C07: Broken business logic

Logic vulnerabilities involve flaws in the business logic or protocols of a smart contract, where the implementation matches the developer's intention, but the underlying logic is inherently flawed.

- Unexpected or missing function invocation sequences (e.g., external calls to dependent contracts, exploitable sequences leading to malicious fund reallocation or manipulation).
- Unexpected environment or contract conditions (e.g., ChainLink returning outdated data or significant slippage occurring).
- Unexpected function arguments.

- C08: Contract implementation-specific bugs. These bugs are difficult to categorize into others categories.
- C09: Lack of signature replay protection, e.g missing nonce, hash collision
- C10: Missing check. Missing Check refers to a critical oversight in a smart contract's code where a necessary condition or validation is not properly implemented.
- C11: lack of segregation between users funds
- C12: Data validation: Data validation vulnerabilities arise when a smart contract does not adequately verify or sanitize inputs, especially those from untrusted sources. This lack of validation can lead to unintended and potentially harmful consequences within the contract's operations.
- C13: Whitelist/Blacklist Match Whitelist/Blacklist Match refers to a potential vulnerability where a smart contract improperly handles addresses based on predefined lists.
- C14: Arrays Vulnerabilities related to arrays can arise when developers do not properly handle array indices or fail to validate user inputs.

would typically be reserved for vulnerabilities that directly arise from mishandling or misinterpreting arrays in the code. For example, if there were out-of-bound reads/writes, deletion mishaps, or issues with array resizing

- C15: DoS Denial of Service (DoS) vulnerabilities occur when an attacker can exploit a contract in a way that makes it unresponsive or significantly less efficient. This category includes cases that are not well described by another class and where the primary consequence is contract shut-down or operational inefficiency.

### 5.5. Dados coletados

### 5.6. Desenvolvimento

### 5.7. Categorias

### 5.8. Dificuldade

## 6. Referências

## References


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

[2] *Ethereum Whitepaper*. URL: https://ethereum.org (visited on 10/02/2023).

[3] *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 (visited on 10/01/2023).

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

[5] *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 (visited on 10/06/2023).