Explorando e classificando bugs comumente encontrados em contratos inteligentes

Ana Julia Bittencourt Fogaça

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

1. Abstract

2. Resumo

3. Introdução

A tecnologia blockchain, primeiramente introduzida por Satoshi Nakamoto em 2008, é identificada como uma megatendência computacional capaz de revolucionar múltiplos setores industriais[5]. As características distintas de segurança, transparência e rastreabilidade inerentes à blockchain têm incentivado uma ampla gama de setores a explorar seu uso na reestruturação de suas operações fundamentais. A aplicabilidade dessa tecnologia ultrapassa o domínio das criptomoedas, abarcando setores como pagamentos, gerenciamento de identidade, saúde, eleições governamentais e outros[1].

A publicação do whitepaper do Ethereum em 2014 simbolizou um avanço considerável evolução da tecnologia blockchain[2]. Diferentemente do Bitcoin, concebido originalmente como uma moeda digital, o Ethereum inaugurou uma funcionalidade disruptiva no campo da tecnologia blockchain: os contratos inteligentes. A inovação trazida pelo Ethereum reside na incorporação de uma máquina virtual capaz de processar códigos em linguagens de programação *Turing complete* na blockchain, habilitando assim a construção de aplicativos descentralizados. Devido as características inerentes a tecnologia blockchain, como o fato de seu código ser aberto e qualquer pessoa pode interagir com os contratos inteligentes - descentralização, os aplicativos que rodam no Ethereum são sucetíveis a vulnerabilidades que podem ser exploradas por hackers, resultando em grande prejuízo financeiro para os protocolos e usuários dos mesmos. Apenas no primeiro trimestre de 2023, 320 milhões de dólares foram perdidos devido a ataque de hackers no Ethereum[3]. Uma maneira de combater a ação de hackers, é através de incentivos financeiros. Procurando proteger seus usuários, protocolos descentralizados costumam oferecer "Bug Bounties", que são concursos oferecendo recurso financeiro em troca de vulnerabilidades encontradas por "hackers do bem". Devido a demanda crescente pela tecnologia de contrato inteligentes nos últimos anos a projeção de crescimento anual de 2023 a 2030 é de 82.2%[4], o presente artigo tem como objetivo identificar os bugs comumente encontrados nas diferentes categorias de contratos inteligentes e classificá-los, identificando possíveis dificuldades na identificação dos mesmos. Para isso, foi feito um estudo com base em competições realizadas entre janeiro a setembro de 2023 retiradas de diferentes plataformas de Bug Bounties.

4. Revisão bibliográfica

O que é EVM, EOA, contracts, transactions (nonce).

5. Metodologia

5.1. Perguntas

- Categorizando bugs

5.2. Categorias dos protocolos

- Liquid Staking: Protocols that enable you to earn staking rewards on your tokens while also providing a tradeable and liquid receipt for your staked position
- Lending: Protocols that allow users to borrow and lend assets
- Dexes: Protocols where you can swap/trade cryptocurrency
- Bridge: Protocols that bridge tokens from one network to another
- CDP: Protocols that mint its own stablecoin using collateralized lending
- Services: Protocols that provide a service to the user
- Yield: Protocols that pay you a reward for your staking/LP on their platform
- RWA: Protocols that involve Real World Assets, such as house tokenization
- Derivatives: Protocols for betting with leverage
- Yield Aggregator: Protocols that aggregated yield from diverse protocols
- Cross Chain: Protocols that add interoperability between different blockchains
- Synthetics: Protocol that created a tokenized derivative that mimics the value of another asset.
- Launchpad: Protocols that launch new projects and coins
- Indexes: Protocols that have a way to track/created the performance of a group of related assets
- Liquidity manager: Protocols that manage Liquidity Positions in concentrated liquidity AMMs
- Insurance: Protocols that are designed to provide monetary protections
- Privacy: Protocols that have the intention of hiding information about transactions
- Infrastructure
- Algo-Stables: Protocols that provide algorithmic coins to stablecoins
- Payments: Protocols that offer the ability to pay/send/receive cryptocurrency
- Leveraged Farming: Protocols that allow you to leverage yield farm with borrowed money
- Staking Pool: Refers to platforms where users stake their assets on native blockchains to help secure the network and earn rewards. Unlike Liquid Staking, users don't receive a token representing their staked assets, and their funds are locked up during the staking period, limiting participation in other DeFi activities
- NFT Marketplace: Protocols where users can buy/sell/rent NFTs
- NFT Lending: Protocols that allow you to collateralize your NFT for a loan
- Options: Protocols that give you the right to buy an asset at a fixed price
- Options Vault: Protocols that allow you to deposit collateral into an options strategy
- Prediction Market: Protocols that allow you to wager/bet/buy in future results
- Decentralized Stablecoin: Coins pegged to USD through decentralized mechanisms
- Farm: Protocols that allow users to lock money in exchange for a protocol token
- Uncollateralized Lending:Protocol that allows you to lend against known parties that can borrow without collaterall

- Reserve Currency: OHM forks: Protocols that uses a reserve of valuable assets acquired through bonding and staking to issue and back its native token
- RWA Lending: Protocols that bridge traditional finance and blockchain ecosystems by tokenizing real-world assets for use as collateral or credit assessment, enabling decentralized lending and borrowing opportunities.
- Gaming: Protocols that have gaming components
- Oracle: Protocols that connect data from the outside world (off-chain) with the blockchain world (on-chain)
- P2P File distributoin system
- DAO: A decentralized autonomous organization (DAO) is an emerging form of legal structure that has no central governing body and whose members share a common goal to act in the best interest of the entity. Popularized through cryptocurrency enthusiasts and blockchain technology, DAOs are used to make decisions in a bottom-up management approach.

  Fonte: `https://defillama.com/categories`

5.3. Classificação dos bugs

- O1: We cannot access the source code of the project.
- O2: Bugs that occur in off-chain components
- O3: Smart contracts are written in another language
- C1: Mempool Manipulation / Front-Running Vulnerabilities, (e.g sandwich attacks, flash-loan exploits)
- C3: Erroneous state updates.
  - C3-1: Missing state update.
  - C3-2: Incorrect state updates, e.g., a state update that should not be there.
- C5: Privilege escalation and access control issues.
  - C5-1: Users can update privileged state variables arbitrarily (caused by lack of ID-unrelated input sanitization).
  - C5-2: Users can invoke some functions at a time they should not be able to do so.
  - C5-3: Privileged functions can be called by anyone or at any time.
  - C5-4: User funds can get locked due to missing/wrong withdraw code
  - C5-6: Privileged users can profit unfarly
- C6: 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.
  - C6-1: Incorrect calculating order.
  - C6-2: Returning an unexpected value that deviates from the expected semantics specified for the contract.
  - C6-3: Calculations performed with incorrect numbers (e.g., x = a + b ==> x = a + c, incorrect precisions).
  - C6-4: Other accounting errors (e.g., x = a + b ==> x = a - b).
- C7: 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.

    - C7-1: Unexpected or missing function invocation sequences (e.g., external calls to dependent contracts, exploitable sequences leading to malicious fund reallocation or manipulation).
    - C7-2: Unexpected environment or contract conditions (e.g., ChainLink returning outdated data or significant slippage occurring).
    - C7-3: A given function is invoked multiple times unexpectedly.
    - C7-4: Unexpected function arguments.
- C8: Contract implementation-specific bugs. These bugs are difficult to categorize into the above categories.
- C9: 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.
- C16: Grielf Attack: A gas griefing attack happens when a user sends the amount of gas required to execute the target smart contract, but not its sub calls. In most cases, this results in uncontrolled behavior that could have a dangerous impact on the business logic.
- C17: 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.
- C18: 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.

5.4. Dados coletados

Foi feito a curadoria de X bugs classificados com severidade alta

5.5. Desenvolvimento

5.6. Categorias

5.7. 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).