UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ALLEX ANTONIOLLO

SEGURANÇA E PRIVACIDADE EM CONTRATOS INTELIGENTES: DESAFIOS E SOLUÇÕES NO AMBIENTE BLOCKCHAIN

CHAPECÓ
2023

ALLEX ANTONIOLLO

SEGURANÇA E PRIVACIDADE EM CONTRATOS INTELIGENTES: DESAFIOS E SOLUÇÕES NO AMBIENTE BLOCKCHAIN

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação na Universidade Federal da Fronteira Sul.

Orientador: Prof. EDIMAR ROQUE MARTELLO JUNIOR

CHAPECÓ
2023

DEDICATÓRIA

Dedico este trabalho a meus pais.

“The art of simplicity is a puzzle of complexity.”
(Douglas Horton)

RESUMO

Este estudo concentra-se na análise crítica de desafios de solução relacionados à segurança e contratos inteligentes dentro do ecossistema blockchain. Explorarei possíveis vulnerabilidades e ameaças à integridade examinando casos de uso práticos. A pesquisa tem como objetivo identificar estratégias eficazes para riscos, destacando melhores práticas tecnológicas que promovam a implementação segura de inteligência. Ao abordar esses tópicos, meu objetivo é encontrar sugestões para uma compreensão mais profunda e melhoria do no contexto de contratos inteligentes e dos contratos inteligentes na blockchain.

Palavras-Chave: blockhain, contratos inteligentes, vulnerabilidade, privacidade, segurança.

ABSTRACT

This study focuses on a critical analysis of challenges and solution strategies related to security and smart contracts within the blockchain ecosystem. I will explore potential vulnerabilities and threats to integrity by examining practical use cases. The research aims to identify effective strategies for risk mitigation, emphasizing technological best practices that foster secure implementation of smart contracts. By addressing these topics, my goal is to provide insights for a deeper understanding and improvement within the context of smart contracts and their implementation in the blockchain.

Keywords: blochchain, smart contracts, vulnerability , privacy, security.

LISTA DE FIGURAS

LISTA DE SIGLAS

BTC – Bitcoin

BC – Blockchain

SM – Smart Contract

ZKP – Zero-Knowledge Proof

DAAPS – Aplicações Descentralizadas

DAOS – Organizações Autônomas Descentralizadas

GGA – Gas Griefing Attack

SUMÁRIO

1. INTRODUÇÃO

- Introdução e Contextualização
  - Definição e Conceitos Básicos de Blockchain
  - História e Evolução da Tecnologia Blockchain
  - Princípios Fundamentais da Blockchain
  - Smart Contracts: Fundamentos e Definições
  - Funcionamento e Aplicações Práticas
- Contratos Inteligentes e Privacidade:
  - Características dos Contratos Inteligentes
  - Privacidade em Contratos Inteligentes
- Riscos de Segurança em Contratos Inteligentes:
  - Ataques e Fraudes em Contratos Inteligentes
  - Exemplo de ataques bem sucedidos
  - Vulnerabilidade Comum
  - Gerenciamento de Risco em Smart Contracts
  - Práticas Recomendadas para Mitigação de Risco

1.1 Definição e Conceitos Básicos de Blockchain

A blockchain representa uma forma de estrutura de dados em cadeia, onde cada bloco recém-criado é incorporado à cadeia existente, aumentando assim o tamanho global da rede. Cada bloco contém dados e informações que estão intrinsecamente interligados, tornando qualquer tentativa de alteração ou exclusão de um bloco existente inviável, uma vez que tal modificação comprometeria a validação por toda a cadeia (EXAME, 2022).

Embora teoricamente exista a possibilidade de modificar informações dentro da blockchain, tal feito exigiria um poder computacional exorbitante em um intervalo de tempo incrivelmente curto. Esse desafio, por sua vez, torna-se uma tarefa não apenas complexa, mas também economicamente custosa de ser efetuada. À medida que mais blocos são adicionados à cadeia ao longo do tempo, a rede se expande,

elevando consideravelmente o nível de complexidade. Isso resulta em uma barreira crescente para qualquer tentativa de influenciar a rede como um todo, conferindo à blockchain uma robustez e segurança notáveis.

A criptografia SHA-256, gera um hash com base, nas informações do bloco que acaba retornando um valor único por meio de uma operação criptografica. As funções hashs são muito utilizadas no quesito blockchain, a fim de adicionar segurança e imutabilidade (PASTOR, 2023).

1.1.1 História e Evolução da Tecnologia Blockchain

Esta tecnologia teve sua origem na década de 90 como um banco de dados público e imutável. Somente em 1992, a criptografia foi incorporada ao seu funcionamento, elevando ainda mais a segurança e a integridade dos dados. No entanto, foi somente em 2008, com a criação do Bitcoin, que a blockchain encontrou seu primeiro uso prático e ganhou notoriedade como uma inovação revolucionária no campo da tecnologia da informação (EXAME, 2022).

Essa abordagem destaca a evolução da blockchain ao longo do tempo, enfatizando seus fundamentos iniciais, os aprimoramentos introduzidos com a criptografia e seu marco significativo com o advento do Bitcoin em 2008.

1.1.2 Princípios Fundamentais da Blockchain

- Descentralização
- Consenso comum
- Trasnparência
- Segurança
- Imutabilidade
- Smart Contract

Descentralização implica na dispersão do poder central de uma autoridade, uma abordagem notável na tecnologia blockchain. Ao invés de concentrar todos os dados em um único servidor, a blockchain distribui essas informações entre os nós da

rede, constituídos por diversos computadores que mantêm cópias sincronizadas da blockchain.

O consenso, elemento fundamental nesse contexto descentralizado, exige que qualquer informação passe por uma prova de concordância antes de ser adicionada à rede. Qualquer desvio em relação à expectativa impede a adição da informação, garantindo a integridade e confiabilidade do sistema.

A transparência é um princípio vital em blockchains públicas. Todas as transações e modificações são visíveis para todos os participantes da rede, estabelecendo um ambiente transparente que promove a confiança entre os usuários.

A segurança emerge como um catalisador fundamental para o crescente acolhimento da tecnologia, sendo a complexidade intrínseca dos algoritmos de hashing um fator preponderante. Atualmente, a violação destes algoritmos revela-se uma tarefa formidável, sobretudo diante da distância que os computadores quânticos modernos ainda necessitam percorrer para alcançarem tal proeza. Adicionalmente, a estratégia de reforçar a segurança por meio do aumento da quantidade de bits na criptografia se configura como uma abordagem eficaz para tornar esta missão ainda mais improvável. Nesse contexto, propõe-se a transição do tradicional SHA-256 para o robusto SHA-512 como medida de fortalecimento da segurança (JOBIM, 2022).

A imutabilidade evita que os dados sejam modificados ou excluídos na rede. Esta característica resulta da combinação do consenso coletivo de segurança criptográfica. Esta união cria uma base sólida na qual a integridade dos dados é preservada com segurança. Imutabilidade, baseada no consenso e segurança criptográfica, confiabilidade e inviolabilidade dos registros, constituindo assim sistemas transparentes e segur

Os Smart Contracts, por sua vez, são como agentes autônomos na rede. São algoritmos programados para aguardar condições específicas antes de executarem suas funções. Ao serem implantados na rede, esses contratos inteligentes permanecem em estado de prontidão, sendo ativados automaticamente quando uma interação específica ocorre, desencadeando funcionalidades previamente programadas (MONEY, 2022).

1.2 Smart Contracts: Fundamentos e Definições

Os smart contracts são programas de computador autônomos e autoexecutáveis, incorporados em blockchain, que automatizam e garantem a execução de acordos sem a necessidade de uma autoridade central. Esses contratos inteligentes são

altamente versáteis e podem ser aplicados em diversas áreas. Embora seu uso seja mais evidente na transferência de valor, propriedade e informações, sua aplicabilidade vai muito além dessas funções essenciais.

1.2.1 Origens e aplicações práticas

O termo "smart contracts"foi cunhado por Nick Szabo em 1993. Szabo, um cientista da computação, propôs a ideia com o objetivo de introduzir práticas avançadas nos protocolos de comércio eletrônico entre indivíduos na internet, eliminando a necessidade de um intermediário central de confiança (ARAUJO, 2022).

A natureza programável e versátil dos smart contracts permite a implementação efetiva em uma variedade de cenários, incluindo jogos, votações, leilões, apostas, empréstimos e várias outras aplicações.

Atualmente, os smart contracts estão desempenhando um papel fundamental na transformação de setores, proporcionando eficiência, transparência e segurança. Sua capacidade de automatizar a execução de acordos, sem depender de intermediários, tem atraído atenção e adoção crescentes em várias indústrias.

1.2.2 Desenvolvimento em Smart Contract

Aplicações Descentralizadas (DAAPS) e Organizações Autônomas Descentralizadas (DAOs) são predominantemente aplicações baseadas na web que surgem do desenvolvimento de smart contracts.

A imutabilidade da blockchain, que impede a alteração de dados uma vez registrados, representa uma característica fundamental desses contratos inteligentes. Isso significa que, uma vez implementado na blockchain, um smart contract não pode ser modificado, são . No entanto, existem métodos programáveis que possibilitam a atualização de smart contracts por meio de um mecanismo conhecido como "smart contract adapter"(DUARTE, 2023a). Esse adaptador aponta para o endereço do smart contract que será utilizado, permitindo a extensão da mesma interface sem a necessidade de alterar diretamente o contrato existente.

Este processo de atualização proporciona flexibilidade e escalabilidade, pois permite que novas funcionalidades sejam incorporadas ou ajustes sejam feitos sem comprometer a integridade da blockchain. Essa abordagem de atualização programável, combinada com a natureza descentralizada das aplicações baseadas em block-

chain, destaca a capacidade única desses sistemas de evoluir e se adaptar ao longo do tempo, sem depender de uma autoridade central. Isso reflete a inovação contínua no campo das tecnologias descentralizadas e ilustra como a combinação de contratos inteligentes e blockchain está moldando o futuro das aplicações na web.

2. CONTRATOS INTELIGENTES E PRIVACIDADE

Com base em (DUARTE, 2023b), a consideração sobre modificadores de acesso é frequentemente negligênciada por desenvolvedores de contratos inteligentes. Em contratos inteligentes, o encapsulamento não age como uma medida de segurança efetiva, dado que na blockchain, o acesso à informação é praticamente público para todos os participantes da rede. Embora os modificadores de acesso ajudem a organizar o código e a gerenciar o acesso entre contratos, é crucial compreender que, em blockchains públicas, a visibilidade das informações não é efetivamente controlada por esses modificadores.

O equívoco comum reside na crença de que marcar uma variável como privada a torna segura. No entanto, ao implantar um contrato na blockchain, esse código e todas as suas variáveis associadas são propagados por toda a rede. Isso significa que qualquer indivíduo que possua uma cópia do blockchain terá acesso às informações, independentemente do modificador de acesso.

Em decorrência disso, armazenar dados sigilosos ou delicados em variáveis privadas na blockchain é uma prática arriscada. Torna-se essencial adotar estratégias adicionais, como a criptografia e técnicas de ofuscação, para proteger informações sensíveis, reconhecendo que a simples marcação como "privado"não proporciona o nível de segurança desejado em um ambiente descentralizado e transparente como o das blockchains públicas.

2.1 Característica dos Contratos Inteligentes

Conforme discutido anteriormente, nos sistemas de blockchains públicas tradicionais, a natureza distribuída e pública dessas plataformas pode impactar a integridade e a privacidade dos dados. Essa consideração é relevante também neste contexto.

Em blockchains públicas, a visibilidade generalizada dos dados representa um desafio para a confidencialidade, pois qualquer pessoa pode acessar e verificar todas as transações registradas. No entanto, ao empregar tecnologias avançadas como a Zero-Knowledge Proof (Prova de Conhecimento Zero - ZKP), é possível mitigar esses desafios.

A implementação bem-sucedida de ZKP em sistemas blockchain públicos oferece uma solução eficaz para preservar a privacidade, permitindo que informações

sensíveis permaneçam ocultas, mesmo em uma infraestrutura distribuída. Isso não apenas aprimora a segurança, mas também mantém a confiança dos usuários, promovendo a aceitação generalizada dessas plataformas.

2.1.1 Desafios de Privacidade em Contratos Inteligentes

Prova de Conhecimento zero: Com os avanços contínuos na tecnologia, a aplicação de ZKP tem se destacado como uma abordagem significativa para aprimorar a privacidade de dados durante transações em blockchain. A ZKP oferece a capacidade de preservar a confidencialidade de detalhes específicos, mesmo quando os dados da transação são registrados na blockchain.

Em ambientes descentralizados, como contratos inteligentes em decentralized applications (dApps), a implementação da prova de conhecimento zero emerge como uma medida crucial para fortalecer a segurança e atratividade dessas aplicações, especialmente aquelas que lidam com informações sensíveis. Ao incorporar ZKP nos contratos inteligentes, é possível assegurar que, mesmo quando os dados da transação são adicionados à blockchain, certos elementos permaneçam estritamente confidenciais.

Essa abordagem não apenas eleva os padrões de privacidade, mas também posiciona as dApps como soluções mais confiáveis para usuários preocupados com a segurança de seus dados. A capacidade de realizar transações sem divulgar informações específicas não apenas atende às demandas de privacidade, mas também impulsiona a confiança dos usuários em ambientes descentralizados (GUPTA S. TANWAR; KIM, 2020).

Armazenamento em Banco de Dados: De fato, a estratégia de armazenar dados sigilosos em um banco de dados, em vez de registrar tudo na blockchain, é uma abordagem alternativa que aborda preocupações de privacidade, mas ao custo da descentralização. Essa consideração também se aplica aqui.

Ao optar por um banco de dados centralizado para armazenar informações confidenciais, a descentralização inerente às blockchains é comprometida. A dependência de um servidor centralizado introduz um ponto único de falha e potenciais vulnerabilidades de segurança, pois um ataque ou falha nesse servidor pode comprometer todos os dados sensíveis.

Embora essa abordagem possa ser eficaz para proteger a privacidade dos dados, ela entra em conflito com os princípios fundamentais da descentralização que muitos sistemas blockchain visam alcançar. A centralização também implica maior

controle e responsabilidade nas mãos da entidade que gerencia o servidor, o que pode gerar desconfiança por parte dos usuários.

Ao avaliar as opções entre descentralização e proteção de dados sigilosos, é fundamental encontrar um equilíbrio que atenda aos requisitos específicos da aplicação.

3. RISCOS DE SEGURANÇA EM CONTRATOS INTELIGENTES

A presente seção visa fornecer uma análise aprofundada das diversas formas de ataque que podem ser direcionadas a smart contracts, destacando as vulnerabilidades que permeiam esse ecossistema complexo (SAYEED; CAIRA, 2020). À medida que as implementações de contratos inteligentes se tornam uma peça fundamental em uma variedade de setores, compreender as possíveis brechas de segurança torna-se essencial para desenvolvedores, auditores e usuários.

Ao empregar smart contracts em uma blockchain, é imperativo reconhecer que uma vez implantados, esses contratos não podem ser atualizados ou modificados. Diante dessa imutabilidade, os patches de segurança da rede desempenham um papel crucial, servindo como uma salvaguarda vital contra potenciais vulnerabilidades.

Essa característica singular da blockchain impõe uma responsabilidade significativa sobre os desenvolvedores, incentivando-os a adotar uma estratégia de segurança robusta antes de lançar suas aplicações. Afinal, qualquer falha identificada após a implementação pode acarretar consequências potencialmente irreparáveis para o funcionamento da aplicação.

3.1 Ataques e Fraudes em Contratos Inteligentes

A crescente popularidade das aplicações de finanças descentralizadas (DeFi) na blockchain tem proporcionado um ambiente inovador para a criação e desenvolvimento de serviços financeiros autônomos. No entanto, à medida que novas plataformas DeFi são lançadas e atraem fundos de usuários, torna-se imperativo abordar os desafios significativos associados à segurança e proteção contra ataques.

3.1.1 Exemplo de ataques bem sucedidos

Os dados foram obtidos em (DUARTE, 2023c).

- The DAO Attack 2016 ($60M)
- BurgerSwap Attack 2021 ($7.2M)
- Lendf.me Attack 2020 ($25M)

- XSurge Attack 2021 ($4M)
- Cream Finance Attack 2021 ($18.8M)
- Siren Protocol Attack 2021 ($3.5M)

3.1.2 Vulnerabilidades Comuns

Reentrancy Attack (DUARTE, 2023c): Em um contexto de pesquisa acadêmica, é relevante destacar que na blockchain, os endereços de carteira e de smart contract têm a capacidade de receber e enviar criptomoedas. Isso permite a interação com smart contracts usando tanto uma carteira quanto outro smart contract. Este cenário é explorado por invasores, como descrito no exemplo a seguir, que visa elucidar um ataque de reentrancy.

```solidity
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.18;

contract Carteira {
    mapping(address => uint) public balances; //user => balance

    constructor() {}

    function deposit() external payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw(uint amount) external {
        require(balances[msg.sender] >= amount, "Insufficient funds");
        payable(msg.sender).transfer(amount);
        balances[msg.sender] -= amount;
    }
}
```

Figura 3.1 – Contrato da Vítima

```solidity
pragma solidity ^0.8.18;

interface IReentrancy {
    function deposit() external payable;

    function withdraw(uint amount) external;
}

contract RA {
    IReentrancy private immutable target;

    constructor(address targetAddress) {
        target = IReentrancy(targetAddress);
    }

    function attack() external payable {
        target.deposit{value: msg.value}();
        target.withdraw(msg.value);
    }

    receive() external payable {
        if (address(target).balance >= msg.value) target.withdraw(msg.value);
    }
}
```

Figura 3.2 – Contrato do Atacante

Em um cenário hipotético, um invasor cria um smart contract, embora de forma mais sofisticada. O exemplo apresentado é simplificado, evitando a divulgação de ferramentas maliciosas. O contrato malicioso é então publicado na blockchain, identificando o endereço do contrato vulnerável.

Após o deploy do contrato malicioso, o atacante aciona a função "attack", realizando um depósito inicial no contrato alvo. Em seguida, ele invoca a função de saque do protocolo vítima, retirando a mesma quantidade recebida no depósito. Quando um smart contract recebe um saldo, a função nativa chamada "receive"é acionada. Embora, por padrão, essa função não execute ações, pode ser adaptada. No interior da função "receive", o invasor insere um código para realizar um novo saque, interrompendo a etapa de atualização do saldo do contrato vítima. Esse ajuste de balanço ocorre após o envio da transação, criando um loop até que o saldo do contrato seja exaurido. Em seguida, o invasor transfere o saldo final para outra conta, concluindo assim o ataque de reentrancy.

Gas Griefing Attack (DUARTE, 2023b): Nos smart contracts de blockchain pública toda escrita é uma transação, e é exigido que toda transação pague uma taxa de gás- as taxas de gás sustentam a rede remunerando os validadores da rede-. Essa técnica consiste em fazer o contrato cobrar uma taxa absurda ao ponto de zerar o saldo da carteira ou inviabilizar a lógica de negócio do contrato. Os contratos sucetiveis a esse ataque devem ser contratos que transferem dinheiro para outro endereço e disparam a função parão receive. E em contratos que o atacante gostaria de interromper o seu contrato.

```solidity
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

contract Auction {
    address public highestBidder;
    uint256 public highestBid;
    uint256 public auctionEnd;

    constructor(){
        auctionEnd = block.timestamp + (7 * 24 * 60 * 60);//7 days in the future
    }

    function bid() external payable {
        require(msg.value > highestBid, "Bid is not high enough");
        require(block.timestamp <= auctionEnd, "Auction finished");

        //refund the previous highest bidder
        if (highestBidder != address(0)) {
            (bool success, ) = highestBidder.call{value: highestBid}("");
            require(success, "refund failed");
        }

        highestBidder = msg.sender;
        highestBid = msg.value;
    }
}

```

Figura 3.3 – Contrato de Leilão

Este contrato de leilão foi concebido para operar ao longo de um período de 7 dias, apresentando uma função fundamental para dar lances. Quando um lance é submetido, o contrato realiza uma verificação para determinar se o novo lance supera

o valor do lance máximo atual. Em caso afirmativo, ocorre uma atualização nas informações, resultando na alteração do vencedor do leilão para o usuário que ofereceu o novo lance. Além disso, os fundos são restituídos ao antigo vencedor do leilão.

Essa lógica de leilão busca promover a competição entre os participantes, garantindo que o usuário com o lance mais alto ocupe a posição de vencedor. A implementação de tal mecanismo proporciona dinamismo ao leilão, criando um ambiente em que os participantes podem ajustar seus lances para manter ou conquistar a liderança no decorrer dos 7 dias estabelecidos.

```
1  pragma solidity ^0.8.18;
2
3  interface IAuction {
4      function bid() external payable;
5  }
6
7  contract GCA {
8
9      function attack(address _auction) external payable {
10         IAuction(_auction).bid{value: msg.value}();
11     }
12
13     receive() external payable{
14         keccak256("Just wasting some gas...")
15         keccak256("Just wasting some gas...")
16         keccak256("Just wasting some gas...")
17         keccak256("Just wasting some gas...")
18         keccak256("Just wasting some gas...")
19         //etc...
20
21     }
22 }
```

Figura 3.4 – Contrato do Atacante

O contrato do atacante opera em consonância com o contrato do leilão, submetendo uma oferta superior à do atual líder. Quando outro participante oferece um lance superior ao do atacante, a função "receive"do contrato do atacante é invocada. Se o código presente em "receive"for extenso ou complexo, isso pode acarretar em aumento das taxas devido à alocação de memória, podendo até ultrapassar o limite permitido para uma transação.

Quando um desses dois obstáculos ocorre, o contrato inteligente entra em um estado de erro, revertendo a transação original e mantendo o atacante como vencedor do leilão. Importante notar que o usuário que acionou a função enfrentará a perda de uma considerável quantia de gás da carteira, e em alguns casos, todo o valor destinado ao pagamento das taxas de transação.

Essa estratégia revela não apenas a manipulação do resultado do leilão pelo atacante, mas também destaca o potencial impacto financeiro adverso para quem efetuou a chamada da função. É crucial, portanto, que desenvolvedores e usuários estejam cientes dessas dinâmicas ao interagir com contratos inteligentes, priorizando

a segurança e compreendendo as possíveis ramificações econômicas de suas ações na blockchain.

3.2 Gerenciamento de Risco em Contratos Inteligentes

3.2.1 Práticas Recomendadas para Mitigação de Riscos

Reentrancy Attack (DUARTE, 2023c): Sabendo a forma que um reentracy attack básico funciona, vamos ver duas principais observações que devem ser feitas para não cair em um ataque desse tipo.

```
pragma solidity ^0.8.18;

contract Carteira {
    mapping(address => uint) public balances; //user => balance

    constructor() {}

    function deposit() external payable {
        balances[msg.sender] += msg.value;
    }

    bool private isProcessing = false;

    function withdraw(uint amount) external {
        require(!isProcessing,"Reentry Blocked");
        isProcessing = true;

        require(balances[msg.sender] >= amount, "Insufficient funds");
        balances[msg.sender] -= amount;
        payable(msg.sender).transfer(amount);

        isProcessing = false;
    }
}
```

Figura 3.5 – Solução Reentrancy

É imperativo sempre manter a ordem das operações, dando prioridade à execução da atualização do saldo antes de qualquer transferência na função de saque. Ao organizar as etapas dessa maneira, evita-se a vulnerabilidade do código a ataques de reentrada com loops infinitos.

Outra medida eficaz para prevenir tais ataques na função de saque é a implementação de uma variável de controle dedicada. Essa abordagem, embora simples, proporciona uma camada adicional de segurança e pode ser aprimorada para uma implementação mais robusta. A lógica consiste em verificar e bloquear a possibilidade de reentrância na função de saque, impedindo potenciais explorações maliciosas.

Essas práticas, quando incorporadas, contribuem significativamente para mitigar a maioria dos ataques de reentrada, fortalecendo a segurança do protocolo.

Gas Griefing Attack (DUARTE, 2023b): Diferentemente de alguns outros ataques a contratos inteligentes, esse tipo de ataque não possui uma solução simples, tornando-se assim uma ameaça significativa.

Em um cenário ideal, uma solução seria a atualização das linguagens de programação de contratos inteligentes para barrar sistemas como loops infinitos nas máquinas que executam os códigos.

Uma proposta mais radical seria alterar completamente o fluxo do contrato inteligente. Em vez de devolver imediatamente o dinheiro dos usuários quando alguém ultrapassa a oferta do atacante, o contrato poderia ser modificado para realizar a devolução apenas após o término do leilão. Essa alteração preservaria a integridade do contrato diante de ataques de GGA. No entanto, vale observar que uma mudança dessa natureza impactaria significativamente a lógica do contrato, exigindo adaptações no código para manter a equidade do leilão.

Por exemplo, se o dinheiro não fosse devolvido imediatamente quando alguém fizesse uma oferta maior, seria necessário implementar uma funcionalidade para permitir que o usuário com o dinheiro retido fizesse um novo lance utilizando os fundos bloqueados. Essas considerações destacam a complexidade de encontrar soluções eficazes que equilibrem a segurança do contrato com a manutenção da funcionalidade esperada.

Os autores do artigo (GUPTA S. TANWAR; KIM, 2020) propõem a utilização da ferramenta "Securify"como uma abordagem abrangente para identificar e compreender os problemas relacionados aos contratos inteligentes. Neste contexto, mais de 18 mil contratos inteligentes foram submetidos a testes, evidenciando a extensão e a diversidade do escopo de análise. A metodologia empregada pela "Securify"permite a identificação de vulnerabilidades, destacando se os contratos inteligentes estão suscetíveis às formas mais básicas de ataques nesse domínio.

Essa abordagem não apenas oferece uma visão holística sobre a segurança dos contratos inteligentes, mas também fornece uma avaliação prática e abrangente da resistência desses contratos a ameaças potenciais. Ao realizar testes em uma quantidade significativa de contratos, os autores visam contribuir para uma compreensão mais profunda e generalizada dos desafios de segurança enfrentados pelos contratos inteligentes na blockchain.

4. CRONOGRAMA

O desenvolvimento deste trabalho se dará da seguinte forma:

1. Elaboração da proposta de TC.
2. Estudo sobre o uso de dapps.
3. Estudo sobre o uso de daos.
4. A implementação da tecnologia ZKP.
5. Uso do Securify em SM.
6. Aplicação de banco de dados em dapps e daos.
7. Sistema de versionamento de SM.
8. Testes unitários e correções.
9. Escrita do TC II.

	2023					2024				
	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ
1										
2										
3										
4										
5										
6										
7										
8										
9										

REFERÊNCIAS BIBLIOGRÁFICAS

ARAUJO, J. P. *Entenda a maior evolução depois do Bitcoin, os contratos inteligentes!* 2022. Disponível em: <https://blog.vectorcrypto.com.br/o-que-e-contrato-inteligente/>. Acesso em: Dez. 2023.

DUARTE, L. *Como atualizar Smart Contracts em Solidity (Adapter)*. 2023. Disponível em: <https://www.luiztools.com.br/post/como-atualizar-smart-contracts-em-solidity/>. Acesso em: Dez. 2023.

DUARTE, L. *Gas Griefing Attack: Aulão de Segurança em Smart Contracts*. 2023. Dez. 2023. Disponível em: <https://www.youtube.com/watch?v=unxNQNdcdvA&ab_channel=LuizTools>.

DUARTE, L. *Reentrancy Attack em Smart Contracts Solidity*. 2023. Disponível em: <https://www.luiztools.com.br/post/reentrancy-attack-em-smart-contracts-solidity/>. Acesso em: Dez. 2023.

EXAME. *O que é blockchain e como funciona a tecnologia por trás do bitcoin?* 2022. Disponível em: <https://exame.com/invest/guia/o-que-e-blockchain-e-como-funciona-a-tecnologia-por-tras-do-bitcoin/>. Acesso em: Dez. 2023.

GUPTA S. TANWAR, F. A.-T. P. I. A. N. R.; KIM, S. W. Smart contract privacy protection using ai in cyber-physical systems: Tools, techniques and challenges. *IEEE Access*, v. 8, n. 24746-24772, p. 27, 2020. Disponível em: <https://ieeexplore.ieee.org/document/8976179>.

JOBIM, C. *Computação quântica será capaz de decifrar algoritmo do Bitcoin em alguns anos, revela estudo*. 2022. Disponível em: <https://br.cointelegraph.com/news/study-reveals-how-long-quantum-computers-will-be-able-to-crack-bitcoin>. Acesso em: Dez. 2023.

MONEY, I. *O que são smart contracts e qual a relação com criptomoedas*. 2022. Disponível em: <https://www.infomoney.com.br/guias/smart-contracts/>. Acesso em: Dez. 2023.

PASTOR, J. *O que é um hash?* 2023. Disponível em: <https://academy.bit2me.com/pt/que-es-hash/>. Acesso em: Dez. 2023.

SAYEED, H. M.-G. S.; CAIRA, T. Smart contract: Attacks and protections. *IEEE Access*, v. 8, n. 24416-24427, p. 12, 2020. Disponível em: <https://ieeexplore.ieee.org/document/8976179>.

ANEXO A – RAVitima.sol SM corrigido do ataque.

ANEXO B – RA.sol SM do atacante de reentrancy

ANEXO C – Auction.sol SM leilão atacado

ANEXO D – GGA.sol SM do atacante de GGA