Global Journal of Computer Science and Technology: D
Neural & Artificial Intelligence
Volume 23 Issue 2 Version 1.0 Year 2023
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Strengthening Smart Contracts: An AI-Driven Security Exploration

By M. Sai Mohan, T. L. N Swamy & V. Chandu Reddy
VIT-AP University

Abstract- Smart contracts are automated agreements in which the conditions between the purchaser and the vendor are encoded directly into lines of code, allowing them to execute automatically. Smart contracts have emerged as a ground-breaking technology, facilitating the decentralized and trustless execution of agreements on blockchain platforms. However, the widespread adoption of smart contracts exposes them to various security threats, leading to substantial financial losses and reputational harm. Artificial Intelligence has the capability to aid in the detection and reduction of vulnerabilities, thereby enhancing the overall strength and resilience of smart contracts. This integration can create highly secure and transparent systems that reduce the risk of fraud, corruption, and other malicious activities, thereby increasing trust and confidence in these systems and improving overall security. This research paper delves into the innovative applications of Artificial Intelligence techniques to enhance the security of smart contracts.

Keywords: *blockchain technology, artificial intelligence, smart contracts (SCs).*

GJCST-D Classification: *LCC Code: QA75.5-76.95*

Strictly as per the compliance and regulations of:

Strengthening Smart Contracts: An AI-Driven Security Exploration

M. Sai Mohan [α], T.L.N Swamy [σ] & V. Chandu Reddy [ρ]

Abstract- Smart contracts are automated agreements in which the conditions between the purchaser and the vendor are encoded directly into lines of code, allowing them to execute automatically. Smart contracts have emerged as a ground-breaking technology, facilitating the decentralized and trustless execution of agreements on blockchain platforms. However, the widespread adoption of smart contracts exposes them to various security threats, leading to substantial financial losses and reputational harm. Artificial Intelligence has the capability to aid in the detection and reduction of vulnerabilities, thereby enhancing the overall strength and resilience of smart contracts. This integration can create highly secure and transparent systems that reduce the risk of fraud, corruption, and other malicious activities, thereby increasing trust and confidence in these systems and improving overall security. This research paper delves into the innovative applications of Artificial Intelligence techniques to enhance the security of smart contracts. Investigating the potential of AI in detecting vulnerabilities, identifying potential attacks, and offering automated solutions for safer smart contracts will significantly contribute to the development and flawless execution of this emerging technology.

Keywords: blockchain technology, artificial intelligence, smart contracts (SCs)

I. Introduction

Some of the most popular blockchain platforms include Ethereum, Corda, EOS, and Tron. Each of these platforms has its own unique programming language for creating smart contracts. For example, Ethereum uses the contract-oriented programming language Solidity, which is designed to be as accessible as JavaScript. Hyperledger Fabric utilizes chain code, which can be written in Go, Java, or JavaScript. Corda, on the other hand, employs Kotlin, a language closely associated with Java, for its smart contract development. EOS and Tron utilize C++ and Solidity, respectively, for smart contract development. Different blockchain platforms employ a range of programming languages for smart contract development, but their common goal is to simplify the process of creating secure and efficient smart contracts that can operate on the blockchain.

Smart contracts are software programs stored on a blockchain, and they are designed to execute automatically when certain predefined conditions or events are fulfilled. By doing so, they eliminate the need for intermediaries and ensure that all participants involved can instantly verify the outcome. This automated process brings several advantages, such as faster execution, reduced delays, and increased certainty compared to traditional contract enforcement. The versatility of smart contracts allows them to automate various workflows and initiate subsequent actions based on predetermined conditions, making them an efficient and powerful tool for optimizing processes in a decentralized and trustless environment. However, alongside its disruptive benefits, this technology also faces significant challenges, with security being a top concern.

In this context, Artificial Intelligence emerges as a promising and innovative solution to bolster the security of smart contracts. Despite the immense potential of smart contracts, their vulnerabilities have been frequently exploited by malicious actors, resulting in catastrophic consequences. From the infamous DAO hack to various token thefts and vulnerabilities, the smart contract ecosystem has witnessed a string of security breaches that have undermined trust in the technology.

Whereas traditional security measures such as manual audits and code reviews have been employed to detect and mitigate smart contract vulnerabilities, these approaches are labour-intensive, time-consuming, They may still overlook certain types of threats. As the smart contract ecosystem continues to evolve and scale, there is an urgent need for more robust and efficient methods to enhance the security of smart contracts.

AI has the capability to analyze vast amounts of data, identify patterns, and detect potential vulnerabilities at a scale and speed that surpass human capabilities. By harnessing AI-powered tools and techniques, the smart contract development community can significantly enhance the detection and prevention of security flaws, thereby reinforcing the overall integrity of blockchain-based applications.

As we strive to unlock the full potential of blockchain technology, ensuring the security and trustworthiness of smart contracts remains an imperative task. By embracing the power of AI to fortify the security landscape, we can pave the way for a more robust and secure future for decentralized applications, fostering confidence and enabling further innovation in this exciting domain.

Author α σ ρ: VIT-AP University, Amaravati, Andhra Pradesh, India.
e-mail: msai45371@gmail.com

This paper explores the intersection of AI and smart contract security, diving into the numerous ways AI can be leveraged to mitigate risks and bolster the robustness of smart contracts. We will investigate AI applications across various phases of the smart contract development lifecycle, encompassing design, coding, auditing, and monitoring. Moreover, we will address the challenges and constraints linked to AI based security measures and propose potential avenues for future research in this rapidly advancing field.

II. Background on Smart Contracts

Smart contracts are automated agreements with their terms written directly into code, and they are utilized on various blockchain platforms. Ethereum is one of the pioneering platforms for smart contracts, hosting a vast ecosystem of decentralized applications (dApps) and using Solidity as its programming language. Binance Smart Chain (BSC) is a popular alternative, offering compatibility with Ethereum's Virtual Machine (EVM) for those seeking lower transaction fees. Cardano distinguishes itself in the blockchain space with a focus on sustainability, scalability, and secure smart contracts, using Plutus as its programming language. Polkadot offers a unique multi-chain environment, allowing custom blockchains with smart contract functionality to interact seamlessly. Other notable blockchain platforms supporting smart contracts include EOS, Tron, and Tezos. Each of these platforms caters to various application domains and contributes to the growth of decentralized applications and programmable financial ecosystems.

Smart contracts function through the use of simple 'if/when…then…' statements written in code on a blockchain. Once predefined conditions are verified, a network of computers executes the actions specified in the contract. These actions can encompass a variety of tasks, including releasing funds to designated parties, registering vehicles, sending notifications, or issuing tickets. The immutable nature of the blockchain guarantees that completed transactions cannot be altered, and access to the results is restricted to authorized parties only.

In a smart contract, participants can include numerous stipulations to ensure the satisfactory completion of the agreed-upon task. To establish the contract's terms, participants collaborate to determine how transactions and their associated data will be represented on the blockchain. They reach a consensus on the "if/when...then..." rules that govern these transactions, consider potential exceptions and devise a framework for dispute resolution.

Smart contracts are typically programmed by developers, but organizations are increasingly providing user-friendly tools such as templates, web interfaces, and online resources to simplify the process for businesses. These advancements aim to promote the broader adoption of blockchain technology across various industries. Within a smart contract, participants can incorporate multiple stipulations to ensure satisfactory task completion. Participants collaboratively decide on the contract's terms, including rules governing transactions, data representation on the blockchain, potential exceptions, and dispute resolution frameworks. This approach facilitates smart contract creation and encourages wider adoption across industries.

Smart contracts have revolutionized contract management by offering several key benefits. They operate on blockchain technology, ensuring enhanced security and trust. Once deployed, smart contracts become immutable, making them resistant to tampering and fraud. Their transparency on the blockchain fosters trust among parties, reducing the chances of disputes. The automation and efficiency of smart contracts streamline processes, eliminating the need for intermediaries and saving time and costs. Furthermore, their global accessibility simplifies cross-border transactions. Smart contracts execute with precision based on predefined conditions, guaranteeing the accurate fulfillment of contractual obligations. They have also fueled innovation by enabling decentralized applications across various industries. With these advantages, smart contracts are transforming contract execution and bringing significant improvements to business operations.

Smart contracts bring numerous advantages that have significantly transformed the traditional contract landscape and unlocked new possibilities across various industries. Operating on blockchain technology, these self-executing digital agreements offer benefits such as enhanced trust, security, transparency, efficiency, cost savings, global accessibility, accuracy, innovation, and eliminating intermediaries.

- Trust is a fundamental aspect of smart contracts, as they eliminate the need for intermediaries like banks or lawyers. The reliance on decentralized blockchain networks ensures that contract execution is guaranteed by the system's consensus mechanism, enhancing trust between parties involved in the agreement.
- The security offered by smart contracts is a critical factor in their adoption. Once deployed on the blockchain, these contracts become immutable, preventing unauthorized alterations or tampering. This cryptographic immutability ensures the integrity of the contract, making it resistant to fraud and unauthorized access.
- Transparency is inherent in blockchain technology, and smart contracts leverage this feature to provide a high level of transparency. All contract

transactions and codes are publicly recorded on the blockchain, enabling easy verification of the contract terms by anyone without the need for a third-party intermediary. This transparency fosters trust and reduces the likelihood of disputes arising from hidden clauses or undisclosed information.

- The accuracy of smart contracts is paramount. With precise coding, these contracts execute actions based on predetermined conditions, minimizing the potential for misinterpretation or human error. This precision ensures contractual obligations are fulfilled as intended, reducing the likelihood of disputes or breaches.

Smart contracts have diverse and impactful applications across various industries. In financial services, they streamline transactions, automate asset management, and facilitate decentralized finance (DeFi) applications. Supply chain management benefits from smart contracts by enhancing transparency, automating processes, and improving traceability. Real estate transactions become more efficient as smart contracts automate property transfers and manage rental agreements. Voting systems become more secure and transparent with smart contracts, ensuring the integrity of the voting process. In the realm of intellectual property, these contracts help manage copyrights and enforce fair compensation. Gaming industries leverage smart contracts to enable the creation and trade of non-fungible tokens (NFTs) for unique digital assets. healthcare benefits from secure patient record management and automated insurance claims processing. Legal and notary services are streamlined by smart contracts, reducing the need for intermediaries.

Additionally, smart contracts empower energy and utilities through peer-to-peer energy trading. Finally, decentralized governance relies on smart contracts for transparent decision making within decentralized autonomous organizations (DAOs). These applications demonstrate the wide-ranging impact of smart contracts in shaping various industries and creating more efficient, transparent, and secure processes.

Security breaches can have severe repercussions for the blockchain ecosystem. The immutability of smart contracts means that any vulnerabilities or exploits can lead to significant financial losses for users and investors. Beyond the economic impact, breaches also damage the reputation and trust in blockchain platforms and applications, hindering mainstream adoption. In extreme cases, security breaches can lead to contentious forks or hard forks in the blockchain, dividing the community and generating significant uncertainty. Regulatory scrutiny may increase, leading to potential stifling of innovation. Project failures, negative media coverage, and losses in DeFi and NFT sectors are also possible outcomes. To mitigate such breaches, developers must conduct thorough security audits, adhere to best practices, and employ formal verification tools. Collaboration among blockchain communities is essential to share knowledge and enhance security practices, safeguarding the integrity of the entire blockchain ecosystem.

a) Classification of Smart-Contract Vulnerabilities

i. *Reentrancy Attack*

A reentrancy attack represents a security vulnerability that can arise in smart contracts. It takes advantage of the asynchronous execution of smart contract functions, allowing an attacker to repeatedly call a contract function before the previous call finishes its execution. Consequently, this vulnerability can lead to unintended consequences, providing unauthorized access to funds or enabling manipulation of the contract's state. Safeguarding against re-entrancy attacks is essential to ensure the security and integrity of smart contracts and the associated decentralized applications.

Simplified bank smart contract is written in Solidity that contains a reentrancy vulnerability.

```
pragma solidity ^0.8.0;

contract SimpleBank {
    mapping(address => uint256) balances;
    bool public locked;

    constructor() {
        locked = true;
    }

    function deposit() public payable {
        require(!locked, "Contract is locked");
        balances[msg.sender] += msg.value;
    }

    function withdraw(uint256 amount) public {
        require(!locked, "Contract is locked");
        require(balances[msg.sender] >= amount, "Insufficient balance");
        (bool success, ) = msg.sender.call{value: amount}("");
        require(success, "Transfer failed");
        balances[msg.sender] -= amount;
    }

    function toggleLock() public {
        locked = !locked;
    }

    function getBalance() public view returns (uint256) {
        return balances[msg.sender];
    }
}
```

In this simplified bank smart contract, users can deposit and withdraw Ether from their accounts. The contract also has a function called 'toggleLock', which allows the contract owner to lock or unlock the contract to prevent further deposits and withdrawals.

```
contract MaliciousContract {
    SimpleBank public bank;

    constructor(address bankAddress) {
        bank = SimpleBank(bankAddress);
    }

    fallback() external payable {
        if (address(bank).balance >= 1 ether) {
            bank.withdraw(1 ether);
        }
    }

    function startAttack() public payable {
        // Initiate the attack by calling the fallback function.
        bank.withdraw(1 ether);
    }

    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

The re-entrancy vulnerability exists in the 'withdraw' function. Here's how it can be exploited.

1. An attacker deploys a malicious contract with a `fallback` function that performs a reentrant call to the `withdraw` function of the 'SimpleBank' contract
2. The attacker then calls the 'startAttack' function of the 'MaliciousContract,' initiating the reentrancy attack.
3. The `withdraw` function of the 'SimpleBank' contract transfers '1' ether to the attacker's contract. However, before the `withdraw` function completes its execution, the fallback function of the attacker's contract is triggered again due to the reentrancy call.
4. The re-entrant fallback function continues to call the `withdraw` function of the `SimpleBank` contract, resulting in multiple withdrawals of 1 ether each, even though the attacker's balance in the `SimpleBank` contract is already zero.

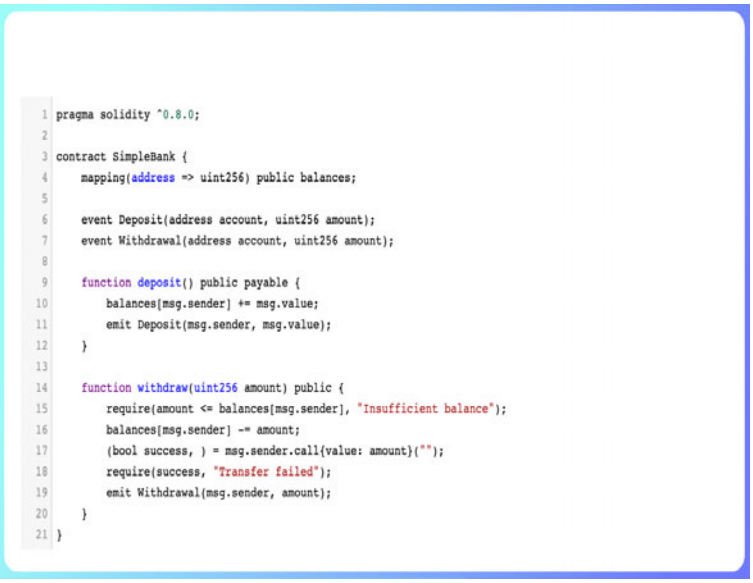
Prevention

Artificial Intelligence (AI) holds the potential to enhance the security of smart contracts by identifying and mitigating reentrancy vulnerabilities. One approach involves utilizing AI-powered tools to detect such vulnerabilities in smart contracts at the EVM bytecode level. Researchers have examined a large dataset of real-world smart contracts, allowing them to identify patterns of false positives and design effective path filters to eliminate them. Another approach employs AI-based fuzz testing to automatically generate inputs that simulate attacks on smart contracts. Subsequently, the execution logs are analyzed to determine the presence and intent of any re-entrancy processes. These AI-driven methods contribute to improving the accuracy and efficiency of detecting and preventing reentrancy vulnerabilities in smart contracts.

ii. *Overflow and Underflow*

Integer overflow and underflow vulnerabilities pose significant risks in the realm of blockchain-based applications, especially concerning smart contracts responsible for managing value transfers and storing sensitive data. These specific vulnerabilities are a subset of the general vulnerabilities we previously addressed. Smart contracts, being self-executing agreements governed by code, find widespread deployment on blockchain platforms such as Ethereum.

A simplified bank smart contract implemented in Solidity with potential integer overflow and underflow vulnerabilities.



```
pragma solidity ^0.8.0;

contract SimpleBank {
    mapping(address => uint256) public balances;

    event Deposit(address account, uint256 amount);
    event Withdrawal(address account, uint256 amount);

    function deposit() public payable {
        balances[msg.sender] += msg.value;
        emit Deposit(msg.sender, msg.value);
    }

    function withdraw(uint256 amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance");
        balances[msg.sender] -= amount;
        (bool success, ) = msg.sender.call{value: amount}("");
        require(success, "Transfer failed");
        emit Withdrawal(msg.sender, amount);
    }
}
```

1. The contract 'SimpleBank' allows users to deposit and withdraw Ether (the native currency of the Ethereum blockchain).
2. The 'balances' mapping stores the balance of each account. When an account deposits Ether, its balance is increased; when it withdraws, the balance is decreased.
3. The 'deposit' function allows users to deposit funds into their account.
4. The 'withdraw' function allows users to withdraws a specified amount of funds from their account. It first checks whether the user has enough balance to withdraw the requested amount before transferring the funds.

Integer Overflow

The 'balances' mapping uses the 'uint256' data type, which has a maximum value of 2^256 - 1. If a user deposits a large enough amount, it could cause an integer overflow when adding to their current balance. This would wrap the balance back to zero and effectively allow the user to withdraw the entire contract balance.

For example, if an account with a balance of 'balances [msg.sender] = 2^256 - 2' tries to deposit 3, the balance will become 1 (due to overflow) instead of the expected value of '2^256 - 2 + 3'.

Integer Underflow

The 'balances' mapping is using the 'uint256' data type, which cannot represent negative values. If a user tries to withdraw more funds than they have, it could cause an integer underflow. In Solidity, underflow on a 'uint256' wraps the value to its maximum value (2^256 - 1).

For example, if an account with a balance of 'balances [msg.sender] = 100' tries to withdraw 200, the 'require' statement will pass because 'amount <= balances [msg.sender]' evaluates to 'false' (since 200 is not less than or equal to 100), and the subtraction operation 'balances [msg.sender] -= amount' will wrap around to the maximum value of 'uint25', i.e., '2^256 - 1'.

Prevention

To mitigate these vulnerabilities, you can use safe math libraries like OpenZeppelin's SafeMath or, starting from Solidity version 0.8.0, use the built-in 'checked' arithmetic operations (e.g., 'a + b', 'a - b', 'a * b', and 'a / b') which automatically revert on overflow/underflow.

iii. *Denial of Service (DoS) attack*

DoS attacks directed at smart contracts represent a significant security threat. In these attacks, malicious individuals aim to disrupt the regular operation of the smart contract intentionally. The main goal is to render the smart contract unavailable to legitimate users, either temporarily or permanently. Such attacks can cause severe consequences, including the disruption of critical functionalities, suspension of contract execution, and depletion of resources. Ultimately, this leads to financial losses and disturbances in decentralized applications, making it a serious concern for the blockchain community.

iv. *Access Control Vulnerabilities*

Access control vulnerabilities in smart contracts refer to security flaws that arise when unauthorized users gain unintended access to certain functions, data, or funds within the contract. These vulnerabilities can have severe consequences, including loss of funds, unauthorized manipulation of critical contract logic, or unauthorized access to sensitive data.

v. *Timestamp Dependence Aulnerability*

Timestamp Dependence vulnerability refers to a security flaw in a smart contract where the contract's logic or behavior is influenced or manipulated by the

timestamp provided by blockchain miners while mining a new block. This vulnerability mainly affects blockchain platforms that include a timestamp as part of the block data, such as Ethereum.

vi. *Gas Griefing Attacks*

These attacks exploit the Gas payment mechanism in the Ethereum network. Gas serves as a unit of measurement for the computational resources required for transactions and smart contract executions. Hackers can employ these attacks to inflate the cost of executing smart contracts, resulting in prohibitively expensive transactions and trading.

vii. *Oracle Manipulation Attacks*

These attacks exploit vulnerabilities in smart contracts associated with oracles. Oracles are third-party services that provide real-world information for smart contracts. If hackers can manipulate the information provided by oracles, they can falsify smart contracts as fraudulent.

b) Real-world Incidents and Consequences

Smart contracts are computer records that are stored on the blockchain and can be used to transact. They are mostly used in decentralized finance (DeFi) and can be used to borrow and exchange cryptocurrencies. However, smart contracts are not immune to hacking. there have been a lot of promising smart hacks in recent years. Some of the most important are

i. *The DAO Hack*

The DAO attack was a significant security breach that occurred in June 2016. DAO, short for Decentralized Autonomous Organization, was a financial resource managed by the Ethereum community, raising over $150 million worth of ether (ETH) through a token sale. However, on June 17, 2016, hackers exploited a vulnerability in the DAO's code, withdrawing 3.6 million ETH, which was valued at around $70 million at the time. This attack triggered turmoil within the Ethereum community, sparking a debate between those who advocated for making it harder to recover stolen funds and those who argued that such actions would compromise the principles of blockchain evolution.

Finally, the Ethereum community has decided to challenge the blockchain. This resulted in two separate blockchains: Ethereum and Ethereum Classic. Ethereum Classic is the first blockchain without a hard fork. Ethereum is a forked blockchain that receives stolen funds. The DAO hack is a big problem for the Ethereum project. However, it also brings some improvements in smart contract security. Smart contracts are more secure today than they were in 2016.

ii. *Yearn Finance hack*

Yearn Finance is a DeFi platform that enables users to generate profits from their cryptocurrency investments through the use of smart contracts. However, on April 13, 2023, Yearn Finance experienced a security breach resulting in the loss of approximately $11.54 million worth of cryptocurrencies. The attackers exploited a vulnerability within yUSDT, a stable currency linked to the US dollar value of the Yearn Finance smart contract. yUSDT is created by depositing USDT into the Yearn Finance platform. The attackers took advantage of this vulnerability to deposit significant amounts of USDT on the platform and subsequently generated large quantities of yUSDT. They then utilized the yUSDT to purchase other tokens on the Yearn Finance platform, causing the tokens' values to increase and enabling the attackers to profit. The hackers managed to steal approximately $11.54 million worth of cryptocurrency before the vulnerabilities were addressed. This hack posed a significant challenge for the Yearn Finance project. However, the project's team has since taken measures to enhance platform security, including the identification of smart contract vulnerabilities and the implementation of new security measures.

iii. *Merlin Hack*

Merlin is a decentralized exchange (DEX) built on top of the zkSync layer 2 scaling solution, offering users the ability to exchange coins without incurring gas fees. However, on April 26, 2023, Merlin fell victim to a security breach in which approximately $1.8 million worth of cryptocurrency was stolen. The attackers exploited a vulnerability in the way Merlin's smart contracts managed liquid pools-collections of tokens used to facilitate DEX transactions. Merlin's smart contracts utilize a single pool for all traded tokens on the platform. Exploiting this vulnerability, hackers removed a substantial number of tokens from the liquid pool, causing their values to plummet. Subsequently, the attackers repurchased these tokens at a lower cost. They then sold the tokens back to the liquidity pool, ultimately profiting by approximately $1.8 million. The Merlin Hack posed a significant challenge for the Merlin project. Nevertheless, the project's team has taken steps to enhance platform security, including the use of multiple repositories and the implementation of new security measures.

iv. *Bonq Dao Exploit*

Bonq DAO is a decentralized autonomous organization (DAO) that facilitates cryptocurrency borrowing and lending through smart contracts to expedite the loan process. On February 1, 2023, Bonq DAO was launched with approximately $120 million worth of cryptocurrencies. However, it fell victim to a security breach when hackers exploited a vulnerability within its smart contract related to price feeds. Price feeds serve as real data sources utilized by smart contracts to determine asset values. In Bonq DAO's case, it relied on the Tellor oracle to obtain price information for the AllianceBlock (ALBT) token.

The attackers took advantage of this vulnerability by manipulating Tellor oracles to provide incorrect values for ALBT tokens. This allowed them to borrow significant amounts of BEUR stablecoins from the Bonq DAO platform at an exceptionally low cost. Subsequently, the hackers drained the pool of ALBT tokens used as collateral for BEUR loans. This action caused the value of the ALBT token to plummet, further reducing the cost of their borrowing. While the attackers ultimately repaid the BEUR loans, they retained the ALBT tokens. This incident resulted in a loss of approximately $120 million for the Bonq DAO platform.

The Bonq DAO vulnerability posed a substantial challenge for the project. Nevertheless, the team has taken measures to enhance platform security, including diversifying the use of different divination services and implementing new security measures

v. *Euler Finance Hack*

Euler Finance is a decentralized finance (DeFi) platform that facilitates cryptocurrency borrowing and lending through the use of smart contracts, streamlining the lending process. However, on March 13, 2023, Euler Finance experienced a security breach resulting in the loss of approximately $196.9 million worth of cryptocurrencies. The attackers exploited vulnerabilities within Euler Finance's smart contracts related to revenue management.

In Euler Finance, a 'call' is a notification requiring borrowers to add additional collateral to their loans. Failure to do so can result in the lender freezing the borrower's position. Hackers capitalized on this vulnerability by sending a large number of 'calls' to the Euler Finance smart contract, causing it to enter a state where it could no longer process any further calls. This effectively granted hackers access to the Euler Finance platform.

The Euler Finance hack represented a significant setback for the project. However, the team managed to recover the majority of the stolen funds.

III. Artificial Intelligence for Smart Contract Security

Artificial Intelligence (AI) plays a pivotal role in enhancing smart contract security by providing advanced tools and techniques to identify vulnerabilities, detect anomalies, and mitigate risks. This integration of smart contracts with AI has the potential to revolutionize various industries and domains, spanning from finance and healthcare to logistics and energy. By harnessing the combined power of smart contracts and AI, developers can create applications that are more efficient, secure, and autonomous, enabling innovative business models and services.

For instance, AI can enhance the adaptability of smart contracts by incorporating logic, neural graphs, and neural networks[2]. This fusion of technologies has the potential to significantly reduce the manpower required to manage both contracts and the entire contracting process, adding substantial value to organizations.

AI offers a wide array of applications within the realm of smart contracts. It can be directly integrated into smart contract code or utilized to validate and ensure contract integrity. Furthermore, the combination of AI techniques with deep learning concepts, such as Tensor, holds promise for advancing blockchain-based smart contracts. Additionally, cognitive computing, a subset of AI, aims to emulate human thought processes within computing infrastructure.

a) AI for Testing and Evaluation of Smart Contracts

AI can play a crucial role in testing smart contracts through various methods, encompassing performance testing, vulnerability detection, and correctness evaluation. By harnessing AI as a utility service for blockchain, the performance of blockchain-based smart contracts can be significantly enhanced, marking a substantial contribution of AI to the field of blockchain technology.

In a study by Marwala et al. [39], the utilization of AI for verifying smart contracts was discussed. The authors highlighted the potential advantages of applying AI to blockchain-based smart contracts, which include heightened security and scalability. Furthermore, they emphasized the feasibility of employing AI-based formal verification techniques to assess the correctness of smart contracts.

b) Federated Learning

Federated learning is an innovative approach to collaborative and decentralized learning, aligning well with the decentralization capabilities of blockchain technology. In this approach, training data remains secure and private, making it particularly valuable in scenarios involving sensitive information, such as healthcare data. By combining federated learning with blockchain, various functionalities, including data access control and enhanced privacy preservation, can be achieved.

In a study by Lu et al. [40], a novel privacy preservation mechanism for industrial IoT was proposed, leveraging a combination of federated learning and blockchain. They integrated federated learning into the consensus process, resulting in improved computing resource consumption and operational efficiency. However, challenges persist, particularly in addressing resource constraints within computing infrastructure, necessitating a deeper exploration of data privacy requirements.

In another work by Kang et al. [41], a federated learning system based on a consortium blockchain was presented. The authors introduced an incentive mechanism based on contract theory to evaluate workers with a high reputation for reliable training, thus

enhancing the learning process. Nevertheless, there is room for further improvement in the realm of reputation calculation.

In summary, the integration of federated learning and blockchain presents exciting research opportunities for enhancing privacy and efficiency across various domains, such as healthcare and industrial IoT. However, certain issues, particularly those related to resource constraints and reputation calculation, warrant additional attention and development.

c) Smart Contracts and Cognitive Computing

Cognitive computing represents an advanced field of AI research that aims to replicate human thinking within computer systems. By adopting human thinking patterns and limitations in its execution, cognitive computing achieves notably higher accuracy than other AI techniques. Integrating blockchain based smart contracts into cognitive computing can potentially enhance service values across various application scenarios.

Blockchain-based smart contracts bring essential features to the forefront within the realm of cognitive computing, including data transparency, decentralized access control capabilities, and decentralized trust. These attributes significantly enhance the applicability of cognitive computing in the healthcare domain. Nonetheless, as emphasized in a study by Daniel et al. [42], implementing blockchain for healthcare is a complex undertaking. It necessitates meticulous consideration of compliance requirements to ensure the utmost data privacy and security.

d) Smart Contracts with Tensor Networks

Smart contracts integrated with tensor networks present a compelling fusion of blockchain technology and quantum computing. Tensor networks, rooted in mathematical constructs from quantum physics, hold the promise of quantum-enhanced computing within smart contracts. This potential allows for the execution of more intricate calculations and simulations than classical computers can handle, offering transformative applications in data analysis, optimization, and cryptography within blockchain-based systems. Furthermore, tensor networks enable secure multiparty computations, facilitating collaborative efforts without compromising sensitive information. Promising areas for advancement encompass quantum machine learning, quantum randomness generation, and decentralized optimization. However, while this integration holds great promise, it faces challenges related to quantum hardware and scalability, demanding careful consideration for its full realization. Charlie et al. [43], in their research, contribute valuable insights and solutions to address some of these challenges, further advancing the field.

IV. Comparison of AI Based Smart-Contract Vulnerability Detection Tools

This table serves as a valuable reference, guiding readers to relevant materials for further exploration. It provides insights into the AI methods employed by each tool, ranging from supervised learning to reinforcement learning and semi-supervised learning. Additionally, the table offers information about dataset sizes used in the research, allowing readers to gauge the impact of data scale on model performance and reliability. Furthermore, the table outlines the AI classification approaches utilized by these tools, elucidating the distinctions between different methods.

By analyzing this comprehensive table, readers can gain a deeper understanding of various AI-based smart contract vulnerability detection tools. It serves as an indispensable resource for both further research and practical applications in this domain.

Table 1: Comparison of AI Based Smart-Contract Vulnerability Detection Tools

References	Classification	Dataset Size	Adopted Technique	Contribution
[1]	Supervised Learning	More than 50,932	DL, Modular and Systematic Vulnerability Detection Framework	DeeSCVHunter is a proposed deep learning-based framework for detecting vulnerabilities such as re entrancy and time dependence in a systematic and modular manner. It offers an innovative approach to identifying and addressing these types of vulnerabilities.
[2]	Supervised Learning	7000	LSTM, ANN, GRU	GRU, ANN, and LSTM, were trained and utilized to predict the presence of vulnerabilities in smart contracts. This approach offers a new way to identify and address potential vulnerabilities in a more efficient and effective manner.

[3]	Supervised Learning	47,398	Deep Learning	ReVulDL is a deep learning-based two-phase smart contract debugger for re-entrancy vulnerability. It integrates the vulnerability detection and localization into a unified debugging pipeline.
[4]	Semi-Supervised Learning	20,829	BERT	ASSBert is a model that uses active and semi-supervised learning with BERT for smart contract vulnerability detection. It aims to improve the accuracy and scalability of vulnerability detection by combining deep learning with expert patterns in an explainable fashion.
[5]	Reinforcement Learning	Not provided	Reinforcement Learning and Fuzzing	Vulnerability-guided fuzzer based on reinforcement learning, namely RLF, for generating vulnerable transaction sequences to detect sophisticated vulnerabilities in smart contracts. The experimental results demon-strate that RLF outperforms state-of-the-art vulnerability-detection tools.
[6]	Supervised Learning	40.932	GNN and Expert Knowledge	The use of graph neural networks and expert knowledge for smart contract vulnerability detection. Empirical results show significant accuracy improvements over state-of-the-art methods on three types of vulnerabilities.
[7]	Supervised Learning	70,000	Machine Learning	SmartMixModel is a machine learning-based vulnerability detection model for Solidity smart contracts. It considers an expanded feature space covering both the source- and byte codes of the Solidity smart contracts, and achieves improved detection performance compared to state of the art models.

V. Conclusion

In conclusion, there exists a compelling need for continued research into the application of artificial intelligence (AI) in the detection of flaws within smart contracts. This research seeks to offer invaluable insights through the comparative evaluation of existing AI-based algorithms for smart contract fault detection, shedding light on the efficacy of various AI approaches. While the potential of combining AI with formal methods has been acknowledged, there remains untapped potential in need of exploration.

Future research endeavors in the realm of smart contracts will pivot towards the development of AI powered detection tools capable of addressing security breaches associated with smart contracts while handling large datasets efficiently and effectively. Additionally, attention must be directed towards the utilization of SSL (Semi-Supervised Learning) and RL (Reinforcement Learning) to potentially overcome the limitations of SL (Supervised Learning). A comprehensive investigation into smart contract flaw detection using AI is imperative, serving as a foundational reference and a wellspring of inspiration for forthcoming research.

Ultimately, the integration of AI with formal techniques holds the promise of substantially enhancing the security of smart contracts, ensuring their reliability and robustness in blockchain-based applications

References Références Referencias

1. X. Yu, H. Zhao, B. Hou, Z. Ying and B. Wu, "DeeSCVHunter: A Deep Learning-Based Framework for Smart Contract Vulnerability Detection," 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 2021, pp. 1-8, doi:10. 1109/IJCNN52387.2021.9534324.
2. Gupta, R.; Patel, M.M.; Shukla, A.; Tanwar, S. Deep learning-based malicious smart contract detection scheme for internet of things environment. Comput. Electr. Eng. 2022, 97, 107583.
3. Zhuo Zhang, Yan Lei, Meng Yan, Yue Yu, Jiachi Chen, Shangwen Wang, and Xiaoguang Mao. 2023. Reentrancy Vulnerability Detection and Localization: A Deep Learning Based Two-phase Approach. In Proceedings of the 37th IEEE/ACM International

Conference on Automated Software Engineering (ASE '22). Association for Computing Machinery, New York, NY, USA, Article 83, 1–13. https://doi.org/10.1145/3551349.3560428.

4. Xiaobing Sun, Liangqiong Tu, Jiale Zhang, Jie Cai, Bin Li, and Yu Wang. 2023. ASSBert: Active and semi-supervised bert for smart contract vulnerability detection. J. Inf. Secur. Appl. 73, C (Mar 2023). https://doi.org/10.1016/j.jisa.2023.103423.
5. Jianzhong Su, Hong-Ning Dai, Lingjun Zhao, Zibin Zheng, and Xiapu Luo. 2023. Effectively Generating Vulnerable Transaction Sequences in Smart Contracts with Reinforcement Learning-guided Fuzzing. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22). Association for Computing Machinery, New York, NY, USA, Article 36, 1–12. https://doi.org/10.1145/3551349.3560429.
6. Z. Liu, P. Qian, X. Wang, Y. Zhuang, L. Qiu and X. Wang, "Combining Graph Neural Networks With Expert Knowledge for Smart Contract Vulnerability Detection," in IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 2, pp. 1296-1310, 1 Feb. 2023, doi: 10.1109/TKDE.2021.3095196.
7. S. Shakya, A. Mukherjee, R. Halder, A. Maiti and A. Chaturvedi, "SmartMixModel: Machine Learning-based Vulnerability Detection of Solidity Smart Contracts," 2022 IEEE International Conference on Blockchain (Blockchain), Espoo, Finland, 2022, pp. 37-44, doi: 10.1109/Blockchain55522.2022.00016.
8. Taherdoost, H. Smart Contracts in Blockchain Technology: A Critical Review. *Information* 2023, *14*, 117. https://doi.org/10.3390/info14020117.
9. S. K. Sood, A. K. Sarje and K. Singh, "An improvement of Wang et al.'s authentication scheme using smart cards," 2010 National Conference On Communications (NCC), Chennai, India, 2010, pp. 1-5, doi:10.1109/NCC.2010.5430153.
10. K. Wüst and A. Gervais, "Do You Need a Blockchain?" in 2018 Crypto Valley Conference on Blockchain Technology (CVCBT). IEEE, 2018, pp. 45–54.
11. C. D. Clack, V. A. Bakshi, and L. Braine, "Smart Contract Tem- plates: Essential Requirements and Design Options," arXiv preprint arXiv:1612.04496, 2016.
12. S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F.-Y. Wang, "An Overview of Smart Contract: Architecture, Applications, and Future Trends," in 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018, pp. 108– 113.
13. N. Reiff, "What Is ERC-20 and What Does It Mean for Ethereum?" 2020. [Online]. Available: https://www.investopedia.com/news/what-erc20-and-what-does-it-mean-ethereum/
14. Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," Future Generation Computer Systems, vol. 105, pp. 475–491, 2020.
15. Deep learning-based malicious smart contract detection scheme for internet of things environment. (2021, November 20). Deep Learning-based Malicious Smart Contract Detection Scheme for Internet of Things Environment - ScienceDirect. https://doi.org/10.1016/j.compeleceng.2021.107583
16. D. He, R. Wu, X. Li, S. Chan and M. Guizani, "Detection of Vulnerabilities of Blockchain Smart Contracts," in IEEE Internet of Things Journal, vol. 10, no. 14, pp. 12178-12185, 15 July15, 2023, doi: 10.1109/JIOT.2023.3241544.
17. Y. Huang, Y. Bian, R. Li, J. L. Zhao and P. Shi, "Smart Contract Security: A Software Lifecycle Perspective," in IEEE Access, vol. 7, pp. 150184-150202, 2019, doi: 10.1109/ACCESS.2019.2946988.
18. How Security Analysts Can Use AI in Cybersecurity. (2023, May 24). freeCodeCamp.org. https://www.freecodecamp.org/news/how-to-use-artificial-intelligence-in-cybersecurity/
19. Artificial Intelligence (AI) vs. Machine Learning. (n.d.). CU-CAI. https://ai.engineering.columbia.edu/ai-vs-machine-learning/
20. Sarker, I. H., Furhad, M. H., &Nowrozy, R. (2021, March 26). AI-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions - SN Computer Science. SpringerLink. https:// doi.org/10.1007/s42979-021-00557-0.
21. Deep reinforcement learning for blockchain in industrial IoT: A survey. (2021, March 18). Deep Reinforcement Learning for Blockchain in Industrial IoT: A Survey - ScienceDirect. https://doi.org/10.1016/j.comnet.2021.108004.
22. Fei, J., Chen, X., & Zhao, X. (2023, January 29). MSmart: Smart Contract Vulnerability Analysis and Improved Strategies Based on Smartcheck. MDPI. https://doi.org/10.3390/app13031733.
23. H., Xu, Y., Hu, G., You, L., & Cao, C. (2021, August 15). A Novel Machine Learning-Based Analysis Model for Smart Contract Vulnerability. A Novel Machine Learning-Based Analysis Model for Smart Contract Vulnerability. https://doi.org/10.1155/2021/5798033.
24. Liu, Z., Qian, P., Wang, X., Zhu, L., He, Q., & Ji, S. (2021, June 17). Smart Contract Vulnerability Detection: From Pure Neural Network to Interpretable Graph Feature and Expert Pattern Fusion. arXiv.org. https://arxiv.org/abs/2106.09282v1.
25. OWASP Smart Contract Top 10 | OWASP Foundation. (n.d.). OWASP Smart Contract Top 10 | OWASP Foundation. https://owasp.org/www-project-smart-contract-top-10/

26. A., & Bhaskar, J. (2023, February 22). A Complete List of Top 10 Smart Contract Platforms [2023]. Semidot Infotech. https://semidotinfotech.com/blog /top-smart-contract-platforms/
27. Smart contract applications within blockchain technology: A systematic mapping study. (2018, October 6). Smart Contract Applications within Blockchain Technology: A Systematic Mapping Study – Science Direct. https://doi.org/10.1016/ j.tele. 2018. 10.004.
28. Li, X., Xing, X., Wang, G., Li, P., & Liu, X. (2023, February 16). Detecting Unknown Vulnerabilities in Smart Contracts with Binary Classification Model Using Machine Learning. Detecting Unknown Vulnerabilities in Smart Contracts With Binary Classification Model Using Machine Learning | SpringerLink. https://doi.org/10.1007/978-981-99-0272-9_12
29. H., Han, D., Li, Q., Zhang, L., & Xu, T. (2023, February 3). A Smart Contract Vulnerability Detection Model Based on Syntactic and Semantic Fusion Learning. A Smart Contract Vulnerability Detection Model Based on Syntactic and Semantic Fusion Learning. https://doi.org/10.1155/2023/ 9212 269.
30. What are smart contracts on blockchain? | IBM. (n.d.). What Are Smart Contracts on Blockchain? | IBM. https://www.ibm.com/topics/smart-contracts
31. Hendrickson, L. (2023, July 17). What Is Blockchain and Why Does It Matter. Identity. https://www. identity.com/what-is-blockchain-and-why-does-it-matter/.
32. Emerging Trends in Blockchain Technology and Applications: A Review and Outlook. (2022, March 16). Emerging Trends in Blockchain Technology and Applications: A Review and Outlook - ScienceDirect. https://doi.org/10.1016/j.jksuci.2022.03.007.
33. What Is Binance Smart Chain? | CoinMarketCap. (n.d.). CoinMarketCap Alexandria. https://coin marketcap.com/alexandria/article/what-is-binance-smart -chain.
34. Cardano Docs. (n.d.). Cardano Docs. https://docs. cardano.org/Academy, S. (2023, May 14). Combining Smart Contracts with Artificial Intelligence in Solidity. Medium. https://medium. com/@solidity101/ combining-smart-contracts-with-artificial-intelligence -in-solidity-99f072d2fcdc.
35. Mahawar, S. (2022, March 10). AI in smart contracts: the magic combo - iPleaders. iPleaders. https:// blog.ipleaders.in/ai-smart-contracts-magic-combo/.
36. How AI Is Changing Contracts. (2018, February 12). Harvard Business Review. https://hbr.org/2018/02/ how-ai-is-changing-contracts.
37. T. M. Hewa, Y. Hu, M. Liyanage, S. S. Kanhare and M. Ylianttila, "Survey on Blockchain-Based Smart Contracts: Technical Aspects and Future Research," in IEEE Access, vol. 9, pp. 87643-87662, 2021, doi: 10.1109/ACCESS.2021.3068178.
38. Marwala, T., & Xing, B. (2018, February 13). Blockchain and Artificial Intelligence. arXiv.org. https:// arxiv.org/abs/1802.04451v2.
39. Y. Lu, X. Huang, Y. Dai, S. Maharjan and Y. Zhang, "Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT," in IEEE Transactions on Industrial Informatics, vol. 16, no. 6, pp. 4177-4186, June 2020, doi: 10.1109/ TII.2019. 2942190.
40. J. Kang, Z. Xiong, D. Niyato, S. Xie and J. Zhang, "Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory," in IEEE Internet of Things Journal, vol. 6, no. 6, pp. 10700-10714, Dec. 2019, doi:10.1109/JIOT.2019. 2940820.
41. Daniel, J., Sargolzaei, A., Abdelghani, M., Sargolzaei, S., & Amaba, B. (2017). Blockchain technology, cognitive computing, and healthcare innovations. *J. Adv. Inf. Technol*, *8* (3).
42. Charlier, J., Lagraa, S., & Francois, J. (2017, September). Profiling smart contracts interactions with tensor decomposition and graph mining. In European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML-PKDD)-Workshop on Mining DAta for financial applicationS (MIDAS).