International Journal of Interdisciplinary Innovative Research & Development (IJIIRD)
ISSN: 2456-236X
Vol. **11** Special Issue **02** | **2026**

ZKP-based Private Blockchain, Smart Contract Vulnerability Detection & Cross-chain Interoperability

Mr.Chetansing.S.Patil[1], Ms.Shailesh.S.Mali[2], Mr.Nikhil.R.Solanke[3], Mr.Gayatri.S..Patil[4], Prof. Madhuri.R.Chaudhari[5]
[1,2,3,4] *Student , Computer science and Engineering, Padm. Dr.V. B.Kolte College of Engineering, Malkapur*
[5]*Professor , Computer science and Engineering, Padm. Dr.V. B.Kolte College of Engineering, Malkapur*

DOI: 10.5281/zenodo.19286013

ABSTRACT

Blockchain's inherent immutability, while transfor mative, creates critical security risks in smart contracts, where undetected vulnerabilities can result in irreversible financial losses. Current auditing tools and approaches often address specific vulnerability types, yet there is a need for a compre hensive solution that can detect a wide range of vulnerabilities with high accuracy. We propose LLM-SmartAudit, a novel framework that leverages Large Language Models (LLMs) to automate smart contract vulnerability detection and analysis. Using a multi-agent conversational architecture with a buffer of-thought mechanism, LLM-SmartAudit maintains a dynamic record of insights generated throughout the audit process. This enables a collaborative system of specialized agents to iteratively refine their assessments, enhancing the accuracy and depth of vulnerability detection. To evaluate its effectiveness, LLM SmartAudit was tested on three datasets: a benchmark for common vulnerabilities, a real-world project corpus, and a CVE dataset. It outperformed existing tools with 98% accuracy on common vulnerabilities and demonstrates higher accuracy in real-world scenarios. Additionally, it successfully identifies 12 out of 13 CVEs, surpassing other LLM-based methods. These results demonstrate the effectiveness of multi-agent collaboration in automated smart contract auditing, offering a scalable, adaptive, and highly efficient solution for blockchain security analysis.

Keywords: - Smart contract auditing, LLMs, multi-agent, vulnerability detection..

1. INTRODUCTION

Recent advances in smart contracts have marked sig nificant progress in areas such as security, finance, and governance. These are self-executing contracts with terms mu tually agreed upon by participants, enacted through predefined actions. However, the immutable nature of blockchain technol ogy inherently makes smart contracts more severe to software attacks. Over the past years, there have been numerous high profile vulnerabilities and exploits in smart contracts, such as the DAO attack [1]. In 2024 alone, total losses exceeded 2.6 billion USD across 192 incidents, with a significant portion resulting from smart contract-level vulnerabilities [2]. Conse quently, the development of secure smart contracts continues to pose significant challenges. Research in Large Language Models (LLMs) has made sig nificant advancements in fields such as Natural Language Pro cessing [3], Computer Vision [4], Code Generation [5], and various AI tasks [6], [7]. A survey [8] indicates a significant increase in LLMs adoption over the past five years, accom panied by a rapid rise in software engineering. LLM tools are primarily categorized into commercial products and open source initiatives. Advanced commercial LLMs include GPT, Claude [9], and Gemini [10] models, while prominent open source projects feature Llama [11], Mixtral [12], and DeepSeek [13]. Both commercial and open-source LLMs demonstrate significant potential in handling complex tasks. GPT, one of the pioneering commercial LLMs, has exhibited remarkable profi ciency in natural language understanding, context processing, and code comprehension, outperforming numerous traditional methods [14].

2. LITERATURE REVIEW

Literature review related to exiting system/methodology

Smart contracts deployed on blockchain platforms like Ethereum are immutable and autonomous, making security vulnerabilities highly critical. Several approaches have been proposed for smart contract vulnerability detection, including static analysis, dynamic analysis, and machine learning-based techniques.Traditional static analysis tools

such as Oyente, Mythril, and Slither analyze smart contract bytecode or source code to detect vulnerabilities like reentrancy, integer overflow, and access control issues. However, these tools rely heavily on predefined rules and patterns, which limits their ability to detect novel or complex vulnerabilities. They also tend to generate a high number of false positives.

Sr. No.	Title of Paper	Review	Analysis/Limitations
[1]	Analysis of Smart Contract Vulnerabilities using Static Analysis Tools	The paper discusses rule-based static analysis tools like Oyente and Mythril for detecting common smart contract vulnerabilities.	Limited to predefined rules, high false positives, and inability to detect unknown or complex vulnerabilities.
[2]	Smart Contract Vulnerability Detection using Machine Learning	The study applies deep learning models such as CNNs and RNNs to identify vulnerability patterns from historical datasets.	Requires large labeled datasets, high computational cost, poor interpretability, and difficulty handling evolving attack patterns
[3]	Large Language Models for Smart Contract Security Analysis	The paper explores the use of LLMs for understanding smart contract semantics and detecting logical vulnerabilities.	Single-agent LLMs face scalability issues and lack cross-validation, leading to inconsistent vulnerability detection.

3. METHODOLOGY

The proposed methodology describes the systematic process used to design and implement an LLM-powered multi-agent framework for detecting vulnerabilities in smart contracts. The system integrates code preprocessing, intelligent agent-based analysis, collaborative decision-making, and report generation to ensure accurate and explainable vulnerability detection.

1. Input Collection
Smart contracts written in Solidity are provided as input to the system. The contracts may be obtained from developers, repositories, or test datasets. Both source code and compiled bytecode can be accepted to support comprehensive analysis.

2. Preprocessing and Code Normalization
Before analysis, the smart contract code undergoes preprocessing to improve consistency and readability:
• Removal of comments and redundant whitespaces
• Standardization of variable names and formatting
• Extraction of contract structure such as functions, modifiers, and state variables
• Generation of an intermediate representation (IR) for semantic understanding
This step ensures that the LLM agents receive clean and structured input.

3. Vulnerability Knowledge Base Initialization
A vulnerability knowledge base is created using:
• Known smart contract vulnerabilities (e.g., reentrancy, integer overflow, access control flaws)
• Security best practices and coding standards
• Historical exploit patterns and real-world attack examples
This knowledge base is used to guide agent prompts and improve contextual reasoning during analysis.

4. LLM-Powered Multi-Agent Analysis
The core of the methodology involves a **multi-agent system**, where each agent is powered by a Large Language Model and specializes in a specific vulnerability domain:
• **Reentrancy Detection Agent** – identifies unsafe external calls and state update issues
• **Arithmetic Vulnerability Agent** – detects integer overflow and underflow problems
• **Access Control Agent** – analyzes authorization and permission checks
• **Logic & Semantic Agent** – examines business logic and contract behavior
• **Gas Optimization Agent** – detects inefficient gas usage and denial-of-service risks

4. CONCLUSIONS

In conclusion, the project "Advanced Smart Contract Vulnerability Detection via LLM-Powered Multi-Agent Systems" presents an effective and intelligent solution to the growing security challenges faced by blockchain-based smart contracts. By integrating Large Language Models with a collaborative multi-agent architecture, the system enables deeper semantic understanding of smart contract code, allowing it to detect both known and previously unseen vulnerabilities with improved accuracy.

The proposed approach combines code preprocessing, specialized vulnerability-focused agents, inter-agent collaboration, and explainable reporting, thereby reducing false positives and enhancing developer trust in the detection results. Unlike traditional rule-based or single-model techniques, the multi-agent framework provides cross-validation of findings, making the system more robust and reliable.

5. REFERENCES

[1] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on Ethereum smart contracts (SoK)," in Proc. Princ. Secur. Trust: 6th Int. Conf., POST 2017, Held as Part Eur. Joint Conf. Theory Pract. Softw. (ETAPS), Uppsala, Sweden, Apr. 2017, pp. 164–186.

[2] getfailsafe, "Failsafe web3 security report 2025." 2025. Accessed: Jan. 1, 2025. [Online]

[3] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, and P. Sen, "A survey of the state of explainable AI for natural language processing," 2020, arXiv:2010.00711.

[4] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hi erarchical text-conditional image generation with clip latents," 2022, arXiv:2204.06125.

[5] Y. Dong, X. Jiang, Z. Jin, and G. Li, "Self-collaboration code generation via ChatGPT," 2023, arXiv:2304.07590.

[6] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "HuggingGPT: Solving ai tasks with ChatGPT and its friends in hugging face," Adv. Neural Inf. Process. Syst., vol. 36, pp. 38154–38180, 2024.

[7] C. Qian et al., "Communicative agents for software development," 2023, arXiv:2307.07924

[8] Q. Zhang et al., "A survey on large language models for software engineering," 2023, arXiv:2312.15223.

[9] I. David, L. Zhou, K. Qin, D. Song, L. Cavallaro, and A. Gervais, "Do you still need a manual smart contract audit?" 2023, arXiv:2306.12338.

[10] G. Team et al., "Gemma: Open models based on Gemini research and technology," 2024, arXiv:2403.08295.

[11] H. Touvron et al., "Llama: Open and efficient foundation language models," 2023, arXiv:2302.13971.

[12] A. Q. Jiang et al., "Mixtral of experts," 2024, arXiv:2401.04088.