Journal of Computer Science and Electrical Engineering
Print ISSN: 2663-1938
Online ISSN: 2663-1946
DOI: https://doi.org/10.61784/jcsee3130

APPLICATION OF LARGE LANGUAGE MODELS IN SMART CONTRACT VULNERABILITY DETECTION

Yun Li[1*], YanLing Liu[1], ChangEr Liu[1], XinMan Luo[2]
[1]*School of Finance and Economics, Hainan Vocational University of Science and Technology, Haikou 571126, Hainan, China.*
[2]*School of Information Science and Technology, Qiongtai Normal University, Haikou 571100, Hainan, China.*
[*]*Corresponding Author: Yun Li*

Abstract: With the rapid iteration of blockchain technology, smart contracts, as core components of decentralized applications, directly impact the stability of on-chain assets and ecosystems through their security. Traditional vulnerability detection methods primarily rely on expert rules and static analysis, facing bottlenecks such as high false positive rates and poor adaptability to complex logical vulnerabilities. In recent years, Large Language Models (LLMs), with their exceptional code understanding and reasoning capabilities, have provided new technical pathways for smart contract security auditing. This paper focuses on LLM-driven smart contract vulnerability detection technologies, systematically reviewing mainstream application paradigms from prompt engineering to model fine-tuning. The paper first reviews the current state of smart contract security and the limitations of traditional methods; subsequently, it provides in-depth analysis of the architectural design and core mechanisms of representative frameworks such as GPTLens and SmartVD, evaluating their performance in detection accuracy and recall rate; finally, addressing current challenges including data scarcity, model hallucinations, and computational overhead, it proposes future evolution directions such as multimodal fusion and human-in-the-loop auditing, providing reference for research and practice in related fields.
Keywords: Large language models; Smart contracts; Vulnerability detection; Blockchain security; Program analysis

1 INTRODUCTION

Smart contracts are Turing-complete programs running on blockchain platforms that can automatically handle digital assets according to preset logic [1]. With the explosion of the Ethereum ecosystem, smart contracts have become the cornerstone of Decentralized Finance (DeFi) and Non-Fungible Tokens (NFTs). However, due to blockchain's immutability, contracts are difficult to patch once deployed, and any minor code defect may lead to catastrophic consequences. From The DAO incident to recent cross-chain bridge attacks, economic losses caused by contract vulnerabilities have reached billions of dollars, making security issues a key bottleneck constraining blockchain development [2].

Traditional defense mechanisms primarily rely on static analysis, dynamic fuzzing, and symbolic execution techniques [3]. Although these tools demonstrate maturity in identifying conventional vulnerabilities such as reentrancy attacks and integer overflows, they often prove inadequate when facing deep defects involving complex business logic: first, they are highly dependent on predefined rules and struggle to address new attack variants; second, they lack deep understanding of code semantics, resulting in high false positive rates and increasing the burden of manual auditing.

In recent years, Large Language Models represented by GPT-4 and CodeLlama have demonstrated remarkable potential in code generation and comprehension tasks [4]. Benefiting from pre-training on massive code corpora, LLMs not only master the syntactic structures of programming languages but also possess semantic understanding capabilities to infer code intent. Research indicates that introducing LLMs into software testing and vulnerability mining can effectively compensate for the semantic blind spots of traditional methods [5]. This paper aims to systematically survey cutting-edge developments in large language models for smart contract vulnerability detection, analyze the strengths and weaknesses of different technical frameworks, and explore the challenges and future opportunities facing this field.

2 BACKGROUND

2.1 Common Smart Contract Vulnerabilities and Threats

According to the OWASP Smart Contract Security Guidelines, current high-risk vulnerabilities are primarily concentrated at the logical and execution levels. Among these, Reentrancy attacks exploit the characteristic of contracts making external calls before state updates, allowing attackers to recursively call contract functions to drain funds [6]; Integer Overflow/Underflow stems from lack of numerical boundary checking (common in older versions of Solidity) [7]. Additionally, Access Control failures often result from missing permission modifiers leading to unauthorized critical operations [8]. At the execution environment level, Timestamp Dependence allows miners to profit by manipulating block timestamps [9], while Unchecked Low-Level Calls may trigger abnormal states due to ignoring return values of failed external calls [10].

2.2 Limitations of Traditional Detection Methods

Existing mainstream detection tools such as Mythril, Slither, and SmartCheck primarily employ static analysis, symbolic execution, and fuzzing techniques [11]. Static analysis discovers known vulnerabilities by matching code patterns but has low detection rates for logical vulnerabilities; symbolic execution attempts to traverse all execution paths but often faces path explosion problems; fuzzing relies on random inputs and struggles to cover deep code branches. Overall, traditional methods lack deep understanding of code semantics and heavily depend on manually defined detection rules, resulting in high false positives or false negatives when facing complex, novel vulnerabilities.

3 APPLICATION OF LARGE LANGUAGE MODELS IN VULNERABILITY DETECTION

3.1 Technical Advantages of LLMs

Compared to traditional tools, large language models demonstrate multiple advantages in vulnerability detection: first is deep semantic understanding—LLMs can comprehend relationships between variable naming, comments, and control flow, thereby identifying business logic vulnerabilities that traditional tools struggle to capture; second is generalization capability—through few-shot learning or fine-tuning, models can quickly adapt to new contract languages or vulnerability patterns; finally is interactive explanation—LLMs not only identify vulnerability locations but also generate natural language descriptions of remediation suggestions, significantly lowering the threshold for manual auditing.

3.2 Analysis of Mainstream Detection Frameworks

Adversarial Detection Framework: GPTLens. The GPTLens framework proposed by Hu et al. innovatively introduces an adversarial "generation-evaluation" design [12]. The framework comprises two core components: the Auditor and the Critic. The Auditor is responsible for divergent thinking, identifying as many potential vulnerabilities as possible to ensure recall rate; the Critic simulates the perspective of security experts, conducting secondary verification and filtering of the auditor's reports. This two-stage mechanism effectively mitigates the LLM's tendency to produce hallucinations. Experiments show that GPTLens achieves significant performance improvements in detecting logical vulnerabilities compared to traditional static analysis tools.

Data-Driven Fine-Tuning Framework: SmartVD. Addressing the insufficient domain-specific knowledge of general LLMs, Alam et al. proposed the SmartVD framework [13]. The core contribution of this work lies in constructing a high-quality, class-balanced instruction fine-tuning dataset VulSmart, and conducting specialized training on this foundation. SmartVD integrates multiple prompting strategies including zero-shot reasoning, few-shot analogy, and chain-of-thought. In benchmark dataset evaluations, fine-tuned GPT-3.5 Turbo and GPT-4o Mini achieved extremely high accuracy in multi-class vulnerability detection tasks, demonstrating the necessity of domain adaptation training for enhancing model security capabilities.

Ensemble Learning Framework: FELLMVP. To further improve detection robustness, Yu et al. proposed the FELLMVP framework [14]. This method combines Control Flow Graphs (CFG) from program analysis with the reasoning capabilities of large models. Its unique feature lies in utilizing CFG to extract structured features of code, assisting LLMs in understanding complex execution logic, and employing an ensemble learning strategy to aggregate predictions from multiple LLMs with different architectures. In large-scale tests containing 15,000 samples, FELLMVP achieved 98.8% accuracy and an 88% F1 score, demonstrating that multi-model collaboration and structural information introduction can effectively break through single-model performance ceilings.

3.3 Detection Performance Analysis

Evaluating LLM vulnerability detection performance typically employs the following core metrics: Accuracy reflects the overall correctness of model judgments; Precision focuses on the proportion of true vulnerabilities among code flagged as vulnerable, directly impacting audit efficiency; Recall measures the model's ability to discover all true vulnerabilities and is the most critical metric for security auditing; F1 score is the harmonic mean of precision and recall, used for comprehensive model performance evaluation. Current research indicates that while LLMs often outperform traditional tools in F1 scores, there is still room for optimization in reducing false positive rates (improving Precision).

4 KEY TECHNICAL CHALLENGES AND SOLUTIONS

4.1 Data Quality and Annotation Bottleneck

High-quality annotated data is the foundation for training models, but vulnerability samples in the smart contract domain are scarce and annotation thresholds are extremely high. Existing public datasets (such as SmartBugs) often suffer from class imbalance or label noise issues. Solutions: Researchers are beginning to attempt data augmentation techniques, generating variant samples through code refactoring and variable renaming; or using synthetic data, having LLMs automatically inject erroneous code based on vulnerability patterns to expand datasets. Additionally, semi-supervised learning techniques are being applied, utilizing unlabeled massive on-chain code to improve model pre-training effects.

4.2 Model Interpretability and Hallucination Issues

The "black box" nature of LLMs and occasional "hallucinations" are major obstacles to their practical deployment. Models may confidently report non-existent vulnerabilities, interfering with auditors' judgments. Solutions: Introduce Chain-of-Thought (CoT) prompting, forcing models to output reasoning steps, making decision processes transparent [15]. Meanwhile, combined with attention mechanism visualization techniques, highlighting code regions the model focuses on helps experts quickly verify the reasonableness of detection results [16].

4.3 Context Limitations and Computational Efficiency

Smart contracts often involve complex cross-contract calls, and code volume may exceed LLM context window limits. Additionally, large model inference costs are high, making it difficult to meet real-time transaction monitoring demands. Solutions: Employ slicing techniques, extracting only code fragments related to critical variables for model input; or utilize knowledge distillation, transferring large model capabilities to lightweight models [17], enabling deployment on edge devices or real-time systems.

5 FUTURE DEVELOPMENT TRENDS

5.1 Multimodal Fusion Auditing

Future detection systems will transcend single code text analysis. By fusing multimodal data such as code, design documents, developer comments, and transaction graphs, LLMs can construct more complete contextual cognition. For example, combining code Control Flow Graph (Graph) modalities with source code (Text) modalities enables more precise identification of complex business logic violations.

5.2 Continuous Learning and Dynamic Evolution

Facing constantly evolving hacker attack methods, static pre-trained models struggle to remain effective long-term. Systems need continuous learning capabilities, able to automatically extract features from newly occurring attack events and daily on-chain contracts, dynamically updating model parameters to achieve self-evolution of defensive capabilities.

5.3 Human-in-the-Loop

Fully automated auditing is difficult to achieve zero false positives in the short term. Human-in-the-loop will become the mainstream model: LLMs serve as "co-pilots" responsible for large-scale preliminary screening and code interpretation, while human experts handle final judgments on core logic. Through expert feedback (RLHF), models are continuously optimized, ultimately achieving deep coupling of intelligence and experience.

6 CONCLUSION

Large language models have brought a paradigm shift to smart contract vulnerability detection. Their advantages in semantic understanding, pattern recognition, and remediation suggestion generation effectively compensate for shortcomings of traditional static analysis tools. Although current technologies still face challenges in data dependency, hallucination control, and computational costs, with the continuous evolution of frameworks such as GPTLens and SmartVD, and the application of technologies like multimodal fusion and continuous learning, LLM-driven intelligent auditing will become a key force in safeguarding blockchain ecosystem security. Future research should focus on constructing more credible benchmark test sets and exploring lightweight, interpretable specialized security models.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

FUNDING

This work was supported in part by the 2025 Hainan Vocational University of Science and Technology Teaching Reform Project “Research on Pathways to Improve the Teaching Ability of ‘Dual-Qualified’ Ideological and Political Teachers in Vocational Undergraduate Education” (No. HKJG2025-44).

REFERENCES

[1] Buterin V. A next-generation smart contract and decentralized application platform. White Paper, 2014, 3(37): 2-1.
[2] Atzei N, Bartoletti M, Cimoli T. A survey of attacks on Ethereum smart contracts (SoK)//International Conference on Principles of Security and Trust. Berlin, Heidelberg: Springer, 2017: 164-186.
[3] Zhang P, Xiao F, Luo X. SolidityCheck: Quickly detecting smart contract problems through regular expressions. arXiv preprint arXiv:1911.09425, 2019.

[4] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners//Advances in Neural Information Processing Systems (NeurIPS), 2020: 1877-1901.
[5] Chen M, Tworek J, Jun H, et al. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021.
[6] Liu C, Liu H, Cao Z, et al. ReGuard: Finding reentrancy bugs in smart contracts//Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings (ICSE), 2018: 65-68.
[7] Tan B, Mariano B, Lahiri S K, et al. SolType: Refinement types for arithmetic overflow in Solidity. Proceedings of the ACM on Programming Languages, 2022, 6(POPL): 1-29.
[8] Liu Z, Qian P, Wang X, et al. Combining graph neural networks with expert knowledge for smart contract vulnerability detection. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(2): 1296-1310.
[9] Kalra S, Goel S, Dhawan M, et al. ZEUS: Analyzing safety of smart contracts//Proceedings of the Network and Distributed System Security Symposium (NDSS), 2018: 1-12.
[10] Luu L, Chu D H, Olickel H, et al. Making smart contracts smarter//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS), 2016: 254-269.
[11] Tikhomirov S, Voskresenskaya E, Ivanitskiy I, et al. SmartCheck: Static analysis of Ethereum smart contracts//Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), 2018: 9-16.
[12] Hu S, Huang T, Ilhan F, et al. Large language model-powered smart contract vulnerability detection: New perspectives//2023 IEEE 5th International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA). IEEE, 2023: 297-306.
[13] Alam M T, Halder R, Maiti A. Detection made easy: Potentials of large language models for Solidity vulnerabilities. arXiv preprint arXiv:2409.10574, 2024.
[14] Luo Y, Xu W, Andersson K, et al. FELLMVP: An ensemble LLM framework for classifying smart contract vulnerabilities//2024 IEEE International Conference on Blockchain. IEEE, 2024: 89-96.
[15] Sun Y, Wu D, Xue Y, et al. LLM4Vuln: A unified evaluation framework for decoupling and enhancing LLMs' vulnerability reasoning. arXiv preprint arXiv:2401.16185, 2024.
[16] Zhao W X, Zhou K, Li J, et al. A survey of large language models. arXiv preprint arXiv:2303.18223, 2023.
[17] Dettmers T, Pagnoni A, Holtzman A, et al. QLoRA: Efficient finetuning of quantized LLMs//Advances in Neural Information Processing Systems (NeurIPS), 2023, 36: 10088-10115.