Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 9
AIDML 2025

From Rule-Driven to Data-Driven: Technological Evolution, Challenges, and Future Trends in Smart Contract Vulnerability Detection

Qiheng Yu *

School of Software, Beihang University, Beijing, China

* Corresponding Author Email: 22371499@buaa.edu.cn

Abstract. Smart contract vulnerability detection is undergoing a paradigm shift from rule-driven to data-driven approaches. While traditional methods relying on predefined patterns struggle with evolving attack vectors, machine learning enables novel solutions through multi-level feature learning. This study systematically traces technological evolution: Early research established foundational frameworks using syntax tree features with classical classifiers. Deep learning advancements introduced temporal models capturing execution dynamics, spatial convolutional networks decoding bytecode structures, and graph neural networks modeling cross-contract dependencies, collectively enabling precise identification of complex vulnerabilities. Semi-supervised learning and hybrid architectures further maintain detection robustness under limited labeled data. Three evolutionary trends emerge: detection scope expanding from single-contract code to multi-protocol interaction networks, feature representation transitioning from manual design to neural self-encoding, and training paradigms shifting from fully-supervised to collaborative weak supervision. Persistent challenges include unverified model interpretability, inadequate adaptation to dynamic on-chain environments, and insufficient traceability of cross-chain attack paths. Future solutions require integrating neuro-symbolic verifiability, dynamic graph networks' real-time perception, and federated knowledge-sharing mechanisms to develop adaptive, auditable next-generation detection systems.

Keywords: Smart Contract Security, Deep Learning, Graph Neural Networks, Vulnerability Detection, Blockchain.

1. Introduction

Since its formal proposal in the Bitcoin whitepaper in 2008 [1], blockchain technology has evolved from a mere cryptocurrency infrastructure to a groundbreaking distributed trust mechanism. As one of its core innovations, smart contracts replace traditional trust intermediaries with programmable logic, demonstrating immense potential in fields such as financial technology (DeFi), supply chain management, and digital identity [2]. However, while the immutability of smart contracts ensures security, it also introduces extremely high costs for vulnerability remediation—once deployed, the contract logic cannot be altered, and any code defect may lead to irreversible losses. The infamous DAO attack in 2016 serves as a prime example: due to an undetected reentrancy vulnerability, attackers exploited a logic flaw in the contract to steal approximately $50 million worth of ETH, ultimately forcing the Ethereum community to execute a hard fork to reverse the transactions [3]. Such incidents not only expose the limitations of traditional static analysis tools but also highlight the urgent need for advancements in smart contract vulnerability detection technology.

Traditional detection methods suffer from rigid patterns, resulting in high false positive and false negative rates when encountering novel vulnerabilities. In contrast, machine learning techniques, with their capabilities in automated feature extraction, high-dimensional data processing, and adaptability to unknown vulnerability patterns, have emerged as a viable solution for smart contract vulnerability detection. Early research primarily relied on features extracted from Abstract Syntax Trees (ASTs), combined with static analysis tools to generate labeled data, and trained models such as Support Vector Machines (SVMs) or Random Forests (RFs) for efficient vulnerability detection [4]. However,

these methods still depend on manual feature engineering. With the advancement of deep learning, researchers began employing advanced neural network models such as Recurrent Neural Networks (RNNs) to model code sequences, enhancing reentrancy vulnerability detection through attention mechanisms [5]. For cases where source code is unavailable, some studies shifted to analyzing contract bytecode, utilizing Convolutional Neural Networks (CNNs) to process bytecode sequences [6]. Meanwhile, other researchers constructed multi-dimensional contract graphs, combining Graph Neural Networks (GNNs) with expert rules to detect cross-contract vulnerabilities [7]. More recent efforts have focused on addressing the challenge of scarce labeled data by integrating paradigms such as semi-supervised and active learning, combining static and dynamic execution information to improve model performance [8].

Although research on smart contract vulnerability detection is flourishing, it remains fragmented, with disparate methods lacking a unified framework in terms of vulnerability coverage, feature representation, and evaluation criteria. This paper aims to fill this gap through a systematic multi-dimensional analysis, categorizing and comparing existing research across several dimensions. At the detection object level, this paper distinguished between analysis methods operating at different granularities, such as source code, bytecode, and on-chain transaction levels. At the technical paradigm level, this paper systematically traces the methodological evolution from conventional machine learning to advanced deep learning architectures. Early approaches predominantly employed feature-engineered supervised learning with classifiers like SVM and Random Forest, later transitioning to sequence-aware models for temporal pattern mining in transaction flows. Subsequent innovations introduced CNN for structural feature extraction from bytecode matrices, while GNN emerged to model control flow dependencies. This paper also highlights how hybrid paradigms combine symbolic execution with neural embeddings to overcome data scarcity limitations. Building on this foundation, this paper systematically reviews and classifies various detection mechanisms, while also offering insights into their limitations, current challenges, and future directions.

2. Method

2.1. Prerequisites for Smart Contract Vulnerability

Smart contract vulnerabilities essentially represent the propagation of abnormal patterns in code logic within distributed ledger systems. Based on differences in attack surfaces and vulnerability trigger conditions, current research commonly adopts the classification framework proposed by Atzei et al. [9], categorizing vulnerabilities into three levels: high-level languages (e.g., Solidity), Ethereum Virtual Machine (EVM), and blockchain. The Solidity level focuses on programming language features and development logic flaws (e.g., reentrancy attacks, integer overflows), the EVM level addresses bytecode execution mechanisms and resource management issues (e.g., stack overflows, abnormal gas consumption), while the blockchain level targets risks arising from distributed environmental characteristics (e.g., timestamp dependence, random number predictability).

2.2. Traditional Detection Methods

2.2.1. Formal Verification

Grishchenko et al. proposed the first complete small-step semantic framework for EVM bytecode [10], constructing a verifiable fine-grained execution model through formal methods. This framework covers key instructions such as contract calls, exception handling, and miner parameter dependencies, and formally defines smart contract security criteria like call integrity, atomicity, and miner parameter independence based on hyper-properties and safety properties. The framework validated the correctness of 304 test cases on Ethereum's official test set and revealed logical flaws in existing tools like Oyente for exception handling and timestamp dependence detection [11].

2.2.2. Fuzz Testing

Liao et al. proposed SoliAudit [12], which leverages Solidity opcode features and dynamic fuzz testing techniques. In tests on 17,979 contracts, it achieved 90% vulnerability detection accuracy, successfully identifying 13 types of DASP Top 10 threats, including reentrancy and arithmetic overflows. Its automatically generated fuzz testing contracts validated effectiveness in TokenBank reentrancy vulnerabilities and CTF challenges, demonstrating rapid adaptation to unknown vulnerabilities without relying on expert-defined rules.

2.2.3. Symbolic Execution

Loi et al. proposed an EVM bytecode analysis method based on symbolic execution and constraint solving [11]. By constructing control flow graphs for multi-path exploration, mapping EVM instructions to symbolic expressions, and generating path constraints, the method uses the Z3 solver to verify path feasibility. Their tool Oyente identified 3,056 problematic contracts in transaction order dependence detection, achieving 93.6% accuracy for timestamp dependence and exception handling vulnerabilities. The method incorporates depth-first search strategies to traverse branch conditions and enables cross-contract vulnerability analysis through symbolic global state and transaction parameters.

2.2.4. Static Analysis

Slither converts Solidity contracts into SlithIR intermediate representation in SSA form [13], combining data flow and taint tracking techniques. In tests on 1,000 contracts, it achieved a low false positive rate of 10.9%, outperforming tools like Securify [14]. It successfully detected real-world vulnerabilities in DAO/SpankChain and optimized detection results, revealing that 56% of contracts contained variables that could be constantized.

2.3. Machine Learning Methods

2.3.1. Traditional Machine Learning Algorithms

Momeni P et al. proposed a supervised learning-based smart contract vulnerability detection method [4]. Using the Solidity compiler to extract abstract syntax trees (ASTs), they quantified 17 code structure features to characterize code complexity and combined Slither and Mythril static analysis tools to detect 46 types of vulnerabilities [13, 15]. Semantic alignment resolved conflicts between tool results, generating a binary labeled dataset. For each vulnerability type, four supervised models (including SVM and RF) were independently trained, with hyperparameters optimized via grid search and performance evaluated through 5-fold cross-validation. Statistical tests showed SVM achieved 99% accuracy in integer underflow detection, while RF achieved an 87% F1-score for reentrancy vulnerabilities, with an average detection accuracy of 95% for 16 core vulnerability types.

2.3.2. Long Short-Term Memory Networks

Compared to RNNs, which are suitable for basic pattern recognition, Long Short-Term Memory (LSTM) networks are more widely applied in smart contract vulnerability detection due to their ability to identify complex vulnerabilities spanning multiple operations. Peng et al. proposed a bidirectional LSTM-ATT method based on semantic fragment serialization [5], converting Solidity source code into token sequences and using bidirectional LSTMs with attention mechanisms to dynamically weight critical code segments. Their model achieved an 88.26% F1-score for reentrancy detection. Wesley et al. focused on opcode sequence modeling [16], disassembling EVM bytecode into instruction sequences and employing a two-layer LSTM to learn latent vulnerability patterns. Their method achieved 99.57% detection accuracy on a dataset of 620,000 contracts and corrected 92.86% of false positives from the symbolic tool Maian. Both methods mapped discrete code units to dense vectors using word2vec and standardized sequence lengths via zero-padding or truncation. To address data imbalance, Wesley's team used Synthetic Minority Over-sampling Technique (SMOTE) to balance rare vulnerability samples (e.g., suicidal and prodigal contracts), while Peng et al. optimized model sensitivity to low-frequency vulnerabilities through 5-fold cross-validation.

2.3.3. Convolutional Neural Networks

Huang et al. innovatively applied CNNs to smart contract vulnerability detection by converting Solidity bytecode into 24-bit RGB images [6]. Using Inception-v3's 1x1 convolutional kernels, they addressed semantic distortion caused by traditional 3x3/5x5 convolutions. Their multi-label CNN model achieved 97.39% accuracy after 500 training epochs, effectively handling sample imbalance via transfer learning and completing single analyses in 1.5 seconds, enabling parallel detection of 17 Solidity compiler vulnerabilities. CodeNet introduced stride-free depthwise separable convolutions to directly analyze bytecode sequences [17], avoiding semantic loss, and incorporated global max pooling (GMP) to retain key features. This improved detection speed to 0.13 seconds per analysis and accuracy to 97.66%, enhancing CNN robustness and efficiency in smart contract vulnerability detection.

2.3.4. Graph Neural Networks

Yuan et al. proposed a multi-dimensional contract graph construction method [18], categorizing code elements into core nodes, auxiliary nodes, and fallback nodes, and encoding execution path dependencies via four types of temporal edges. To enhance the model's ability to capture vulnerability-sensitive paths, they designed a Degree-Removed Graph Convolutional Network (DR-GCN), eliminating traditional GCN degree matrix normalization and using adjacency matrix squaring to strengthen connectivity in heterogeneous graphs. They also introduced a Temporal Message Passing Network (TMP) to dynamically propagate information along temporal edges, addressing long-range dependencies in cross-function call chains. Liu et al. combined expert-defined security patterns with graph features [7], jointly optimizing convolutional and max-pooling layers to improve sensitivity to complex vulnerabilities. Experiments showed GNN methods achieved detection accuracies of 89.15%, 89.02%, and 83.21% for reentrancy, timestamp dependence, and infinite loop vulnerabilities, respectively, on a dataset of 300,000 contract functions from Ethereum and VNT Chain, outperforming the symbolic tool Oyente (77.1%) by over 12%. For cross-contract loop vulnerabilities (e.g., cyclic calls between functions and fallback), GNNs achieved an 85.3% F1-score, a 15% improvement over LSTM models like BLSTM-ATT.

2.3.5. Hybrid Deep Learning Architectures

While deep learning has demonstrated high efficiency in smart contract vulnerability detection, its reliance on massive labeled data poses practical deployment challenges. To address scarce labeled data, researchers proposed hybrid architectures that combine active learning to select high-value samples (reducing manual labeling costs) with semi-supervised learning to leverage unlabeled data potential, optimizing model performance under limited labeled resources. Sun et al. proposed the ASSBert framework [8], integrating active and semi-supervised learning to collaboratively select high-uncertainty samples for expert labeling and high-confidence samples for pseudo-labeling. Experimental results showed ASSBert achieved an average detection accuracy of 82.3% for six vulnerability types at a 20% labeling ratio, with timestamp vulnerability accuracy improving by 6.5 percentage points over traditional BERT (72.1%). For reentrancy detection, it achieved 79% accuracy and 73% recall, validating the hybrid architecture's advantages in scenarios with scarce labeled data.

3. Results and Discussion

3.1. The Performance of Models

Experimental results demonstrate that machine learning-based detection methods significantly outperform traditional techniques in both accuracy and efficiency. Among supervised learning approaches, the SVM model achieved 99% accuracy in integer underflow detection, while the RF model attained an 87% F1-score for reentrancy vulnerabilities, with an average detection accuracy of 95% across 16 core vulnerability types. These results validate the effectiveness of combining structural features with statistical learning models, while also revealing the dependency on manual expertise in feature engineering.

Deep learning models demonstrate superior pattern recognition capabilities. The LSTM-ATT model achieved an 88.26% F1-score for reentrancy detection, representing a 15 percentage point improvement over traditional methods. The CNN model attained 97.39% detection accuracy through bytecode image representation, with single analysis requiring only 1.5 seconds. The GNN approach reached an 85.3% F1-score for cross-contract vulnerability detection, outperforming LSTM models by 15%. These findings suggest that deep learning's automatic feature extraction can better capture complex semantic relationships in code, though such performance gains often come at the cost of higher computational resource consumption.

Notably, hybrid architecture methods demonstrate unique advantages in data-scarce scenarios. The ASSBert framework achieves an average detection accuracy of 82.3% for six types of vulnerabilities using only 20% labeled data, representing a 6.5 percentage point improvement over traditional BERT models. This result confirms the effectiveness of combining active learning with semi-supervised learning strategies in alleviating data annotation bottlenecks. However, the method's performance on low-frequency vulnerabilities represented by suicidal contracts still has room for improvement, reflecting the current models' sensitivity to class imbalance issues.

From the perspective of technological evolution, this study reveals three key development trends in the field of smart contract vulnerability detection: 1) detection granularity expanding from single contracts to cross-contract systems; 2) feature representation shifting from manual design to automatic learning; and 3) learning paradigms transitioning from fully supervised to semi-supervised/weakly-supervised approaches.

3.2. Challenges and Future Prospects

The continuous evolution of smart contract vulnerability detection technologies jointly drives detection techniques towards more intelligent and practically adaptive directions. However, current methods still exhibit three fundamental limitations.

The primary challenge lies in the inherent conflict between model interpretability and security verification requirements: Although deep neural networks demonstrate exceptional accuracy in vulnerability pattern recognition, their opaque decision-making processes cannot provide auditable evidence required for regulatory compliance and post-attack forensics. This deficiency becomes particularly evident in cross-contract interaction scenarios where vulnerabilities propagate through multi-layered transactions.

Another critical limitation is that existing detection techniques (such as symbolic execution and static analysis) struggle to effectively capture vulnerability propagation paths in cross-contract call chains. This shortcoming is especially apparent in DeFi protocol combination scenarios (including flash loan attacks and cross-contract reentrancy).

Furthermore, static training paradigms prove inadequate in addressing the inherent dynamism of blockchain ecosystems. Emerging vulnerability patterns related to new contract standards and exponentially growing transaction throughput create moving targets for detection systems, exposing flaws in models trained on historical data. The latency inherent in traditional detection processes further exacerbates this issue, potentially causing DeFi applications to miss critical vulnerability identification windows in time-sensitive scenarios.

These challenges demand paradigm shifts across three synergistic research directions: Neuro-symbolic methods, which integrate the logical reasoning of symbolic systems with the perceptual learning capabilities of neural networks, enable "training-inference integration" intelligent frameworks. Such systems aim to generate human-understandable vulnerability proofs during model training while addressing black-box limitations and enhancing extrapolation capabilities. Concurrently, the combination of dynamic graph neural networks and incremental learning technologies will break through current detection method constraints. By constructing real-time updated contract interaction graphs, this approach can not only capture intra-contract vulnerability

characteristics but also trace complex dependency relationships in cross-contract calls, specifically targeting novel attack patterns transmitted through multi-contract layers. Finally, the introduction of adaptive federated learning frameworks offers multi-faceted solutions to model update latency issues. Through collaborative training and knowledge sharing among distributed nodes, detection systems can continuously assimilate new attack patterns while maintaining existing knowledge through elastic parameter adjustment mechanisms, thereby preserving detection timeliness and accuracy in rapidly evolving blockchain environments.

4. Conclusion

This paper reveals a paradigm shift in smart contract vulnerability detection from rule-driven to data-driven approaches, progressing from localized analysis to global contextual awareness. While deep learning's automated feature extraction overcomes traditional limitations in semantic modeling and graph-based learning enables cross-contract vulnerability identification, three fundamental challenges persist: the opacity of neural models conflicts with security verification requirements, static analysis frameworks mismatch dynamic blockchain environments, and limited labeled data struggles to capture diverse attack patterns. Future advancements should integrate neuro-symbolic systems for verifiable vulnerability reasoning, develop dynamic graph representations for real-time transaction topology analysis, and establish federated learning mechanisms for continuous cross-chain knowledge accumulation. Achieving self-adaptive detection capabilities through multimodal fusion frameworks will be crucial for building trustworthy security foundations in decentralized ecosystems.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," (2008). [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] S. N. Khan, F. Loukil, C. Ghedira-Guegan et al., Peer-to-Peer Netw. Appl. 14, 2901–2925 (2021).

[3] Metz C. A $50 Million Hack Just Showed That the DAO Was All Too Human [EB/OL]. WIRED,2016-06-17[2025-04-04]. https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/.

[4] P. Momeni, Y. Wang, and R. Samavi, "Machine learning model for smart contracts security analysis," in 2019 17th International Conference on Privacy, Security and Trust (PST), IEEE, (2019), pp. 1–6.

[5] P. Qian, Z. Liu, Q. He et al., IEEE Access 8, 19685–19695 (2020).

[6] T. T. H. D. Huang, "Hunting the ethereum smart contract: Color-inspired inspection of potential attacks," arXiv preprint arXiv:1807.01868, (2018).

[7] Z. Liu, P. Qian, X. Wang et al., IEEE Trans. Knowl. Data Eng. 35, 1296–1310 (2021).

[8] X. Sun, L. Tu, J. Zhang et al., J. Inf. Secur. Appl. 73, 103423 (2023).

[9] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts," in International Conference on Principles of Security and Trust, Springer Berlin Heidelberg, (2017), pp. 164–186.

[10] I. Grishchenko, M. Maffei, and C. Schneidewind, "A semantic framework for the security analysis of ethereum smart contracts," in Principles of Security and Trust: 7th International Conference, POST 2018, Springer International Publishing, (2018), pp. 243–269.

[11] L. Luu, D. H. Chu, H. Olickel et al., "Making smart contracts smarter," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, (2016), pp. 254–269.

[12] J. W. Liao, T. T. Tsai, C. K. He et al., "Soliaudit: Smart contract vulnerability assessment based on machine learning and fuzz testing," in 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), IEEE, (2019), pp. 458–465.

[13] J. Feist, G. Grieco, and A. Groce, "Slither: A static analysis framework for smart contracts," in 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), IEEE, (2019), pp. 8–15.

[14] P. Tsankov, A. Dan, D. Drachsler-Cohen et al., "Securify: Practical security analysis of smart contracts," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, (2018), pp. 67–82.

[15] B. Mueller, "Smashing ethereum smart contracts for fun and real profit," HITB SECCONF Amsterdam 9, 4–17 (2018).

[16] W. J. W. Tann, X. J. Han, S. S. Gupta et al., "Towards safer smart contracts: A sequence learning approach to detecting security threats," arXiv preprint arXiv:1811.06632, (2018).

[17] S. J. Hwang, S. H. Choi, J. Shin et al., IEEE Access 10, 32595–32607 (2022).

[18] Y. Zhuang, Z. Liu, P. Qian et al., "Smart contract vulnerability detection using graph neural networks," in Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, (2021), pp. 3283–3290.