The Comprehensive Investigation of Smart Contract Security via Machine Learning

Yulun Miao

Computer Science, University Academy 92, Manchester, United Kingdom

l39601331@student.ua92.ac.uk

Abstract. Smart contracts are widely used in blockchain systems to decentralize and execute the agreements. However, they undergo several security vulnerabilities due to their immunity feature. Therefore, Machine Learning (ML) has emerged as a solution for detecting the vulnerabilities in smart contract security. This paper demonstrates a comprehensive review of current ML-based technologies applied in this field. The ML-based methods are divided into two groups: deep-learning, and traditional ML methods. The techniques of those methods include Graph Neural Networks (GNNs), contrastive learning, expert pattern fusion, and data-free knowledge distillation. They are examined in terms of their ability to enhance accuracy, effectiveness, scalability, and sematic understanding. Despite the advantages, these methods face challenges such as limited datasets, poor model interpretability, and high computational overhead, hindering their real-world deployment. To solve those issues, future research directions are conducted, including self-supervised learning, cross-chain data expansion, model simplification, and Explainable Artificial Intelligence (XAI). In conclusion, the application of ML in smart contract threat detection provides a scalable and intelligent framework.

INTRODUCTION

Blockchain technology is widely used in various industries to ensure decentralized and transparent transactions. Smart contracts, which ensure contract clauses are written in code and execute automatically when standards are met, play a significant role in applications such as decentralized finance (DeFi), the Internet of Things (IoT), voting systems in government, and supply chain management [1, 2]. For instance, DeFi platforms include Uniswap and Curve, which allows transparent transactions without middlemen, and dYdX and Compound, which provides Flash Loans that users can borrow without collateral but must repay within the same transaction [3].

The concept of smart contracts was first proposed by Nick Szabo in 1994, who defined it as a computerized transaction protocol that executes the terms of a contract [4]. After decades, the connection between blockchain and smart contracts was popularized by the development of Ethereum [5]. Ethereum is a typical representative of smart contract usage [6]. Solidity, a Turing-complete programming language, allows Ethereum a better scripting capability than Bitcoin, ensuring the low transaction fee and automatic execution [7].

However, due to the rapid adoption and immutability of smart contracts, the challenges of security vulnerability would result in crucial economic losses, making it face various security concerns. For example, the infamous TheDAO hack in 2016 caused the loss of approximately 60 million because of a Re-entrancy Attack [8]. Moreover, the attacks such as the Parity Wallet vulnerability in 2017 and the dForce attack in 2020 which caused 150 million dollars and 25 million dollars in loss respectively. Those incidents demonstrate that blockchain could be hacked by various methods. It is crucial to figure out a robust security mechanism to reduce the vulnerabilities in smart contracts.

To address these issues, using Machine Learning (ML) on smart contracts security has emerged as a promising approach. By applying ML, it allows automating vulnerability detection, reducing reliance on manual intervention, and enhancing the effectiveness of analysing large codebases [8].

In recent years, researchers worldwide have explored majority methods to enhance the security in smart contracts. ML algorithms represented by the learning modes such as supervised learning and unsupervised learning, have been implemented to detect the vulnerability more effectively. Specifically, techniques including Support Vector Machines (SVM), decision trees, Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) have been conducted for vulnerability checking and developing a stronger smart contract analysis tool [9, 10].

Due to the rapid development of ML in this field. This paper aims to provide a comprehensive review of the applications of ML on smart contract security. To be more specific, this review systematically categorizes different ML-based methodologies and analyses their effectiveness in detecting vulnerabilities and enhancing the security of smart contracts. In addition, this paper discusses the key challenges that ML encounters in smart contract security, including the lack of a majority of available datasets, generalization performance, and limitations in complex deep learning models [11, 12]. Those challenges primarily create resistance to the application of ML in smart contracts. For the future outlook, this paper highlights several research directions that could enhance the application of ML, scalability, efficiency, and trustworthiness of ML-powered smart contract security mechanisms.

PRELIMINARIES OF SMART CONTRACT AND MACHINE LEARNING

Smart Contract

The smart contract aims to enhance the trust, efficiency, and transparency of transactions on blockchain platforms like Ethereum. The key components of smart contracts include 1) programming code, typically written in Solidity to define the rules and logic agreements. 2) A distributed ledger, which is also the fundamental feature of blockchain. It allows the status of contracts are immutable when storing records [13]. 3) A trigger mechanism: when meeting the standard or agreement, the contract executes automatically based on pre-set conditions. For instance, if person A and person B made an agreement of punishment on contract, if one of them triggered that condition, that person would automatically receive the punishment (e.g., cancel the transaction or pay extra gas).

The workflow of smart contracts contains several steps that are shown in Figure 1. Firstly, the user writes a smart contract in a programming language (e.g., Solidity) and defines the rules and conditions for execution. Once written, it would automatically deploy on blockchain and be handled by miners through a transaction (Tx_{create}). Smart contracts can be called by users (Tx_2) or other contracts (Tx_1) and triggered by them. When triggered, miners validate, execute, and update its status on World State so that all participants could know the result of each smart contract. After successfully executing, the transaction is added to a new block in blockchain (e.g., $block_{i+2}$) ensuring immutability and public verifiability through consensus mechanisms [14].

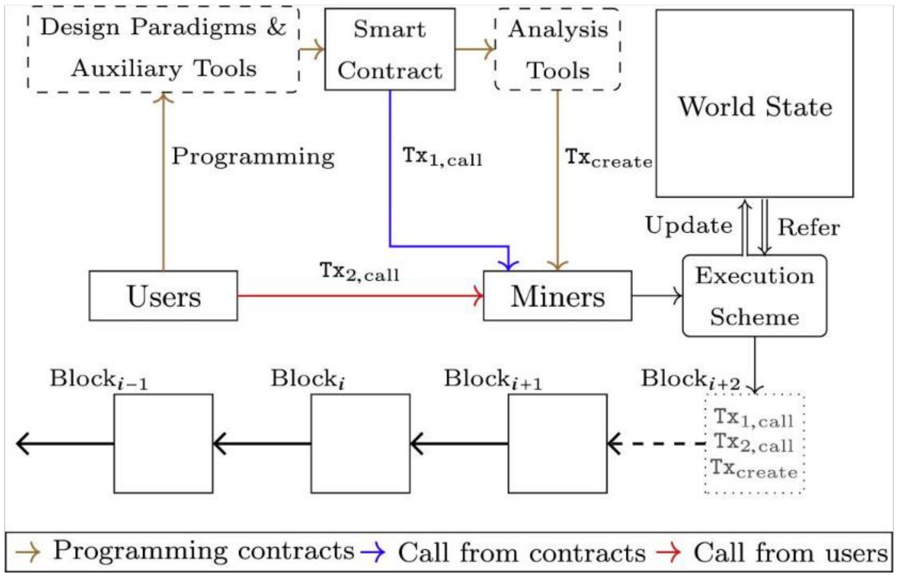

FIGURE 1. Workflow of smart contracts in blockchain [14].

Due to the transparency and immutability, smart contracts are getting attacked by hackers who exploit code vulnerabilities, such as re-entrancy attacks and integer overflows. Therefore, applying advanced technologies like ML to analysing and detecting those vulnerability become more and more essential.

Machine Learning

Machine Learning is a subfield of Artificial Intelligence (AI) that targets enhancing the efficiency and accuracy of problem-solving and decision-making based on training data [15].

As Sculley et al. highlighted, the implementation of machine learning relies on multiple components, including data dependencies, model architecture, feature engineering, configuration, and Infrastructure [16]. Those processes revealed the reliance between each stage during the ML lifecycle. For instance, the feature engineering that transfers the raw data into informative data directly influences the efficiency of ML while configuration has an impact on accuracy due to the number of config lines exceeding the lines of code [16].

Figure 2 demonstrates a typical workflow of ML, containing several steps from problem defining to deployment and inference. Firstly, the prediction, regression, clustering, or decision-making of the task is defined clearly in step 1. In step 2, the data for training needs to be collected from diverse resources. After collecting, step 3 involves data analysis, which includes preprocessing and feature extraction, ensuring the data is ready for use. In step 4, a proper model is trained based on the processed data, undergoing offline training and tuning. After training, the model experiences validation to evaluate its performance. If any error occurs or it fails to meet requirements, it loops back to Step 2 for adjustment and retraining. Finally, the model is deployed into environments so that it can make decisions and detections in a real-time application [17]. The workflow reflects the foundation for building a scalable and reliable ML-based system.

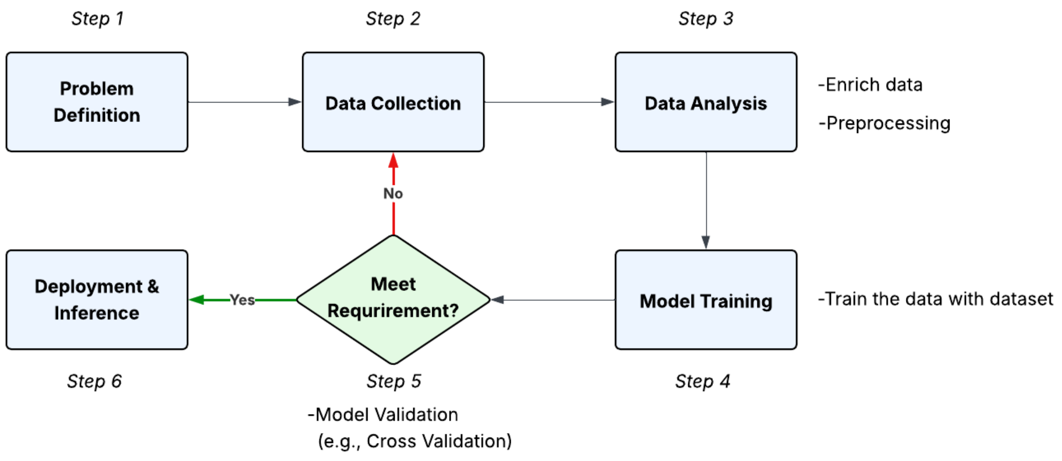

FIGURE 2. The typical workflow of machine learning for networking (Picture credit: Original).

ML-BASED APPROACHES IN DETECTING VULNERABILITIES AND ENHANCING THE SECURITY OF SMART CONTRACTS

Deep Learning-Based Methods

GNN-based Model

Zhuang et al. conducted a contract graph with two different models for classification: Degree-Free GCN (DR-GCN) and Temporal Message Propagation Network (TMP) [18]. DR-GCN, which based on GCN improves the effectiveness of learning on a normalized graph by removing the diagonal node degree matrix. TMP generates predictions by broadcasting information across the edges of time. By conducting over 300,000 functions on ESC (Ethereum Smart Contracts) and VSC (VNT chain Smart Contracts), TMP reached the highest performance in terms of accuracy, recall, precision, and F1 score across three vulnerability types (re-entrancy, timestamp dependency, infinite loops). Those results highlight the potential of GNN-based methods for smart contract security [18].

Further advancing GNN architecture, MANDO, which is a multi-level heterogeneous graph learning framework proposed by Nguyen et al. [19]. MANDO aims to discover the smart contract vulnerabilities by applying heterogeneous contract graphs, which integrate both control-flow and call relations to capture the semantic information. The multi-level graph neural network is employed to extract dynamic metapaths, combining topological GNNs and a novel node-level heterogeneous attention network. This component consists of a two-phase detection system: Coarse-Grained Detection and Fine-Grained Detection. These phases make a significant improvement compared to previous models, which only operated at the function level. The result of MANDO achieves the F1-score of 70.81% and exact lines of vulnerable code with F1-score of 90.51%.

To enhance the interpretability of GNNs, Liu et al. introduced Attentive Multi-Encoder (AME), which combined strengths of graph neural networks and local expert pattern extraction for vulnerability detection [20]. This method aims to address the drawback of traditional neural networks, which is the lack of incorporation and low explainability. AME consists of three core components. It begins with extracting expert patterns for three vulnerabilities (e.g., re-entrancy, Block timestamp dependence, Infinite loop). Then, smart contract source code transforms into a semantic control-data flow graph, in which global semantic features are generated by using the Temporal Message Propagation network (TMP). Next, AME applies the Attentive Multi-Encoder Network to combine the expert pattern, and graph features through a self-attention and cross-attention mechanism. This enables the improvement of prediction accuracy of the model and assigns interpretable weights to each input feature.

Sequence-Based Model

Shifting focus to sequence-based analysis, a Transformer-based method called Clear was proposed by Chen et al. to enhance the vulnerability detection of smart contracts through contrastive learning (CL) [21]. To enhance the understanding of semantic and structural elements, Contextual Augmentation i.e., masked language model (MLM), was applied for training. A contrastive loss function was deployed to cluster similar contracts and different ones are separated in the embedding space. During the vulnerability identification phase, the author concentrated on fine-tuning the Transformer model to ensure the accuracy of SCV. The Clear conducted an excellent result across over 40,000 real-world smart contracts, reaching the F1-score of 94.52%, which outperformed 13 cutting-edge tools, including DMT, TMP, and LineVul. Additionally, the Clear strengthens the performance of the RNN-based module by over 40%.

Complementing supervised sequence models, Wang and Jiang proposed a novel approach called AF-STip, which transfers knowledge from a complex teacher network to a lightweight student network without the required access to the primary training database [22]. This method starts with a data preprocessing phase, deploying the Word2Vec embedding method to extract feature vectors for smart contracts. The adaptive fusion module which combined local features and global modelling was used in the teacher network to extract the global vulnerability representations. These features are transferred into the student network by using the data-free knowledge distillation method, which utilizes prior knowledge training to improve the generalization performance of the student model. AF-STip achieves an F1-score of 91.16% across various vulnerability types, including reentrancy, timestamp dependency, delegatecall, and integer overflows [22]. Comparing with traditional static analysis methods such as Oyente and Mythril, as well as deep learning baselines like GCN, AF-STip performed better on accuracy, scalability, and strong ability to generalize via transfer learning without real data.

Hybrid Models

Bridging graph and sequence paradigms, Liu et al. proposed a method that integrates Graph Neural Networks (GNNs) with expert pattern extraction [23]. This modal constructs a Contract Graph that extracts the type of nodes into core nodes, normal nodes and callback nodes. Then node elimination is implemented to remove the normal nodes so that the model has a better ability to focus on the vulnerable code segments due to the reduction of noise. Temporal Message Propagation (TMP) Network was employed to capture the sequential independence and execution flow in smart contracts. By combining the Contract Graph with expert knowledge which demonstrates the typical vulnerability characteristics such as re-entrancy, timestamp dependency, and infinite loops. Experimental results illustrated that this hybrid approach significantly enhanced the precision of detecting vulnerabilities in smart contracts compared to traditional static analysis methods.

Traditional ML

Feature Engineering-Based Approaches

A representative feature engineering-based method is Eth2Vec, based on neural network static analysis proposed by Ashizawa et al. which is designed to address limitations of supervised learning methods in terms of the feature extraction robustness [24]. Ethn2Vec leveraged the Paragraph Vector—Distributed Memory (PV-DM) Model that could be implemented in natural language processing to encode the smart contract code into vectorized representations. EVM Extractor was developed as a language sequence to obtain the syntax and semantics of smart contracts. When Eth2Vec combined with Support Vector Machine (SVM) for vulnerability checks, the Eth2Vec-based system reached an average accuracy of 77.0%, better performing than other traditional supervised methods that rely on handcrafted features [24].

Unsupervised Behavior Analysis

Complementing supervised techniques, Agarwal et al. proposed a framework, which targets to detect the malicious smart contracts by combining the vulnerability analysis and temporal transaction behaviour [25]. The method begins with using the vulnerability analysis tools (e.g., Mythril, Slither, Oyente,) to scan the smart contracts code, followed by extracting the temporal behavioural features from both internal and external transactions. These features are divided into different time gap including 1-Day, 3-Day, 1-Month, ensuring the analysis changes over time. Multiple unsupervised machine learning algorithms are evaluated (e.g., K-Means, Spectral Clustering, One-Class SVM), K-Means stands out with the best performance. This study highlights the malicious behaviour varies across time, and vulnerability severity scores are useful in intra-cluster malicious SC detection.

DISCUSSION

Limitations and Challenges

Data-related Limitations

Although the applications of ML in smart contracts are expanding, the limitations and challenges occur at the same time. The most obvious field is data-related limitations, including a lack of diverse datasets, poor generalization, and imbalanced vulnerability distribution. The drawback of lacking a high-quality dataset restricts the evaluation and training for supervised and deep learning-based models. Specifically, the existing datasets of vulnerability are mostly focused on the well-known threats like re-entrance, while the new ones such as delegate call misuse and denial-of-service conditions, are hard to capture and limit in scoping and generalization, leading to class imbalance and skewed learning outcomes.

Additionally, most models illustrate the weakness in generalization capacity. They perform better on normal patterns than on obfuscated code, novel syntactic structures, or cross-chain contracts. In real-world scenarios, where consistent predictions are needed under various inputs, it is essential to ensure high robustness. Moreover, the quality of the label plays a vital role in model accuracy. As the vulnerability labels are often relied on, automated tools or manual writing results in potential label noise, which misleads the learning process.

Interpretability

Another challenge is model interpretability, especially for deep learning-based matrices. While some models, like AME, could output the reason for vulnerability by combining the expert rules and attention mechanism to achieve high accuracy and transparency. While others, such as CNN, RNN, and Transform, operate as block boxes that only detect the malicious behaviours without clear rationales or explanations, leading to restlessness and usage of such models in regulatory environments.

Computing Consumption

Furthermore, the computational complexity of architectures (e.g., GNNs and transformers) makes them hard to train due to the large, annotated corpora requirements and specialised hardware. These factors obstruct the actual deployment of the modals, especially in time-sensitive or on-chain verification systems, where scalability is significant.

Future Research

To address the challenges above, several research directions could be determined.

Firstly, it is necessary to construct a high-quality, diverse dataset. Future work should focus on building datasets that include both rare and emerging vulnerability types. These involve the cross-chain contract collection (e.g., BNB, Solana) as well as the consistent strategy that is based on expertise-reviewed labelling strategy. A sufficient dataset could enhance the generalisation ability.

Secondly, data-efficient learning strategies such as knowledge distillation and self-supervised learning present alternative strategies to conventional the traditional supervised learning. For example, Wang and Jiang emphasised AF-STip achieves high performance via data-free distillation without actual data. Similarly, Ashizawa et al. demonstrate that Eth2Vec reached high efficiency in feature transfer [20, 21]. Therefore, the future work could be based on those ideas and design a tailored self-supervised task for smart contract structures.

Thirdly, enhancing model interpretability remains the key priority. AME contributed to a strong foundation for combining the expert pattern with an attention-based mechanism for explanation. In the future models, the explainable AI (XAI) should be incorporated to offer reasoning traces, line-level notes and developer-friendly insights, thereby strengthening trust and usability [25].

Finally, simplifying the complexity of models' deployment scalability is prioritized. Although the models like Clear and MANDO scored high F1 scores, their architectures limit real-world deployment [22, 24]. To ensure high efficiency in deployment, techniques such as model pruning, compression, or hardware-aware architecture design can be implemented in real-world scenarios.

CONCLUSION

In this paper, a comprehensive review of ML-based technologies for smart contract vulnerability detection is conducted. The techniques were categorised into two groups: deep learning, and traditional ML. Methods including graph neural networks (GNNs), contrastive learning, expert pattern fusion, and knowledge distillation demonstrated the significance of enhancing the detection accuracy, scalability, and semantic understanding. After analysing, the evidence showed the limitations and challenges they encountered, such as data-related drawbacks, interpretability, and computational overhead. Addressing those limitations enhances the practical applicability of implementing the ML-based systems in real-world, time-sensitive environments.

The future research focusses on self-supervised learning, cross-chain vulnerability datasets, explainable AI (XAI) and simplified complexity, which enhance robustness and deploy ability. Overall, the ML-based smart contract guarantees scalability, accuracy, and efficiency when detecting the vulnerabilities or malicious behaviours in the blockchain environment.

REFERENCES

1. J. Abou Jaoude and R. G. Saade, “Blockchain applications–usage in different domains,” IEEE Access 7, 45360–45381 (2019).
2. 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 Gener. Comput. Syst. 105, 475–491 (2020).
3. F. Schär, “Decentralized finance: on blockchain and smart contract-based financial markets,” Rev. Fed. Reserve Bank St. Louis 103(2), 153–174 (2021).
4. N. Szabo, “The idea of smart contracts,” Nick Szabo’s Pap. Concise Tutor. 6, 199 (1997).
5. M. G. Vigliotti, “What do we mean by smart contracts? Open challenges in smart contracts,” Front. Blockchain 3, 553671 (2021).
6. H. Taherdoost, “Smart contracts in blockchain technology: A critical review,” Information 14(2), 117 (2023).

7. V. Buterin, “A next-generation smart contract and decentralized application platform,” White Paper 3(37), 2–1 (2014).
8. Y. Sun and L. Gu, “Attention-based machine learning model for smart contract vulnerability detection,” J. Phys.: Conf. Ser. 1820(1), 012004 (2021).
9. Y. Xu, G. Hu, L. You, and C. Cao, “A novel machine learning-based analysis model for smart contract vulnerability,” Secur. Commun. Netw. 2021(1), 5798033 (2021).
10. Z. Gao, “When deep learning meets smart contracts,” in Proc. 35th IEEE/ACM Int. Conf. on Automated Software Engineering, 1400–1402 (2020).
11. N. Atzei, M. Bartoletti, and T. Cimoli, “A survey of attacks on ethereum smart contracts (sok),” in Int. Conf. on Principles of Security and Trust, Springer, Berlin, Heidelberg, 164–186 (2017).
12. L. Luu, D. H. Chu, H. Olickel, P. Saxena, and A. Hobor, “Making smart contracts smarter,” in Proc. 2016 ACM SIGSAC Conf. on Computer and Communications Security, 254–269 (2016).
13. P. L. Seijas, S. Thompson, and D. McAdams, “Scripting smart contracts for distributed ledger technology,” Cryptol. ePrint Arch. (2016).
14. B. Hu, Z. Zhang, J. Liu, Y. Liu, J. Yin, R. Lu, and X. Lin, “A comprehensive survey on smart contract construction and execution: paradigms, tools, and systems,” Patterns (N.Y.) 2(2), 100179 (2021).
15. K. Sharifani and M. Amini, “Machine learning and deep learning: A review of methods and applications,” World Inf. Technol. Eng. J. 10(07), 3897–3904 (2023).
16. D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, et al., “Hidden technical debt in machine learning systems,” in Adv. Neural Inf. Process. Syst. 28 (2015).
17. C. Janiesch, P. Zschech, and K. Heinrich, “Machine learning and deep learning,” Electron. Markets 31(3), 685–695 (2021).
18. Y. Zhuang, Z. Liu, P. Qian, Q. Liu, X. Wang, and Q. He, “Smart contract vulnerability detection using graph neural networks,” in Proc. 29th Int. Joint Conf. on Artificial Intelligence, 3283–3290 (2021).
19. H. H. Nguyen, N. M. Nguyen, C. Xie, Z. Ahmadi, D. Kudendo, T. N. Doan, and L. Jiang, “MANDO: Multi-level heterogeneous graph embeddings for fine-grained detection of smart contract vulnerabilities,” in Proc. 2022 IEEE 9th Int. Conf. on Data Sci. and Adv. Analytics (DSAA), 1–10 (2022).
20. Y. Chen, Z. Sun, Z. Gong, and D. Hao, “Improving smart contract security with contrastive learning-based vulnerability detection,” in Proc. IEEE/ACM 46th Int. Conf. on Software Engineering, 1–11 (2024).
21. Z. Liu, P. Qian, X. Wang, L. Zhu, Q. He, and S. Ji, “Smart contract vulnerability detection: From pure neural network to interpretable graph feature and expert pattern fusion,” arXiv preprint arXiv:2106.09282 (2021).
22. L. Wang and W. Jiang, “Knowledge Migration Framework for Smart Contract Vulnerability Detection,” arXiv preprint arXiv:2412.11175 (2024).
23. 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,” IEEE Trans. Knowl. Data Eng. 35(2), 1296–1310 (2021).
24. N. Ashizawa, N. Yanai, J. P. Cruz, and S. Okamura, “Eth2vec: learning contract-wide code representations for vulnerability detection on ethereum smart contracts,” in Proc. 3rd ACM Int. Symp. on Blockchain and Secure Critical Infrastructure, 47–59 (2021).
25. R. Agarwal, T. Thapliyal, and S. K. Shukla, “Vulnerability and transaction behavior based detection of malicious smart contracts,” in Cyberspace Safety and Security: 13th Int. Symp., CSS 2021, Virtual Event, Nov. 9–11, 2021, Proc. 13, Springer, 79–96 (2022).