Original Article
ISSN (Online): 2350-0530
ISSN (Print): 2394-3629

HIGH LEVEL OF SECURITY AND CONTINUOUS MONITORING FOR ANALYZING SMART CONTRACT BEHAVIORS

Sangeetha R. [1], Dr. Veena M. N. [2]

[1] Assistant Professor, CIST, Manasa Gangortri, University of Mysore, Mysuru-570006, India
[2] Professor, Department of MCAPES College of Engineering, Shankar Gowda Road, Mandya-571401, India

Received 25 February 2025
Accepted 09 March 2025
Published 30 April 2025

Corresponding Author
Sangeetha R.,
sangeethauom9@gmail.com

DOI
10.29121/granthaalayah.v13.i4.2025.6031

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ABSTRACT

"Smart contracts" are software documented on block chains under specific circumstances that control the allocation of assets between individuals. In a smart healthcare supply chain, product traceability is a major issue. Two enabling technologies in the smart healthcare supply chain that ensure product traceability and safeguard against data manipulation are block chain and smart contracts. A smart contract workflow must be developed and carried out in a block-chain-based supply chain in accordance with the input data. This paper has an objective function to meet the entire system as a parallel composition of smart contracts and users this paper analyze the behavior of smart contracts and a core language of programs with an essential set primitive. The experimental results show that the proposed method can accurately detect security vulnerabilities and logic flaws in smart contracts through formal verification and other analysis techniques before smart contracts are deployed.

Keywords: Block Chain, Behavioral Contracts, Formal-Model, Smart-Contract, Stratified Datalog, Vulnerability

1. INTRODUCTION

Smart contracts are applications that operate on the blockchain. The block-chain is a revolutionary technology that both industry and academia. A smart contract can encode various rules for asset management within its source code. The established rules of a contract will be adhered to rigorously and automatically during its execution, embodying the principle that 'code is law' Liu et al. (2021). Smart contracts enable the automatic fulfillment of contract terms, thereby supporting intricate decentralized applications. smart contracts have been deployed in various blockchain platforms, enabling a wide range of applications that could serve in various domains, such as supply chain, healthcare, and the Internet of Things, might utilize it as their basis. This popularity can be related to its enduring and decentralized personality traits. Smart contracts running on block chains are dynamic programs that respond to events generated by external actors, triggering function calls. Participants can emit events in an asynchronous manner. Nonetheless, certain functionalities need to be limited to participants occupying designated roles within the system, roles that can change over time as the system develops Xu et al. (2025).

The smart supply chain can play an essential role using Internet-of-Things (IoT) and smart devices, during the pandemic situation, such as COVID-19. Blockchain is an instance of distributed shared ledger which employs distributed storage to maintain Bitcoin transaction data in distinguish blocks. Each block's data controls whether its sub-blocks are formed. However, Bitcoin's practical application value is restricted since to its merely single-function form and inability to develop intricate applications on the blockchain network. Blockchain provides data traceability and protection from tampering at the same time Rahman et al. (2020).

One of the most significant challenges in a block-chain based supply chain system is coping with the additional cost associated with implementing inaccurate smart contracts. likewise, it is challenging to regulate the dynamic flow of smart contracts. The control flow of smart contracts is not guaranteed to be immutable. The process for creating a smart contract workflow must be accurately completed prior to the deployment of associated smart contracts.

The interactions between smart contracts can lead to a growing network of interconnected contracts over time. As a result, predicting the behaviors of these contracts can be difficult. Furthermore, many current methods primarily concentrate on identifying possible logical flaws in smart contracts [Rahman et al. (2020),Rahman et al. (2021)].

Key Security Measures for Smart Contracts Yashavant et al. (2024)

1) **Formal Verification:**
   - Utilizing mathematical proofs to confirm the accuracy of smart contract logic.
   - Ensuring adherence to expected behaviors prior to deployment.
2) **Security Audits:**
   - Performing comprehensive code reviews by expert security auditors.
   - Detecting and addressing vulnerabilities before deployment.
3) **Best Coding Practices:**
   - Adopting secure coding methodologies.
   - Leveraging well-established libraries and frameworks.
   - Installing reentrancy guards and appropriate access control measures.
4) **Upgradable Smart Contracts:**

Implementing proxy contract frameworks to facilitate future upgrades without jeopardizing security.

5) **Multi-Signature Authorization:**

Mandating multiple approvals for critical operations to strengthen control and prevent unauthorized access.

1.1. CONTINUOUS MONITORING APPROACHES

1) **On-Chain Monitoring Tools:**
   - Employing blockchain analytics platforms to identify suspicious transactions and anomalies in real-time.
   - Monitoring contract interactions and fund transfers.
2) **Automated Threat Detection:**
   - Deploying AI-driven monitoring systems to recognize behavioral patterns and potential security threats.
   - Utilizing anomaly detection algorithms for proactive alerts.
3) **Logging and Alerts:**
   - Creating solid logging systems for tracking contract execution.
   - Establishing real-time alert mechanisms to inform administrators of unusual activities.
4) **Bug Bounty Programs:**
   - Motivating ethical hackers to uncover vulnerabilities through incentive programs.
   - Strengthening security via community-driven assessments and feedback.
5) **Regular Smart Contract Updates:**

Conducting periodic reviews and updates of smart contracts to tackle emerging vulnerabilities and enhance performance.

2. RELATED WORK

In the past few years of research authors suggested a novelty to analyze smart contract behaviors using formal methods with the aim of monitoring the security of high level potential. Some review articles have focuses on tools on detecting vulnerabilities rather than algorithms. One of the most cited motivation of the research is TheDAO attack Laneve et al. (2019).

In the paper Laneve et al. (2019) discusses the conduct of smart contracts and their engagement with outside participants is analyzed to optimize objective functions. We establish a fundamental programming language with a basic collection of smart contract elements and portray the entire system as a parallel arrangement of smart contracts and users. Consequently, we represent the behaviors of the system as a first logic formula in Presburger arithmetic and examine the highest profit for each participant by resolving arithmetic constraints.

In this paper Fu et al. (2023), we present the challenges associated with detecting re-entrancy attacks on Ethereum, focusing on the origins of these attacks, their behavioral traits, and the limitations of current detection techniques. Initially, we investigate the origins of re-entrancy attacks by examining the execution patterns of smart contracts in real transactions, and we highlight two shortcomings in the run-time detection of such attacks. Next, we chose actual re-entrancy attack transactions that have been reported officially and conducted a manual analysis of the smart contracts and their call sequences involved in these occurrences. We distilled two critical components of re-entrancy attacks and explored various types of these attacks, summarizing their behavioral characteristics from a theoretical perspective.

This paper Prajapati et al. (2023) introduces a defense mechanism that tracks the timestamps of DAOs between parent and child nodes, highlighting suspicious nodes that surpass a set threshold within a time period, blacklisting, and eliminating DAOs from detected malicious nodes. Additionally, it restricts the amount of DAO sent by a child node within a defined time frame to lessen the effects of an attack. The results of the experiments indicate that the insider attack on DAO adversely affects network performance (packet delivery ratio, average end-to-end delay, and throughput) across different intervals of DAO replay. The proposed defense mechanism reinstates optimal network performance with a high detection success rate.

This paper Xu et al. (2025) suggested utilizing dynamic condition response (DCR) graphs for both role-based and declarative access control in smart contracts, along with approaches for test-driven modeling and refinement of DCR graphs to facilitate the secure design and evolution of smart contracts. We demonstrate that these graphs enable the representation and visualization of a type of dynamic access control where access permissions change as the state of the contract advances. Their implementation aids in the clear declaration of access control permissions, enhances code auditing, supports test-driven modeling, allows for multiple iterations of smart contracts, and fosters a better understanding for users.

This paper Ding et al. (2020) introduces a method for dynamic monitoring and analysis at the function level for smart contracts, along with a prototype system implementation. The approach incorporates a "shadow stack" and associated data structures into the virtual machine of the testing block chain platform by examining the function management principles alongside the original stack. It then observes the byte code following code instrumentation, documenting the relationships of function calls and gathering relevant metrics such as time, instruction count, and gas consumption. The prototype system detects inefficient behaviors within contracts using visualization and intelligent analysis techniques, thereby creating an optimization feedback loop for smart contracts through iterative enhancements. Ultimately, the paper demonstrates the high feasibility and practicality of the monitoring and analysis method, as well as the performance of the prototype system, through experimental validation.

The authors in paper [Elia et al. (2022),Acquaviva (2025)] introduces a blockchain-based framework designed for the ongoing monitoring of applications, which facilitates the authorized deletion of IoT data in critical safety databases. The framework supports the implementation of data-evaluation policies that can identify redundant or outdated measurements within the database, thereby allowing them to be flagged for removal. The innovation of our method lies in the execution of the data-evaluation policy through a smart contract. In addition, utilizing a blockchain guarantees that essential operations in the database (such as deletions) are secure from tampering and adhere to the standards set forth by system stakeholders. We illustrate the effectiveness of the proposed framework through a real-world case study involving accelerometer data from a bridge monitoring application, while also assessing the transactional overhead caused by interactions with the blockchain.

This paper Alzahrani et al. (2021) facilitates innovation in collaborative business processes and encourages the RCM data market within the rail industry. By utilizing existing smart contract-based models for the trading and sharing of IoT data across blockchain networks, we pinpoint effective methods for enforcing agreements and ensuring equitable cost distribution among stakeholders without relying on a trusted third party. We provide an outline for a block chain-based RCM data audit framework, specifying appropriate data access agreements and accounting models in detail. Additionally, we evaluate three permissioned block chain platforms Hyperledger Fabric, Sawtooth, and Iroha for their feasibility in implementation. Lastly, the chapter discusses plans for future work focused on validating the tools using two industrial scenarios: monitoring systems for unattended overhead line equipment and axle bearings.

3. METHODOLOGY AND IMPLEMENTATIONS

The state-of-the-art performances in tasks related to programming language processing. During data prepossessing phase, collect different number of datasets and subsequently perform data cleaning. Finally, employ the proposed model to encode the data. Data cleaning is the length of a smart contract often varies based on its functions and intricacies is as shown in Figure 1. The source code is processed through tokenization, breaking it down into tokens that signify semantic units. After this, the tokenized data is converted into numerical formats, with each token assigned a distinct integer ID, creating the input token ID sequence. To fulfill the model's input specifications, padding and truncation techniques are utilized, guaranteeing a consistent sequence length.

Figure 1

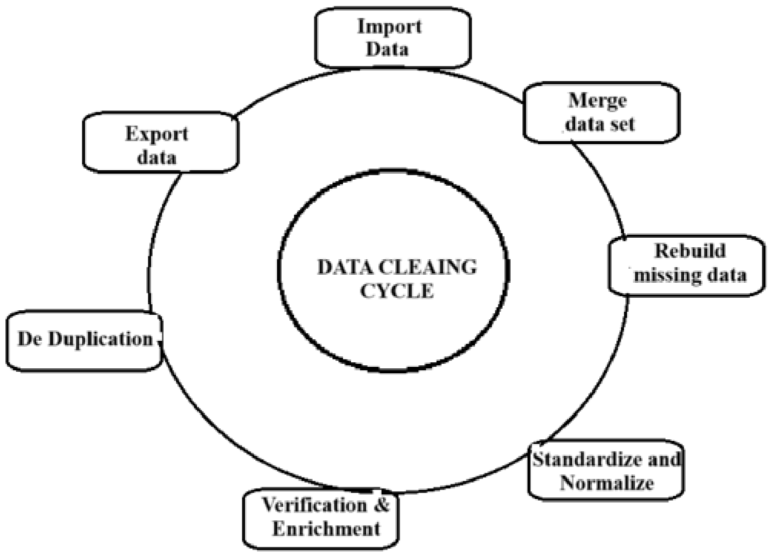

Figure 1 Data cleaning cycle

Certain complex contracts can go beyond several thousand tokens. However, managing lengthy text has historically presented difficulties in deep learning. Transformer models like BERT and GPT do have constraints when it comes to handling sequences that go beyond their token limit (typically around 512 tokens for numerous models). For Optimized LSTM and Optimized CNN models, direct processing of input IDs and masks is not feasible.

The proposed suggested model has been specifically adjusted to improve its alignment with the intended task by utilizing pre-processed input IDs and attention masks. Nevertheless, for the Optimized LSTM and Optimized-CNN models, we refrain from performing any fine-tuning on the proposed model regarding data pre-processing [Tang et al. (2023), Hwang et al. (2022)].

Figure 2

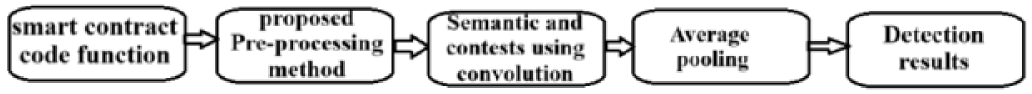
smart contract code function
proposed Pre-processing method
Semantic and contests using convolution
Average pooling
Detection results

Figure 2 Proposed Model

The proposed model is shown in Figure 2 built upon the Transformer architecture, which comprises multiple encoder layers. Prior to entering the encoder layers. The input data undergoes an embedding process followed by the encoding stage; fully connected layers are added for classification purposes. The proposed model for implementation is depicted in Figure 3.

Figure 3

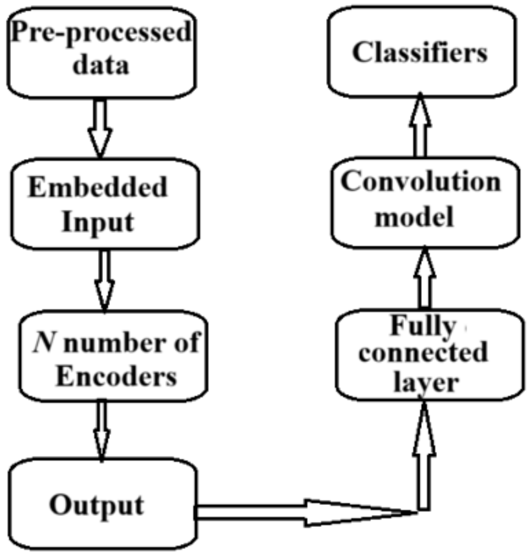

Figure 3 Optimized Proposed Model

In the training phase of this proposed model, the tokenizer utilizes embedding techniques that convert text or symbol data into vector forms. This procedure changes each word into a 512-dimensional word embedding. The suggested method aims to help the model grasp the positional information found within the sequence. It links each position to a particular vector representation to convey the relative positions of tokens throughout the sequence.

The suggested model utilizes multiple encoder layers to carry out deep representation learning. Each encoder layer consists of two sub-components: multi-head self-attention and a feed-forward neural network. The self-attention mechanism aids in capturing the relationships and dependencies among various positions in the input sequence. The feed-forward neural network is tasked with independently transforming and mapping the features for each position.

4. EXPERIMENTAL RESULTS AND DISCUSSION

In order to assure an unbiased assessment of various methods, we conducted training and testing in the same environments. Each experiment was carried out on a laptop equipped with system parameters are shown in Table 1.

Table 1

Table 1 System Device Parameters

SL no.	Device type	Parameter
1	Processor	13th Gen Intel(R) Core(TM) i7-1355U 1.70 GHz Installed RAM 16.0 GB (15.7 GB usable)
2	System type	64-bit operating system, x64-based processor
3	Software used	Python and MATLAB 2021b

4.1. EVALUATION METRICS

To assess different approaches, apply performance metrics such as accuracy, F1 score, recall, and precision. Accuracy represents the proportion of accurately predicted instances (which includes both true positives and true negatives) compared to the overall number of instances. It offers a broad indication of overall correctness.

The models are evaluated based on the metrics listed as

1) Precision

$$Precision = \frac{Total\ number\ of\ Positive\ instance}{Total\ number\ of\ positive\ instance\ +False\ Positive\ instance}$$

2) Recall

$$Recall = \frac{Total\ number\ of\ Positive\ instance}{Total\ number\ of\ positive\ instance\ +False\ negaticve\ instance}$$

3) F1 Score

$$F1\ score = \frac{2(precision\ X\ Recall)}{Precision + Recall}$$

4) Accuracy

$$Accuracy = \frac{Sum\ (Positive\ instance + Negative\ instance)}{Sum(Positive\ and\ Negative\ instance) + Sum\ (Faslse\ Positive\ nad\ Negative\ instance)}$$

The performance configuration parameters for the Optimized-proposed model is shown in Table 2.

Table 2

Table 2 Performance Configuration of Optimized-Proposed Model

SL no.	Parameters	Configuration
1	Learning Rate	0.0001
2	Dropout values	0.5
3	Epochs	60
4	Size for each batch	128
5	Hidden dimension	128

6	Filter	64
7	Folds	5
8	Swapped node	NA
9	layer	2
10	Output dimensions	4

5. RESULTS

In labeling the invulnerable dataset, we excluded contracts that exhibit characteristics possibly indicating vulnerabilities. The characteristics identified for four categories of vulnerabilities are defined as follows Xu et al. (2025).

- ***Reentrancy:*** Contracts that include the call.value function may be susceptible to reentrancy vulnerability.
- ***Timestamp Dependency:*** Contracts featuring block.timestamp and now variables may have a vulnerability related to timestamp dependency.
- ***Unchecked Low-Level Calls:*** Contracts that utilize end, call, callcode, and delegatecall functions can lead to unchecked low-level calls vulnerability.
- ***Tx.origin:*** Contracts that reference the tx.origin variable might be at risk of Tx.origin vulnerability.

The comparison of performance evaluation metrics results are shown in Table 3and Table 4. The pictorial chart has been drawn is shown in Figure 4and Figure 5.

Table 3

Table 3 Comparison of Evaluation Metrics Results Models

Sl no.	Evaluation Metrics	Optimized CNN in %	Optimized LSTM in %	Proposed optimized model in %
1	Precision	71.76	73.61	95.78
2	Recall	71.36	64.06	92.56
3	F1 Score	70.62	63.05	92.54
4	Accuracy	85.54	81.96	95.77

Figure4

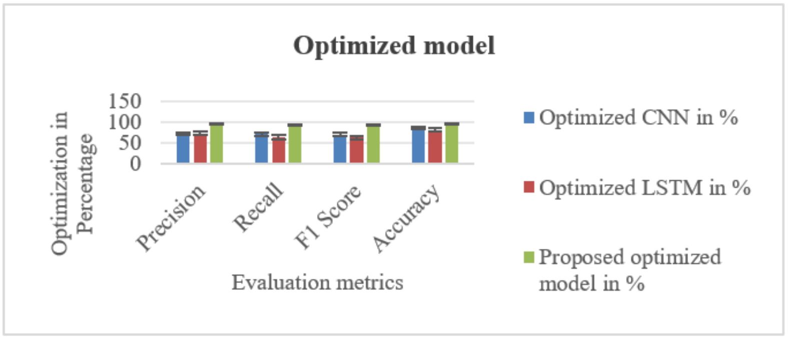

Figure 4 Comparison of Evaluation Metrics Results Models.

The Comparison of Evaluation Metrics Results for each type of Vulnerability is shown in Table 4.

Table 4

Table 4 Comparison of Vulnerability Metrics

Vulnerability	Precision	Recall	F1 score	Accuracy
Reentrancy	84.49	88.46	86.3	91.56
Time stamp dependency	91.43	98.92	93.94	97.29
Unchecked Low-Level Calls	99.98	99.87	99.66	96.32
Tx.origin	99.96	86.06	92.47	98.94

Figure 5

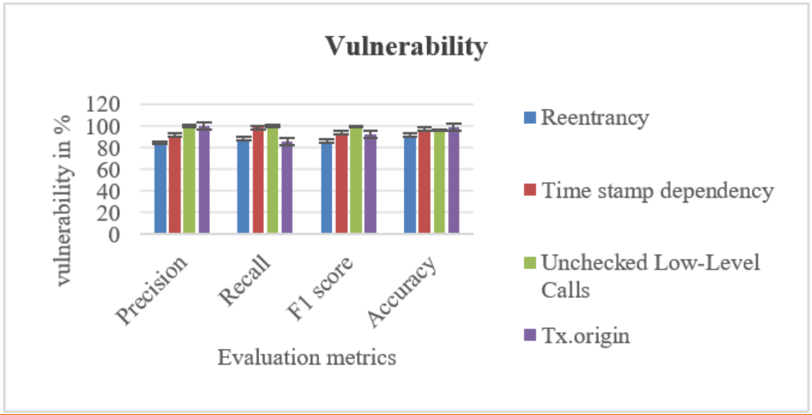

Figure 5 Comparison of Vulnerability Metrics

6. CONCLUSIONS

In this paper, we introduced a novel method for detecting vulnerabilities in smart contracts that emphasizes a high level of security and ongoing monitoring for assessing smart contract behaviors. According to the evaluation results, we demonstrated that the suggested prototype outperformed existing methods in terms of speed. Across a range of vulnerability types, this work illustrated that the proposed technique exhibits superior effectiveness in detection performance and quicker detection times compared to leading smart contract vulnerability detection tools. On average, proposed prototype showed the best performance on precision 95.78%, accuracy 95.77, recall 92.56 and F1-score with 92.54% respectively. Also, the proposed takes 100ms which is at least 12 times faster compared to the other tools.

CONFLICT OF INTERESTS

None.

ACKNOWLEDGMENTS

None.

REFERENCES

Acquaviva, A. (2025). Smart Contracts for Certified and Sustainable Safety-Critical Continuous Monitoring Applications.

Alzahrani, R. A., Herko, S. J., & Easton, J. M. (2021). Blockchain-Hosted Data Access Agreements for Remote Condition Monitoring in Rail. The Journal of The

British Blockchain Association. https://doi.org/10.31585/jbba-4-2-(3)2021

Ding, Y., Wang, C., Zhong, Q., Li, H., Tan, J., & Li, J. (2020). Function-Level Dynamic Monitoring and Analysis System for Smart Contract. IEEE Access, 8, 229161-229172. https://doi.org/10.1109/ACCESS.2020.3046005

Elia, N., Barchi, F., Parisi, E., Pompianu, L., Carta, S., Bartolini, A., & Acquaviva, A. (2022, August). Smart Contracts for Certified and Sustainable Safety-Critical Continuous Monitoring Applications. In European Conference on Advances in Databases and Information Systems (pp. 377-391). Springer International Publishing. https://doi.org/10.1007/978-3-031-15740-0_27

Fu, J., Liu, W., Zeng, C., & Huang, W. (2023, May). On the Detection Limitations of the Re-Entrancy Attacks on Ethereum. In International Conference on Computational & Experimental Engineering and Sciences (pp. 59-72). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-44947-5_5

Hwang, S. J., Choi, S. H., Shin, J., & Choi, Y. H. (2022). CodeNet: Code-Targeted Convolutional Neural Network Architecture for Smart Contract Vulnerability Detection. IEEE Access, 10, 32595-32607. https://doi.org/10.1109/ACCESS.2022.3162065

Laneve, C., Coen, C. S., & Veschetti, A. (2019). On the Prediction of Smart Contracts' Behaviors. In From Software Engineering to Formal Methods and Tools, and Back: Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday (pp. 397-415). Springer International Publishing. https://doi.org/10.1007/978-3-030-30985-5_23

Liu, Z., Qian, P., Wang, X., Zhuang, Y., Qiu, L., & Wang, X. (2021). Combining Graph Neural Networks with Expert Knowledge for Smart Contract Vulnerability Detection. IEEE Transactions on Knowledge and Data Engineering, 35 (2), 1296-1310. https://doi.org/10.1109/TKDE.2021.3095196

Prajapati, A. K., Pilli, E. S., Battula, R. B., & Verma, A. (2023, October). A Defense Solution to Secure Low-Power and Lossy Networks Against Dao Insider Attacks. In TENCON 2023-2023 IEEE Region 10 Conference (TENCON) (pp. 261-266). IEEE. https://doi.org/10.1109/TENCON58879.2023.10322374

Rahman, M. S., Khalil, I., & Bouras, A. (2020). Formalizing Dynamic Behaviors of Smart Contract Workflow in Smart Healthcare Supply Chain. In Security and Privacy in Communication Networks: 16th EAI International Conference, SecureComm 2020, Washington, DC, USA, October 21-23, 2020, Proceedings, Part II (pp. 391-402). Springer International Publishing. https://doi.org/10.1007/978-3-030-63095-9_25

Rahman, M. S., Khalil, I., & Bouras, A. (2021, January). A Framework for Modeling Blockchain-Based Supply Chain Management System to Ensure Soundness of Smart Contract Workflow. In HICSS (pp. 1-10). https://doi.org/10.24251/HICSS.2021.675

Tang, X., Du, Y., Lai, A., Zhang, Z., & Shi, L. (2023). Deep Learning-Based Solution for Smart Contract Vulnerabilities Detection. Scientific Reports, 13 (1), 20106. https://doi.org/10.1038/s41598-023-47219-0

Xu, Y., Slaats, T., Düdder, B., Hildebrandt, T. T., & Van Cutsem, T. (2025). Safe Design and Evolution of Smart Contracts Using Dynamic Condition Response Graphs to Model Generic Role-Based Behaviors. Journal of Software: Evolution and Process, 37 (1), e2730. https://doi.org/10.1002/smr.2730

Yashavant, C. S., Chavda, M., Kumar, S., Karkare, A., & Karmakar, A. (2024). SCRUBD: Smart Contracts Reentrancy and Unhandled Exceptions Vulnerability Dataset. Arxiv Preprint arXiv:2412.09935.