JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A SMART CONTRACT VULNERABILITY DETECTION MECHANISM BASED ON DEEP LEARNING AND EXPERT RULES

[1] Pendota Alekhya, [2]M.Sai Pranavi, [3]T.Harshita,[4]Mrs.Megha Dabas
[123]UG Scholars,[4]Asst. Professor
[1234]Department of Computer Science Engineering (Internet of Things)
Guru Nanak Institutions Technical Campus (Autonomous), Hyderabad, India

Abstract : Conventional methods for detecting smart contract vulnerabilities depend on predefined expert criteria, which limits their accuracy, scalability, and generalizability. Although some deep learning algorithms can encode real expert knowledge and remain comprehensible, most do not solve these problems. To increase the possibility of developing detect smart contract vulnerabilities and overcome the shortcomings of both detection methods, in this project we propose a step-by-step smart contract vulnerability detection mechanism using neural networks Graph and expert deep learning models. Testing shows that our vulnerability detection technique detects vulnerabilities on average six times better than the original deep learning model and the following transactions Contracts involving potentially risky activities can also be blocked at the Ethernet level by the second stage of the verification mechanism.

***IndexTerms* - Smart Contract Vulnerabilities, Detection Methods, Conventional Approaches, Expert Criteria, Deep Learning Algorithms, Neural Networks, Graph Models, Vulnerability Detection Mechanism, Testing Results, Risky Activities, Transaction Blocking, Ethereum Network, Verification Mechanism, Scalability, Accuracy, Generalizability.**

I. INTRODUCTION

In the past few years, many people have been interested in blockchain technology and digital money. Blockchain technology is becoming better and cryptocurrencies are becoming more popular. It works with decentralized consensus protocols and proof of work mechanisms to make transactions transparent and easy to change. Smart contracts are computer protocols that can be used in a wide range of applications and services. But smart contracts are becoming more complicated because they are becoming more popular. This makes it more important to keep them safe from mistakes. The gas mechanism, the delegate call mechanism, the exception passing mechanism, and some other special smart contract mechanisms are all used to create smart contracts. Even though these qualities have helped smart contracts spread and develop quickly on the blockchain, they have also led to different vulnerabilities in many smart contracts that are based on them. In the case of Ethereum, even though modern contracts can be updated by Create2 to prevent vulnerabilities, the maintenance cost of frequent contract updates is not easy and the damage caused by a vulnerability attack can no longer be recovered. It requires a thorough security review of the smart contracts to be used. There are two main types of smart contract vulnerability detection methods: traditional methods that use experts' knowledge and some automated tools, and newer methods that use deep learning. Typical ways of doing things use rules from experts that can be wrong and are getting harder to make because more people want them. Because of the fixed model, it is hard to guarantee that smart contracts written and generated with ulterior goals will bypass certain established restrictions, such as Oyente, Securify, etc.

Deep learning-based models detect vulnerabilities with higher detection efficiency and better generalization, but they are dependent on datasets, can't easily encode valid expert knowledge, and most methods are hard to understand. To make smart contracts better, we use graph neural networks (GNNs) in deep learning. These networks can detect weaknesses in smart contracts. They can also detect and stop dangerous transactions at the EVM level. This makes smart contract vulnerability checking better.

II. LITERATURE SURVEY

S. He, Y. Lu, Q. Tang, G. Wang, and C. Q. Wu provide an overview of blockchain technology, focusing on its decentralization, security, and trustworthiness. They introduce blockchain classification and architecture, analyzing its core principles and function in each layer, particularly in relation to Bitcoin. The paper discusses the current level of blockchain technology development and applications, along with the challenges and future trends.

Y. Ni, C. Zhang, and T. Yin address security vulnerabilities in smart contracts, crucial for driving Web 3.0 in areas like digital finance. They categorize vulnerabilities and analyze detection techniques, comparing existing tools based on various criteria. The paper presents the Co-Governed Sovereignty Multi-Identifier Network (CoG-MIN) as a case study, emphasizing the importance of smart contract security in Web 3.0, and suggests future research directions.

Y. Zhang and J. Ma focus on Ethereum Solidity smart contracts, outlining their mechanisms, common vulnerabilities, and traditional detection tools such as symbolic execution and formal verification. They also discuss recent machine learning-based vulnerability detection methods and suggest improving detection efficiency and standardizing information databases.

A. Joshi, H. Kebriaei, V. Mariani, and L. Glielmo propose a method to detect smart contract vulnerabilities by combining graph neural networks with expert knowledge. They explain how they transform the source code into a contract graph, normalize it, and then employ a temporal message propagation network to extract features. Results show significant accuracy improvements over existing methods, especially for reentrancy, timestamp dependence, and infinite loop vulnerabilities.

N. V. Chawla and D. A. Davis introduce Hessian graph convolutional networks (HesGCN), addressing the poor extrapolating ability of traditional GCN due to unchanged null space of Laplacian along the manifold. HesGCN optimizes one-order spectral graph Hessian convolutions, incorporating richer null space information to improve feature learning. Experimental results validate its superiority over existing methods in semi-supervised learning tasks.

R. Sanchez-Marquez and J. M. J. Vivas systematically review security vulnerabilities in Ethereum blockchain smart contracts, emphasizing their significance due to potential financial losses. They discuss detection tools, real-life attacks, and preventive mechanisms. Comparisons among analysis tools are provided, and future research directions are suggested to address issues in Ethereum blockchain-based smart contracts.

III. Proposed System

Chart Neural Systems (GNN) and Master Designs: Utilizes GNN in profound learning for shrewd contract powerlessness detection. Collaborates with master models to make strides discovery proficiency and generalization. Aims to relieve interpretability concerns of profound learning by combining conventional master rules with profound learning methods. The GNN demonstrate and master demonstrate work together to improve shrewd contract screening. Introduces imperative rules for particular powerlessness sorts to square unsafe exchanges at the Ethereum Virtual Machine (EVM) level. Proposed Framework Advantage Gives a more steady shrewd contract environment .Enhances the adequacy of shrewd contract powerlessness recognizable proof.

IV. System Architecture

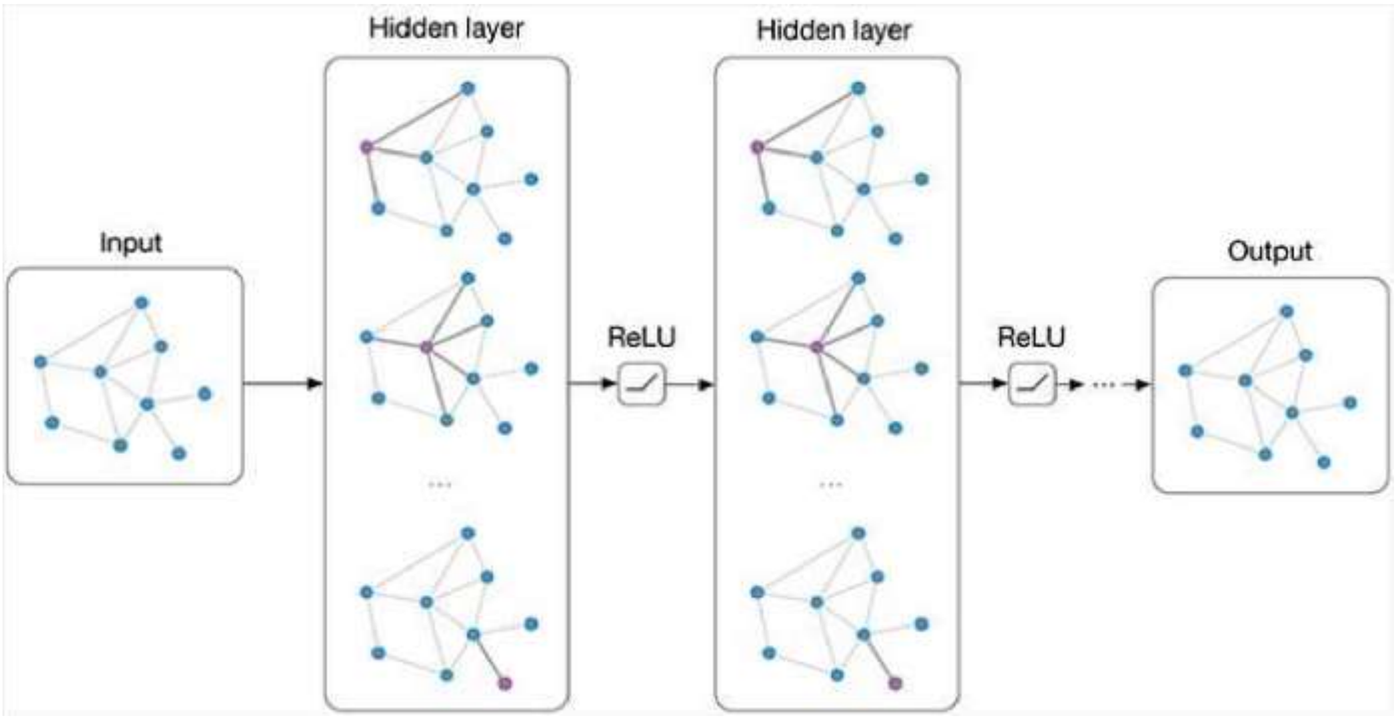


Fig.1.System Architecture

V. Hardware Requirements

The hardware requirements serve as a comprehensive specification for the system implementation and are essential for initiating the system design process. They outline what the system should encompass rather than dictating how it should be executed.

- Processor: Dual-core Intel Core 2 Duo
- RAM : 4GB DDR RAM
- Hard Disk: 250GB Storage Capacity

VI. Software Requirements

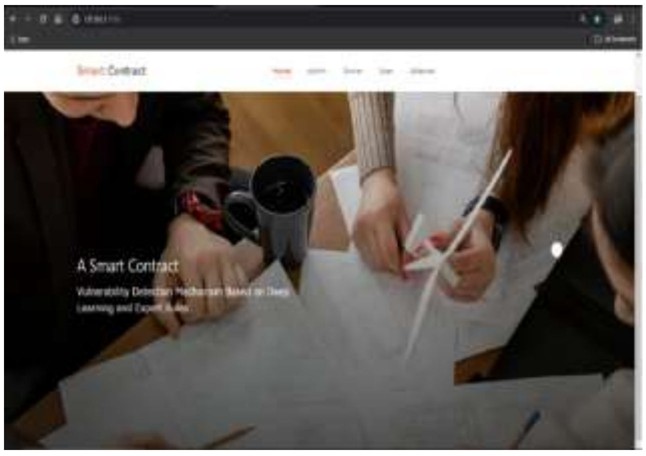
The software requirements document serves as a specification for the system, outlining what the system should accomplish rather than detailing the methods for achieving it. It provides a foundation for creating the software requirements specification and is instrumental in estimating costs, planning team activities, performing tasks, and tracking progress throughout the development process.

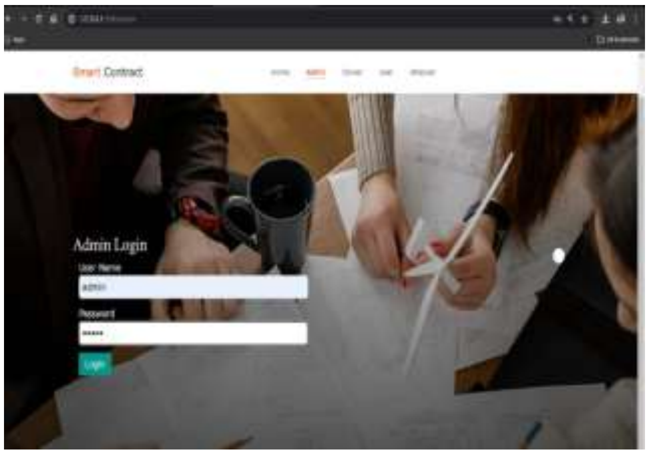
- Operating System: Windows 7/8/10
- Platform: Spyder3
- Programming Language: Python
- Front End: Spyder3

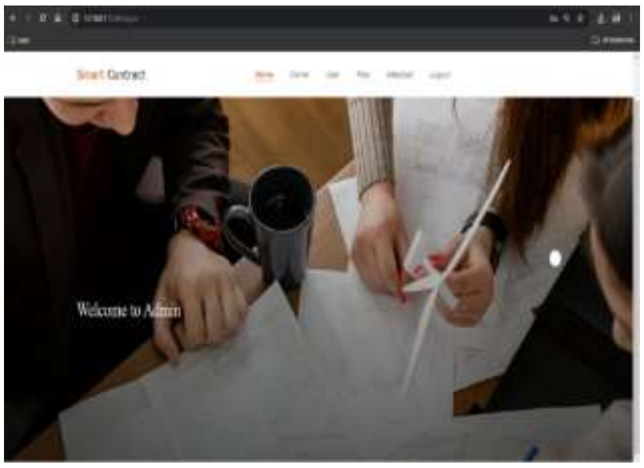
VII. Future Enhancement

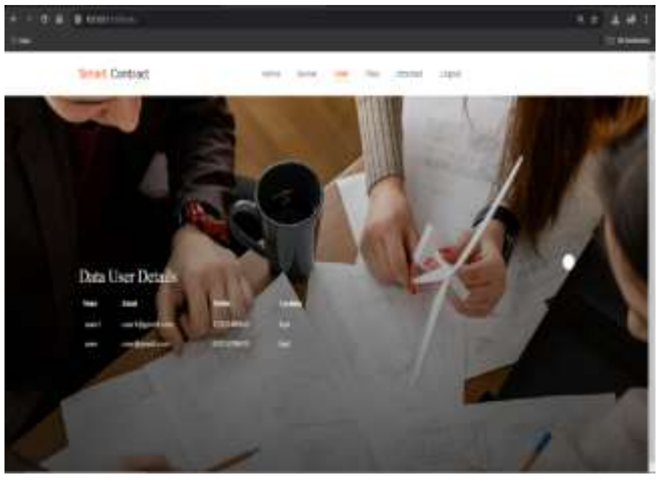
[1**]. Progressed Chart Neural Systems (GNNs):** Proceeded inquire about and advancement in GNNs might lead to more advanced models able of capturing indeed more nuanced vulnerabilities inside shrewd contracts.
[2]. **Crossover Approaches:** Combining conventional master rules with machine learning methods seem give a more comprehensive approach to powerlessness discovery. By leveraging the qualities of both strategies, such as the interpretability of master rules and the productivity of machine learning models, the generally location capabilities might be improved.
[3]. **Integration of Formal Confirmation:** Formal confirmation methods may be coordinates into the helplessness discovery handle to numerically demonstrate the rightness of savvy contracts and distinguish potential vulnerabilities at an early arrange of improvement

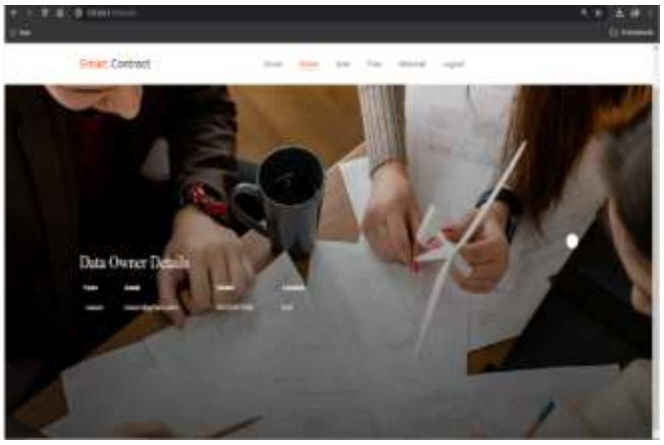
VIII. Snapshots

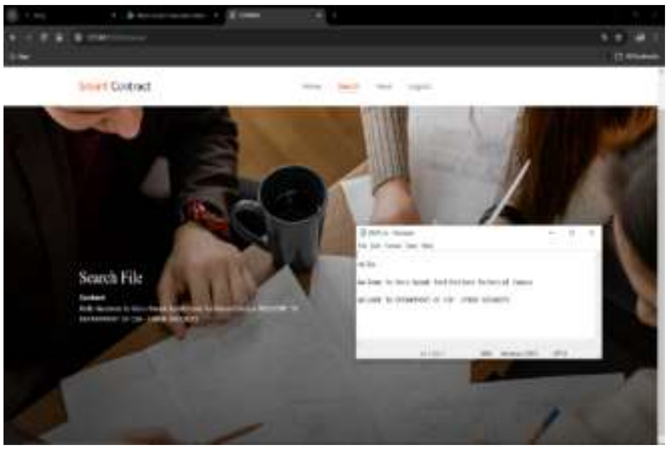

Smart Contract
A Smart Contract
Vulnerability Detection Mechanism Based on Deep Learning and Expert Rules

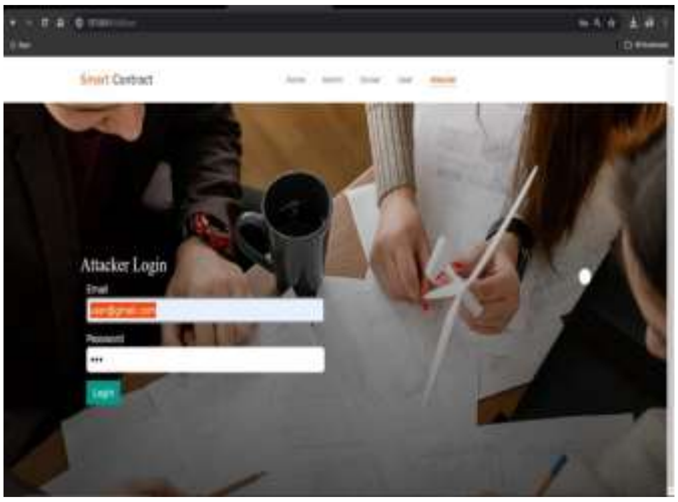


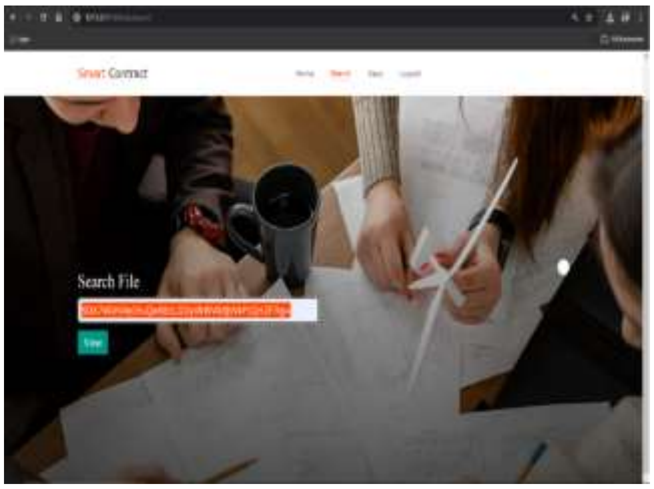


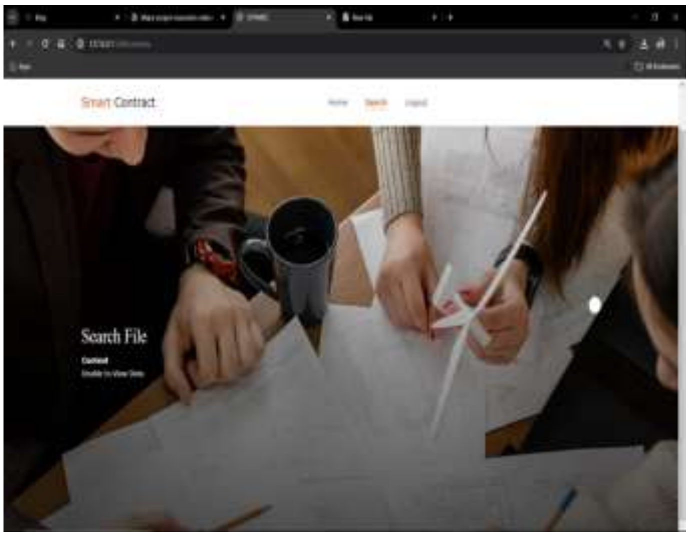


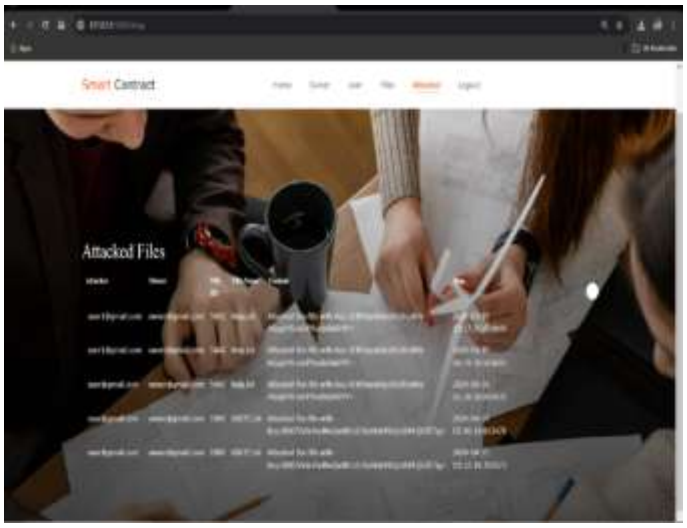












IX. Conclusion

This article describes the work of combining GNN models and EVM level expert models to detect vulnerabilities in smart contracts.The smart contract vulnerability detection technique presented in this study improves both its detection capabilities as a vulnerability detection tool and its ability to act as a stopper for contracts performing risky operations.Although more time consuming than EVM without the second phase of the protection mechanism, this mechanism allows you to decide whether to activate the second phase of the protection mechanism to avoid wasting resources such as time .User can choose.We believe that the research in this document will play an important role in the future development of a more secure and reliable smart contract environment.

X . REFERENCES

[1] J. Cheng, ''Current status and prospects of blockchain technology,'' in Proc. Int. Conf. Artif. Intell. Secur. Singapore: Springer, 2020, pp. 674–684.

[2] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, ''HotStuff: BFT consensus with linearity and responsiveness,'' in Proc. ACM Symp. Princ. Distrib. Comput., Jul. 2019, pp. 347–356.

[3] Y. Ni, C. Zhang, and T. Yin, ''A review of research on smart contract security vulnerabilities,''J. Inf. Secur., vol. 5, no. 3, pp. 78–99, 2020.

[4] V. Buterin, ''A next-generation smart contract and decentralized application platform,'' White Paper, vol. 3, no. 37, p. 37, 2014.

[5] S. Badruddoja, R. Dantu, Y. He, K. Upadhayay, and M. Thompson, ''Making smart contracts smarter,'' in Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC), May 2021, pp. 1–3.

[6] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Bünzli, and M. Vechev, ''Securify: Practical security analysis of smart contracts,'' in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Oct. 2018, pp. 67–82.

[7] A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck, ''Evaluating explanation methods for deep learning in security,'' in Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P), Sep. 2020, pp. 158–174.

[8] Y. Zhuang, ''Smart contract vulnerability detection using graph neural network,'' in Proc. IJCAI, 2020, pp. 3283–3290.

[9] J. Feist, G. Grieco, and A. Groce, ''Slither: A static analysis framework for smart contracts,'' in Proc. IEEE/ACM 2nd Int. Workshop Emerg. Trends Softw. Eng. Blockchain (WETSEB), May 2019, pp. 8–15.

[10] J. Gao, H. Liu, C. Liu, Q. Li, Z. Guan, and Z. Chen, ''EASYFLOW: Keep Ethereum away from overflow,'' in Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Companion (ICSE-Companion), May 2019, pp. 23–26.

[11] C. F. Torres and M. Steichen, ''The art of the scam: Demystifying honeypots in Ethereum smart contracts,'' in Proc. 28th USENIX Secur. Symp. (USENIX Secur.) Santa Clara, CA, USA: USENIX Assoc., 2019, pp. 1591–1607.

[12] R. Baldoni, E. Coppa, D. C. D'elia, C. Demetrescu, and I. Finocchi, ''A survey of symbolic execution techniques,'' ACM Comput. Surv., vol. 51, no. 3, pp. 1–39, May 2019.

[13] Y. Fu, M. Ren, F. Ma, H. Shi, X. Yang, Y. Jiang, H. Li, and X. Shi, ''EVMFuzzer: Detect EVM vulnerabilities via fuzz testing,'' in Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng., Aug. 2019, pp. 1110–1114.

[14] N. Ashizawa, N. Yanai, J. P. Cruz, and S. Okamura, ''Eth2 Vec: Learning contract-wide code representations for vulnerability detection on Ethereum smart contracts,'' in Proc. 3rd ACM Int. Symp. Blockchain Secure Crit. Infrastruct., May 2021, pp. 47–59.

[15] W. Wang, J. Song, G. Xu, Y. Li, H. Wang, and C. Su, ''ContractWard: Automated vulnerability detection models for Ethereum smart contracts,'' IEEE Trans. Netw. Sci. Eng., vol. 8, no. 2, pp. 1133–1144, Apr. 2021.