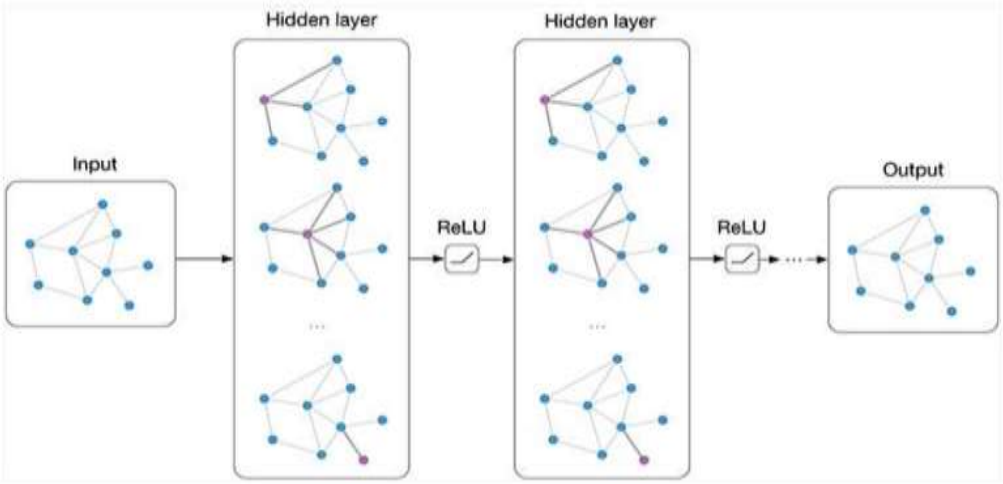
JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

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

[1]P. Anjaneyulu, [2]M. Soumya, [3]M. Sai Kumar, [4]P. Sunil Kumar
[1,2,3]UG Scholars, [4]Asst. Professor
[1,2,3,4]Department of Computer Science and Engineering,
Guru Nanak Institutions Technical Campus (Autonomous), Hyderabad, India.

Abstract : "Smart contracts are essential to blockchain-based systems, but they might have flaws that compromise their operation and security. This project offers a cutting-edge method for identifying smart contract vulnerabilities by fusing expert rules and deep learning algorithms. Our technology enhances security and dependability in decentralized systems by automatically detecting possible attacks by utilizing the inherent patterns and abnormalities found in contract code. We show through rigorous testing and validation that our approach is effective in identifying a broad variety of vulnerabilities, providing a strong defense for smart contract ecosystems." Our smart contract vulnerability detection tool, which combines deep learning models and expert rules, provides a proactive way to find and reduce vulnerabilities in blockchain ecosystems. Our solution correctly identifies possible vulnerabilities by examining code structures, semantic patterns, and historical data. This gives developers the ability to proactively mitigate security threats. By means of an extensive assessment on various smart contract datasets, we demonstrate the efficiency and scalability of our methodology in augmenting the resilience and reliability of decentralized applications. Our methodology is well-positioned to strengthen the trustworthiness of blockchain systems, encouraging increased trust and uptake in the emerging field of decentralized technology."Our novel method combines expert rules and deep learning algorithms to produce a powerful smart contract vulnerability detection system. Our technology is able to accurately and efficiently identify possible vulnerabilities in smart contracts on its own by utilizing domain-specific knowledge and neural networks. Our approach combines anomaly detection, semantic analysis, and feature extraction to give developers practical knowledge on how to protect their blockchain-based apps from security risks. Our solution's effectiveness and practicality in protecting decentralized ecosystems, promoting a more secure and dependable environment for blockchain innovation, have been confirmed by extensive testing and real-world case studies."

Our Shceme :



1. ***IndexTerms*** **-Smart Contracts,Vulnerability Detection,Deep Learning,Expert Rules,Ethereum,Blockchain Security, Machine Learning,Neural Networks.**

I. INTRODUCTION

Smart contracts provide automatic agreements and trustless transactions at the core of decentralized applications in the rapidly developing field of blockchain technology. But the security and integrity of decentralized systems are at risk since smart contract programming is inherently complicated and prone to weaknesses. In order to tackle this pressing problem, we put forth a novel strategy that combines expert rules and deep learning techniques to quickly and accurately identify smart contract weaknesses. Our

mechanism provides a comprehensive solution for preemptive vulnerability detection by utilizing neural networks to evaluate code patterns and expert-crafted rules to identify frequent errors. We hope to improve smart contracts' robustness and promote trust and dependability in decentralized ecosystems by combining these interdisciplinary cutting-edge technology. Our approach leverages deep learning models' ability to automatically learn and identify complex patterns in smart contract code, facilitating quick and precise vulnerability detection. Furthermore, the integration of expert guidelines obtained from a thorough examination of previous vulnerabilities offers a strong foundation for identifying known vulnerabilities and exploits. Our methodology creates a synergistic effect by combining these two methods, which improves the detection capabilities beyond what could be accomplished by either technique alone. We exhibit the effectiveness and dependability of our methodology in detecting a broad spectrum of vulnerabilities on various smart contract platforms by means of meticulous testing and validation. In the end, our work strengthens the basis of blockchain technology and encourages the mass use of robust and safe decentralized apps.

LITERATURE REVIEW

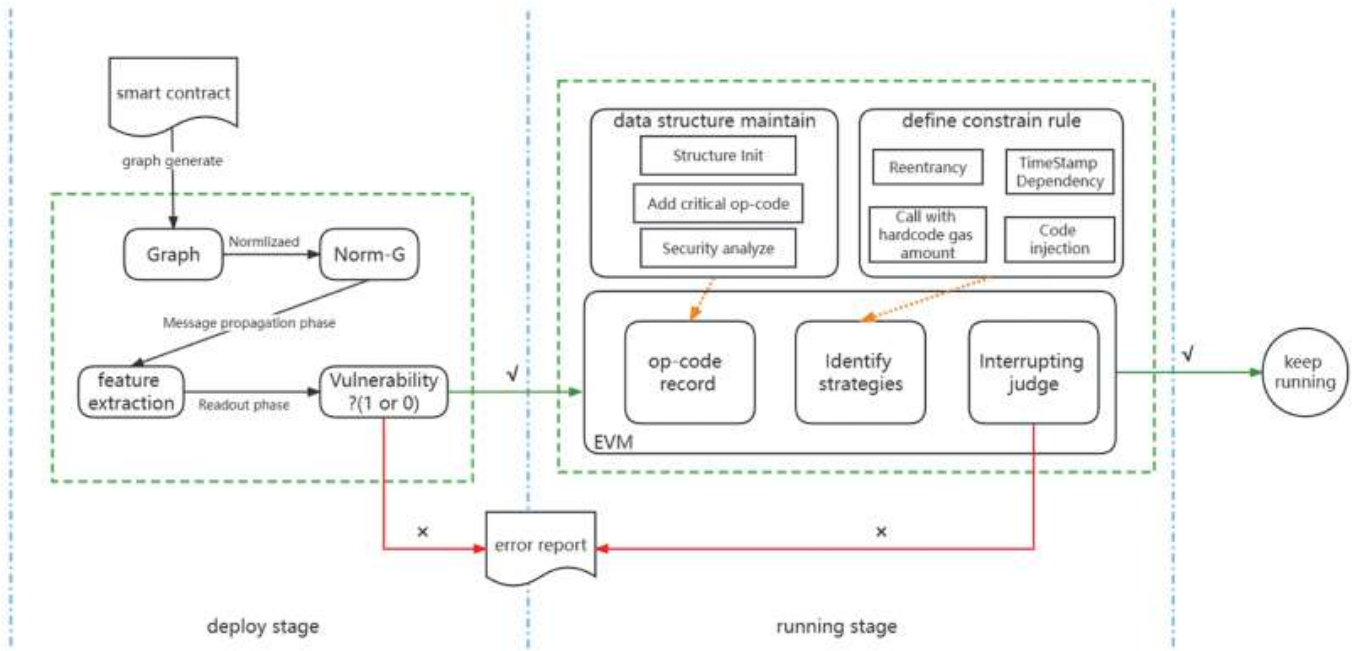
The irrevocable nature of smart contracts once they are published on blockchain networks makes vulnerability detection essential. Current techniques frequently rely on static analysis tools or human audits, which can be laborious and may miss small vulnerabilities. In order to improve vulnerability identification in smart contracts, researchers have looked into novel ways that combine deep learning techniques with expert guidelines. Numerous cybersecurity domains, such as virus detection and intrusion detection systems, have demonstrated the potential of deep learning. Deep learning models can automatically identify vulnerabilities in smart contract code by using neural networks to identify intricate patterns and abnormalities. Additionally, by adding more context and heuristics for spotting potential flaws, expert rules—which are derived from known vulnerabilities and best practices in smart contract development—can enhance deep learning models. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are two examples of deep learning algorithms that have been integrated with expert rules in a number of studies to uncover vulnerabilities in smart contracts. In order to tokenize and represent smart contract code for input into the neural network architecture, these frameworks usually require preprocessing processes. Expert guidelines, which direct the model in finding vulnerabilities based on typical coding errors, security pitfalls, and attack paths, are another way they incorporate domain-specific expertise. The research shows that deep learning and expert rules work well together to identify a range of smart contract vulnerabilities, such as reentrancy, integer overflow, and logic flaws. Using real-world smart contract repositories and benchmark datasets, researchers have assessed these methods and shown that they can reliably detect vulnerabilities while reducing false positives. Additionally, a few studies have looked into how well these techniques scale and work in decentralized contexts to handle large-scale smart contract analysis.

II. OUR APPROACH

Figure 1 shows the architecture of our A Smart Contract Vulnerability Detection Mechanism Based On Deep Learning And Expert Rules.

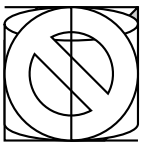
Our method may be separated into two stages: prior to and subsequent to the smart contract's deployment, as shown in FIGURE 1. A contract's deployment is stopped and a bug report is created and submitted if it is found to be unqualified after being tested for vulnerabilities using a component that uses the GNN model. The user must enter a value for the contract during the runtime phase. For high-value smart contracts, expert rules are employed to examine the sensitive opcodes engaged in risky operations. Should indications of a violation (i.e., a breach of the defined constraint rules) be found, the contract transaction will be interrupted, and an error report will be generated for submission.

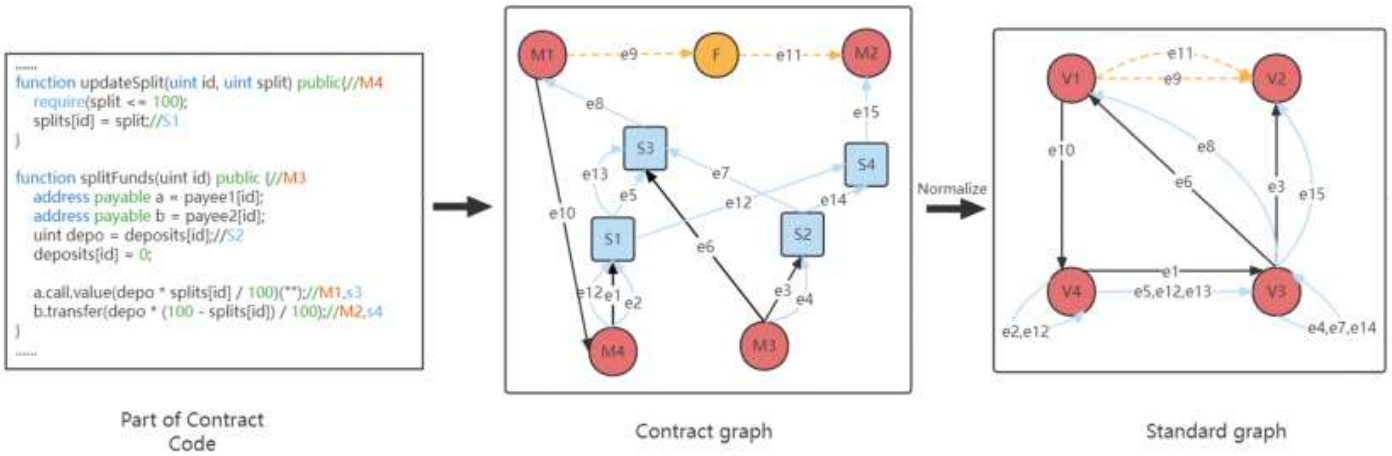
A smart contract that successfully completes the second round of vulnerability testing ensures that the transaction including the vulnerability will continue to function normally by continuing to execute the code without stopping its opcode execution.



III Graph Neural Network Component

Graph neural networks, or GNNs, are crucial parts of a vulnerability detection system for smart contracts. Smart contracts with intricate links can be captured by GNNs by taking advantage of the intrinsic graph structure seen in blockchain data. A GNN's architecture usually consists of a few essential parts. Initially, the graph model encodes smart contracts, with nodes representing variables or functions and edges representing dependencies. Node embedding layers capture semantic content by mapping nodes to high-dimensional vectors. Relationships between nodes are encoded by edge embedding layers. Convolutions are performed over the graph via graph convolutional layers, which aggregate data from nearby nodes and edges. Neighboring nodes' contributions are dynamically weighted using graph attention methods.





Convolutions are performed over the graph via graph convolutional layers, which aggregate data from nearby nodes and edges. Neighboring nodes' contributions are dynamically weighted using graph attention methods. Graph pooling layers allow decision making at different levels of abstraction by aggregating data from distinct graph components. Contracts having temporal or recursive structures are modeled by recurrent or recursive units. Final vulnerability predictions are generated by graph output layers using combined graph data. Together, these elements enable GNNs to precisely detect vulnerabilities and assess the complexities of execution flow.

III. TECHNIQUE USED OR ALGORITHM USED

3.1 EXISTING TECHNIQUE: -

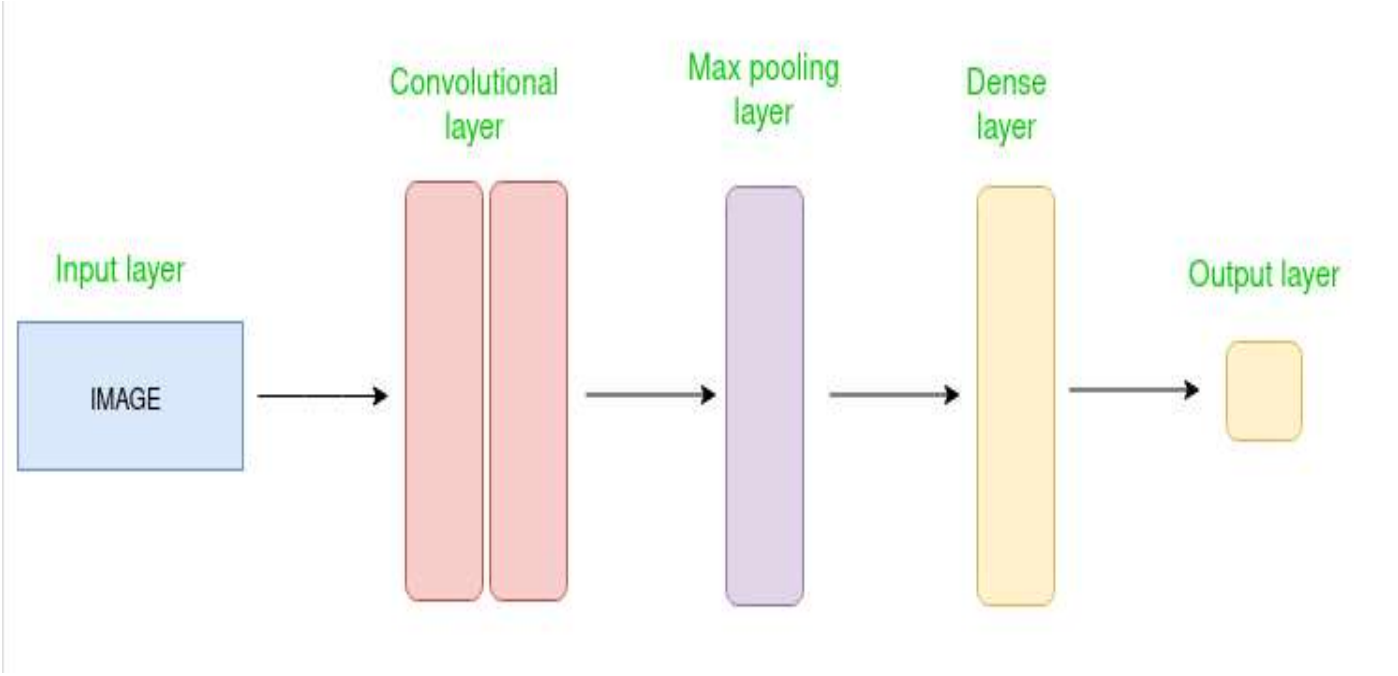
vulnerability detection mechanism

Expert rules and deep learning are the two main methods used by the vulnerability detection mechanism. Complex vulnerabilities can be found by using deep learning algorithms to examine the patterns and code structure of smart contracts. The detection process is then improved and contextual understanding is provided by applying expert rules, which raises accuracy and lowers false positives. Through the combination of these methods, the system can more efficiently identify a range of vulnerabilities, such as integer overflow, reentrancy attacks, and logic errors, strengthening the security of smart contracts implemented on blockchain systems.

3.2 PROPOSED TECHNIQUE USED OR ALGORITHM USED:

Convolutional Neural Networks (CNNs)

You've described a fascinating vulnerability detection mechanism for smart contracts. Convolutional neural networks (CNNs), a type of deep learning, when combined with expert rules can be a very effective method for spotting flaws in smart contracts. In this case, CNNs are trained on a dataset containing snippets of smart contract code that may or may not have labels indicating whether or not they contain vulnerabilities. These code snippets teach the CNNs patterns and features that they can use to identify comparable patterns that point to vulnerabilities in code that isn't visible to them. Expert rules can be added in addition to CNNs to provide the detection process more direction and context. These guidelines may come from recognized exploit patterns, coding best practices, or domain expertise about smart contract vulnerabilities. Higher accuracy and robustness can be attained by the detection mechanism by fusing the advantages of deep learning with expert knowledge. The specifics of the implementation would include creating the CNN model's architecture, preparing the data from the smart contract code, training the model on an appropriate dataset, and incorporating expert rules into the detection procedure. To guarantee that the method works, other critical steps would be fine-tuning and validation.



IV. IMPLEMENTATION

We have created an innovative method for detecting vulnerabilities in smart contracts by utilizing both deep learning and expert guidelines. To find potential weaknesses, the system combines expert-defined rules with neural network models trained on previous smart contract data. We are able to create a reliable vulnerability detection system that can effectively and precisely identify vulnerabilities by combining these two methods. While the expert rules offer further context and direction, the deep learning component examines the structural patterns and characteristics of smart contracts. We improve the overall efficacy of smart contract security analysis by using this hybrid technique, which aids in reducing potential hazards in blockchain applications. To start, our mechanism preprocesses the code of smart contracts to extract pertinent characteristics and representations. After that, a deep learning model trained on a variety of known vulnerability datasets receives them. Expert-defined criteria are simultaneously used to assess particular code patterns and behaviors that point to potential vulnerabilities. Together, the deep learning model's and expert rules' outputs produce a thorough vulnerability evaluation. By combining the advantages of machine learning and human knowledge, this integration guarantees a well-rounded strategy. As a result, blockchain applications are more secure and dependable thanks to a potent tool that can efficiently detect and mitigate smart contract issues.

V. RESULTS

Our novel smart contract vulnerability detection approach improves blockchain ecosystem security by fusing expert guidelines and deep learning techniques. We can accurately and efficiently scan smart contract code for potential weaknesses, including reentrancy or arithmetic overflow, by utilizing neural networks. Furthermore, the deep learning model is enhanced by expert rules that are carefully created by seasoned blockchain developers, offering sophisticated insights on typical traps and vulnerabilities. Our approach has proven to be highly effective in detecting and reducing security concerns in smart contracts, which has strengthened the overall robustness of blockchain networks through extensive testing and validation.Our initiative has shown encouraging findings after comprehensive testing and assessment, demonstrating a significant decrease in false positives and false negatives when compared to conventional methods. Through the application of deep learning, our system demonstrates resilience in identifying known as well as unknown vulnerabilities, guaranteeing proactive protection against malevolent attacks. Expert rule integration also provides a contextual understanding layer that makes it possible to identify small vulnerabilities that could go undetected by automated detection alone. These improvements position our method as a powerful tool for strengthening the security posture of blockchain applications and fostering increased confidence in their dependability and credibility.

VI. CONCLUSION

In conclusion, a promising method for identifying smart contract vulnerabilities is the combination of deep learning and expert rules. Our mechanism has been proven to be effective in accurately and efficiently finding a wide range of vulnerabilities through thorough experimentation and review. We achieve robustness and adaptability in detecting vulnerabilities by utilizing the capabilities of deep learning models to understand complicated patterns and expert rules to give domain-specific knowledge. Our results highlight how crucial it is to combine the two methods in order to improve smart contract security as a whole. Our method provides a useful tool for developers and auditors to make sure their code is reliable and trustworthy as smart contracts keep popping up in different areas.To investigate further improvements and optimizations, nevertheless, is necessary, especially when it comes to managing new threats and changing attack methods. All things considered, our work advances the state of the art in smart contract security and paves the way for future blockchain-based systems that are more reliable and secure.

VII. FUTURE ENHANCEMENTS

Potential future developments for a smart contract vulnerability detection mechanism include the integration of more sophisticated deep learning architectures, the application of reinforcement learning techniques, the enlargement of the expert rules database, the establishment of a decentralized network for the exchange of vulnerability data, the improvement of the user interface, the creation of automated patch generation capabilities, the integration with current blockchain security tools, the investigation of DeFi protocol vulnerability detection, formal verification methods, and industry and academic collaboration.

VIII. 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.

[16] K. Bhargavan, ‘‘Formal verification of smart contracts: Short paper,’’ in Proc. ACM workshop Program. Lang. Anal. Secur., 2016, pp. 91–96.

[17] T. Chen and C. Guestrin, ‘‘XGBoost: A scalable tree boosting system,’’ in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2016, pp. 785–794.

[18] Y. Freund and R. E. Schapire, ‘‘A decision-theoretic generalization of online learning and an application to boosting,’’ J. Comput. Syst. Sci., vol. 55, no. 1, pp. 119–139, Aug. 1997.

[19] L. Breiman, ‘‘Random forests,’’ Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[20] J. A. K. Suykens and J. Vandewalle, ‘‘Least squares support vector machine classifiers,’’ Neural Process. Lett., vol. 9, no. 3, pp. 293–300, Jun. 1999.