AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
Peer Reviewed, Rferred & Indexed Journal
E-ISSN:3069-0102
VOL.6, NO. 2(2026)

Automated Vulnerability Detection in Smart Contracts Using Deep Learning

J Chanbasha[1], R R Shantha Spandana[2]

[1]P.G Scholar, Department of MCA, Sri Venkatesa Perumal College of Engineering & Technology, Puttur,
E-mail: cc5214685@gmail.com, ORC-ID: https://orcid.org/0009-0009-0361-8520

[2]Assistant Professor, Department of MCA, Sri Venkatesa Perumal College of Engineering & Technology, Puttur,
E-mail: shanthaspandana@gmail.com, ORC-ID: https://orcid.org/0009-0003-4236-1250

Abstract: Smart contracts, essential to Blockchain functionality, can be compromised by vulnerabilities like re-entrancy attacks, allowing unscrupulous entities to misappropriate funds. A universal and efficient multi-modal vulnerability detection framework is created to tackle detection issues that exceed the capability of standard methods such as fuzzy testing and symbolic execution. The methodology incorporates BiLSTM, EfficientNet, and Transformer architectures, augmented by CNN2D and BiGRU for better feature extraction and sequence modeling. The SMARTBUG dataset is employed in two formats: compiled OPCODES and features extracted via Word2Vec from smart contract source code. Preprocessing entails utilizing Word2Vec to produce N-gram numerical representations, succeeded by an 80-20 division for training and testing. The system analyzes multi-modal inputs, such as grayscale image attributes, opcode frequency statistics, and source code sequences, facilitating comprehensive vulnerability characterisation. The experimental assessment assesses the proposed model in comparison to existing algorithms, including MLP, GRU, and BiLSTM, utilizing criteria such as accuracy, precision, recall, and F-score. The CNN2D + BiGRU + EfficientNet + Transformer setup attains the greatest detection accuracy of 91.9%, surpassing all benchmarks. The system reduces dependence on domain knowledge by automating feature extraction, enabling adaptation across diverse smart contract forms and improving security in blockchain contexts.

"Index Terms: *Smart contracts, Blockchains, Big Data, Feature extraction, Security, Testing, Source coding".*

1. INTRODUCTION

Blockchain technology and Big Data are two revolutionary inventions that have garnered significant attention in recent years due to their extensive potential and applicability across several fields. Big Data, defined by its volume, velocity, variety, validity, and value, is widely utilized in areas such as healthcare, banking, transportation, and e-commerce for informed decision-making and predictive analytics. The intricate and extensive characteristics of Big Data present numerous significant issues, such as safeguarding data security, preserving integrity, thwarting fraud, overseeing quality, and facilitating effective analysis and mining [2]. If unaddressed, these vulnerabilities can severely compromise the reliability and value of Big Data-driven systems.

Blockchain, a decentralized, immutable, transparent, and secure ledger system, presents intriguing solutions to numerous difficulties. The intrinsic characteristics—decentralization, traceability, tamper resistance, and automated execution—facilitate effective data management, verifiable transactions, and reliable audit trails [4]. Integrating blockchain into Big Data infrastructures enhances security, fosters trust, and reduces risks associated with manipulation, unauthorized access, and fraudulent operations [5]. This integration has become a fast expanding field, showcasing the ability to surpass the constraints of conventional centralized data administration.

A fundamental element of blockchain systems is the smart contract, a self-executing program that facilitates, validates, and enforces agreements autonomously, eliminating the necessity for intermediaries. Smart contracts are progressively utilized across several sectors, including logistics monitoring, decentralized finance (DeFi), asset management, and securities trading. Their automated characteristics facilitate expedited execution, diminished operational expenses, and improved transparency. Nevertheless, the swift increase in their usage has heightened the risk of exploitation, particularly in systems with substantial financial interests [7]. Exploitable vulnerabilities in smart contracts may be used by unscrupulous entities to execute unauthorized transactions, deplete funds, or distort company logic, resulting in significant economic and reputational repercussions.

The 2016 Distributed Autonomous Organization (DAO) attack exemplifies these concerns, as hackers exploited a reentrancy vulnerability in a smart contract

to extract almost $60 million in cryptocurrency. This event underscored the technical flaws in smart contract development and highlighted the pressing necessity for sophisticated vulnerability detection and prevention techniques. As blockchain technology advances, it is imperative to tackle these security concerns to guarantee the secure, dependable, and sustainable implementation of blockchain-based applications.

2. LITERATURE REVIEW

Idelberger et al. (2016) [9] conducted an assessment of logic-based methodologies for the implementation of smart contracts in blockchain contexts. They investigated the viability of encoding contractual provisions in formal logic to facilitate automated reasoning, verification, and enforcement. The authors examined how this methodology could enhance clarity and accuracy in the execution of agreements, highlighting its potential to diminish ambiguities and errors in contract interpretation. Their efforts facilitated the convergence of legal contract representation and blockchain automation by emphasizing benefits such as transparency, auditability, and deterministic execution. They acknowledged constraints in scalability and the intricacies of converting legal language into formal logic, so prompting future study on the integration of legal semantics with efficient computational models.

Tsankov et al. (2018) [10] presented Securify, a pragmatic security analysis instrument tailored for Ethereum smart contracts. Their approach integrated compliance and violation patterns, systematically evaluating contracts against security regulations based on established vulnerabilities and best practices. Securify operated directly on the Ethereum bytecode, allowing developers to scrutinize deployed contracts in the absence of source code. The research emphasized the significance of scalable and automated vulnerability identification, since the authors evaluated their program on thousands of contracts and identified multiple security vulnerabilities. This work represented a notable enhancement in static analysis capabilities, providing developers with actionable input to bolster contract robustness and underscoring the necessity for continuous updates as new attack patterns arise.

Feist et al. (2019) [11] created Slither, a static analysis system tailored for Solidity smart contracts. Constructed using a modular architecture, Slither offered an intermediate representation for identifying a wide range of vulnerability categories and code quality concerns. The framework seamlessly integrated into development workflows, providing rapid analysis and detailed reporting to facilitate security assessments. Their research highlighted performance and extensibility, enabling customers to create bespoke detectors and adjust to changing security requirements. By facilitating continuous integration with security analysis, Slither responded to the increasing need for early vulnerability identification during the development stage, establishing a basis for the incorporation of security-centric practices within the Ethereum ecosystem.

Tikhomirov et al. (2018) [12] introduced SmartCheck, a static analysis tool designed for Solidity code, emphasizing pattern-based vulnerability identification. Their methodology converted Solidity code into an XML-based intermediate representation, facilitating the use of XPath queries to detect potential vulnerabilities such as reentrancy, integer overflows, and unchecked external calls. The authors assessed SmartCheck on a collection of real-world smart contracts, illustrating its efficacy in identifying both straightforward and intricate security vulnerabilities. This work enhanced the resources available to Ethereum developers by providing a lightweight and efficient detection mechanism. It highlighted the difficulty of balancing precision and recall in automated security solutions to reduce false positives while ensuring thorough coverage.

Beksultanova and Tkachenko (2023) [13] performed a comparative analysis of diverse smart contract security methods, emphasizing their applicability, advantages, and drawbacks. The authors examined both static and dynamic analytic techniques, addressing the compromises among detection precision, computing expense, and integration simplicity within development workflows. The authors emphasized deficiencies in current technologies, especially in identifying newly developing classes of vulnerabilities and assuring compatibility with advancing blockchain platforms. Their study offered essential guidance for developers and researchers by delineating the criteria for picking suitable security tools based on contract complexity, performance requirements, and desired coverage.

Wang et al. (2021) [14] introduced ContractWard, an automated system for vulnerability identification that

utilized machine learning to improve detection precision for Ethereum smart contracts. ContractWard employed opcode sequences derived from smart contract bytecode, and feature engineering methods to identify pertinent patterns suggestive of vulnerabilities. The authors proved that their methodology surpassed conventional static analysis regarding both precision and recall, especially for recognized vulnerability categories. Their research connected classic rule-based systems with contemporary data-driven methodologies, highlighting the capacity of machine learning to adapt to changing attack vectors and enhance detection over time.

Yu et al. (2021) [15] presented DeeSCVHunter, a deep learning framework designed to find vulnerabilities in smart contracts more efficiently than conventional approaches. Utilizing neural network topologies, their system acquired the capacity to recognize vulnerability patterns from labeled datasets, eliminating the necessity for manually constructed detection algorithms. The authors demonstrated that their methodology attained competitive accuracy and generalization ability, underscoring deep learning's capacity to identify intricate, non-linear relationships inside contract code. Their research emphasized the significance of curated datasets and the difficulty of acquiring high-quality labeled instances for training dependable models. DeeSCVHunter's flexibility to novel vulnerability types further showed the efficacy of AI-driven methodologies in blockchain security.

Cai et al. (2023) [16] introduced an innovative approach that integrates sliced joint graph representation with graph neural networks (GNNs) for the discovery of vulnerabilities in smart contracts. By integrating control flow with data flow modeling, their approach encapsulated both structural and semantic links within the code. This integration facilitated the identification of vulnerabilities that may not be evident through solely sequential or pattern-based analysis. The authors indicated substantial enhancements in detection efficacy, especially in scenarios with intricate logical linkages and multi-functional vulnerabilities. Their methodology demonstrated the benefits of GNNs in comprehending program structure and semantics more profoundly, paving the way for more sophisticated and contextually aware security measures for smart contracts.

3. MATERIALS AND METHODS

The suggested approach presents a comprehensive and effective multi-modal framework for identifying vulnerabilities in smart contracts, seeking to overcome the shortcomings of traditional methods such as fuzzy testing and symbolic execution. The system processes the SMARTBUG dataset in two formats—compiled OPCODES and Word2Vec-generated N-gram numeric features derived from smart contract source code—using advanced deep learning architectures. Multi-modal feature extraction integrates grayscale image characteristics, opcode bag frequency statistics, and source code sequences, employing EfficientNet for image pattern analysis, BiLSTM or BiGRU for modeling sequential dependencies, Transformer networks for contextual representation learning, and CNN2D for automated spatial feature extraction. The preparation workflow utilizes Word2Vec to transform code into high-dimensional embeddings, subsequently dividing the dataset into training and testing subsets. The system facilitates modular, pluggable components for parallel processing, allowing flexibility to various smart contract formats while minimizing reliance on domain-specific expertise through automated, data-driven feature learning.

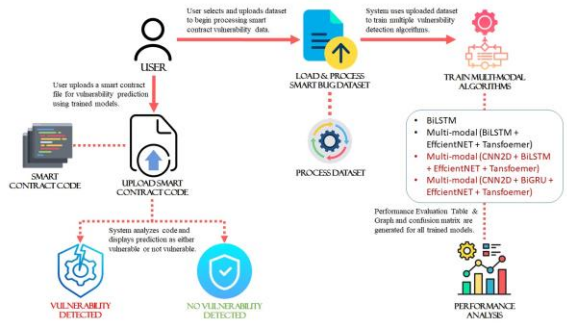


Fig.1 Proposed Architecture

The proposed system architecture facilitates the automated identification of vulnerabilities in smart contracts through the utilization of multimodal deep learning models. The procedure commences with the user providing a vulnerability dataset for training or a smart contract code for prediction. Uploaded datasets are utilized to train several algorithms, including BiLSTM, CNN2D, EfficientNet, BiGRU, and Transformers in diverse combinations. The technology evaluates uploaded smart contracts and forecasts the presence of vulnerabilities in the code. Performance evaluation is executed using tables, graphs, and

confusion matrices to assess the accuracy and efficiency of the trained models.

a) Dataset Collection:

The SMARTBUG dataset, intended for the detection of vulnerabilities in smart contracts, consists of two main formats: compiled OPCODES and features extracted via Word2Vec from Solidity source code. It comprises a variety of contract samples exhibiting multiple vulnerabilities, including re-entrancy and logical errors, facilitating thorough model training. Word2Vec produces N-gram numerical representations for sequential analysis, whereas OPCODE data facilitates instruction-level scrutiny. This dual-format structure enables multi-modal learning, allowing deep learning models to capture both semantic and operational attributes of contracts, thus improving detection accuracy and adaptability across Blockchain contexts.

b) Pre-Processing:

The preprocessing phase commences with the implementation of the Word2Vec approach on the Solidity source code to transform textual contract components into dense vector embeddings, thereby maintaining the semantic links among instructions. N-gram modeling is employed to capture contextual patterns and sequential dependencies, hence improving vulnerability representation. The processed data is ultimately partitioned into an 80-20 ratio for training and testing, facilitating balanced assessment while allowing deep learning models to effectively learn from both semantic and structural attributes.

The preprocessing pipeline transforms smart contract source code and opcode data into numerical and visual formats, allowing deep learning models to effectively identify semantic, syntactic, and structural patterns for precise vulnerability detection.

c) Training and Testing:

The training and testing procedure employs the preprocessed SMARTBUG dataset, with 80% designated for training and 20% for testing. During training, the deep learning framework—comprising CNN2D, BiGRU, EfficientNet, and Transformer architectures—discerns vulnerability patterns from multi-modal inputs, including opcode frequency data, grayscale image characteristics, and source code sequences. Testing assesses the model's efficacy on unobserved data, guaranteeing its precision, generality, and resilience in identifying various smart contract vulnerabilities across different blockchain contexts.

d) Algorithms:

Existing BiLSTM: BiLSTM analyzes smart contract source code sequences in both directions, identifying past and future dependencies for efficient vulnerability pattern detection. It improves contextual comprehension and sequence modeling, facilitating precise identification of security vulnerabilities in contract execution logic.

Proposed Multi-modal (BiLSTM + EfficientNet + Transformer): This hybrid model incorporates BiLSTM for sequence analysis, EfficientNet for extracting grayscale picture features from opcode representations, and Transformer for capturing long-range relationships. The integration guarantees thorough risk assessment at structural, contextual, and semantic dimensions.

Multi-modal (CNN2D + BiLSTM + EfficientNet + Transformer): CNN2D collects spatial information from visual opcode patterns, BiLSTM models sequential dependencies, EfficientNet analyzes high-level picture properties, and Transformer identifies intricate relationships. They enhance detection accuracy by integrating spatial, sequential, and contextual information.

Multi-modal (CNN2D + BiGRU + EfficientNet + Transformer): CNN2D delineates intricate spatial opcode patterns, BiGRU adeptly models sequential dependencies with diminished complexity, EfficientNet analyzes comprehensive image features, and Transformer elucidates contextual relationships, augmenting adaptability and efficacy in detecting various smart contract vulnerabilities.

4. EXPERIMENTAL RESULTS

Accuracy: The accuracy of a test refers to its capacity to correctly distinguish between patient and healthy cases. To assess the accuracy of a test, one must compute the ratio of true positives and true negatives across all assessed cases. This can be expressed mathematically as:

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} (1)$$

Precision: Precision assesses the proportion of accurately classified cases among those identified as positive. Consequently, the formula for calculating precision is expressed as:

$$Precision = \frac{\text{True Positive}}{\text{True Positive } + \text{False Positive}} (2)$$

Recall: Recall is a metric in machine learning that assesses a model's capability to identify all pertinent instances of a specific class. The ratio of accurately predicted positive observations to the total actual positives, offering insights into a model's efficacy in identifying occurrences of a specific class.

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} (3)$$

F1-Score: The F1 score is a metric for evaluating the accuracy of a machine learning model. It combines the precision and recall scores of a model. The accuracy metric quantifies the frequency of true predictions generated by a model throughout the entire dataset.

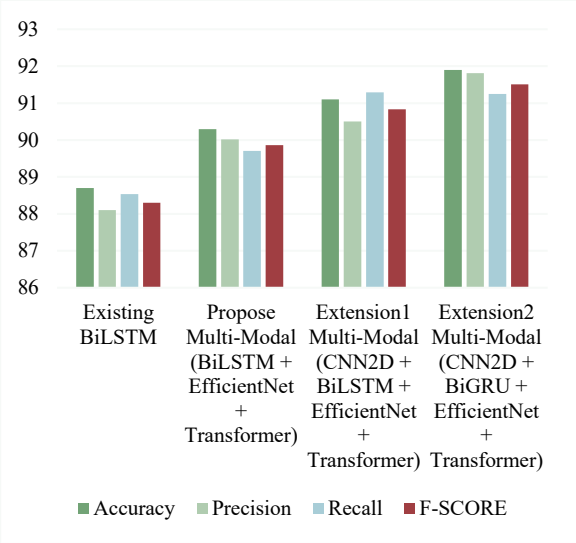
$$F1\ Score = 2 * \frac{Recall\ \text{X}\ Precision}{Recall + Precision} * 100 (1)$$

Performance evaluation was undertaken in Table 1 utilizing accuracy, precision, recall, and F-score. The Extension2 Multi-Modal technique earned the greatest overall metrics among all models, exhibiting superior predictive capability relative to other assessed methods.

Table.1 Performance Evaluation Table

Algorithm Name	Accuracy	Precision	Recall	F-Score
Existing BiLSTM	88.7	88.1	88.54	88.3
Propose Multi-Modal (BiLSTM + EfficientNet + Transformer)	90.3	90.02	89.71	89.86
Extension1 Multi-Modal (CNN2D + BiLSTM + EfficientNet + Transformer)	91.1	90.5	91.29	90.83
Extension2 Multi-Modal (CNN2D + BiGRU + EfficientNet + Transformer)	**91.9**	**91.81**	**91.25**	**91.51**

Graph.1 Comparison Graph



The Comparison Graph (1) depicts model performance in terms of accuracy (green), precision (light green), recall (blue), and F-score (red). Extension2 Multi-Modal attains the highest scores across all parameters, demonstrating better efficacy compared to other assessed methods.

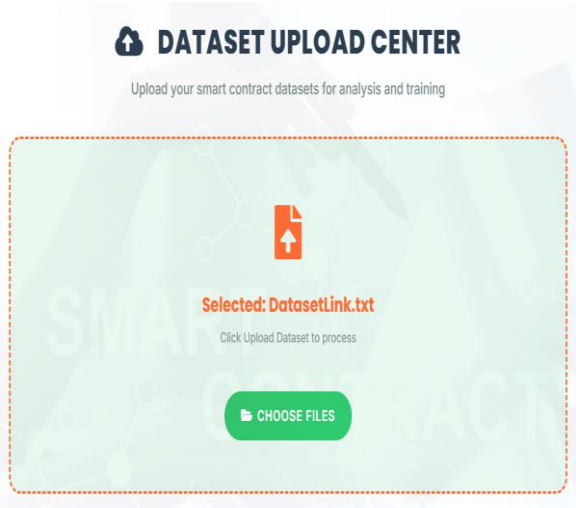


Fig.2 Dataset Upload Center

AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
Peer Reviewed, Rferred & Indexed Journal
E-ISSN:3069-0102
VOL.6, NO. 2(2026)

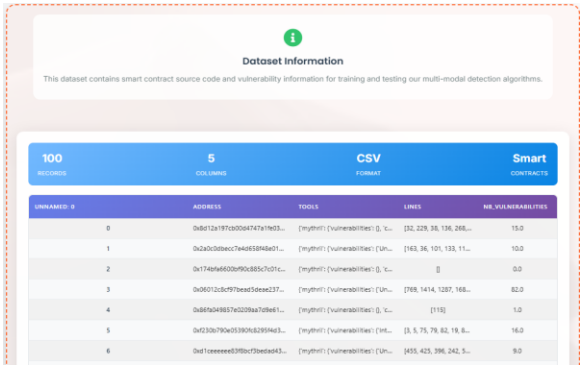


Fig.3 Predict Result

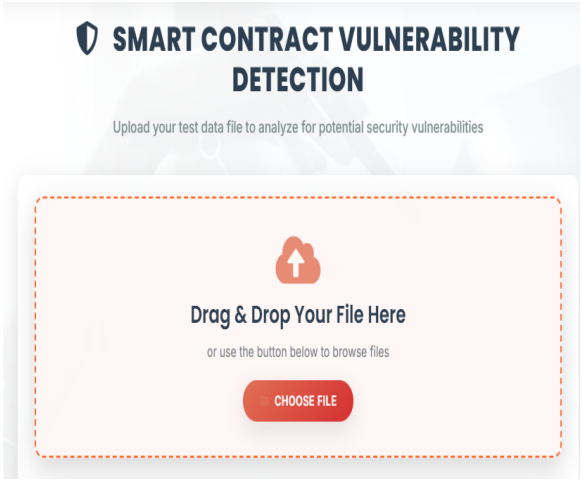


Fig.4 Drag & Drop Your File

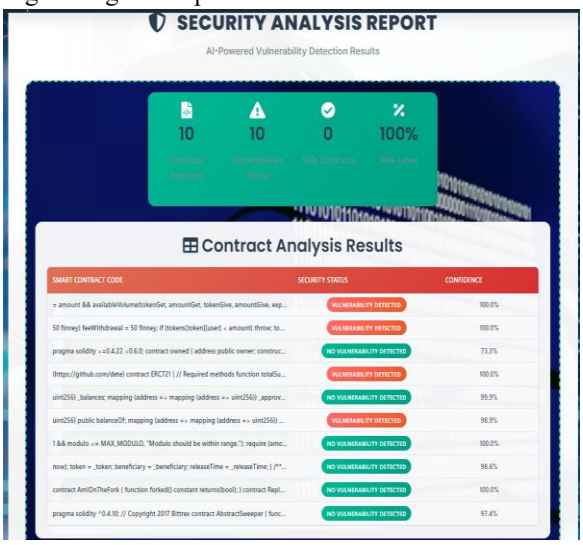


Fig.5 Predict Result

5. CONCLUSION

The proposed multi-modal smart contract vulnerability detection system represents a notable enhancement in Blockchain security with the integration of CNN2D, BiGRU, EfficientNet, and Transformer architectures for thorough feature extraction and sequence modeling. The system employs the SMARTBUG dataset in two formats—compiled OPCODES and Word2Vec-derived features from source code—utilizing Word2Vec-based N-gram numeric representations and an 80-20 train-test split for optimal learning. Multi-modal inputs, comprising grayscale picture attributes, opcode bag frequency data, and source code sequences, provide comprehensive vulnerability characterisation across various contract forms. The experimental findings indicate that the CNN2D+BiGRU+EfficientNet+Transformer configuration attains the greatest detection accuracy of 91.9%, surpassing benchmarks including MLP, GRU, and BiLSTM in precision, recall, and F-score. The system reduces reliance on domain-specific knowledge by automating feature learning, hence improving flexibility to diverse coding styles and formats. The incorporation of advanced deep learning models enhances detection accuracy and fortifies resilience against complex attacks such as re-entrancy and logic manipulation, thereby facilitating the secure implementation of Blockchain-based smart contracts in practical applications. This method creates a scalable and effective strategy for tackling emerging dangers in decentralized systems.

The suggested system can be enhanced by integrating larger and more varied smart contract datasets outside SMARTBUG to augment generality across Blockchain platforms. Future endeavors may incorporate sophisticated embeddings such as CodeBERT or Graph Neural Networks to enhance semantic comprehension. Real-time deployment in blockchain nodes may facilitate proactive vulnerability discovery during contract execution. Furthermore, adaptive learning processes can be employed to revise models in response to emerging attack patterns, hence maintaining accuracy and resilience in swiftly growing decentralized application ecosystems.

REFERENCES

[1] Bresil, M., Prasad, P., Sayeed, M. S., & Bukar, U. A. (2025). Deep Learning-based Vulnerability Detection Solutions in Smart Contracts: A Comparative and Meta-Analysis of Existing Approaches. IEEE Access.

[2] J. A. Zhang, “An overview of signal processing techniques for joint communication and radar

sensing," IEEE J. Sel. Topics Signal Process., vol. 15, no. 6, pp. 1295–1315, Nov. 2021.

[3] Gudditti, V., & Krishna, P. V. (2021). Adaptive Light Weight Encryption Algorithm for Securing Multi-Cloud Storage. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(9), 545–554.

[4] JJ, L., Singh, K., & Chakravarthi, B. (2024). Digital forensic framework for smart contract vulnerabilities using ensemble models. Multimedia Tools and Applications, 83(17), 51469-51512.

[5] Ding, H., Liu, Y., Piao, X., Song, H., & Ji, Z. (2025). SmartGuard: An LLM-enhanced framework for smart contract vulnerability detection. Expert Systems with Applications, 269, 126479.

[6] Zheng, P., Zheng, Z., & Luo, X. (2022). Park: Accelerating smart contract vulnerability detection via parallel-fork symbolic execution. *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, 740–751. https://doi.org/10.1145/3533767.3534406

[7] Jiang, B., Liu, Y., & Chan, W. K. (2018). ContractFuzzer: Fuzzing smart contracts for vulnerability detection. *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 259–269. https://doi.org/10.1145/3238147.3238177

[8] Kalae, U. K. (2025). Optimizing cost-effective cloud data pipeline orchestration across multiple cloud providers. Journal of Information Systems Engineering and Management, 10(63s), e726–e741.

[9] Idelberger, F., Governatori, G., Riveret, R., & Sartor, G. (2016). Evaluation of logic-based smart contracts for blockchain systems. In *Proceedings of the 10th International Symposium on Rule Technology: Research, Tools, and Applications* (pp. 167–183). Springer. https://doi.org/10.1007/978-3-319-42019-6_11

[10] Patyrykin, K., & Vasyukova, L. (2025). Environmental Accountability or Symbolic Compliance? A Critical Review of ESG Ratings, Greenwashing, and Indirect Emissions in the Global Insurance Sector. International Journal of Energy Economics and Policy, 15(6), 917–925. https://doi.org/10.32479/ijeep.22770

[11] Viswanath G., Krishna Prasad K ., Dr. J Maha Lakshmi., Dr.G.Swapna (2024). Health Prediction Using Machine Learning with Drive HQ Cloud Security. Frontiers in HealthInformatics, 13(8), 2755-2761, https://doi.org/10.5281/zenodo.19128870

[12] Tikhomirov, S., Voskresenskaya, E., Ivanitskiy, I., Takhaviev, R., Marchenko, E., & Alexandrov, Y. (2018). SmartCheck: Static analysis of ethereum smart contracts. *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*, 9–16. https://doi.org/10.1145/3194113.3194115

[13] Beksultanova, A., & Tkachenko, A. (2023). Analysis tools for smart contract security. *Proceedings of the 2nd International Conference on Computer Applications and Management Sustainable Development Production and Industry*, 221–228. https://doi.org/10.1007/978-3-031-35689-0_23

[14] Wang, W., Song, J., Xu, G., Li, Y., Wang, H., & Su, C. (2021). ContractWard: Automated vulnerability detection models for Ethereum smart contracts. *IEEE Transactions on Network Science and Engineering, 8*(2), 1133–1144. https://doi.org/10.1109/TNSE.2020.3013689

[15] Yu, X., Zhao, H., Hou, B., Ying, Z., & Wu, B. (2021). DeeSCVHunter: A deep learning-based framework for smart contract vulnerability detection. *Proceedings of the International Joint Conference on Neural Networks*, 1–8. https://doi.org/10.1109/IJCNN52387.2021.9534336

[16] Bajarang Bhagwat, V. (2023). Optimizing Payroll to General Ledger Reconciliation: Identifying Discrepancies and Enhancing Financial Accuracy. JOURNAL OF ADVANCE AND FUTURE RESEARCH, 1(4). https://doi.org/10.56975/jaafr.v1i4.501636

[17] Poojari, R. Frameworks for Data Management and Lineage in Large-Scale Healthcare Data Systems..

[18] Zhang, L. (2022). A novel smart contract vulnerability detection method based on information graph and ensemble learning. *Sensors, 22*(9), 3581. https://doi.org/10.3390/s22093581

[19] Zhuang, Y., Liu, Z., Qian, P., Liu, Q., Wang, X., & He, Q. (2021). Smart contract vulnerability detection using graph neural networks. *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 3283–3290. https://doi.org/10.24963/ijcai.2021/452

[20] Huang, T. H.-D. (2018). Hunting the Ethereum smart contract: Color-inspired inspection of potential attacks. *arXiv preprint* arXiv:1807.01868. https://arxiv.org/abs/1807.01868