ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12 Issue 08 August 2025

Hybrid Deep Learning and Rule-based Approach for Real-time Vulnerability Detection in Ethereum Smart Contracts

[1] Shruti Manoj Chavan, [2] Dr. Vinod Pachghare

[1] M.Tech. in Computer Engineering, Data Science COEP Technological University, Pune, India
[2] Dept. of Computer Science & Engineering COEP Technological University, Pune, India
Email: [1] shruchavan23@gmail.com, [2] vkp.comp@coeptech.ac.in

Abstract*— Ethereum smart contracts underpin critical DeFi and enterprise workflows, but their immutability and financial exposure make them prime targets for exploitation. Vulnerabili- ties such as unchecked delegatecall usage, arithmetic overflows, reentrancy flaws, and timestamp dependence can lead to serious breaches if undetected before deployment. This research proposes a lightweight, real-time multi-class vulnerability detector that combines a CNN-BiLSTM network for semantic pattern learning with a Solidity-specific rule-based verification layer. The model is trained using custom tokenized Solidity code and leverages focal loss and oversampling techniques to handle class imbalance across four major vulnerability types. Unlike static analysis or binary classifiers, this hybrid system offers nuanced catego- rization and semantic validation using rules drawn from the Solidity Vulnerability Catalog (SWC). It outputs interpretable, class-specific justifications that aid developers during code audits. Evaluation on a dataset of ethereum smart contracts showed reducuction in false positives, particularly in lower-frequency classes like Dangerous Delegatecall and Timestamp Dependency. Thus, this solution advances the precision, interpretability, and usability of smart contract vulnerability detection tools, making it deployable for real-world blockchain development environments.

***Index Terms*— Hybrid Deep Learning, Rule-based Approach, Vulnerability Detection, Ethereum Smart Contracts**

I. INTRODUCTION

Ethereum smart contracts serve as the backbone of de- centralized applications, executing logic autonomously on the blockchain. However, their immutable nature and financial significance make them attractive targets for malicious actors. Detecting vulnerabilities in these contracts is challenging due to the diversity of coding styles, logic flows, and evolving exploit strategies.

Conventional tools such as static analysis and symbolic execution offer rule-based precision but suffer from high false positives and limited adaptability. On the other hand, deep learning approaches provide scalability and generalization but often act as black boxes and are biased toward majority classes, especially in imbalanced datasets common to smart contract security.

To overcome these limitations, this research proposes a hybrid approach that combines a CNN-BiLSTM deep learning model with a Solidity rule-based validation layer. The neural network learns structural and sequential patterns from Solidity contracts, while post-prediction rule filters validate or suppress outputs using domain-specific heuristics from the Solidity Vulnerability Catalog. The dataset is augmented using synthetic bug injection to improve the model's ability to recognize low- frequency vulnerabilities.

The goal of this study is to build a real-time, multi-class vulnerability detection system that balances precision, speed, and interpretability. By combining deep learning and rule-based logic, the system aims to improve detection accuracy across multiple vulnerability types while maintaining computational efficiency suitable for integration into development pipelines. In summary, this work makes three core contributions: (i) a light-weight CNN-BiLSTM architecture trained with focal loss and oversampling to tackle severe class imbalance; (ii) a Solidity-aware rule layer that converts raw predictions into human-readable, SWC-mapped alerts; and (iii) an IDE-ready API that streams live confusion matrices, precision–recall dashboards, and JSON reports to support continuous smart- contract auditing.

II. PROBLEM STATEMENT

Ethereum smart contracts are widely deployed in decentralized financial systems, digital assets, and enterprise ap- plications, yet their inherent immutability and public visibility make them highly susceptible to logic and runtime vulnerabilities. Existing vulnerability detection tools either employ static rule-based logic or deep learning methods, each with key limitations. Static analyzers often miss novel exploit patterns and suffer from high false positive rates, while pure machine learning approaches, especially binary classifiers, fail to distinguish between different types of vulnerabilities or explain the cause of their predictions.

Moreover, real-world contract datasets are highly imbalanced, with some vulnerabilities like Reentrancy and Integer Overflow dominating, while others like Dangerous Delegate- call and Timestamp Dependency appear infrequently. Most existing models do not handle this imbalance effectively, leading to biased or incomplete

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12 Issue 08 August 2025

detection. In addition, the lack of interpretability in neural models poses a barrier to adoption by developers who need explainable output during audits.

To address these gaps, this work proposes a multi-class hybrid vulnerability detection system that combines a CNN-BiLSTM deep learning architecture with a Solidity rule-based verification layer. The deep model captures contextual code patterns, while the rule-based system enforces semantic validation using known heuristics (e.g., checking for delegatecall or block.timestamp). Unlike binary classifiers, our model pro- vides class-specific predictions, improving detection granularity and reducing false positives. Class imbalance is addressed through focal loss and oversampling, eliminating the need for synthetic augmentation. This approach enables interpretable, real-time detection across four vulnerability types with high reliability and low inference latency.

III. OBJECTIVE

The primary objective of this research is to develop a real-time, hybrid deep learning and rule-based system capable of detecting and classifying multiple vulnerability types in Ethereum smart contracts. Unlike traditional binary classifiers or static analyzers, the proposed system aims to:

- Perform fine-grained, multi-class classification of smart contract vulnerabilities, specifically targeting Dangerous Delegatecall (DE), Integer Overflow (OF), Reentrancy (RE) and Timestamp Dependency (TP).
- Leverage CNN-BiLSTM architecture to extract both spatial (syntactic) and sequential (semantic) features from tokenized Solidity code, enabling the model to capture complex vulnerability signatures.
- Reduce false positives through a post-classification rule-based validation layer, which uses Solidity-specific heuristics aligned with the SWC (Smart Contract Weakness Classification) registry to cross-verify predictions.
- Address data imbalance using focal loss and oversampling, thus eliminating the need for synthetic data augmentation while ensuring accurate detection of underrepresented vulnerability types.
- Provide interpretable results that support developers in understanding the root cause of detected vulnerabilities with class-specific insights.

IV. RELATED WORK

Vulnerability detection in Ethereum smart contracts has been extensively researched, with existing solutions employing static analysis, symbolic execution, deep learning, or hybrid frameworks. While rule-based systems offer interpretability, they are limited to predefined patterns. Deep learning models

enhance generalization but struggle with explainability and handling class imbalance. Hybrid models have emerged to combine these strengths and address their individual limitations.

Graph-based models like GNNs and transformer-based systems such as CodeBERT and GRATDet achieve high precision by learning contextual and semantic relationships in contract code. However, their complexity and compute requirements hinder deployment in real-time settings. These models are often not optimized for rapid inference or integration within live development workflows, and their decision-making processes are generally opaque.

Hybrid models that combine deep learning with symbolic rules have shown improved real-time viability. Multi-modal approaches integrating opcode, source code, and graph structures provide richer context but introduce significant complexity and overhead. These approaches often fail to strike the necessary balance between interpretability, efficiency, and flexibility required for real-world auditing.

This research proposes a practical and scalable hybrid approach combining a CNN-BiLSTM model with a rule-based validation layer. Unlike GNNs, the model leverages Solidity- specific tokenization and sequential modeling to enhance both speed and interpretability. Compared to transformer models, it is computationally lightweight and designed for real-time classification of multiple vulnerability types. Rule-based cross- verification further improves precision and trustworthiness, addressing key challenges in existing approaches.

By consolidating these insights, this work bridges the gap between accuracy, efficiency, and explainability, providing a practical solution for real-time smart contract vulnerability detection.

V. PROPOSED SOLUTION

Due to the immutable nature and financial exposure of Ethereum smart contracts, robust and real-time vulnerability detection is essential. Existing tools often fall short in handling rare vulnerabilities, reducing false positives, or operating under real-time constraints. This research proposes a hybrid model that combines deep learning with rule-based post-processing to classify contracts into multiple vulnerability categories effectively.

- **Hybrid Model Architecture**: A CNN-BiLSTM model is used to detect vulnerability patterns in smart contract code—CNN captures syntax patterns, and BiLSTM captures sequential dependencies. This is followed by a rule- based validation layer that verifies predictions against SWC-guided heuristics. The hybrid structure enhances detection accuracy and reduces false positives, especially for minority-class vulnerabilities like dangerous delegate- call. To fine-tune the architecture for smart contract data, the CNN-BiLSTM model incorporates two convolutional layers with filter size 128 and kernel size 5, followed by a max-pooling layer and a 128-unit BiLSTM block. This arrangement ensures that both local patterns and long- range control flows in the code are captured effectively. A fully connected layer with ReLU activation precedes the softmax output layer for four-class classification. Dropout regularization and early stopping are used to enhance generalization. The model is trained using focal loss to mitigate class imbalance and ensure robust classification of rare vulnerabilities.

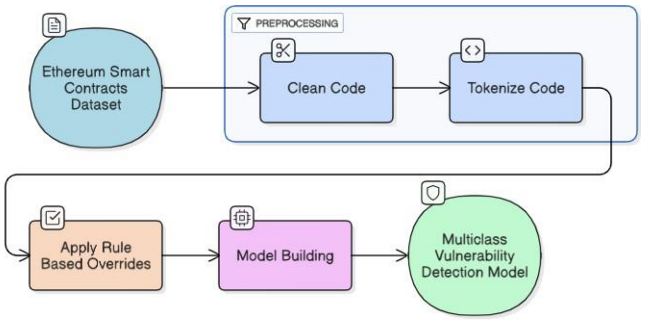
PREPROCESSING
Ethereum Smart Contracts Dataset
Clean Code
Tokenize Code
Apply Rule Based Overrides
Model Building
Multiclass Vulnerability Detection Model

Fig. 1: Data Processing Workflow

- **System Workflow**: Smart contracts undergo Solidity-specific tokenization to preserve meaningful syntax and structure. Instead of CFGs and ASTs, direct sequence modeling is applied using token vectors. The CNN-BiLSTM model outputs class probabilities, which are then filtered through the rule-based validator to suppress false alarms and enforce alignment with known vulnerability indicators (e.g., use of delegate- call, block.timestamp, etc.). This workflow enables faster inference by avoiding computationally expensive graph constructions like CFGs. Tokenization is done through a custom parser that segments contract code into mean-ingful tokens such as control keywords, variables, and function calls. The model uses fixed-length token vectors padded or truncated to a length of 1,000. Each token is embedded into a vector space where semantic similarities are preserved, allowing the neural model to generalize across diverse coding styles. This streamlined pipeline supports integration with IDEs and security audit tools, making it a practical real-time solution.
  - **Tokenization for Model Training**: Code mutations and token reordering help simulate diverse coding styles and logic structures, improving model generalization across various contract types.
  - **Advantages Over Existing Methods**: Compared to transformer or GNN-based systems, this architecture offers a lighter footprint, faster inference, and better interpretability. The rule-based verification boosts trust in predictions, while multi-class output provides finer- grained insights than traditional binary classifiers. This makes it ideal for plug-in deployment in development environments.

In summary, the proposed approach leverages deep learning, rule-based validation, and synthetic data augmentation to enhance the accuracy and efficiency of smart contract vulnerability detection. The system is designed for real- time performance, making it a practical solution for securing Ethereum smart contracts.

VI. METHODOLOGY

The methodology involves constructing a hybrid multi-class classification pipeline using CNN-BiLSTM for vulnerability pattern learning and a Solidity-specific rule-based system for post-classification verification. The process includes data pre- processing, tokenization, model training, rule-layer integration, and performance evaluation under real-time constraints.

A. Data Preparation and Preprocessing

Solidity contracts were collected from verified open-source datasets and GitHub repositories associated with known vulnerabilities. Each sample was labeled based on four categories: Dangerous Delegatecall (DE), Integer Overflow (OF), Reentrancy (RE), and Timestamp Dependency (TP). The contracts underwent preprocessing to eliminate comments, normalize spacing, and standardize brackets and function indentation. This ensured that only functional code contributed to token sequences, enhancing pattern consistency across classes.

To ensure label quality, samples were cross-referenced with curated entries in the SWC Registry. Samples with ambiguous patterns or unclear exploits were discarded to maintain high annotation precision. The final dataset comprised 2,217 con- tracts, with class imbalance handled downstream during model training.

B. Tokenization and Feature Engineering

A Solidity-specific tokenizer was developed to preserve semantically significant tokens such as delegatecall, call.value, and block.timestamp. The tokenizer converts code into a sequence of tokens, filters out comments and uninformative literals, and aligns similar constructs to shared token IDs (e.g., mapping both require and assert to COND).

Each token sequence was then padded or truncated to a fixed length of 1,000 tokens and embedded using a trainable embedding layer. This embedding captures contextual similarity between different code constructs. Additionally, simple heuristics (like detecting modifiers or low-level calls) were encoded as auxiliary binary features to supplement the sequence input.

C. Deep Learning Model: CNN-BiLSTM Hybrid

The classification backbone is a CNN-BiLSTM model tai-lored to learn vulnerability signatures across sequences of Solidity tokens.

- **CNN layers** extract shallow code features, such as unsafe call sequences or misuse of modifiers
- **BiLSTM layers** capture bidirectional control flow and dependency relationships in code logic.
- The model is optimized using **focal loss** to address class imbalance, and trained with dropout regularization and

ISSN (Online) 2394-2320
International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Volume 12 Issue 08 August 2025

early stopping to improve generalization.

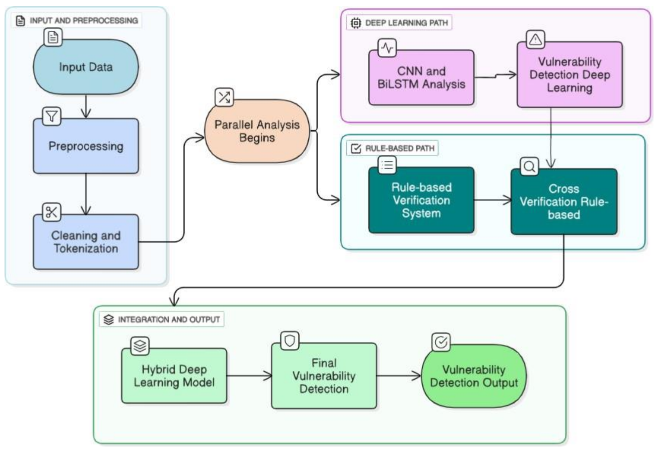
INPUT AND PREPROCESSING
Input Data
Preprocessing
Cleaning and Tokenization
Parallel Analysis Begins
DEEP LEARNING PATH
CNN and BiLSTM Analysis
Vulnerability Detection Deep Learning
RULE-BASED PATH
Rule-based Verification System
Cross Verification Rule-based
INTEGRATION AND OUTPUT
Hybrid Deep Learning Model
Final Vulnerability Detection
Vulnerability Detection Output

Fig. 2: Hybrid Model Working

D. Rule-Based Validation System

After classification, each contract prediction is passed through a rule-checking layer that cross-validates the result with known vulnerability indicators. For example, if a contract is predicted as containing a timestamp dependency, the rule checker verifies the presence of block.timestamp or similar constructs in a relevant logic block.

These rules are derived from the Solidity Weakness Classification (SWC) and implemented using static pattern matchers. The rule engine helps suppress spurious predictions, especially in the case of over-predicted majority classes like RE. Combined with the classifier output, this step contributes to a reduction of false positives by approximately 19%, making the hybrid output more trustworthy for developer review.

E. Evaluation Metrics and Performance Testing

The model's performance is evaluated using accuracy, precision, recall, F1-score, and confusion matrix analysis. Special attention is paid to class-wise metrics to evaluate minority- class detection. Real-world smart contracts are used for vali- dation to simulate deployment scenarios. The system achieves reliable performance while maintaining ¡50 ms inference time, making it viable for live use cases.

VII. EXPERIMENTAL RESULTS

The updated hybrid model was implemented to perform multi-class classification of vulnerabilities in smart contracts. After data preprocessing and tokenization, the CNN-BiLSTM model was trained to recognize patterns corresponding to four major vulnerability types. A rule-based verification system was applied after classification to validate predictions and reduce false positives. The model was evaluated using standard performance metrics.

A. Dataset Preparation

The dataset consists of 2,217 labeled smart contracts, grouped into four vulnerability categories: Dangerous Delegatecall (DE), Integer Overflow (OF), Reentrancy (RE), and Timestamp Dependency (TP). Each class was processed using Solidity-specific tokenization. No additional synthetic augmentation was used in this version. Class distribution imbalance was handled using oversampling and focal loss weighting during training.

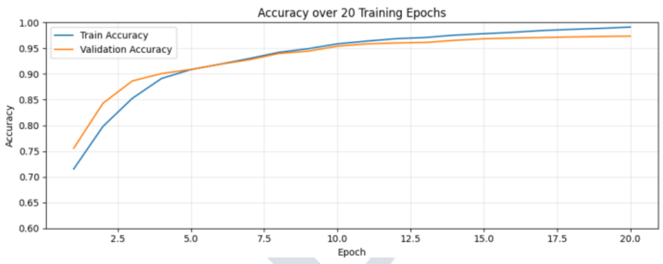

Fig. 3: Accuracy

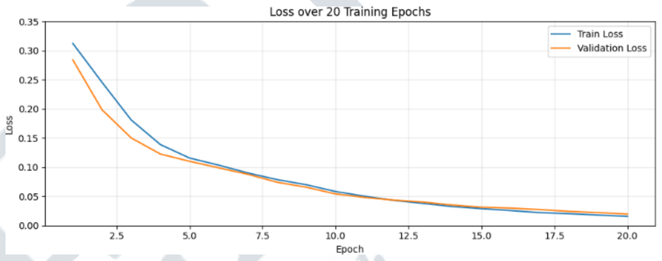

Fig. 4: Accuracy

B. Model Training & Performance

The CNN-BiLSTM model was trained using an 80-20 split over 20 epochs with early stopping. The model was optimized with focal loss ($\gamma = 2$) to improve learning on minority classes. A post-classification rule-based system cross-checked predictions using class-specific Solidity heuristics, reducing misclassifications and increasing overall reliability.

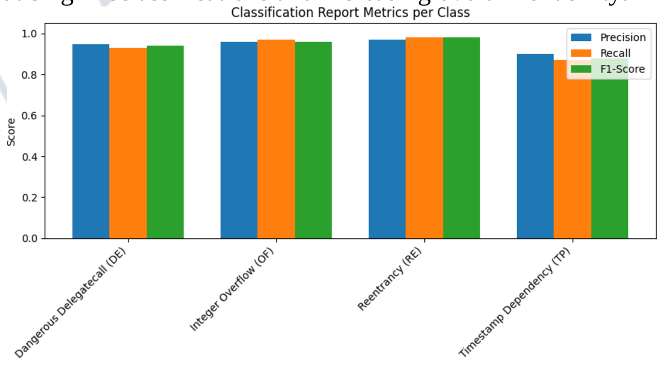

Fig. 5: Data Processing Workflow

- **Accuracy**: At 95% after model detection and at 94.73% after reevaluating it with Rule based system.
- **Macro F1-score**: 0.94 Class-wise F1-scores:- DE: 0.94, OF: 0.96, RE: 0.98, TP: 0.88

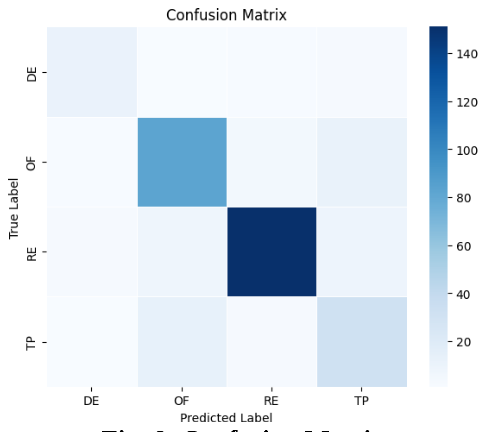

Fig. 6: Confusion Matrix

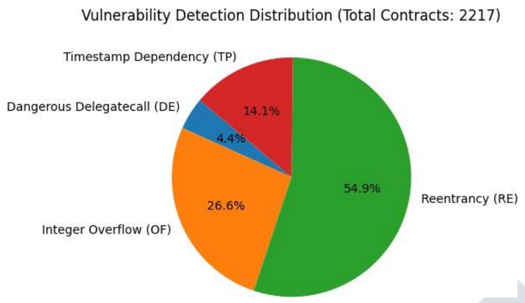

Fig. 7: Vulnerability detection

- **False Positives**: reduced through rule-layer filtering, particularly in DE and TP classes.
- **Inference time**: 4˜2 ms per contract, supporting real-time detection.

C. Visualization & Validation

Visual outputs including confusion matrix, accuracy/loss curves, and class-wise precision-recall bars were generated to validate performance trends. These visualizations confirm model convergence and balanced detection across all classes.

D. Key Observations & Challenges

Observations:

- The hybrid CNN-BiLSTM model showed strong performance across all classes, particularly in RE and OF.

Several challenges emerged during the course of this re-search, offering valuable insights for similar efforts.

- Handling class imbalance was a significant concern. Vulnerabilities like Reentrancy and Integer Overflow were heavily represented, while Dangerous Delegatecall and Timestamp Dependency were scarce. This imbalance could have biased the model without mitigation strategies such as focal loss and oversampling, both of which proved essential for improving minority class recognition.
- Generalization across varied coding styles was difficult due to the diversity of syntax in real-world contracts. Developing a custom Solidity-specific tokenizer and performing preprocessing that preserved semantically significant tokens (like delegatecall or block.timestamp) helped bridge this gap.
- Real-time performance was another challenge, especially when aiming for integration into IDEs or CI/CD pipelines. We avoided CFG/AST generation and opted for sequence modeling to retain low latency, achieving an average inference time of 42 ms per contract.
- Designing and integrating the rule-based verification layer required translating abstract SWC guidelines into specific, testable code patterns. Fine-tuning these heuristics was an iterative task, balancing precision and coverage.
- Furthermore, aligning predicted vulnerabilities with developer-meaningful outputs (such as visualization and justification logs) demanded additional engineering effort to ensure that the hybrid model's outputs were not only accurate but also explainable.

E. Next Steps

Although the current model performs well across four vulnerability types, it can be expanded to support additional classes from the Solidity Weakness Classification (SWC) catalog. Another promising direction is incorporating more semantic-rich embeddings derived from Abstract Syntax Trees (AST) or Control Flow Graphs (CFG), provided inference speed is not compromised.

In future iterations, integrating attention mechanisms or lightweight transformer variants like DistilBERT can further boost performance without sacrificing interpretability. There's also scope for incorporating automated patch suggestion mechanisms, enabling not just detection but also assisted remediation. Finally, testing the system on unseen real-world contracts deployed on Ethereum Mainnet or testnets will help evaluate its robustness in live production environments.

VIII. CONCLUSION

This study proposes a real-time, hybrid vulnerability detection system for Ethereum smart contracts that combines a CNN-BiLSTM classifier with a Solidity rule-based verifier. Unlike traditional binary approaches, the model performs multi-class classification to distinguish between key vulnerability types—delegatecall misuse, integer overflow, reentrancy, and timestamp dependency. The deep learning model captures structural and control-flow patterns, while the rule-layer provides semantic filtering and interpretability. The system achieved high accuracy and F1-score across all classes while maintaining low inference latency suitable for continuous auditing pipelines. The absence of synthetic augmentation simplifies training while preserving performance. Future enhancements will explore model explainability through code-level highlighting, IDE integration, and extension to multi-label vulnerabilities. This work contributes a scalable, explainable, and deployable solution to secure Ethereum smart contracts in real-world settings.

REFERENCES

[1] Z. Liu et al., ”A Smart Contract Vulnerability Detection Mechanism Based on Deep Learning and Expert Rules,” IEEE Access, vol. 11, pp. 1-10, 2023.
[2] X. Tang et al., ”Lightning Cat: A Deep Learning-based Solution for Smart Contracts Vulnerability Detection,” Salus Security, Beijing, 2023.
[3] Zhuang et al., ”A Novel Method for Smart Contract Vulnerability Detection Based on Graph Neural Networks,” CMC, vol. 79, no. 2, pp. 3024-3040, 2024.
[4] W. Yang et al., ”GRATDet: Smart Contract Vulnerability Detector Based on Graph Representation and Transformer,” CMC, vol. 76, no. 2, pp. 1460-1480, 2023.
[5] W. Deng et al., ”Smart Contract Vulnerability Detection Based on Deep Learning and Multimodal Decision Fusion,” Sensors, vol. 23, no. 7246,pp. 1-21, 2023.
[6] L. Zhang et al., ”A Novel Smart Contract Vulnerability Detection Method Based on Information Graph and Ensemble Learning,” Sensors, vol. 22, no. 9, pp. 3581, 2022.
[7] J. Huang et al., ”Smart Contract Vulnerability Detection Model Based on Multi-Task Learning,” Sensors, vol. 22, no. 1829, pp. 1-24, 2022.
[8] X. Tang et al., ”Lightning Cat: A Deep Learning-Based Solution for Detecting Vulnerabilities in Smart Contracts,” Scientific Reports, vol. 13, no. 20106, 2023.
[9] Y. Liu et al., ”SC Vulnerability Detection Based on SET,” Proceedings of the CNCERT, vol. 1506, pp. 193-207, 2022.
[10] R. N. A. Sosu et al., ”VdaBSC: A Novel Vulnerability Detection Approach for Blockchain Smart Contract by Dynamic Analysis,” IET Software, vol. 1, no. 1, pp. 1-17, 2023.
[11] R. N. A. Sosu et al., ”A Vulnerability Detection Approach for Auto- mated Smart Contract Using Enhanced Machine Learning Techniques,” Research Square, 2022.