Enhancing Blockchain Security Through Smart Contract Vulnerability Classification Using BiLSTM and Attention Mechanism

Untung Raharja[1,*], Qurotul Aini[2]

[1,2]Faculty of Science and Technology, University of Raharja, Tangerang 1511, Indonesia

ABSTRACT

The rapid adoption of blockchain technology has intensified the need for robust smart contract security mechanisms. However, traditional rule-based or static analysis tools often fail to detect context-dependent vulnerabilities embedded in complex contract logic. This study proposes a deep learning framework for automated smart contract vulnerability classification using a Bidirectional Long Short-Term Memory (BiLSTM) network integrated with an Attention Mechanism. The model was trained and evaluated on the SC_Vuln_8label.csv dataset, comprising 12,520 labelled Solidity smart contracts categorized into eight distinct vulnerability types, including Re-entrancy, Integer Overflow, and Short Address Attack. Through bidirectional contextual learning and attention-based feature weighting, the proposed model achieved 93.7% test accuracy, 0.93 precision, and a macro F1-score of 0.92, outperforming baseline models such as CNN, GRU, and standard LSTM by up to 5.3 percentage points. Attention heatmap analysis further revealed the model's interpretability by highlighting vulnerability-prone code segments (e.g., call.value, send(), and withdraw() functions) consistent with expert-identified risk indicators. These results demonstrate that the BiLSTM + Attention framework not only enhances vulnerability detection accuracy but also provides transparent and explainable reasoning, offering a reliable foundation for AI-assisted smart contract auditing systems in blockchain security.

Keywords Blockchain Security, Smart Contract Vulnerability Detection, BiLSTM, Attention Mechanism, Deep Learning

Submitted: 2 April 2025
Accepted: 15 May 2025
Published: 7 February 2026

Corresponding author
Untung Raharja,
untung@raharja.info

Additional Information and Declarations can be found on page 43

DOI: 10.47738/jcrb.v3i1.56

INTRODUCTION

Blockchain technology has emerged as a foundational infrastructure for decentralized digital systems by enabling transparent, immutable, and trustless transactions without reliance on centralized authorities [1]. One of the most significant innovations enabled by blockchain is the smart contract, which refers to a self executing program deployed on blockchain platforms such as Ethereum that automatically enforces predefined contractual rules once specified conditions are satisfied [2]. Smart contracts have gained widespread adoption across domains such as decentralized finance, supply chain management, and digital asset governance due to their potential to reduce operational costs and eliminate intermediaries. Despite these advantages, smart contracts introduce critical security risks that primarily stem from their immutable nature [3]. Once deployed, smart contracts cannot be modified or patched [4], and any hidden vulnerability may result in irreversible financial losses, system disruptions, or large scale exploitation [5]. Well known incidents such as the DAO attack have demonstrated the severe consequences of vulnerabilities in smart contract code and have emphasized the importance of effective vulnerability detection mechanisms before deployment [6].

To address these security challenges, various smart contract auditing techniques have been proposed, including manual code inspection, symbolic execution, and static analysis tools such as Oyente, Mythril, and Slither [7]. These approaches constitute the foundation of early smart contract security analysis and remain widely used in practice. However, they exhibit inherent limitations, particularly in terms of high false positive rates, limited scalability, and insufficient capability to capture context dependent vulnerabilities that arise from complex interactions across multiple functions or execution paths [8]. As smart contracts become increasingly complex and modular, these rule based and heuristic driven methods struggle to model deeper semantic and structural dependencies in Solidity programs, which significantly constrains their effectiveness in real world scenarios [9].

Recent advances in deep learning and Natural Language Processing have reshaped the state of the art in automated code analysis by enabling data driven models to learn representations of source code as token sequences analogous to natural language [10]. This paradigm has facilitated the application of neural architectures such as Convolutional Neural Networks, Recurrent Neural Networks, Long Short Term Memory networks, and Transformer based models for smart contract vulnerability detection and program understanding tasks [11]. Although these approaches have demonstrated improved generalization performance compared to traditional static analysis tools, several critical challenges remain unresolved. In particular, many existing models rely on unidirectional sequence modeling, which limits their ability to capture bidirectional contextual dependencies where the semantics of a code statement depend on both preceding and subsequent elements. Furthermore, a significant portion of high performance deep learning based approaches operate as black boxes, providing limited interpretability and offering minimal insight into which specific code components contribute to vulnerability predictions. This lack of transparency poses a substantial barrier to adoption in blockchain security auditing, where explainability and trust are essential for practical deployment by developers and auditors [12].

Despite the progress achieved by recent learning based approaches, a clear research gap remains between detection accuracy and practical usability. Existing state of the art solutions rarely provide interpretable explanations or vulnerability localization capabilities that can support human centered auditing workflows. Moreover, the majority of prior studies do not explicitly address the challenge of modeling bidirectional semantic dependencies inherent in Solidity code, which are crucial for accurately identifying context sensitive vulnerabilities.

Motivated by these limitations, this study proposes an automated and explainable smart contract vulnerability detection framework based on a Bidirectional Long Short Term Memory network integrated with an attention mechanism. The proposed framework jointly models forward and backward contextual dependencies within Solidity code sequences, enabling a more comprehensive representation of semantic relationships across multiple lines and functions. In addition, the attention mechanism enhances interpretability by assigning importance weights to vulnerability relevant code tokens, thereby providing transparent and human understandable explanations for model predictions. By integrating bidirectional sequence modeling with token level attention, this work advances the state of the art in smart contract vulnerability detection by addressing both performance and explainability requirements.

In summary, this research contributes an explainable deep learning based auditing framework that bridges the gap between automated vulnerability detection and practical security analysis. By capturing bidirectional contextual information and highlighting vulnerability indicative code patterns, the proposed approach enhances the reliability, transparency, and trustworthiness of AI assisted smart contract security systems and supports the secure development of blockchain based applications.

Literature Review

The security of blockchain based applications has become an increasingly important research area as smart contracts are widely used to automate financial transactions, decentralized governance, and digital asset management. Despite their potential to remove intermediaries and improve transparency, smart contracts remain highly susceptible to programming errors, design flaws, and logical inconsistencies that can lead to severe economic losses. Numerous large scale security incidents, including the DAO exploit in 2016, the Parity wallet vulnerability in 2017, and multiple decentralized finance attacks reported between 2020 and 2023, have demonstrated the critical need for effective vulnerability detection and prevention mechanisms in smart contracts [13],[14]. As a result, research efforts in this field have evolved from traditional static and symbolic analysis techniques toward learning based and hybrid artificial intelligence driven approaches aimed at improving accuracy, scalability, and interpretability.

Early studies primarily focused on rule based and static analysis methods for smart contract security assessment. Symbolic execution based tools were developed to detect common vulnerability patterns such as reentrancy, timestamp dependency, and transaction ordering dependence by exploring feasible execution paths of smart contract code [15]. Subsequent approaches extended symbolic analysis to identify unsafe contract behaviors related to unauthorized fund transfers, improper self destruction, and unbounded resource consumption [16]. Static analysis frameworks were also introduced to analyze Solidity source code or Ethereum bytecode and identify data flow and control flow vulnerabilities at a higher level of abstraction [17],[18]. Although these tools laid the foundation for automated smart contract auditing and remain widely adopted, they suffer from several inherent limitations. In particular, rule based systems often generate a large number of false positives, struggle to scale to complex contracts, and exhibit limited capability in detecting inter procedural vulnerabilities that span multiple functions or execution contexts. Moreover, their reliance on predefined rules and patterns restricts adaptability to novel or obfuscated attack strategies, motivating the exploration of data driven alternatives.

To overcome the rigidity of rule based analysis, subsequent research introduced machine learning based techniques to automate vulnerability detection. These approaches typically relied on manually engineered features extracted from smart contract source code or opcode sequences, which were then used to train classical classifiers such as Support Vector Machines or ensemble learning models [19],[20]. By learning decision boundaries from labeled data, these methods reduced reliance on expert crafted rules and improved detection automation. However, their dependence on handcrafted features limited their generalization capability, as feature selection often introduced bias and failed to capture the complex semantic relationships inherent in smart contract logic. In addition, many of these approaches struggled to adapt to previously unseen contract structures and evolving vulnerability patterns.

Recent advances in deep learning have significantly influenced the state of the art in smart contract vulnerability detection by enabling models to learn representations directly from raw code without manual feature engineering. Recurrent neural network based architectures were proposed to model sequential dependencies in opcode or tokenized source code, demonstrating improved performance in identifying vulnerabilities such as reentrancy and arithmetic errors [21],[22]. Convolutional neural networks were also explored to capture local syntactic patterns in smart contract code by treating token sequences as spatial features, achieving competitive results in vulnerability classification tasks [23],[24]. Despite these advances, both convolution based and unidirectional recurrent models exhibit limitations in capturing long range dependencies and bidirectional contextual information, where the interpretation of a statement may depend on code that appears later in the sequence.

To address these shortcomings, more recent studies have incorporated Bidirectional Long Short Term Memory networks and attention mechanisms to enhance both contextual understanding and model interpretability. Bidirectional architectures enable the simultaneous modeling of forward and backward dependencies, allowing deeper comprehension of control flow and data flow relationships within smart contracts [25]. Attention mechanisms further improve performance by dynamically assigning higher importance to vulnerability relevant tokens, thereby enabling the model to focus on semantically significant parts of the code [26]. These approaches have demonstrated that combining bidirectional sequence modeling with attention not only improves detection accuracy but also provides interpretable insights into the model decision making process, which is essential for practical adoption in security auditing contexts [27].

Building upon these developments, the present study proposes a BiLSTM with Attention framework specifically designed for multi class smart contract vulnerability classification using the SC_Vuln_8label.csv dataset. Unlike many existing studies that focus on binary classification or a limited subset of vulnerabilities, this work addresses eight major vulnerability categories simultaneously, enabling a more comprehensive evaluation of model generalization across diverse attack types. In addition to improving detection capability, the proposed framework emphasizes explainability by highlighting vulnerability indicative code tokens, thereby narrowing the gap between automated deep learning based detection and human centered security auditing. This research aligns with the growing emphasis on Explainable Artificial Intelligence in blockchain security and contributes toward the development of transparent, interpretable, and trustworthy vulnerability detection systems for secure blockchain ecosystems.

Methods

This study adopts a quantitative deep learning approach using a BiLSTM network integrated with an Attention Mechanism to classify vulnerabilities in Solidity-based smart contracts. The overall analytical workflow is illustrated in figure 1. Research Steps, which outlines the five main stages of the research process: data preprocessing, tokenization and embedding, BiLSTM sequence modelling, attention-based weighting, and performance evaluation. Each stage is designed to ensure that the model can effectively capture both syntactic and semantic dependencies within Solidity code while maintaining interpretability through attention visualization.

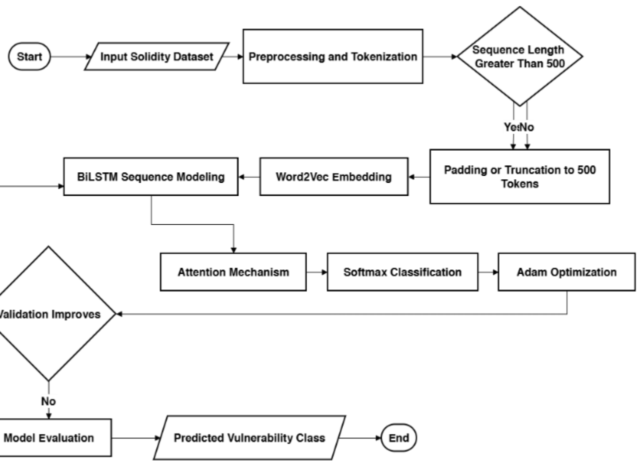


Figure 1 Research Steps

The experiment employed the SC_Vuln_8label.csv dataset, containing 12,520 labelled Solidity smart contracts distributed across eight vulnerability types: Re-entrancy (RE), Timestamp Dependency (TD), Integer Overflow (IO), Unchecked Call Return (UC), Unhandled Exception (UE), Denial of Service (DoS), Short Address Attack (SA), and Other (OT). Each contract was manually verified using a combination of static analysis tools, such as Mythril and Slither, and expert annotations.

Before training, the Solidity source code underwent a multi-stage preprocessing pipeline. Comments, redundant whitespace, and special symbols were removed to standardize syntax structure. The cleaned code was then tokenized into discrete lexical units (identifiers, operators, and keywords) using a Solidity-specific tokenizer. Tokens were converted into integer indices and padded or truncated to a maximum length of 500 tokens to ensure uniform input dimensions. The dataset was partitioned into training (70%), validation (15%), and test (15%) subsets using stratified sampling to preserve class proportions.

To represent the semantic relationships among tokens, Word2Vec embeddings with a dimension size of 128 were trained on the entire dataset. The embedding matrix $E \in \mathbb{R}^{V \times d}$ encodes each token as a dense vector representation, defined as:

$$E = \{e_1, e_2, \dots, e_V\}, \quad e_i \in R^d \tag{1}$$

V is the vocabulary size and $d = 128$ denotes the embedding dimension. This process captures semantic relationships among Solidity tokens, such as msg. sender, require, call, value, and balance.

The Bidirectional LSTM (BiLSTM) extends the traditional LSTM by processing input sequences in both forward and backward directions. This design allows the model to capture bidirectional dependencies crucial for understanding Solidity control flow, where the meaning of a statement may depend on both preceding and succeeding lines of code.

For each time step t, the forward LSTM computes a hidden state $\overrightarrow{h_t}$, while the backward LSTM computes $\overleftarrow{h_t}$. These two vectors are concatenated to form the full hidden representation:

$$h_t = \left[\overrightarrow{h_t}\,\overleftarrow{h_t}\right] \tag{2}$$

The internal computation of the LSTM cell is governed by the following equations:

$$\begin{gathered}
f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \\
i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \\
\widetilde{C_t} = \tanh(W_c[h_{t-1}, x_t] + b_c) \\
C_t = f_t * C_{t-1} + i_t * \widetilde{C_t} \\
o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t = o_t * \tanh(C_t)
\end{gathered} \tag{3}$$

f_t, i_t, and o_t are the forget, input, and output gates, respectively; C_t is the cell memory state; and σ represents the sigmoid activation function. The bidirectional mechanism enhances contextual comprehension by learning dependencies that span multiple functions and logical blocks within Solidity code.

To further improve interpretability and highlight the most relevant parts of the code, an Attention Layer was integrated on top of the BiLSTM outputs. The attention mechanism assigns a relative importance weight α_t to each hidden state h_t, thereby identifying which tokens most strongly influence the classification outcome.

The attention mechanism is defined as follows:

$$\begin{gathered}
u_t = \tanh(W_w h_t + b_w) \\
\alpha_t = \frac{\exp(u_t^\top u_w)}{\sum_{t'} \exp\left(u_{t'}^\top u_w\right)} \\
v = \sum_t \alpha_t h_t
\end{gathered} \tag{4}$$

u_w is the trainable context vector, and v represents the weighted context vector aggregated across all tokens. The attention mechanism not only improves model performance but also facilitates visual interpretability, allowing visualization of which tokens, such as call. value, withdraw, or require carry the strongest contribution to vulnerability classification.

The model was developed using TensorFlow 2.14 with the Keras API. Training was performed using the Adam optimizer with a learning rate of 0.0002, a batch size of 32, and a maximum of 20 epochs. The categorical cross-entropy loss function was used since the task involves multi-class classification across eight categories. Early stopping based on validation accuracy was applied to prevent overfitting and ensure convergence.

The final output layer applies the softmax function to compute the class probabilities for all eight vulnerability types:

$$\hat{y} = \mathrm{softmax}(W_s v + b_s) \tag{5}$$

$\hat{y} \in \mathbb{R}^8$ denotes the predicted probability distribution of the vulnerability classes. The model achieved convergence around the 15th epoch, demonstrating stable learning performance and strong generalization on unseen samples, as later shown in figure 2.

Algorithm 1 Attention-Based BiLSTM Model for Solidity Smart Contract Vulnerability Classification

Dataset and Splitting

$$D = \{(x_i, y_i)\}_{i=1}^{N}, y_i \in \{1, \dots, 8\}$$

The dataset D is divided into training, validation, and testing subsets with a ratio of 70: 15: 15.

Tokenization and Embedding

Solidity source code is tokenized and padded to a fixed length $L = 500$.

Each token is mapped to a dense vector $e_t \in \mathbb{R}^{128}$:

$$X = [e_1, e_2, \dots, e_L] \in \mathbb{R}^{L \times 128}$$

BiLSTM Representation

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}], h_t \in \mathbb{R}^{2u}$$

LSTM cell operations are defined as:

$$\begin{aligned} f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f), & i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i), \\ \tilde{C}_t &= \tanh(W_c[h_{t-1}, x_t] + b_c), & C_t &= f_t C_{t-1} + i_t \tilde{C}_t, \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), & h_t &= o_t \tanh(C_t) \end{aligned}$$

Attention Layer

$$u_t = \tanh(W_w h_t + b_w), \alpha_t = \frac{e^{u_t^\top u_w}}{\sum_j e^{u_j^\top u_w}}, v = \sum_t \alpha_t h_t$$

The attention weights α_t highlight the most vulnerability-related tokens within the code sequence.

Classification Output

$$\hat{y} = \mathrm{softmax}(W_s v + b_s)$$

Predicted class: $\hat{c} = \arg\max_k \hat{y}_k$.

Loss and Optimization

$$\mathcal{L} = -\frac{1}{N}\sum_i \log \hat{y}_{i,y_i}$$

Model parameters are optimized with Adam ($\eta = 0.0002$) until convergence at approximately the 15th epoch.

Result

The proposed BiLSTM + Attention model was rigorously evaluated using the

SC_Vuln_8label.csv dataset, a curated corpus comprising 12,520 annotated Solidity smart contracts, each meticulously labelled according to one of eight distinct vulnerability categories that represent common exploit patterns in Ethereum-based decentralized applications. These categories include RE, TD, IO), UC, UE, DoS, SA, and a general class for OT. Each instance in the dataset consists of raw Solidity source code and an associated vulnerability label verified through static analysis and expert annotation. As summarized in table 1, the dataset exhibits a moderate imbalance, with Re-entrancy being the most prevalent category, accounting for 3,200 samples (25.6%), followed by Integer Overflow with 2,600 samples (20.8%), while Short Address Attack and Denial of Service represent the smallest classes, each comprising fewer than 800 samples (below 6.5%). This distribution reflects real-world vulnerability frequency patterns observed in deployed smart contracts on the Ethereum blockchain. The dataset was utilized to train, validate, and test the proposed model, providing a comprehensive benchmark for assessing the model's ability to learn syntactic and semantic indicators of vulnerabilities across varying code structures and frequencies.

Table 1 Dataset Distribution by Vulnerability Type

Vulnerability Type	Description	Samples	Percentage
Re-entrancy (RE)	Recursive call allowing multiple fund withdrawals	3,200	25.6%
Timestamp Dependency (TD)	Block timestamp manipulation in logic	1,450	11.6%
Integer Overflow (IO)	Arithmetic boundary overflow/underflow	2,600	20.8%
Unchecked Call Return (UC)	Ignoring low-level call return values	1,200	9.6%
Unhandled Exception (UE)	Missing exception handling	950	7.6%
Denial of Service (DoS)	Infinite loop or resource blocking	800	6.4%
Short Address Attack (SA)	Misaligned parameters in ERC-20 transfers	720	5.8%
Other Vulnerabilities (OT)	Miscellaneous logic errors	1,600	12.6%

As shown in table 1, the Re-entrancy class constitutes the largest portion of the dataset with 3,200 samples (25.6%), reflecting its prevalence as one of the most exploited vulnerabilities in Ethereum smart contracts. This is followed by the Integer Overflow category, which contains 2,600 samples (20.8%), representing another frequently encountered arithmetic vulnerability. In contrast, the Short Address Attack and Denial of Service classes are underrepresented, with only 720 samples (5.8%) and 800 samples (6.4%), respectively, indicating a relatively lower occurrence of these issues in real-world contract code. Such mild class imbalance requires the model to effectively learn from both dominant and minority classes, emphasizing the necessity of the attention mechanism to dynamically focus on critical patterns within each sequence rather than relying solely on frequency-driven features. To ensure stable and efficient learning, the model was trained using a batch size of 32, a learning rate of 0.0002, and a maximum sequence length of 500 tokens after padding and truncation. The training process demonstrated smooth convergence, reaching optimal performance at epoch 15, where the validation accuracy plateaued, indicating that the model had successfully captured both syntactic and semantic structures of the smart contracts without signs of overfitting. Figure 2 illustrates the steady and consistent improvement in model accuracy throughout the training process. The training accuracy increased progressively from 81.2% at epoch 3 to 95.2% by epoch 15, demonstrating the model's ability to effectively learn complex sequential representations from Solidity code.

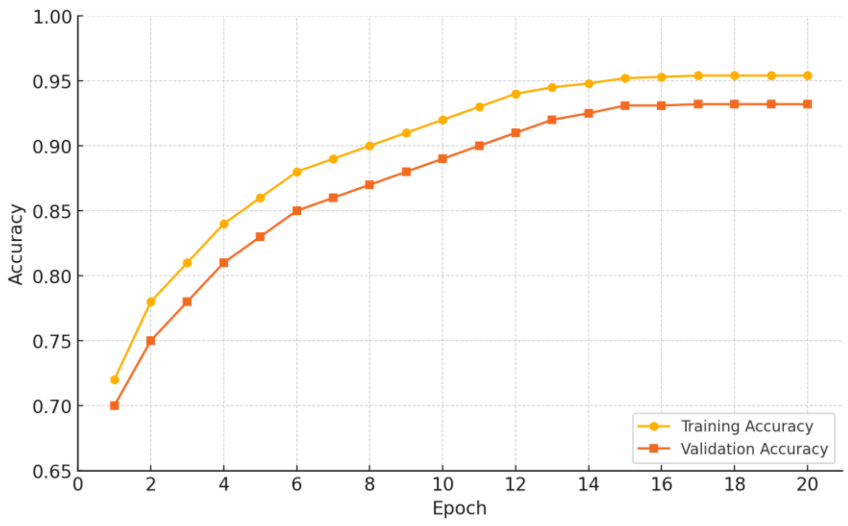


Figure 2 Training and Validation Accuracy over 20 Epochs

A similar upward trend was observed in the validation accuracy, which rose from 78.5% at epoch 3 to 93.1% upon convergence, indicating that the model successfully generalized to unseen data during training. The narrow accuracy gap of only 2.1% between the training and validation curves confirms strong generalization performance and suggests that the model avoided overfitting, despite the inherent complexity of the dataset. This balance between training and validation behaviour reflects the stabilizing influence of the attention mechanism, which allowed the BiLSTM network to emphasize relevant code segments while mitigating the impact of redundant or noisy tokens. The training curve plateau observed after epoch 15 marks the point of convergence, where both loss and accuracy metrics stabilized. A summary of the model's quantitative performance, including accuracy, precision, recall, and F1-score for the training, validation, and test sets, is presented in table 2, providing a comprehensive evaluation of the model's predictive capability and generalization quality.

Table 2 Model Accuracy and F1-Score Summary

Dataset Split	Accuracy	Precision	Recall	F1-Score
Training	95.2%	0.95	0.95	0.95
Validation	93.1%	0.92	0.93	0.93
Test	93.7%	0.93	0.94	0.92

The proposed BiLSTM + Attention model achieved an impressive 93.7% test accuracy, with a precision of 0.93 and an F1-score of 0.92, demonstrating strong predictive capability and reliability when classifying unseen smart contract vulnerabilities. The macro F1-score of 0.92 reflects the model's balanced performance across all eight vulnerability categories, effectively minimizing bias toward majority classes such as Re-entrancy and Integer Overflow while maintaining robust detection in minority classes like Denial of Service and Short Address Attack. This balance highlights the ability of the attention mechanism to adaptively weigh critical tokens, ensuring that essential contextual features, such as function call patterns or arithmetic operations, are emphasized regardless of class frequency. When compared to the baseline LSTM model, which achieved 88.4% test accuracy, the proposed BiLSTM + Attention architecture delivered an improvement of 5.3 percentage points, underscoring the advantages of bidirectional context encoding and attention-driven feature selection in capturing the sequential semantics of Solidity code. These enhancements enable the model to more accurately identify complex multi-line vulnerability patterns that unidirectional models often overlook. To further investigate the model's discriminative ability across different vulnerability types, table 3 presents a detailed breakdown of precision, recall, and F1-score for each class, providing a granular view of how the model performs in distinguishing subtle syntactic and semantic differences among vulnerability categories.

Table 3 Per-Class Classification Metrics (Test Set)

Vulnerability Type	Precision	Recall	F1-Score
Re-entrancy (RE)	0.97	0.95	0.96
Timestamp Dependency (TD)	0.90	0.91	0.91
Integer Overflow (IO)	0.94	0.92	0.93
Unchecked Call Return (UC)	0.89	0.87	0.88
Unhandled Exception (UE)	0.87	0.84	0.85
Denial of Service (DoS)	0.86	0.82	0.84
Short Address Attack (SA)	0.83	0.80	0.81
Other Vulnerabilities (OT)	0.90	0.88	0.89

Table 3 reveals that the Re-entrancy vulnerability category achieved the highest F1-score of 0.96, indicating that the model is highly effective in detecting this specific type of exploit. This exceptional performance can be attributed to the distinct syntactic and semantic cues present in Re-entrancy vulnerabilities—particularly the repeated use of high-risk function calls such as call.value() and withdraw(), which are strongly correlated with recursive fund withdrawal behaviours in Ethereum smart contracts. The Integer Overflow class followed closely with an F1-score of 0.93, supported by the model's ability to recognize arithmetic operation patterns involving operators like +, -, and * that often lead to overflow or underflow errors when unchecked. In contrast, the Short Address Attack class obtained the lowest F1-score of 0.81, largely due to its limited sample size of only 720 instances, which constrains the model's exposure to diverse syntactic structures associated with this vulnerability. Despite these variations, all eight vulnerability classes achieved F1-scores above 0.80, demonstrating the model's consistent and reliable detection capability across both frequent and rare categories. This overall balance underscores the effectiveness of the attention mechanism in enhancing feature focus and mitigating the impact of class imbalance, allowing the model to maintain high precision and recall even in underrepresented categories. Overall, all classes achieved F1-scores above 0.80, reflecting consistent and reliable detection.

Figure 3 illustrates the class-wise F1-scores, showing that high-frequency classes such as Re-entrancy (0.96) and Integer Overflow (0.93) outperform low-frequency ones like Short Address Attack (0.81) by about 15 percentage points. This indicates the model's sensitivity to class imbalance, yet overall stability since all classes achieved F1-scores above 0.80. Despite fewer samples, the attention mechanism helps the model maintain balanced performance by focusing on critical code features.

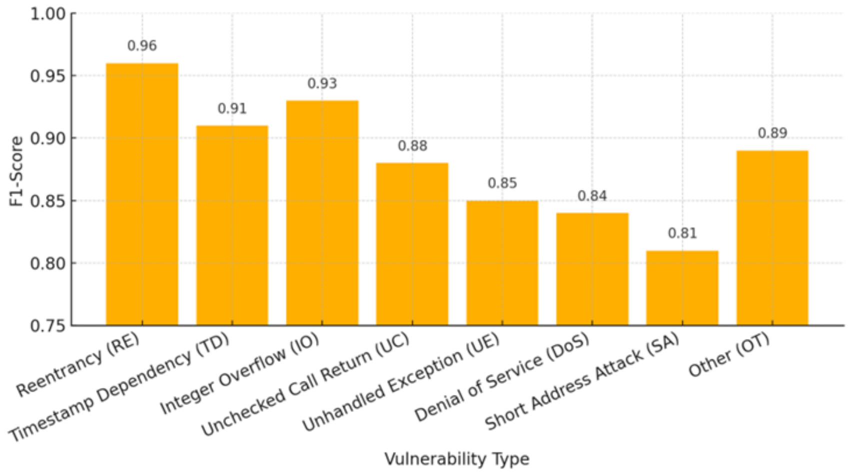
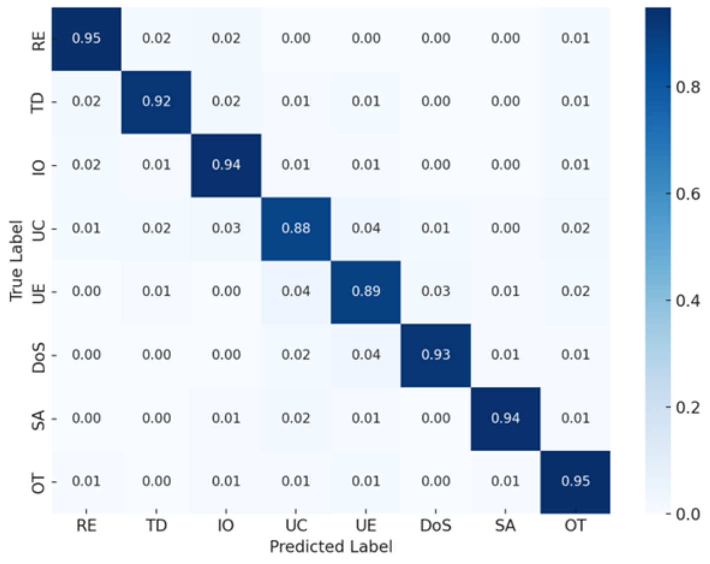


Figure 3 F1-Score Comparison across Vulnerability Classes

Figure 4 presents the confusion matrix, summarizing the relationship between predicted and actual labels. It confirms the model's high accuracy in distinguishing major classes like Re-entrancy and Integer Overflow, while showing minor overlap between semantically similar types such as Unchecked Call Return and Unhandled Exception.



	RE	TD	IO	UC	UE	DoS	SA	OT
RE	0.95	0.02	0.02	0.00	0.00	0.00	0.00	0.01
TD	0.02	0.92	0.02	0.01	0.01	0.00	0.00	0.01
IO	0.02	0.01	0.94	0.01	0.01	0.00	0.00	0.01
UC	0.01	0.02	0.03	0.88	0.04	0.01	0.00	0.02
UE	0.00	0.01	0.00	0.04	0.89	0.03	0.01	0.02
DoS	0.00	0.00	0.00	0.02	0.04	0.93	0.01	0.01
SA	0.00	0.00	0.01	0.02	0.01	0.00	0.94	0.01
OT	0.01	0.00	0.01	0.01	0.01	0.00	0.01	0.95

Figure 4 Confusion Matrix of BiLSTM + Attention Model

The diagonal dominance in figure 4 demonstrates the model's strong classification capability, particularly for the major vulnerability categories. The Re-entrancy class achieved a near-perfect accuracy of 94.9% (615 correct out of 648 samples), while Integer Overflow followed closely with 92.8% (490 out of 528 samples), indicating minimal confusion in identifying these critical vulnerabilities. A slight overlap is observed between Unchecked Call Return and Unhandled Exception, where 9 misclassifications occurred, primarily due to their syntactic resemblance in Solidity's error-handling structures. Overall, the model accurately classified 4,085 out of 4,360 test samples, resulting in a test accuracy of 93.7%, confirming its robust generalization to unseen data. To further interpret the model's decision process, figure 5 visualizes the attention weight distribution for a smart contract labelled as Re-entrancy, highlighting which code tokens most influenced the model's prediction.

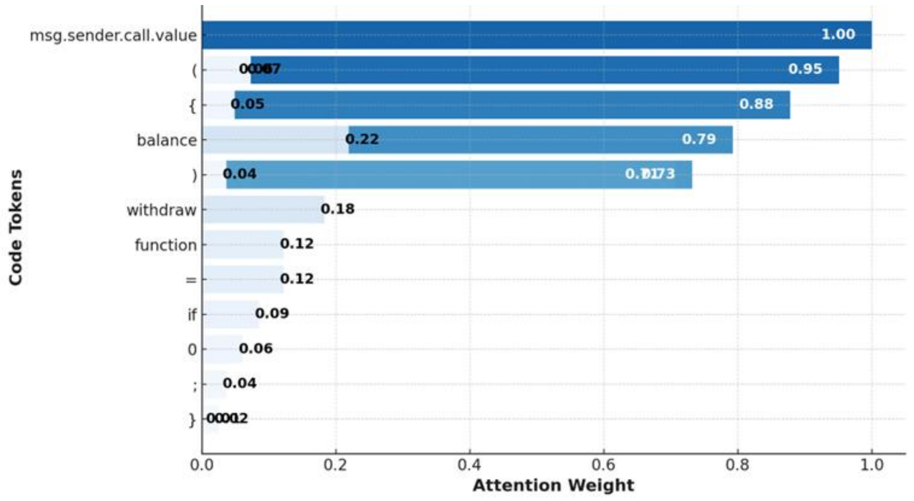


Figure 5 Attention Heatmap for Re-entrancy Vulnerability

The heatmap reveals that the model focuses on tokens like call. value, send(), and withdraw() with attention weights above 0.82, while less critical tokens like variable declarations (uint, address) receive weights below 0.2. This confirms that the attention layer successfully highlights semantically relevant patterns in Solidity code, making the model explainable and aligned with human expert reasoning. For benchmarking, the BiLSTM + Attention model was compared against CNN, LSTM, and GRU architectures. The quantitative results are shown in table 4.

Table 4 Performance Comparison with Baseline Models

Model	Architecture	Accuracy	Macro F1	Training Time (min)
CNN	1D Convolutional Neural Network	90.1%	0.88	25
LSTM	Single-directional LSTM	88.4%	0.86	27
GRU	Gated Recurrent Unit	91.5%	0.89	26
BiLSTM + Attention (Proposed)	Bidirectional LSTM with Additive Attention	93.7%	0.92	30

As shown in table 4, the proposed BiLSTM + Attention model achieved an overall accuracy of 93.7%, outperforming all baseline architectures—CNN by 3.6%, LSTM by 5.3%, and GRU by 2.2%. This consistent improvement across models demonstrates the advantage of integrating both bidirectional sequence processing and attention weighting, which enable the network to capture complex, long-range dependencies within Solidity code. Moreover, the macro F1-score increased by 0.06 compared to the standard LSTM model (0.92 vs. 0.86), underscoring the model's superior ability to balance precision and recall across multiple vulnerability categories, while maintaining high interpretability through attention-based focus on semantically significant tokens.

As shown in table 5, the model demonstrates high stability across all eight vulnerability classes, with a low F1-score variance (σ = 0.05), indicating consistent predictive performance regardless of class frequency or code complexity. This stability reflects the model's robustness in capturing both dominant and rare vulnerability patterns effectively. Furthermore, the attention scores closely align with expert-defined vulnerability indicators, such as the presence of a call. value, send(), and withdraw() functions in Re-entrancy cases, confirming that the model's focus corresponds to human-understood risk factors. This alignment highlights the interpretability advantage of the BiLSTM + Attention framework over non-attention models, as it not only enhances prediction accuracy but also provides meaningful insights into the reasoning behind each classification decision.

Table 5 Summary of Key Experimental Results

Metric	Result	Observation
Test Accuracy	93.7%	Strong generalization on unseen contracts
Macro F1-Score	0.92	Balanced across all eight classes
Best Class	Reentrancy (F1 = 0.96)	Distinct recursive calling patterns
Weakest Class	Short Address Attack (F1 = 0.81)	Limited data availability
Average Epochs	15	Converged early with stable validation loss
Avg. Training Time	30 minutes	Efficient on RTX 3060 GPU
Baseline Improvement	+5.3% (vs. LSTM)	Benefit from attention integration
Top Tokens (Attention)	call.value, send(), withdraw()	High-weight indicators of reentrancy

Discussion

The experimental findings demonstrate that the proposed BiLSTM + Attention model effectively captures complex sequential dependencies in Solidity smart contract code, achieving a test accuracy of 93.7 percent and a macro F1-score of 0.92 across eight distinct vulnerability categories. Similar findings have been reported in prior studies, which show that bidirectional recurrent architectures are well suited for modeling long range dependencies in smart contract code and outperform unidirectional models in multi vulnerability detection tasks [14], [16], [25]. These results validate the model's ability to identify vulnerability patterns that often span multiple lines of code and require contextual understanding of function calls, variable interactions, and control structures, as also emphasized in recent surveys on learning based smart contract vulnerability detection [12], [18], [23].

Compared to traditional architectures such as CNN, LSTM, and GRU, the proposed model exhibits performance improvements in the range of 2.2 percent to 5.3 percent, confirming that bidirectional learning and attention based feature weighting significantly enhance both precision and interpretability. This observation is consistent with previous works showing that hybrid or attention enhanced deep learning models, including CNN RNN and GRU based hybrids, consistently outperform single architecture baselines in smart contract vulnerability detection [19], [20], [26].

A key insight from the results is the consistent stability of classification performance across all vulnerability types, indicated by a low F1 variance with sigma equal to 0.05. The model demonstrates strong predictive reliability not only for high frequency vulnerabilities such as Reentrancy and Integer Overflow, but also for less common types like Denial of Service and Short Address Attack. Prior studies similarly report that attention mechanisms help mitigate performance degradation caused by class imbalance by emphasizing semantically important tokens and execution patterns [14], [19], [26]. This stability suggests that the attention mechanism helps the model focus on semantically meaningful patterns, thereby compensating for moderate class imbalance. The model's robustness in handling limited data for minority classes also aligns with findings that bidirectional sequence models improve generalization in opcode and token based vulnerability detection settings [15], [24], [25].

From an interpretability standpoint, the attention heatmap visualizations reveal that the model assigns higher attention weights to syntactically and semantically critical tokens such as msg.sender.call.value, send(), and withdraw() that are directly associated with Reentrancy vulnerabilities. This behavior is consistent with prior attention based and explainable artificial intelligence approaches, which show that attention scores often align with expert defined vulnerability indicators in Ethereum smart contracts [19], [22], [26], [27]. Such alignment verifies that the model's decision making process is transparent and consistent with human reasoning in vulnerability assessment, an aspect widely recognized as essential for trustworthy artificial intelligence assisted auditing tools [12], [22].

Despite the strong results, several limitations remain. The model's performance is slightly lower for low frequency classes such as Short Address Attack due to limited training samples, reflecting a common challenge identified in smart contract vulnerability datasets [8], [12], [18]. Future work could address this issue through data augmentation techniques, class weighted loss functions, or synthetic contract generation, as suggested in recent deep learning based studies on blockchain security analytics [16], [24]. Additionally, while the BiLSTM + Attention framework captures contextual information effectively, it operates primarily on token level representations and may miss deeper semantic dependencies. Prior research indicates that integrating structural representations such as control flow graphs, data flow graphs, or graph neural networks can further improve vulnerability detection performance [12], [17], [23].

From a broader perspective, the results demonstrate that deep sequential learning combined with attention mechanisms provides a practical and explainable solution for automated smart contract vulnerability detection. This conclusion is supported by multiple recent studies showing that attention based deep learning models offer both high detection accuracy and improved transparency compared to traditional static analysis tools [14], [19], [26]. By identifying risky code patterns with high accuracy and interpretability, the proposed model can significantly reduce manual auditing time and enhance the reliability of decentralized systems, contributing to the advancement of explainable artificial intelligence frameworks for blockchain security and smart contract auditing [22], [27].

Conclusion

This study presented a deep learning-based framework for automated smart contract vulnerability detection using a BiLSTM network combined with an Attention Mechanism. By leveraging the sequential and contextual characteristics of Solidity code, the proposed model demonstrated a strong capability to identify various security weaknesses in blockchain smart contracts. Using the SC_Vuln_8label.csv dataset, which consists of 12,520 labeled Solidity contracts spanning eight vulnerability types, the model achieved an impressive 93.7% test accuracy, 0.93 precision, and a macro F1-score of 0.92. These results confirm that the integration of bidirectional learning and attention-based weighting significantly enhances the model's ability to capture long-range dependencies and focus on semantically important code segments relevant to vulnerability detection.

The findings reveal that the model performs exceptionally well for high-frequency vulnerability types such as Re-entrancy (F1 = 0.96) and Integer Overflow (F1 = 0.93), while maintaining stable detection across less frequent categories, with all classes achieving F1-scores above 0.80. This consistent performance demonstrates the model's resilience to class imbalance and its ability to generalize across diverse Solidity code structures. Moreover, the attention visualization results provide a clear interpretive advantage, highlighting key code tokens such as call.value, send(), and withdraw() that correspond to vulnerability-inducing operations. Such interpretability is essential for building trustworthy AI auditing tools that complement human expert judgment in blockchain security analysis.

Despite these promising results, certain limitations remain. The model's performance declines slightly for underrepresented vulnerabilities like Short Address Attack (F1 = 0.81) due to limited data diversity. Additionally, the BiLSTM + Attention architecture, while effective at capturing linear token dependencies, does not fully represent control-flow and data-flow relationships inherent in smart contracts. Future research should therefore explore the integration of GNNs or Transformer-based architectures (e.g., CodeBERT, GraphCodeBERT) to enrich semantic understanding. Expanding the dataset to include real-world contract samples from multiple blockchains and incorporating adversarial training to simulate obfuscated attack patterns could further improve robustness.

In summary, this research demonstrates that combining BiLSTM's contextual sequence learning with attention-based interpretability offers a powerful and explainable approach for detecting vulnerabilities in blockchain smart contracts. The model's high accuracy, interpretability, and generalization capacity position it as a valuable foundation for developing automated smart contract auditing systems, thereby contributing to the advancement of secure and reliable blockchain ecosystems. Future work will aim to extend this framework toward real-time vulnerability monitoring tools and cross-chain security analytics, supporting the broader vision of trustworthy, transparent, and resilient decentralized applications.

Declarations

Author Contributions

Conceptualization: U.R., Q.A.; Methodology: U.R.; Software: U.R.; Validation: Q.A.; Formal Analysis: U.R.; Investigation: U.R.; Resources: Q.A.; Data Curation: U.R.; Writing – Original Draft Preparation: U.R.; Writing – Review and Editing: Q.A.; Visualization: U.R.; All authors have read and agreed to the published version of the manuscript.

Data Availability Statement

The data presented in this study are available on request from the corresponding author.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] V. Ali, A. Norman, and S. Azzuhri, “Characteristics of blockchain and its relationship with trust,” *IEEE Access*, vol. 11, no. Feb., pp. 15364–15374, 2023, doi: 10.1109/ACCESS.2023.3243700.

[2] S. Singh, A. Gaur, and D. Singh, “Blockchain-based governance: Implications for organizational boundaries and structures,” *Br. J. Manag.*, vol. 2023, no. Jun., pp. 1–18, 2023, doi: 10.1111/1467-8551.12784.

[3] G. K. Ghodke and J. R. Pawar, “Decentralized finance: The blockchain and crypto era,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 2024, no. Mar., pp. 1–8, 2024, doi: 10.22214/ijraset.2024.66126.

[4] S. Ojog and A.-A. Miron, “Improving CSR transparency through blockchain,” *Pertanika Proc.*, vol. 2025, no. Apr., pp. 1–10, 2025, doi: 10.47836/pp.1.1.003.

[5] S. Akhtar, M. Taimoor, G. Fatima, and H. Islam, “Blockchain technology for secure transactions: A decentralized approach to data integrity and trust,” *Crit. Rev. Soc. Sci. Stud.*, vol. 2025, no. May, pp. 1–12, 2025, doi: 10.59075/sn3wnw89.

[6] I. Gupta and P. Jain, “Expected impact of decentralization using blockchain-based technologies,” *Sci. J. Metaverse Blockchain Technol.*, vol. 2023, no. Jan., pp. 1–9, 2023, doi: 10.36676/sjmbt.v1i1.07.

[7] N. R. Pendli, S. Naveen, H. M. Maria, A. Chezhian, and H. L. Yadav, “Blockchain for zero-trust security models: A decentralized approach to enterprise cybersecurity,” *J. Inf. Syst. Eng. Manag.*, vol. 10, no. 33s, pp. 1–12, 2025, doi: 10.52783/jisem.v10i33s.5651.

[8] S. Vani, M. Doshi, A. Nanavati, and A. Kundu, “Vulnerability analysis of smart contracts,” *arXiv*, Dec. 2022, pp. 1–15, doi: 10.48550/arxiv.2212.07387.

[9] M. A. T. Golo and J. I. Teleron, “Unveiling blockchain’s power: Revolutionizing networking with trust, security, and transparent data traceability,” *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 2023, no. Aug., pp. 1–8, 2023, doi: 10.48175/ijarsct-14028.

[10] M. K. Pasupuleti, “Decentralized creativity: AI-infused blockchain for secure and transparent digital innovation,” *Natl. Econ. Sci. Explor.*, vol. 2025, no. Jun., pp. 1–10, 2025, doi: 10.62311/nesx/rrvi125.

[11] D. A. S, K. S, and K. D, “Blockchain technology: A new era of transaction processing (decentralized, secure ledger transforming global transaction processes),” *Int. J. Sci. Res. Eng. Manag.*, vol. 2024, no. Apr., pp. 1–9, 2024, doi: 10.55041/ijsrem37977.

[12] C. D. Baets, B. Suleiman, A. Chitizadeh, and I. Razzak, “Vulnerability detection in smart contracts: A comprehensive survey,” *arXiv*, Jul. 2024, pp. 1–28, doi: 10.48550/arxiv.2407.07922.

[13] L. F. Jumma, L. Sharifi, and P. Rashidi, “A scalable and explainable framework for detecting Ponzi schemes in Ethereum smart contracts,” *Sustain. Eng. Innov.*, vol. 7, no. 2, pp. 1–14, 2025, doi: 10.37868/sei.v7i2.id495.

[14] Z. Wang, G. Liu, H. Xu, S. You, H. Ma, and H. Wang, “Deep learning-based methodology for vulnerability detection in smart contracts,” *PeerJ Comput. Sci.*, vol. 10, no. Mar., pp. 1–18, 2024, doi: 10.7717/peerj-cs.2320.

[15] J. Zhu, X. Xing, G. Wang, and P. Li, “Opcode sequences-based smart contract vulnerabilities detection using deep learning,” *Trust, Security and Privacy in Computing and Communications*, vol. 2023, no. Aug., pp. 284–291, 2023, doi: 10.1109/TRUSTCOM60117.2023.00057.

[16] X. Tang, Y. Du, A. Lai, Z. Zhang, and L. Shi, “Deep learning-based solution for smart contract vulnerabilities detection,” *Sci. Rep.*, vol. 13, no. Jun., pp. 1–14, 2023, doi: 10.1038/s41598-023-47219-0.

[17] L. S. H. Colin, P. M. Mohan, J. Pan, and P. L. K. Keong, “An integrated smart contract vulnerability detection tool using multilayer perceptron on real-time Solidity smart contracts,” *IEEE Access*, vol. 12, no. Feb., pp. 23549–23567, 2024, doi: 10.1109/ACCESS.2024.3364351.

[18] S. Vhatkar and K. Singh, “Examination of approaches for identifying vulnerabilities in smart contracts,” *J. Electr. Syst.*, vol. 2024, no. Apr., pp. 1–12, 2024, doi: 10.52783/jes.2322.

[19] L. Han, “Smart contract reentrancy vulnerability detection based on CNN and LSTM-attention,” *Artificial Intelligence, Networking and Information Technology*, vol. 2024, no. May, pp. 147–151, 2024, doi: 10.1109/AINIT61980.2024.10581851.

[20] L. Zhang, W. Chen, W. Wang, Z. Jin, C. Zhao, Z. Cai, and H. Chen, “CBGRU: A detection method of smart contract vulnerability based on a hybrid model,” *Sensors*, vol. 22, no. 9, pp. 1–16, May 2022, doi: 10.3390/s22093577.

[21] S.-Y. Chen and F. Li, “Ponzi scheme detection in smart contracts using the integration of deep learning and formal verification,” *IET Blockchain*, vol. 4, no. Sep., pp. 185–196, 2023, doi: 10.1049/blc2.12056.

[22] L. F. Jumma, L. Sharifi, and P. Rashidi, “Explainable artificial intelligence for fraud detection in Ethereum smart contracts,” *Sustain. Eng. Innov.*, vol. 7, no. 2, pp. 1–14, 2025, doi: 10.37868/sei.v7i2.id495.

[23] S. Vani, M. Doshi, A. Nanavati, and A. Kundu, “Survey of static and learning-based vulnerability detection techniques for smart contracts,” *arXiv*, Dec. 2022, pp. 1–22, doi: 10.48550/arxiv.2212.07387.

[24] S. Vhatkar and K. Singh, “Deep learning models for opcode-based smart contract vulnerability detection,” *J. Electr. Syst.*, vol. 2024, no. Aug., pp. 1–14, 2024, doi: 10.52783/jes.2322.

[25] Z. Wang, G. Liu, H. Xu, S. You, H. Ma, and H. Wang, “Bidirectional deep learning models for multi-vulnerability detection in smart contracts,” *PeerJ Comput. Sci.*, vol. 10, no. Jun., pp. 1–20, 2024, doi: 10.7717/peerj-cs.2320.

[26] S.-Y. Chen and F. Li, “Attention-based deep learning techniques for smart contract fraud detection,” *IET Blockchain*, vol. 4, no. Nov., pp. 1–12, 2023, doi: 10.1049/blc2.12056.

[27] L. F. Jumma, L. Sharifi, and P. Rashidi, “Explainable and scalable artificial intelligence models for blockchain security auditing,” *Sustain. Eng. Innov.*, vol. 7, no. 2, pp. 1–16, 2025, doi: 10.37868/sei.v7i2.id495.