Explainable Machine Learning for Smart Contract Vulnerability Detection in Ethereum Blockchain

Vasantakumaren Seri Ramaloo[1,*]

[1]Faculty of Data Science and Information Technology (FDSIT), INTI International University, Nilai, Malaysia

ABSTRACT

Smart contracts play an important role in Ethereum blockchain systems by enabling decentralized and automated transactions without intermediaries. However, vulnerabilities in Solidity smart contracts remain a major security issue because attackers can exploit coding weaknesses such as reentrancy attacks, unsafe external calls, and insecure access control mechanisms. Traditional vulnerability detection methods, including manual auditing and static analysis, often require high expertise and lack interpretability. Therefore, this study proposes an Explainable Machine Learning framework for smart contract vulnerability detection using Solidity source code analysis. The proposed framework integrates TF-IDF feature extraction, XGBoost classification, and SHAP (SHapley Additive Explanations) to provide high detection performance and transparent model interpretability. A dataset consisting of 691 Ethereum smart contracts, including 346 safe contracts and 345 vulnerable contracts, was used in this study. The experimental results show that the proposed model achieved an accuracy of 97.84%, with precision, recall, and F1-score values above 97%. The ROC analysis produced an AUC score of 0.9946, indicating excellent classification capability between vulnerable and safe smart contracts. The SHAP analysis revealed that features such as success, staticcall, gas, and origin significantly influenced vulnerability prediction, while security mitigation features such as nonreentrant and reentrancyguard were also identified as important indicators. The results demonstrate that the proposed framework provides accurate, interpretable, and transparent vulnerability detection for Ethereum smart contracts and can assist blockchain developers and security auditors in improving blockchain security analysis.

Keywords Explainable Machine Learning, Smart Contract Vulnerability Detection, Ethereum Blockchain, Solidity Security Analysis, SHAP Explainability

INTRODUCTION

Blockchain technology has rapidly evolved as a decentralized system that enables secure and transparent digital transactions without centralized authorities [1]. Among various blockchain platforms, Ethereum is one of the most widely adopted ecosystems due to its support for smart contracts and decentralized applications (DApps) [2]. Smart contracts are self-executing programs deployed on the blockchain that automatically execute predefined operations when specific conditions are satisfied [3]. These contracts are widely used in decentralized finance (DeFi), non-fungible tokens (NFTs), digital asset management systems, healthcare systems, and supply chain management applications.

Despite their advantages, smart contracts remain vulnerable to multiple security threats caused by insecure coding practices and programming errors [4]. Vulnerabilities such as reentrancy attacks, unsafe external calls, integer overflow, and insecure access control mechanisms can be exploited by

Submitted 10 January 2026
Accepted 25 February 2026
Published 31 August 2026

Corresponding author
Vasantakumaren Seri Ramaloo, vasantakumaren.seri@newinti.edu.my

Additional Information and Declarations can be found on page 230

DOI: 10.47738/jcrb.v3i3.81

How to cite this article: V. S. Ramaloo, "Explainable Machine Learning for Smart Contract Vulnerability Detection in Ethereum Blockchain," *J. Curr. Res. Blockchain*, vol. 3, no. 3, pp. 216-232, 2026.

attackers to compromise blockchain systems and steal digital assets [5]. One of the most significant incidents was the DAO attack, where attackers exploited a reentrancy vulnerability and caused financial losses worth millions of dollars [6]. Since smart contracts are immutable after deployment, vulnerabilities become difficult to patch once deployed on the blockchain network. Therefore, smart contract vulnerability detection has become an important research area in blockchain security analysis [7].

Traditional vulnerability detection approaches mainly rely on manual auditing and static analysis tools such as Slither, Mythril, and Oyente [8]. Although these approaches can identify predefined vulnerability patterns, they often require extensive expertise and may produce false positive results. In addition, static analysis methods may face scalability limitations when processing large volumes of Solidity source code. These limitations motivate researchers to develop automated vulnerability detection approaches using machine learning techniques.

Recent studies have demonstrated that machine learning algorithms can effectively detect vulnerabilities within Solidity smart contracts [9]. Several approaches have utilized Random Forest, Support Vector Machine (SVM), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) models for vulnerability classification. These methods achieved promising classification accuracy and demonstrated the capability of machine learning techniques in identifying security-related patterns from source code analysis.

However, most existing machine learning-based approaches still operate as black-box models that provide predictions without clear explanations regarding how the decisions are generated [10]. In blockchain security environments, explainability is critically important because developers and security auditors require interpretable systems to understand why a smart contract is classified as vulnerable. The lack of interpretability reduces trustworthiness and limits the practical adoption of machine learning models for real-world blockchain auditing processes.

Based on the state-of-the-art analysis, previous studies primarily focused on improving vulnerability detection accuracy while paying limited attention to explainability and transparency [11]. Existing methods rarely analyze the contribution of individual Solidity features toward vulnerability prediction results. Consequently, security analysts often experience difficulties in understanding which coding patterns significantly influence vulnerability detection decisions.

To address this research gap, this study proposes an Explainable Machine Learning framework for smart contract vulnerability detection in Ethereum blockchain using Solidity source code analysis. The proposed framework integrates TF-IDF feature extraction, XGBoost classification, and SHAP (SHapley Additive Explanations) to provide both high detection performance and interpretable prediction results. The proposed approach not only detects vulnerable smart contracts but also explains the contribution of Solidity features associated with security vulnerabilities and mitigation mechanisms.

The main contributions of this study are as follows. First, this study proposes a machine learning-based framework for Ethereum smart contract vulnerability detection using Solidity source code analysis. Second, the framework integrates SHAP explainability analysis to improve transparency and interpretability in

blockchain security detection. Third, the proposed framework identifies important Solidity features associated with vulnerability patterns and security mitigation mechanisms. Finally, the experimental results demonstrate that the proposed approach achieves high classification performance while maintaining explainable and interpretable predictions for blockchain security analysis.

Literature Review

Blockchain is a decentralized distributed ledger technology that enables secure and transparent digital transactions without centralized control [12]. Ethereum extends blockchain functionality by supporting smart contracts, which are self-executing programs deployed on the blockchain network [13]. Smart contracts automatically execute predefined rules and conditions, enabling decentralized applications (DApps) in various domains such as finance, healthcare, supply chain management, and digital asset trading [14].

Solidity is the primary programming language used for Ethereum smart contract development. Although Solidity provides flexibility for decentralized application development, programming errors and insecure coding practices may introduce security vulnerabilities into smart contracts [15]. Since deployed smart contracts are immutable, vulnerabilities can lead to irreversible financial losses and exploitation attacks.

Smart contract vulnerabilities represent security weaknesses that can be exploited by attackers to manipulate blockchain systems or steal digital assets [16]. One of the most common vulnerabilities is the reentrancy attack, where malicious contracts repeatedly invoke external calls before the contract state is updated [17]. Other common vulnerabilities include integer overflow, unsafe external calls, denial of service attacks, and insecure access control mechanisms [18].

Several high-profile blockchain attacks have demonstrated the severe impact of smart contract vulnerabilities. The DAO attack remains one of the most significant incidents in Ethereum history, where attackers exploited a reentrancy vulnerability to steal millions of dollars worth of cryptocurrency [19]. These incidents highlight the importance of implementing reliable vulnerability detection techniques for blockchain security analysis.

Machine learning has recently gained attention as an effective approach for automated vulnerability detection in smart contracts [20]. Unlike traditional static analysis methods, machine learning techniques can learn complex security-related patterns directly from source code data [21]. Previous studies have utilized algorithms such as Random Forest, Support Vector Machine (SVM), Naïve Bayes, and XGBoost for smart contract classification tasks [22].

Deep learning approaches, including Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM), have also been explored to improve vulnerability detection performance [23]. These models can capture semantic and sequential relationships within Solidity source code, enabling more accurate vulnerability prediction. In addition, transformer-based architectures and graph neural networks have recently been introduced to analyze smart contract structures more effectively [24].

Although machine learning approaches have demonstrated promising classification accuracy, many existing models still operate as black-box systems

that provide limited interpretability regarding prediction decisions [25].

Explainable Artificial Intelligence (XAI) refers to techniques that improve transparency and interpretability in machine learning systems [26]. In cybersecurity and blockchain environments, explainability is essential because security analysts require understandable explanations regarding why a system identifies certain activities as malicious or vulnerable [27].

SHAP (SHapley Additive Explanations) is one of the most widely used explainability methods for machine learning interpretation [28]. SHAP calculates feature contribution values based on Shapley theory from cooperative game theory, enabling detailed analysis of how individual features influence model predictions [29]. Previous studies have shown that SHAP can effectively improve transparency in cybersecurity, malware detection, fraud detection, and intrusion detection systems [30].

In smart contract vulnerability analysis, explainability techniques can help developers and auditors understand which Solidity features contribute significantly to vulnerability prediction. This capability improves trustworthiness and supports more reliable blockchain security auditing processes.

Method

This study proposes an Explainable Machine Learning framework for detecting vulnerabilities in Ethereum smart contracts using Solidity source code analysis. The proposed framework integrates data preprocessing, feature extraction, machine learning classification, and explainability analysis to provide accurate and interpretable vulnerability detection. The overall research workflow is presented in figure 1.

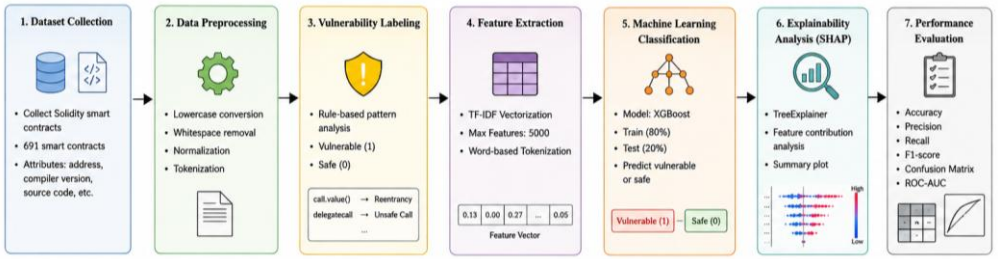

Figure 1 Proposed Research Framework

The framework consists of several stages, including dataset collection, data preprocessing, vulnerability labeling, feature extraction, machine learning classification, explainability analysis, and performance evaluation.

Dataset Collection

The dataset used in this study consists of Ethereum smart contracts collected from Solidity source code repositories. The dataset contains 691 smart contracts categorized into vulnerable and safe contracts. Several attributes were utilized during the analysis process, including smart contract addresses, compiler versions, and Solidity source code. A detailed description of the dataset attributes is presented in table 1. The dataset contains both metadata and source-code attributes that provide information about the characteristics of each smart contract. Among these attributes, CleanCode was selected as the

primary feature source because it contains preprocessed Solidity source code that has been cleaned and normalized for subsequent machine learning analysis. The PureSolidityCode attribute was retained as the original source-code reference, while the remaining metadata attributes provide contextual information about each contract.

Table 1 Dataset Description

Attribute	Description
ContractAddress	Ethereum smart contract address
ContractName	Name of smart contract
CompilerVersion	Solidity compiler version
LicenseType	License information
Proxy	Proxy contract indicator
CleanCode	Preprocessed Solidity source code
PureSolidityCode	Original Solidity source code

The CleanCode attribute was selected as the primary feature source because it contains cleaned Solidity source code suitable for machine learning analysis.

Data Preprocessing

Data preprocessing was performed to prepare Solidity source code before feature extraction and classification. The preprocessing stage included lowercase conversion, whitespace removal, normalization, and tokenization. This process helps reduce noise within the dataset and improves the quality of feature representation. The preprocessing workflow used in this study is illustrated in figure 2.

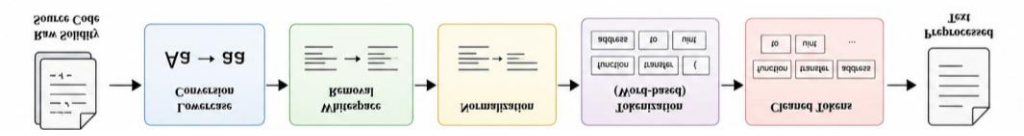

Figure 2 Smart Contract Processing Pipeline

The processed Solidity source code was transformed into normalized textual representations before numerical feature extraction.

Vulnerability Labeling

Vulnerability labeling was conducted using rule-based pattern analysis to classify smart contracts into vulnerable and safe categories. Several Solidity patterns commonly associated with security vulnerabilities were used during the labeling process. The vulnerability patterns and their corresponding vulnerability categories are presented in table 2.

Table 2 Vulnerability Labeling Rules

Solidity Pattern	Vulnerability Type
call.value()	Reentrancy
delegatecall	Unsafe External Call
tx.origin	Access Control Issue
selfdestruct	Dangerous Operation
send()	Unchecked External Call

Smart contracts containing these patterns were categorized as vulnerable contracts, while contracts without these patterns were categorized as safe contracts. This labeling process enabled the creation of a binary classification dataset for machine learning analysis. The vulnerability labeling process can be represented as follows:

$$Label(x) = 1 \text{ if vulnerability pattern exists}$$
$$Label(x) = 0 \text{ otherwise} \quad (1)$$

In this equation, $Label(x)$ represents the classification result of a smart contract. A value of 1 indicates that the smart contract contains vulnerability-related patterns, while a value of 0 indicates that the contract is categorized as safe.

Feature Extraction

Feature extraction was performed using the TF-IDF (Term Frequency–Inverse Document Frequency) vectorization technique. TF-IDF converts textual Solidity source code into numerical feature vectors based on term importance within the dataset. This method helps identify significant Solidity terms that contribute to vulnerability detection.

$$TF\text{-}IDF(t,d) = TF(t,d) \times \log\left(\frac{N}{DF(t)}\right) \quad (2)$$

In this equation, $TF\text{-}IDF(t,d)$ represents the frequency of term t in document d, while $DF(t)$ represents the number of documents containing the term t. The variable N represents the total number of documents within the dataset. TF-IDF assigns higher weights to terms that frequently appear in a specific document but rarely appear across other documents, making the features more informative for classification. The TF-IDF configuration used in this study is presented in table 3.

Table 3 Feature Extraction Configuration

Parameter	Value
Vectorization Method	TF-IDF
Max Features	5000
Text Preprocessing	Lowercase and whitespace removal
Tokenization	Word-based

The TF-IDF vectorization process generated a feature matrix consisting of 5000 numerical features extracted from Solidity source code.

Machine Learning Classification

The classification process utilized the XGBoost algorithm to classify vulnerable and safe smart contracts. XGBoost is a gradient boosting-based machine learning algorithm known for its high classification performance and computational efficiency in structured data analysis.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \quad (3)$$

In this equation, $\hat{y}_i$ represents the predicted output for input data x_i, while f_k represents the decision tree function used within the boosting process. The variable K represents the total number of decision trees used in the model. XGBoost combines multiple decision trees iteratively to improve classification performance and reduce prediction errors. The hyperparameter configuration used in this study is shown in table 4.

Table 4 Machine Learning Parameters

Parameter	Value
Model	XGBoost
n_estimators	100

max_depth	6
learning_rate	0.1
objective	binary:logistic

The dataset was divided into training and testing sets using an 80:20 ratio. The XGBoost model was trained using the training dataset and evaluated using the testing dataset.

Explainability Analysis

To improve model interpretability, this study implemented SHAP (SHapley Additive Explanations) to analyze the contribution of Solidity features toward vulnerability prediction. SHAP provides transparent explanations regarding how individual features influence the prediction results generated by the machine learning model.

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!} [f(S \cup \{i\}) - f(S)] \quad (4)$$

In this equation, ϕ_i represents the SHAP value of feature i, which measures the contribution of a specific feature toward the prediction result. The variable F represents the complete set of features, while S represents a subset of features used during the contribution calculation. SHAP evaluates how much each feature contributes to increasing or decreasing the model prediction output. The explainability workflow used in this study is illustrated in figure 3.

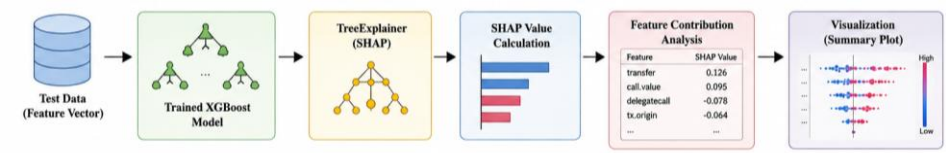

Figure 3 Explainable AI Workflow

The SHAP configuration used in this study is presented in table 5.

Table 5 SHAP Configuration

Component	Method
Explainability Method	SHAP
Explainer Type	TreeExplainer
Output Interpretation	Feature contribution analysis
Visualization	Summary Plot

The SHAP analysis enabled interpretation of feature importance and provided transparent explanations regarding how Solidity source code features influenced vulnerability prediction results.

Performance Evaluation

The performance of the proposed framework was evaluated using accuracy, precision, recall, F1-score, confusion matrix analysis, and ROC-AUC analysis. Accuracy was used to measure the overall correctness of the classification model in identifying vulnerable and safe smart contracts. Precision was utilized to evaluate how many smart contracts predicted as vulnerable were actually vulnerable, while recall measured the capability of the model in detecting vulnerable smart contracts from all vulnerable instances within the dataset.

The F1-score was used to evaluate the balance between precision and recall. In addition, confusion matrix analysis was performed to provide detailed information regarding correct and incorrect classifications generated by the model. ROC-AUC analysis was also conducted to measure the discrimination capability of the proposed framework in distinguishing vulnerable and safe smart contracts across different classification thresholds.

These evaluation metrics were used to comprehensively assess the effectiveness and reliability of the proposed Explainable Machine Learning framework for Ethereum smart contract vulnerability detection.

Result and Discussion

Dataset Distribution

The dataset used in this study consists of 691 Ethereum smart contracts obtained from Solidity source code repositories. The contracts were categorized into two classes, namely safe contracts and vulnerable contracts. The dataset distribution is relatively balanced, consisting of 346 safe contracts and 345 vulnerable contracts, which helps reduce classification bias during model training and evaluation. The distribution of the two classes is presented in table 6.

Table 6 Dataset Distribution

Class	Number of Samples
Safe	346
Vulnerable	345

As shown in table 6, the difference between the two classes is only one contract, indicating a highly balanced dataset. This relatively balanced distribution reduces the potential for substantial class bias during model training and evaluation and provides a suitable basis for binary classification of smart contract vulnerabilities.

Classification Performance

The proposed Explainable Machine Learning framework employed the XGBoost classifier combined with TF-IDF feature extraction for vulnerability detection in Solidity smart contracts. The evaluation results demonstrate that the proposed model achieved excellent classification performance across all evaluation metrics. The classification performance results are presented in table 7.

Table 7 Classification Performance

Metric	Score
Accuracy	97.84%
Precision	97.85%
Recall	97.84%
F1-Score	97.84%

The model achieved an accuracy score of 97.84%, indicating that the proposed approach successfully classified most smart contracts correctly. The precision and recall scores also demonstrate that the model maintained a strong balance between detecting vulnerable contracts and minimizing incorrect predictions. These results indicate that the proposed framework is highly effective for smart contract vulnerability detection tasks. In cybersecurity applications, reducing false negative predictions is extremely important because undetected vulnerabilities may expose blockchain systems to severe security risks. The

obtained results show that the proposed framework successfully minimizes such risks.

Confusion Matrix Analysis

To provide a more detailed assessment of the classification results, a confusion matrix was generated to examine the relationship between the actual and predicted classes. The numerical confusion matrix results are presented in table 8, while the corresponding visualization is shown in figure 4.

Table 8 Confusion Matrix Results

Actual / Predicted	Safe	Vulnerable
Safe	68	2
Vulnerable	1	68

The confusion matrix indicates that 68 safe contracts and 68 vulnerable contracts were correctly classified by the model. Only three contracts were misclassified during the testing process. This result demonstrates that the proposed model has a very low error rate and maintains highly reliable detection performance.

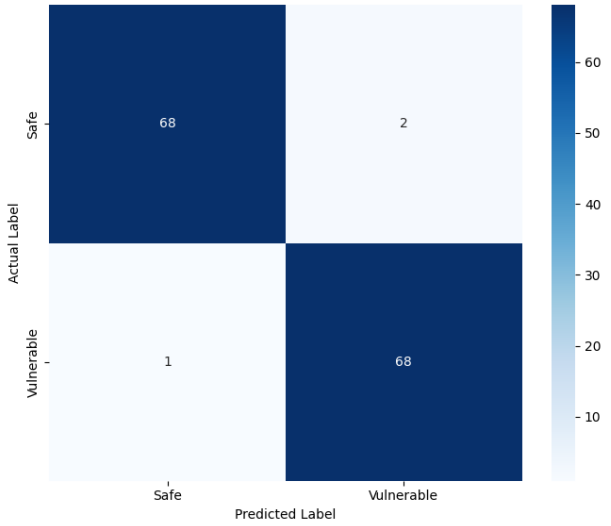

Figure 4 Confusion Matrix

The low number of false positive and false negative predictions further confirms the robustness of the proposed machine learning framework for blockchain security analysis.

SHAP Explainability Analysis

To improve model interpretability, SHAP (SHapley Additive Explanations) was implemented to explain how individual Solidity features contribute to vulnerability prediction. The SHAP summary plot illustrates the influence of each feature on the classification output.

The results (figure 5) show that features such as staticcall, success, and gas have strong impacts on the model predictions. These features are commonly associated with low-level external interactions and Ethereum Virtual Machine (EVM) operations, which are often related to security vulnerabilities such as reentrancy attacks and unsafe external calls.

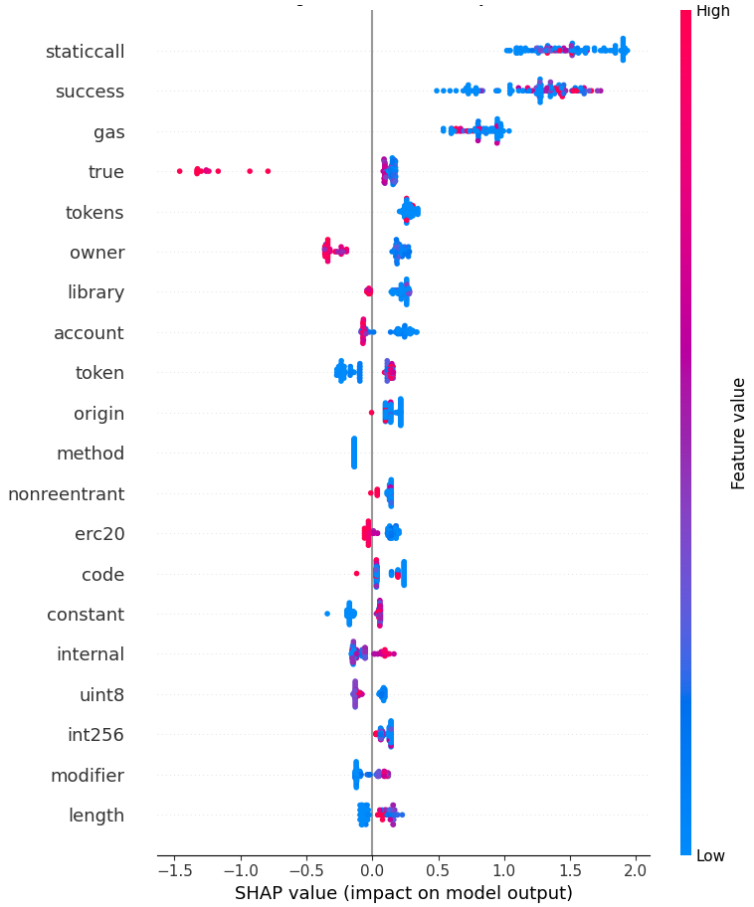

Figure 5 SHAP Summary Plot

The feature origin also contributes significantly to vulnerability prediction. This feature is related to the use of tx. origin, which is widely recognized as an insecure access control mechanism in Solidity smart contracts. Furthermore, security-related features such as nonreentrant and reentrancy guard were identified by the model, indicating that the proposed framework can recognize both vulnerable coding patterns and security mitigation mechanisms.

Feature Importance Analysis

Feature importance analysis was conducted to identify the most influential features contributing to smart contract vulnerability detection. The XGBoost model generated importance scores based on the contribution of each feature during the classification process. The ten most important features identified by the model are presented in table 9, while their relative contributions are visualized in figure 6.

Table 9 Top Important Features

Feature	Importance
success	0.120
tokens	0.087
gas	0.055
let	0.046
reentrancyguard	0.040
true	0.040
32	0.031

data	0.028
mul	0.022
nonreentrant	0.022

As shown in table 9, the feature success achieved the highest importance score of 0.120, indicating that low-level external call validation patterns play a significant role in vulnerability prediction. The features tokens and gas ranked second and third, with importance scores of 0.087 and 0.055, respectively. These results suggest that token-related operations and gas management patterns provide valuable information for distinguishing between vulnerable and safe smart contracts.

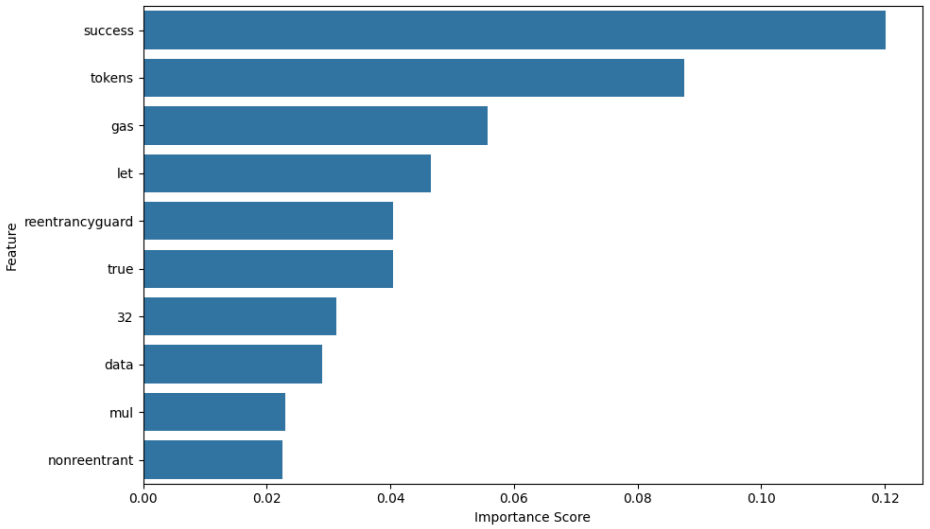

Figure 6 Top 10 Feature Importance

The feature importance distribution illustrated in **Figure 6** further confirms the dominance of *success*, *tokens*, and *gas* in the classification process. Several additional features, including *let*, *true*, *data*, and *mul*, also contributed to the model's predictive capability, indicating that the framework captures a diverse set of lexical and programming-related patterns from Solidity source code.

Notably, the features *reentrancyguard* and *nonreentrant* achieved importance scores of 0.040 and 0.022, respectively. These terms are commonly associated with security mechanisms designed to mitigate reentrancy attacks. Their presence among the most influential features indicates that the proposed model is capable of recognizing not only vulnerable coding patterns but also defensive programming practices implemented within smart contracts.

Overall, the results presented in table 9 and figure 6 demonstrate that the proposed explainable machine learning framework successfully identifies security-relevant features embedded in Solidity source code. The feature importance analysis confirms that the model relies on meaningful vulnerability-related indicators, thereby supporting both the predictive performance and interpretability of the proposed smart contract vulnerability detection framework.

ROC Curve Analysis

Receiver Operating Characteristic (ROC) analysis was performed to evaluate the discrimination capability of the proposed machine learning model. The

proposed framework achieved an AUC score of 0.9946 (figure 7), which indicates excellent classification capability between vulnerable and safe smart contracts. The ROC curve demonstrates that the model maintains a high true positive rate while minimizing false positive predictions. This result confirms the effectiveness and stability of the proposed approach in vulnerability detection tasks.

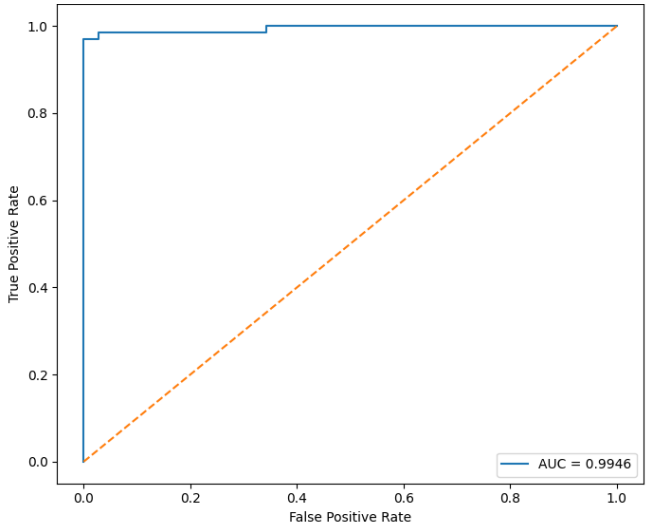

Figure 7 ROC Curve

SHAP Feature Importance Interpretation

Global feature interpretation was further analyzed using SHAP feature importance visualization. The resulting SHAP feature importance visualization is presented in figure 8. As shown in figure 8, *staticcall* has the highest mean absolute SHAP value, followed by *success* and *gas*. This indicates that these features have the greatest overall influence on the model output among the evaluated Solidity tokens. In particular, *staticcall* represents a mechanism for performing read-only external calls, while *success* and *gas* are closely related to the execution and validation of external contract interactions. Their high SHAP contributions suggest that patterns associated with low-level contract execution provide important information for distinguishing vulnerable and safe smart contracts.

Other features, including *true*, *tokens*, *owner*, *library*, *account*, *token*, and *origin*, also contribute to the model's predictions, although their overall influence is lower than that of the three leading features. The presence of *origin* among the influential features is particularly relevant because the tx.origin pattern can be associated with access-control weaknesses in Solidity contracts.

The results illustrated in figure 8 demonstrate that SHAP provides a global perspective on feature influence by measuring the average magnitude of each feature's contribution to the model output. This analysis complements the conventional XGBoost feature importance analysis by providing an alternative interpretation of which source-code patterns most strongly affect vulnerability predictions. Overall, the explainability analysis indicates that the proposed framework not only achieves high classification performance but also provides interpretable evidence regarding the Solidity features that influence its decision-

making process.

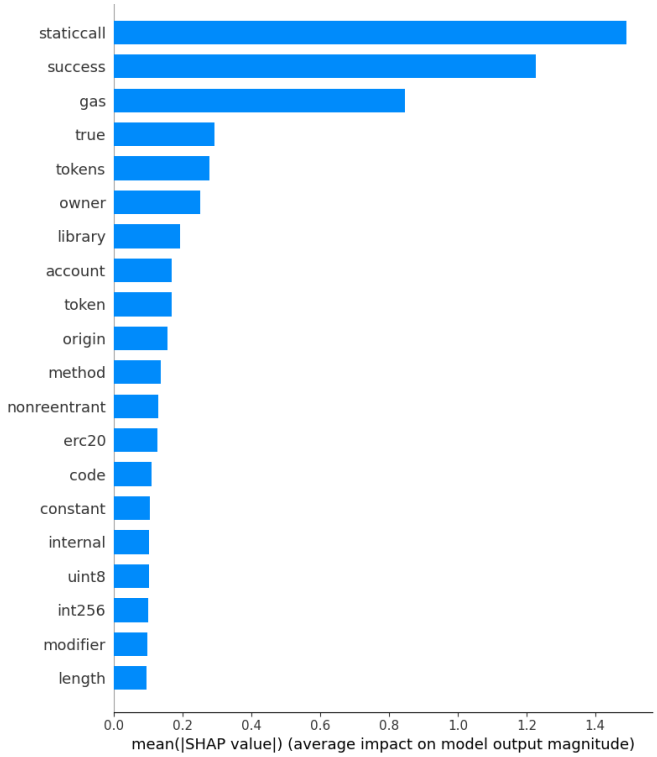

Figure 8 SHAP Feature Importance

Vulnerability Interpretation Analysis

To further analyze the security implications of the extracted features, vulnerability interpretation analysis was conducted. The security interpretation of the selected features is presented in table 10.

Table 10 Vulnerability Interpretation

Feature	Security Interpretation
tx.origin	Risky access control mechanism
delegatecall	Unsafe external interaction
nonReentrant	Reentrancy mitigation mechanism
success	Low-level external call handling
call.value	Potential reentrancy vulnerability

As shown in table 10, the extracted features represent both potentially vulnerable coding patterns and security mitigation mechanisms. The tx.origin feature is associated with potentially unsafe access-control practices, while delegatecall represents an external interaction mechanism that can introduce security risks when improperly implemented. Similarly, call.value is commonly associated with potential reentrancy vulnerabilities because it can transfer Ether while invoking external contract functionality.

In contrast, nonReentrant represents a defensive programming mechanism designed to mitigate reentrancy attacks. The presence of success reflects the handling of low-level external call outcomes, which can provide important information for identifying potentially unsafe contract execution patterns. These findings indicate that the model captures both vulnerability-related patterns and security-oriented programming constructs rather than relying solely on individual

vulnerability keywords.

Overall, the results presented in Table 10 demonstrate that the proposed Explainable Machine Learning framework can associate influential Solidity features with meaningful security characteristics. Combined with the feature importance and SHAP analyses presented previously, these findings support the interpretability and transparency of the proposed framework for smart contract vulnerability detection and blockchain security analysis.

Discussion

The experimental results show that the proposed Explainable Machine Learning framework successfully detected vulnerabilities in Ethereum smart contracts with high classification performance [7], [8], [13], [15]. The model achieved an accuracy of 97.84%, indicating that XGBoost combined with TF-IDF feature extraction is effective for Solidity source code classification. The confusion matrix analysis demonstrated that only three contracts were misclassified during testing. The model correctly classified 68 safe contracts and 68 vulnerable contracts, which indicates strong detection capability and low classification error.

The ROC analysis produced an AUC score of 0.9946, showing that the proposed model has excellent capability in distinguishing vulnerable and safe smart contracts [9], [22], [23]. This result confirms that the framework maintains stable classification performance across different prediction thresholds. The SHAP explainability analysis provided important insights into the decision-making process of the model. Features such as *success*, *staticcall*, *gas*, and *origin* significantly influenced vulnerability prediction. These features are closely related to low-level external interactions and access control mechanisms that frequently appear in vulnerable Solidity contracts [5], [7], [15], [18].

The feature *success* obtained the highest importance score of 0.120094, indicating that external call validation patterns strongly contribute to vulnerability detection. In addition, features such as *reentrancy guard* and *nonReentrant* were also identified as important indicators, demonstrating that the model successfully recognizes security mitigation mechanisms against reentrancy attacks [15], [16], [18]. The integration of SHAP improved the interpretability of the proposed framework by providing transparent explanations regarding feature contributions. This explainability capability is important in blockchain security analysis because security auditors require interpretable systems to understand why a smart contract is classified as vulnerable [10], [24]–[26].

Conclusion

This study proposed an Explainable Machine Learning framework for smart contract vulnerability detection in Ethereum blockchain using Solidity source code analysis. The proposed framework integrated TF-IDF feature extraction, XGBoost classification, and SHAP explainability analysis to improve both detection performance and model interpretability.

The experimental results demonstrated that the proposed model achieved an accuracy of 97.84%, with precision, recall, and F1-score values exceeding 97%. The confusion matrix analysis showed that the model successfully classified most vulnerable and safe smart contracts with only a small number of

misclassifications. Furthermore, the ROC analysis produced an AUC score of 0.9946, indicating excellent classification capability.

The SHAP explainability analysis revealed that features related to low-level external interactions, gas handling, and access control mechanisms significantly influenced vulnerability prediction. Features such as success, staticcall, gas, and origin were identified as the most influential indicators during the detection process. The framework also successfully recognized security mitigation mechanisms such as nonreentrant and reentrancyguard.

The results indicate that the proposed Explainable Machine Learning framework not only achieves high vulnerability detection accuracy but also provides transparent and interpretable explanations for blockchain security analysis. The integration of Explainable AI improves the trustworthiness of machine learning models and supports security auditors in understanding vulnerability-related patterns within Solidity smart contracts.

Future research may focus on implementing multiclass vulnerability classification, integrating deep learning approaches, and expanding the dataset using real-world blockchain vulnerability repositories to further improve detection capability and generalization performance.

Declarations

Author Contributions

Conceptualization: V.S.R.; Methodology: V.S.R.; Software: V.S.R.; Validation: V.S.R.; Formal Analysis: V.S.R.; Investigation: V.S.R.; Resources: V.S.R.; Data Curation: V.S.R.; Writing Original Draft Preparation: V.S.R.; Writing Review and Editing: V.S.R.; Visualization: V.S.R.; The author has 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] M. Singh and S. Kim, “Blockchain technology for decentralized autonomous organizations,” *Advances in Computers*, vol. 115, no. 2019, pp. 115–140, 2019,

doi: 10.1016/bs.adcom.2019.06.001.

[2] S. Aufiero *et al.*, "DApps ecosystems: Mapping the network structure of smart contract interactions," *EPJ Data Sci.*, vol. 13, no. 1, pp. 1–31, Sep. 2024, doi: 10.1140/epjds/s13688-024-00497-8.

[3] S. K. Dutta, "Smart contracts," *The Definitive Guide to Blockchain for Accounting and Business: Understanding the Revolutionary Technology*, vol. 2020, no. Sep., pp. 61–78, Sep. 2020, doi: 10.1108/978-1-78973-865-020201005.

[4] G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali, and R. Hierons, "Smart contracts vulnerabilities: A call for blockchain software engineering?" *IWBOSE*, vol. 2018, no. Mar., pp. 19–25, Mar. 2018, doi: 10.1109/IWBOSE.2018.8327567.

[5] S. M. Nzuva, "Revisiting blockchain technologies and smart contracts security: A pragmatic exploration of vulnerabilities, threats, and challenges," *Asian J. Res. Comput. Sci.*, vol. 17, no. 7, pp. 11–30, Jun. 2024, doi: 10.9734/ajrcos/2024/v17i7474.

[6] C.-C. Tsai, C.-C. Lin, and S.-W. Liao, "Unveiling vulnerabilities in DAO: A comprehensive security analysis and protective framework," *Blockchain*, vol. 2023, no. Nov., pp. 151–158, Nov. 2023, doi: 10.1109/Blockchain60715.2023.00034.

[7] D. He, R. Wu, X. Li, S. Chan, and M. Guizani, "Detection of vulnerabilities of blockchain smart contracts," *IEEE Internet Things J.*, vol. 10, no. 14, pp. 12178–12185, Jul. 2023, doi: 10.1109/JIOT.2023.3241544.

[8] A. Shewale, D. Thallapalli, and S. Udhayakumar, "Enhancing smart contract security: A comprehensive vulnerability assessment framework," *ICCDS*, vol. 2025, no. 2025, pp. 1–6, 2025, doi: 10.1109/ICCDS64403.2025.11209716.

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

[10] N. Khan, M. Nauman, A. S. Almadhor, N. Akhtar, A. Alghuried, and A. Alhudhaif, "Guaranteeing correctness in black-box machine learning: A fusion of explainable AI and formal methods for healthcare decision-making," *IEEE Access*, vol. 12, no. 2024, pp. 90299–90316, 2024, doi: 10.1109/ACCESS.2024.3420415.

[11] Z. Zhang, H. A. Hamadi, E. Damiani, C. Y. Yeun, and F. Taher, "Explainable artificial intelligence applications in cyber security: State-of-the-art in research," *IEEE Access*, vol. 10, no. 2022, pp. 93104–93139, 2022, doi: 10.1109/ACCESS.2022.3204051.

[12] S. Dong, K. Abbas, M. Li, and J. Kamruzzaman, "Blockchain technology and application: An overview," *PeerJ Comput. Sci.*, vol. 9, no. Nov., pp. 1–20, Nov. 2023, doi: 10.7717/peerj-cs.1705.

[13] T. M. Hewa, Y. Hu, M. Liyanage, S. S. Kanhare, and M. Ylianttila, "Survey on blockchain-based smart contracts: Technical aspects and future research," *IEEE Access*, vol. 9, no. 2021, pp. 87643–87662, 2021, doi: 10.1109/ACCESS.2021.3068178.

[14] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, "Blockchain-enabled smart contracts: Architecture, applications, and future trends," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 49, no. 11, pp. 2266–2277, Nov. 2019, doi: 10.1109/TSMC.2019.2895123.

[15] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart contract: Attacks and protections," *IEEE Access*, vol. 8, no. 2020, pp. 24416–24427, 2020, doi: 10.1109/ACCESS.2020.2970495.

[16] S. Yang, J. Chen, M. Huang, Z. Zheng, and Y. Huang, "Uncover the premeditated attacks: Detecting exploitable reentrancy vulnerabilities by identifying attacker contracts," *ICSE*, vol. 2024, no. Apr., pp. 1–12, Apr. 2024, doi: 10.1145/3597503.3639153.

[17] I. Butun, P. Österberg, and H. Song, "Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 616–644, Jan. 2020, doi: 10.1109/COMST.2019.2953364.

[18] T. Dey, R. K. Lenka, S. Senapati, D. P. Mishra, and S. R. Mallick, "Reentrancy vulnerability in the blockchain ecosystem: Historical attacks, detection approaches, and mitigation strategies," *NETCRYPT*, vol. 2025, no. 2025, pp. 1307–1311, 2025, doi: 10.1109/NETCRYPT65877.2025.11102145.

[19] S. T. Gandhi, "AI-driven smart contract security: A deep learning approach to vulnerability detection," *Int. J. Adv. Res. Comput. Sci. Technol.*, vol. 8, no. 1, pp. 1–8, Feb. 2025, doi: 10.15662/ijarcst.2025.0801004.

[20] H. Chen and M. A. Babar, "Security for machine learning-based software systems: A survey of threats, practices, and challenges," *ACM Comput. Surveys*, vol. 56, no. 6, pp. 1–38, Feb. 2024, doi: 10.1145/3638531.

[21] G. M. Ali and H. Chen, "Contract-guardian: A bagging-based gradient boosting decision tree for detection vulnerability in smart contract," *Cluster Comput.*, vol. 28, no. 8, pp. 1–15, Aug. 2025, doi: 10.1007/s10586-025-05230-2.

[22] F. Wu, J. Wang, J. Liu, and W. Wang, "Vulnerability detection with deep learning," *ICCC*, vol. 2017, no. Nov., pp. 1298–1302, Nov. 2017, doi: 10.1109/CompComm.2017.8322752.

[23] V. K. Kasula, A. R. Yadulla, M. Yenugula, B. Konda, and A. Alshboul, "Enhancing vulnerability detection in smart contracts using transformer-based embeddings and graph neural networks," *ICCTA*, vol. 2024, no. 2024, pp. 177–182, 2024, doi: 10.1109/ICCTA64612.2024.10974898.

[24] V. Hassija *et al.*, "Interpreting black-box models: A review on explainable artificial intelligence," *Cogn. Comput.*, vol. 16, no. 1, pp. 45–74, Aug. 2023, doi: 10.1007/s12559-023-10179-8.

[25] K. Kalasampath, K. N. Spoorthi, S. Sajeev, S. S. Kuppa, K. Ajay, and A. Maruthamuthu, "A literature review on applications of explainable artificial intelligence (XAI)," *IEEE Access*, vol. 13, no. 2025, pp. 41111–41140, 2025, doi: 10.1109/ACCESS.2025.3546681.

[26] F. M. Khan *et al.*, "XAI-driven data mining for self-defending IoT systems: Enhancing cybersecurity transparency in the age of smart cities," *Cogn. Comput.*, vol. 18, no. 1, pp. 1–15, Feb. 2026, doi: 10.1007/s12559-026-10559-w.

[27] K. Miao *et al.*, "Exploring explainable machine learning and Shapley additive explanations (SHAP) technique to uncover key factors of HNSC cancer: An analysis of the best practices," *Biomed. Signal Process. Control*, vol. 89, no. Mar., pp. 1–15, Mar. 2024, doi: 10.1016/j.bspc.2023.105752.

[28] Y. Liu, Y. Fu, Y. Peng, and J. Ming, "Clinical decision support tool for breast cancer recurrence prediction using SHAP value in cooperative game theory," *Heliyon*, vol. 10, no. 2, pp. 1–15, Jan. 2024, doi: 10.1016/j.heliyon.2024.e24876.

[29] A. A. Chandi, "Explainable artificial intelligence applications in cybersecurity: Enhancing transparency in intrusion detection systems," *Int. J. Appl. Math.*, vol. 38, no. 11s, pp. 1239–1253, Nov. 2025, doi: 10.12732/ijam.v38i11s.1244.