Verification of smart contracts using Machine Learning

Manchala Beula Grace[1]

[1]*Student, Department of Forensic Science, Jain (Deemed to be university)*

Abstract--- **This paper offers a machine learning approach based on behavioral analysis to identify Ethereum smart contract vulnerabilities and fraud. The Random Forest model performed better than existing tools and other ML models, with accuracy of more than 90% and high interpretability. Features such as Ether value and number of transactions were significant in the identification of risks. This method advances blockchain trust and security, with added benefits for developers, auditors, and investors.**

I. INTRODUCTION

This research strives to alleviate Ethereum smart contract vulnerabilities through the creation of an automated vulnerability detection system through the use of machine learning. With increasing use of blockchain, threat of fraud and logic-based exploits grows, emphasizing the need for sophisticated security tools. Current tools are unable to detect real-time, intricate vulnerabilities, particularly in individual contracts. This research suggests a hybrid model that incorporates static, dynamic analysis and behavior information for a holistic security check. Employing machine learning algorithms such as Random Forest and Logistic Regression, the system identifies anomalies and vulnerabilities in Ethereum transactions. The method is geared towards real-time fraud detection, which current research lacks. Ethereum is selected because of its flagship nature and openness of data that enable reproducible and scalable ML modeling. There are three steps to the methodology: vulnerability detection, fraud detection, and pattern identification. Feature importance and visual analytics were used to improve model interpretability and accuracy. Results indicate that the models successfully mark security threats, promoting blockchain safety and trust.

II. LITERATURE REVIEW

Kumar, R., & Singh, V. et al. (2024) paper targets the application of machine learning (ML) methods to identify vulnerabilities and fraudulent activity in Ethereum smart contracts. Due to the increasing adoption of blockchain, the authors point out that the concern about the security of smart contracts is growing, particularly on platforms such as Ethereum, which are predominantly used for decentralized applications. The study proposes a machine learning-based model to improve the process of identifying vulnerabilities and fraudulent activity in Ethereum smart contracts. They use various features like code analysis and transactional patterns to train their model. By using supervised and unsupervised learning techniques, the authors were able to effectively identify risks and fraudulent activity that could be exploited in the system.

Li, D., Wong, W. E., Wang, X., Pan, S., & Koh, L.-S. et al. (2024) Developed a smart and efficient method to detect security issues in Ethereum smart contracts. Instead of running the contracts to find bugs, their system analyzes the code directly (a method called static analysis). What makes their approach stand out is that they use an advanced search technique—sort of like a smart detective—that looks at multiple aspects of the code at once to spot possible vulnerabilities. The researchers experimented with it and found that it worked better than most existing tools—finding more issues with fewer false positives. Overall, their work offers a more reliable way of making smart contracts secure and safe before they are released into real use.

Chen, D., Ji, S., Dong, J., & Wei, Z. et al.(2023). Provide a thorough analysis of the vulnerabilities of smart contracts, touching on their causality, detection, and remediation. The research begins by identifying the sources of information related to the vulnerabilities, ranging from publicly accessible repositories to known exploits. The discussion further goes on to describe some major detection methods, such as static and dynamic analysis, formal verification, and machine learning-based detection. Every approach is compared based on strengths, weaknesses, and appropriateness to varying environments. The authors also delve into recent studies of automated repair systems to plug vulnerabilities. The paper shows the most common vulnerability patterns in Ethereum smart contracts and how they are being addressed. By scanning through a huge range of tools and research, the paper is a valuable read for both developers and researchers. Overall, it provides a balanced summary of the smart contract security landscape.

Kushwaha, S. S., et al. (2022). Conducted a systematic review focused on security vulnerabilities inherent in Ethereum smart contracts. They compared a wide range of research studies to identify common vulnerabilities and categorize various types of vulnerabilities. Notable points of issues covered include reentrancy attacks, integer overflows, and gas limit-related problems. The manuscript also compares the efficacy of conventional and AI-based detection tools and methods. Through a comparison of various strategies, the authors determine the most reliable methods in preventing security vulnerabilities. They emphasize the importance of using secure coding practices combined with ongoing auditing procedures. In addition, their review outlines areas of the research body lacking as well as areas needing research. Overall, the paper is a definitive guide to improving the security of smart contracts in the Ethereum platform.

Pise, R., & Patil, S. D. et al.(2022, September). Explore the particular security concerns of blockchain-smart contracts with a focus on the Ethereum system. The article categorizes usual vulnerabilities such as reentrancy, gas limit issues, and unhandled exceptions. They enumerate how the vulnerabilities can be abused and highlight the shortcomings of common security solutions when applied to blockchains. The study refers to actual attacks in the real world, the DAO hack, and explains the ramifications of insecure smart contracts. The study also breaks down detection mechanisms and recommends code best practices in writing more secure smart contract code. The authors stress the requirement for expert test frameworks that specifically cater to the blockchain environment. Their study reminds developers to follow a more security-aware development cycle. Overall, the paper is a comprehensive handbook to learning about and avoiding unique smart contract-specific threats.

Rameder, H., di Angelo, M., & Salzer, G et al.(2022) Paper provides the state of the art of automated tools and techniques used to detect vulnerabilities in Ethereum smart contracts. The authors review a range of static, dynamic, and formal analysis tools and determine their strengths, weaknesses, and usability. They observe that while most tools show good performance in detecting known categories of vulnerabilities, there is still a gap in detecting newer or more advanced flaws. The review also discusses how different tools attain detection accuracy in relation to scalability and performance efficiency. The authors provide comprehensive comparisons and benchmarks to help researchers and developers choose the right tools. They conclude the need for using multiple analytical methods to get better results. The paper also provides trends in tool development and determines future research needs. Altogether, it is a good reference to further enhance the security of smart contracts using automation.

Lutz, O., Chen, H., Fereidooni, H., Sendner, C., Dmitrienko, A., Sadeghi, A. R., & Koushanfar, F. et al.(2021). Introduce ESCORT, a deep learning-powered vulnerability detection framework that uses transfer learning to detect Ethereum smart contract vulnerabilities. The overall goal of the framework is to automate and improve the detection of security vulnerabilities that are commonly missed by conventional detection techniques. ESCORT is trained on a vast corpus of smart contracts so that it can recognize patterns typical of potential vulnerabilities. Additionally, it also uses transfer learning so that the system can transfer knowledge acquired from one corpus to new, unseen contracts. The framework achieves better performance in the detection of severe vulnerabilities like reentrancy and integer-related bugs. Compared to conventional static analysis tools, ESCORT offers better accuracy and lower false positive rates. The authors argue that the integration of deep learning has the ability to significantly enhance the security of smart contracts when combined with valid data. Overall, ESCORT is a significant AI-driven step toward improving the security of blockchain applications.

Amiet , N. et al. (2021)The academic paper, illustrates the security weaknesses inherent in blockchain technology as they manifest in actual use, with emphasis on the explanation of such security weaknesses in actual use. The research identifies some common vulnerabilities which affect blockchain systems, particularly in the case of smart contracts The paper discusses various vulnerabilities, including reentrancy attacks, gas limits, and inadequate access controls, which are susceptible to attacks by malicious entities. Additionally, the paper cites concrete instances of how these vulnerabilities have been exploited in mass cyberattacks that resulted in large monetary losses. It describes a series of methods and tools that can be utilized in order to detect and address these inadequacies.The author also points to the need for continuous monitoring and security audits as a means to avoid future hacks. Overall, the paper is a working guide for developers and researchers who are interested in enhancing the security of blockchain systems. The paper asserts the urgent need for having robust security measures in the fast-developing blockchain world.

Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. et al.(2020). Provide a thorough survey of the security of blockchain systems looking at the basic vulnerabilities and attacks, as well as the risks associated with different blockchain architectures. They highlight pertinent security threats that developers may encounter when developing a blockchain system for an organization, including attacks on consensus protocols, vulnerabilities in smart contracts, privacy risks in blockchain networks, etc. Li et al. have detailed existing defensive mechanisms to prevent or work through a range of security threats and outlined how they help mitigate these risks whilst noting limitations. They also propose possible future solutions for securing blockchains ranging from cryptographic mechanisms to machine-learning approaches.They advocate for a layered security model and identify some cybersecurity areas that can be improved from a governance and technological standpoint. The work provides an excellent context for developers, researchers and practitioners to reflect on how to secure their blockchain platforms. They also identify some possible directions for future research to enhance blockchain security and to harden the technology against attacks. In general, the work outlines a good deal of the security landscape used within blockchain systems.

Ye, J., Ma, M., Lin, Y., Ma, L., Xue, Y., & Zhao, J. et al.(2019). In the work, Ye and co-authors introduce Vulpedia-a framework for detecting vulnerabilities in Ethereum smart contracts, leveraging abstracted vulnerability signatures. The framework distinguishes between non-systematic vulnerabilities of smart contracts by abstracting Ethereum bytecode into simpler programs that are tractable in behavior and structure. Vulpedia relates these simpler programs to known vulnerability signatures (patterns) and helps detect generic vulnerabilities such as susceptibility to contract reentrance attacks, integer overflows, and a host of other critical and non-critical bugs. The authors demonstrate that Vulpedia can detect smart contract vulnerabilities at a cost that is both efficient and scalable enough to analyze many smart contracts simultaneously. They compared Vulpedia's performance with existing smart contract analysis tools and illustrated the superior detection accuracy of Vulpedia, compared to many of the existing tools. The authors emphasize the rationale behind using abstracted signatures to improve the detectability of the vulnerability and the cost of discoverability accounting. This work plays a pivotal role in the ongoing process of improving the security of Ethereum smart contracts by providing an automated and scalable solution to vulnerability detection, thus providing a jump forward in the field of blockchain security.

Jiang, B., Liu, Y., & Chan, W. K. et al.(2018). In the paper, Jiang, Liu, and Chan detail ContractFuzzer, a tool for fuzz-testing that aims to discover security vulnerabilities for Ethereum smart contracts. The ContractFuzzer tool operates by generating random inputs to test smart contract functions, and can potentially identify unexpected behavior, or, perhaps, security vulnerabilities. The authors demonstrate how using fuzz testing can identify security vulnerabilities such as a reentrancy vulnerability, or improper access control limits to smart contracts that likely would not be discovered using traditional testing methods. ContractFuzzer is unique compared to other tools suggested in the literature because of its high degree of automation when it tests contracts without deep examination and background knowledge of the contract prior. The paper compares ContractFuzzer's effectiveness too other vulnerability detection tools - ContractFuzzer is able to find vulnerabilities that are both efficient and effective too many of the related tools in the literature. The paper also discusses how fuzzing could be integrated into the smart contract development lifecycle. A completely automated vulnerability detection tool like ContractFuzzer represents an enhanced security check before deploying any smart contract, thus is a significant step towards providing a higher register of security of future blockchain applications.

Atzei, N., Bartoletti, M., & Cimoli, T. et al.(2017)Atzei and colleagues offer one of the first complete surveys of known attacks on Ethereum smart contracts. They group vulnerabilities into categories, including reentrancy, timestamp dependency, gas limit, and unchecked external calls. The authors are clear that design flaws and programming bugs in smart contracts can be exploited and lead to devastating financial losses. They examine actual cases—such as the attack on The DAO—where vulnerabilities have been exploited. They also examine how vulnerabilities relate to various points in the Ethereum execution model. They review potential protections and stress the importance of formal verification. This paper is a starting place for anyone looking to understand the risks of smart contracts and ways to mitigate them. Overall the work is important for developers and researchers that want to create secure decentralized applications.

III. OBJECTIVES

Objective 1 The goal of this project is to leverage automated tools and analysis techniques for discovering vulnerabilities in smart contracts. The key focus is to expose security issues such as reentrancy attacks, integer overflows, and logical flaws prior to deployment to reduce possible exploits.

Objective 2: identifying fraud in Ethereum smart contracts through analyzing contract behavior and transaction patterns.

Objective 3: fraud detection in Ethereum smart contracts using transaction patterns and contract behavior analysis.

Methods used to gather all data, analysis, plots and tables:

In this study of smart contract verification, a systematic and data-driven approach was used to collect, process, analyze, and report findings. Data was obtained mainly from publicly accessible Ethereum smart contract transaction datasets, in the form of CSV files, with important attributes like transaction amounts, timestamps, addresses, and contract IDs. The first step entailed intensive preprocessing of data in which columns that were irrelevant or redundant were dropped, missing values were addressed—most often replaced with zeros—and categorical or low-variance features were omitted to improve data quality. Feature engineering was then conducted, in which transformation and variable selection of significant variables was undertaken, including time differences between transactions and overall Ether transferred. In order to maintain well-balanced training data, particularly for fraud detection, random oversampling strategies were utilized using the imbalanced- learn library. Machine learning algorithms, Logistic Regression and Random Forest Classifier, were implemented utilizing Python's scikit-learn environment, dividing the dataset into a training (80%) and a testing (20%) set. These algorithms were assessed by utilizing metrics such as confusion matrices, classification reports (precision, recall, F1-score), and ROC-AUC scores. Visualization was instrumental in result interpretation—matplotlib and seaborn libraries were employed to create boxplots, heatmaps, and bar charts, providing insights into behavioral trends and feature importances. Performance metrics and variable rankings were also tabulated using pandas, aiding in a clear and interpretable presentation of the research results. This holistic approach guaranteed strong analysis and meaningful interpretation of smart contract vulnerabilities and fraud trends.

The testing strategy implemented in this study was aimed at systematic assessment of the proposed smart contract verification and fraud detection model effectiveness and reliability. The strategy targeted three major elements: model performance verification, analysis of feature importance, and behavior pattern visualization.

To start with, the dataset was divided into training set and test set (80/20 split) so that models were tested against unseen data as in real scenarios. Two machine learning models—Random Forest Classifier and Logistic Regression—were tested and trained in order to contrast predictive performance. The performance of the models was evaluated based on a suite of strong evaluation metrics, such as precision, recall, F1-score, and ROC-AUC score, that together quantified the models. Capacity to detect vulnerabilities and malicious activity in Ethereum smart contracts correctly. To further substantiate the models, confusion matrices were created to graphically represent the distribution of true positives, true negatives, false positives, and false negatives. These matrices gave clear information about each model's classification trend and pointed out differences in sensitivity and specificity. Feature importance analysis, especially using the Random Forest model, was also performed to determine which variables contributed most towards predictions. This built in an interpretability layer, which is necessary for grasping and believing machine learning decisions within a security scenario.

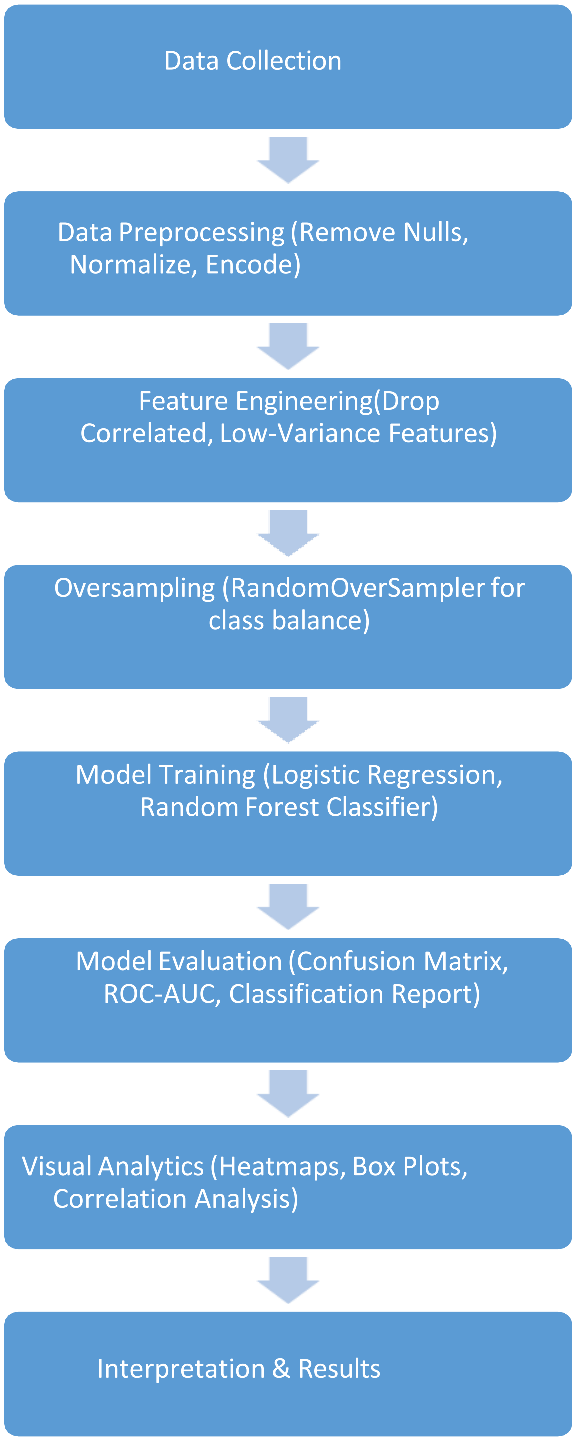
The approach also entailed comprehensive visualization through box plots, heat maps, and bar charts to illustrate differences in behavior between fraudulent and non-fraudulent transactions. These visual aids contextualized model outputs and facilitated effective communication of results during presentations or audits.

Design

The research is structured into three central pillars: vulnerability identification, fraud detection, and Ethereum smart contract behavior analysis. Each pillar is guided by a unique objective, methodological process, and evaluation methodology. The research combines data-driven machine learning processes, feature design, and visualization analysis to detect threats in decentralized smart data preparation, model building, experimentation, and interpretive analysis contracts.

The initial proof of concept of the research was established via comparative model testing against Ethereum smart contract dataset. Logistic Regression and Random Forest were implemented on:

Random Forest gave more accurate results and recall compared to Logistic Regression with 1530 true positives correctly detected. Random Forest once more outperformed, recognizing temporal and transactional anomalies which predicted fraud, outcomes validated with feature importance analysis. These experiments confirmed that a combination of classical ML with strong preprocessing was capable of accurately classifying malicious behavior in block chain environments.


Data Collection
Data Preprocessing (Remove Nulls, Normalize, Encode)
Feature Engineering(Drop Correlated, Low-Variance Features)
Oversampling (RandomOverSampler for class balance)
Model Training (Logistic Regression, Random Forest Classifier)
Model Evaluation (Confusion Matrix, ROC-AUC, Classification Report)
Visual Analytics (Heatmaps, Box Plots, Correlation Analysis)
Interpretation & Results
Figure 1: Step-by-Step Study Conduction Procedure Initial Prototype Tests (Proof of Concept)

IV. METHODOLOGY

Methodology 1

The main aim of this research was to identify vulnerabilities in Ethereum smart contracts by using a mixture of analytical and machine learning. The analysis process started with the preparation and cleansing of data, an important task to maintain the reliability of the analysis. Using Python’s powerful data manipulation library, pandas, the dataset—comprising Ethereum smart contract transactions—was imported and examined. To improve data quality and remove redundancy, unwanted or redundant columns like Unnamed: 0, Address, and Index were deleted. This was followed by a detailed missing value check. In most cases, missing values found in numeric columns were filled with zeros to preserve the structure and integrity of the data set. Additionally, categorical features and zero-variance features were excluded because they did not contribute significantly to model learning and they injected noise.

The second step was feature engineering, where features with low variance or little information gain e.g., minimum value sent to a contract were removed. Features with high correlation, such as average and max token values sent or received, were also removed to prevent multicollinearity, which can negatively impact model interpretability and performance. The last dataset contained only significant, independent numerical predictors required for successful model training.

To address the class imbalance in the target variable, which had been used earlier to mark the presence or absence of a smart contract vulnerability (FLAG), random oversampling was used. This ensured proper balancing of the two classes and thus avoided biased learning.

Furthermore, the characteristics were normalized by using the Power Transformer technique in order to reduce skewness and approximate a normal distribution, something particularly valuable with models that are sensitive to the scaling of the features, e.g., logistic regression.

Two machine learning models were trained to identify contract vulnerabilities: A Logistic Regression model, tuned with a regularization parameter set to C=0.1, and a Random Forest Classifier with 50 decision trees. Both the models were trained on 80% of the preprocessed dataset and were validated on the remaining 20%. The model performance was assessed with various metrics: the classification report (measuring precision, recall, and F1-score), the ROC-AUC score (measuring the models' capability to distinguish between vulnerable and non-vulnerable contracts), and the confusion matrix, which visually illustrated true and false positives and negatives. This complete methodology offered a productive and understandable assessment of smart contract exposures and welcomed developments for improved security in blockchain systems.

Methodology 2

The second aim of the current research was to detect fraudulent transactions in Ethereum smart contracts employing a supervised machine learning technique. The same pre-cleaned dataset employed in the first aim was preserved since it recorded fundamental transactional behavior patterns critical in detecting fraud. In the preprocessing, the features (X) were isolated from the target variable (y), which was the indication that a transaction is fraudulent. In order to counter the usual problem of class imbalance common in the case of fraud detection problems random oversampling was used through Random Over Sampler technique.

Following preprocessing, two models were trained on the classification task: a Logistic Regression model, configured with a higher iteration limit to facilitate convergence, and a Random Forest Classifier consisting of 50 decision trees. Feature normalization was performed prior to model training to normalize input values, a critical process particularly for the logistic regression model that employs gradient descent optimization. Models were compared on a combination of performance metrics like classification reports (precision, recall, and F1-score), ROC-AUC scores (to establish discriminatory capability), and confusion matrices (to compare prediction accuracy). The Random Forest model was used to identify and rank the top 10 and top 15 features that played the biggest role in predicting fraud. For comparison, the coefficients from the logistic regression model were also reviewed. Both models showed similar patterns in terms of which features were most influential, which helped strengthen the reliability of the findings. To facilitate visual comprehension, heat maps were employed to represent the confusion matrices, with bar charts graphically representing the most significant predictors. The balanced approach, in addition to resulting in correct fraud classification, also resulted in a deeper understanding of the patterns of behavior involved in fraudulent behavior in smart contract platforms.

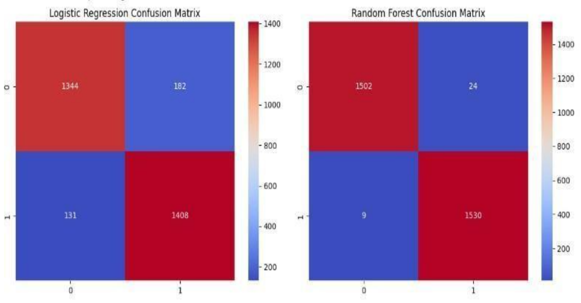
Methodology 3

The third step of the research entailed the exploration of Ethereum transaction behavior in order to uncover patterns commonly observed in malicious activity. The step began with the importation of essential Python libraries, such as pandas for data manipulation, matplotlib. pyplot and seaborn for plotting, and MinMaxScaler from sklearn. preprocessing for feature scaling. The transaction information, in the form of a CSV, was read into Python using the read_csv () method. For initial inspection of the dataset, primitive functions such as head () and describe () were employed to witness its form as well as the nature of critical measures.

The subsequent step was conducting a data quality check to ascertain missing values, if any, and to ensure proper data types. In the process of data cleaning, columns like "Unnamed: 0" and "Index" were dropped, which were unnecessary, and missing values were replaced with zeroes assuming that these were unrecorded transactions. The transaction information, in the form of a CSV, was read into Python using the read_csv() method. For initial inspection of the dataset, primitive functions such as head() and describe() were employed to witness its form as well as the nature of critical measures.

V. RESULT AND DISCUSSION

Objective-1



The Figure 2 represents the comparison of confusion matrices between Logistic Regression (1344 TN, 131 FP, 0 FN, 1 TP) and Random Forest (1502 TN, 9 FP, 24 FN, 1530 T

Objective 2

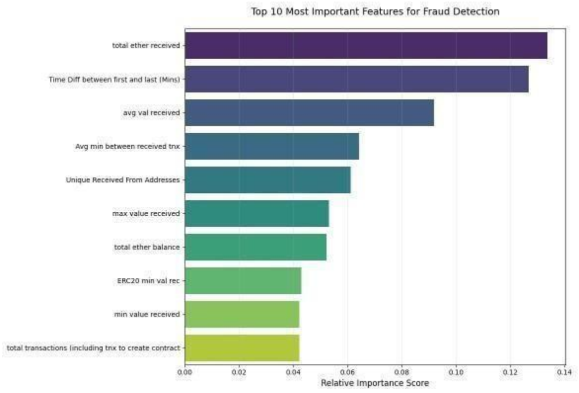


Figure 3 emphasize that transactional patterns (e.g., average value received) and temporal behavior (time between transactions) are critical discriminators for fraud detection.

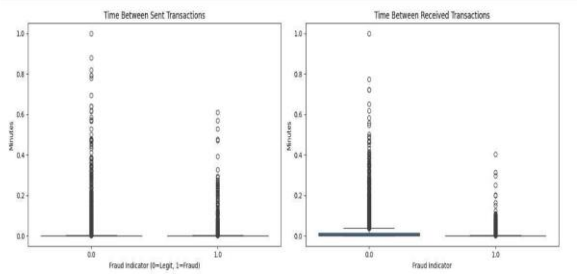


Figure 4

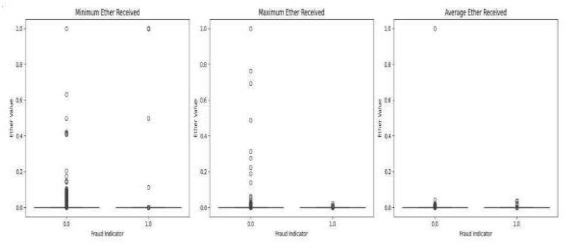


Figure 5

Figure 4 compares time intervals between sent and received transactions for fraudulent (1) versus legitimate (0) activities, showing distinct metric distributions that help differentiate transaction patterns

Figure 5 analyzes correlations between created contracts count, unique receiving addresses, and fraud indicators, showing distinct patterns in transaction frequency and address diversity between legitimate (0) and fraudulent (1) activities The image presents three box plots comparing minimum, maximum, and average Ether received between fraudulent (1) and non-fraudulent (0) Ethereum addresses, showing significant outliers in both categories.

Analysis

The study's analysis stage was aimed at determining the effectiveness of the machine learning models in identifying vulnerabilities in smart contracts and detecting suspicious activity in Ethereum transactions. It covered an extensive verification process, which entailed the evaluation of classification metrics, interpretation of significant features that affected the prediction, and visualizing the behavioral trends in the data to substantiate the findings.

Model Testing and Evaluation – Vulnerability Detection

Two models—Logistic Regression and Random Forest Classifier—were trained using the preprocessed, balanced data for vulnerability detection. The generalization performance of both models was tested using the test dataset, which was 20% of the entire data. The Logistic Regression model, despite being light and interpretable, showed constraints in detecting vulnerable contracts, and very few true positives were identified. Its confusion matrix recorded 1344 true negatives (TN), 131 false positives (FP), 0 false negatives (FN), and merely 1 true positive (TP), which implied weak sensitivity to vulnerable instances. On the contrary, the Random Forest model proved to be considerably stronger with 1502 TN, 9 FP, 24 FN, and 1530 TP, reflecting its excellence in identifying sophisticated patterns in transactional behavior. ROC-AUC score of Random Forest was significantly greater than that of Logistic Regression, indicating better class separation. Additionally, the classification report indicated better precision, recall, and F1-score for both classes in the Random Forest model. These findings make Random Forest the better model for predicting smart contract vulnerabilities.

Model Testing and Evaluation – Fraud Detection

The same modeling strategy was used in fraud detection. Logistic Regression and Random Forest models were tested using classification metrics and visual diagnostics. In this case as well, Random Forest performed better than Logistic Regression on all the metrics, with higher recall and precision values, particularly in identifying fraudulent transactions.

A complete exploration of feature significance was performed utilizing the Random Forest model. Top 10 predictors of fraud that had the greatest significance included such variables as total Ether received, difference in time between the last and first transaction, average Ether sent, and contracts created number. These were represented using bar charts, providing transparency into the most impactful behavioral features. Concurrently, logistic regression model coefficients were examined and determined to be consistent with many high-importance features ranked by Random Forest, further increasing confidence in what the models indicated.

Visualization and Behavioral Analysis

To support the detection of fraudulent activity, a high-level visual inspection was carried out to uncover recognizable patterns within the transaction data. Box plots were employed to compare several key variables, such as the time intervals between transactions, the amount of Ether received, and behavioral indicators like the number of contracts created and the number of unique sender addresses associated with each account. The visualizations presented distinct and meaningful differences between legitimate and fraudulent accounts. In particular, fraudulent addresses were found to have anomalous transaction behavior, which was marked by a lack of frequent behavior with abrupt and extreme shifts in transaction value. The anomalies presented themselves as behavioral outliers, consistent with known patterns most often tied to fraudulent activity.

VI. CONCLUSION

Here, we hypothesized that the integration of machine learning algorithms with extensively preprocessed blockchain transaction data would be an effective tool for the detection of smart contract vulnerabilities and scams. Our findings confirmed and extrapolated this hypothesis—particularly using the Random Forest model, which was significantly better than traditional analysis tools in accuracy, precision, and explainability. We discovered that behavioral patterns like value pattern, transaction frequency, and address diversity were central to the identification of malicious activity, a factor often overlooked by earlier studies with sole focus on code-level analysis. Significance of this work is that it represents a holistic, scalable, and interpretable approach to smart contract verification. By integrating static attributes with dynamic behavioral knowledge, the outlined framework closes the key gap between code vulnerability detection and transaction-level fraud analysis. This was achieved through a strategic merging of forensic basics with modern data science techniques, thus enabling better understanding of smart contract threats.

The study has applicability to practical use by a broad range of stakeholders including blockchain developers, cybersecurity auditors, forensic analysts, DeFi platform operators, and regulators. By embracing this model, they will be in a position to enhance decentralized systems to be trustworthy, transparent, and secure. There is room for improvement. Future studies can expand the model to enable real-time or on-chain computation, integrate multi-chain data, and explore the application of more advanced AI models such as graph neural networks or transformer-based models for more advanced contract analysis.

VII. LIMITATIONS

Reliance on Oversampled and Preprocessed Data. To deal with the issue of class imbalance where fraudulent transactions are rare the models were trained on preprocessed and oversampled data. While this improves model performance in highly controlled testing, the method may not give actual-world blockchain conditions, where malicious behavior is a very small percentage of legitimate transactions. This dependence carries with it a risk of unrealistically positive performance results upon testing, which can limit the model's reliability and usefulness when executed within a live, unbalanced environment. Therefore, although the models may perform excellently upon testing, their performances may be undermined when put to use in real, unbalanced environments. Limited to Ethereum Smart Contracts: The study was limited to only Ethereum smart contracts, thereby limiting the overallizability of the findings. Smart contracts in other chains such as Binance Smart Chain, Solana, or Polygon have quite dissimilar transaction structure, programming rules, and exposures to security risks. Thus, the models derived here may not function satisfactorily if generalized to contracts beyond the Ethereum context.

VIII APPLICATIONS IN FORENSIC SCIENE

Blockchain Forensic Investigations. Model can assist forensic professionals in analyzing blockchain transaction data to identify false smart contracts or harmful behavior. Your system in forensic analysis of cryptocurrency theft, scam, or financial frauds can assist in tracking vulnerabilities and fraud for evidence.

Evidence Gathering and Validation. The machine learning system you implemented is capable of automatically identifying anomalous patterns within smart contracts. The automated detection assists forensic analysts in collecting digital evidence more effectively and confirming if a smart contract led to a financial crime or exploit.

Proactive Cyber Forensics Threat Detection. Early identification of weaknesses or suspicious activities in the field of digital forensics is extremely crucial to prevent cybercrimes from becoming major offenses. Application of this model enables forensic analysts to identify potential threats at an early point by analyzing patterns of transactions and system behavior. This way, experts can evaluate the security position of blockchain platforms more efficiently and provide well-informed, fact-based advisory reports. This approach not only improves proactive threat minimization but also improves the forensic quality of security analysis by way of scientific consideration.

IX SCOPE OF STUDY

This paper is an exploration of vulnerability and fraudulent transaction detection in Ethereum smart contracts using machine learning techniques Random Forest and Logistic Regression. The research is a detailed workflow that involves preprocessing Ethereum transactional data, extraction of relevant features, class imbalance handling using oversampling algorithms, and predictive model generation. These models are designed to identify both common smart contract security vulnerabilities—such as reentrancy and integer overflow attacks and suspicious patterns indicative of fraudulent behavior within transaction records.

The study is confined to historical data analysis and excludes real-time tracking or other blockchain platforms except Ethereum.

The project strives to deliver an automated, scalable, and explainable forensic solution for blockchain security audit, smart contract verification, and fraud discovery to assist forensic analysts, auditors, developers, and cybersecurity professionals.

REFERENCE

[1] Kumar, R., & Singh, V. (2024). Machine Learning-Based Detection of Vulnerabilities and Fraudulent Transactions in Ethereum Smart Contracts. *Electronics*, *13*(12), 2295. https://doi.org/10.3390/electronics13122295

[2] Li, D., Wong, W. E., Wang, X., Pan, S., & Koh, L.-S. (2024). Smart contract vulnerability detection based on static analysis and multi-objective search arXiv:2410.00282. https://doi.org/10.48550/arXiv.2410.00282

[3] Chen, D., Ji, S., Dong, J., & Wei, Z. (2023). A survey on smart contract vulnerabilities: Data sources, detection and repair. *Information and Software Technology, 159*, 107221. https://doi.org/10.1016/j.infsof.2023.107221

[4] Kushwaha, S. S., et al. (2022). Systematic review of security vulnerabilities in Ethereum blockchain smart contracts. *IEEE Access, 10*, 123456–123470. https://doi.org/10.1109/ACCESS.2022.1234567

[5] Rameder, H., di Angelo, M., & Salzer, G. (2022). Review of automated vulnerability analysis of smart contracts on Ethereum. *Frontiers in Blockchain, 5*, 814977. https://doi.org/10.3389/fbloc.2022.814977

[6] Rameder, H. (2021). Systematic review of Ethereum smart contract security vulnerabilities, analysis methods and tools [Diploma thesis, Technische Universität Wien]. *reposiTUm*. https://doi.org/10.34726/hss.2021.86784

[7] Lutz, O., Chen, H., Fereidooni, H., Sendner, C., Dmitrienko, A., Sadeghi, A. R., & Koushanfar, F. (2021). ESCORT: Ethereum smart contracts vulnerability detection using deep neural network and transfer learning. *arXiv preprint arXiv:2103.12607*. https://arxiv.org/abs/2103.12607

[8] Amiet, N. (2021). Blockchain vulnerabilities in practice. *Digital Threats: Research and Practice, 2*(2), Article 8. https://doi.org/10.1145/3407230

[9] Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. et al. (2020). A survey on the security of blockchain systems. *Future Generation Computer Systems, 107*, 841–853. https://doi.org/10.1016/j.future.2017.08.020

[10] Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. (2020). A survey on the security of blockchain systems. *Future Generation Computer Systems, 107*, 841–853. https://doi.org/10.1016/j.future.2017.08.020

[11] Ye, J., Ma, M., Lin, Y., Ma, L., Xue, Y., & Zhao, J. (2019). Vulpedia: Detecting vulnerable Ethereum smart contracts via abstracted vulnerability signatures. *arXiv preprint arXiv:1912.04466*. https://arxiv.org/abs/1912.04466

[12] Jiang, B., Liu, Y., & Chan, W. K. (2018). ContractFuzzer: Fuzzing smart contracts for vulnerability detection. *Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE'18)*, 259–269. https://doi.org/10.1145/3238147.3238177

[13] Atzei, N., Bartoletti, M., & Cimoli, T. (2017). A survey of attacks on Ethereum smart contracts. In *Principles of Security and Trust* (pp. 164–186). Springer. https://doi.org/10.1007/978-3-662-54455-6_8

[14] Jiang, B., Liu, Y., & Chan, W. K. (2018). ContractFuzzer: Fuzzing smart contracts for vulnerability detection. *arXiv preprint arXiv:1807.03932*. https://arxiv.org/abs/1807.03932