International Journal of
**INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING**

ISSN:2147-6799 www.ijisae.org Original Research Paper

# Enhanced DOS Anomaly Detection Framework for Smart Contract Using Blockchain Metadata Analysis

**Monali Shetty [1], Dr. Sharvari Tamane [2]**

**Submitted:** 06/02/2024 **Revised:** 14/03/2024 **Accepted:** 22/03/2024

**Abstract:** Smart contracts (SCs) deployed on blockchain platforms like Ethereum blockchain provide a platform with the purpose of managing commercial arrangements. Though, the visibility of SCs makes them vulnerable to exploitation and misuse, including Denial-of-Service (DoS) attacks. This research introduces an advanced anomaly detection framework aimed at strengthening smart contract security by leveraging blockchain metadata analysis. Unlike traditional methods relying solely on transaction data, this framework incorporates comprehensive metadata like transaction sources, gas fees, and timestamps to provide contextual insights for detecting suspicious activities within smart contracts. This expanded feature set gives useful context for detecting suspicious transactions or activities in smart contracts. Transaction timestamp analysis allows for the identification of temporal patterns and trends, which in turn allows for the detection of anomalous activity, such as sharp spikes or drops in transaction frequency. Transaction-related gas fees provide information on network congestion and transaction complexity, which helps identify anomalies like unusually high or low fees that could be signs of spam or exploit attempts. Also, over the past years, many ML models has been developed to perform anomaly detection from the smart contract. The existing schemes are unable to achieve good performance due to lack in feature reduction and class balancing. In this research article novel framework is proposed which will solve class imbalancing problem and also reduce features efficiently. The Synthetic Minority Over-Sampling Technique (SMOTE) model is used for the class balancing and Principal Component Analysis (PCA) is used for the reduction of attributes. The voting classification technique is put forward to classify anomalies. The voting classification method is the combination of various classifiers like SVM, Random Forest, KNN and it use bagging approach for the final prediction.Diverse parameters, like accuracy, precision, and recall are employed to simulate the projected framework. Such parameters lead to give visions to compute this framework while classifying anomalies and regular transactions at higher accuracy. The results analysed that the projected framework yielded 91% value for all the parameters which is approx. 30% higher than existing methods.

***Keywords:*** *Blockchain, Ethereum, Smart Contract, Anomaly Detection, Blockchain Metadata, DOS attack, Machine Learning, Voting Classifier.*

## 1. Introduction

The notable advantage of blockchain technology lies in its ability to decentralize traditionally centralized services. With a market capitalization of nearly $200B and an everyday transaction volume of approximately $100B, Bitcoin [1] serves as a prominent example. Apart from Bitcoin, numerous "alt-coins" have emerged, each with its own platform and service offerings. One such platform is Ethereum, which enables the direct control of digital assets through smart contracts, which are coded instructions governing the automatic movement of assets based on predefined rules [2]. DAOs is a kind of SCs of longer period governed via shareholders. It is considered as a crowdfunded undertaking wealth fund implemented as a SC, experienced a theft of over $50 million because of a vulnerability in the Solidity SC language. This occurrence prompted hard fork of Ethereum and the creation of "alt-coin" known as Ethereum Classic. Several SCs of higher level are suffered similar attacks, resulting in the loss of their funds. The rapid development of SCs utilization results in bugs whose prevention is required in their implementation and in the Solidity language.

[1] *Research Scholar, Department of Computer Science and Engineering, Jawaharlal Nehru Engineering College, MGM University, Aurangabad, India. shettymonalin@gmail.com*
[2] *Professor and HOD, UDICT, MGM University, Aurangabad, India. hoditjnec@mgmu.ac.in*

### 1.1. Classification of Smart-Contract Vulnerabilities

Smart-Contract Vulnerabilities have diverse classes which are defined as follow:

i. Re-entrancy Attack: Re-entrancy attacks are a prevalent susceptibility in SCs, in which an attacker focuses on invoking functions at rapid level when a contract is interacted and stealing the assets or disturbing the contract logic [3]. The check-effects-interactions principle, which guarantees the completion of state updates before interacting with external contracts, served as an obstacle to this type of attack. The locking systems are effective to prevent re-entrancy. Thorough code review, analysis, and effective programming practices are essential for protecting against re-entrancy attacks and other security challenges.

ii. Integer Overflow and Underflow: The aforementioned susceptibilities are commonly observed when an integer variable's value begins to increase above or fall below the extreme or minimum value necessary to show the type of data it represents. The SCs are suffered from computing errors or random behaviour due to these attacks [4]. Thus, their safety is infected at large extent. The SafeMath library is introduced for tackling these vulnerabilities. Its protective mathematical models assist in prevent these attacks.

iii. Uninitialized Storage Pointer: These susceptibilities in SCs are occurred in the form of a class of disregarded but potentially hazardous security attacks. The major issues of these attacks is the utilization of storage pointers in SCs by coders for which they do not initialize them and do not complete this process. The malicious users focus on such pointers for accessing and modifying the data in the storage region, due to which the contract data is read, written, or altered imperfectly [5].

iv. Access Control Vulnerability: these attacks are occurred when SCs become incapable of properly controlling the permission actions. Consequently, the assailants attain potential for executing unauthentic tasks. Proper access control implementation is crucial, where contracts aims at validating the identity of a user prior to permitting some operations. By bypassing authentication measures, attackers can execute unauthorized operations, potentially resulting in asset stealing, contract tampering, or closure. Contracts should limit functions to the authority or authenticated users and capable of deploying require () statement for verifying permissions, following the principle of least privilege [6].

v. Front-End Runtime Error Vulnerability: These susceptibilities arise in some SCs portions having an interaction among user interfaces. Errors such as logic, input validation errors, result in causing downfalls in execution or random behavior, and effecting the safety of SC. These vulnerabilities are typically found in the front-end applications running smart contracts.

vi. Time Dependency: Time dependency attacks exploit time-based tasks in SCs and the features of blockchain. Attackers' potential is higher for manipulating the block timestamps for affecting time-based SC logic, potentially leading to failures or unexpected outcomes. These attacks can cause losses to contract participants.

vii. Denial-of-Service (DoS) Attack: These attacks are launched in order to disrupt access to a specific target's network or resources. When such an attack is carried out from multiple locations simultaneously, it is referred to as a DDoS attack [7]. In the context of blockchain, a DDoS attack can potentially paralyze the entire blockchain network. To mitigate DoS attacks, Ethereum implements a transaction fee mechanism known as "gas" where a specific volume of gas is essential in sending transactions. Such technique provides some resistance against DoS attacks. However, a successful DoS attack on Ethereum has been observed due to the low gas consumption price of the EXTCODESIZE opcode. The EXTCODESIZE opcode is responsible for retrieving the code size of a smart contract [8]. By exploiting the low gas consumption of this opcode, an attacker can perform a significant number of EXT-CODESIZE operations within a single transaction [attack details]. Such attacks consume considerable computing and network resources, result in network jamming or blocking. Another attack method involves the use of the SUICIDE opcode [9]. Attackers leverage the SUICIDE operation to create numerous empty accounts, which results in wasted disk resources because their storage is required in the state tree. This attack significantly slows down the process of synchronizing the node and executing the transaction within the blockchain. Additionally, denial-of-service attacks can also occur through excessive authority granted to smart contract token owners [10]. In case of objective of the token contract holder to freeze the contract at rapid level, other users of this SC become unable of executing transactions.

### 1.2. Machine Learning

In ML, a subset of AI, activities are carried out automatically without the need for explicit programming. By virtue of existent data properties, ML algorithms construct mathematical representations. They then continually learn from new sample data and modify their models as necessary [11]. These models learn patterns, adjust actions, and make decisions in an automatic way, without requiring manual intervention. In visualized era, having generation of enormous volume of data, it is impractical for humans to process and analyse all of it. ML enables automated processing of large data sets and extraction of relevant features. The ML is capable of continuously learning from novel training data, allowing its enhancement over time in case of unexpected results from the model. ML methods find extensive applications for classifying the data, detecting and predicting anomaly, etc. [12]. Examples of ML applications in everyday life include recognizing face or emotion, detecting fraud in credit card, analysing sentiments, etc.

ML techniques had diverse categories for example supervised, unsupervised and reinforcement. The initial models require data with labels for developing a mathematical approach. By feeding input data and desired results, the algorithm becomes applicable for extracting the association among the input data and labels, enabling it to accurately predict outcomes for hidden input data. It is an extensive approach for forecasting or classifying particular outcomes [13].

In contrast, unsupervised learning algorithms use data of no labels for training the model. This approach aims to discover hidden insights and structures within the dataset, grouping similar data points into categories or clusters. Unsupervised approach frequently uses cluster analysis, which groups together comparable data points so that outliers that don't fit into any clusters can be identified as anomalies.

Reinforcement learning algorithms learn when an exterior environment is related, and receive feedback on their actions. Different ML techniques are suitable for different operations. For instance, prior two approaches are exploited to analyse the data, while this approach is employed to tackle the issues related to make decision [14]. Intelligent agents can find the optimum course of action to maximize the overall profit over the long term by investigating and studying their surroundings. We refer to this technique as reinforcement learning.

Deep learning has gained popularity in recent years for tackling complex tasks [15]. It is planned on the basis of ANNs with multiple layers for extracting high-level attributes from inputs. Common models in DL include MLP, CNN, and RNN. Deep learning's potential for attaining abstractions of higher quality to model the data, has found applications in regions of recognizing pictures and NLP.

### 1.3. Reinforcement learning for Anomaly detection in Blockchain

In the field of detecting security of SC, reinforcement learning (RL) has emerged as a prominent ML approach, demonstrating notable achievements across various application domains [16]. Through interactions with the scenario, RL allows intelligent agents for discovering and to learn effective strategies in an autonomous way for maximizing long-term collective rewards.

- Q-Learning: The update of action-value function Q(s, a) can be done using the model-free RL method known as Q-Learning. This function estimates the expected total return when a specific action is carried out in an available state [17], which helps agent in selecting the precise actions on the basis of Q-values. This approach is expressed as

$$new\ Q(s,a) \leftarrow Q(s,a) + \propto [\,r(s,a) + \gamma\, Max\, Q'(s',a') - Q(s,a)] \quad - \qquad (1)$$

In equation (1), α denotes the learning rate and the discount factor is represented with γ. It is an extensively applied approach in diverse domains, such as automated control and network forwarding, showcasing its classical significance.

- Deep Q-Network: It is other RL method in which Deep Neural Network (DNNs) is integrated with Q-Learning method. It leverages Neural Network for approximating Q(s, a), allowing it to handle complicated input state spaces, including high-dimensional raw pictures. The updated formula for this approach as:

$$Q(s,a;\theta) \leftarrow Q(s,a;\theta) + \alpha\,[\gamma\, Max\, Q(s',a';\theta) - Q(s,a;\theta)] \quad - \qquad (2)$$

In this equation 2, $\theta$ represents the metrics of the existing NN, while $\theta'$ represents the metrics of the target NN.

- GAIL: Generative Adversarial Imitation Learning (GAIL) is an approach that integrates GANs with RL [18]. It enables agents to learn effective rules by imitating expert policies. GAIL involves two key components: the generator loss, which guides the policy learning, and the discriminator loss, which distinguishes expert demonstrations from policy-generated trajectories.
- MCTS: Monte Carlo Tree Search (MCTS) is an approach that utilizes MC simulations to construct a Seek Tree and iteratively find an approximate explanation when the examination and utilization is balanced. No general formula is included in this approach. However, it involves 4 main phases such as to select the data, expand it, simulate, and BP. MCTS has been successfully combined with deep learning in notable examples like AlphaGo [19], showcasing the effectiveness of integrating MCTS with deep learning techniques in various domains.

Recent research has highlighted the susceptibility of Ethereum Smart Contracts developed using Solidity to various attacks. One notable vulnerability is the DoS Unexpected Revert flaw, which can lead to Denial of Service (DoS) incidents [20]. In essence, this vulnerability arises from mishandling incomplete transactions, whether due to errors or intentional reversals, resulting in Smart Contracts becoming non-functional. Current techniques of detecting vulnerabilities primarily rely upon predefined rules set via professionals, and not very effective and struggle with accessibility. Though some studies employed ML for extracting contract attributes in identifying vulnerabilities, these approaches often overlook crucial aspects and fail to fully leverage the information within SCs. A novel method is essential to detect vulnerabilities in SCs for tackling drawbacks of traditional methods [21].

## 2. Literature Review

N. F. Samreen, et.al (2021) projected a framework known as SmartScan in which the static analysis was integrated with dynamic one for detecting DoS fragility brought on by a sudden relapse in Ethereum Smart Contracts [22]. This methodology was helpful for scanning the smart contracts being tested so that vulnerable patterns might be found. The latter analysis was conducted for determining whether the DoS-Unexpected Revert vulnerability was exploitable. It led to enhance the efficacy and attain more optimal outcomes. A set containing 500 smart contracts generated via the Etherscan was utilized in experimentation. The findings depicted that the suggested model was useful for enhancing the precision and recall in contrast to existing methods. This model was not adaptable on all kinds of applications.

Fadi, et.al (2024) proposed the utilization of adaptive augmentation in conjunction with contrastive learning as a means to detect reentrancy and endless loop attacks in smart contracts [23]. This innovative approach had been shown to enhance performance in downstream tasks, such as smart contract categorization, by effectively learning task-agnostic representative data features. Through the use of this technique, researchers were able to address concerns about interpretability and the lack of representative datasets, as well as dive into the complexities of smart contract characteristics. The study demonstrated how well contrastive learning, GNN networks, and adaptive augmentation work together to detect re-entrancy and infinite loop assaults in real-time smart contracts. Empirical data showed that this method worked better than conventional baseline models. Contrastive learning must be combined with neural network-based models because the latter cannot identify timestamp attacks on its own.

Duan, et.al (2023) introduced an innovative method for identifying susceptibilities in SCs after extracting data from different tiers of smart contracts and utilizing these features for training ML algorithms to detect vulnerability efficiently [24]. This approach involved utilizing a pre-trained CodeBERT model which extracted token attributes from the source code and 2-gram attributes from the opcodes of SCs. Hence, the semantic content of SCs was captured at multiple stages. To uncover vulnerabilities in contracts, the aggregation of extricated attributes was done and they were also fused, and inputted into machine learning models. To validate this approach, over 10,266 smart contracts were analysed. The experiments demonstrated detection accuracies of up to 98%, 98%, and 94% for three different vulnerabilities, respectively. The suggested method demonstrated efficacy in automatically detecting smart contract vulnerabilities, with an average detection time of 0.99 seconds for SC. It is crucial to remember that although ContractGuard might be excellent at finding vulnerabilities, it might not be as good at preventing logical mistakes in complicated predicates.

A. Ghaleb, et.al (2022) analyzed those smart contracts (SCs) which were deployed on Ethereum blockchain, made the utilization of gas [25]. The users submitted the invocation requests of contracts for this gas. However, Denial-of-Service (DoS) was occurred on these patterns for activating abnormal behaviour in the targeted victim contracts. These attacks were named as gas-related vulnerabilities. Thus, a static analyser called eTainter was developed to detect such susceptibilities on the basis of taint tracking in the bytecode of SCs. A comparison of the new strategy with the standard approaches was done. According to results on a dataset of annotated contracts, the developed approach yielded a precision and recall up to 90%. However, the analysis depicted the existence of gas-related vulnerabilities in 2,763 of these contracts.

Wang, et.al (2020) introduced ContractGuard, the pioneering IDS specifically engineered for defending against attacks on ESCs [26]. ContractGuard identifies intrusion attempts through the detection of aberrant control flow, a method commonly employed by traditional program IDSs. To achieve this, the model was developed to profile context-tagged acyclic pathways, embedded within the contracts, and optimized using the Ethereum gas-oriented performance framework. The primary goal of this model was to minimize costs, a critical consideration given the necessity of upfront payment for digital concurrency. Through the implementation of this methodology, only 36.14% of overhead in implementation and 28.27% in running were increased. Remarkably, the suggested model successfully thwarted attacks on 83 percent of seeded issues and all real-world vulnerabilities.

Though, the practical application of this notion may be limited in smart contract programs that heavily rely on features requiring frequent external calls to unprotected contracts.

Yang, et.al (2024) introduced CrossFuzz, a fuzz testing-based technique designed to detect cross-contract vulnerabilities [27]. This innovative approach addressed two key issues: the generation of constructor parameters and the manipulation of transaction sequences. To begin, CrossFuzz generated constructor parameters for the contract under examination subsequent to trace the data flow of address-kind metrics. It then identified function definitions and state variable usages by analysing the interactions between function calls across contracts. Additionally, CrossFuzz optimized the mutation of transaction sequences using the Inter-Contract Data Flow (ICDF) method, ultimately conducting comprehensive cross-contract fuzz testing. The CrossFuzz test findings showed a notable enhancement in security fault detection, with the suggested approach finding 1.82 times more vulnerabilities than the state-of-the-art methods. Moreover, CrossFuzz demonstrated its efficacy in detecting cross-contract vulnerabilities by increasing bytecode coverage to 0.1058 against the standard. It was significant to highlight that the introduced model does not take into account scenarios where branch requirements are already satisfied by default constructor values for state variables, nor does it apply to test cases where SC interacts with addresses.

Ndiaye, et.al (2023) discussed that smart contracts (SCs) were effective and cost-effective targets for aggressors due to storage of huge volume of money in them [28]. Thus, a novel anomaly detection (AD) model called ADEFGuard was developed on the basis of behaviour of SCs, as a novel feature. A learning and monitoring module was executed for determining the deceitful behaviours of SCs. This model was performed better in contrast to other techniques concerning 3 facets. Initially, a unified solution was generated for diverse kinds of scams so skills required to analyse code were avoided. Secondly, its inference was instructions of magnitude which worked quickly rather than analysing code. Lastly, the developed model yielded accuracy up to 85%, precision up to 75%, and recall up to 90% to detect malicious contracts. Moreover, this model was robust to detect new malevolent behaviours of SCs. But, this model was ineffective of determining whether SS was authentic or not.

N. Ashizawa, et.al (2022) presented a ML-based method known as Eth2Vec for detecting susceptibilities in SC [29]. This approach remained robust against code rewrites that was useful to detect the vulnerabilities even in rewritten codes. Unlike other methods, a neural network (NN) model was adopted to process the language with the objective of learning the attributes of vulnerable contracts automatically. The aforementioned approach sought to identify the weaknesses in SCs when the similarities of codes of a target contract were contrasted against learning ones. In the light of experiments on Etherscan, the presented technique had yielded superior precision, recall, and F1-score. It efficacy was mitigated under diverse areas/settings.

M. Eshghie, et.al (2021) established a monitoring system called Dynamit to find Ethereum smart contracts' re-entrancy issues [30]. The blockchain system's transaction metadata along with balanced data were used by the framework. Notably, Dynamit didn't need unique execution environments, code instrumentation, or domain-specific knowledge. Instead, it used an algorithm contingent on machine learning (ML) to identify transactions as either legitimate or fraudulent by extracting attributes from the information about the transactions. This model was adaptable to detect the vulnerable contracts to re-entrancy attacks and attain execution trace for re-generating the assault. The results indicated that the devised model offered 90% above accuracy with random forest (RF) algorithm on 105 transactions. Diversity of data features led to make this system ineffective in some scenarios.

Hu, et.al (2020) suggested a method for Ethereum smart contracts that uses transaction-based categorization and detection to solve these problems [31]. Ethereum is considered to extract ten thousand SCs, which concentrated on the data behaviour produced via users and SCs. Through manual analysis, this work found four patterns of behaviour from the transactions that can be utilized to differentiate among diverse contract kinds. From this, fourteen basic characteristics of SC were then built. This paper developed a data slicing algorithm to partition the collected SCs in order to build database. After that, our datasets were trained and tested using an LSTM network in this work. The comprehensive testing results demonstrated that this method can effectively identify various contract types and be used for detecting anomalies and malevolent SC concerning recall, f1-measure, and precision. In datasets, the imbalance problem arises.

Shen, et.al (2021) suggested a novel detection methodology utilizing bytecode to identify Ponzi schemes within smart contracts [32]. Two primary strategies were used in the model to demonstrate this novel approach: first, the bytecode was effectively transformed into a high-dimensional matrix that included all proposed attributes, with two bytes denoting a single feature. Subsequently, the identification of Ponzi schemes was ingeniously reframed as an anomaly detection challenge. In the end, this anomaly detection method was successful in identifying Ponzi schemes inside smart contracts. Experimental results show that the proposed detection methodology significantly increased the detection accuracy of Ponzi scheme contracts. Hence, the suggested methodology achieved F1-measure up to 0.88, surpassing the performance of traditional detection models. This high F1-score of 0.88 significantly outperformed other conventional detection models. Moving forward, a more comprehensive analysis of the properties of bytecodes is essential to further enhance the identification of Ponzi scheme contracts.

Huang, et.al (2022) created a MTL-relied technique to detect anomaly in SC [33]. The major intend was to improve the potentials of this technique for recognizing and detecting susceptibilities when additional tasks were allocated for learning more significant attributes. The hard-sharing model containing 2 elements were considered as the basis. Initially, the last layer was employed to learn the semantic information of inputted SC. Subsequently, task-specific layer was executed for finding the operation of every task. The objectives of every task were completed using existing CNN. This model assisted in developing a classifier to learn and extract attributes from the common layer, which trained the data. in experimentation, the created technique was proved effective to detect vulnerabilities and proved suitable for detecting various kinds of susceptibilities. In comparison to a single-task model, this model was less costly in terms of computing, storage, and time. This study used the HPS approach that diminished the efficacy of model and limited its capacity for generalization.

Zhang, et.al (2022) suggested an innovative hybrid DL algorithm called CB-GRU, which cleverly mixed many Deep Learning techniques with several word embeddings (Word2Vec, FastText) [34]. In order to discover smart contract vulnerabilities, the model merged features that it had gathered using various deep learning models. Through a set of studies, this paper showed that the suggested algorithm was more robust to detect vulnerabilities on SC on the dataset SmartBugs Dataset-Wild, which is presently available to the public. When comparing the suggested model's performance to earlier research, it was found that the suggested algorithm was performed better to detect anomalies in SCs. For detecting susceptibilities in SCs with cryptic attributes, it is necessary to find several SC susceptibilities within similar SC and to increase the precision of this algorithm.

### 2.1. Research Gaps

Diverse research gaps are defined as:

1. **Integration of Comprehensive Metadata**: Many existing frameworks focus on specific aspects of transaction data or contract code analysis. For instance, methods like SmartScan and ContractGuard primarily use static and dynamic analyses of smart contracts but do not extensively incorporate broader blockchain metadata such as transaction sources, gas fees, and timestamps. This comprehensive metadata can provide more contextual insights and improve the detection of anomalies and vulnerabilities.
2. The methods proposed in the earlier years are based on quantitative analysis. By quantitative analysis, data can be analyzed but future trends cannot be predicted. Therefore, some new smart contract data classification methods must come into existence in order to predict future trends
3. The smart contract dataset has the problem of class imbalancing due to which class overfitting or under fitting problems get raised. In the existing methodologies very less work is done to solve problem of class imbalancing so that performance can be improved for the smart contract data classification.

## 3. Research Methodology

The principal target of this research is to detect anomalies in blockchain networks. The classification process involves several phases, such as to pre-process data, extract features, and classify data. This is how the work is conducted:

**1. Dataset input and pre-processing:** The preliminary phase involves using data collected from authentic sources as input. In this analysis, the data from Ethereum block chain was fetched via Etherscan API. By applying the Ethereum blockchain, Etherscan API stores the transaction data from Ethereum. An in-depth look at Ethereum blockchain structure has been provided by the study of markets of smart contracts and transactions activity that resulted in the provided extensive dataset. By sending requests to the API we were capable to acquire the transaction details, namely (nonce or block numbers), gas information and so. This dataset includes information on over 12,000 entries of Ethereum Blockchain node data. But the main challenge was to find malicious smart contract from Ethereum network. All previous smart contracts were non-malicious.

Detection of DOS attacks in live smart contract requires monitoring on-chain activity and leveraging a combination of techniques. We have used following approaches on smart contracts to detect vulnerabilities:

a) Transaction monitoring
   - Track transaction rate: Monitor the frequency of transactions interacting with the smart contract. Sudden spikes or drops in transaction volume can indicate a DoS attempt.
   - Analyze Gas Usage: Keep an eye on gas consumption per transaction. Unusually high or low gas fees might be a sign of malicious activity. Transactions with minimal gas might be attempting to clog the network, while exceptionally high gas fees could be an attempt to exploit the contract.

b) Log Analysis
   - Identify Repetitive Transactions: If the same transaction or a slight variation keeps appearing, it could be a DoS attempt.
   - Unusual Function Calls: Monitor calls to specific contract functions. If uncommon functions are being called repeatedly, it could be a sign of exploration for vulnerabilities.

c) Proactive measures
   - Gas Limit Checks: Implement checks within the smart contract to prevent transactions exceeding a set gas limit.

Based on above approaches we have prepared dataset. Our dataset is split between malicious nodes and non-malicious nodes. This dataset is perfect for blockchain researchers interested in data visualization and analysing node behaviour on the blockchain.

**Purpose of Dataset**

The purpose of this dataset is to facilitate the detection and analysis of Denial-of-Service (DOS) attacks targeting smart contracts in blockchain systems. By providing a comprehensive collection of data, researchers and security professionals can utilize this dataset to develop and validate algorithms, models, and techniques for identifying and mitigating DOS attacks in smart contracts.

**Dataset Structure**

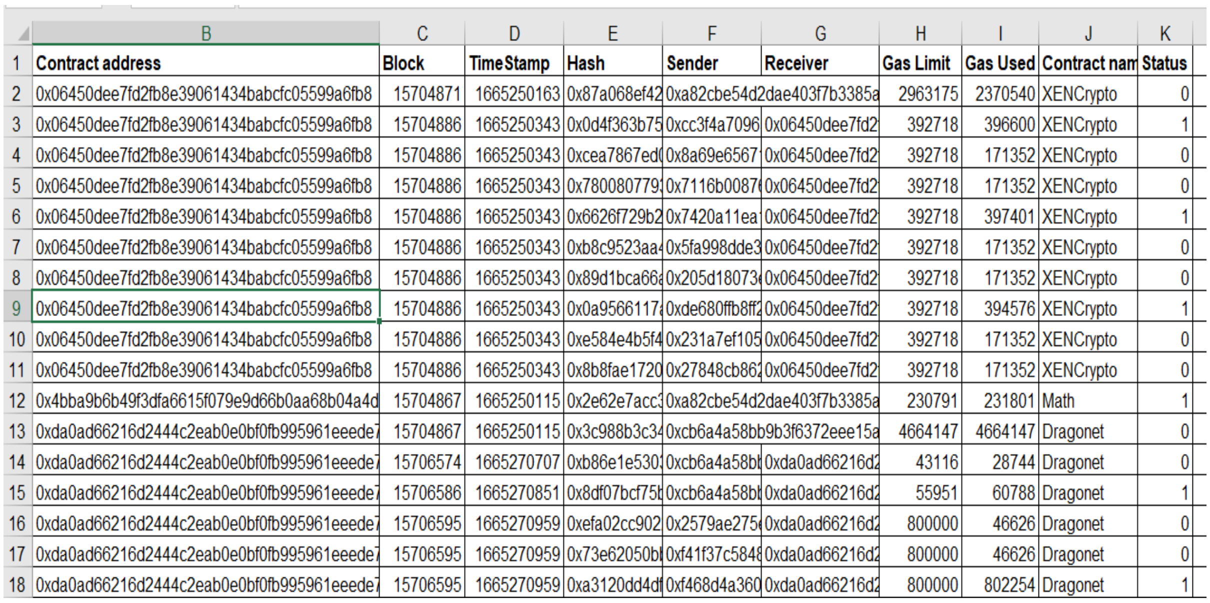
The dataset contains the following columns, which serve as crucial parameters for analysing smart contract transactions:

1. Contract Address: The unique identifier of the smart contract involved in the transaction.
2. Block Number: The block number in which the transaction was included.
3. TimeStamp: The timestamp indicating the date and time when the transaction occurred.
4. Hash: The transaction hash, serving as a unique identifier for the transaction.
5. Sender Address: The address of the sender who initiated the transaction.
6. Receiver Address: The address of the recipient or the smart contract being interacted with.
7. Gas Limit: The maximum amount of gas allocated for the transaction.
8. Gas Used: The actual amount of gas consumed by the transaction.
9. Contract Name: The title commonly used to identify the contract.
10. Status (Output Label): A label indicating the nature of the transaction, where '0' denotes a non-malicious transaction and '1' denotes a potentially malicious transaction associated with a DOS attack.

By utilizing this dataset, researchers and practitioners can analyse the characteristics, patterns, and behaviours of transactions to identify potential DOS attacks on smart contracts. The dataset serves as a valuable resource for developing machine learning models, anomaly detection algorithms, and other techniques aimed at enhancing the security and resilience of smart contracts in blockchain systems.

It is important to note that this dataset primarily focuses on DOS attacks in smart contracts and does not encompass other forms of attacks or vulnerabilities.

**Samples from the Dataset**



| | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Contract address | Block | TimeStamp | Hash | Sender | Receiver | Gas Limit | Gas Used | Contract nam | Status |
| 2 | 0x06450dee7fd2fb8e39061434babcfc05599a6fb8 | 15704871 | 1665250163 | 0x87a068ef42 | 0xa82cbe54d2dae403f7b3385a | | 2963175 | 2370540 | XENCrypto | 0 |
| 3 | 0x06450dee7fd2fb8e39061434babcfc05599a6fb8 | 15704886 | 1665250343 | 0x0d4f363b75 | 0xcc3f4a7096 | 0x06450dee7fd2 | 392718 | 396600 | XENCrypto | 1 |
| 4 | 0x06450dee7fd2fb8e39061434babcfc05599a6fb8 | 15704886 | 1665250343 | 0xcea7867ed | 0x8a69e6567 | 0x06450dee7fd2 | 392718 | 171352 | XENCrypto | 0 |
| 5 | 0x06450dee7fd2fb8e39061434babcfc05599a6fb8 | 15704886 | 1665250343 | 0x780080779 | 0x7116b0087 | 0x06450dee7fd2 | 392718 | 171352 | XENCrypto | 0 |
| 6 | 0x06450dee7fd2fb8e39061434babcfc05599a6fb8 | 15704886 | 1665250343 | 0x6626f729b2 | 0x7420a11ea | 0x06450dee7fd2 | 392718 | 397401 | XENCrypto | 1 |
| 7 | 0x06450dee7fd2fb8e39061434babcfc05599a6fb8 | 15704886 | 1665250343 | 0xb8c9523aa | 0x5fa998dde3 | 0x06450dee7fd2 | 392718 | 171352 | XENCrypto | 0 |
| 8 | 0x06450dee7fd2fb8e39061434babcfc05599a6fb8 | 15704886 | 1665250343 | 0x89d1bca66 | 0x205d18073 | 0x06450dee7fd2 | 392718 | 171352 | XENCrypto | 0 |
| 9 | 0x06450dee7fd2fb8e39061434babcfc05599a6fb8 | 15704886 | 1665250343 | 0x0a9566117 | 0xde680ffb8ff2 | 0x06450dee7fd2 | 392718 | 394576 | XENCrypto | 1 |
| 10 | 0x06450dee7fd2fb8e39061434babcfc05599a6fb8 | 15704886 | 1665250343 | 0xe584e4b5f4 | 0x231a7ef105 | 0x06450dee7fd2 | 392718 | 171352 | XENCrypto | 0 |
| 11 | 0x06450dee7fd2fb8e39061434babcfc05599a6fb8 | 15704886 | 1665250343 | 0x8b8fae1720 | 0x27848cb86 | 0x06450dee7fd2 | 392718 | 171352 | XENCrypto | 0 |
| 12 | 0x4bba9b6b49f3dfa6615f079e9d66b0aa68b04a4d | 15704867 | 1665250115 | 0x2e62e7acc | 0xa82cbe54d2dae403f7b3385a | | 230791 | 231801 | Math | 1 |
| 13 | 0xda0ad66216d2444c2eab0e0bf0fb995961eeede7 | 15704867 | 1665250115 | 0x3c988b3c34 | 0xcb6a4a58bb9b3f6372eee15a | | 4664147 | 4664147 | Dragonet | 0 |
| 14 | 0xda0ad66216d2444c2eab0e0bf0fb995961eeede7 | 15706574 | 1665270707 | 0xb86e1e530 | 0xcb6a4a58b | 0xda0ad66216d2 | 43116 | 28744 | Dragonet | 0 |
| 15 | 0xda0ad66216d2444c2eab0e0bf0fb995961eeede7 | 15706586 | 1665270851 | 0x8df07bcf75 | 0xcb6a4a58b | 0xda0ad66216d2 | 55951 | 60788 | Dragonet | 1 |
| 16 | 0xda0ad66216d2444c2eab0e0bf0fb995961eeede7 | 15706595 | 1665270959 | 0xefa02cc902 | 0x2579ae275 | 0xda0ad66216d2 | 800000 | 46626 | Dragonet | 0 |
| 17 | 0xda0ad66216d2444c2eab0e0bf0fb995961eeede7 | 15706595 | 1665270959 | 0x73e62050b | 0xf41f37c584 | 0xda0ad66216d2 | 800000 | 46626 | Dragonet | 0 |
| 18 | 0xda0ad66216d2444c2eab0e0bf0fb995961eeede7 | 15706595 | 1665270959 | 0xa3120dd4d | 0xf468d4a360 | 0xda0ad66216d2 | 800000 | 802254 | Dragonet | 1 |

**Figure 1.** Dataset Samples: Live Smart Contract Extracted from Etherscan.io

To address the issue of class imbalance, the SMOTE technique will be employed.

**2. Feature extraction:** This function intends to determine the relationship between each attribute and the target set. FN is a case, in which specimen is truly an invasion but is classed as regular, and in FP, a specimen seems regular but is identified as an invasion. Poor FN occurs when an intrusion goes undetected. A layered approach utilizing multiple intrusion detection systems (IDSs) is employed to enhance detection. Principal Component Analysis method is assisted in alleviating the number of features.

**3. Classification:** The classification task involves dividing the whole dataset into 2 sections. A voting mechanism is presented to classify network traffic. Diverse methods, including random forest, SVM, and KNN, are integrated to achieve the objective of classifying anomalies. The RF algorithm creates a hybrid by creating an array of distinct decision trees. Each tree predicts the data based on majority votes. Random sampling with replacement is employed to develop the trees, ensuring randomness and reducing the sensitivity of Decision Trees when the data is trained. This approach helps maintain feature randomness by allowing each tree to make decisions based on a random subset of attributes. SVM (Support Vector Machine) utilizes kernel representation and optimized margin. It constructs a hyperactive plane to separate different classes of information. The kernel interplanetary explores the least hyper range, including all working out cases, and determines the location of an examination case in the hyper sphere. KNN (K-Nearest Neighbour) algorithm often yields optimal results. It can enhance existing algorithms by integrating prior knowledge. The majority label among the nearest neighbours, determined based on a distance metric, is used to classify unlabelled instances using the KNN rule. The effectiveness of such classifiers is based on diverse components, like choice of distance metric and the number of neighbours considered in KNN. The voting classification used bagging method to form final prediction. The weights are assigned to each classifier and best prediction of each classifier get merged to form final prediction. The pseudo code of voting classifier is presented below: -

**Voting Classifier Pseudo Code**

```
Input: A Stream of pairs (x,y),
      Parameter $\beta \in (0,1)$
Output: A Stream of prediction $\hat{y}$ for each $x$
1. Initialize experts $C_1 \dots . C_n$ with weight $\omega_i = 1/N$ each
2. for each $x$ in stream
      do Collect Predictions $C_1(x) \dots C_N(x)$
            $P \leftarrow \sum_i \omega_i \cdot C_i(x)$
            $\hat{y} \leftarrow Sign\ (P - \frac{1}{2})$
            for $i \in 1 \dots N$
                  do if $(C_i(x) \neq y)$ then
                  $\omega_i \leftarrow \beta \cdot \omega_i$
      $S \leftarrow \sum_i \omega_i$
      for $i \in 1 \dots . . N$
            do $\omega_i \leftarrow \omega_i / s$
```

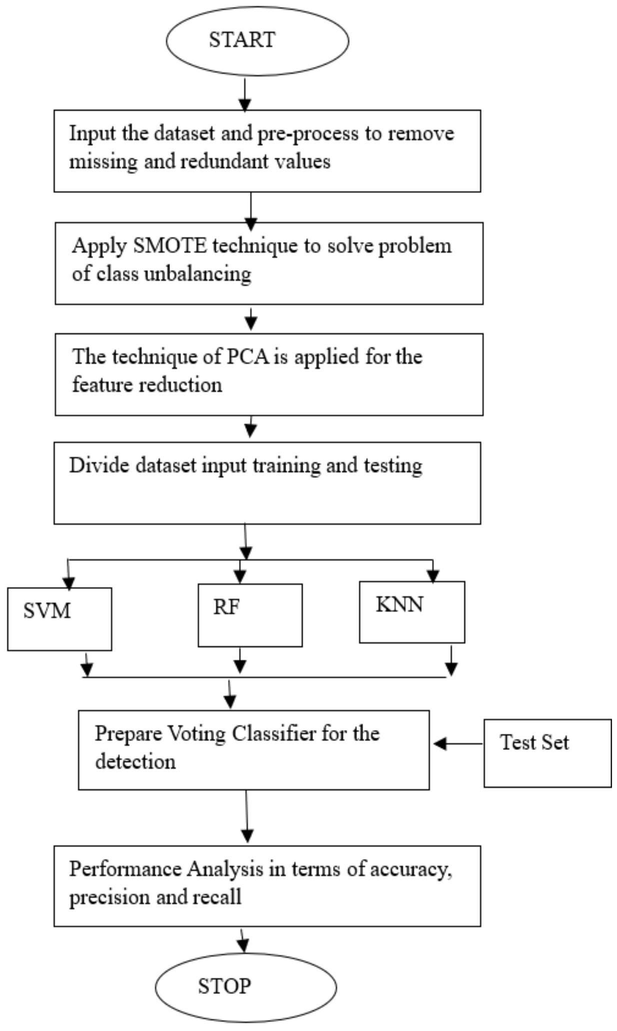

START
Input the dataset and pre-process to remove missing and redundant values
Apply SMOTE technique to solve problem of class unbalancing
The technique of PCA is applied for the feature reduction
Divide dataset input training and testing
SVM
RF
KNN
Prepare Voting Classifier for the detection
Test Set
Performance Analysis in terms of accuracy, precision and recall
STOP

**Figure 2.** Flowchart of the Proposed Framework

## 4. Result and Discussion

This research focuses on the classification of network traffic, specifically in the context of anomaly detection. The classification framework involves several phases, such as to pre-process data, extract features, and classify data. The dataset used for testing the utilized model is based on blockchain technology. The dataset contains the columns for 'Contract address, Block Number, TimeStamp, Hash, Sender Address, Receiver Address, Gas Limit, Gas Used and Output Label'. The data extracted was all non-malicious, so in order to get malicious transactions we had to add some threshold value to simulate the DOS attack. In the dataset the '0' status denotes a non-malicious transaction and '1' denotes a malicious transaction.  The efficacy level of the introduced technique is evaluated using three metrics (i.e., accuracy, precision, recall).
The devised framework proposed, combines multiple models for anomaly detection. It integrates the use of Synthetic Minority Oversampling Technique, Principal Component Analysis, and a Voting mechanism.

**Table 1:** Performance Analysis

| Model | Accuracy | Precision | Recall | Dataset Used |
|---|---|---|---|---|
| GA based model [14] |  | 77 % | 76 % | Own dataset -BCCC-VolSCs-2023- |
| CNN Model [24] | 77 % | 77.50 % | 74.46 % | Smart Contract Sanctuary [31] |
| Proposed Model | 91.37 % | 90 % | 90 % | Own Dataset extracted from Etherscan.io and checked on simulated environment. |

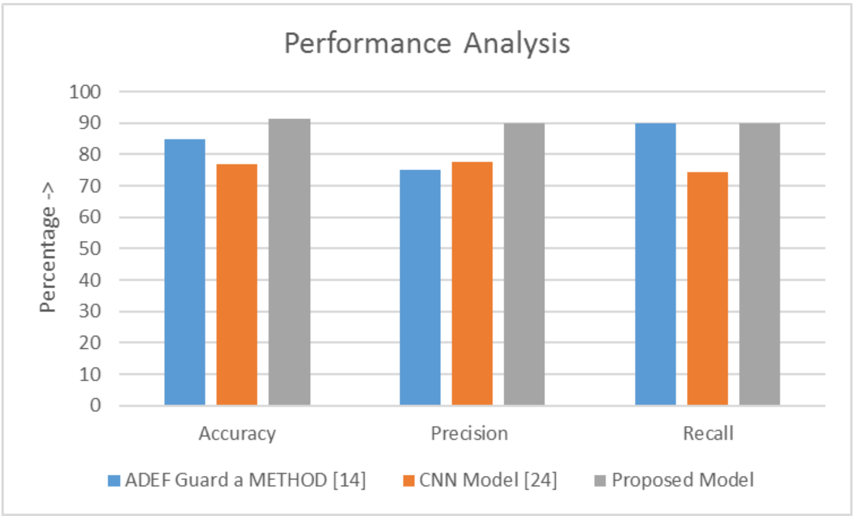


**Figure 3.** Result Analysis

Figure 3. illustrates the result evaluation of machine learning algorithms, specifically comparing SVM, KNN, and the developed architecture in view of three parameters. In investigation, the suggested mechanism gets higher accuracy rate up to over 90%. It is analysed from the results that existing models like SVM and KNN models able to achieve accuracy of 68 and 66 percent respectively which is approx. 30 percent low as compared to proposed model. The accuracy of proposed model is increased because it solves class imbalancing problem using SMOTE model and feature of the dataset reduced efficiently using PCA model. In the last voting classifier is applied which use bagging method to form final prediction.

## 5. Conclusion

Smart contract technologies allow users to create decentralized digital agreements without relying on intermediaries. These technologies have gained popularity in diverse domains like medical, business management, shareholder agreements, and insurance. Though, with developing usage of SCs, the interest of potential attackers is increased, leading to the discovery of numerous vulnerabilities and exploitations. This study highlights the presence of vulnerabilities and the occurrence of attacks in smart contract technology, emphasizing that it is not immune to security risks. This research work introduces a novel model that combines three key components: SMOTE, PCA, and Voting Classification. The Voting method is a blend of SVM, RF and K-Nearest Neighbor. The proposed model is compared with the existing classification models like SVM and KNN which achieves accuracy of 68 percent and 66 percent respectively. The accuracy of suggested mechanism is counted 91% that is 30% higher than existing models. The proposed model achieve accuracy because it solves class imbalancing problem and also reduce feature efficiently. In future proposed model can be further extended using optimization algorithms for the feature reduction and it is expected that with the use of optimization algorithms above 95 percent accuracy can achieved.

## 6. Conflict of Interest

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

## 7. Ethical Approval

All authors declare that they adhere to the ethical principles of the journal.

## References

[1] Xu Y, Hu G, You L, Cao C. A Novel Machine Learning-Based Analysis Model for Smart Contract Vulnerability. Security and Communication Networks. 2021; 7(10): 63-71. doi: 10.1155/2021/5798033.

[2] Mandloi J, Bansal P. A Machine Learning-Based Dynamic Method for Detecting Vulnerabilities in Smart Contracts. International Journal of Applied Engineering & Technology. 2022; 4(2): 110-118. doi: ijaet%20v4-2-2022-17.pdf

[3] Chen D et al. Privacy-Preserving Anomaly Detection of Encrypted Smart Contract for Blockchain-Based Data Trading. IEEE Transactions on Dependable and Secure Computing. 2024; 78: 13-20. doi: 10.1109/TDSC.2024.3353827.

[4] Haritha P, Kavitha V and Manimala G. Protection & Privacy Embedding Blockchain Established Fraud Detection. In: International Conference on Applied Artificial Intelligence and Computing (ICAAIC). Salem. India; 2022. p. 1437-1444. doi: 10.1109/ICAAIC53929.2022.9792962.

[5] Chu H, Zhang P and Li W. A survey on smart contract vulnerabilities: Data sources, detection and repair. Information and Software Technology. 2023; 159: 906-914, 10.1016/j.infsof.2023.
107221

[6] Shah H, Shah D, Jadav N K, Gupta R, Tanwar S, Alfarraj O, Tolba A, Raboaca M S, Marina V. Deep Learning-Based Malicious Smart Contract and Intrusion Detection System for IoT Environment. Mathematics. 2023; 11(2): 418-425. doi: 10.3390/math11020418.

[7] Gogineni A K, Swayamjyoti S, Sahoo D, Sahu K K, Kishore R. Multi-Class classification of vulnerabilities in smart contracts using AWD-LSTM, with pre-trained encoder inspired from natural language processing. IOP Science, 2020; 8: 16-24. doi: 10.1088/2633-1357/abcd29.

[8] Krichen M. Strengthening the Security of Smart Contracts through the Power of Artificial Intelligence. Computers. 2023; 12: 107-116. doi:10.3390/computers12050107.

[9] Sosu R N I, Chen J, Brown-Acquaye W, Owusu E, Boahen E. A Vulnerability Detection Approach for Automated Smart Contract Using Enhanced Machine Learning Techniques. Research Square. 2022; 2: 96-104. doi: 10.21203/rs.3.rs-1961251/v1.

[10] Moubarak J, Chamoun M and Filiol E. Developing a K-ary malware using blockchain. In: IEEE/IFIP Network Operations and Management Symposium. Taipei. Taiwan; 2018. p. 1-4. doi: 10.1109/NOMS.2018.8406331.

[11] Liu L, Tsai W-T, Liu M. Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. Future Generation Computer Systems. 2021; 128: 158-166. doi: 10.1016/j.future.2021.08.023.

[12] Demertzis K, Iliadis L, Tziritas N and Kikiras P. Anomaly detection via blockchained deep learning smart contracts in industry 4.0. Neural Computing and Applications. 2020; 32: 36-42. doi: 10.1007/s00521-020-05189-8.

[13] Liu H, Fan Y and Wei Z. Vulnerable smart contract function locating based on Multi-Relational Nested Graph Convolutional Network. Journal of Systems and Software. 2023; 204: 11-22. doi: 10.1016/j.jss.2023.111775

[14] Haji S H, Lashkari A H and Oskui A M. Unveiling vulnerable smart contracts: Toward profiling vulnerable smart contracts using genetic algorithm and generating benchmark dataset. Blockchain: Research and Applications. 2023; 5(1): 1007-1015, doi: 10.1016/j.bcra.2023.100171.

[15] Wang L, Cheng H and Zhu X. Ponzi scheme detection via oversampling-based Long Short-Term Memory for smart contracts. Knowledge-Based Systems. 2021; 228: 132-139. doi: 10.1016/j.knosys.2021.107312.

[16] Su S et al. Detecting Smart Contract Project Anomalies in Metaverse. In: IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom). Kyoto. Japan; 2023. p. 524-532. doi: 10.1109/MetaCom57706.2023.00095.

[17] Liu X, Jiang F and Zhang R. A New Social User Anomaly Behavior Detection System Based on Blockchain and Smart Contract. In: IEEE International Conference on Networking, Sensing and Control (ICNSC). Nanjing. China; 2020. p. 1-5. doi: 10.1109/ICNSC48988. 2020.9238118.

[18] Ndiaye M, Konate K and Ndoye E H M. Anomaly Detection Algorithm Based on Smart Contracts Behaviours in Ethereum Ecosystem. In: 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME). Tenerife. Canary Islands. Spain; 2023. p. 1-7. doi: 10.1109/ICECCME57830.2023.10252520.

[19] Zkik K, Sebbar A, Fadi O, Mustapha O and Belhadi A. A Graph Neural Network Approach for Detecting Smart Contract Anomalies in Collaborative Economy Platforms Based on Blockchain Technology. In: 9th International Conference on Control, Decision and Information Technologies (CoDIT), Rome. Italy; 2023. p. 1285-1290. doi: 10.1109/CoDIT58514.2023.10284080.

[20] Jiang Z, Chen K and Zheng Z. Applying blockchain-based method to smart contract classification for CPS applications. Digital Communications and Networks. 2022; 8(6): 964-975. doi: 10.1016/j.dcan.2022.08.011.

[21] Reddy C M K, Chandrashekar R and Bajaj R. Smart Contracts and Anomaly Detection in SDN environment using Cloud-Edge Integration Model. In: International Conference on Emerging Research in Computational Science (ICERCS). Coimbatore. India; 2023. p. 1-6. doi: 10.1109/ICERCS57948.2023.10434076.

[22] Samreen N F and Alalfi M H. SmartScan: An approach to detect Denial of Service Vulnerability in Ethereum Smart Contracts. In: IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB). Madrid. Spain; 2021. p. 17-26. doi: 10.1109/WETSEB52558.2021.00010.

[23] Fadi O, Bahaj A, Zkik K, Ghazi A, Ghogho M and Boulmalf M. Smart Contract Anomaly Detection: The Contrastive Learning Paradigm. Journal of LATEX. 2024; 4(2): 631-639. doi: 10.2139/ssrn.4720935

[24] Duan L, Yang L, Liu C, Ni W and Wang W. A New Smart Contract Anomaly Detection Method by Fusing Opcode and Source Code Features for Blockchain Services. IEEE Transactions on Network and Service Management. 2023; 20(4): 4354-4368. doi: 10.1109/TNSM.2023.3278311.

[25] Ghaleb A, Rubin J and Pattabiraman K. eTainter: Detecting Gas-Related Vulnerabilities in Smart Contracts. International Symposium on Software Testing and Analysis (ISSTA). 2022; 5(7): 728-739. doi: 10.1145/3533767.3534378.

[26] Wang X, He J, Xie Z, Zhao G and Cheung S C, ContractGuard: Defend Ethereum Smart Contracts with Embedded Intrusion Detection. IEEE Transactions on Services Computing. 2020; 13(2): 314-328. doi: 10.1109/TSC.2019.2949561.

[27] Yang H, Gu X and Cui Z. CrossFuzz: Cross-contract fuzzing for smart contract vulnerability detection. Science of Computer Programming. 2024; 234:139-145. doi: 10.1016/j.scico.2023.
103076.

[28] Ndiaye M, Diallo T A and Konate K. ADEFGuard: Anomaly detection framework based on Ethereum smart contracts behaviours. Blockchain: Research and Applications. 2023; 4(3): 162-170. doi: 10.1016/j.bcra.2023.100148.

[29] Ashizawa N, Yanai N and Okamura S. Eth2Vec: Learning contract-wide code representations for vulnerability detection on Ethereum smart contracts. Blockchain: Research and Applications. 2022; 24(1): 139-148. doi: 10.1016/j.bcra.2022.100101.

[30] Eshghie M, Artho C and Gurov D. Dynamic Vulnerability Detection on Smart Contracts Using Machine Learning. Evaluation and Assessment in Software Engineering (EASE 2021). 2021; 43(12): 17-25. doi: 10.1145/3463274.3463348.

[31] Hu T, Liu X and Liu Y. Transaction-based classification and detection approach for Ethereum smart contract. Information Processing & Management. 2020; 58(4): 106-113. doi: 10.1016/j.ipm.2020.102462.

[32] Shen X, Jiang S and Zhang L. Mining Bytecode Features of Smart Contracts to Detect Ponzi Scheme on Blockchain. Computer Modeling in Engineering & Sciences. 2021; 45(1): 96-105. doi: 10.32604/cmes.2021.015736.

[33] Huang J, Zhou K, Xiong A and Li D. Smart Contract Vulnerability Detection Model Based on Multi-Task Learning. Sensors (Basel). 2022; 22(5): 1829-1836, doi: 10.3390/s22051829

[34] Zhang L, Chen W and Chen H. CBGRU: A Detection Method of Smart Contract Vulnerability Based on a Hybrid Model. Sensors. 2022; 22(9): 3577-3583.doi: 10.3390/s22093577