VulnFusion: Exploiting Multimodal Representations for Advanced Smart Contract Vulnerability Detection

Conference Paper · November 2024

DOI: 10.1109/BCCA62388.2024.10844483

Jiblal Upadhya
Middle Tennessee State University
10 PUBLICATIONS **21** CITATIONS
SEE PROFILE

Kritagya Upadhyay
Middle Tennessee State University
30 PUBLICATIONS **273** CITATIONS
SEE PROFILE

Samir Poudel
Middle Tennessee State University
16 PUBLICATIONS **46** CITATIONS
SEE PROFILE

Md Nahid Hasan
Middle Tennessee State University
7 PUBLICATIONS **15** CITATIONS
SEE PROFILE

2024 6th International Conference on Blockchain Computing and Applications (BCCA) | 979-8-3503-5153-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/BCCA62388.2024.10844483

VulnFusion: Exploiting Multimodal Representations for Advanced Smart Contract Vulnerability Detection

Jiblal Upadhya†, Arpan Sainju*, Kritagya Upadhyay*,
Samir Poudel*, Md Nahid Hasan†, Khem Poudel*, Jaishree Ranganathan*
Department of Computer Science*, Computational and Data Science†
Middle Tennessee State University
Murfreesboro, TN, 37132
Email: {ju2i, sp2ai, mh2ay}@mtmail.mtsu.edu,
{arpan.sainju, kritagya.upadhyay, khem.poudel, jaishree.ranganathan}@mtsu.edu

***Abstract*—In the fast-evolving domain of blockchain technology, smart contracts have become instrumental, facilitating billions of dollars in transactions daily. However, the increasing prevalence of smart contract vulnerabilities poses significant risks, potentially undermining trust in this innovative technology. To address these security challenges, our research introduces a novel approach for detecting and mitigating vulnerabilities through a comprehensive analysis of four principal modalities: Solidity source code, bytecode, opcode, and intermediate representations. This multimodal analysis employs a Transformer architecture, VulnFusion, which uniquely integrates these modalities to provide a deeper understanding of both the semantic and syntactical nuances of smart contracts. We propose an interleaving-based data fusion technique that outperforms traditional fusion methods such as concatenation and cross-attention. Experimental evaluations demonstrate that our interleaving-based fusion technique achieves an impressive average F1 score of 89% across source code, opcode, and intermediate representations, demonstrating the effectiveness of our approach.**

***Index Terms*—VulnFusion, Security, Blockchain, Smart Contract, Multimodal, Concatenation, Cross Attention, Interleaving-based Fusion**

I. Introduction

A smart contract is a self-executing digital agreement with terms directly written into code, running on a blockchain network [11]. It automatically enforces and executes actions when predefined conditions are met, ensuring efficiency and reducing the need for intermediaries. Smart contracts are immutable, meaning they cannot be altered once deployed, enhancing security and trust. They operate in a decentralized manner, providing transparency as all actions are recorded on the blockchain. This technology is secure and cost-effective, making it ideal for applications in finance, supply chain management, insurance, real estate, and more. For example, companies like IBM and Maersk [4] are using smart contracts to track goods in the supply chain. Each step of the product's journey is recorded on the blockchain, ensuring transparency and reducing the risk of fraud. Similarly, AXA has developed a smart contract-based insurance product called Fizzy [24]. It automatically compensates passengers for flight delays without needing to file a claim manually. Furthermore, some governments and organizations are experimenting with blockchain-based voting systems to ensure secure and transparent elections [27]. For example, Voatz has conducted pilot projects for voting in West Virginia, USA [25].

However, these smart contracts also pose notable security concerns. For instance, in 2020, the bZx protocol experienced multiple attacks due to flawed code, leading to a loss of nearly $1 million in just a few days [5]. Another example is the 2018 incident with the Bancor network, where a vulnerability in the smart contract was exploited, resulting in a loss of approximately $23.5 million [18]. Additionally, in 2021, the Poly Network hack resulted in a loss of over $600 million due to a flaw in the contract's cross-chain interoperability code [13]. Similarly, the Wormhole bridge hack in 2022 saw attackers exploit a vulnerability, causing a loss of around $325 million [14]. Notable incidents from 2023 include the Euler Protocol exploit [19], where a lending platform lost approximately $197 million due to a vulnerability in exchange rate mechanisms, and the Mutant Ape Planet NFT rug pull [23], which resulted in a $2.9 million loss due to deceptive practices. These incidents signify the critical need for rigorous security audits and robust coding practices to prevent vulnerabilities and safeguard assets in the rapidly evolving domain of smart contracts.

Below we present and briefly discuss the major problem motivations of our work:

1) Intricacy and Complexity of Smart Contracts: The intricacy of smart contracts emerges once they are deployed on the blockchain, as they become immutable and cannot be altered or corrected if bugs or security flaws are present. Detecting and promptly addressing these issues is crucial to ensuring their integrity and reliability. Moreover, the dynamic and decentralized characteristics of smart contracts increases their susceptibility to exploitation by malicious actors. Operating within a transparent and public blockchain environment exposes them to unauthorized access and manipulation. Their automated execution further exacerbates risks, as vulnerabilities can persist indefinitely without immediate mitigation. Additionally, interactions with diverse parties and external data sources broaden potential attack vectors, underscoring the need for robust security measures to effectively protect smart contracts from malicious activities.

2) Diverse Attack Vectors: Smart contracts are highly attractive to malicious actors due to their role in facilitating high-value transactions. These vectors encompass a wide range of vulnerabilities, including but not limited to reentrancy attacks, where malicious contracts repeatedly call back into vulnerable functions before prior executions complete; integer overflow/underflow, which manipulates arithmetic operations to unintended values; and front-running, where attackers exploit transaction ordering to gain advantage in decentralized exchanges [22]. Other vectors include denial-of-service attacks through gas limit manipulation, insecure random number generation leading to predictable outcomes, and the exploitation of external contract dependencies. Mitigating these risks requires rigorous testing, secure coding practices, and continuous monitoring to safeguard the integrity and functionality of smart contracts in blockchain ecosystems.

3) Limited Scalability of Manual Audits: Manual audits involve human experts meticulously reviewing code to identify vulnerabilities, which is a time-intensive process [30]. As the adoption of smart contracts grows rapidly across various industries, the demand for audits has surged, often overwhelming the capacity of audit firms to keep pace. Moreover, skilled auditors are scarce, leading to bottlenecks in the audit pipeline. The complexity of smart contract code, coupled with the need for thorough analysis to uncover nuanced vulnerabilities, further impedes scalability. This limitation underscores the necessity for automated tools and frameworks that can complement manual audits, enhancing efficiency and coverage while mitigating risks more effectively in the dynamic landscape of blockchain technologies.

4) Adversarial Nature of Smart Contract Ecosystem: Smart contracts operate in decentralized environments where transactions occur autonomously and irreversibly based on predefined code. The transparent and immutable nature of blockchain, while offering security benefits, also exposes contracts to scrutiny and potential attack. Adversaries leverage a range of tactics, including exploiting coding errors like reentrancy or overflow vulnerabilities, manipulating transaction ordering for profit (front-running), and deploying sophisticated phishing or social engineering schemes to deceive users into interacting with malicious contracts [30]. Additional vulnerabilities may arise as developers patch and update existing vulnerabilities, redeploying smart contracts across the network. This emphasizes the necessity for developers to continually enhance their defense systems. Nevertheless, a significant drawback of this approach is its reactive nature rather than proactive, which can leave systems vulnerable to emerging threats that exploit newly discovered weaknesses.

In current practices, vulnerabilities in smart contracts are predominantly identified using various methods such as manual code review, automated static analysis, symbolic execution, and machine learning approaches [1]–[3], [6], [16]. Manual code review relies on human expertise, which can be time-consuming and subjective, potentially missing subtle vulnerabilities. Automated tools like The Mythril, Slither, and SmartCheck [35] may yield false positives or negatives, leading to inaccuracies in identifying vulnerabilities. Symbolic execution requires significant computational resources and may struggle with complex smart contract logic. Machine learning methods often focus on single-mode analysis, limiting their ability to detect a wide range of smart contract vulnerabilities comprehensively, as they typically target specific known vulnerability patterns or signatures.

Our paper aims to develop a deep learning model based on multimodal analysis to detect vulnerabilities in smart contracts. Multimodal deep learning integrates multiple data sources and metadata, offering a comprehensive approach that enables the model to recognize complex attack patterns and uncover hidden correlations that are not discernible through single-mode analysis [17], [20], [28], [29], [31]. Unlike traditional single-mode approaches, multimodal analysis provides

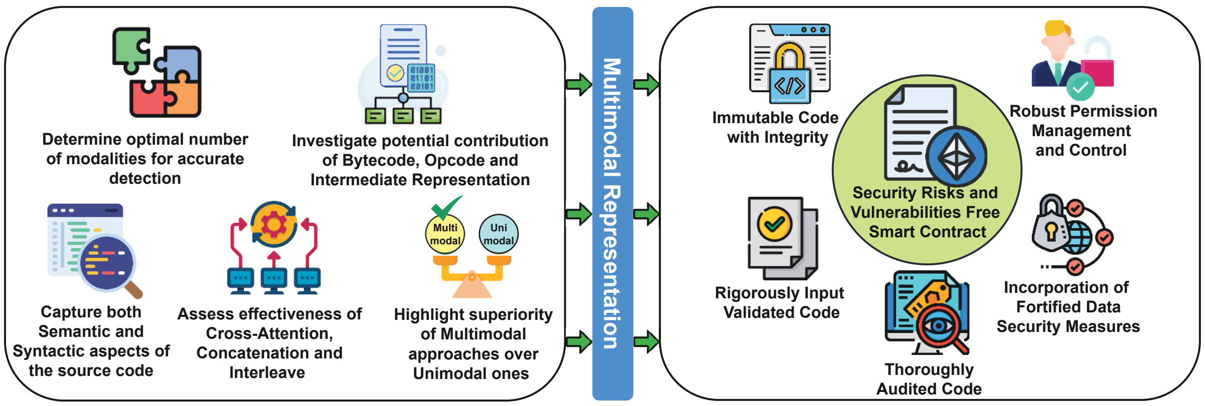

Determine optimal number of modalities for accurate detection
Investigate potential contribution of Bytecode, Opcode and Intermediate Representation
Capture both Semantic and Syntactic aspects of the source code
Assess effectiveness of Cross-Attention, Concatenation and Interleave
Multi modal
Uni modal
Highlight superiority of Multimodal approaches over Unimodal ones
Multimodal Representation
Immutable Code with Integrity
Robust Permission Management and Control
Security Risks and Vulnerabilities Free Smart Contract
Rigorously Input Validated Code
Incorporation of Fortified Data Security Measures
Thoroughly Audited Code

Fig. 1. Diagram illustrating the essential role of multimodal representation in detecting smart contracts vulnerable to security risks. By employing multimodal approaches, we can effectively identify and address these issues, potentially saving millions of dollars. The ultimate aim of this work is to ensure smart contracts are free from security risks and vulnerabilities through thorough multimodal detection.

a more complete understanding of both the semantic and syntactic contexts of smart contracts, enhancing the detection of underlying vulnerabilities [32]. Fig. 1 illustrates the critical role of multimodal representation in identifying smart contracts susceptible to security risks and vulnerabilities.

We proposed a novel multimodal Transformer network, VulnFusion, which integrates multiple sources of data to enhance the analysis and detection of vulnerabilities in smart contracts. VulnFusion leverages four distinct modalities of smart contracts: source code, bytecode, intermediate representation, and opcode. We leverage the BERT model to extract features from different combinations of modalities that are used as input in VulnFusion.

Furthermore, we investigate the potential significance of each combination of modalities by applying various fusion methods, aiming to identify the most crucial ones for detecting vulnerabilities in smart contracts using VulnFusion. Our investigation primarily focuses on three key data fusion methods: concatenation, cross-attention, and interleaving.

In summary, we make the following contributions:

- We propose a novel Multimodal Transformer Network, called VulnFusion, robust in detecting potential vulnerabilities in smart contracts, thereby enhancing the security and reliability of blockchain-based systems.
- We explored and experimented with all the 12 possible combinations of the multimodal architecture for the proposed system, showing that source code, opcode and intermediate representation combination fused with interleaving fusion technique is best in classifying and detecting the vulnerability.
- We have shown that our proposed multimodal approach was able to achieve an avg. F score of 89% outperforming all of the unimodal approaches.

II. Related Work

In this section, we explore existing research on detecting vulnerabilities in smart contracts. Li et al. [15] address security issues in smart contracts, such as reentrancy, timestamp dependency, tx.origin, and integer overflow vulnerabilities. Their method employs multimodal feature fusion, integrating static and dynamic features through deep learning models like Graph Convolutional Networks (GCNs) and bidirectional Long Short-Term Memory networks (bi-LSTMs). Experimental results show high detection accuracies: 85.73% for reentrancy, 85.41% for timestamp dependency, 83.58% for tx.origin, and 90.96% for integer overflow vulnerabilities. However, existing research typically exhibits low detection accuracy and a narrow focus on single vulnerability types. Nevertheless, while their method improves detection performance, it does not fully address the trade-offs between computational efficiency and accuracy.

Similarly, Deng et al. [7] address the security challenges of smart contracts in blockchain applications, emphasizing vulnerabilities such as arithmetic, reentrancy, transaction order dependence, and Ethereum locking. Their method leverages deep learning and multimodal decision fusion to integrate source code, operation code, and control-flow modes, achieving high detection accuracies across various vulnerabilities. Nonetheless, their approach, while robust, does not extensively address the potential inefficiencies in handling extremely large datasets. Additionally, although their method shows high accuracy, it still faces challenges in scalability and real-time detection.

Likewise, Yuan et al. [34] proposed an entropy embedding technique to address vulnerability concerns. However, their study did not sufficiently address issues related to false positives and negatives, nor did it adequately consider the trade-offs between accuracy and computational efficiency, particularly regarding the modality gap. Similarly, Duy et al. [9] also took a limited approach by using only three modal representations—source code, opcode, and intermediate representation—without exploring a variety of fusion methods. Their reliance solely on concatenation for data fusion lacks diversity, necessitating further investigation into the rationale behind this choice and its comparative effectiveness. Tang et al. [26] conducted a comprehensive investigation, particularly in comparing standalone unimodal approaches. However, their study was constrained by the use of only three modalities and did not incorporate a multimodal framework for a more thorough evaluation. Yang et al. [33] notably did not integrate multimodal methods, relying exclusively on word2vec embeddings. Compared to BERT, word2vec offers limited contextual understanding as it only considers surrounding words in a single direction. Additionally, their categorization of vulnerabilities was limited to just six types, excluding other commonly recognized categories. Khodadadi et al. [12] investigated methods for detecting smart contract vulnerabilities using various input representations, including deep learning techniques like BiGRU and word embedding methodologies. Their study found that the HyMo model performed well in detecting vulnerabilities compared to contemporary models. However, the authors limited their analysis to only two modalities: source code and opcode. Furthermore, although their model's accuracy exceeded that of existing models, it still fell slightly short of expectations.

III. Methodology

This section introduces a novel multimodal framework called VulnFusion.

A. Data Collection

We obtain the dataset of 106,000 smart contracts audited by Slither from the Hugging Face database [21] . This dataset includes the source code and corresponding bytecode of smart contracts written in the Solidity programming language and sourced from Etherscan [10]. We generated Opcode: Assembly code obtained from bytecode using an Opcode-Bytecode converter tool from Etherscan and Intermediate Representation: obtained using the Slither tool for our research as two other modalities. Additionally, it provides a classification of vulnerabilities based on the Slither static analysis framework. Each data entry includes the following

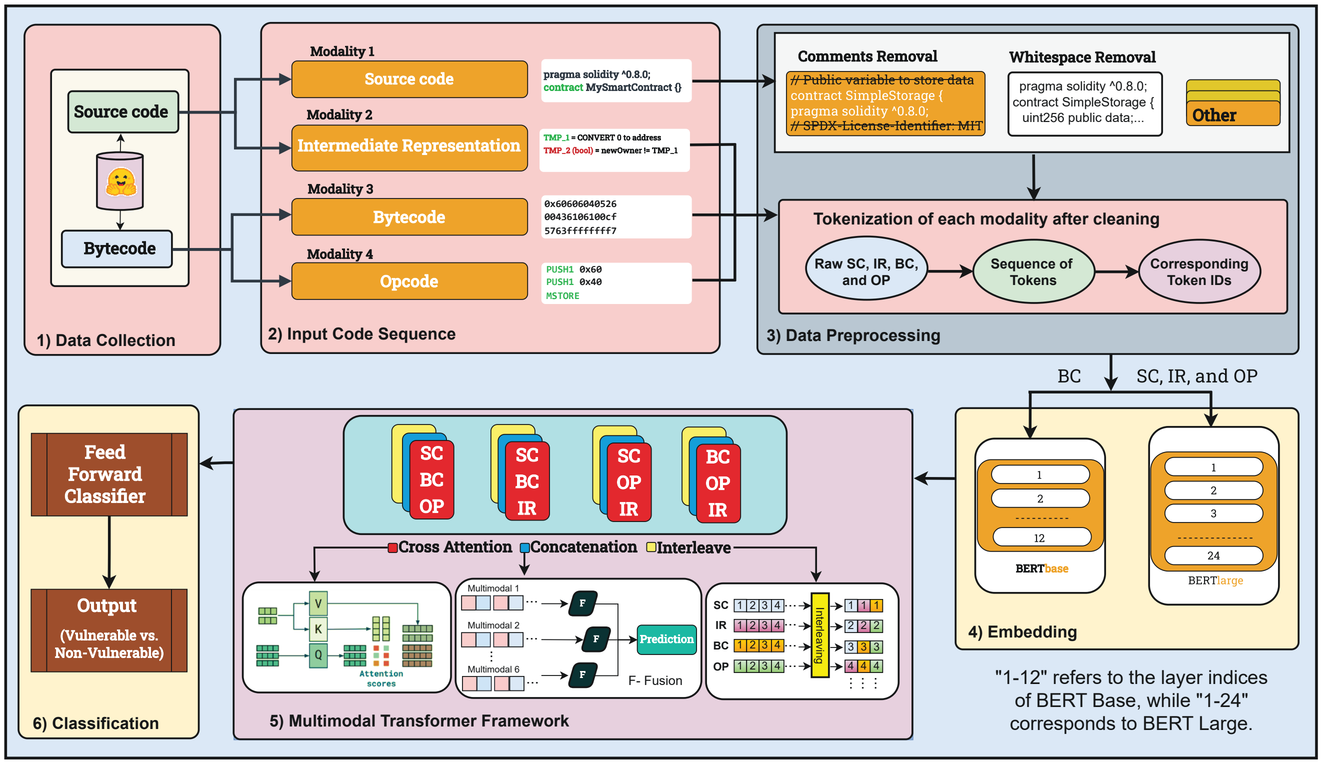


Fig. 2. This diagram illustrates a system architecture for a proposed multimodal Transformer designed to classify security risks and vulnerabilities in deployed smart contracts. The process involves extracting smart code and bytecode, processing multiple code representations using a transformer-based framework with BERT embeddings for feature extraction, and finally using a classifier to predict a smart contract's vulnerability status. Our Multimodal Transformer Framework incorporates three different fusion methods: Concatenation, Cross Attention, and Interleaving.

attributes: address, source code, and bytecode. In this study, we focus on binary classification to determine the presence or absence of vulnerabilities in the smart contracts.

B. Model Architecture

The overall model's architecture is illustrated in Fig. 2.

The input code sequences from each of the four modalities are shown in the step 2 of Fig. 2.

In step 3, we first preprocess the source code by standard preprocessing techniques such as removing white spaces and comments. Next, we employ the tokenization step in which the words in the source code, intermediate representation and opcode are converted into series of tokens. However, we cannot use the standard tokenization method for bytecode because the instructions are not separated like words in source code. Hence, for bytecode, tokens are extracted by simply tokenizing the hexadecimal values.

In step 4, we performed embedding transformation to convert tokens into dense vectors within a continuous vector space, leveraging pre-trained models such as BERT (Bidirectional Encoder Representations from Transformers) [8]. BERT is a widely-used language representation model trained on large corpora of text. The key innovation of BERT is its ability to generate contextualized word representations by jointly conditioning on both the left and right context of a word. This bidirectional training approach allows BERT to better capture the semantic meaning and relationships between words, which is crucial for understanding natural language. BERT comes in different sizes, with 'BERT-base' representing a smaller version and 'BERT-large' a larger and more complex one. The source code, intermediate representation, and opcode were embedded using BERT-large model (340M parameters) whereas, the bytecode was embedded using BERT-base model (110M parameters). In Step 5, we introduce our Multimodal Transformer Framework, wherein we implemented three distinct data fusion methods: cross-attention, concatenation, and interleaving. In Step 6, the integrated multimodal data is subsequently input into a feed-forward classification model to generate the final prediction. This model leverages the combined features extracted from multiple modalities, ensuring a comprehensive analysis. By utilizing a feedforward architecture, the model processes the data through multiple layers, optimizing the classification performance through a series of weighted transformations and activations.

C. Multimodal Transfomer Framework

The architecture initiates with unimodal feature extraction from various sources: source code (SC), bytecode (BC), opcode (OP), and intermediate representation (IR). Each of these modalities is processed into uniform feature vectors. To achieve an integrated multimodal representation, we employ three distinct feature fusion methods: cross-attention, concatenation, and interleaving. These method facilitate the seamless integration of multimodal data into a cohesive representation. Fig. 3 outlines the multimodal feature fusion network, illustrating how the input feature vectors are synthesized and subsequently utilized for the classification task.

1) Concatenation-Data Fusion Mechanism: Let's denote the output vectors of the BERT model for source code as $BERT_{SC}$, bytecode as $BERT_{BC}$, and opcode as $BERT_{OP}$, with each having dimensions $768 \times L$. The concatenation operation for three modalities can be represented as:

$$\text{Combined} = \text{Concatenate}(BERT_{SC}, BERT_{BC}, BERT_{OP}) \quad (1)$$

$$\text{Combined} \in \mathbb{R}^{2304 \times L} \quad (2)$$

The sequence of $L \times 2304$ vectors is flattened into a single vector, resulting in a dimension of $L \cdot 2304$.

This concatenated vector is subsequently processed through a series of dense blocks, or feed-forward networks, designed to reduce the dimensionality of the input vector. These transformations can be represented as follows:

- Flatten: $\mathbb{R}^{L \times 2304} \to \mathbb{R}^{L \cdot 2304}$
- Dense Block 1: $\mathbb{R}^{L \cdot 2304} \to \mathbb{R}^{1024}$
- Dense Block 2: $\mathbb{R}^{1024} \to \mathbb{R}^{512}$
- Dense Block 3: $\mathbb{R}^{512} \to \mathbb{R}^{256}$
- Dense Block 4: $\mathbb{R}^{256} \to \mathbb{R}^{128}$
- Dense Block 5: $\mathbb{R}^{128} \to \mathbb{R}^{64}$
- Dense Block 6: $\mathbb{R}^{64} \to \mathbb{R}^{32}$

The output of the last dense block is passed to a final dense layer with a sigmoid activation function, producing a scalar output representing the probability of vulnerability:

$$\text{Output} = \sigma(\text{Dense}(\mathbb{R}^{32} \to \mathbb{R})) \quad (3)$$

where σ represents the sigmoid activation function.

Similarly, we use the above concatenation data fusion mechanism to concatenate (SC, BC, and IR), (SC, OP, and IR), and (BC, OP, and IR).

2) Cross-Attention Data-Fusion Mechanism: In Cross-Attention Data-Fusion mechanism, we apply the `CrossAttention` layer to the output embeddings of the BERT models for every trio of modalities. The combination includes (SC, BC, and OP), (SC, BC, and IR), (SC, OP, and IR), and (BC, OP, and IR).

Given the number of heads (`num_heads`), and the dimensions for keys (`key_dim`) and values (`value_dim`), the dimensionality of the attention outputs for each head will be $L \times$ key_dim. For `num_heads` = 5, `key_dim` = 64, and `value_dim` = 64, each head produces an output of dimension $L \times 64$. These are concatenated to give a tensor of dimensions $L \times 320$.

However, due to the residual connections and layer normalization, the final output dimensionality remains $L \times 768$, consistent with the input embeddings.

The embeddings are flattened, transforming the tensor into a single-dimensional vector, maintaining a size of $L \times 768$. This vector is processed through a series of dense blocks (feed forward network), each reducing the dimensionality as follows:

- Flatten: $\mathbb{R}^{L \times 768} \to \mathbb{R}^{L \cdot 768}$
- Dense Block 1: $\mathbb{R}^{L \cdot 768} \to \mathbb{R}^{512}$
- Dense Block 2: $\mathbb{R}^{512} \to \mathbb{R}^{256}$
- Dense Block 3: $\mathbb{R}^{256} \to \mathbb{R}^{128}$
- Dense Block 4: $\mathbb{R}^{128} \to \mathbb{R}^{32}$
- Dense Block 5: $\mathbb{R}^{32} \to \mathbb{R}^{16}$
- Dense Output: $\mathbb{R}^{16} \to \mathbb{R}$ with a sigmoid activation function to obtain a probability in the range [0,1].

The final output is a scalar value representing the probability of vulnerability in the smart contract.

3) Interleaving Data-Fusion Mechanism: In the interleaving fusion technique, we integrate multiple types of data (modalities) by interweaving their feature to create a unified representation.

For each combination of three modalities, we create an interleaved embedding $\mathbf{E}$ using the following equation:

$$\mathbf{E}_{i,j} = \begin{cases} \mathbf{A}_{i,k} & \text{if } j = 3k \\ \mathbf{B}_{i,k} & \text{if } j = 3k+1 \\ \mathbf{C}_{i,k} & \text{if } j = 3k+2 \end{cases} \quad (4)$$

Where:

- $i \in [0, 32000)$ (sample index)
- $j \in [0, 2304)$ (position in the interleaved embedding)
- $k \in [0, 768)$
- $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ represent the three chosen modalities

We generate four different interleaved embeddings, one for each combination of three modalities (SC, OP, and BC), (SC, OP, and IR), (SC, IR, and BC), (OP, IR, and BC)

After interleaving, the embedding is flattened and processed through several layers to reduce its size:

- Flatten: $\mathbb{R}^{32000 \times 2304} \to \mathbb{R}^{75456000}$
- Dense Layer 1: $\mathbb{R}^{75456000} \to \mathbb{R}^{512}$
- Dense Layer 2: $\mathbb{R}^{512} \to \mathbb{R}^{256}$
- Dense Layer 3: $\mathbb{R}^{256} \to \mathbb{R}^{128}$
- Dense Layer 4: $\mathbb{R}^{128} \to \mathbb{R}^{32}$
- Dense Layer 5: $\mathbb{R}^{32} \to \mathbb{R}^{16}$

Finally, the output is passed through a sigmoid activation function to produce a probability of vulnerability.

D. Unimodal Training

We used the Big-Multilabel dataset [21], which includes smart contract addresses and Slither labels. A label of [4] indicates a non-vulnerable contract, while labels [0, 1, 2, 3, 5] indicate different vulnerabilities. For binary classification, contracts with any vulnerabilities were labeled as "Vulnerable", and those without were labeled as "Non-Vulnerable".

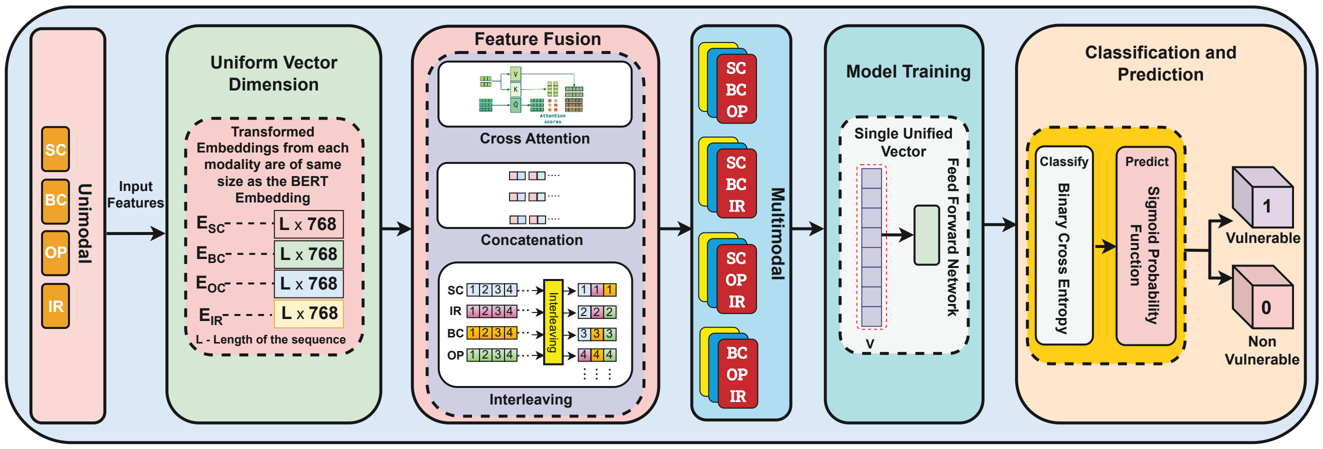


Fig. 3. Our Multimodal feature fusion framework aims to detect vulnerabilities in smart contracts. The process involves transforming input features from each unimodal source and their combinations into embeddings, which then pass through multimodal feature fusion blocks. These blocks consolidate the embeddings into a unified vector for prediction. The steps include: (a) extracting modality features, (b) ensuring uniform transformed vector dimensions, (c) conducting feature fusion (involving concatenation, cross attention, and interleaving), (d) training the model, (e) classifying features, and (f) making predictions.

TABLE I
CLASS DISTRIBUTION

Dataset	Vulnerable	Non-Vulnerable
Train	6,640	17,668
Test	830	2209
Validation	830	2209

Table I provides a comprehensive overview of the class distribution across distinct datasets employed in the study. The datasets, categorized into Train, Test, and Validation partitions, adhere to a stratified sampling strategy, ensuring consistent proportions of instances labeled as ”Vulnerable” and ”Non-Vulnerable” across all subsets.

TABLE II
UNIMODAL PERFORMANCE COMPARISON

Modality	Class	Precision	Recall	F	Avg. F
BC	Non-Vuln.	0.86	0.89	0.88	0.76
	Vuln.	0.68	0.62	0.65	
IR	Non-Vuln.	0.91	0.87	0.89	0.80
	Vuln.	0.68	0.76	0.72	
OP	**Non-Vuln.**	**0.92**	**0.90**	**0.91**	**0.84**
	Vuln.	**0.75**	**0.80**	**0.78**	
SC	Non-Vuln.	0.84	0.91	0.88	0.74
	Vuln.	0.70	0.54	0.61	

All the modalities were trained on the feed-forward network with the same hyperparameters. The loss function was Binary Crossentropy and the Adam Optimizer with 1e-4 learning rate was employed. Our results, illustrated in Table II, demonstrate various performance metrics, offering a comprehensive comparison of each unimodal representation when trained independently. We observed that the opcode modality achieved the highest performance, with an average F1 score of 0.84.

E. Multimodal Training

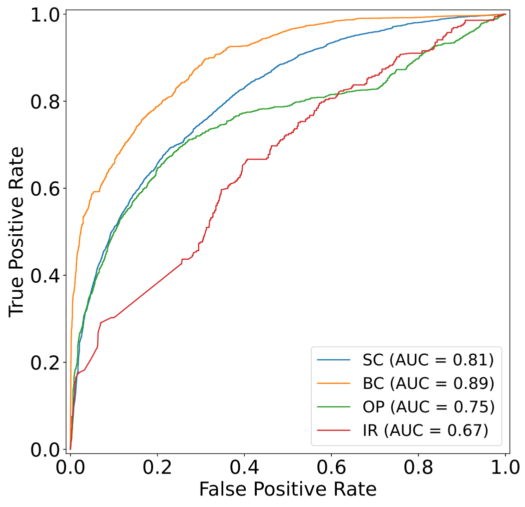
In a multimodal context, we generate 12 fused datasets by applying cross-attention, concatenation, and interleaving concatenation data fusion methods to each possible combination of three modality trios, as illustrated below:

- **SC-OP-BC-concat**: Concatenation on SC, OP and BC
- **SC-OP-BC-attn**: Cross-attention on SC, OP and BC
- **SC-OP-BC-inter**: Interleave on SC, OP and BC
- **SC-OP-IR-concat**: Concatenation on SC, OP and IR
- **SC-OP-IR-attn**: Cross-attention on SC, OP and IR
- **SC-OP-IR-inter**: Interleave on SC, OP and IR
- **SC-IR-BC-attn**: Cross-attention on SC, IR and BC
- **SC-IR-BC-concat**: Concatenation on SC, IR and BC
- **SC-IR-BC-inter**: Interleave on SC, IR and BC
- **OP-IR-BC-concat**: Concatenation on OP, IR and BC
- **OP-IR-BC-attn**: Cross-attention on OP, IR and BC
- **OP-IR-BC-inter**: Interleave on OP, IR and BC

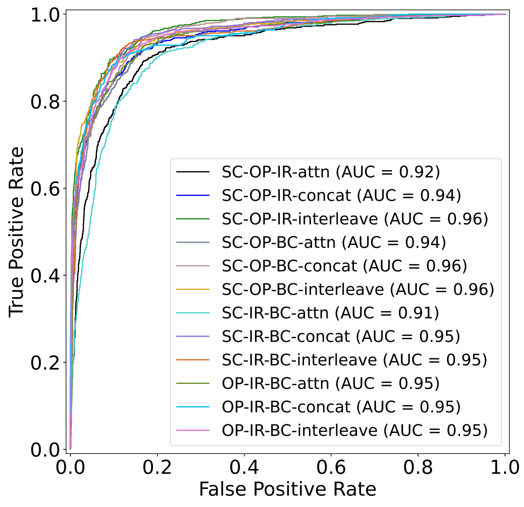
Similar to the unimodal training, All the multi-modalities were trained on the feed-forward network with the same hyperparameters. We used Binary Crossentropy loss and Adam Optimizer with a learning rate of 1e-4.

IV. RESULTS AND DISCUSSION

The study evaluated four different modalities for smart contract representation: SC, BC, OP, and IR. Each modality was assessed using three fusion algorithms: concatenation, cross-attention, and interleaving. The performance metrics include precision, recall, and F1-score as seen from the Table III. The combination of source code, opcode, and intermediate representations achieved the highest average F1-score of 0.89. The interleaving fusion mechanism demonstrates high effectiveness in combining information from multiple modalities. The average F1 scores across different combinations based on interleaving fusion are consistently high, indicating robust performance in detecting vulnerabilities. Unlike simple concatenation, which just appends one type of data to another, or cross-attention, which aligns elements from different modalities based on their relevance to each



(a)



(b)

Fig. 4. Receiver Operating Curve for the performance of the smart contract vulnerability detection (a) Unimodal Analysis (b) Multimodal Analysis for all 12 possible combinations of the proposed multimodal Framework. This AUC score for all the multimodality combinations showed that our VulnFusion has a strong discriminatory power for vulnerable detection.

TABLE III
MULTIMODAL PERFORMANCE COMPARISON-THREE MODALITY

Multimodality	Class	Precision	Recall	F	Avg. F
OP-IR-BC-attn	Non-Vuln.	0.93	0.93	0.93	0.87
	Vuln.	0.81	0.81	0.81	
OP-IR-BC-concat	Non-Vuln.	0.91	0.95	0.93	0.87
	Vuln.	0.72	0.80	0.75	
OP-IR-BC-inter	**Non-Vuln.**	**0.94**	**0.91**	**0.93**	**0.87**
	Vuln.	**0.78**	**0.85**	**0.82**	
SC-IR-BC-attn	Non-Vuln.	0.92	0.90	0.91	0.83
	Vuln.	0.74	0.79	0.76	
SC-IR-BC-concat	Non-Vuln.	0.93	0.93	0.93	0.87
	Vuln.	0.81	0.82	0.81	
SC-IR-BC-inter	**Non-Vuln.**	**0.91**	**0.96**	**0.93**	**0.87**
	Vuln.	**0.88**	**0.74**	**0.80**	
SC-OP-BC-attn	Non-Vuln.	0.94	0.89	0.92	0.85
	Vuln.	0.76	0.85	0.80	
SC-OP-BC-concat	Non-Vuln.	0.94	0.90	0.92	0.86
	Vuln.	0.76	0.85	0.80	
SC-OP-BC-inter	**Non-Vuln.**	**0.92**	**0.96**	**0.94**	**0.88**
	Vuln.	**0.88**	**0.78**	**0.82**	
SC-OP-IR-attn	Non-Vuln.	0.92	0.89	0.91	0.84
	Vuln.	0.73	0.80	0.77	
SC-OP-IR-concat	Non-Vuln.	0.90	0.96	0.93	0.86
	Vuln.	0.88	0.71	0.79	
SC-OP-IR-inter	**Non-Vuln.**	**0.94**	**0.94**	**0.94**	**0.89**
	Vuln.	**0.83**	**0.84**	**0.84**	

other, interleaving fusion strategically blends elements from different data sources by placing similar features from different modalities together. Through experimental evaluation, we demonstrated that the proposed interleaving-based fusion approach effectively captures intricate relationships and dependencies between modalities, enabling the model to better understand and learn from the integrated information. As shown in Fig. 4(a) and 4(b), all the multimodal approaches exhibit a high area under the ROC curve (AUC), with values exceeding 0.9, significantly higher than those of the unimodal approaches, further justifying our findings.

Furthermore, the experiments demonstrate that models incorporating multiple modalities consistently outperform single-modality approaches across various metrics. These results strongly indicate that combining different modalities holds significant potential for enhancing the detection of vulnerabilities in smart contracts.

V. CONCLUSION AND FUTURE WORKS

We presented VulnFusion, a novel Multimodal Transformer Network designed to detect and mitigate vulnerabilities in smart contracts. Our research explored all 12 possible combinations of these modalities within the proposed multimodal architecture. This method demonstrated significant improvements over traditional fusion methods such as concatenation and cross-attention. Experimental evaluations showcased the efficacy of VulnFusion, with the interleaving-based fusion technique achieving an impressive average F1 score of 89 %. In conclusion, the VulnFusion framework represents a robust and effective solution for enhancing the security and reliability of blockchain-based systems. We identified that our proposed interleaving-based fusion technique performed best. Future work will focus on further refining this model, exploring additional modalities, and expanding the application of VulnFusion to other domains within blockchain technology. Likewise, we will examine multi-class classification to detect particular types of vulnerabilities.

References

[1] Mauro Argañaraz, Mario Berón, Maria João Pereira, and Pedro Rangel Henriques. Detection of vulnerabilities in smart contracts specifications in ethereum platforms. In *9th Symposium on Languages, Applications and Technologies (SLATE 2020)*, volume 83, pages 1–16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020.

[2] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In *Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings 6*, pages 164–186. Springer, 2017.

[3] Massimo Bartoletti and Livio Pompianu. An empirical analysis of smart contracts: platforms, applications, and design patterns. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pages 494–509. Springer, 2017.

[4] Sanja Bauk. Blockchain conceptual framework in shipping and port management. In *Maritime Transport Conference*, number 9, 2022.

[5] Hongsong Chen, Xietian Luo, Lei Shi, Yongrui Cao, and Yongpeng Zhang. Security challenges and defense approaches for blockchain-based services from a full-stack architecture perspective. *Blockchain: Research and Applications*, 4(3):100135, 2023.

[6] Kevin Delmolino, Mitchell Arnett, Ahmed Kosba, Andrew Miller, and Elaine Shi. Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers 20*, pages 79–94. Springer, 2016.

[7] Weichu Deng, Huanchun Wei, Teng Huang, Cong Cao, Yun Peng, and Xuan Hu. Smart contract vulnerability detection based on deep learning and multimodal decision fusion. *Sensors*, 23(16):7246, 2023.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Phan The Duy, Nghi Hoang Khoa, Nguyen Huu Quyen, Le Cong Trinh, Vu Trung Kien, Trinh Minh Hoang, and Van-Hau Pham. Vulnsense: Efficient vulnerability detection in ethereum smart contracts by multimodal learning with graph neural network and language model. *arXiv preprint arXiv:2309.08474*, 2023.

[10] A Etherscan. Etherscan—the ethereum blockchain explorer. *https://etherscan. io/*, 2021.

[11] Shafaq Naheed Khan, Faiza Loukil, Chirine Ghedira-Guegan, Elhadj Benkhelifa, and Anoud Bani-Hani. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-peer Networking and Applications*, 14:2901–2925, 2021.

[12] Mohammad Khodadadi and Jafar Tahmoresnezhad. Hymo: Vulnerability detection in smart contracts using a novel multi-modal hybrid model. *arXiv preprint arXiv:2304.13103*, 2023.

[13] Vladimir Kustov, Grokhotov Aleksey, Beksaev Nikolay, Selanteva Ekaterina, and Renjith V Ravi. Three sources of blockchain technology vulnerabilities-how to deal with them? In *2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pages 1–8. IEEE, 2022.

[14] Sung-Shine Lee, Alexandr Murashkin, Martin Derka, and Jan Gorzny. Sok: Not quite water under the bridge: Review of cross-chain bridge hacks. In *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–14. IEEE, 2023.

[15] Jinggang Li, Gehao Lu, Yulian Gao, and Feng Gao. A smart contract vulnerability detection method based on multimodal feature fusion and deep learning. *Mathematics*, 11(23):4823, 2023.

[16] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 254–269, 2016.

[17] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[18] Kris Oosthoek. Flash crash for cash: Cyber threats in decentralized finance. *arXiv preprint arXiv:2106.10740*, 2021.

[19] Bahareh Parhizkari, Antonio Ken Iannillo, Christof Ferreira Torres, Sebastian Banescu, Joseph Xu, and Radu State. Timely identification of victim addresses in defi attacks. In *European Symposium on Research in Computer Security*, pages 394–410. Springer, 2023.

[20] Peng Qian, Zhenguang Liu, Qinming He, Butian Huang, Duanzheng Tian, and Xun Wang. Smart contract vulnerability detection technique: A survey. *arXiv preprint arXiv:2209.05872*, 2022.

[21] Martina Rossini. Slither audited smart contracts dataset. 2022.

[22] Sarwar Sayeed, Hector Marco-Gisbert, and Tom Caira. Smart contract: Attacks and protections. *Ieee Access*, 8:24416–24427, 2020.

[23] Jason Scharfman. Non-fungible token (nft) fraud, hacks, and controversies. In *The Cryptocurrency and Digital Asset Fraud Casebook, Volume II: DeFi, NFTs, DAOs, Meme Coins, and Other Digital Asset Hacks*, pages 307–325. Springer, 2024.

[24] Soraya Sedkaoui and Nawal Chicha. Blockchain-based smart contract technology application in the insurance industry: The case of "fizzy". *MJBS,(2)*, 2021.

[25] Michael A Specter, James Koppel, and Daniel Weitzner. The ballot is busted before the blockchain: A security analysis of voatz, the first internet voting application used in {US}. federal elections. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1535–1553, 2020.

[26] Xueyan Tang, Yuying Du, Alan Lai, Ze Zhang, and Lingzhi Shi. Deep learning-based solution for smart contract vulnerabilities detection. *Scientific Reports*, 13(1):20106, 2023.

[27] Alvin Julian Tjahajadi, Theresia Herlina Rochadiani, et al. Ethereum blockchain and smart contract modelling for presidential e-voting system in indonesia. *Int. J. Technol. Eng. Stud.*, 4(2):50–56, 2018.

[28] Jiblal Upadhya, Khem Poudel, and Jaishree Ranganathan. Advancing medical image diagnostics through multi-modal fusion: Insights from mimic chest x-ray dataset analysis. In *2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*, pages 1–8, 2024.

[29] Jiblal Upadhya, Khem Poudel, and Jaishree Ranganathan. A comprehensive approach to early detection of workplace stress with multimodal analysis and explainable ai. In *Proceedings of the 2024 Computers and People Research Conference*, SIGMIS-CPR '24, New York, NY, USA, 2024. Association for Computing Machinery.

[30] Kritagya Upadhyay, Ram Dantu, Yanyan He, Syed Badruddoja, and Abiola Salau. Auditing metaverse requires multimodal deep learning. In *2022 IEEE 4th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*, pages 39–46. IEEE, 2022.

[31] Kritagya Upadhyay, Ram Dantu, Yanyan He, Abiola Salau, and Syed Badruddoja. Paradigm shift from paper contracts to smart contracts. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 261–268. IEEE, 2021.

[32] Zhen Yang, Jacky Keung, Xiao Yu, Xiaodong Gu, Zhengyuan Wei, Xiaoxue Ma, and Miao Zhang. A multi-modal transformer-based code summarization approach for smart contracts. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, pages 1–12. IEEE, 2021.

[33] Zhongju Yang and Weixing Zhu. Improvement and optimization of vulnerability detection methods for ethernet smart contracts. *IEEE Access*, 2023.

[34] Dawei Yuan, Xiaohui Wang, Yao Li, and Tao Zhang. Optimizing smart contract vulnerability detection via multi-modality code and entropy embedding. *Journal of Systems and Software*, 202:111699, 2023.

[35] William Zhang, Sebastian Banescu, Leonardo Pasos, Steven Stewart, and Vijay Ganesh. Mpro: Combining static and symbolic analysis for scalable testing of smart contract. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, pages 456–462. IEEE, 2019.