TokenScout: Early Detection of Ethereum Scam Tokens via Temporal Graph Learning

Cong Wu
Nanyang Technological University
Singapore
cong.wu@ntu.edu.sg

Jing Chen
Wuhan University
Wuhan, China
chenjing@whu.edu.cn

Ziming Zhao
Northeastern University
Boston, USA
z.zhao@northeastern.edu

Kun He
Wuhan University
Wuhan, China
hekun@whu.edu.cn

Guowen Xu
University of Electronic Science and Technology of China
Chengdu, China
guowen.xu@uestc.edu.cn

Yueming Wu
Nanyang Technological University
Singapore
yueming.wu@ntu.edu.sg

Haijun Wang
Xi'an Jiaotong University
Xian, China
haijunwang@xjtu.edu.cn

Hongwei Li
University of Electronic Science and Technology of China
Chengdu, China
hongweili@uestc.edu.cn

Yang Liu
Nanyang Technological University
Singapore
yangliu@ntu.edu.sg

Yang Xiang
Swinburne University of Technology
Melbourne, Australia
yxiang@swin.edu.au

Abstract

Decentralized finance has experienced phenomenal growth, revolutionizing the landscape of financial transactions and asset management via blockchain. Yet, this swift growth brings with it substantial challenges, notably the surge in scam tokens, imposing significant security threats on cryptocurrency investments and trading. Existing detection methods of scam token, primarily relying on analyzing contract codes or transaction patterns, struggle to catch increasingly sophisticated tactics employed by scammers. For example, contract-based analysis are unable to identify scams lacking overt malicious code, e.g., most rugpulls, while transaction-based methods generally lack the foresight to early-detect potential risks.

In this paper, we present TokenScout, the first temporal graph neural network-based framework for scam token early detection. TokenScout formulates token transfer data as a dynamic temporal attributed multigraph and leverages the temporal graph learning model to learn graph representations. It also builds a graph representation refining model based on contrastive learning to learn a more discriminative representation space for risk identification. We evaluated TokenScout using a comprehensive dataset of 214,084 standard ERC20 tokens from 2015 to February 2023. TokenScout achieves a balanced accuracy of 98.41%. Additionally, from March to May 2023, deploying TokenScout on Ethereum effectively identified 706 rugpulls, 174 honeypots, and 90 Ponzi schemes, thereby alerting to potential risks exceeding $240 million.

CCS Concepts

• **Security and privacy** → *Web application security*; **Distributed systems security**.

Keywords

Decentralized Finance; Ethereum; scam tokens; graph neural network

ACM Reference Format:
Cong Wu, Jing Chen, Ziming Zhao, Kun He, Guowen Xu, Yueming Wu, Haijun Wang, Hongwei Li, Yang Liu, and Yang Xiang. 2024. TokenScout: Early Detection of Ethereum Scam Tokens via Temporal Graph Learning. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3690234

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690234

1 Introduction

Decentralized Finance (DeFi) has revolutionized the world of finance, facilitating open and borderless services on blockchain platforms like Ethereum [10] and Binance [3]. By April 2023, the DeFi ecosystem had amassed approximately $83.3 billion in locked assets [1]. At the heart of DeFi's success are *tokens* equipped with programmable smart contracts [35, 43], exemplified by USDT [21], USDC [25], and SHIB [19]. These tokens serve as the foundational

units of digital assets, empowering a wide range of decentralized financial services, including trading, lending, and borrowing, while obviating the need for traditional intermediaries.

In Ethereum, Ethereum Request for Comments 20 (ERC20) is the predominant standard for Initial Coin Offerings (ICOs), token sales, and utility tokens across diverse decentralized applications. Unfortunately, the ERC20 standard is not immune to abuse by scam and fraudulent tokens, which lure investors with promises of enticing returns. Once a critical mass of funds is amassed, malicious actors manipulate the market or withdraw liquidity, rendering the tokens valueless and inflicting financial losses upon unsuspecting investors [8, 18, 28, 29, 31, 33, 37, 50, 63, 67]. In 2022, over 117k scam tokens resulted in financial losses amounting to billions of US dollars [14].

In light of the substantial harm inflicted by scam tokens, contract-based and transaction-based solutions were proposed for their detection. Contract-based methods involve the analysis of token contract code to uncover malicious patterns or functionalities that may indicate a scam [30, 32, 40, 47–49, 60, 76, 78]. They utilize static and dynamic program analysis to detect malicious code patterns that frequently appeared in scam tokens. While effective for detecting Ponzi tokens, which exhibit malicious patterns of redistributing new investors' funds to earlier participants, these solutions fall short in identifying other prevalent scam tokens. For instance, rugpull tokens, constituting a significant portion of scam tokens, usually appear benign in their code but engage in malicious transactional activities [27, 69].

On the contrary, transaction-based methods typically leverage deep learning approaches to scrutinize the transactional characteristics of tokens [50, 67]. These methods concentrate on statistical features from token transactions, encompassing transfer rates, volume, and sentiment. Nevertheless, they fall short in capturing the intricate and evolving behavioral dynamics of token transaction flow. For example, these methods overlook the timing of transactions by scammers, fail to detect coordinated manipulative activities using multiple addresses, or miss recurrent patterns that indicate preparatory scam activities. Therefore, they can only flag already-occurred scam tokens, lacking the foresight to proactively predict upcoming threats and safeguard potential investor losses.

In this paper, we advance transaction-based solutions to enhance the real-time risk monitoring and early detection of scam tokens by introducing a novel Graph Neural Network (GNN) based approach. While it seems natural to apply GNNs to token transfer graphs, there are some unique challenges: (1) *representation learning in token transfer graph.* Applying GNN to token transfer graphs involves the challenge of robustly learning from extensive node and edge attributes and token transfer activities. The main difficulty is in encoding the attributes of nodes and edges into GNN models to effectively capture the complexities of token transfers; (2) *encoding temporal dynamics of token transfer graphs.* It is challenging to encode the temporal dynamics of token transfer graph using GNNs. The inherent structure of most GNNs and their variants typically conceptualizes the graph as static, or at best, operates on fixed-timeframe graph snapshots, assuming uniform time intervals between events [51, 75]. However, token transfer activities occur sporadically and unpredictably, making it difficult to adapt these models. Furthermore, it is difficult to capture long-term dependencies in rapidly evolving and irregular token flow [51].

Additionally, identifying scam tokens before fraud occurs is challenging because it involves detecting subtle signs within token transactions that could indicate a scam is imminent. These early warning signals are often hidden among numerous legitimate transactions, making it hard to spot them in advance. A thorough and precise analysis of transaction patterns is required to effectively identify these indicators and prevent scams before they happen.

To address these challenges, we present TokenScout, the first temporal GNN-based framework to monitor real-time token transactional behaviors for the early detection of scam tokens. To enrich node and edge attributes for graph learning, we analyze centrality of node and edge, and their related transfer behavior. We formulate token transfer activities as a novel Dynamic Temporal Attributed Multigraph (DTAM), in which nodes and edges having rich attributes. DTAM is capable of continuous evolution and allows for multiple edges between any pair of nodes.

To learn the robust temporal representation for the token transfer activities on DTAM, we devise a novel Token-Flow Graph Neural Network (TF-GNN), which is a GNN. TF-GNN incorporates a temporal encoder to encode timestamps as temporal semantic representations, next aggregates these with edge and node representations strategically, then learns the representation of overall token transfer graph by aggregating the representations of token creator, investors, and all transfer events. Through this, TF-GNN adeptly captures the dynamic and structural characteristics of token transactions.

To early-detect scam tokens, TokenScout employs contrastive learning-based refining that distinguishes between similar and dissimilar token transaction patterns, effectively capturing behaviors specific to scam tokens. Specifically, it clusters instances with similar scam behaviors closer together in the embedded space and separates those with distinct patterns. It ensures that subtle early-stage scam discrepancies are effectively identified. The enhanced graph representation is analyzed by a detection model to identify potential risks and classify the type of scam. With real-time monitoring of token transfer behaviors, TokenScout can deliver prompt scam alerts.

To evaluate TokenScout, we compiled a new large-scale token transfer dataset from 214,084 standard ERC20 marketplace tokens on Ethereum, spanning from the first token in 2015 to February 2023. This dataset contains 9,711,502 token transfer events. The token labeling involved four experienced token risk auditors, costing over 800 man-hours for labeling of benign tokens and scam token, including rugpull, honeypot, and Ponzi tokens. The evaluation results show that TokenScout achieves a precision of 98.03%, recall of 97.47%, F1 score of 97.75%, and balanced accuracy (BAC) of 98.41% in early-detecting scam tokens. In identifying token types, TokenScout achieves a precision of 95.71%, a recall of 94.90%, an F1 score of 94.85%, and a BAC of 94.90%.

We also deployed TokenScout on Ethereum from March 1 to May 31, 2023. It exhibited a low False Negative Rate (FNR) of 3.55%, 4.92%, and 0 in detecting rugpull, honeypot, and Ponzi tokens, respectively. Over this period, TokenScout is estimated to have alerted financial losses amounting to $245.01 million. Furthermore,

we conducted the evasion study by evaluating TokenScout's resilience against adversarial manipulations. Even under combination of three manipulations, TokenScout demonstrates robust performance, with an FNR of 2.95%, 3.83%, and 4.13% for rugpull, honeypot, and Ponzi tokens, respectively. We also performed interpretability analysis to better understand TokenScout's decisions in risk assessments.

The contributions of this paper are summarized as follows:

- We propose TokenScout, the first temporal graph learning-based framework for monitoring the real-time token transactional behavior to predict the future risk of ERC20 tokens;
- We introduce DTAM to formulate intricate token transfer activities and devise effective temporal graph learning strategies on DTAM;
- We compiled a large dataset for scam token detection, consisting of over 9 million token transfer events for all historical tokens as of February 2023 on Ethereum, and the token risk labels built on over 800 man-hours. To aid future research in the community, we open-source the raw and processed datasets at bit.ly/erc20-token-transfers;
- We comprehensively evaluated TokenScout on our labeled dataset across diverse settings. The results demonstrate its effectiveness in early-detecting scam tokens, outperforming existing methods and baseline graph models. We also evaluated TokenScout on real world Ethereum transactions and performed evasion study.

2 Background

In this section, we brief the background of Ethereum tokens and scam tokens.

2.1 Ethereum Tokens

Ethereum facilitates smart contract execution and houses two account types: Externally-Owned Accounts (EOAs) for user-driven interactions using public-private key pairs, and Contract-Owned Accounts (COAs) that automate operations based on embedded code. ERC20 specifies six essential functions interfaces and two events, facilitating token transfers and ensuring transparency through a public ledger [73]. This allows tokens to be easily exchanged and transferred via the execution of specific functions (e.g., `transfer`). Token transfer on Ethereum can be launched by EOA or COA, require gas fee to process, serving as a spam preventation and a resource allocation mechanism [46, 54, 79].

To enable the exchange of tokens among various accounts, Decentralized Exchanges (DEXs), e.g., Uniswap [24] and SushiSwap [20], play a crucial role. They operate using liquidity pools, which are essentially reserves of token pairs that enable trading. Liquidity Providers (LPs) are accounts that deposit specific pairs of tokens into these pools to ensure there is enough of each token available for trading. For example, an LP could deposit a combination of widely-recognized tokens (e.g., `WETH`), and a newly-issued marketplace token (e.g., `Dogecoin`), into a liquidity pool. In return for their contribution of helping maintain the exchange's liquidity, LPs receive LP tokens from DEXs. On the contrary, unlike LP tokens tied to specific liquidity pools, marketplace tokens represent investments and governance rights, making them broadly traded and highly susceptible to scams. The open nature of DeFi, coupled with its minimal regulatory oversight, exposes these marketplace tokens to fraudulent schemes like rugpulls, where their value is artificially inflated for exploitation to deceive investors [44, 53].

Note that token creators, LPs, and/or investors can engage in malicious scam activities. For example, token creators may artificially manipulate market supply by burning tokens, while LPs might suddenly remove their stakes from liquidity pools, destabilizing token values. Investors, on their part, may engage in deceptive practices like wash trading, creating false market activity by buying and selling the same tokens, or initiating pump-and-dump schemes to inflate token prices before selling for a profit. These manipulative strategies not only threaten the DeFi marketplace's fairness but also expose unsuspecting participants to significant risks. The scam has grown with the popularity of ICOs as a means of raising funds, heightening the potential for scams, particularly among DeFi newcomers [7, 15, 18, 50]. Thus, our research aims to develop effective methods to identify the potential risks of ERC20 marketplace tokens, ensuring the integrity and security of the DeFi market.

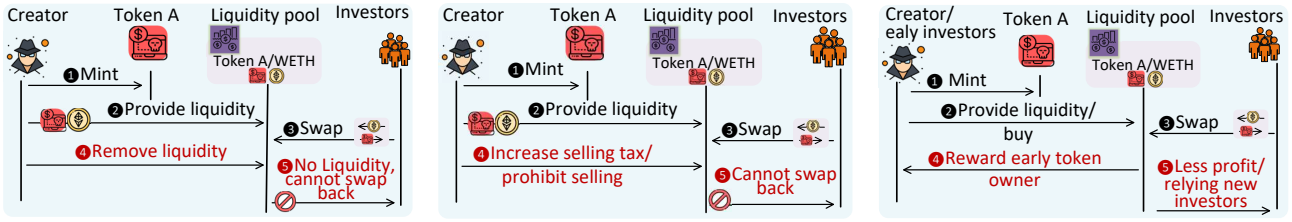
2.2 Scam Tokens

Scam tokens, stemming from fraudulent activities, are broadly classified into three categories: rugpull, honeypot, and Ponzi tokens.

Rugpull tokens entice investors with the allure of a promising project, only for the initiators to suddenly abandon the endeavor and abscond with the invested funds, typically before investors can liquidate their holdings [18, 27, 29]. These tokens often employ deceptive strategies such as misleading tokenomics, strategic liquidity withdrawals, concealed ownership, and price manipulation. As depicted in Figure 1(a), a typical rugpull involves the perpetrator establishing a token, marketing it, and providing liquidity on a DEX. Furthermore, they might mimic the names of popular or emerging legitimate tokens to further lure unsuspecting investors. At the trading volume's apex, the liquidity is abruptly removed by the fraudster, rendering the tokens worthless for investors.

Honeypot tokens exploit potential investors by manipulating smart contracts to enforce restrictive selling conditions. They entice investors with appealing offers but, post-acquisition, restrict trading, making the tokens essentially illiquid, as depicted in Figure 1(b). Unlike traditional honeypot scams [61], which are evident from coding flaws and attract funds to a deceptive contract, honeypot tokens subtly introduce conditions such as high selling taxes or limits. They often involve liquidity pool manipulations, external contracts, or blacklists/whitelists. For instance, it might whitelist only select investors, allowing only them to sell the token.

Ponzi tokens operate by using new investors' funds to pay returns to earlier participants, giving an illusion of legitimacy and enticing further investments, as depicted in Figure 1(c). Such schemes have become notable gas consumers on the Ethereum [9]. As the influx of new investments wanes and the investor pool expands, the scheme becomes unsustainable and collapses, resulting in significant losses for later entrants. To prolong the scheme's viability, scammers might modify contracts or impose selling fees, eventually depleting liquidity or misappropriating investors' assets. These tokens, which often last longer than expected, congest the network



(a) Rugpull token (b) Honeypot token (c) Ponzi token

Figure 1: Illustration of rugpull, honeypot, and Ponzi token

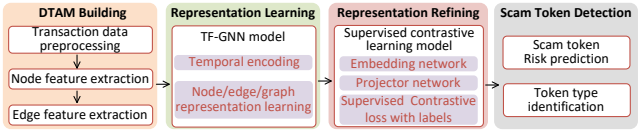
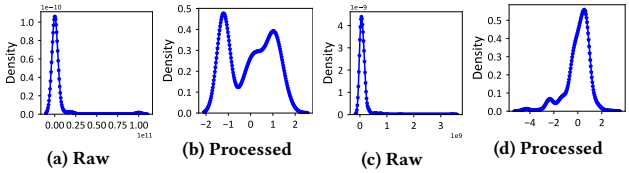


Figure 2: Workflow of TokenScout



(a) Raw (b) Processed (c) Raw (d) Processed

Figure 3: Example of token 0x55b..7855's (a,b) and token 0x225..d1e1's (c, d) raw value and processed value

and tarnish the industry's reputation. Telltale signs include transactions concentrated among few addresses, incrementally increasing transaction values, and a centralized distribution with a majority of tokens retained by a limited set of addresses.

It is crucial to recognize that these categories are not mutually exclusive; fraudsters can integrate multiple scam strategies within a single token. For instance, a Ponzi token might initially reward early investors using fresh investments, but finally the scammer suddenly draining all remaining funds, mirroring rugpull tactics [64]. Driven by the critical influence of token creators, LP providers, inexperienced investors, and the presence of abnormal token transfer behaviors like wash trading and pump-and-dump schemes, we aim to design temporal GNN-based method to meticulously characterize these aspects and the early-detect scam tokens.

3 TokenScout

In this section, we present the overview of TokenScout and detail how it works.

3.1 System Overview

Figure 2 illustrates the workflow of TokenScout, consisting of four stages, namely, DTAM building, representation learning, representation refining, and scam token detection. In this DTAM building stage, TokenScout constructs its token transfer activities into a DTAM for each token. It operates via preprocessing transaction data and extracting features for the nodes and edges. Representation learning is designed on a novel temporal graph learning model, Token Flow GNN (TF-GNN), to learn a comprehensive and robust graph representation for each token's token transfer graph. To represent global and local graph structural and temporal characteristics of the graph, it encodes temporal dynamics using a learnable function, and aggregates this temporal representation with the features of neighboring nodes and edges. TF-GNN allows to learn the comprehensive structural characteristics and temporal patterns of DTAM, providing a comprehensive understanding of the token flow and highlighting potential anomalies indicative of scam activities. To represent the transaction behavior characteristics of scam tokens, representation refining, serving as the decoder, employs a supervised contrastive learning-based model to refine the representations from TF-GNN. It is trained by incorporating token labels and learned graph representations. Scam token detection is a classification model trained using the refined representations and token labels, predicting the risk of potential scam tokens and token types. This process integrates encoding, refining, and predictive analytics into a coherent workflow for scam tokens identification and classification.

3.2 DTAM Building

To capture the temporal dynamics of the transfer activities of an ERC20 token, we introduce a novel Dynamic Temporal Attributed Multigraph (DTAM), denoted as $G = (V, E, \boldsymbol{X}_V, \boldsymbol{X}_E)$, for modeling these activities. V is the set of nodes, consisting of Ethereum account addresses, including both EOA and COA, involved in token transfers. E consists of the set of edges, where each edge $e = (u, v, t)$ represents a token transfer from account u to v, annotated with the timestamp t. Given that two nodes can be connected by multiple edges, each distinguished by unique timestamps, G is thereby a temporal attributed multigraph. Furthermore, attributes are associated with each node and edge. Specifically, each node $v \in V$ is represented by a feature vector $\boldsymbol{x}_v \in \boldsymbol{X}_V$. Similarly, each edge e is described by a feature vector $\boldsymbol{x}_e \in \boldsymbol{X}_E$. To construct the DTAM from a token's transfer activities, it encompasses the following steps.

Transaction data preprocessing. Tokens can exhibit varying denominations due to different decimal places, often spanning several orders of magnitude, with a few extremely high values. To prepare the token transfer values for effective representation learning in GNN models, the data undergoes two critical transformations.

The first transformation, unit standardization, adjusts the scale of token values to a standardized unit by applying $vl'_e = vl_e/10^{\text{dec}}$, where vl'_e and vl_e explicitly denote the standardized and original values of edges e. *dec* represents the number of decimal places in the token's smallest unit. It ensures that the values across different tokens are comparable by normalizing them to a common scale, thus avoiding distortions caused by tokens with vastly different value magnitudes. Figure 3 depicts the raw value and processed value of two tokens for better understanding. To address the skewed distribution typically observed with token values, log-scaling is applied, normalizing the distribution and reducing the impact of extreme outliers. Following log-scaling, standard normalization is employed to adjust the distribution further, setting its mean to zero and standard deviation to one. This results in a distribution that is more Gaussian-like, which is beneficial for model training as it enhances learning efficiency and model performance by providing a standardized input feature set, thus facilitating the model's ability to learn general patterns across different tokens [56].

Timestamp normalization. Transaction data inherently carry crucial temporal information that is essential for understanding transaction behaviors. To effectively utilize this information, each edge e in the graph is assigned a normalized timestamp t'_e calculated as: $t'_e = t_e - t_{\text{start}}$, where t_e represents the actual timestamp of the transaction corresponding to edge e, and t_{start} is the timestamp of the first token transfer event in the dataset. This aligns all transactions relative to the start of the data capture, emphasizing their chronological sequence and aiding in the identification of temporal patterns indicative of anomalous behaviors.

The intuition behind normalizing timestamps is to enhance the model's ability to analyze temporal token transfer activities by ensuring uniform transaction intervals. This approach reduces noise from external factors and mitigates variability from unrelated off-chain events. Additionally, it helps temporal encoders capture temporal relationships and dependencies, reflecting irregularities caused by off-chain events in a controlled manner. Normalizing timestamps also provides a consistent basis for comparison across different time periods and tokens, aiding in the model's generalization. While off-chain events matter, our model focuses on detecting scam behaviors rooted in on-chain activities, such as token transfer traits, token creator behavior, and investor behavior. This ensures that detection is not compromised by external temporal variability, allowing the model to maintain its focus on the essential on-chain data for more accurate identification of scam behaviors.

Node feature extraction. In the original token transfer graph, nodes have no inherent features. To build DTAM and better understand the roles and behaviors of nodes, we compile the node's features from two perspectives, including node importance and its transfer behavior, as presented in Table 1. The intuition is that scammers exhibit anomalous transactional behavior to deceive the investors, and these nodes have high node importance. By capturing node importance and token transferring behavioral characteristics from various perspectives in the graph, we can create accurate and informative node features of the underlying dynamics and relationships. For example, a higher importance score may indicate active investor and token owner participating in token transfer events.

TokenScout extracts node centrality and its transfer frequency to represent the node importance and its transfer behavior. The extracted centrality features include degree centrality, indegree centrality, outdegree centrality, betweenness centrality, closeness centrality, eigenvector centrality, Katz centrality, and clustering coefficient. Besides, it analyzes the behavioral characteristics by extracting the incoming and outgoing transfer frequency in long-term and short-term windows respectively. The features offer insights into structural and dynamic interactions of token transfers, playing a pivotal role in early scam detection. For example, a small node degree centrality could hint at scam attempts. Similarly, frequent transfers in transfer patterns might indicate fraud. Katz centrality unveils abnormal network structures common in scams, while a high clustering coefficient might reveal closely coordinated scam activities.

Edge feature extraction. Edges in the token transfer graph have only the transfer value and timestamp. The intuition is that leveraging the features of neighboring nodes and edges enriches the edge attributes, enabling a more comprehensive analysis of the token transfers. As outlined in Table 1, TokenScout compiles the edge features encompassing transfer value, harmonic transfer value, accumulated degrees of incoming and outgoing transfers. To encapsulate the dynamics of transfer behavior, it also analyzes accumulated token transfer frequency over long-term and short-term windows for each edge, which captures the frequency and patterns of token transfers. These edge features illustrate relationships between nodes, behavioral characteristics of token movements, and ongoing activities, offering a comprehensive insight into token transfer dynamics for risk detection.

For instance, an unusual high transfer value might indicate a potential scam, a common tactic involving asset movement anomalies. Similarly, irregular patterns in accumulated transfer degrees or frequencies could signify orchestrated, abnormal token transfer behaviors associated with fraud. The harmonic transfer value, emphasizing smaller transactions, aids in spotting networks where multiple small transactions camouflage scam activities. Finally, for each token, the transfers are constructed into a DTAM, where each node and edge are represented with 16 and 14-dimensional attributes, respectively.

3.3 Representation Learning

Although, existing GNNs or their variants, e.g., GCN [42] GraphSAGE [38], EvolveGCN [51], and TGAT [68], achieved great successes in various graph-related tasks, e.g., node classification, clustering, and link prediction, they operates on static graph, or fixed-timeframe graph snapshots, assuming uniform time intervals between events [59, 71, 72]. To our knowledge, there are none research tackling the challenge of learning graph representation from the complex and fast-paced movements of token transfers in a DTAM. We devise TF-GNN, an effective graph representation learning approach for token transfers.

TF-GNN-based graph learning. *Temporal encoding.* To encode the temporal dynamics in DTAM, we employ a learnable temporal encoder network inspired by TGAT [68] for encoding the temporal information as a representation vector. The key idea is to map interval from the token-level first transfer event to temporal feature

Table 1: Extracted edge and node features in building DTAM

Type (#)	Feature	Description
Node (16)	Degree centrality	Proportion of edges attached to the node relative to total number of other nodes
	Indegree centrality	Proportion of incoming edges attached to the node relative to total number of other nodes
	Outdegree centrality	Proportion of outgoing edges attached to the node relative to total number of other nodes
	Betweenness centrality	Degree to which of a node acting as a bridge or gateway in a graph
	Closeness centrality	Proximity of a node to all other nodes in a graph
	Eigenvector centrality	Importance of a node by considering the quality of edges
	Katz centrality	Importance of a node by considering direct and indirect connections
	Clustering coefficient	Local density of the graph around the node
	Long-term incoming transfer frequency	Average transfer value and number of incoming transfers to a node within a long-term window
	Short-term incoming transfer frequency	Average transfer value and number of incoming transfers to a node within a short-term window
	Long-term outgoing transfer frequency	Average transfer value and number of outgoing transfers from a node within a long-term window
	Short-term outgoing transfer frequency	Average transfer value and number of outgoing transfers from a node within a short-term window
Edge (14)	Transfer value	Normalized log-scaled transfer value
	Harmonic transfer value	Harmonic mean of the nodes' degree centrality modulated by the transfer value
	Accumulated incoming transfer degree	Total transfer value and number of incoming transfers to the same destination before the edge's timestamp
	Accumulated outgoing transfer degree	Total transfer value and number of outgoing transfers from the same source before the edge's timestamp
	Long-term accumulated incoming transfer frequency	Average transfer value and number of an edge's accumulated indegree within a long-term window
	Short-term accumulated incoming transfer frequency	Average transfer value and number of an edge's accumulated indegree within a short-term window
	Long-term accumulated outgoing transfer frequency	Average transfer value and number of an edge's accumulated outdegree within a long-term window
	Short-term accumulated outgoing transfer frequency	Average transfer value and number of an edge's accumulated outdegree within a short-term window

vectors using a generalizable and learnable time encoder, i.e., $\Phi : t \rightarrow \mathbb{R}^{2d}$. The temporal encoding function is formulated as Eq. 1.

$$\Phi(t) = \frac{1}{\sqrt{d}} \left[\cos\left(\omega_1 t\right), \sin\left(\omega_1 t\right) \ldots \cos\left(\omega_d t\right), \sin\left(\omega_d t\right)\right] \quad (1)$$

where $\omega_1 \ldots \omega_d$ represent learnable weights of the temporal encoder. The time encoding function can generate temporal representation to capture token transfer dependencies, better our understanding of temporal relationships in DTAM.

Node representation learning. In traditional GNNs, learning node representations involves two fundamental steps: aggregating the representation of a node and its neighboring nodes, and then, propagating these to their neighbors. The intuition is to capture node representations by combining the historical and current embeddings of neighboring nodes, while also integrating the temporal information. In DTAM, TF-GNN operates through three additional steps: **N❶** Each edge's interval from the token's first transfer event is encoded into a temporal embedding. **N❷** The embeddings of neighboring nodes are aggregated, integrating the corresponding temporal embeddings and considering the characteristics of the connecting edge, formulated as Eq. 2.

$$z^l_{N(v)} \leftarrow Agg(z^{l-1}_u || \Phi(t_v - t_u), \forall u \in N(v)) \quad (2)$$

where Agg is the aggregation operator. $||$ is the concatenation operator. $N(v)$ is the neighboring nodes of node v. z^{l-1}_u is the representation of node u at graph layer $l-1$. **N❸** The updated node's embedding is then aggregated and propagated to its neighboring nodes, with the connecting edge and temporal embedding being factored into the propagation process, formulated as Eq. 3.

$$z^l_v \leftarrow \sigma(\boldsymbol{W}^l_V \cdot (z^{l-1}_v || z^l_{N(v)})) \quad (3)$$

where $\boldsymbol{W}_V$ is the learnable weigh matrix in updating node embedding. σ is the activation function.

Edge representation learning. We devise the following strategies for learning edge representation: **E❶** it aggregates representations of incoming and outgoing neighboring edges, formulated as Eq. 4.

$$\begin{aligned} f^l_{IN(e)} &\leftarrow Agg(f^{l-1}_e || \Phi(t_e - t_{ie}), \forall ie \in IN(e)) \\ f^l_{ON(e)} &\leftarrow Agg(f^{l-1}_e || \Phi(t_e - t_{oe}), \forall oe \in ON(e)) \end{aligned} \quad (4)$$

where $IN(e)$ is the incoming transfer to the same destination node of e before the timestamp of the current edge. $ON(e)$ is the outgoing transfer to the same source node of e before the timestamp of the current edge. f^{l-1}_e is the embedding of edge e at graph layer $l-1$. The behind idea stems from our observations that token transfers to the same node (incoming) or from the same node (outgoing) might have similar objectives or intentions, such as facilitating a scam. **E❷** It also incorporates the representations of the two nodes connected by an edge, formulated as Eq. 5.

$$f^l_{uv} \leftarrow Sum(z^l_u, z^l_v) \quad (5)$$

The intuition is that the attributes of connected nodes provide critical structural information, enriching the edge representations by considering both node and edge relationships. **E❸** Lastly, edge representations are aggregated and propagated across multiple layers in DTAM to iteratively refine them, enhancing the model's ability to expressively capture the graph's structure and temporal dynamics, formulated as Eq. 6.

$$f^l_e \leftarrow \sigma(\boldsymbol{W}^l_E \cdot (f^{l-1}_e || f^l_{IN(e)} || f^l_{ON(e)} || f^l_{uv} || \Phi(t_i))) \quad (6)$$

where $\boldsymbol{W}_E$ is the learnable weigh matrix in updating edge embedding. σ is the activation function.

Graph representation learning. TF-GNN is designed to learn the holistic representation of DTAM, consisting of three strategies. **G❶** It aggregates updated node and edge representations in the graph. The intuition is to collectively analyze the interactions and relationships among nodes and edges, which can help unveil patterns indicative of scam activities. **G❷** It aggregates and propagates graph representations, merging the updated node and edge representations with the previous graph representation. This helps capture higher-order interactions, reflecting both the local and global context, thereby enriching the representations. **G❸** It aggregates the representations of the token creator and token investor respectively,

Algorithm 1 Representation Learning of TF-GNN on DTAM

Input: DTAM $G = (V, E, \boldsymbol{X}_v, \boldsymbol{X}_e)$, # layers L, node and edge weight matrix $\boldsymbol{W}_v$ and $\boldsymbol{W}_e$, temporal encoder Φ, activation function σ
Output: Graph representations h_g

1: $z_v^0 \leftarrow \boldsymbol{X}_v, \forall v \in V$; $f_e^0 \leftarrow \boldsymbol{X}_e, \forall e \in E$; $h_g^0 \leftarrow \boldsymbol{0}$
2: **for** $l = 1$ to L **do**
3: **for** $v \in V$ **do**
4: $z^l_{N(v)} \leftarrow Agg(z_u^{l-1}||\Phi(t_v - t_u), \forall u \in N(v))$ # N❶❷
5: $z_v^l \leftarrow \sigma(\boldsymbol{W}_V^l \cdot (z_v^{l-1}||z^l_{N(v)}))$ # N❸
6: **end for**
7: **for** $e(u, v, t_e) \in E$ **do**
8: $f^l_{IN(e)} \leftarrow Agg(f_e^{l-1}||\Phi(t_e - t_{ie}), \forall ie \in IN(e))$ # E❶
9: $f^l_{ON(e)} \leftarrow Agg(f_e^{l-1}||\Phi(t_e - t_{oe}), \forall oe \in ON(e))$ # E❶
10: $f_{uv}^l \leftarrow Sum(z_u^l, z_v^l)$ # E❷
11: $f_e^l \leftarrow \sigma(\boldsymbol{W}_E^l \cdot (f_e^{l-1}||f^l_{IN(e)}||f^l_{ON(e)}||f_{uv}^l||\Phi(t_i)))$ # E❸
12: **end for**
13: $h_v^l \leftarrow Agg(z_v^l, \forall v \in V)$ # G❶
14: $h_e^l \leftarrow Agg(f_e^l, \forall e \in E)$ # G❶
15: $h_g^l \leftarrow Sum(h_g^{l-1}, h_v^l, h_e^l)/(|V| + |E|)$ # G❷
16: **end for**
17: $tc, ti \leftarrow split\_ceator\_inverstor(V)$
18: $h_{tc} \leftarrow Agg(z_v^L, \forall v \in tc)$; $h_{ti} \leftarrow Agg(z_v^L, \forall v \in ti)$
19: $h_g \leftarrow [h_g^L||h_{ti}||h_{tc}]$ # G❸
20: **return** h_g

then merge these with the graph representation to form the final token representation. The intuition to fuse different perspectives, i.e., the creator's intent, the investor's behavior, and the overall token transaction patterns, into a unified representation, illustrating the token's diverse roles and interactions.

Algorithm 1 outlines the graph representation learning process of TF-GNN. It operates on a DTAM G, and with the inputs of node and edge weight matrix W_v and W_e, the number of GNN layers L, an activation function σ, and a temporal encoder Φ to incorporate time information. Initially, each node v and edge e are assigned representations based on the extracted features, and the graph representation is initialized to zero. z_v^0, f_e^0, and h_g^0 are representations of nodes, edges, and graph at the initial graph layer, respectively. In each of the L layers, TF-GNN proceeds through three steps: learning node representations in the first inner loop (Line 3-6), learning edge representations in the second inner loop (Line 7-12), and finally, updating the graph representation (Line 13-15). In the first inner loop, each node's representation z_v^l is updated by aggregating its neighbors' representations considering time difference and its previous layer representation. In the second loop, TF-GNN aggregates incoming and outgoing edge representations, combines connected nodes' representations, and updates edge representation. At each layer's end, the graph representation is updated by aggregating (Line 13, 14) and averaging the node and edge representations (Line 15). After all layers are processed, The nodes are split into in creators and investors based on the number of token transfer events (Line 16), and the representation of creators and investors are aggregated respectively (Line 17). Finally, the last layer's token graph representation f_{tg} is fused with representations of token creator f_{tc} and token investor f_{ti} representations to form the graph representation h_g (Line 19). After graph learning, the output of graph learning model is then used for refining. The output is a 192-dimensional representation vector, where f_{tg}, f_{ti}, and f_{tc} are with 64-dimensional.

3.4 Representation Refining

To enhance the model's generalization ability in the open-world ERC20 token risk monitoring, we also design the representation refining to yield the semantically more representative features for token transfer behaviors. Our key idea is to learn an encoder mapping the extracted graph feature representations into a lower-dimensional latent space, which robustly and effectively express the token transfer behavioral characteristics and its token risks.

To fulfill the goal, we design representation refining based on contrastive learning (CL). It works by minimizing the distance between representations of instances within the same class while maximizing distance between instances from different classes. It incorporates the token labels and minimize the contrastive loss to learn the robust encoder. To train such model on the imbalanced dataset, we use the asymmetric contrastive loss, formulated as Eq. 7.

$$\textbf{min} - \sum_{i=1}^{N}\Big(\sum_{j\in P_i}\log(p_{ij})/|P_i| + \eta \sum_{j\in N_i}\log(1-p_{ij})/|N_i|\Big) \quad (7)$$

where $p_{ij} = \frac{\exp(z_i \cdot z_j/\tau)}{\sum_{j\in P_i \cup N_i}\exp(z_i \cdot z_j/\tau)}$. N is the total number of instances in a batch. z_i is the refined representation of instance i in the lower-dimensional space. The positive set, $P_i = \{x_j : y_j = y_i, j \neq i\}$, includes all instances that share the same label as z_i. The negative set, $N_i = \{x_j : y_j \neq y_i\}$, includes all instances that have a different label from z_i. y_i and y_j represents the label of instance i and j, τ is the temperature parameter, controlling the concentration of the probability distribution, η is the fixed hyperparameter to balance the positive and negative sets. The loss function tries to maximize the similarity for the embeddings from the same class and minimize the similarity for the embeddings from different classes.

In detail, the representation refining model comprises two primary networks: an encoder and a projector. These two networks collaboratively map the initial graph representations into a more discriminative latent space, serving our goal of enhancing scam token detection performance. The encoder, acting as the core of feature extraction, is designed as a fully connected neural network with two hidden layers. Specifically, it follows an Input(192) → FC(96) → ReLU→ FC(48) → ReLU → FC(24) architecture, ensuring an optimal abstraction of meaningful features from the input representations. In contrast, the projector further refines these feature spaces, ensuring suitability for the application of contrastive loss. The projector comprises a fully connected network with a structure of Input(24) → FC(12) → ReLU → FC(24). By applying this structure, the features from the encoder are effectively mapped into a space where the contrastive loss can be effectively optimized. The combination of the encoder and projector assures the effective refinement of initial graph representations, delivering robust and discriminative feature representations for scam token classification. After training the representation refining model, the encoder network is used to refine graph representations. The projector, optimizing the contrastive loss during training, is discarded at inference

phase. The output is a 24-dimensional feature vector, which is then fed into the token classifier to identify the token risk.

3.5 Scam Token Detection

To monitor token risk and scam type, the detection model uses a neural network, structured as FC(24) → ReLU → Softmax(4). It utilizes the refined graph representation as input, transferring the learned token representation into the scam-token detection task. The final SoftMax layer yields a four-dimensional output vector, corresponding to four distinct token detection results: trustworthy, rugpull, honeypot, and Ponzi token. Each element in the output is the categorical probability of the corresponding token class, representing the prediction score for a specific token class. The final classification result for each token is determined by the class corresponding with the highest logit value in the SoftMax output.

4 Dataset

Token transfer data collection. Our experimental dataset was collected spanned from 2015 to Feb. 28, 2023, before the start of our experiment. We utilized the QuickNode archive node [16] to query the entire transaction history of the Ethereum main network. By employing the `web3.py` library [26], we created a filter that monitored the `transfer` event logs of standard ERC20 tokens. We targeted the ERC20 transfer function signature (0xa9059cbb) and verified that the `to` field matched the token contract address, allowing to capture all token transfers. Besides, we excluded LP tokens if the token was issued by DEXs, such as UniSwap v2 and v3.

For each transfer, we extracted the fields including token address, sender, recipient addresses, transfer value, and its timestamp. We focused on the earliest 500 token transfer events for each token, since these transfers represent a critical phase in a token's life, reflecting initial adoption patterns, liquidity provision, and initial user behaviors, serving as a window into the token's characteristics and early activities. They encapsulate essential insights into potential fraudulent activities or anomalous patterns. Training a detection model on these initial transfers can enable real-time risk tracking and early detection of scam tokens.

Token labeling. To achieve precise token groundtruth labeling, four token auditors in an anoymous leading Web3 security team contribute scam tokens labeling, including already-happened and suspicious scam tokens, where each auditor has at least one-year-experience in auditing. They identified scam tokens based on their domain knowledge of token risk by considering various factors, e.g., token contract codes, historical transactions, pool liquidity, price fluctuations, auditing reports from other famous auditing platforms, and project's whitepaper. Specifically, they were required to follow a rigorous auditing process, where a token might be labeled as a scam based on characteristics suggesting potential fraudulent activity, even if it had not yet scammed or stolen investors' funds at that point. Each token was reviewed by at random three auditors independently. If two or more auditors disagreed on a token's label, they would discuss and reach a consensus. A random subset of labeled contracts was cross-verified by another auditor to ascertain the quality of labeling. Additionally, for the token that have already scammed, the auditors documented marking scam occurrences, e.g., the instance when a significant token price drop happened.

To accelerate the labeling of tokens with obvious scam characteristics, they designed useful heuristic rules implementing part of auditors' domain knowledge. These rules inspect the token contract and simulates trading for fast determining the obvious scam token. Specifically, a token matching multiple rules with high confidence, it will be labelled as honeypot or Ponzi token with high priority. Besides, for most tokens not matching any rules or with low confidence, they further verified through the following criteria ranked by priority: (i) a token is labelled as scam token if reported as a rugpull, honeypot, or Ponzi by reputable blockchain auditing communities, including e.g., Goplus, CertiK, PeckShield, Blocksec, and MetaTrust [2, 4, 11–13]; (ii) a token is labeled as a Ponzi if it incentivizes existing investors to recruit new ones through referral bonuses or additional tokens, as stated in the whitepaper or social media, or if it relies on a constant flow of new investments to provide returns to older investors in past token transfers; (iii) a token is labeled as rugpull, if there is a sharp drop on token price and the sender of this transaction is the token contract owner or has connection with the owner in the past, or the contract owners hold more than 10% tokens; (iv) tokens are labelled as suspicious rugpull candidates if they meet any of the following conditions: lacking intrinsic value, remaining inactive for more than a week, being absent from mainstream DEXs, or having a token contract that is not open-sourced; (v) a token is labeled as trustworthy if it is known as a famous trustworthy token, e.g., USDT, WETH, or listed on mainstream platforms, including Etherscan [23], CoinGecko [5], and CoinMarketCap [6]. Tokens on these platforms undergo careful manual cross-verification to ensure their credibility. Besides, old token addresses lacking transfer activity but migrated to new verified contract address were labeled as trustworthy.

Also, we recorded specific timestamps for each type of scam occurrence. For rugpull tokens, we noted the moment when the token price dropped sharply. For honeypot tokens, we tracked when the token became unsellable or when the contract owner raised the sell tax beyond 10%. Additionally, we recorded the times when auditing communities reported tokens as honeypots. For Ponzi schemes, we highlighted the timestamps when these schemes were flagged by auditing communities and when there were very few buy transactions within a specific period.

The labeling process took about 56 days, consuming over 800 man-hours. The compiled dataset comprises 9,711,502 token transfer events and 214,084 ERC20 marketplace tokens, among which 1,806 were labeled as trustworthy, and 212,278 were scams. These scam tokens include 179,995 rugpull (84.08%), 22,800 honeypot (10.65%), and 9,483 Ponzi tokens (4.43%). Among these tokens, 37,439 were publicly reported as scams by Goplus, CertiK, PeckShield, Blocksec, and MetaTrust, where 35,547 are rugpulls, 1,421 are honeypots, and 471 are Ponzi tokens. Additionally, 9,452 contracts did not open-source. This presents a significant imbalance with approximately 99.16% tokens being scams and only 0.84% being legitimate, mirroring the prevailing fraudulent activities, particularly rugpulls. The high scam token percentage reflects our strict labeling criteria, based on the auditors' expertise, where many tokens, especially those lacking liquidity and transaction activity, are viewed as potential scams.

5 Performance Evaluation

In this section, we report performance of TokenScout.

5.1 Experimental Setup

Evaluation metrics. To rigorously evaluate performance in such an imbalanced dataset, we used precision, recall, F1 score, BAC as evaluation metrics, calculating over false positives, false negatives, true positives, and true negatives. BAC is the average of recall obtained on each class, i.e., average of true positive rate and true negative rate for detecting scam token, while F1 score is a measure of the harmonic mean of precision and recall. For detecting scam tokens, a false positive is a trustworthy token that is misclassified as scam, while a false negative refers to a scam token misclassified as trustworthy. For predicting token types, we compute the average false positive and false negative across each scam token type to yield the overall false positive and false negative rates. A false positive for a token type arises when a token is misidentified as that type, while a false negative occurs when a token of that type is wrongly classified as another.

Graph baseline models. We considered the following baseline graph model for comparison. *(i) Deepwalk* [52] utilizes random walks to generate node sequences and Skip-Gram to model these sequences, capturing local and global graph structure. *(ii) GCN* [42] is a semi-supervised method that learns node representations through local neighborhood information aggregation. *(iii) GraphSAGE* [38] is an inductive embedding method that employs aggregator functions to generate embeddings, which can produce embeddings for previously unseen nodes or entirely new graphs. *(iv) TGAT* [68] is a temporal graph model that utilizes self-attention mechanisms alongside time encoding, grounded on Bochner's theorem. This approach treats node embeddings as functions that vary over time, allowing for nuanced temporal insights.

Implementation and hyperparameters. To real-time retrieve token transfer data, it follows these steps: Specifically, (i) utilizing the Web3 library [26], it first *connects* to an Ethereum remote procedure call (RPC) node (e.g., Quicknode [16] or Infura [17]), which provides access to the latest blocks, state information,and token transfer data; (ii) transactions from both past and latest blocks are *fetched* using the interface `getBlock` with `full_transactions` as `True`; (iii) it then *extracts* the transactions data involving token transfers by matching the `to` field with the token contract address and verifying that the `input` field starts with the ERC20 transfer function signature (`0xa9059cbb`); (iv) for token transfer event, the `input` field is *decoded* to extract the details, e.g., `sender`, `recipient`, and `transfer value` and the decoded token transfer data are saved for building DTAM. A decoded transfer event can be denoted as (`token_addr`, `from`, `to`, `value`, `ts`), representing the sender address, `from`, transfers a specific amount, `value`, of the token `token_addr` to the recipient address, `to`, at the timestamp `ts`. As an example, a transfer event of token `DGX` [22] is: `{0x55...5,0x38...1, 0x9d...3,2e8,1454624524}`.

Experiments were conducted with Ubuntu 18.04 LTS and the machine with two Intel CPUs (Intel(R) Xeon(R) W-3265M CPU@ 2.70GHz CORE 24), an NVIDIA GPU (A100 PCIE Gen4) and 512G RAM. The environment utilized Python 3.10.7, PyTorch 1.13.0, torch-geometric 2.2.0, and networkx 2.8.8. We implemented neighbor sampling on the CPU and stochastic minibatch training on the GPU to handle the massive graph with millions of nodes and billions of edges. During training, we selected small batches of nodes at each gradient descent step, computing their final representations at the L_{th} layer, and included some or all neighbors of these nodes at the $L-1$ layer. The training, validation, and test sets were split in a 70%-15%-15% ratio, and results were averaged over 10 runs.

We set our two-layer graph model TF-GNN (L=2) to use the mean aggregator for calculating element-wise mean vectors of all selected neighbors. Other aggregators and selection methods are also applicable in Algorithm 1. Short and long-term window for node and feature extraction is 6 hours and 48 hours. Training utilized the Adam optimizer [65] with parameters learning rate as 0.001, τ and η as 0.1 and 300 for contrastive loss. Batch sizes were set to 256, 512, and 1024 for the graph model, representation refining, and classifier respectively, with 60 epochs for the graph model and 50 for the refining and classifier. Negative interactions were equally sampled with positive ones during training, and BAC is used as the reference metric. Early stopping (patience: 5 epochs) was implemented to prevent overfitting. Random undersampling was applied to train the detection model by limiting each batch to 5% scam token.

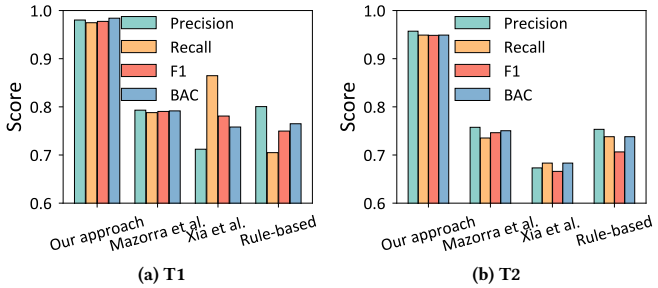
5.2 Performance of TokenScout

Overall performance. To evaluate the performance of TokenScout, we used the node and edge features, DTAM + TF-GNN as the graph model to learn the graph representation, and representation refining to encode the graph representation into a lower-dimensional latent space. Table 2 (setting 1) reports the performance of TokenScout in scam token detection (T1) and token type prediction (T2). In T1, TokenScout achieved an FNR of 2.53%, an FPR of 0.65%, an F1 score of 97.75%, a BAC of 98.41%. In T2, TokenScout maintained high performance, achieving an FNR of 5.10%, an FPR of 1.69%, an F1 score of 94.85%, and a BAC of 94.90%. Results underscore the effectiveness in detecting scam tokens and types. The low FNR and FPR indicate that TokenScout can reliably distinguish between scam and non-scam tokens, minimizing false negatives and false positives. The high F1 scores reflect a balanced precision and recall, highlighting TokenScout's robustness in real-world scenarios. The BAC values, close to 100%, emphasize TokenScout's capability in providing accurate and consistent predictions across different types of scam tokens. These insights affirm that TokenScout not only excels in early detection but also in detailed classification, enhancing the security and integrity of the DeFi ecosystem.

Comparison with baseline graph models. We compared TokenScout with SOTA graph models, including GCN, DeepWalk, GraphSAG, GraphSAGE with LSTM, and TGAT. Originally these models are designed for node representation learning, while we adapted them for whole-graph representation learning. Specifically, for GCN, DeepWalk, and GraphSAGE, we employed mean aggregation on node embeddings to derive graph-level representations. GraphSAGE with LSTM used the final hidden state for graph representation. TGAT's were harnessed for node embeddings, with mean aggregation applied for graph-level representation.

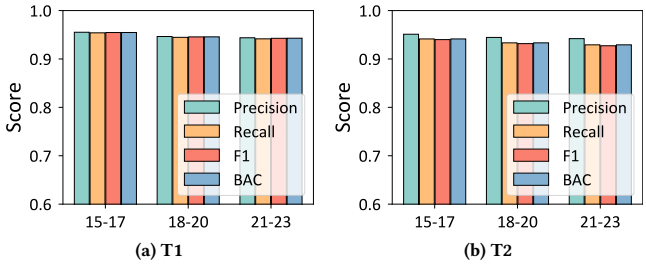
Table 2: Performance of TokenScout (bold). Ablation study results (not in bold) are presented for comparative analysis under different settings (S). * BAC is the average of recall from each class in T2.

# S	Graph model	Node	Edge	Refining	T1 - scam token detection (binary)						T2 - token type identification (four-class)					
					Precision	Recall	FPR	FNR	F1	BAC	Precision	Recall	FPR	FNR	F1	BAC*
1	**DTAM + TF-GNN**	✓	✓	✓	**98.03%**	**97.47%**	**0.65%**	**2.53%**	**97.75%**	**98.41%**	**95.71%**	**94.90%**	**1.69%**	**5.10%**	**94.85%**	**94.90%**
2	Deepwalk	✓	✓	✓	71.45%	75.74%	30.14%	24.26%	73.53%	72.80%	67.17%	67.34%	10.86%	32.66%	64.90%	67.34%
3	GCN	✓	✓	✓	77.80%	73.31%	20.85%	26.69%	75.49%	76.23%	72.31%	71.96%	9.33%	28.04%	70.22%	71.96%
4	Graphsage	✓	✓	✓	78.75%	75.27%	20.23%	24.73%	76.97%	77.52%	76.46%	75.53%	8.14%	24.47%	74.36%	75.53%
5	Graphsage + LSTM	✓	✓	✓	80.79%	79.14%	18.76%	20.86%	79.95%	80.19%	78.70%	77.66%	7.43%	22.34%	76.48%	77.66%
6	TGAT	✓	✓	✓	83.05%	82.92%	16.86%	17.08%	82.99%	83.03%	82.04%	80.60%	6.45%	19.40%	79.58%	80.60%
7	DTAM + TF-GNN	✗	✓	✓	77.53%	74.72%	21.58%	25.28%	76.10%	76.57%	75.75%	75.26%	8.23%	24.74%	73.82%	75.26%
8	DTAM + TF-GNN	✓	✗	✓	85.82%	85.62%	14.10%	14.38%	85.72%	85.76%	84.92%	83.48%	5.49%	16.52%	82.84%	83.48%
9	DTAM + TF-GNN	✗	✗	✓	72.69%	74.33%	27.81%	25.67%	73.50%	73.26%	69.43%	69.47%	10.15%	30.53%	67.25%	69.47%
10	DTAM + TF-GNN	✓	✓	✗	88.40%	90.99%	11.89 %	9.01%	89.68%	89.55%	86.94%	85.93%	4.74%	14.07%	85.09%	85.93%



(a) T1 (b) T2

Figure 4: Comparison of existing scam token detections



(a) T1 (b) T2

Figure 5: Performance under different periods

Table 3: Detailed comparison of different methods in detecting different scam tokens types

Method	Type	FNR	FPR
TokenScout	Rugpull token	3.58%	0.40%
	Honeypot token	4.72%	1.80%
	Ponzi token	6.03%	1.35%
	Trustworthy token	6.07%	3.21%
Mazorra et al. [50]	Rugpull token	24.06%	17.21%
	Honeypot token	22.89%	22.04%
	Ponzi token	27.96%	24.56%
	Trustworthy token	30.97%	29.99%
Xia et al. [67]	Rugpull token	23.84%	33.26%
	Honeypot token	31.71%	31.61%
	Ponzi token	32.34%	36.13%
	Trustworthy token	38.83%	25.72%
Rule-based	Rugpull token	23.62%	25.93%
	Honeypot token	26.80%	25.56%
	Ponzi token	27.17%	22.83%
	Trustworthy token	27.21%	30.48%

As presented in Table 2 (Setting 2, 3, 4, 5 and 6), TokenScout outperformed the graph models across both tasks, achieving the highest F1 scores and BAC. Results show TokenScout'effectiveness in learning feature representation for scam tokens, highlighting its robustness and superiority of TokenScout over SOTA graph models. For example, for task T1, TokenScout achieved a 97.75% F1 score, surpassing the second-best model, TGAT, by 14.76%. In task T2, TokenScout's F1 score of 94.85% was significantly better than the second-best model, GraphSAGE with LSTM.

Comparison with existing methods. At the end of our experiments, we compared our method with two related works, Mazorra et al.[50] and Xia et al.[67], which mainly focus on detecting rugpull scams using token transfer data. For a fair comparison, we used the features and methodologies from these works to train detection models on our dataset for predicting future scam tokens. We adapted their methodologies to detect multiple scam types, including honeypot and Ponzi scams, to evaluate their broader effectiveness. We also compiled the rule-based detection, where we used the label with highest confidence as the decision result for each token.

Figure 4 shows that TokenScout outperforms existing approaches in both scam token detection and type identification. TokenScout achieved a BAC of 98.41% for scam token detection, significantly higher than the 79.16% and 75.81% reported by Mazorra et al. and Xia et al., respectively. In identifying specific scam token types, TokenScout also excelled with a BAC of 94.90%, compared to 75.04% for Mazorra et al. and 68.32% for Xia et al. Table 3 reports the detailed comparison results in detecting different types of scam tokens. TokenScout consistently outperformed both methods across all scam token categories. For example, TokenScout achieved an FNR of 3.58% and an FPR of 0.40% for rugpull tokens, significantly lower than the 24.06% FNR and 17.21% FPR reported by Mazorra et al. Similarly, for honeypot tokens, TokenScout's FNR of 4.72% and FPR of 1.80% were much lower than the 22.89% and 22.04% reported by Mazorra et al. Xia et al. was less effective overall, particularly in detecting Ponzi and honeypot tokens, where it exhibited higher FNRs and FPRs, indicating lower sensitivity and specificity compared to TokenScout.

Robustness over different periods. To evaluate TokenScout's adaptability to evolving scam patterns, we used the datasets of different periods by updating the model with new token transfer data and testing its performance. Starting with training on 70% of the dataset from 2015-2017 and testing on the remaining 30%, we then expanded training to include data from 2018-2020, and finally up to 2021-2023. Figure 5 shows performances under different periods. It achieved a BAC of 95.47%, 94.57%, and 94.29% for respective periods in detecting scam tokens. For scam token prediction, it achieved a BAC of 94.14%, 93.33%, and 92.92%. Results demonstrate TokenScout consistently maintains high performance across different periods, and is resilient against evolving scam behaviors. It also reveals that while the BAC slightly decreases over time, TokenScout remains robust, suggesting its ability to adapt to new scam tactics and patterns. The slight decline in BAC indicates that scammers are continually evolving their methods, but TokenScout's architecture allows it to stay effective by leveraging updated training data. This resilience underscores the importance of continuous model updates and retraining to capture emerging scam strategies, ensuring long-term reliability and efficacy in detecting and predicting scam tokens.

5.3 Ablation Studies

Effectiveness of node features. We evaluated node features' effectiveness by holding TokenScout's other components constant, running tests with and without node features. Analyzing results between setting 1 (with node features) and setting 7 (without node features) in Table 2, there is a performance decrease when node features were not included. In T1, F1 score diminished from 97.75% to 76.10%, and BAC from 98.41% to 76.57%. Similarly, for T2, F1 score and BAC fell from 94.85% and 94.90% to 73.82% and 75.26%, respectively. Results highlight the critical importance of node features in TokenScout's effectiveness. These features, which indicate node significance and token transfer patterns, are crucial for identifying scam tokens. Without them, the model struggles to grasp token transfers, negatively impacting its performance.

Effectiveness of edge features. Similarly, comparing results between setting 1 (with edge features) and setting 8 (without edge features) in Table 2, there is a decrease in performance in the absence of edge features. In T1, F1 score dropped from 97.75% to 85.72%, and BAC from 98.41% to 85.76%. In T2, F1 score and BAC declined from 94.85% and 94.90% to 82.84% and 83.48%, respectively. Edge features also play a pivotal role in TokenScout's performance, capturing key relationships between token transfers and nodes in the graph for effective scam token identification.

Effectiveness of incorporating node and edge features. We also evaluated the joint impact of node and edge features by testing TokenScout in two scenarios: one incorporating both features and another omitting them. Comparing setting 1 and 9 in Table 2, we observed a decline in performance without both node and edge features. In T1, the F1 score and BAC diminished from 97.75% and 98.41% to 73.50% and 73.26% respectively. Likewise, in T2, they decreased from 94.85% and 94.90% to 67.25% and 69.47%. The results underscore the vital role of both node and edge features in TokenScout. Their exclusion diminishes the model's capacity to discern complex token transfer patterns, affirming their essentiality in achieving accurate scam token detection.

Effectiveness of representation refining. Comparing setting 1 and 10, the results exhibit the refining's crucial role. Without it, TokenScout's performance metrics dip: F1 score in T1 falls from 97.75% to 89.68% and in T2 from 94.85% to 85.09%; BAC in T1 drops from 98.41% to 89.55% and in T2 from 94.90% to 85.93%. Clearly, the refining significantly enhances TokenScout's accuracy in detecting scam tokens and identifying their types.

5.4 Real-world Evaluation

To evaluate the effectiveness in predicting future risk, we measured how long in advance our system can predict scam tokens ahead of a scam and how much economic loss can be alerted by early detection. TokenScout was trained using at most 500 transfers prior to the scam, and predicts future token risk using the latest 500 transfers for each token. We also deployed TokenScout on Ethereum to monitor real-time transactions to evaluate its effectiveness. Specifically, the lead time of alerting risk is measured by calculating the interval of detection time and marked scam time. To gauge alerted losses, we calculated the sum value of trustworthy tokens (e.g., WETH, USDT, BNB, WBTC) transferred before TokenScout flags a scam risk, multiplied by the token price when the scam is identified. We also measured FNR of TokenScout, which represents the ratio of undetected scam tokens to the total scam tokens. Table 4 presents performance of TokenScout in alerting scams ahead of time and potential economic loss, during past periods (2015 - 2023.2) and real-world evaluation (2023.3 - 2023.5). When testing the historical token transfer dataset, it detected the rugpull, honeypot and Ponzi tokens with an FNR of 1.68%, 2.98%, and 2.66%, respectively, alerting the loss of 99.66%, 92.14%, and 97.17%. When testing it for real-time risk monitoring, it detected the rugpull, honeypot and Ponzi tokens with an FNR of 3.55%, 4.92%, and 0%, alerting the loss of 90.52%, 93.36%, and 100%, respectively. Results suggest the effectiveness in early-detecting scam tokens risk before their occurrences.

5.5 Evasion Study

Evasions. Scammers may attempt to evade the early-detection of scam token by intentionally manipulating the token transfers. We consider the following black-box evasions, aiming to evade detection through random and inconspicuous attempts:

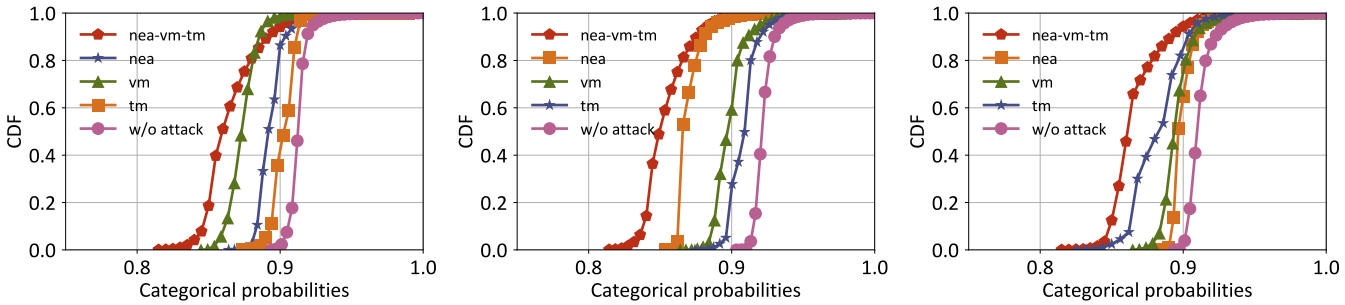
(i) Node and edge addition. The adversary simultaneously introduces new nodes v_{new} into V and new edges e_{new} into E such that $V' = V \cup v_{new}$ and $E' = E \cup e_{new}$, respectively. The new edges e_{new} connect the nodes in V', thereby altering the graph's structural characteristics.

(ii) Value manipulation. The adversary manipulates transaction values by adding noise to the original values X_e, with each noise element drawn from a normal distribution $N_e \sim N(\mu_e, \lambda_e \sigma_e)$, where μ_e and σ_e are the mean and standard deviation of the token transfers' value. λ_e is the factor controlling value manipulation. The manipulation is element-wise, therefore each value in $\boldsymbol{X_e}$ has a unique noise value, i.e., $\boldsymbol{X'_e} = \boldsymbol{X_e} + \boldsymbol{N_e}$.

(iii) Timestamp manipulation. The adversary manipulates timestamps T using a random permutation, i.e., $T' = T + N_t$. N_t is a random permutation of T following a normal distribution, i.e.,

Table 4: Lead time of alerting risk and alerted loss (million USD) under historical transfers and ongoing real-time transfers

	Past periods (2015 - 2023.2)						Real-world evaluation (2023.3 - 2023.5)					
Type	# Detected / Total	FNR	Time (hour)	Prev./Total (million $)	Ratio	FPR	# Detected / Total	FNR	Time (hour)	Prev./Total (million $)	Ratio	FPR
Rugpull	176,971 / 179,995	1.68%	9.67±3.75	7,826.74 / 7,853.12	99.66%	2.54%	706 / 732	3.55%	8.55 ± 4.31	108.07 / 119.38	90.52%	5.95%
Honeypot	22,121 / 22,800	2.98%	78.91±15.22	3,619.44 / 3,928.11	92.14%	3.96%	174 / 183	4.92%	45.32±9.84	73.71 / 78.95	93.36%	6.22%
Ponzi	9,230 / 9,483	2.66%	236.43±49.10	928.38 / 955.37	97.17%	0.42%	92 / 92	0%	343.23±37.10	63.24 / 63.24	100%	2.71%



(a) Rugpull tokens (b) Honeypot tokens (c) Ponzi tokens

Figure 6: Categorical probabilities of rupull (a), honeypot (b), and Ponzi (c) token under adversarial manipulations.

$N_t \sim N(\mu_t, \lambda_t \sigma_t)$, where μ_t and σ_t are the mean and standard deviation the time difference of two consecutive token transfers. λ_t is the factor controlling timestamp manipulation.

Evasion ii and iii differ from strategy i in their impact on the graph. While strategy i adds new transactions, thereby expanding the graph's topology, strategies ii and iii subtly modify existing transactions through value and timestamp manipulation. This approach alters transactional patterns and timings within the graph, rather than merely increasing its size. Testing with such a normal distribution can reflect this scenario, providing a realistic assessment of the model's resilience against untargeted, random inputs. Specifically, λ_e and λ_t are set as 0.1 for evaluation.

Results. To evaluate TokenScout'resilience against potential adversarial manipulations, we simulated black-box testing scenarios, where the attacker has none knowledge about the detection model and attempts to inject noise into historical transactions randomly to evade detection. Table 5 presents FNRs and categorical probabilities (generated by the Softmax layer) of rugpull, honeypot, and Ponzi tokens under three adversarial manipulations, including node and edge addition (nea), value manipulation (vm), and timestamp manipulation (tm). Results suggest that TokenScout is resilient against adversarial manipulations. For example, when not applying adversarial manipulations, FNR is 1.68%, 2.98%, and 2.66% for rugpull, honeypot, and Ponzi tokens, respectively. Under a combination of three attacks, it achieved an FNR of 2.95% for rugpull tokens, 3.83% for honeypot tokens, and 4.13% for Ponzi tokens. Figure 6 presents cumulative distribution function (CDF) of these probabilities under adversarial manipulations. The mean categorical probabilities are 91.28%, 92.24%, and 91.17% in normal conditions to 86.45%, 85.34%, and 86.55% under these attacks, respectively.

Table 5: Mean/standard deviation of categorical probabilities and FNRs under adversarial manipulations

Type	Setting	Probabilities	FNR
Rugpull token	nea-vm-tm	86.45% / 1.89	**2.95%**
	nea	89.31% / 0.94	2.42%
	vm	87.34% / 0.96	2.67%
	tm	90.23% / 0.77	2.33%
	w/o attack	91.28% / 0.67	**1.68%**
Honeypot token	nea-vm-tm	85.34% / 3.95	**3.83%**
	nea	86.95% / 0.95	3.69%
	vm	89.84% / 1.11	3.27%
	tm	90.79% / 1.03	3.15%
	w/o attack	92.24% / 0.72	**2.98%**
Ponzi token	nea-vm-tm	86.55% / 1.76	**4.13%**
	nea	89.94% / 0.79	3.75%
	vm	89.52% / 1.08	3.21%
	tm	88.14% / 1.79	3.84%
	w/o attack	91.17% / 0.96	**2.66%**

6 Discussion

Despite our efforts to maintain the effectiveness of our study, there still exist several limitations. In our evaluation, we disregarded price fluctuations resulting from attack events or broader economic factors. These shifts can be triggered by unexpected contract vulnerabilities, foundational protocol issues, or external influences impacting market sentiment. Particularly, 0-day exploits pose distinct challenges, eluding detection through mere on-chain transaction analysis. Moreover, off-chain events like geopolitical developments or regulatory news can influence token prices, making predictions based solely on transactional patterns complex. Additionally, our large-scale dataset compilation process might have resulted in a few mislabeled tokens and overlooked new types of scams, highlighting the complexity of accurately categorizing scam tokens.

Model limitation. TokenScout, while effective, has its constraints. It can potentially be bypassed by strategic manipulations adapting to its design, also a challenge typical in security models. Its performance also depends heavily on the quality and availability of historical data; thus, novel scam tactics differing from past patterns may slip through. False positives and negatives, inherent

to detection models, could impact its trustworthiness and usability in TokenScout. Additionally, despite our efforts to ensure accurate manual token labeling, the process remains prone to human errors and biases. Such inaccuracies can disrupt the training phase, potentially compromising TokenScout's detection capabilities. Addressing these limitations, especially in adversarial resilience, detection efficiency, and reducing false positive/negative rates will be the focus of future updates.

Token labeling rules. We note that these statistic rules, created from the auditors' knowledge, can not be leveraged for scam token early detection, but to help quickly label tokens. Also, rules can cause false results, like mislabeling benign tokens with high initial transaction taxes designed to prevent fraud. Based on rule-based detection results, each token undergoes thorough review by auditors, ensuring consensus before finalizing the token label. Instead, TokenScout focuses on token transfer activities and aims for real-time monitoring, capable of identifying potential future scams and confirming existing tokens. It offers a more dynamic and forward-looking analysis compared to rule-based methods.

Imbalanced dataset. The proliferation of rugpull tokens has led to a skewed dataset with an overrepresentation of these scams. This imbalance poses a challenge, potentially biasing TokenScout towards over-identifying rugpulls and increasing the risk of false negatives for trustworthy tokens. To mitigate this, we employed strategies like undersampling to balance the dataset before training, and incorporated asymmetric contrastive loss in model training. Besides, we used comprehensive metrics, e.g., precision, recall, FPR, FNR, F1, and BAC, for fair performance reporting.

Efficacy of TokenScout for different scam token stages. We did not evaluate TokenScout's efficacy at various stages of a scam token's lifecycle in our current study. However, we hypothesize that TokenScout's detection efficacy is higher for scam tokens with extensive transaction histories. The increased data allows the model to reveal more transfer traits and behavioral patterns indicative of scams. While TokenScout is capable of performing early detection, its accuracy and robustness improve with more transaction data, making it more efficient at identifying scam tokens with extensive histories. This improvement is due to the richer set of interactions and behaviors that become apparent over time, providing the model with more information to make accurate detections. Future work will include a detailed evaluation of TokenScout's performance at different stages of a scam token's lifecycle to validate this hypothesis.

TokenScout's detection capabilities and limitations. Detecting tokens with embedded vulnerabilities: If a token is maliciously created with embedded vulnerabilities designed to scam investors, TokenScout can help identify it. By analyzing transfer patterns and transaction behaviors of investors and creators using TF-GNN, TokenScout detects suspicious tokens. Modeling temporal dynamics and relational features enables the identification of anomalous behaviors linked to these embedded vulnerabilities aimed at scamming investors. While TokenScout primarily focuses on transaction-based detection, this approach can also highlight potential vulnerabilities indicative of price manipulation attacks, such as price oracle manipulation. However, there are still several exploits that TokenScout cannot detect, including cross-layer attacks, NFT token-related vulnerabilities, DeFi arbitrage, and other sophisticated exploit techniques that may require more advanced detection mechanisms or cross-domain analysis.

7 Related Work

Scam tokens in DeFi. Over past years, there has been a growing interest in studying the scam risk of ERC20 token from various perspectives [31, 37, 50, 63, 67]. Gao et al. studied counterfeit tokens by analyzing aspects such as token popularity, creators, holders, and fraudulent behaviors [37]. Wang et al. conducted an empirical study on the risk of unlimited approval in ERC20 tokens, revealing that 22% of users faced a high risk of approved token theft [63]. Chen et al. used graph analysis to investigate ERC20 tokens and proposed a clustering approach to uncover relationships between tokens and accounts [31]. Federico et al. conducted an empirical study on the ERC20 tokens, focusing on spammers, rugpulls, and sniperbots [29]. Xia et al. [67] and Mazorra et al. [50] explored traditional machine learning methods to flag rugpull tokens after the rugpull occurred, utilizing transaction events related to DEXs and statistical features.

Graph neural networks. Unlike traditional neural networks designed for grid-like data structures, GNNs and their variants operate directly on graph data, enabling them to model the complex interconnections between entities. Inspired by convolutional mechanisms, GNNs leverage message passing to iteratively aggregate and update node representations by considering their neighborhood structure, allowing to capture the local and global structure of graphs [66]. Many GNN variants, e.g., GAT [62], GraphSAGE [38], GIN [70], TGAT [68], and TGN [55] have been explored for different graph task, including directed graphs, heterogeneous graphs, and dynamic graphs. These models have been successfully applied to graph-structured data modeling, e.g., point cloud classification [58], action recognition [57], recommendation systems [36], molecular fingerprints [34], and drug discovery [41].

TokenScout vs. previous scam token detections. (i) *Novel framework and graph learning techniques.* Differ from existing GNNs [51, 68], constrained by fixed time intervals, or lacking adaptability for asynchronous events and long-term dependencies, TF-GNN applied to DTAM, captures the inherent dynamism and swift shifts in token transfers, providing an in-depth grasp of evolving interactions. It is also uniquely designed for graph modeling on TF-GNN with contrastive learning for risk monitoring. (ii) *Early detection.* TokenScout provides real-time alerts for potential token risks before scams occur, advancing beyond previous studies that primarily centered on pre-detecting Ponzi tokens through contract analysis or post-identification of rugpulls. (iii) *Broad scope.* TokenScout extends beyond specific DEXs or token types, encompassing all standard ERC20 marketplace tokens on Ethereum, offering a more comprehensive coverage compared to previous researches.

TokenScout vs. previous GNN-based detections in DeFi. Besides, we would like to highlight that while some previous methods used temporal GNNs in certain [39, 45, 74, 77] for detecting cryptocurrency phishing, TokenScout represents a significant advancement in the field of graph representation learning within temporal GNNs. Specifically, TokenScout introduces several pioneering features that differentiate it from earlier approaches: (i) TokenScout employs a fully temporal GNN that encodes time as a

learnable vector through a novel temporal encoder, enabling the model to robustly capture the dynamic nature of token transfers by considering the intervals between transactions, which are crucial for identifying evolving patterns in scam activities. (ii) TokenScout designs several new learning strategies for GNNs, including comprehensive representation learning for edges (E1-3) and nodes (G1-3). The model learns the representations of token transfer traits, token creators' behaviors, and investors' behaviors, leveraging insights specific to scam tokens. This holistic approach ensures a deeper and more accurate understanding of the underlying patterns associated with scam tokens. (iii) TokenScout identifies and utilizes informative edge and node features for constructing the Dynamic Temporal Attribute Model (DTAM), as shown in Table 1. This contrasts with previous methods, such as TTAGN [45], which primarily relied on static features like in/out degrees. By incorporating a broader and more dynamic set of features, TokenScout ensures more efficient and effective graph learning. In summary, TokenScout not only introduces a novel fully temporal GNN model but also implements advanced GNN learning strategies and an enhanced feature set for nodes and edges. These innovations allow TokenScout to detect the future risk of scam tokens with greater accuracy and efficiency compared to previous methods, which focused mainly on predicting the risk of transactions and addresses.

8 Conclusion

We propose TokenScout, the scam token early-detection framework using temporal GNN. By formulating token transfer activities as a DTAM, incorporating strategic graph learning techniques TF-GNN to capture the complex temporal dynamics of token transfers, and leveraging contrastive learning to learn robust token representations, TokenScout effectively identifies potential scam tokens with high robustness and accuracy, even in adversarial conditions. It captures essential scam token characteristics and alerts real-world financial losses of DeFi tokens. We also contribute the largest reliable scam tokens detection dataset, which will bolster the security and trust of tokens in the future DeFi ecosystem.

Acknowledgments

We thank all anonymous reviewers for their comments that greatly helped improve the presentation of this paper. This research is supported by the National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN). This research is also supported in part by the National Key R&D Program of China under grant No.2021YFB2700200. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Cyber Security Agency of Singapore. The corresponding author is Jing Chen.

References

[1] 2023. TVL (total value locked) across multiple Decentralized Finance (DeFi) blockchains from November 2018 to April 24, 2023. https://www.statista.com/statistics/1272181/defi-tvl-in-multiple-blockchains/.
[2] 2024. blocksec. https://blocksec.com/.
[3] 2024. BscScan. https://bscscan.com.
[4] 2024. certik. https://www.certik.com/.
[5] 2024. coingecko. https://www.coingecko.com/en/all-cryptocurrencies.
[6] 2024. Coinmarketcap. https://coinmarketcap.com/view/defi/.
[7] 2024. Crypto honeypot detector. https://github.com/malvaphe/Crypto-Honeypot-Detector.
[8] 2024. Ethereum honeypot detector. https://honeypot.is/ethereum.
[9] 2024. Ethereum's top gas guzzlers are ponzi schemes. https://cryptonews.net/news/ethereum/384739/.
[10] 2024. Etherscan. https://etherscan.io.
[11] 2024. goplus. https://gopluseco.io/.
[12] 2024. MetaTrust. https://metatrust.io/.
[13] 2024. PeckShield. https://peckshield.com/.
[14] 2024. Pepe coin's rise inspired a series of scam tokens. https://www.binance.com/en/feed/post/512823.
[15] 2024. Ponzi schemes Using virtual Currencies. https://www.sec.gov/files/ia_virtualcurrencies.pdf.
[16] 2024. Quicknode. https://www.quicknode.com/.
[17] 2024. Quicknode. https://www.infura.io/.
[18] 2024. Rugpull. https://coinmarketcap.com/alexandria/glossary/rug-pull.
[19] 2024. SHIB token. https://www.shibatoken.com/.
[20] 2024. sushiswap. https://www.sushi.com/.
[21] 2024. Tether. https://tether.to/.
[22] 2024. Token DGX. https://etherscan.io/address/0x55b9a11c2e8351b4ffc7b11561148bfac9977855.
[23] 2024. Token Tracker (ERC-20). https://etherscan.io/tokens.
[24] 2024. uniswap. https://uniswap.org/.
[25] 2024. USD Coin. https://www.centre.io/usdc.
[26] 2024. web3.py. https://github.com/ethereum/web3.py.
[27] Sharad Agarwal, Gilberto Atondo-Siu, Marilyne Ordekian, Alice Hutchings, Enrico Mariconti, and Marie Vasek. 2023. Short Paper: DeFi Deception—Uncovering the Prevalence of Rugpulls in Cryptocurrency Projects. In *International Conference on Financial Cryptography and Data Security*.
[28] Udit Agarwal, Vinay Rishiwal, Sudeep Tanwar, and Mano Yadav. 2024. Blockchain and crypto forensics: Investigating crypto frauds. *International Journal of Network Management* (2024).
[29] Federico Cernera, Massimo La Morgia, Alessandro Mei, and Francesco Sassi. 2022. Token spammers, rug pulls, and sniperbots: an analysis of the ecosystem of tokens in ethereum and the binance smart chain (bnb). *arXiv:2206.08202* (2022).
[30] Weimin Chen, Xinran Li, Yuting Sui, Ningyu He, Haoyu Wang, Lei Wu, and Xiapu Luo. 2021. Sadponzi: Detecting and characterizing ponzi schemes in ethereum smart contracts. *ACM on Measurement and Analysis of Computing Systems* (2021).
[31] Weili Chen, Tuo Zhang, Zhiguang Chen, Zibin Zheng, and Yutong Lu. 2020. Traveling the token world: a graph analysis of ethereum erc20 token ecosystem. In *Proc. of WWW*.
[32] Weili Chen, Zibin Zheng, Jiahui Cui, Edith Ngai, Peilin Zheng, and Yuren Zhou. 2018. Detecting ponzi schemes on ethereum: Towards healthier blockchain technology. In *Proc. of WWW*.
[33] Yizhou Chen, Heng Dai, Xiao Yu, Wenhua Hu, Zhiwen Xie, and Cheng Tan. 2021. Improving ponzi scheme contract detection using multi-channel textcnn and transformer. *Sensors* (2021).
[34] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proc. of NeurIPS*.
[35] Shuhui Fan, Shaojing Fu, Yuchuan Luo, Haoran Xu, Xuyun Zhang, and Ming Xu. 2022. Smart contract scams detection with topological data analysis on account interaction. In *Proc. of CIKM*.
[36] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proc. of WWW*.
[37] Bingyu Gao, Haoyu Wang, Pengcheng Xia, Siwei Wu, Yajin Zhou, Xiapu Luo, and Gareth Tyson. 2020. Tracking counterfeit cryptocurrency end-to-end. In *Proc. of SIGMETRICS*.
[38] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proc. of NeurIPS*.
[39] Hexiang Huang, Xuan Zhang, Jishu Wang, Chen Gao, Xue Li, Rui Zhu, and Qiuying Ma. 2024. PEAE-GNN: Phishing Detection on Ethereum via Augmentation Ego-Graph Based on Graph Neural Network. *IEEE Transactions on Computational Social Systems* (2024).
[40] Taichi Igarashi and Kanta Matsuura. 2024. Scam Token Detection Based on Static Analysis Before Contract Deployment. https://fc24.ifca.ai/wtsc/WTSC24_4.pdf.
[41] Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. 2021. Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *Journal of Cheminformatics* (2021).
[42] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *Proc. of ICLR*.
[43] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. 2016. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Proc. of S&P*.

[44] Kai Li, Darren Lee, and Shixuan Guan. 2023. Understanding the Cryptocurrency Free Giveaway Scam Disseminated on Twitter Lists. In *IEEE International Conference on Blockchain (Blockchain)*.

[45] Sijia Li, Gaopeng Gou, Chang Liu, Chengshang Hou, Zhenzhen Li, and Gang Xiong. 2022. TTAGN: Temporal transaction aggregation graph network for ethereum phishing scams detection. In *Proc. of WWW*.

[46] Zihao Li, Jianfeng Li, Zheyuan He, Xiapu Luo, Ting Wang, Xiaoze Ni, Wenwu Yang, Xi Chen, and Ting Chen. 2023. Demystifying defi mev activities in flashbots bundle. In *Proc. of CCS*. 165–179.

[47] Ruichao Liang, Jing Chen, Kun He, Yueming Wu, Gelei Deng, Ruiying Du Du, and Cong Wu. 2024. PonziGuard: Detecting Ponzi Schemes on Ethereum with Contract Runtime Behavior Graph (CRBG). In *Proc. of ICSE*.

[48] Ruichao Liang, Jing Chen, Cong Wu, Kun He, Yueming Wu, Ruochen Cao, Ruiying Du, Yang Liu, and Ziming Zhao. 2024. Vulseye: Detect Smart Contract Vulnerabilities via Stateful Directed Graybox Fuzzing. *arXiv preprint arXiv:2408.10116* (2024).

[49] Ruichao Liang, Jing Chen, Cong Wu, Kun He, Yueming Wu, Weisong Sun, Ruiying Du, Qingchuan Zhao, and Yang Liu. 2024. Towards Effective Detection of Ponzi schemes on Ethereum with Contract Runtime Behavior Graph. *arXiv preprint arXiv:2406.00921* (2024).

[50] Bruno Mazorra, Victor Adan, and Vanesa Daza. 2022. Do not rug on me: Leveraging machine learning techniques for automated scam detection. *Mathematics* (2022).

[51] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proc. of AAAI*.

[52] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: online learning of social representations. In *Proc. of SIGKDD*.

[53] Kaihua Qin, Jens Ernstberger, Liyi Zhou, Philipp Jovanovic, and Arthur Gervais. 2023. Mitigating decentralized finance liquidations with reversible call options. In *International Conference on Financial Cryptography and Data Security*.

[54] Kaihua Qin, Liyi Zhou, and Arthur Gervais. 2022. Quantifying blockchain extractable value: How dark is the forest?. In *Proc. of S&P*.

[55] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. In *Proc. of ICML*.

[56] Jie Shao, Kai Hu, Changhu Wang, Xiangyang Xue, and Bhiksha Raj. 2020. Is normalization indispensable for training deep neural network? *Advances in Neural Information Processing Systems* 33 (2020), 13434–13444.

[57] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. 2019. Skeleton-based action recognition with directed graph neural networks. In *Proc. of CVPR*.

[58] Weijing Shi and Raj Rajkumar. 2020. Point-gnn: graph neural network for 3d object detection in a point cloud. In *Proc. of CVPR*.

[59] Sandra Siby, Umar Iqbal, Steven Englehardt, Zubair Shafiq, and Carmela Troncoso. 2022. {WebGraph}: Capturing advertising and tracking information flows for robust blocking. In *Proc. of USENIX Security*.

[60] Weisong Sun, Guangyao Xu, Zijiang Yang, and Zhenyu Chen. 2020. Early detection of smart ponzi scheme contracts based on behavior forest similarity. In *Proc. of QRS*.

[61] Christof Ferreira Torres, Mathis Steichen, et al. 2019. The art of the scam: demystifying honeypots in ethereum smart contracts. In *Proc. of USENIX Security*.

[62] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *Proc. of ICLR*.

[63] Dabao Wang, Hang Feng, Siwei Wu, Yajin Zhou, Lei Wu, and Xingliang Yuan. 2022. Penny wise and pound foolish: quantifying the risk of unlimited approval of erc20 tokens on ethereum. In *Proc. of RAID*.

[64] Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. 2022. Cyclic arbitrage in decentralized exchanges. In *Companion Proceedings of the Web Conference 2022*. 12–19.

[65] Cong Wu, Kun He, Jing Chen, Ziming Zhao, and Ruiying Du. 2020. Liveness is not enough: Enhancing fingerprint authentication with behavioral biometrics to defeat puppet attacks. In *Proc. of USENIX Security*.

[66] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *TNNLS* (2020).

[67] Pengcheng Xia, Haoyu Wang, Bingyu Gao, Weihang Su, Zhou Yu, Xiapu Luo, Chao Zhang, Xusheng Xiao, and Guoai Xu. 2021. Trade or trick? Detecting and characterizing scam tokens on uniswap decentralized exchange. In *Proc. of SIGMETRICS*.

[68] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *Proc. of ICLR*.

[69] Jiahua Xu and Benjamin Livshits. 2019. The anatomy of a cryptocurrency pump-and-dump scheme. In *Proc. of USENIX Security*.

[70] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv:1810.00826* (2018).

[71] Zhiqiang Xu, Pengcheng Fang, Changlin Liu, Xusheng Xiao, Yu Wen, and Dan Meng. 2022. Depcomm: Graph summarization on system audit logs for attack investigation. In *Proc. of S&P*.

[72] Zhiju Yang, Weiping Pei, Monchu Chen, and Chuan Yue. 2022. Wtagraph: Web tracking and advertising detection using graph neural networks. In *Proc. of S&P*.

[73] Changjin Zhang. 2023. The analysis of the risks and improvements of erc20 tokens. *Highlights in Science, Engineering and Technology* (2023).

[74] Jiale Zhang, Hao Sui, Xiaobing Sun, Chunpeng Ge, Lu Zhou, and Willy Susilo. 2024. GrabPhisher: Phishing Scams Detection in Ethereum via Temporally Evolving GNNs. *IEEE Transactions on Services Computing* (2024).

[75] Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. 2022. Dynamic graph neural networks for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[76] Yanmei Zhang, Siqian Kang, Wei Dai, Shiping Chen, and Jianming Zhu. 2021. Code will speak: Early detection of Ponzi smart contracts on Ethereum. In *IEEE International Conference on Services Computing*.

[77] Haibin Zheng, Minying Ma, Haonan Ma, Jinyin Chen, Haiyang Xiong, and Zhijun Yang. 2023. Tegdetector: a phishing detector that knows evolving transaction behaviors. *IEEE Transactions on Computational Social Systems* (2023).

[78] Zibin Zheng, Weili Chen, Zhijie Zhong, Zhiguang Chen, and Yutong Lu. 2023. Securing the ethereum from smart ponzi schemes: Identification using static features. *ACM Transactions on Software Engineering and Methodology* (2023).

[79] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V Le, and Arthur Gervais. 2021. High-frequency trading on decentralized on-chain exchanges. In *Proc. of S&P*.