E2E-EmbedDetector: A Lightweight Entity-Embedding Model for Ethereum Phishing Detection

Abhishree Sinha
Department of Computer Science and Engineering,
SRM Institute of Science and Technology, Delhi NCR Campus,
Modinagar, Ghaziabad, Uttar Pradesh, India 201204
as3751@srmist.edu.in

Aashi Srivastava
Department of Computer Science and Engineering,
SRM Institute of Science and Technology, Delhi NCR Campus,
Modinagar, Ghaziabad, Uttar Pradesh, India 201204
as2132@srmist.edu.in

Gautami Bahuguna
Department of Computer Science and Engineering,
SRM Institute of Science and Technology, Delhi NCR Campus,
Modinagar, Ghaziabad, Uttar Pradesh, India 201204
gb8103@srmist.edu.in

Nishchay Khanna
Department of Computer Science and Engineering,
SRM Institute of Science and Technology, Delhi NCR Campus,
Modinagar, Ghaziabad, Uttar Pradesh, India 201204
nk5082@srmist.edu.in

***Abstract*— Phishing attacks pose a significant security issue in Ethereum-based blockchain systems. Existing solutions, like TEGDetector, address these attacks by analysing how transactions evolve over time using Transaction Evolution Graphs (TEGs) constructed via time slicing, followed by a dynamic graph classifier that captures both spatial structure and temporal evolution with learned time coefficients. However, building and managing these graphs across multiple stages makes the overall approach complex and difficult to implement. In this work, we propose E2E-EmbedDetector, a lightweight end-to-end neural classification model that works directly with raw transaction data. The model learns embedding representations for important entities such as From, To, and ContractAddress, and also used two additional numeric features: transactional value and a derived input length. We train and evaluate the model on a balanced dataset of 50,000 Ethereum transaction using an 80/20 stratified split. The model achieves an accuracy of 95.63%, precision of 0.9265, recall of 0.9912, an F1 score of 0.9578, a ROC-AUC score of 0.9915 and a PR-AUC score of 0.9909. These results show that strong phishing can be achieved using a simpler and more practical tabular approach, without relying on complex temporal graph-based networks.**

Keywords—phishing detection, blockchain security, Ethereum transaction, entity embeddings, neural networks.

I. INTRODUCTION

As blockchain based financial systems continue to grow rapidly, they have also seen a parallel increase in malicious activities. Ethereum, the second largest blockchain by market value, processes millions of transactions daily and supports a diverse ecosystem of decentralized applications (dApps), exchanges (DEXs), and smart contract-based services. While this open and transparent environment promotes innovation, it also makes the network vulnerable to phishing attacks. Attackers often create fake contracts and misleading wallet addresses to trick users into sending their funds to illegitimate destinations.

Unlike the conventional mode of web phishing, where URL blocking or domain termination can block threats to a degree, phishing in blockchain networks is far graver since the transactions cannot be reversed. In case a user consents to a malicious transaction unwittingly, then the funds transferred cannot be recovered. This has led to dire necessity of accurate and quick detection systems so that threats are detected before they are processed by users.

The current phishing detection techniques can be divided into two realms. The former concerns creating explicit graph representations of the transaction data, including interaction among entities and its frequency and using graph neural networks (GNNs) to identify the unusual structural patterns [4][5][6]. The second model, which is the TEGDetector [1], is a result of this concept by adding the temporal information. In this approach, a transaction data is separated into time segments and a Transaction Evolution Graphs (TEG) is built on each segment. The behavior of phishing with regard to dynamic changes in response to time is then learned with the help of a dynamic classification model, which adapts temporally to attention. Although these methods are highly performance and methodology, they need extensive preprocessing including time slicing, subgraph building, and multiple stages inferences that complicate their implementation in real time deploying settings.

Practically and complementarily in this work, we can consider a practical and complementary approach by studying whether an end-to-end model, which works directly with raw tabular transaction data, without the graph construction required, can perform well in the detection of phishing. To solve this, we propose E2E-EmbedDetector, a proposed model that is trained on embedding representation that is compact to learn the three important address fields, From, To and ContractAddress that bear important transaction details. These embeddings are shared with two lightweight numerical features, that is, transaction value and input calldata length. Combining these elements into a single system, the model is able to fully capture both relational and behavioral patterns, which can be classified accurately through a single-stage training process.

Contributions:

- We present E2E-EmbedDetector, end-to-end phishing detector, lightweight, and when trained on 50,000 raw transactions, performs well (ROC-AUC of 0.9915 and PR-AUC of 0.9909).
- We develop an open-world encoding scheme which uses a token representing the restrictive unit <UNK> to effectively deal with addresses that are not seen in the training set, and so the model can be deployed to a real-world environment.
- We apply a validation-based threshold to attain an improved precision-recall trade off than a fixed threshold.
- We briefly compare it to TEGDetector [1], point out some of the most critical evaluation problems, including entity and temporal leakage, and recommend a more robust benchmarking practice.

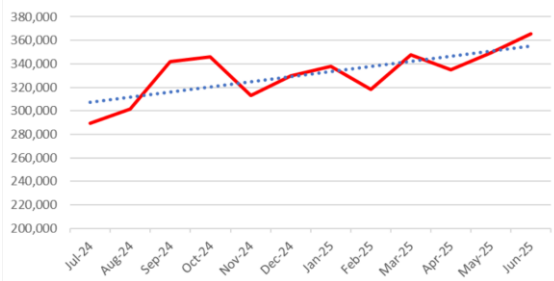

Fig. 1. Reported phishing attacks (3Q2024-2Q2025). The upward trend underscores the urgency of automated blockchain phishing detection. Data from [13].

II. Background and Related Work

A. TEGDetector [1].

TEGDetector is a framework that models time-varying behavior of phishing by splitting transaction histories into consecutive time periods and building a Transaction Evolution Graphs (TEG) during each period. It then integrates spatial graph features with time patterns with a learned time-attention mechanism. This design enables the model to include behavioral drift, which represents how phishing is adapted such that it alters its interaction patterns with targets over time [1].

B. Graph-based detection methods.

Graph neural networks (GNN)-based solutions have taken over as a leading approach in Ethereum phishing detection studies. Li et al., as an example, present the problem as a graph classification problem solved by Chebyshev-GCN on subgraphs of the transactions. Similarly, Li et al. [6] present TTAGN, the model that builds on time transaction patterns aggregation based on an interaction. Wang et al. [10] build transaction subgraph networks (TSGN) on an account level to retrieve local structure, and Xia et al. [12] train address representation by using network embedding models on the entire Ethereum interaction graph.

C. Feature-based and hybrid methods.

Chen et al. [3][4] have created initial baselines with explicit transaction properties, including gas consumption, call distribution, and the number of transactions, input into conventional machine learning classifiers. Kabla et al. [7] compared various ML models with standard feature extraction. Lin et al. [8] introduced a network embedding algorithm to learn account representations and Wen et al. [9] introduced a hybrid deep neural networks model that integrates the different types of features to enhance detection accuracy.

D. Our positioning.

Chen et al. [3][4] established early baselines using explicit transaction features, such as gas usage, call patterns, and transaction counts, fed into traditional machine learning classifiers. Kabla et al. [7] evaluated multiple ML models under standardized feature extraction. Lin et al. [8] proposed a network embedding method to learn account representations, while Wen et al. [9] developed a hybrid deep neural networks that combines multiple features types to improve detection accuracy.

III. Problem Formulation

Let $T = \{t_1, t_2, \ldots, t_N\}$ represent a set of Ethereum transaction records. Each transaction t_i is a structured tuple consisting of:

- Identifier fields: {TxHash, BlockHeight, TimeStamp} that provide provenance and ordering of information.
- Entity fields: {From, To, ContractAddress} representing the sender, receiver, and the contract being invoked.
- Attribute fields: {Value, Input} capturing the transferred Ether amount and the ABI-encoded calldata
- Binary Label Class ϵ {0,1}, where Class = 1 indicates a phishing-related transaction and Class = 0 indicates a benign transaction.

The goal is to train a parametric classifier f_θ: $t_i \rightarrow [0,1]$ that predicts the likelihood of each transaction being linked to phishing, while ensuring the model can generalize to addresses that were not seen during training. This ability to handle unseen addresses is critical in practice, since the Ethereum address space is essentially unlimited and attackers constantly generate new addresses to avoid detection.

Given a labeled training set $T_{train} = \{(t_i, y_i)\}$ and a test set T_{test}, the goal is to train the model by minimizing binary cross-entropy loss while maximizing PR-AUC on a held-out data. Predictions are made as $\hat{y}_i = 1$ if $f_\theta(t_i) \geq \tau^*$, where τ^* is chosen to maximize the F1-score on the validation set.

IV. Dataset

The experiments are based on the publicly available dataset of 50,000 Ethereum transactions obtained via Kaggle. Preprocessing the data and splitting it equally into 25,000 phishing transaction (Class = 1) and 25,000 benign transaction (Class = 0) ensures the strict balance of classes. This 1:1 ratio allows avoiding bias in one of the classes and allows stable training of the models.

The fields in each record are nine. The TxHash is a unique identifier, whereas BlockHeight and TimeStamps give a time-related information concerning the conformation block and the Unix timestamp. Fields that are address related are Like From (sender), To (receiver), and ContractAddress (invoked smart contract). Value field displays the number of Ether transferred; input shows the Tx payload in ABI format and Class is the binary label.

The address fields (From, To and ContractAddress) are very large and they contain considerable number of distinct values making them hard to model using conventional encoding models such as one-hot encoding. To solve this, we employ learned dense embeddings, which translate addresses into fixed-size vectors, which enables the model to learn interaction patterns that are meaningful without increasing the dimensionality of the vectors.

Raw hexadecimal calldata are present in the Input field and have contract-specific encoding formats, and are hard to read. To make this simpler we derive a new feature, the InputLength, which is simply the character length of the hex-coded calldata with the help of len(). Meaningful behavioral patterns are also maintained by this straightforward transformation, with the contract interactions usually possessing unique input lengths, and standard Ether transfers typically producing zero-length inputs.

Before training, preprocessing of the dataset is carried out in three steps. To begin with, all instances of missing ContractAddress are substituted with the token place holder ("None") so that embeddings can be treated as a unified entity. Second, empty strings are used to fill in missing Input entries, and InputLength can be calculated appropriately. Lastly, the numerical characteristics (Value and InputLength) are normalized with z-score. In order to prevent the data leakage, the normalization parameters are computed on the basis of the training set only and used to calculate the normalization parameters on the validation set.

V. Proposed Method: E2E-EmbedDetector

A. Feature Construction

The choice of From, To, and ContractAddress as entity feature is based on the observation that phishing behavior typically revolve around a limited set of malicious address that repeatedly appear in multiple victim transactions. By learning dense embeddings. As a result, addresses involved in phishing tend to form similar representation in the embedding space, enabling the classifier to generalize even when it encounters previously unseen addresses.

B. Safe Entity Encoding

Ethereum operates in an open-world address space, where any 20-byte hexadecimal string can function as a valid address and new addresses are continuously generated. In this setting, a standard label encoder may fail at inference when it encounters unseen addresses. To prevent this, we use a safe encoding strategy by adding a reserved <UNK> (unknown) token to the vocabulary during the training. At inference time, any unseen addresses are mapped to <UNK>, whose embedding is learned during training to represent novel entities. This ensures stable inference without lookup errors and provides a meaningful fallback representation rather than a zero-vector.

C. Model Architecture

Let V_F, V_T, and V_C denote the vocabulary sizes for From, To, and ContractAddress, respectively, each including the <UNK> token. Three embedding matrices $E_F \in R^{\{V_F \times d\}}$, $E_T \in R^{\{V_T \times d\}}$, and $E_C \in R^{\{V_C \times d\}}$ are learned, where d = 32. For a transaction t, the embeddings e_F, e_T, and e_C are obtained and concatenated with the normalized numerical features to form $x = [e_F \| e_T \| e_C \| Value_{std} \| InputLen_{std}] \in R^{98}$.

This is a three-layer MLP whose hidden dimension is 128, 128 and 64. The layers consist of each a linear transformation and dropout (p = 0.3), followed by batch normalization, and the ReLU activation. The last layer generates a legit that is transformed to a probability with a sigmoid function in the inference process. BCEWithLogitsLoss is used to perform training numerically, and it is stable. The model contains about 600K parameters (mostly due to embedding layers) and is thus much lighter than graph-based models.

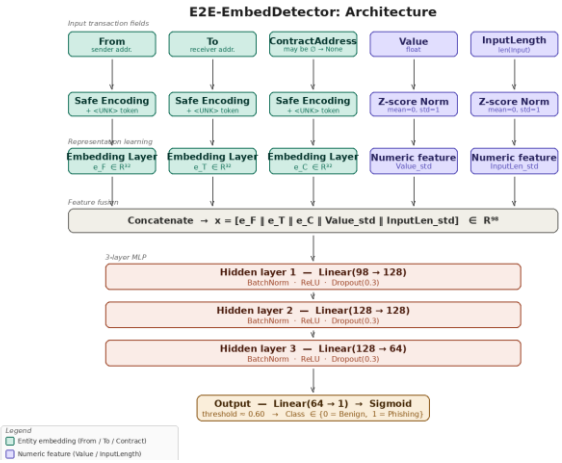

Fig. 2. Architecture of E2E-EmbedDetector. Entity embeddings for From, To, and ContractAddress are concatenated with numeric features and proceed through a 3-layer MLP for classification.

D. Training Configuration

BCEWithLogitsLoss is used to train the model, which combines the sigmoid function with binary cross-entropy calculation in a single numerically stable loss. In order to enhance robustness, the pos_weight parameter is dynamically calculated as the ratio of samples with negative to positive values in the training data; it estimates to 1.0 in balanced dataset utilized in this research, but allows smooth adjustment to imbalanced environments without training architecture changes. The optimization is performed with the Adam optimizer and the learning rate value of 1×10^{-3} and default momentum parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$). An RLReduceLROnPlateau scheduler uses a PR-AUC validator to reduce the learning rate (after each epoch) in case no improvement is detected in three consecutive epochs. The training is performed until a maximum of 50 epochs, early stopping is used after 5 successive epochs without any PR-AUC improvement to guarantee convergence without overfitting.

E. Threshold Optimization

The model gives an output of a score in the form of a sigmoid value ranging 0-1 which is the likelihood that a transaction is a phishing related transaction. A threshold τ is needed to transform this probability into a binary decision. Although the default is set to 0.5 (when it is normal), it is not the best default to use in phishing detection because a false transaction resulting in a false negative is worse than a false transaction resulting in a false positive.

To achieve this, we threshold-search $\tau \in [0.1, 0.9]$ with a step size of 0.01 on the validation set and choose the maximizing threshold of the F1-score i.e. $\tau^* = \text{argmax}_\tau F1(\tau)$ and in the given case, this would be 0.60. The increased threshold is an indication of more certain phishing prediction preference, which can enhance the general quality of detection.

VI. Experimental Setup

A. Train/Validation Split

A stratified random split is used to divide the labeled dataset into training (80%), and validation (20%), where the original proportion of classes (1:1) is maintained in each of the two portions. This will prevent unbalanced validation splits, which otherwise would result in untrustworthy estimates of precision and recall.

Two big limitations of this split strategy are recognised by us. First, it is not address disjointness, i.e. the same Ethereum

address may be represented in both the training and validation sets. Because the model learns address-specific embeddings, this overlap may result in partial memorization and an overestimation of performance in case of complete face-to-face assessment using purely unseen addresses. Second, the split is not time-ordered, since some of the training transactions can be later than the ones in the validation set. This may bring about leakage of future behavior unintentionally. These two problems have been observed in previous studies on phishing and fraud detection [1][8], and underscore the necessity of more rigorous evaluation setups in future studies.

B. *Evaluation Metrics*

Model performance is evaluated on six metrices: Accuracy, Precision, Recall, F1-score, ROC-AUC, and PR-AUC. Accuracy reflects overall correctness, while precision and recall measure the quality and completeness of phishing detection. The F1 score, defined as the harmonic mean of precision and recall, is used for threshold selection. ROC-AUC evaluates class separation across all threshold, whereas PR-AUC (Average Precision) focuses on performance in the precision-recall space. Among these, PR-AUC is given particular importance, as it better reflects classifier performance in scenarios where maintaining both high precision and high recall is critical.

C. *Hyperparameter Settings*

All parameters are defined in advance and kept fixed during training. The embedding dimensions was set to d = 32 for each entity field, and the MLP used hidden layer sizes of 128, 128, and 64. A dropout rate of p = 0.3 was applied after each ReLU layer. The model size was trained with a batch size of 64 for up to 50 epochs, with early stopping applied is no improvement was observed for 5 consecutive epochs. The learning rate was initialized at 1×10^{-3}, and a ReduceLROnPlateau scheduler (factor 0.5, patience 3) was It was adjusted by ReduceLROnPlateau scheduler (factor 0.5, patience 3) when training. pos weight in BCEWithLogitsLoss was calculated as |Class =0| / |Class = 1| here being 1.0 with balanced dataset. No hyperparameter optimization was done instead, these were set based on common tabular neural network practices and were not optimized to prevent overfitting to the validation data.

VII. RESULTS

Table I gives the summary of validation of E2E-EmbedDetector upon reaching convergence. The model has good discriminating capacity on all measures with a ROC-AUC of 0.9915, PR-AUC of 0.9909 and F1-score of 0.9578 at the optimal decision threshold of $\tau^* \approx 0.60$.

TABLE I. VALIDATION PERFORMANCE OF E2E-EMBEDDETECTOR

Metric	**Value**
Accuracy	0.9563
Precision	0.9265
Recall	0.9912
F1-Score	0.9578
ROC-AUC	0.9915
PR-AUC	0.9909
Best Threshold	~0.60

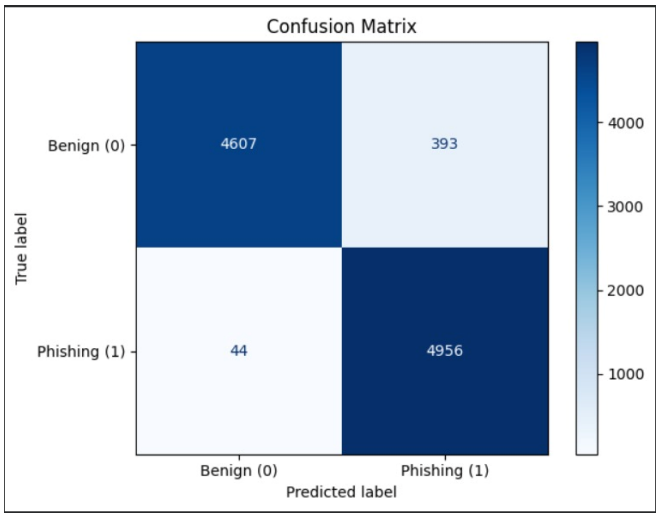

Fig. 3. Confusion matrix of E2E-EmbedDetecor on the validation set (threshold ≈ 0.60). High true-positive rate reflects the model's strong phishing recall.

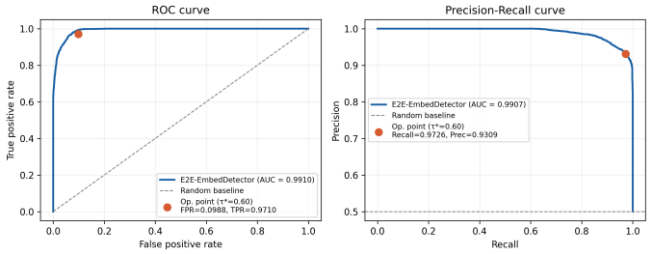

Fig. 4. ROC curve (left) and Precision-Recall curve (right) of E2E-EmbedDetector evaluated on the validation set.

The recall of 0.9912 indicates that the model almost identifies all the phishing transactions, and the percentage of true positives that are picked is less than 1. The difference between the precision (0.927) and the recall (0.991) shows the level of false positives in the model. Specifically, the False Discovery rate is approximately 7.3% i.e. a small percentage of transactions flagged is benign. Practically, such a trade-off is usually tolerable: one can review and filter false positives by the analysts, but false negatives can lead to attack being undetected and even financial loss.

It is well worth noting that the ROC-AUC (0.9915) and PR-AUC (0.9909) are very close. These measures tend to be different where PR-AUC is more prone to false positives particularly when recall values are high. This similarity implies that E2E-EmbedDetector does not show very high precision in a certain threshold of the recall, but has a stable precision throughout the recall range. Although the idea of using a balanced dataset helps in achieving the stable training, in actual implementation, attention should be paid to the selection of thresholds and the consideration of the cost trade-offs in the application of interest.

This feature is useful in practical implementations, where the decision threshold might be required to be set to different security demands. The model has also stable and efficient convergence. The early stopping is done at epoch 25 and the optimal validation checkpoint is epoch 20 (PR-AUC = 0.9909). The comparatively quick convergence is probably explained by the usage of entity embeddings, which are compact and information-rich representatives. Instead of training the model in a direct interaction of all feature interactions, the model can make use of structural patterns that are implicitly learned by address co-occurrence.

VIII. COMPARISON WITH TEGDETECTOR

TABLE II. CONCEPTUAL COMPARISON WITH TEGDETECTOR

Feature / Aspect	TEGDetector	E2E-EmbedDetector (This Work)
Ethereum transaction data	Yes	Yes
Entity info (From/To)	Yes	Yes
Contract address handling	Implicit	Yes (sentinel embedding)
Explicit time slicing	Yes	No
Temporal evolution modeling	Yes	No
TEG graph construction	Yes	No
Dynamic graph classifier	Yes	No
Entity embeddings (core design)	No	Yes
Single-stage end-to-end training	No	Yes
Lightweight inference	No	Yes
Deployment simplicity	Moderate	High

Table II shows the key trade-offs between two approaches. TEGDetector has a more advanced design as it explicitly modelled the temporal dynamics using graph-based representations, and it is therefore an excellent choice to capture the changing behaviour of phishing with time. Instead, E2E-EmbedDetector pays more attention to practical deployment aspects and is based on a one-step training, lightweight preprocessing, and efficient tabular inference. Although it is simple, it is competitive with respect to the tested dataset.

IX. Threats to Validity

A. Entity leakage

The stratified random split adopted in this study cannot assure address disjointness to the training and validation sets. As the model uses a learned embedding per address, the fact that two partitions overlap can allow address-label relationships to be partially memorized instead of being generalised based on the patterns of transactions. This has the potential to give positive performance estimates. To solve this problem the future assessments must use address-disjoints splits, in which no address will occur on partitions hence giving a better evaluation of generalization.

B. Temporal leakage

The random split generates no sequence with regard to the time of the transactions in the BlockHeight and TimeStamp fields. In the real world, phishing behaviors change over time, whereby there is a shift in the strategies and the behavior of the attack, as well as, the patterns of targeting. The ability to mix between transactions across time can thus lead to priori leaks in the model during training and over-optimistic performance. A better option would be a time-based split wherein the model is trained using previous data and tested using subsequent transactions, which will give a more realistic measure of a time-based generalization.

C. Label reliability

All the findings are based on the premise that the Class = 1 is accurate in detecting phishing-related transaction. In practice blacklists, heuristics, or community reports are used to provide ground-truth labels in blockchain datasets which can be not completely accurate. Due to this, it is possible that the dataset may have both false positives (benign transactions that are labeled phishing) and false negative (phishing transactions that are not being labeled). An example of this label noise is that the reported performance measures may be affected by the noise, resulting in the model learning inaccurate decision boundaries.

D. Distribution shift

Phishing on Ethereum is a very dynamic activity that keeps on changing with time, with new attacks strategies like approval-based scams, false contract schemes and flash-loan based exploits, emerging on a regular basis to alter the features of malicious transactions which have to be observed. Consequently, the statistical trends involved in phishing might be different at different points in time, network upgrades, or market circumstances. Therefore, the results of performance based on one, balanced dataset snapshot are likely to be inaccurate with regards to the model on predictive or off-distribution data.

X. Conclusion

In this work, E2E-EmbedDetector, a lightweight and deployable end-to-end neural framework was introduced in order to detect Ethereum phishing. The model is trained on small embeddings of the From, To and ContractAddress fields of raw transaction data and is combined with two normalized numerical features, Value and InputLength. This results in a three-layered MLP, with batch normalization and dropout regularization, to classify. The model on balanced dataset of 50,000 Ethereum transactions with a stratified 80/20 split score an accuracy of 95.63, an F1-score of 0.9578, an ROC-AUC score of 0.9915, and a PR-AUC score of 0.9909 with an optimised threshold of $\tau^* \approx 0.60$.

The primary conclusion of this research is that it is possible to do good phishing detection without explicit graph construction, temporal segmentation, and dynamic graph-based models. E2E-EmbedDetector by modeling address fields as categorical inputs and learning their embeddings end-to-end, learns their relational patterns, which are usually represented by graph-based techniques. Simultaneously, it has a more straightforward single-stage tabular pipeline, which is simpler to implement, can be trained in a shorter time and is more feasible to deploy in real-world systems.

Compared to TEGDetector [1], E2E-EmbedDetector adopts an opposite approach, recording the behavioral evolution with time-sliced Transaction Evolution Graphs and dynamic attention systems. It eliminates explicit temporal modeling and adopts a simpler and more efficient design, based on deployment efficiency and quick inference. All approaches are not more appropriate than the others, but the appropriateness of a given approach depends on the context of application. In the case of real time, high throughput transaction monitoring, the proposed tabular method has feasible benefits. By contrast, temporal graph-based approaches can be richly representative to analyze the changing pattern of phishing offline.

Future work:

This work can be enhanced in a number of ways. First, the model should be tested with address-disjoint and time-splits to obtain a more realistic and conservative evaluation of generalization and be able to compare its performance with the temporal graph-based methods fairer. Second, rerunning the TEGDetector baseline with the same dataset and same evaluation conditions would enable the more stringent quantitative comparison of the conceptual analysis. Third, adding lightweight temporal features of timeStamp sinusoidal encodings, or address-level window-based transaction statistics, or simple sequential models modeling transaction history might be useful to trade off the simplicity of E2E-

EmbedDetector with more elaborate temporal modeling graph-based techniques, perhaps a compromise between the two.

ACKNOWLEDGEMENT

The authors also admit to using Claude, an artificial intelligence-based language model created by Anthropic that can be used to help with language refinement, grammar correction, and formatting to write this manuscript. No tool was selected but only to enhance clarity and presentation quality of text.

The authors developed, verified, and validated all technical content, the design of the experiment, the analysis and conclusions. Full responsibility of the integrity and accuracy of the work in this paper is left to the authors.

The open-source Ethereum data community and publicly available data also credit the authors with the ability to conduct this study.

REFERENCES

[1] H. Zheng, M. Ma, H. Ma, J. Chen, H. Xiong, and Z. Yang, "TEGDetector: A phishing detector that knows evolving transaction behaviors," *IEEE Trans. Comput. Social Syst.*, vol. 11, no. 3, pp. 3988–4000, Jun. 2024.

[2] Seaocn, "TEGDetector (GitHub Repository)," 2021. [Online]. Available: https://github.com/Seaocn/TEGDetector

[3] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu, "Phishing scam detection on Ethereum: Towards financial security for blockchain ecosystem," in *Proc. 29th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2020, pp. 4506–4512.

[4] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, and Z. Zheng, "Phishing scams detection in Ethereum transaction network," *ACM Trans. Internet Technol.*, vol. 21, no. 1, pp. 1–16, 2021.

[5] P. Li, Y. Xie, X. Xu, J. Zhou, and Q. Xuan, "Phishing fraud detection on Ethereum using graph neural network," in *Proc. Int. Conf. Blockchain and Trustworthy Systems (BlockSys)*, Springer, 2022, pp. 362–375.

[6] S. Li, G. Gou, C. Liu, C. Hou, Z. Li, and G. Xiong, "TTAGN: Temporal transaction aggregation graph network for Ethereum phishing scams detection," in *Proc. ACM Web Conf. (WWW)*, 2022, pp. 661–669.

[7] A. H. H. Kabla, M. Anbar, S. Manickam, and S. Karuppayah, "Eth-PSD: A machine learning-based phishing scam detection approach in Ethereum," *IEEE Access*, vol. 10, pp. 118043–118057, 2022.

[8] Z. Lin, X. Xiao, G. Hu, Q. Li, B. Zhang, and X. Luo, "Tracking phishing on Ethereum: Transaction network embedding approach for accounts representation learning," *Comput. Secur.*, vol. 135, p. 103479, 2023.

[9] T. Wen, Y. Xiao, A. Wang, and H. Wang, "A novel hybrid feature fusion model for detecting phishing scam on Ethereum using deep neural network," *Expert Syst. Appl.*, vol. 211, p. 118463, 2023.

[10] J. Wang, C. Chen, P. Yu, S. Hu, X. Chen, and Q. Luo, "TSGN: Transaction subgraph networks for identifying Ethereum phishing accounts," in *Proc. Int. Conf. Blockchain and Trustworthy Systems (BlockSys)*, Springer, 2021, pp. 187–200.

[11] Y. Xia, J. Liu, and J. Wu, "Phishing detection on Ethereum via attributed ego-graph embedding," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, pp. 2538–2542, 2022.

[12] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, and Z. Zheng, "Who are the phishers? Phishing scam detection on Ethereum via network embedding," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, pp. 1156–1166, 2022.

[13] Anti-Phishing Working Group (APWG), "Phishing Activity Trends Report, Q2 2025," 2025. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q2_2025.pdf