DeepTx: Real-Time Transaction Risk Analysis via Multi-Modal Features and LLM Reasoning

Yixuan Liu
Nanyang Technological University
Singapore
liuy0255@e.ntu.edu.sg

Xinlei Li
Nanyang Technological University
Singapore
xinlei003@e.ntu.edu.sg

Yi Li
Nanyang Technological University
Singapore
yi_li@ntu.edu.sg

***Abstract*—Phishing attacks in Web3 ecosystems are increasingly sophisticated, exploiting deceptive contract logic, malicious frontend scripts, and token approval patterns. We present DeepTx, a real-time transaction analysis system that detects such threats before user confirmation. DeepTx simulates pending transactions, extracts behavior, context, and UI features, and uses multiple large language models (LLMs) to reason about transaction intent. A consensus mechanism with self-reflection ensures robust and explainable decisions. Evaluated on our phishing dataset, DeepTx achieves high precision and recall (demo video: https://youtu.be/4OfK9KCEXUM).**

***Index Terms*—Blockchain security, Phishing detection, Transaction semantics**

I. Introduction

Blockchain users frequently interact with decentralized applications (DApps) through Web3 wallets and browser-based frontends. These interactions often require users to sign transactions that are encoded in low-level calldata, which hides critical semantic information such as the actual destination address, function being called, or asset being transferred. As a result, attackers increasingly exploit this semantic gap by crafting phishing transactions that mislead users through benign-looking interfaces but invoke malicious behaviors after signing.

A recent example of such an attack occurred in February 2025, when Bybit, one of the largest cryptocurrency exchange in the world, suffered a loss of over 400,000 ETH (around $1.5 billion) due to a manipulated wallet interface [1]. The attacker injected a forged User Interface (UI) that displayed a legitimate transaction to the operator, while the signed calldata secretly executed a privileged operation such as transferring ownership to a malicious contract. This enabled subsequent unauthorized asset withdrawals. The attack bypassed all traditional on-chain protections and signature verifications, highlighting a critical security risk arising not from contract vulnerabilities, but from misleading user interactions and transaction representations.

Several tools have been developed to mitigate phishing transactions, each adopting different detection strategies with varying levels of openness and effectiveness. PTXPhish [2] is an open-source tool that analyzes behavioral and contextual features within transactions. However, its rule-based design lacks adaptability and often fails to detect previously unseen attack vectors. In contrast, ScamSniffer [3] and Pocket Universe [4] focus on call data simulation, matching results against hard-coded patterns. While ScamSniffer partially discloses its block lists and browser extension code, both tools retain proprietary detection logic, limiting their transparency and extensibility. Forta [5], a decentralized alerting system, employs community-operated bots to monitor live on-chain activity. Although Forta supports real-time data streams, its alerts are typically issued only after transactions have been mined, and together with other tools that rely on static rules or post-hoc pattern recognition, it offers limited protection against phishing tactics involving UI forgery, calldata obfuscation, or execution flows at the time of signing.

In this paper, we propose DeepTx, a tool that simulates user-intended transactions as if they were already signed, and extracts multi-modal features including behavioral traces, contextual behaviors, and user interface indicators. It then uses LLMs to analyze the transaction's intent and potential security risks. The system generates a natural language risk report that includes a severity level, explanatory description, and suggested user action. To enhance reliability, DeepTx performs consensus checking across multiple LLMs and applies self-reflection techniques to resolve inconsistencies or ambiguous results.

We summarize our contributions as follows:

- We propose **DeepTx**, a real-time transaction semantics analysis tool that detects phishing and deceptive behaviors before user confirmation.
- We design a multi-modal feature extraction pipeline, incorporating behavioral traces, contextual signals, and UI metadata from simulated user transactions.
- We integrate large language models with a consensus checking and self-reflection mechanism to produce reliable and explainable security assessments.
- We publicly release both the tool and the accompanying dataset, which captures the full phishing lifecycle—including frontend interaction scripts, confirmed victim transactions, and detailed call chain data. All resources are available at https://github.com/yxsec/DeepTx.

II. System Design and Implementation

DeepTx performs pre-signing transaction simulation and multi-perspective analysis to detect phishing and deceptive behaviors. It extracts features from transaction execution, evaluates contextual and UI signals, queries known malicious indicators, and synthesizes a final risk assessment using a consensus-guided LLM reasoning module. Figure 1 outlines the overall architecture.

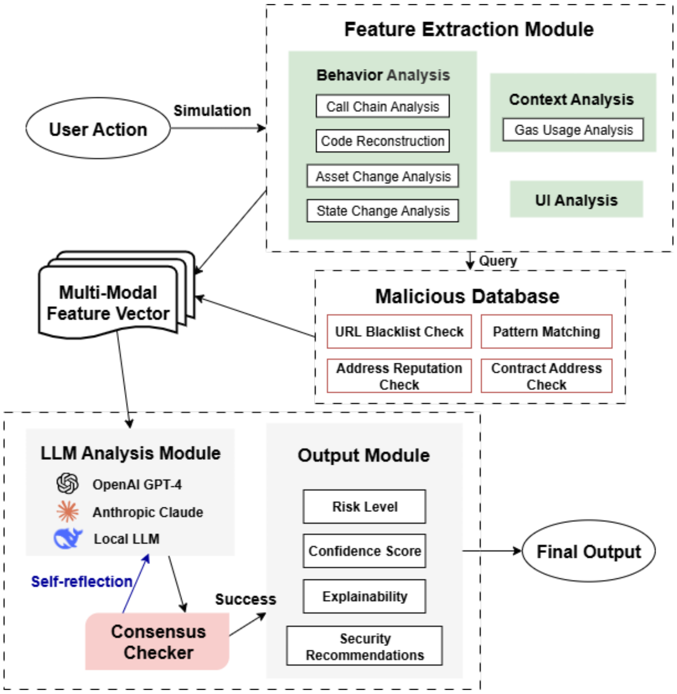


Fig. 1. Overview of DeepTx

A. System Overview

When a user initiates a transaction, DeepTx simulates its execution in a forked EVM state. It extracts semantic features from the trace, including behavioral effects (e.g., call chain, token flows, and state changes), contextual signals (e.g., gas usage), and optional UI features (e.g., JavaScript constructing the transaction). These features are augmented by querying a malicious database with blacklists and pattern rules.

The extracted information is encoded into a multi-modal feature vector, partitioned into four modules (`behavior`, `context`, `UI`, `database`), each assigned a custom weight summing to one. This vector is submitted to n LLMs, each producing a structured response containing the transaction's risk level, confidence, explanation, and user recommendations.

A consensus mechanism ensures better robustness. If the LLMs agree, one model summarizes the results into the final decision. If disagreement occurs, DeepTx enters a self-reflection phase where each model is re-prompted with others' responses as counterexamples. If no agreement is reached after a fixed number of iterations, DeepTx falls back to weighted voting based on confidence scores. The final output is composed from the last round's outputs and shown to the user.

B. Transaction Simulation

DeepTx uses local or remote simulation to replay transactions. It supports Foundry's `cast run` [6] with Anvil and Tenderly's simulation API [7]. These provide execution traces, calldata, internal calls, and storage changes, enabling downstream semantic analysis. No state is broadcast on-chain, ensuring safety.

C. Feature Extraction Module

After simulating the transaction, DeepTx extracts a structured set of semantic features to characterize its intent and effect. The extracted features are grouped into three categories: behavioral, contextual, and UI-related. These collectively form a multi-modal representation that supports subsequent LLM reasoning.

1) Behavioral Features: Behavioral analysis captures the concrete effects of the transaction on the EVM state. DeepTx extracts the full call chain from the execution trace, including all internal and external function invocations, along with the corresponding code segments executed. For each contract involved in the call chain, the system retrieves its code: if the contract is verified, DeepTx fetches the source-level functions using the Etherscan API [8]; otherwise, it uses Heimdall [9] to decompile the bytecode and approximate its logic.

Asset-related actions are also identified. DeepTx tracks native ETH and ERC-20 token transfers by analyzing value fields and standard interfaces, such as `transfer` and `transferFrom`. Additionally, storage writes (`SSTORE`) are collected and analyzed to detect role updates, ownership changes, and balance modifications. These behaviors offer insight into the transaction's operational intent.

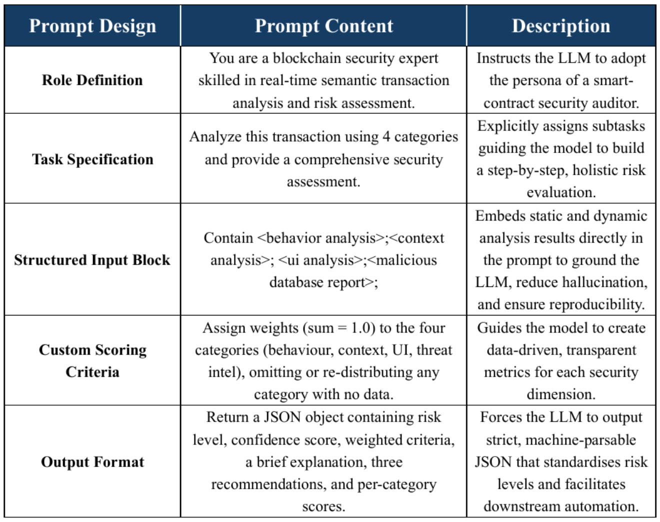
2) Contextual Features: Contextual features capture the transaction environment and help detect anomalous or deceptive behavior. DeepTx compares the gas limit and actual usage, flagging transactions with excessive unused gas as potentially misleading. It also analyzes the effective gas price relative to the base fee to detect possible transaction acceleration or frontrunning. Sender address and nonce patterns are used to identify rapid transaction sequences, which may result from automation or repeated confirmations triggered by phishing interfaces.

3) UI Features: If the transaction is initiated from a DApp frontend, DeepTx optionally performs interface-level analysis. It statically analyzes the associated JavaScript code to identify logic responsible for calldata construction and signature initiation, including inline and external scripts when available.

To support phishing detection, DeepTx also extracts the interaction page's URL and main domain. These signals are combined with threat intelligence data and encoded into the feature vector. UI analysis is performed only when such metadata is accessible at the time of simulation.

D. Malicious Database

DeepTx queries multiple malicious data sources to enrich semantic reasoning. These include: (1) ScamSniffer-based domain and address blacklists, (2) contract address tags, and (3) manually defined calldata or function selector patterns indicative of scams. These results are incorporated into the LLM reasoning module.



Prompt Design	Prompt Content	Description
Role Definition	You are a blockchain security expert skilled in real-time semantic transaction analysis and risk assessment.	Instructs the LLM to adopt the persona of a smart-contract security auditor.
Task Specification	Analyze this transaction using 4 categories and provide a comprehensive security assessment.	Explicitly assigns subtasks guiding the model to build a step-by-step, holistic risk evaluation.
Structured Input Block	Contain <behavior analysis>;<context analysis>; <ui analysis>;<malicious database report>;	Embeds static and dynamic analysis results directly in the prompt to ground the LLM, reduce hallucination, and ensure reproducibility.
Custom Scoring Criteria	Assign weights (sum = 1.0) to the four categories (behaviour, context, UI, threat intel), omitting or re-distributing any category with no data.	Guides the model to create data-driven, transparent metrics for each security dimension.
Output Format	Return a JSON object containing risk level, confidence score, weighted criteria, a brief explanation, three recommendations, and per-category scores.	Forces the LLM to output strict, machine-parsable JSON that standardises risk levels and facilitates downstream automation.

Fig. 2. LLM prompt template used in DeepTx (generalized for n models)

Algorithm 1: Consensus Checker with Self-Reflection and Weighted Voting (generalized to n models)

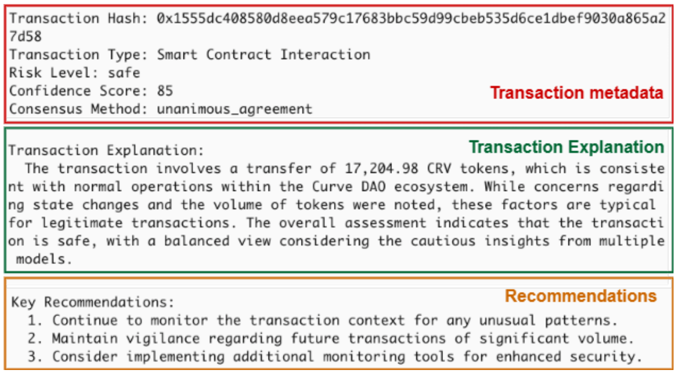
```
Input: Initial LLM outputs ${O_1, O_2, ..., O_n}$
Output: Final consensus decision $O^*$
Initialize $R \leftarrow 0$ (round counter), $M \leftarrow 3$ (default max rounds)
while $R < M$ do
    Extract predicted risks $\{r_i\}_{i=1}^n$ and confidences $\{c_i\}_{i=1}^n$ from $\{O_i\}$
    if $r_1 = r_2 = \cdots = r_n$ then
        $O^* \leftarrow$ summarize($O_1, O_2, \ldots, O_n$) via a primary LLM
        return $O^*$
    // Perform self-reflection with counterexamples
    for $i \in \{1, \ldots, n\}$ do
        Let $O_{\text{own}} \leftarrow O_i$, $O_{\text{counter}} \leftarrow \{O_j \mid j \neq i\}$
        $O'_i \leftarrow$ selfReflect($O_{\text{own}}, O_{\text{counter}}$)
    Update all $O_i \leftarrow O'_i$ for $i \in \{1, \ldots, n\}$
    $R \leftarrow R + 1$
// Fallback: weighted voting by confidence
Initialize score map $S[r] \leftarrow 0$ for all possible labels $r$
for $i \in \{1, \ldots, n\}$ do
    $S[r_i] \leftarrow S[r_i] + c_i$
$r^* \leftarrow \arg\max_r S[r]$
return any $O_i$ such that $r_i = r^*$
```

E. Semantic Reasoning with LLMs

The final stage is to classify whether the transaction is phishing or malicious, estimate its risk level and intent, and provide a human-interpretable explanation before user confirmation.

1) Prompt Design: To support reliable and explainable reasoning, DeepTx constructs a structured prompt that encodes all available information, including the behavior trace, gas context, transaction-specific code snippets, frontend scripts, and threat intelligence from the malicious database. The prompt assigns the LLM the role of a blockchain security analyst and specifies an evaluation task across four categories: behavior, context, UI, and database indicators.

As illustrated in Figure 2, the prompt includes a strict output schema, requiring each model to return a JSON object containing the predicted risk label (`safe`, `suspicious`, or `malicious`), a numerical confidence score, a justification for the decision, a transaction summary, feature importance weights summed to 1, and a list of actionable security recommendations. This standardization ensures consistency across different models and supports consensus checking.



```
Transaction Hash: 0x1555dc408580d8eea579c17683bbc59d99cbeb535d6ce1dbef9030a865a2
7d58
Transaction Type: Smart Contract Interaction
Risk Level: safe
Confidence Score: 85
Consensus Method: unanimous_agreement
```

```
Transaction Explanation:
 The transaction involves a transfer of 17,204.98 CRV tokens, which is consiste
nt with normal operations within the Curve DAO ecosystem. While concerns regardi
ng state changes and the volume of tokens were noted, these factors are typical
for legitimate transactions. The overall assessment indicates that the transacti
on is safe, with a balanced view considering the cautious insights from multiple
 models.
```

```
Key Recommendations:
  1. Continue to monitor the transaction context for any unusual patterns.
  2. Maintain vigilance regarding future transactions of significant volume.
  3. Consider implementing additional monitoring tools for enhanced security.
```

Fig. 3. Final analysis summary produced by DeepTx. The output provides a human-readable explanation, transaction-specific confidence score, and actionable recommendations for user decision making.

2) Multi-Model Reasoning and Consensus: The structured prompt is submitted to n LLMs—such as GPT-4, Claude, local fine-tuned models, or other security-focused systems—which independently generate risk assessments. If all models produce consistent risk labels, one LLM is selected to summarize the results into a final user-facing report. Otherwise, DeepTx initiates a self-reflection phase, as outlined in Algorithm 1. In this phase, each model is re-prompted with the outputs of the others provided as counterarguments and asked to reassess its original decision. This iterative reasoning process continues until either a consensus is achieved or a predefined round limit M is reached.

3) Fallback Voting Mechanism: If no consensus is reached after M rounds of self-reflection, DeepTx falls back to a weighted voting strategy, as shown in the second part of Algorithm 1. The final risk label is selected based on the cumulative confidence-weighted votes for each class. Specifically, the total score for each candidate label r is computed as:

$$\text{Score}(r) = \sum_{i=1}^{n} \mathbb{I}[r_i = r] \cdot c_i$$

where r_i is the label assigned by model i with associated confidence c_i, and $\mathbb{I}[\cdot]$ is the indicator function. The final predicted label r^* is selected as:

$$r^* = \arg\max_r \text{Score}(r).$$

The output from any model satisfying $r_i = r^*$ is used as the final decision. The system merges all available outputs from the final round to produce a user-facing summary that includes supporting explanations, risk rationale, and suggested actions.

F. Output Module

While the user-facing summary shown in Figure 3 provides a concise view of the transaction's risk level, explanation, and recommendations, DeepTx also generates a corresponding structured JSON report that contains additional information not visible in the summary view. Specifically, the JSON file includes detailed component-level scores and reasoning across four semantic dimensions—behavior patterns, gas context, UI indicators, and threat intelligence—each with an assigned weight reflecting its contribution to the final decision. It also records metadata from the consensus process, such as whether the decision was reached unanimously or through weighted voting, the number of self-reflection rounds used, and the identity of the primary model that produced the final output. This structured format supports logging, auditing, and integration into automated security workflows.

III. Tool Usage

DeepTx can be used after installing Python dependencies and setting environment variables such as `RPC_URL`, `ETHERSCAN_API_KEY`, and `OPENAI_KEY`. Given a transaction prepared on the RPC node, run `python3 main.py <transaction_hash>` to retrieve and analyze it. The tool returns a structured report describing its semantic properties.

IV. Preliminary Evaluation

To provide a preliminary evaluation of DeepTx, we constructed a dataset of phishing victim transactions with associated web data and conducted ablation experiments under different UI feature settings.

A. Dataset Collection

We manually labeled 12 phishing cases from two sources, including 5 simulation-based phishing challenges from UnPhishable [10] and 7 real-world cases collected from ScamSniffer database and archived phishing websites [11]. These cases cover a range of semantic and deception strategies, including malicious token approvals, proxy spoofing, and impersonation via fake interfaces. For each transaction, we collected (1) the associated frontend webpage and JavaScript if available, (2) verified or decompiled smart contract code using Heimdall, and (3) the original victim transaction from the chain.

To enable false positive evaluation, we included 2 benign transactions selected from verified protocols (e.g., Aave [12]) and manually confirmed them to be non-malicious.

B. Experiment Setup and Metrics

All transactions were replayed in a forked EVM state to reproduce their original blockchain execution environment. DeepTx then analyzed each transaction through a sequence of analysis stages, including behavioral simulation, contextual feature extraction, UI signal analysis, and LLM-based reasoning.

We evaluated three LLM configurations within DeepTx: `gpt-4o-mini`, `gpt-3.5-turbo`, and `gpt-4o`, all accessed via the OpenAI API. The reported results use `gpt-4o` as the default model.

Each configuration was executed over three independent runs to account for non-determinism in LLM output. A phishing transaction is considered successfully detected if it is labeled as either `malicious` or `suspicious`. A false positive is recorded when a benign transaction is incorrectly flagged as risky. We report the mean and standard deviation of precision, recall, and F1 score across runs.

TABLE I
Detection performance on phishing and benign cases (mean ± std)

Configuration	Precision	Recall	F1 Score
Full DeepTx	**1.00 ± 0.00**	**0.89 ± 0.05**	**0.94 ± 0.03**
No UI & No Database	0.92 ± 0.14	0.22 ± 0.05	0.35 ± 0.06

As shown in Table I, DeepTx achieves high precision and recall under the full configuration. Disabling UI features significantly reduces recall, highlighting the importance of interface-level signals.

V. Conclusion

DeepTx is a real-time transaction analysis system that inspects user-intended EVM transactions before signing. By simulating execution and extracting behavioral, contextual, and interface-level features, the system uses multiple LLMs with consensus and self-reflection to assess transaction intent and potential risks. Evaluation on our phishing dataset demonstrates that DeepTx achieves high precision and recall, offering practical protection against deceptive transactions. Future work will extend the dataset to cover more scenarios and incorporate blind signature analysis to further improve risk assessment.

VI. Acknowledgements

This research is supported by the Singapore Ministry of Education Academic Research Fund Tier 2 (T2EP20224-0003) and the Nanyang Technological University Centre for Computational Technologies in Finance (NTU-CCTF). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of MOE and NTU-CCTF.

References

[1] Bybit, "Bybit security incident: Timeline of events and faqs," 2025, accessed: 2025-07-20. [Online]. Available: https://learn.bybit.com/en/this-week-in-bybit/bybit-security-incident-timeline

[2] Z. Chen, Y. Hu, B. He, D. Luo, L. Wu, and Y. Zhou, "Dissecting payload-based transaction phishing on ethereum," in *Network and Distributed Systems Security (NDSS) Symposium*, 2025.

[3] Scam Sniffer, "Scam sniffer: All-in-one web3 anti-scam solution," 2025, accessed: 2025-07-20. [Online]. Available: https://www.scamsniffer.io/

[4] Pocket Universe, "Pocket universe: Protecting your web3 assets," 2025, accessed: 2025-07-20. [Online]. Available: https://www.pocketuniverse.app/

[5] Forta, "Forta network," 2025, accessed: 2025-07-20. [Online]. Available: https://www.forta.org/

[6] Foundry Project, "Foundry ethereum development framework," https://book.getfoundry.sh/, accessed: 2025-07-20.

[7] Tenderly, "Tenderly simulation api," https://docs.tenderly.co/, accessed: 2025-07-20.

[8] Etherscan, "Etherscan developer apis," https://docs.etherscan.io/, accessed: 2025-07-20.

[9] Jon Becker, "heimdall-rs: Ethereum smart contract analysis tool," https://github.com/Jon-Becker/heimdall-rs, accessed: 2025-07-20.

[10] DeFiHackLabs, "Unphishable," https://github.com/DeFiHackLabs/Unphishable/tree/main, accessed: 2025-07-20.

[11] Internet Archive, "Wayback machine," https://web.archive.org/, accessed: 2025-07-22.

[12] Aave, "Aave: Open source liquidity protocol," https://aave.com/, accessed: 2025-07-23.