Secure Transaction Semantics: Analysis, Vulnerability Detection, and Attack Modeling

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

***Abstract*—Blockchain transactions are often interpreted by off-chain systems through call traces, event logs, and storage modifications. However, these artifacts can diverge from the actual on-chain execution due to semantic mismatches caused by reverts or misleading logs. Existing tools largely assume consistency between observable effects and final state, overlooking semantic mismatches. We present a semantic framework for smart contract security analysis that models and leverages transaction-level semantics to detect vulnerabilities, synthesize attacks, and explain off-chain inconsistencies. Our approach identifies mismatches between real execution effects and intent-oblivious interpretations by off-chain systems. We plan to implement three tools: *PEvent-Catcher* for detecting log forgery vulnerabilities, *RollGain* for synthesizing rollback-based state-reverting attacks, and *DeepTx* for real-time intent detection.**

***Index Terms*—Blockchain security, Smart contracts, Transaction semantics**

I. Introduction

Blockchain transactions entail complex sequences of on-chain operations that produce observable outputs (such as events, internal call traces, and storage modifications) which external systems monitor to determine a transaction's effects. These outputs are critical for blockchain explorers, wallet interfaces, and anti-money laundering (AML) tools to interpret the results and intent of each transaction. However, there exists a semantic gap between the actual execution semantics of a transaction and its off-chain interpretation by these observers. Off-chain monitoring often treats emitted events and traces at face value, assuming they directly correspond to successfully committed state changes, without considering conditional logic or failure paths during execution. In practice, this assumption can be violated—for example, a transaction might emit events or perform internal calls that are later reverted, leading to discrepancies between the recorded artifacts and the final state. This lack of context creates opportunities for adversaries to mislead observers about a transaction's true outcome [1].

This mismatch between observed and actual behavior is not merely theoretical—it has been exploited in real-world attacks. Adversaries can actively exploit the semantic gap by crafting transactions that produce deceptive on-chain outputs. One attack vector is *event forgery*: a malicious contract can emit logs that mimic legitimate protocol events (such as fake ERC-20 `Transfer` or `Approval` events) without actually executing the corresponding token transfer or state update [1]. Since many clients and block explorers display token transfers based solely on events, such *phantom events* can mislead users or monitoring systems into believing a transfer occurred. Another tactic is to use *conditional reverts*: a contract may emit expected events or traces, then intentionally revert the transaction so no state changes persist—even though off-chain observers may still log the intermediate artifacts. These strategies exploit the fact that many off-chain systems interpret logs and traces without access to the complete execution semantics.

Despite these risks, most existing smart contract analysis tools focus on detecting well-known code-level vulnerabilities such as reentrancy, integer overflow, or access control flaws [2]. Numerous analyzers—such as Oyente [3], Securify [4], and Osiris [5]—use symbolic execution or static analysis to uncover contract-level bugs. However, these tools generally assume that observed events and traces correspond to committed changes and do not model the external interpretation pipeline. As a result, they miss semantic inconsistencies that arise from deceptive event emission or revert-based obfuscation.

In light of these limitations, we propose a multi-layer semantic analysis framework to bridge the gap between on-chain execution and off-chain interpretation. Our approach detects inconsistencies between a transaction's actual effects and its observable outputs, synthesizes attack contracts that exploit these mismatches, and classifies transaction intent to support real-time risk assessment. These contributions aim to strengthen the reliability of transaction analysis for both automated tools and end users, ensuring that off-chain interpretations more accurately reflect on-chain reality.

II. Background and Related Work

Smart contract security has evolved in response to empirical failures and shifting attack surfaces. The DAO exploit in 2016 exposed the danger of reentrancy, enabling recursive calls to drain over $60M in Ether and prompting an Ethereum hard fork [6]. Subsequent research has cataloged numerous bugs, including integer overflows, unchecked calls, and improper access control [2], [5]. More recent work highlights subtler flaws in proxy upgrades, delegatecall misuse, and constructor-based impersonation [7], [8], emphasizing the need to reason not only about functional correctness but also about semantic behavior and user-facing effects. Many tools focus on intra-contract bugs, assuming that logs and state are consistent.

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00398

Oyente uses symbolic execution to find reentrancy and arithmetic flaws [3]; Securify applies Datalog-based compliance checking [4]; Slither performs taint and dataflow analysis [9]; and Mythril combines symbolic and concrete execution [10]. However, these tools do not consider how off-chain systems interpret observable outputs like logs or traces.

Beyond control and data-flow vulnerabilities, a growing set of issues arises from how contracts emit logs consumed by external observers. Transactions produce artifacts—logs, traces, and storage diffs—used by wallets and explorers to infer transaction effects. Malicious contracts can emit fake events (e.g., bogus `Transfer` logs) that misrepresent state [1]. Off-chain systems often treat logs as ground truth, even if transactions revert. Such misuse can lead to false user interfaces and misclassification by analytics or AML tools. This challenge has been framed as an on-chain/off-chain synchronization issue. DArcher [11] shows DApps often mishandle off-chain state when transactions revert or reorg. Interfaces may show outdated information, enabling attacks that exploit off-chain trust in incomplete execution data. TXSPECTOR [12] replays transactions to detect known attack patterns but lacks generalization to novel semantic inconsistencies. XScope [13] verifies cross-chain event consistency, and DocCon [14] checks for documentation-code mismatches. However, none of these tools address the issue of forged events or interpretation errors arising from mismatches between execution contexts and off-chain observations. As a result, existing approaches leave significant gaps in detecting vulnerabilities where off-chain systems misinterpret or overlook execution context, making these tools inadequate for capturing the full scope of security risks in smart contracts. This semantic gap between on-chain execution and off-chain interpretation creates exploitable blind spots in contract security, which adversaries can exploit.

III. Proposed Approach

Our approach focuses on analyzing *semantic inconsistencies* between on-chain behavior (e.g., actual storage updates and transaction outcomes) and off-chain interpretations (e.g., event logs consumed by wallets and blockchain explorers). In particular, we target scenarios where off-chain systems fail to accurately represent the actual state changes or events that occur during contract execution. The proposed framework comprises three main components: (1) detecting semantic inconsistencies in event usage, (2) modeling rollback-based attack vectors, and (3) preventing transaction-level semantic attacks via intent classification and execution semantics modeling.

A. Vulnerability Detection

Our vulnerability detection focuses on identifying inconsistencies between logged events and actual contract behavior. In our prior work [1], we systematically classified five types of abnormal event usage patterns: *Event Counterfeiting*, *Inconsistent Logging*, *Contract Imitation*, *Transfer Event Spoofing*, and *Event Handling Errors*. Building on this, we focus here on three on-chain detectable categories: *Event Counterfeiting*, *Inconsistent Logging*, and *Contract Imitation*.

To detect these issues, we develop a hybrid analysis pipeline that operates across three levels of analysis: transaction level, bytecode level, and source code level. At the transaction level, we track event emissions and verify if they correspond to actual state changes and committed storage updates. At the bytecode level, we use decompilation to analyze control flow and data dependencies, ensuring that event emissions are guarded by valid control conditions. At the source code level, we employ Slither for symbolic reasoning to check whether event parameters reflect committed storage and whether emitted events align with the contract's intended behavior. For example, we flag logs that reference storage updates that never occur or events emitted before conditional reverts.

This stage enables scalable vulnerability screening by identifying semantic inconsistencies in event usage, helping developers and auditors discover misleading behaviors.

B. Attack Modeling

Our attack modeling focuses on how the EVM rollback mechanism can be manipulated for both on-chain and off-chain exploits.

a) On-Chain Attack Modeling.: We develop an automated synthesis framework, *RollGain*, that exploits rollback-based state reverting vulnerabilities to extract profit from smart contracts. RollGain begins by reconstructing the contract's source and storage state at a specific block, then builds a Comprehensive Contract Dependency Graph (CCDG) and a Role-based Token Transfer Graph (RTTG) to model execution logic and token flows. It locates profit-bearing paths in the RTTG, encodes call sequences into SMT constraints based on symbolic summaries, and validates them. Sequences that yield net attacker profit upon replay on a forked chain are reported as successful exploits. These attacks exploit the rollback capability to revert unprofitable attempts while persisting only those that yield gains.

b) Off-Chain Attack Modeling.: Rollback behavior can also introduce inconsistencies in how transactions are interpreted by off-chain systems. We formally define two categories of rollback-based off-chain inconsistencies: full rollback, where all state changes and logs are reverted; and partial rollback, where inner call failures (e.g., via `try/catch`) suppress certain effects while committing others. These inconsistencies reveal that off-chain systems often misinterpret rollback semantics, making them vulnerable to manipulation.

C. Attack Prevention

a) Intent-aware Semantic Analysis.: We developed a tool, *DeepTx*, designed to analyze and interpret the intent behind blockchain transactions before user confirmation. DeepTx serves two primary purposes: first, it helps users determine whether a transaction is potentially a phishing attempt, and second, it aids in identifying attack transactions that may exploit vulnerabilities.

The first use case involves real-time analysis of unsigned transactions. DeepTx extracts key features such as calldata,

token flows, call sequences, and event emissions, and then applies semantic reasoning to infer the transaction's intended effect. If the system detects inconsistencies between the expected and observed behavior—such as a transaction masquerading as a legitimate action but actually attempting to approve malicious contracts or transfer funds—it flags the transaction as potentially phishing-related. The tool generates a detailed report for the user, explaining the risks and giving clear recommendations, ensuring the user can make an informed decision before signing the transaction.

The second use case of DeepTx focuses on identifying attack transactions. By analyzing the execution context and comparing it to known attack patterns, DeepTx can detect whether a transaction exploits a vulnerability, such as a overflow or access control. This proactive detection allows security systems to assess potential threats in real time, providing the necessary information to block or flag transactions that could compromise the security of users or smart contract. In this way, DeepTx assists in both detecting attack transactions and preventing them from being executed, offering a defense mechanism for both individuals and platforms.

b) Failure Reason Modeling in Asynchronous Systems.: Beyond the EVM, we also study asynchronous execution environments such as The Open Network, where messages are executed under bounded gas and time conditions. In such systems, a transaction may fail due to message queue overflows, delayed resource allocation, or partial execution failures. We build semantic models to capture these failure modes and evaluate how off-chain observers interpret the incomplete or reordered transaction traces. By explicitly modeling failure semantics and message-passing logic, we extend the semantic definition of a transaction beyond EVM-style atomicity, allowing more accurate reasoning across diverse execution platforms.

Together, these prevention strategies aim to improve the robustness of both on-chain and cross-layer semantic security analysis.

IV. Preliminary Results

Our preliminary work explores the feasibility and effectiveness of applying semantic analysis to detect vulnerabilities, synthesize attacks, and reveal inconsistencies in off-chain interpretations. We organize the results into three components: log-forgery detection, rollback-based exploit analysis, and phishing transaction detection.

A. Log Forgery Detection

We developed *PEventCatcher*, a hybrid analysis tool to detect phantom event vulnerabilities in smart contracts. The system integrates symbolic validation, taint analysis, and transaction-level monitoring to identify three categories of log misuse: event counterfeiting, inconsistent logging, and contract imitation.

The analysis revealed several instances of suspicious logging behavior, including forged, inconsistent, and misleading events. Manual validation confirmed that these represent real vulnerabilities capable of causing off-chain misinterpretation and user deception, which were subsequently disclosed to the project teams. Some of these vulnerabilities were acknowledged and resulted in bug bounties.

B. Rollback-Based Exploits and Off-Chain Misinterpretation

We developed *RollGain*, an automated framework for synthesizing rollback-based exploits. By reconstructing the historical on-chain state of a contract, RollGain builds token transfer graphs, identifies potential profit paths, and generates candidate call sequences using symbolic constraint solving and iterative refinement. When applied to 30 real-world contracts, RollGain identified 22 exploitable cases where attackers could leverage conditional state reversion to secure on-chain profits.

Beyond on-chain analysis, we also investigated how rollback behaviors are interpreted by off-chain systems. We submitted controlled transactions with full or partial rollback to 15 token trackers, RPC providers, and explorers. This revealed 16 misinterpretation bugs, including phantom internal transactions and incorrect balance displays. We reported all issues, five of which were assigned CVE IDs and acknowledged by vendors.

C. Phishing Transaction Detection

We developed *DeepTx*, a real-time transaction analysis tool that simulates unsigned transactions and extracts multimodal features including token flow, call sequences, event emissions, and user interface metadata. This tool integrates both LLM-based semantic inference and rule-based detection to identify phishing behavior prior to transaction confirmation. *DeepTx* achieved second place in the ETHDenver25 Ora Track competition, demonstrating its effectiveness in real-time transaction analysis for detecting phishing attempts and fraudulent behavior.

V. Evaluation Plan

A. Vulnerability Detection Evaluation

We evaluate the precision and recall of *PEventCatcher* using two datasets: (1) a curated benchmark of contracts containing known log misuse cases, and (2) a large-scale set of actively deployed contracts on Ethereum. For each detection result, we perform manual validation and measure how often phantom events can cause incorrect assumptions about contract state, including user deception and oracle misclassification. Additionally, we will compare *PEventCatcher* with existing tools such as SmartAxe, XGuard, and XScope [15], [13], [16]. These comparisons will focus on detecting event counterfeiting, inconsistent logging, and contract imitation, measuring the precision, recall, and false positive rates. We also report the number of confirmed bugs and bug bounty acknowledgements.

B. Attack Synthesis Evaluation

We will benchmark the RollGain framework on a diverse dataset of real-world contracts exhibiting complex token and state interactions. The metrics include: (1) the number of profitable rollback-based attack sequences synthesized, (2) the symbolic feasibility rate (SMT-satisfiable), (3) the execution

success rate on local forks, and (4) the actual profit extracted per attack. We will compare RollGain against existing tools, focusing on attack path generation and exploit feasibility. This will provide a clear picture of RollGain's effectiveness in rollback-based exploit synthesis compared to existing fuzzing and symbolic execution tools. We will also measure the runtime cost and effectiveness of each analysis stage (graph modeling, enumeration, SMT solving, and fork validation).

C. Defense Evaluation and Semantic-Based Classification

We plan to extend DeepTx to support intent-aware classification of transactions. We define three main categories—benign transactions, phishing transactions, and attacks—with subtypes for each. For example, phishing transactions may involve fake token approvals or deceptive airdrops, while attacks may include rollback-based exploits. We will curate labeled datasets for these categories and evaluate detection precision. Our tool will classify transactions by inferring their semantics, and we will assess the effectiveness of our semantic features in identifying malicious or misleading activity by measuring detection accuracy.

To generalize beyond the EVM, we will explore generalization by testing our semantic modeling approach on asynchronous systems such as The Open Network. We will analyze failure modes in message-driven transactions and evaluate how semantic intent modeling can improve off-chain synchronization and UI feedback accuracy.

VI. Dissemination and Community Contribution

We plan to release our tools and datasets as open-source projects to support reproducibility, collaboration, and broader adoption. These tools include *PEventCatcher*, for detecting phantom event vulnerabilities in smart contracts; *RollGain*, for synthesizing rollback-based exploits; and *DeepTx*, for intent-aware transaction detection and phishing analysis. By making these tools publicly available, we aim to encourage further development and integration within the broader security community.

In addition to responsible vulnerability disclosures, which have resulted in CVE assignments and bug bounties, we are developing a generalized semantic extraction framework. This framework enables other tools to reuse our extracted models for transaction intent, flow, and rollback detection. It is designed to be modular and composable, making it easy to integrate into existing EVM analyzers and transaction monitoring systems, improving semantic security across various blockchain platforms.

Our future dissemination efforts will focus on engaging with academic venues, public bug tracking databases, and developer communities. We also aim to contribute to the design of security guidelines and EVM improvement proposals, sharing our insights on semantic-based vulnerabilities to foster more reliable and secure blockchain transaction processing.

References

[1] Y. Liu, Y. Dong, Y. Liu, X. Luo, and Y. Li, "Phantom Events: Demystifying the Issues of Log Forgery in Blockchain," *arXiv preprint arXiv:2502.13513*, 2025.

[2] N. Atzei, M. Bartoletti, and T. Cimoli, "A Survey of Attacks on Ethereum Smart Contracts (SoK)," in *Principles of Security and Trust (POST)*, ser. LNCS, vol. 10204. Springer, 2017, pp. 164–186.

[3] L. Luu, D. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making Smart Contracts Smarter," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2016, pp. 254–269.

[4] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Bünzli, and M. Vechev, "Securify: Practical Security Analysis of Smart Contracts," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2018, pp. 67–82.

[5] C. F. Torres, J. Schütte, and R. State, "Osiris: Hunting for Integer Bugs in Ethereum Smart Contracts," in *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC)*. ACM, 2018, pp. 664–676.

[6] M. A. Mehar, C. L. Shier, A. Kenai, W. L. Quesnelle, D. L. Hyland-Wood, P. K. C. Lee, G. Srivastava, and H. Safavi-Naini, "Understanding a revolutionary and flawed grand experiment in blockchain: The dao attack," *Journal of Cases on Information Technology (JCIT)*, vol. 21, no. 1, pp. 19–32, 2019.

[7] I. Grishchenko, M. Maffei, and C. Schneidewind, "A semantic framework for the security analysis of ethereum smart contracts," 2018.

[8] Y. Liu, W. Meng, and Y. Zhang, "Detecting smart contract state-inconsistency bugs via flow divergence and multiplex symbolic execution," *Proc. ACM Softw. Eng.*, vol. 2, no. FSE, Jun. 2025. [Online]. Available: https://doi.org/10.1145/3715712

[9] J. Feist, G. Grieco, and A. Groce, " Slither: A Static Analysis Framework for Smart Contracts ," in *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2019, pp. 8–15. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/WETSEB.2019.00008

[10] ConsenSys, "Mythril - security analysis tool for ethereum smart contracts," https://github.com/ConsenSys/mythril, 2018.

[11] W. Zhang, L. Wei, S. Li, Y. Liu, and S.-C. Cheung, "Darcher: detecting on-chain-off-chain synchronization bugs in decentralized applications," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 553–565. [Online]. Available: https://doi.org/10.1145/3468264.3468546

[12] M. Zhang, X. Zhang, Y. Zhang, and Z. Lin, "Txspector: uncovering attacks in ethereum from transactions," in *Proceedings of the 29th USENIX Conference on Security Symposium*, ser. SEC'20. USA: USENIX Association, 2020.

[13] J. Zhang, J. Gao, Y. Li, Z. Chen, Z. Guan, and Z. Chen, "Xscope: Hunting for cross-chain bridge attacks," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '22. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3551349.3559520

[14] C. Zhu, Y. Liu, X. Wu, and Y. Li, "Identifying solidity smart contract api documentation errors," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '22. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3551349.3556963

[15] Z. Liao, Y. Nan, H. Liang, S. Hao, J. Zhai, J. Wu, and Z. Zheng, "Smartaxe: Detecting cross-chain vulnerabilities in bridge smart contracts via fine-grained static analysis," *Proc. ACM Softw. Eng.*, vol. 1, no. FSE, Jul. 2024. [Online]. Available: https://doi.org/10.1145/3643738

[16] K. Wang, Y. Li, C. Wang, J. Gao, Z. Guan, and Z. Chen, "Xguard: Detecting inconsistency behaviors of crosschain bridges," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024. New York, NY, USA: Association for Computing Machinery, 2024, p. 612–616. [Online]. Available: https://doi.org/10.1145/3663529.3663809