Real-Time Transaction-Level Exploit Detection at Blockchain Scale

Tamer Abdelaziz, Ph.D.

tamer.m@nyu.edu
tamer-abdelaziz.github.io

Abstract

Multi-transaction exploits (flash-loan, sandwich, oracle-manipulation) evade static tools because they require real-time reasoning over ordered cross-contract effects at mempool scale. Existing detectors suffer from high latency or poor recall on novel sequences. This project develops a two-stage streaming system (lightweight sketch filter + TGNN-transformer) with adversarial training and cost-sensitive calibration to achieve sub-second, high-recall detection on production nodes. Extending our prior contract-analysis research [2, 3, 4, 5, 6, 7], it builds on recent DeFi-tailored monitoring frameworks [8, 9]. Deliverables include open-source components and mitigation policy co-design with relayers and exchanges.

Problem Formalization

Existing real-time detectors are limited by either prohibitive latency on full TGNN inference or insufficient semantic modeling of cross-contract flows, leading to missed sophisticated exploits and high operational cost. This project solves these by a two-stage pipeline with streaming sketches for early filtering and on-demand TGNN reasoning, combined with explicit adversarial training under realistic mempool perturbations.

Real-time detection of exploitative transaction sequences (including flash-loan based, sandwich, oracle-manipulation, and cross-contract chain-of-call attacks) is critical for reducing loss and enabling defensive interventions. This project develops a low-latency, high-recall detection system that operates on mempool and block data streams and reasons about ordered sequences of transactions and their cross-contract effects.

We formalize the problem as follows. Let $\mathcal{T} = (\tau_1, \tau_2, \dots)$ be the stream of incoming transactions; each transaction τ_i executes against a current global state σ_{i-1} and yields $\sigma_i = \delta(\sigma_{i-1}, \tau_i)$. An exploit event is a finite subsequence $\tau_{i:i+k}$ such that the composed execution produces an undesirable postcondition:

$$\exists i, k \quad \Phi\big(\sigma_{i-1}, \tau_{i:i+k}\big) = \texttt{true},$$

where Φ is a predicate capturing property violations (e.g., loss to a liquidity pool beyond an expected threshold, or unauthorized transfer). The detection task is an online decision function f that at time t maps a sliding window $W_t = \tau_{t-L+1:t}$ (plus mempool context) to a set of alert scores $s \in [0, 1]$ with stringent latency bound ℓ.

We model transaction interactions as dynamic graphs where nodes represent accounts/contracts and directed hyperedges represent call sequences and value flows during a transaction. The core detection model uses temporal graph neural networks (TGNNs) and sequence models to predict $\Pr(\Phi|W_t)$, combining features from call graphs, value flows, oracle prices, and storage-write patterns. The TGNN update across time step t for node v uses:

$$\mathbf{h}_v^{(t)} = \mathrm{GRU}\big(\mathbf{h}_v^{(t-1)}, \mathrm{MP}\big(\{\mathbf{h}_u^{(t-1)} : u \in \mathcal{N}_v^{(t)}\}, \mathbf{x}_v^{(t)}\big)\big),$$

where MP is a message-passing aggregator and $\mathbf{x}_v^{(t)}$ encodes node-level features (balance deltas, storage access counts, role indicators).

Practical deployment requires extremely low latency and incremental updates. We therefore design a two-stage pipeline: (1) a streaming lightweight anomaly filter that computes streaming statistics and graph-summarization sketches to flag suspicious windows, and (2) an on-demand, higher-capacity TGNN + transformer model that executes only for flagged windows. The pipeline minimizes false positives by calibrating thresholds using cost-sensitive objectives: let C_{FP} and C_{FN} denote costs; the objective minimizes $C_{\mathrm{FP}} \cdot \mathrm{FP} + C_{\mathrm{FN}} \cdot \mathrm{FN}$ subject to latency $\leq \ell$.

Robustness is essential because adversaries intentionally craft sequences to evade detectors. We therefore incorporate adversarial training by explicitly solving a min-max objective:

$$\min_{\theta} \max_{\Delta \in \mathcal{A}} \mathbb{E}_{W \sim \mathcal{D}} \big[\ell(f_\theta(W + \Delta), y_W) \big],$$

where $\mathcal{A}$ denotes allowable perturbations (semantic-preserving code transformations, slight reorderings within mempool bounds), and ℓ is a loss function.

Evaluation uses a historical reconstruction of known multi-transaction attacks and prospective red-team simulations. Metrics include time-to-detect against exploit commit time, precision/recall, and computational cost per transaction. We also quantify economic impact reduction by simulating mitigation actions (e.g., alerting relayers, delaying inclusion, or gas-price penalization).

This project benefits from collaboration with exchanges, relayers, and node operators who can provide mempool telemetry and realistic latencies. We seek partners interested in deploying testbed integrations and co-designing mitigation policies; open-source components will provide a baseline for future research and production adoption.

References

[1] T. Abdelaziz, A. Sedky Adly, B. Rossi, and M.-S. Mostafa. Identification and assessment of software design pattern violations. *Informatics Bulletin*, 1(2):6–13, 2019. https://fcihib.journals.ekb.eg/article_107517_62d89752f7d871844b0e5dd1601da4f5.pdf.

[2] T. Abdelaziz and A. Hobor. Smart learning to find dumb contracts. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1775–1792, 2023. https://www.usenix.org/conference/usenixsecurity23/presentation/abdelaziz.

[3] T. Abdelaziz and A. Hobor. Smart learning to find dumb contracts (extended version). arXiv:2304.10726, 2023. https://arxiv.org/abs/2304.10726.

[4] T. Abdelaziz and A. Hobor. USENIX'23 artifact appendix: Smart learning to find dumb contracts. USENIX Association, 2023. https://www.usenix.org/system/files/usenixsecurity23-appendix-abdelaziz.pdf.

[5] T. Abdelaziz and A. Hobor. Schooling to exploit foolish contracts. In *2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)*, pages 388–395, 2023. https://ieeexplore.ieee.org/document/10338924.

[6] T. A. Abdelmegid Mohamed. Towards secure smart contracts: A deep learning approach for detecting security threats. PhD thesis, National University of Singapore, 2023. https://scholarbank.nus.edu.sg/handle/10635/247301.

[7] T. Abdelaziz, S. Alsaghir, and K. Ali. Where do smart contract security analyzers fall short? *Proceedings of the 2026 IEEE/ACM 23rd International Conference on Mining Software Repositories (MSR)*, 2026.

[8] X. Li et al. Penetrating the Hostile: Detecting DeFi Protocol Exploits through Cross-Contract Analysis. arXiv:2511.00408, 2025. https://arxiv.org/abs/2511.00408.

[9] X. Su et al. From Transactions to Exploits: Automated PoC Synthesis for Real-World DeFi Attacks. arXiv:2601.16681, 2026. https://arxiv.org/abs/2601.16681.

[10] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014. https://ethereum.github.io/yellowpaper/paper.pdf.

[11] L. Luu, D.-H. 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)*, 2016.

[12] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on ethereum smart contracts (soK). In *Proceedings of the 6th International Conference on Principles of Security and Trust (POST)*, 2017.