Foundation Models for Smart Contract Semantics and Vulnerability Reasoning

Tamer Abdelaziz, Ph.D.

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

Abstract

Current smart-contract analyzers rely on shallow syntactic or heuristic features and fail to capture deep EVM operational semantics, gas dynamics, reentrancy interleavings, and cross-contract state dependencies, resulting in high false-negative rates on realistic exploits. This project introduces EVM-FM, an EVM-native multimodal foundation model that jointly embeds bytecode, control-flow graphs, and execution traces under multi-task supervision (classification, contrastive, symbolic alignment, and generative objectives) to enforce semantic consistency. Building upon our team's prior machine-learning work on dumb-contract detection and analyzer evaluation [2, 3, 4, 5, 6, 7], the framework advances beyond recent LLM-based auditors [8, 9] by integrating native EVM trace modeling and symbolic constraints. We outline dataset curation, evaluation axes, and open-source deliverables, inviting collaborations in formal methods and large-scale pretraining.

Problem Formalization

Existing approaches are limited by their dependence on token-level or CFG-only representations that ignore full EVM execution semantics and symbolic state transitions, leading to poor generalization on obfuscated or cross-contract attacks. This project solves these limitations by training a multimodal foundation model with explicit symbolic-alignment and generative-trace objectives, enabling precise vulnerability reasoning and automated exploit synthesis at scale.

This project addresses the gap between large-scale foundation models and precise semantic reasoning of smart contracts. The primary observation is that most existing neural and hybrid analyzers operate on shallow representations (tokens, signatures, or heuristics) and thus fail to capture the operational semantics of the Ethereum Virtual Machine (EVM), gas dynamics, reentrancy interleavings, and cross-contract state dependencies. We therefore propose to design and train an EVM-native foundation model (henceforth *EVM-FM*) that jointly models bytecode, control-flow, and execution traces, and that is trained with supervision that enforces semantic and symbolic consistency.

We begin by introducing formal notation. A smart contract is a program $\mathcal{C}$ whose compiled form is bytecode B. The global blockchain state at a discrete time t is σ_t, which is a mapping from addresses to account states; each account state a consists of balance, code B_a, and storage map S_a. A transaction τ is an ordered tuple $(from, to, v, d, g)$ where v is value, d is call data, and g is gas limit. The state transition function of the EVM is δ: $(\sigma, \tau) \mapsto \sigma'$, and an execution trace is the sequence of internal states $T(\sigma, \tau) = (\sigma_0, \sigma_1, \ldots, \sigma_n)$ generated by successive EVM opcodes. Vulnerabilities are predicates over execution traces; formally, contract $\mathcal{C}$ is vulnerable to class V iff $\exists \tau$ such that $\phi_V(T(\sigma_0, \tau)) = \texttt{true}$, where ϕ_V is a boolean predicate encoding the violation (e.g., unauthorized state change, invariant breach).

EVM-FM learns embeddings $\mathbf{z}_B$ for bytecode B, embeddings $\mathbf{z}_{CFG}$ for control-flow graphs, and embeddings $\mathbf{z}_T$ for execution traces. The training objective uses multi-task losses incorporating detection, contrastive differentiation between vulnerable/patched variants, symbolic alignment with abstract interpreters, and generative objectives that allow the model to produce exploit traces. Formally, for a dataset $\mathcal{D} = \{(B^{(i)}, T^{(i)}, y^{(i)})\}$ where y indicates vulnerability labels, the loss is

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda_{\text{ctr}}\mathcal{L}_{\text{contrast}} + \lambda_{\text{sym}}\mathcal{L}_{\text{sym}} + \lambda_{\text{gen}}\mathcal{L}_{\text{gen}}.$$

Here $\mathcal{L}_{\text{cls}}$ is cross-entropy for vulnerability classification, $\mathcal{L}_{\text{contrast}}$ is a contrastive loss that pulls embeddings of patched and original contracts apart in representation space when they differ semantically, $\mathcal{L}_{\text{sym}}$ penalizes discrepancies between predicted state transitions and symbolic/SMT-derived transitions, and $\mathcal{L}_{\text{gen}}$ is a sequence-generative loss that trains the model to produce attack traces $\hat{T}$ with likelihood maximization. Symmetry constraints are enforced by aligning the model's predicted post-state $\hat{\sigma}'$ with symbolic transition $\delta_{\text{sym}}(\sigma, \tau)$ via a distance $d(\hat{\sigma}', \delta_{\text{sym}}(\sigma, \tau))$.

Architecturally, EVM-FM is multi-modal: a transformer handles tokenized bytecode and AST-like sequences, a graph neural network encodes the CFG and storage access graphs, and a sequence model (autoregressive transformer or RNN) models traces. Inter-modal attention layers enable trace-to-bytecode attention, enabling the model to attribute trace events to code locations. We adopt message-passing GNN updates for the storage-access graph $G = (V, E)$:

$$\mathbf{h}_v^{(k+1)} = \sigma\Big(W_1\mathbf{h}_v^{(k)} + \sum_{u \in \mathcal{N}(v)} W_2\mathbf{h}_u^{(k)} + b\Big),$$

and use readout functions to obtain $\mathbf{z}_{CFG}$.

We evaluate EVM-FM on three axes: (1) detection accuracy versus state-of-the-art static and ML analyzers, (2) semantic consistency measured by agreement with symbolic execution for synthesized test cases, and (3) exploit generation quality measured by whether produced traces successfully violate the target predicate in an instrumented testnet. We also quantify robustness under adversarially-minimal code transformations (semantic-preserving obfuscation). The dataset is curated from verified contracts, known exploit traces, and synthetic mutated examples; we annotate with symbolic counterexamples where possible to support $\mathcal{L}_{\text{sym}}$.

This project produces open-source datasets and model checkpoints, and yields new insights into the tradeoffs of data scale, symbolic supervision, and model capacity. We invite collaborators who bring expertise in formal methods (SMT/abstract interpretation), large-scale pretraining, and deployment partners to test low-latency variants on validators and relayers. Funded doctoral projects and postdocs are welcome to join; industry partnerships for trace collection and compute credits accelerate progress.

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] Y. Yuan et al. Large Language Model based Smart Contract Auditing with LLMBugScanner. arXiv:2512.02069, 2025. https://arxiv.org/abs/2512.02069.

[9] Z. Wei et al. LLM-SmartAudit: Advanced Smart Contract Vulnerability Detection. arXiv:2410.09381, 2024. https://arxiv.org/abs/2410.09381.

[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.