Smart Contract Invariant Synthesis and Mining

Mojtaba Eshghie[1], Gabriele Morello[1], Gustav Andersson Kasche[1], Martin Monperrus[1], and Gerardo Schneider[1]

KTH Royal Institute of Technology, Stockholm, Sweden
`eshghie@kth.se, morello@kth.se, gustavak@kth.se, monperrus@kth.se`

1 Introduction

Theoretical work establishes that monitoring execution steps of systems against logical predicates or invariants is highly effective for most security properties [8]. Recent work utilizes formal methods and invariants for smart contract security [5, 11, 12]. Formal methods heavily rely on specifications and invariants which define the correct behavior of programs. Therefore, high-quality invariants are crucial for defining smart contract's proper behavior. Defining invariants manually can be costly and error-prone. Prior work in the field seeks to automate the process by creating tools for invariant generation. Given the transparent nature of dApps execution environment, recent work mines invariants using historic executions [5]. Recent advances in Artificial Intelligence (AI) and large language models (LLM) have also been applied to the field of smart contracts invariants and specifications to synthesize invariants [4, 12]. Prior studies have assessed the effectiveness of other methods for smart contract security, such as fuzzing, static analysis, and symbolic execution. These assessments focused on their capacity to mitigate real-world smart contract attacks. These methods, however, have demonstrated limited efficacy in addressing real-world attacks [3, 13]. The capability of current state-of-the-art tools for invariant mining in preventing these attacks has not been thoroughly evaluated.

2 Invariant Mining

Invariant mining aims to extract system properties that hold universally over the history of transactions of the deployed smart contracts.

Tools such as InvCon+ combine dynamic and static analysis methods to infer and verify invariant properties [5]. The process of mining invariants typically involves dynamic inference where Likely invariants are generated based on observed transaction history. After generating likely invariants, these are then verified statically against the contract code to confirm their correctness. This involves a Houdini-like algorithm which iteratively refines and verifies invariants [1].

In this work we use InvCon+ specifically on smart contracts that are already successfully exploited in recent years to determine if the tool can mine any invariants that are useful in preventing the mentioned exploits [13].

3 Invariant Synthesis

To enhance the security of smart contracts on the Ethereum main network, our research proposes a novel approach to invariant synthesis leveraging large language models, specifically fine-tuning CodeLlama model. We target *require* statements in smart contracts to learn invariant patterns that protect contract's invariants against malicious and unwanted behavior.

As a first step of this work, we collected and published the largest dataset of the source code of 3.2 million verified smart contracts on Ethereum [6].

Inspired by the enhancements seen in works such as the analysis conducted using GPTScan [10] and RepairLlama [9], we adapt the CodeLlama model to learn from the mentioned collected dataset [7]. The model fine-tuning focuses on understanding and generalizing the logical conditions the require statements enforce, thus capturing invariants that are implicitly defined by developers in the source code. The goal is to build a specialized model for smart contracts written in Solidity as the most used programming language for contract development [2]. This model is useful in many downstream tasks such as automatic program repair and code synthesis.

References

1. Induction duality: Primal-dual search for invariants | Proceedings of the ACM on Programming Languages
2. Solidity documentation (Aug 2023), https://docs.soliditylang.org/en/latest/, accessed: 2023-08-29
3. Chaliasos, S., Charalambous, M.A., Zhou, L., Galanopoulou, R., Gervais, A., Mitropoulos, D., Livshits, B.: Smart Contract and DeFi Security Tools: Do They Meet the Needs of Practitioners? In: Proceedings of the 46th IEEE/ACM International Conference on Software Engineering. pp. 1–13 (Feb 2024). https://doi.org/10.1145/3597503.3623302, http://arxiv.org/abs/2304.02981, arXiv:2304.02981 [cs]
4. Liu, J., Chen, Y., Tan, B., Dillig, I., Feng, Y.: Learning Contract Invariants Using Reinforcement Learning. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. pp. 1–11. ASE '22, Association for Computing Machinery, New York, NY, USA (Jan 2023). https://doi.org/10.1145/3551349.3556962, https://dl.acm.org/doi/10.1145/3551349.3556962
5. Liu, Y., Zhang, C., Li., Y.: Automated Invariant Generation for Solidity Smart Contracts (Dec 2023). https://doi.org/10.48550/arXiv.2401.00650, http://arxiv.org/abs/2401.00650, arXiv:2401.00650 [cs]
6. Morello, G., Eshghie, M., Bobadilla, S., Monperrus, M.: DISL: Fueling Research with A Large Dataset of Solidity Smart Contracts (Mar 2024). https://doi.org/10.48550/arXiv.2403.16861
7. Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X.E., Adi, Y., Liu, J., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C.C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., Synnaeve, G.: Code Llama: Open Foundation Models for Code (Aug 2023). https://doi.org/10.48550/arXiv.2308.12950

8. Schneider, F.B.: Enforceable security policies | ACM Transactions on Information and System Security, https://dl.acm.org/doi/10.1145/353323.353382
9. Silva, A., Fang, S., Monperrus, M.: RepairLLaMA: Efficient Representations and Fine-Tuned Adapters for Program Repair (Dec 2023). https://doi.org/10.48550/arXiv.2312.15698
10. Sun, Y., Wu, D., Xue, Y., Liu, H., Wang, H., Xu, Z., Xie, X., Liu, Y.: GPTScan: Detecting Logic Vulnerabilities in Smart Contracts by Combining GPT with Program Analysis (Dec 2023). https://doi.org/10.48550/arXiv.2308.03314
11. Tolmach, P., Li, Y., Lin, S.W., Liu, Y., Li, Z.: A survey of smart contract formal specification and verification. ACM Computing Surveys (CSUR) **54**(7), 1–38 (2021)
12. Wang, S.J., Pei, K., Yang, J.: SMARTINV: Multimodal Learning for Smart Contract Invariant Inference. pp. 125–125. IEEE Computer Society (Feb 2024). https://doi.org/10.1109/SP54263.2024.00126, https://www.computer.org/csdl/proceedings-article/sp/2024/313000a126/1Ub23GNTAeQ, iSSN: 2375-1207
13. Zhou, L., Xiong, X., Ernstberger, J., Chaliasos, S., Wang, Z., Wang, Y., Qin, K., Wattenhofer, R., Song, D., Gervais, A.: SoK: Decentralized Finance (DeFi) Attacks (Apr 2023). https://doi.org/10.48550/arXiv.2208.13035, http://arxiv.org/abs/2208.13035, arXiv:2208.13035 [cs]