Solvent: liveness verification of smart contracts

Massimo Bartoletti[1], Angelo Ferrando[2], Enrico Lipparini[3], Vadim Malvone[4]

[1] Università degli Studi di Cagliari, Cagliari, Italy
[2] Università degli Studi di Modena e Reggio Emilia, Modena, Italy
[3] Università degli Studi di Genova, Genova, Italy
[4] Télécom Paris, Palaiseau, France

1 Introduction

In recent years we have seen an explosive rise of decentralized applications that implement disintermediated financial ecosystems on top of public permissionless blockchains. These applications are based on so-called smart contracts, i.e. programs which are run by blockchain networks and automatize the exchange of cryptoassets between users, without relying on trusted authorities. It is estimated that more than 100 billions of dollars worth of cryptoassets are today controlled by smart contracts [7]. The peculiarities of the setting (i.e., the absence of intermediaries, the immutability of code after deployment, the quirks in smart contract languages) make smart contracts an appealing target for attackers, as bugs in the code of a smart contract might be exploited to steal cryptoassets or just cause disruption. This is witnessed by a long history of attacks, which overall caused losses exceeding 6 billions of dollars [5].

Formal methods provide an ideal defense against these attacks, since they enable the creation of tools to detect bugs in smart contracts before they are deployed. Indeed, smart contracts verification tools have been mushrooming in the last few years, many of which are based on formal methods: for Ethereum alone — largely the main smart contract platform — dozens of tools exist today [8]. Still, the actual effectiveness of these tools in countering real-world attacks is debatable: indeed, attacks to smart contracts have continued to proliferate, refining their strategies from exploits of known vulnerability patterns to sophisticated attacks to the contract's business logics. Indeed, the vast majority of the losses due to real-world attacks are caused by logic errors in the contract code [5], which are outside the scope of most vulnerability detection tools.

The few tools that support the verification of the contracts' business logic have serious limitations concerning their soundness, completeness and expressivity, making them largely ineffective even for simple cases [4]. The reasons of these limitations are more inherent to the target languages of the tools (Solidity, the main smart contract language for Ethereum, and EVM bytecode, its compilation target) than to the tools themselves. First, Solidity/EVM have some problematic design choices that make reasoning about the contract properties quite hard. The well-known reentrancy issue, which is an ongoing cause of problems since the infamous DAO attack, is only one of the causes: in general, the glitches of high-level abstractions over the low-level target (e.g., gas costs, vulnerable tokens, *etc.*) are a common cause of bugs. Second, liveness properties of smart contract, e.g. requiring that a given user will receive some crypto-assets from a smart contract upon performing some action, are not expressible in general [4]. This is a serious limitation, since the business logic of smart contracts mostly involves exchanges of crypto-assets. Indeed, several real-world attacks to smart contracts exploited liveness weaknesses, allowing attackers to steal or freeze large amounts of crypto-assets [1].

To address these issues we propose Solvent[5], a tool that verifies liveness properties of smart contracts: in particular, Solvent can practically express and verify properties of the exchanges of assets between users. To overcome the above-mentioned limitations due to Solidity/EVM, Solvent operates on a version of Solidity purified from the main semantical quirks of the real language. Solvent takes as input a contract and a set of user-defined liveness properties, and translates them into constraints in the SMT-LIB standard [3]. This translation reduces the problem of verifying a liveness property of a smart contract to a bounded model checking problem over the space of symbolic executions of the contract. The obtained constraints can be solved by SMT solvers like Z3 [6] or cvc5 [2]. Experiments on a benchmark of real-world smart contracts show that Solvent can efficiently verify relevant liveness properties of their behaviour, which are out of the scope of mainstream verification tools operating on the real Solidity, like e.g. SolCMC and Certora. Although Solvent does not target the real Solidity, it provides developers with useful feedback about their smart contracts, since it can detect logical errors that would otherwise remain unnoticed. In particular, when Solvent detects that a property is violated, it can produce a witness of the violation in terms of a concrete execution trace leading the real Solidity contract to a state from which the required transfers of crypto-assets are not realisable.

In case this oral communication is accepted, our talk will be focussed on the main design choices of our tool, and on the challenges related to verifying liveness properties in Solidity, both in its purified and concrete version.

References

1. Alois, J.: Ethereum Parity hack may impact ETH 500,000 or $146 million. https://www.crowdfundinsider.com/2017/11/124200-ethereum-parity-hack-may-impact-eth-500000-146-million/ (2017), accessed on April 9, 2024
2. Barbosa, H., Barrett, C.W., Brain, M., Kremer, G., Lachnitt, H., Mann, M., Mohamed, A., Mohamed, M., Niemetz, A., Nötzli, A., Ozdemir, A., Preiner, M., Reynolds, A., Sheng, Y., Tinelli, C., Zohar, Y.: cvc5: A versatile and industrial-strength SMT solver. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS, vol. 13243, pp. 415–442. Springer (2022). https://doi.org/10.1007/978-3-030-99524-9_24
3. Barrett, C., Fontaine, P., Tinelli, C.: The SMT-LIB Standard: Version 2.6. Tech. rep., Department of Computer Science, The University of Iowa (2017), available at https://smtlib.cs.uiowa.edu/language.shtml

[5] https://github.com/AngeloFerrando/Solvent

4. Bartoletti, M., Fioravanti, F., Matricardi, G., Pettinau, R., Sainas, F.: Towards benchmarking of Solidity verification tools. In: Formal Methods in Blockchain (FMBC) (2024), to appear
5. Chaliasos, S., Charalambous, M.A., Zhou, L., Galanopoulou, R., Gervais, A., Mitropoulos, D., Livshits, B.: Smart contract and DeFi security: Insights from tool evaluations and practitioner surveys. In: IEEE/ACM International Conference on Software Engineering (ICSE). pp. 60:1–60:13. ACM (2024). https://doi.org/10.1145/3597503.3623302, https://doi.org/10.1145/3597503.3623302
6. De Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS, vol. 4963, pp. 337–340. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_24
7. Defillama. https://defillama.com/, accessed on April 9, 2024
8. Kushwaha, S.S., Joshi, S., Singh, D., Kaur, M., Lee, H.N.: Ethereum smart contract analysis tools: A systematic review. IEEE Access **10**, 57037–57062 (2022). https://doi.org/10.1109/ACCESS.2022.3169902