Static Analysis of Ethereum Smart Contracts by Abstract Interpretation

Master 2 research internship proposal, 2018–2019

Supervisor:	Antoine Miné (antoine.mine@lip6.fr)
Internship location:	APR team, LIP6 Sorbonne Université Jussieu Campus, Paris, France
Duration:	4.5 to 6 months
Related project:	Mopsa project
Relevant courses:	– MPRI 2.6: Abstract interpretation: application to verification and static analysis – Master STL: Typage et analyse statique

Other internships are possible on the topic of static analysis and abstract interpretation. Contact the internship supervisor for more information.

Motivation

The goal of the internship is to develop, prove correct, and implement novel static analyses by Abstract Interpretation to verify the correctness of smart contracts.

The internship focuses on Ethereum [1], a cryptocurrency framework that emploies blockchain techniques and supports the execution of smart contracts. Smart contracts can be written in a variety of languages, which are compiled into bytecode that is run on the Ethereum Virtual Machine (EVM). EVM provides a Turing-complete, stack-based machine with a byte-array based memory and an associative storage (with 256-bit keys and values). The EVM is completely specified formally [1], giving it an unambiguous semantics. Hence, the intership will focus on analyzing EVM bytecode.

Application of formal methods to smart contracts is a recent area of research, with few results yet. Examples include preliminary analyses using type and effect systems with F* [2], and using model-checking with Spin [4]. We are not aware of works based on Abstract Interpretation.

Verifying Smart Contracts

Several causes of vulnerabilities in Ethereum smart contracts have been uncovered [3]. Given the novelty of the subject, a preliminary study must be performed to determine precisely which vulnerabilities can be efficiently detected by static analysis, and which properties must be inferred to detect them. Nevertheless, one promising direction is to focus on vulnerabilities related to exceptional behaviors (i.e., run-time errors). The intern may focus primarily on the following analyses:

1. **Checking exceptions.**
   Erroneous operations in the execution of a contract cause exceptions to be raised and the contract to be terminated. Some exceptions, though, are propagated back to the caller contract, while others result in return codes, which may not be checked appropriately, causing vulnerabilities [3]. A static analysis could detect that possible exceptions are correctly handled, even through calls to contracts.

2. **Call stack depth.**
   The EVM stack depth is limited to 1024, and exceeding this limitation causes an exception that may not be properly handled by a contract and be the base of attacks [3]. A static stack depth analysis can alleviate this kind of errors.

3. **Atomicity vulnerabilities.**
   Contracts do not operate atomically: a contract can call another contract, that can access and exploit the intermediate state the calling contract is in for ill effects. For instance, a reentrancy bug, where a contract is called again from within its own execution, is the basis of the The DAO attack, which caused a 60 million US dollar loss in June 2016 [3].

4. **Transaction fees.**
   Every operation in the EVM comes with a cost, measured in an abstract unit called gas. The cost of a contract is specified up-front, and the execution stops (with an exception) whenever this cost is exceeded. A cost analysis can check statically whether a contract will meet its cost target. Such an analysis can leverage the large body of literature on worst case execution time of binary programs.

The intern will leverage existing abstractions, such as numeric abstractions, and develop new abstractions if needed. Most static analyses target source-level programs. Analyzing bytecode-level programs poses additional challenges due to the very low-level of instructions, and the tendency of fine-grained abstract transfer functions to accumulate imprecision. Methods including developing specific relational domains or high-level expression reconstruction may be required to reach the intended level of precision.

Extension to checking other classes of vulnerabilities will be considered if time permits. One possible direction is the verification of temporal properties, as proposed in [4], but employing a direct abstract interpretation of the EVM bytecode instead of model-checking hand-crafted models with Spin.

Expected Work

The intended work will include a theoretical side: developing abstract semantics and proving formally their soundness. It will also include a practical side: implementing the semantics in a static analyzer and validating their benefit experimentally on sample smart contracts.

The host team is developing a static analysis platform, Mopsa, that includes several ready-to-use abstractions, and a framework that allows easily extending the analyses both to new abstractions, and to new languages. For experimental purposes, the intern will thus write an EVM bytecode frontend for Mopsa, reuse existing Mopsa abstractions, and add the necessary abstractions to carry the static analysis.

Requested Skills

The internship requires a strong knowledge of static analysis by abstract interpretation. The intern should have followed one of the following Master 2 courses: "Abstract interpretation: application to verification and static analysis" from MPRI, "Typage et analyse statique" from the STL Master at Sorbonne Université, or an equivalent course. Basic knowledge of blockchain principles is a plus, but not a requirement.
Knowledge of the OCaml language is also required, for the implementation effort within the Mopsa platform.

Context of the Internship

The internship will take place in the APR team, in the LIP6 laboratory, Jussieu Campus, Sorbonne Université, Paris. It is proposed in the scope of the Mopsa ERC research project. If the internship is successful, the project may provide opportunities for a funded PhD on a follow-up subject.

References

[1] Gavin Wood, Nick Savers et al. Ethereum Yellow Paper. `https://github.com/ethereum/yellowpaper`

[2] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Anitha Gollamudi, Georges Gonthier, Nadim Kobeissi, Aseem Rastogi, Thomas Sibut-Pinote, Nikhil Swamy, and Santiago Zanella-Béguelin. Formal Verification of Smart Contracts: Short Paper. In Proc. PLAS'16, 91–96, Vienna, Austria, Oct. 2016. ACM Press.

[3] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making Smart Contracts Smarter. In Proc. CCS'16, 254–269, Vienna, Austria, Pct. 2016. ACM Press.

[4] Xiaomin Bai, Zijing Cheng, Zhangbo Duan, and Kai Hu. Formal Modeling and Verification of Smart Contracts. In Proc. ICSCA 2018, 322–326, Kuantan, Malaysia, Feb. 2018. ACM Press.