Static Profiling and Optimization of Ethereum Smart Contracts using Resource Analysis

- IEEE Access
  `https://ieeeaccess.ieee.org/`
- Artículo:
- Justificación Información Artículo:
  - Copia de la primera y última página del artículo
  - Página web con la aceptación del artículo
  - Página web DOI: `https://doi.org/10.1109/ACCESS.2021.3057565`
- Justificación Índice Impacto:
  - Copia de la información Índice JCR 2019 sobre IEEE Access

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 12, 2020, accepted February 2, 2021, date of publication February 5, 2021, date of current version February 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3057565

Static Profiling and Optimization of Ethereum Smart Contracts Using Resource Analysis

JESÚS CORREAS[1], PABLO GORDILLO[1], AND GUILLERMO ROMÁN-DÍEZ[2]
[1]Departamento de Sistemas Informáticos y Computación, Facultad de Informática, Complutense University of Madrid, 28040 Madrid, Spain
[2]Lenguajes, Sistemas Informáticos e Ingeniería de Software, E.T.S. de Ingenieros Informáticos, Universidad Politécnica de Madrid, 28660 Madrid, Spain

Corresponding author: Pablo Gordillo (pabgordi@ucm.es)

This work was supported in part by the Spanish Ministerio de Ciencia, Innovación y Universidades (MCIU) through the Agencia Estatal de Investigación (AEI) and Fondo Europeo de Desarrollo Regional (FEDER) (EU) Project under Grant RTI2018-094403-B-C31, in part by the Comunidad de Madrid (CM) Projects co-funded by the Fondos Estructurales y de Inversión Europeos (EIE Funds) of the European Union under Grant S2018/TCS-4314 and Grant S2018/TCS-4339, and in part by the Universidad Complutense de Madrid (UCM) under Grant CT27/16-CT28/16.

ABSTRACT Profiling tools have been widely used for studying the behavior of the programs with the objective of reducing the amount of resources consumed by them. Most profilers collect the information with dynamic techniques, i.e., execute an instrumented version of the program with some specific input arguments to profile the measures of interest. This article presents a novel static profiling technique for Ethereum smart contracts that, using static resource analysis, is able to generate upper-bound expressions that can be used to produce profiling information about the measure of interest. Unlike traditional profiling tools, we get upper-bounds on the measures of interest expressed in terms of the input arguments or the state variables of the smart contracts. The information that can be obtained by the upper-bounds allows us to detect gas-expensive fragments of a Solidity program or to spot resource-related vulnerabilities at specific program points of the program. Moreover, in this article we propose an automatic optimization of Solidity programs which reduces their *gas* consumption replacing the accesses to state variables by gas-efficient accesses to local variables. We have experimentally evaluated our technique and we have detected that 6.81% of the public functions analyzed can be optimized and 1.43% are vulnerable to execute arbitrary code.

INDEX TERMS Blockchain, Ethereum, resource analysis, smart contracts, static analysis.

I. INTRODUCTION

Ethereum [37] is an open-source platform for decentralized applications and nowadays has become the world's leading programmable blockchain. One of the reasons of this success is that Ethereum smart contracts can be programmed using a Turing complete language and it includes a powerful set of tools for its development. An immutable version of the compiled smart contract can be deployed in the Ethereum platform and will be executed using the Ethereum Virtual Machine (EVM). As other blockchains, Ethereum has its native cryptocurrency named *Ether* and the execution fees for running smart contracts on the Ethereum blockchain are metered in units of *gas*. It is a measure of the amount of computational effort spent on executing each single EVM bytecode operation. The gas consumption of each EVM instruction is detailed in [37]. Miners get paid an amount in *Ether* that results of applying a gas price to the total amount of gas that took them to execute a complete transaction. Using this model, Ethereum prevents the emitters from wasting computational power, discourages the programmers to use gas-expensive operations (e.g. as the cost of replicating data in a decentralized environment is high, storage bytecodes are gas-expensive) and prevents from DoS attacks and non-terminating executions.

Solidity [15] is a programming language to write smart contracts and its compiler produces EVM code to be deployed in the Ethereum platform. The Solidity compiler includes several static analyses that produce useful information during the contract development phase. Among this information it can be found the amount of gas that a function will consume for its execution. The Solidity compiler is able to produce precise *constant* gas bounds, however, when the cost expression depends on input parameters (or information stored in the contract state), the compiler simply returns ∞ as gas bound, and we found it occurs in almost one in every ten public functions [3]. Furthermore, minimal modifications on the code make the compiler unable to detect unbounded loops and it does not warn the programmer about this potential risk.

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Hammad Memon.

to storage. It proposed a sound transformation that replaces the accesses to storage by accesses to memory that consume less gas. We have applied them to analyze more than 40,000 real public function of smart contracts getting that a 9.02% are parametric, a 6.81% of them can be optimized, and 4.19% may be potentially vulnerable.

An interesting direction for future work is to improve the precision of our tool optimizing, not only basic type variables, but also complex type variables such as arrays, structs or maps. In addition, we plan to relax the conditions defined to optimize storage accesses in Section V and generalize the optimization to functions which access to the state variables in other functions of the contract different to the one under analysis. Additionally, we would study the applicability of the optimization at EVM level, which is a more complex case as we would modify the structure of the EVM bytecode and it may affect to the size and the addresses of the original bytecode.

REFERENCES

[1] E. Albert, P. Arenas, S. Genaim, and G. Puebla, "Closed-form upper bounds in static cost analysis," *J. Automated Reasoning*, vol. 46, no. 2, pp. 161–203, Feb. 2011.

[2] E. Albert, J. Correas, P. Gordillo, G. Román-Díez, and A. Rubio, "GASOL: Gas analysis and optimization for Ethereum smart contracts," in *Proc. 26th Int. Conf. Tools Algorithms Construct. Anal. Syst.*, in Lecture Notes in Computer Science, vol. 12079. Dublin, Ireland: Springer, 2020, pp. 118–125.

[3] E. Albert, P. Gordillo, A. Rubio, and I. Sergey, "Running on fumes: Preventing out-of-gas vulnerabilities in Ethereum smart contracts using static resource analysis," in *Proc. 13th Int. Conf. Verification Eval. Comput. Commun. Syst. (VECoS)*, in Lecture Notes in Computer Science, vol. 11847. Porto, Portugal: Springer, 2019, pp. 63–78.

[4] D. E. Alonso-Blas and S. Genaim, "On the limits of the classical approach to cost analysis," in *Static Analysis* (Lecture Notes in Computer Science), vol. 7460, A. Miné and D. Schmidt Eds. Deauville, France: Springer, 2012, pp. 405–421.

[5] N. Ambroladze, "Fast and scalable analysis of smart contracts," M.S. thesis, Swiss Federal Inst. Technol., Zürich, Switzerland, 2018.

[6] M. M. A. Aldweesh, M. Alharby, and A. V. Moorsel, "OpBench: A CPU performance benchmark for Ethereum smart contract operation code," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 274–281.

[7] R. Bagnara, M. P. Hill, and E. Zaffanella, "The parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems," *Sci. Comput. Program.*, vol. 72, nos. 1–2, pp. 3–21, 2008.

[8] C. Boogerd and L. Moonen, "Prioritizing software inspection results using static profiling," in *Proc. 6th IEEE Int. Workshop Source Code Anal. Manipulation*. Washington, DC, USA: IEEE Computer Society, Sep. 2006, pp. 149–160.

[9] C. Boogerd and L. Moonen, "On the use of data flow analysis in static profiling," in *Proc. 8th IEEE Int. Work. Conf. Source Code Anal. Manipulation*, Sep. 2008, pp. 79–88.

[10] G. Canfora, A. D. Sorbo, S. Laudanna, A. Vacca, and C. AaronVisaggio, "Gasmet: Profiling gas leaks in the deployment of solidity smart contracts," *CoRR*, vol. abs/2008.05449, pp. 1–13, Dec. 2020.

[11] J. Charlier, S. Lagraa, R. State, and J. François, "Profiling smart contracts interactions tensor decomposition and graph mining," in *Proc. 2nd Workshop Mining Data Financial Appl. Eur. Conf. Mach. Learn. Princ. Pract. Knowl. Discovery Databases*, Skopje, Macedonia, vol. 1941, Sep. 2017, pp. 31–42.

[12] T. Chen, Y. Feng, Z. Li, H. Zhou, X. Luo, X. Li, X. Xiao, J. Chen, and X. Zhang, "GasChecker: Scalable analysis for discovering gas-inefficient smart contracts," *IEEE Trans. Emerg. Topics Comput.*, early access, Mar. 6, 2020, doi: 10.1109/TETC.2020.2979019.

[13] T. Chen, X. Li, X. Luo, and X. Zhang, "Under-optimized smart contracts devour your money," in *Proc. IEEE 24th Int. Conf. Softw. Anal., Evol. Reeng. (SANER)*. Washington, DC, USA: IEEE Computer Society, Feb. 2017, pp. 442–446.

[14] T. Chen, Z. Li, H. Zhou, J. Chen, X. Luo, X. Li, and X. Zhang, "Towards saving money in using smart contracts," in *Proc. 40th Int. Conf. Softw. Eng., New Ideas Emerg. Results*, Gothenburg, Sweden, May 2018, pp. 81–84.

[15] Ethereum. (2018). *Solidity*. [Online]. Available: https://solidity.readthedocs.io

[16] A. Flores-Montoya and R. Hähnle, "Resource analysis of complex programs with cost equations," in *Programming Languages and Systems* (Lecture Notes in Computer Science), vol. 8858. Singapore: Springer, 2014, pp. 275–295.

[17] R. W. Floyd, "Assigning meanings to programs," in *Program Verification*. Dordrecht, The Netherlands: Springer, 1993, pp. 65–81.

[18] A. Garcia, C. Laneve, and M. Lienhardt, "Static analysis of cloud elasticity," in *Proc. 17th Int. Symp. Princ. Pract. Declarative Program.*, Siena, Italy, Jul. 2015, pp. 125–136.

[19] N. Grech, M. Kong, A. Jurisevic, L. Brent, B. Scholz, and A. Smaragdakis, "Madmax: Surviving out-of-gas conditions in Ethereum smart contracts," in *Proc. PACMPL*, 2018, pp. 116:1–116:27.

[20] I. Grishchenko, M. Maffei, and C. Schneidewind, "A semantic framework for the security analysis of Ethereum smart contracts," in *Principles of Security and Trust* (Lecture Notes in Computer Science), vol. 10804. Thessaloniki, Greece: Springer, 2018, pp. 243–269.

[21] S. Grossman, I. Abraham, G. Golan-Gueta, Y. Michalevsky, N. Rinetzky, M. Sagiv, and Y. Zohar, "Online detection of effectively callback free objects with applications to smart contracts," in *Proc. ACM Program. Lang.*, vol. 2, Jan. 2018, pp. 1–28.

[22] R. Haemmerlé, P. López-García, U. Liqat, M. Klemen, J. P. Gallagher, and M. V. Hermenegildo, "A transformational approach to parametric accumulated-cost static profiling," in *Functional and Logic Programming* (Lecture Notes in Computer Science), vol. 9613. Kochi, Japan: Springer, 2016, pp. 163–180.

[23] J. He, M. Balunović, N. Ambroladze, P. Tsankov, and M. Vechev, "Learning to fuzz from symbolic execution with application to smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, London, U.K., Nov. 2019, pp. 531–548.

[24] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "ZEUS: Analyzing safety of smart contracts," in *Proc. Netw. Distrib. Syst. Secur. Symp.* Reston, VA, USA: Internet Society, 2018, pp. 1–12.

[25] J. Krupp and C. Rossow, "Teether: Gnawing at Ethereum to automatically exploit smart contracts," in *Proc. USENIX Secur. Symp.* Berkeley, CA, USA: USENIX Association, 2018, pp. 1317–1333.

[26] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 254–269.

[27] M. Marescotti, M. Blicha, A. E. J. Hyvärinen, S. Asadi, and A. N. Sharygina, "Computing exact worst-case gas consumption for smart contracts," in *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice* (Lecture Notes in Computer Science), vol. 11247. Limassol, Cyprus: Springer, 2018, pp. 450–465.

[28] R. G. Morgan and S. A. Jarvis, "Profiling large-scale lazy functional programs," *J. Funct. Program.*, vol. 8, no. 3, pp. 201–237, May 1998.

[29] J. Nagele and M. A. Schett, "Blockchain superoptimizer," in *Proc. 29th Int. Symp. Logic-Based Program Synth. Transformation (LOPSTR)*, 2019, pp. 1–15. [Online]. Available: https://arxiv.org/abs/2005.05912

[30] I. Nikolic, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor, "Finding the greedy, prodigal, and suicidal contracts at scale," in *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, Dec. 2018, pp. 653–663.

[31] A. Podelski and A. Rybalchenko, "A complete method for the synthesis of linear ranking functions," in *Verification, Model Checking, and Abstract Interpretation* (Lecture Notes in Computer Science). Venice, Italy: Springer, 2004, pp. 239–251.

[32] C. Signer, "Gas cost analysis for Ethereum smart contracts," M.S. thesis, Swiss Federal Inst. Technol., Zürich, Switzerland, 2018.

[33] K. Toyoda, K. Machi, Y. Ohtake, and A. N. Zhang, "Function-level bottleneck analysis of private proof-of-authority Ethereum blockchain," *IEEE Access*, vol. 8, pp. 141611–141621, 2020.

[34] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Bünzli, and M. Vechev, "Securify: Practical security analysis of smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 67–82.

Author Dashboard

2 Manuscripts with Decisions

Start New Submission

Legacy Instructions

5 Most Recent E-mails

Manuscripts with Decisions

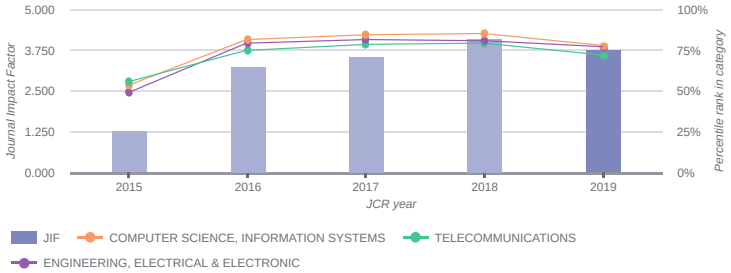
ACTION	STATUS	ID	TITLE	SUBMITTED	DECISIONED
Copyright transferred on 03-Feb-2021	EIC: Abbott, Derek ADM: Yadav, Devendra • Accept (02-Feb-2021) view decision letter	Access-2020-59835	Static Profiling and Optimization of Ethereum Smart Contracts using Resource Analysis View Final Submission	12-Dec-2020	02-Feb-2021
	EIC: Abbott, Derek ADM: Sinha, Namrata • Reject (01-Dec-2020) view decision letter	Access-2020-55087	Static Profiling and Optimization of Ethereum Smart Contracts using Resource Analysis View Submission	12-Nov-2020	01-Dec-2020

InCites Journal Citation Reports

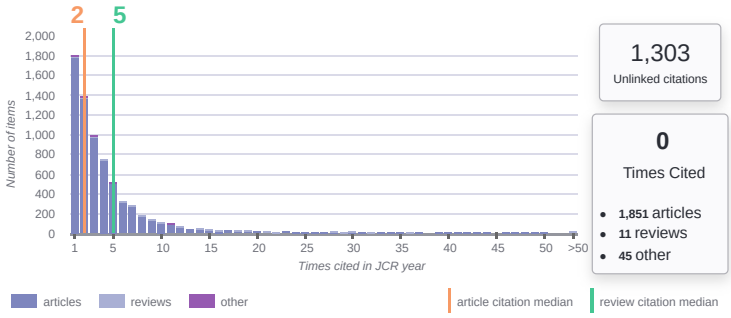
The data in the two graphs below and in the Journal Impact Factor calculation panels represent citation activity in 2019 to items published in the journal in the prior two years. They detail the components of the Journal Impact Factor. Use the "All Years" tab to access key metrics and additional data for the current year and all prior years for this journal.

2019 Journal Impact Factor & percentile rank in category for: IEEE Access

3.745
2019 Journal Impact Factor



2019 JIF Citation Distribution for: IEEE Access



1,303
Unlinked citations

0
Times Cited

- **1,851** articles
- **11** reviews
- **45** other

Rank

Rank

JCR Impact Factor

JCR Year	COMPUTER SCIENCE, INFORMATION SYSTEMS			ENGINEERING, ELECTRICAL & ELECTRONIC			TELE
	Rank	Quartile	JIF Percentile	Rank	Quartile	JIF Percentile	Rank
2019	35/156	Q1	77.885	61/266	Q1	77.256	26/90
2018	23/155	Q1	85.484	52/266	Q1	80.639	19/88
2017	24/148	Q1	84.122	48/260	Q1	81.731	19/87
2016	27/146	Q1	81.849	54/262	Q1	79.580	23/89
2015	68/144	Q2	53.125	131/257	Q3	49.222	37/82