How are Smart Contracts Vulnerabilities Fixed? Bridging the Gap between Theory and Practice

Francesco Salzano[1,*], Lodovica Marchesi[2,*], Remo Pareschi[1], Roberto Tonelli[2], Simone Scalabrino[1] and Rocco Oliveto[1]

[1]STAKE Lab, University of Molise, Pesche, Italy
[2]Dep. of Mathematics and Computer Science, University of Cagliari, Italy

Keywords
Smart Contract engineering, Smart Contract vulnerabilities, Smart Contract vulnerability fixing

Smart contracts are self-executing contracts that use blockchain technology to facilitate, verify, or enforce the negotiation or performance of a contract [1]. The Smart Contracts are programmable and can automate the transfer of digital assets. However, vulnerabilities in Smart Contracts can lead to significant losses, as was the case with the DAO attack, where around $60 million was fraudulently stolen [2]. Therefore, it is crucial to prioritize Smart Contract security to minimize such risks. Various fields have explored Smart Contract security aspects, including research that identifies security defects and suggests potential solutions to fix vulnerabilities [3, 4]. Additionally, past research has provided security vulnerability detection tools based on static analysis and fuzzing, which have been empirically evaluated [5, 6]. However, it is still unclear to what extent developers follow these research outlines and whether there are other viable solutions available.

The purpose of this study is to examine the changes made by Solidity developers when fixing Smart Contracts hosted in GitHub repositories. The main goal is to determine the extent to which developer-made fixes align with the guidelines outlined in the literature and to identify new and practical fixing methods. To begin, we gathered guidelines for fixing vulnerabilities from various sources in the literature. We considered a fixing approach that provided secure code to address a security threat as a guideline. Next, we searched Solidity GitHub repositories for commits that patched vulnerabilities and verified if these fixes were already included in the collected recommendations. As a second step, we identified vulnerability fixes that were not mentioned in the literature. For this analysis, we carefully evaluated each identified solution to determine if it was appropriate.

To steer our research project towards our goals, we have formulated two research questions as follows:

- **RQ1**: To what extent do developers follow the fixing guidelines provided by the literature?
- **RQ2**: Are there valid fixing approaches other than those found in the literature?

To answer our first research question, we gathered academic solutions for addressing vulnerabilities. We then searched for commits that fixed vulnerabilities in Solidity GitHub repositories that had at least ten stars. To exclude irrelevant commits, we used an NLP filter that was tuned to exclude commit messages that were not related to security fixes. During this phase, we considered vulnerabilities included in the DASP TOP 10 taxonomy.

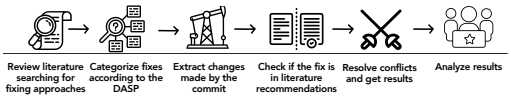

Figure 1: Overall workflow to answer RQ1

After that, we manually evaluated these commits and excluded those that did not report the fixed vulnerability and the patched file. We made this choice to limit the bias introduced by the evaluators and to follow the instructions left by the developers of the commit messages. The remaining commits were analyzed to evaluate the extent to which the obtained fixes complied with the literature fixing guidelines. If we found fixes that did not comply with the guidelines, we conducted experiments to address our second research question. Figure 1 depicts the workflow we designed to respond **RQ1**.

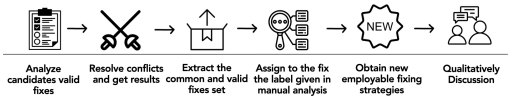

Figure 2: Overall workflow to answer RQ2

6th Distributed Ledger Technology Workshop, May, 14-15 2024 – Turin, Italy
*Corresponding author.
f.salzano1@studenti.unimol.it (F. Salzano); lodovica.marchesi@unica.it (L. Marchesi); remo.pareschi@unimol.it (R. Pareschi); roberto.tonelli@unica.it (R. Tonelli); simone.scalabrino@unimol.it (S. Scalabrino); rocco.oliveto@unimol.it (R. Oliveto)
0000-0002-1029-4861 (F. Salzano); 0000-0002-0627-5043 (L. Marchesi); 0000-0002-4912-582X (R. Pareschi); 0000-0002-9090-7698 (R. Tonelli); 0000-0003-1764-9685 (S. Scalabrino); 0000-0002-7995-8582 (R. Oliveto)

Figure 3: Denial of service patched by a commit.

Regarding **RQ2**, we examined the patches that were applied to address security threats in the set of resulting commits. The authors evaluated whether these solutions are included in academic recommendations. If not, they analyzed them to determine if the patches are valid and should be considered as usable.

In relation to answering both **RQs** regarding whether conflicts among authors arise, the research plan includes a conflict-resolution phase. This phase consists of a critical discussion of the instances in conflict between the analyzers. The flow dedicated to addressing **RQ2** is summarized in Figure 2.

We executed the experimental plan we designed to validate our proposal, obtaining promising results. Figure 3 shows an instance, in which developers used a fix that we do not observe in the reviewed literature. In detail, the function was vulnerable to *denial of service* because the unbounded iteration can consume more gas than the block gas limit, permanently reverting the transaction.

The change made by the commit limits the upper limit, and the gas consumption becomes bounded, resolving the *denial of service*. Thus, we marked this fix as valid.

To sum up, previous research has provided guidelines for fixing vulnerabilities. Our study aims to evaluate whether developers follow these recommendations and to provide new options for fixing vulnerabilities. The datasets and results from our study will be made publicly available to encourage further research, which will help to expand the possibilities of auto-repair tools and deepen our understanding of fixing changes.

Acknowledgments

This project was partially funded through TruMaN (Italian Ministry of University and Research, 2022, PRIN, Project 2022F5CLN2).

References

[1] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, B. Xu, Smart contract development: Challenges and opportunities, IEEE Transactions on Software Engineering 47 (2019) 2084–2106.

[2] M. I. Mehar, C. L. Shier, A. Giambattista, E. Gong, G. Fletcher, R. Sanayhie, H. M. Kim, M. Laskowski, Understanding a revolutionary and flawed grand experiment in blockchain: the dao attack, Journal of Cases on Information Technology (JCIT) 21 (2019) 19–32.

[3] J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo, T. Chen, Defining smart contract defects on ethereum, IEEE Transactions on Software Engineering 48 (2020) 327–345.

[4] X. Zhou, Y. Chen, H. Guo, X. Chen, Y. Huang, Security code recommendations for smart contract, in: 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2023, pp. 190–200.

[5] T. Durieux, J. F. Ferreira, R. Abreu, P. Cruz, Empirical review of automated analysis tools on 47,587 ethereum smart contracts, in: Proceedings of the ACM/IEEE 42nd International conference on software engineering, 2020, pp. 530–541.

[6] J. F. Ferreira, P. Cruz, T. Durieux, R. Abreu, Smartbugs: A framework to analyze solidity smart contracts, in: Proceedings of the 35th IEEE/ACM international conference on automated software engineering, 2020, pp. 1349–1352.