Looking for Lacunae in Bitcoin Core's Fuzzing Efforts

Alex Groce
Northern Arizona University
United States

ABSTRACT

Bitcoin is one of the most prominent distributed software systems in the world, and a key part of a potentially revolutionary new form of financial tool, cryptocurrency. At heart, bitcoin exists as a set of nodes running an implementation of the bitcoin protocol. This paper describes an effort, requested by Chaincode labs, which exists to support and develop bitcoin, to investigate and enhance the effectiveness of the bitcoin core implementation fuzzing effort. The effort initially began as a query about how to escape *saturation* in the fuzzing effort, but developed into a more general exploration once it was determined that saturation was largely illusory, a byproduct of the (then) fuzzing configuration. This paper reports the process and outcomes of the two-week focused effort that emerged from that initial contact between Chaincode labs and academic researchers. While no smoking guns indicating serious issues were found, a large number of additional fuzz corpus entries were added to the bitcoin QA assets, some long-standing problems in OSSFuzz triage were clarified, the set of documented fuzzers was increased, and the first mutation analysis of bitcoin core code revealed opportunities for further improvement. We also contrast the bitcoin core transaction verification testing with similar tests for the most popular Ethereum and dogecoin implementations. This paper provides an overview of the challenges involved in building testing infrastructure, processess, and documentation for a highly visible open source target system, from both the state-of-the-art research perspective and the practical engineering perspective.

CCS CONCEPTS

• **Software and its engineering** → **Dynamic analysis**; **Software testing and debugging**.

KEYWORDS

fuzzing, saturation, test diversity, mutation analysis

ACM Reference Format:
Alex Groce. 2021. Looking for Lacunae in Bitcoin Core's Fuzzing Efforts. In *Proceedings of the 2021 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! '21), October 20–22, 2021, Chicago, IL, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3486607.3486772

Onward! '21, October 20–22, 2021, Chicago, IL, USA

ACM ISBN 978-1-4503-9110-8/21/10…$15.00
https://doi.org/10.1145/3486607.3486772

1 INTRODUCTION

Bitcoin is the most popular cryptocurrency, and one of the most visible "new" systems based on software to rise to prominence in the last decade. As we write, while volatile, bitcoin consistently has a market cap of over half a trillion dollars since January of 2021. Due to its distributed, decentralized nature, bitcoin in some sense is the sum of the operations of the code executed by many independent bitcoin nodes, especially nodes that mine cryptocurrency. Bitcoin core (https://github.com/bitcoin/bitcoin) is by far the most popular implementation, and serves as a reference for all other implementations. To a significant degreee, the code of bitcoin core *is* bitcoin. The main bitcoin core repo on GitHub has over 57,000 stars, and has been forked more than 30,000 times.

Because of its fame and the high monetary value of bitcoins, the bitcoin protocol and its implementations are a high-value target for hackers (or even nation states interested in controlling cryptocurrency developments).

2 INITIAL CONTACT AND THE PROBLEM OF SATURATION

3 ADDING FUZZER DIVERSITY: USING ECLIPSER AND TRYING ENSEMBLE FUZZING

Eclipser [1] is a fuzzer that combines afl-like[1] coverage-driven mutation-based fuzzing with a scalable form of grey-box concolic testing.

4 SIDE ISSUES: TRIAGE, SPACE USAGE, AND FRUSTRATING BUGS

5 TRYING SWARM FUZZING

DeepState [2] provides strong support for swarm fuzzing. However, while there are similarities to the bitcoin core developed infrastucture for fuzz targets and the DeepState API, re-writing the fuzz targets to use DeepState is too large an effort for the likely payoff at this time, even though DeepState would also provide additional fuzzer diversity.

6 MUTATION ANALYSIS

We used the universal mutator (https://github.com/agroce/universalmutator) [3] to mutate the code.

6.1 Comparison with Geth and Dogecoin

7 CONCLUSIONS

REFERENCES

[1] Jaeseung Choi, Joonun Jang, Choongwoo Han, and Sang Kil Cha. 2019. Grey-box Concolic Testing on Binary Code. In *Proceedings of the International Conference on*

[1]In fact, the latest version of Eclipser simply uses afl for non concolic fuzzing.

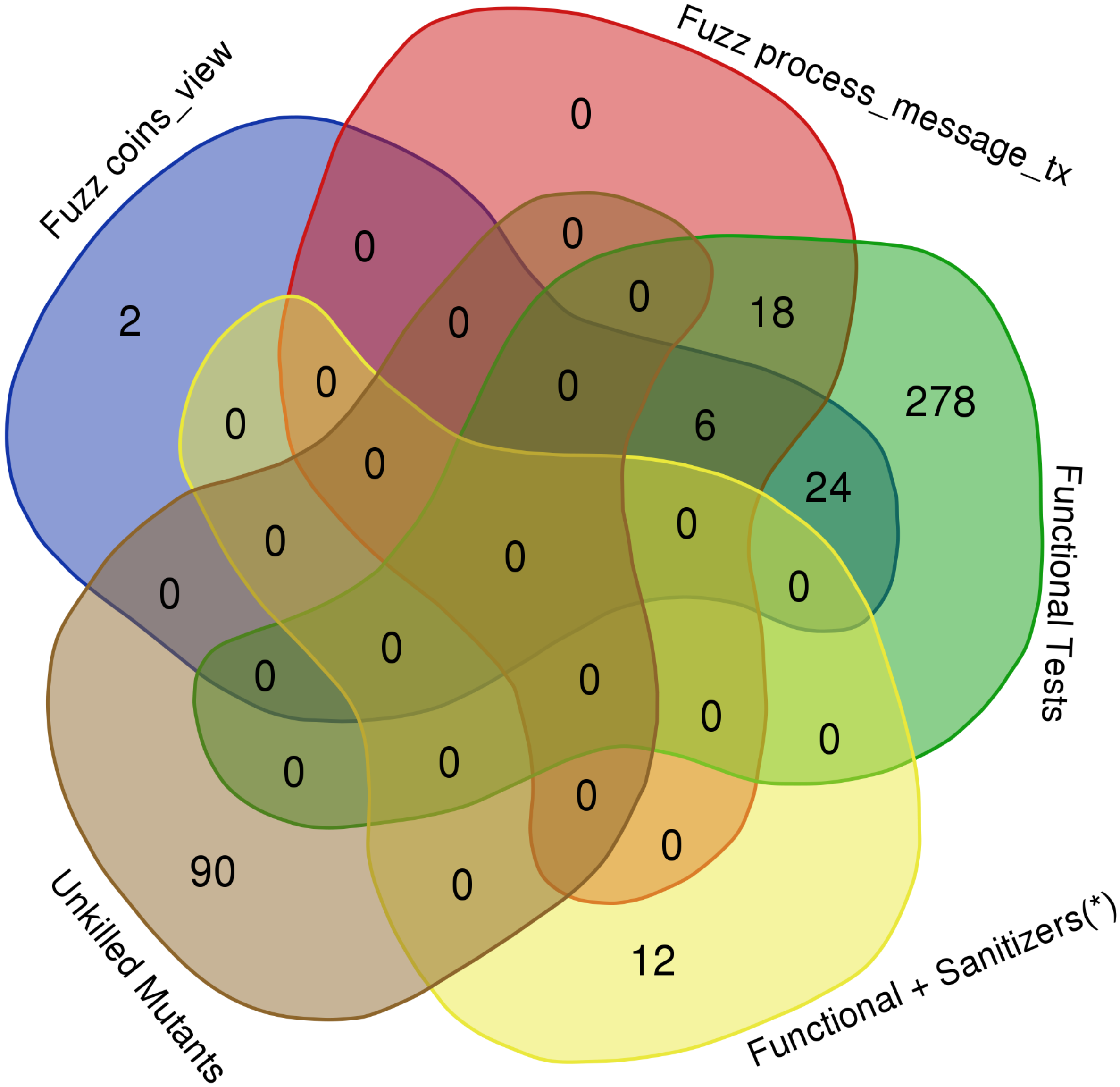

Figure 1: The Gaping Maw of Cthulhu

Software Engineering. 736–747.

[2] Peter Goodman and Alex Groce. 2018. DeepState: Symbolic unit testing for C and C++. In *NDSS Workshop on Binary Analysis Research.*

[3] Alex Groce, Josie Holmes, Darko Marinov, August Shi, and Lingming Zhang. 2018. An Extensible, Regular-expression-based Tool for Multi-language Mutant Generation. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings* (Gothenburg, Sweden) *(ICSE '18)*. ACM, New York, NY, USA, 25–28. https://doi.org/10.1145/3183440.3183485