Why SStuBs are Missed by a Static Analyzer?

Nazmus Sakeef (sakeef@ualberta.ca), Sakib Hasan (sakib2@ualberta.ca)

1 Problem Background

Identifying bugs in a software project is one of the most crucial obstacles prior to release. Continuous development, integration and updates made over time initiates the evolution of different kinds of bugs in a released system. These bugs can be found in single statements, or can be found in a block of code.

Some of these bugs have fixes that are so basic that they are referred to as "Simple Stupid Bugs(*SStuBs*)"[1]. Although some types of SStuBs are identified by static analysers[1], [3], these analyzers also fail to identify many other categories*. Thus, in this study, we will try to investigate which categories of SStuBs are generally missed by a static analyzer called *SonarQube*, and also will address why those go unnoticed.

2 Related Works & Motivation

Karampatsis et. al.[1] investigated whether a static analyzer *(SpotBugs)* could detect SStuBs and discovered only 12% of them were flagged. Additionally, they discovered 200M potential issues, rendering static analyzers useless for detecting SStuBs. However, they did not provide any explanations for this failure. *Mosolygo, B. et al.*[3] tried another static analyzer, *PMD*, to check if it could identify any lines containing SStuBs, but it failed to do so for the studied dataset. The authors concluded that static analyzers had difficulty finding these bugs, although they did not elaborate on why.

Moreover, *Lavazza et. al.*[2] attempted to quantify the amount of "False Positive" issues raised by *SpotBugs* in five open-source projects. *David A. Tomassi*[4], on the other hand, investigated the effectiveness of *ErrorProne* and *SpotBugs* trying to find 320 kinds of Java bugs from *BugSwarm* dataset[5]. However, none of them addressed the reasons behind the failures of the used analyzers. This raises a research direction focusing on the possible reasons behind the poor performance of the analyzers.

3 Research Questions

To the best of our knowledge, no study has yet addressed why these SStuBs are not identified by static analyzers. Therefore, the main research questions on which we will concentrate are-

- **RQ1:** What categories of *SStuB*s are often missed by a static analyzer called *SonarQube*?
- **RQ2:** What could the probable reasons behind the categories of bugs remaining undetected?

4 Data

We plan on using one of the largest available datasets containing *SStuB*s from open-source Java projects named *ManySStuB*[1]. There are about 153, 652 single statement bug-fix information from 1000 projects categorized under *16* templates.

5 Methodology and Evaluation Strategy

For ***RQ1***, we will be- 1) using *SonarQube* on a feasible number of projects from the dataset, 2) collecting the reported bugs by the analyzer, and 3) extracting the gold data to compare from *ManySStuBs*[1]. To evaluate the efficiency, we will calculate how many Bugs (in percentage) are missed by *SonarQube* overall and by category with respect to the dataset.

For ***RQ2***, the rough idea would be to manually label probable reasons for each of the bugs those were missed by *SonarQube*. Then, we gradually reduce to some general label patterns expressing plausible explanations for bugs being ignored by *SonarQube*.To do so, we must first comprehend how the tool analyses bugs in a codebase, as well as why those bugs are identified. However, we appreciate that evaluating the *RQ2*, will be a bit of analysis-based evaluation. We haven't found any available evaluation methods for the target we are pursuing yet. We'll continue to search this and, hopefully, develop a way for evaluating our output.

*See Appendix for some official websites

References

[1] Karampatsis, R.-M., and Sutton, C. How often do single-statement bugs occur? the manysstubs4j dataset. In *Proceedings of the 17th International Conference on Mining Software Repositories* (New York, NY, USA, 2020), MSR '20, Association for Computing Machinery, p. 573–577.

[2] Lavazza, L., Tosi, D., and Morasca, S. *An Empirical Study on the Persistence of SpotBugs Issues in Open-Source Software Evolution*. 08 2020, pp. 144–151.

[3] Mosolygó, B., Vándor, N., Antal, G., and Hegedüs, P. On the rise and fall of simple stupid bugs: a life-cycle analysis of sstubs. *CoRR abs/2103.09604* (2021).

[4] Tomassi, D. A. Bugs in the wild: Examining the effectiveness of static analyzers at finding real-world bugs. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (New York, NY, USA, 2018), ESEC/FSE 2018, Association for Computing Machinery, p. 980–982.

[5] Tomassi, D. A., Dmeiri, N., Wang, Y., Bhowmick, A., Liu, Y.-C., Devanbu, P. T., Vasilescu, B., and Rubio-González, C. Bugswarm: Mining and continuously growing a dataset of reproducible failures and fixes. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (2019), pp. 339–349.

A Appendix

Links to the official website of some Static Analyzers and Issue Tracking Systems are as follows:

- *SonarQube*: https://www.sonarqube.org/
- *SpotBugs*: https://spotbugs.github.io/
- *ErrorProne*: https://github.com/google/error-prone
- *BugZilla*: https://www.bugzilla.org/
- *Mantis*: https://www.mantisbt.org/
- *JIRA*: https://www.atlassian.com/software/jira