Authors’ Response to the Review of EMSE-D-20-00300: “Fixing Vulnerabilities Potentially Hinders Maintainability”

Sofia Reis, Rui Abreu, Luis Cruz

Editor

> Overall, I would like to give you a chance to respond to these issues in a revision of your manuscript but I have to make it clear that it does not guarantee a path towards acceptance. Your rebuttal to #1 will be of critical importance for me judge the validity of the use of the BCH tool and in making a final decision. Methodology is, of course, very important for ESE journal and #2 & #3s comments on methodological details need to be carefully addressed, too.

Response:
We thank the editor and reviewers for their valuable feedback. In the following points, we address the concerns raised by the reviewers.

Reviewer 1

Reviewer comment 1.1:

> This paper investigates the impact of patches to improve security on the maintainability of open-source software. The paper is well written and easy to read. I really appreciated the replication package: complete, and well described. Really a good work!
>
> However, I found a crucial issue in this paper. The authors adopted as static analysis tool Better Code Hub. Better Code Hub’ model includes 10 guidelines that can help the developers to write better code. Unfortunately, these guidelines are not related to maintainability. Avoiding introducing the issues associated to these guidelines do not imply increasing the maintainability of the code, since the tools has no possibility to measure ”maintainability”.

Response:
Thank you for your positive feedback on the paper presentation and replication package. BCH is not a static analysis tool instead it checks codebases for compliance with the ten guidelines. The set of guidelines presented in our paper is the same published by Software Improvement Group (SIG) in their ebook called ”Building Maintainable Software: Ten Guidelines for Future-Proof Code” [1]. According to the authors, all the guidelines were infered from analyzing hundres of real-world systems and measure maintainability.

We clarified the difference between BCH and static analysis tools in the Section 1 (Introduction)

[1] Available here: https://www.softwareimprovementgroup.com/resources/ebook-building-maintainable-software/

and Section 3 (Methodology).

Reviewer comment 1.2:

> As far as I know, the company that developed Better Code Hub developed also the Delta Maintainability Model [diBiase2019] that aims at measuring the maintainability of a code change and a score and compare change-based maintainability measurements.
>
> [diBiase2019] M. di Biase and A. Rastogi and M. Bruntink and A. van Deursen. The Delta Maintainability Model: Measuring Maintainability of Fine-Grained Code Changes. IEEE/ACM International Conference on Technical Debt (TechDebt) in 2019.

Response:
Thank you for suggesting Marco's paper. Our paper uses a model for measuring maintainability that was published previously by one of the co-authors of this paper [Cruz2019]. The entire methodology and calculations were actually validated by Marco.

The model created by Marco focus on measuring maintainability of fine-grained changes using 5 of the same guidelines we use. The SIG-MM model in Marco's paper is the model behind Better Code Hub.

We added Marco's work to our related work.

[Cruz2019] Luis Cruz, Rui Abreu, John Grundy, Li Li, Xin Xia. Do Energy-oriented Changes Hinder Maintainability?. International Conference on Software Maintenance and Evolution (ICSME) in 2019.

Reviewer comment 1.3:

> Moreover, I am not sure that Better Code Hub is the best static analysis tools to detect security issues. One of the most adopted in security domain is Coverity Scan for examples, but also other tools might be a better choice (e.g. Kiuwan).

Response:
Thank your for your comment. In this paper, we do not try to use static analysis to detect security issues. We have already a dataset of security patches (vulnerable versions and safe versions) and our goal is to understand what is the impact of those patches on the maintainability guidelines/metrics. We arguee that these guidelines/metrics can in the future complement static analysis tools by assiting developers with more information on the risks associated with their patches.

SonarQube uses 4 of the set of metrics we are using and also rates maintainability. In Section X, we compare the results between BCH and SonarQube. However, we arguee that BCH performs a more complete analysis.

Reviewer comment 1.4:

> The authors could have adopted both approaches, Better Code Hub guidelines or another one first and then measure the code maintainability with the Delta Maintainability Model since the

goal of the paper is to improve security on the maintainability.

Response:
Thank you for your comment. As we explained before, while addressing comment 1.2, both models assess maintainability. Delta Maintainability Model is an alternative to our approach that uses only 5 of the set of guidelines our study considers.

Reviewer comment 1.5:

In my opinion, the idea behind this work is very interesting, but the research questions cannot be answered with the tools and metrics selected.

My recommendation is to select a specific static analysis tool for security issues and to include a maintainability model to evaluate the maintenance.

Response:
Thank you for the positive interest on the work. As we explained before Better Code Hub calculates maintainability metrics. Our goal was to measure the impact of security patches on software maintainability. For future research, it would be interesting to understand if the results of these metrics can actually help security engineers assessing the risk of their patches and guide them on performing better patches. However, in this work, we only focus on assessing software maintainability.

Reviewer 2

Reviewer comment 2.1:

Question: Is your dataset available?

Response:
Thank you for your question. The dataset is fully available at `dataset/db_release_security_fixes.csv` in the figshare package provided in the contributions of the paper: `https://figshare.com/s/4861207064900dfb3372`. If the paper is accepted, the package will be available on GitHub.

Reviewer comment 2.2:

Abstract: Throughout the paper, ”hypothesize” is probably a better word than ”suspect” for sounding more scientific.

Your results should be more specific than ”show evidence of trace-off”. Briefly tell us about what metrics you used and what the numerical results indicate.

Response:
We thank the reviewer for the suggestions. We addressed both of the issues in the abstract. For the later issue, we reported the overall result for maintainability and results for the two of the guidelines with more negative impact in software maintainability: software complexity and unit

size.