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.

We would like to rebut your concern about Better Code Hub (BCH)'s guidelines not being related to maintainability. First, as mentioend in the paper, BCH checks GitHub codebases against 10 engineering guidelines as devised by Software Improvement Group (SIG). SIG has devised these guidelies after many years of experience: analyzing more than 15 million lines of code every week, SIG maintains the industry's largest benchmark, containing more than 10 billion lines of code across 200+ technologies; SIG is the only lab in the world certified by TÜViT to issue ISO 25010 certificates.

Second, each of the guidelines in isolation is pretty simple to understand, but, when combined with the others, and, when evaluated as a whole, it becomes easy to see why they would comprise the definition of good software. Furthermore, the relation of the guidelines to maintainability

is well detailed in the SIG's book "Building Maintainable Software: Ten Guidelines for Future-Proof Code" [1]. In fact, the underlying maintainability model for BCH is documented with several (non-academic) reports linked at this page: `https://www.softwareimprovementgroup.com/methodologies/iso-iec-25010-2011-standard/` as well as validated by the academic community in the following paper:

```
Heitlager, Ilja, Tobias Kuipers, and Joost Visser.
‘‘A practical model for measuring maintainability.’’
6th international conference on the quality of information and
        communications technology (QUATIC 2007).
IEEE, 2007.
```

BCH's compliance criterion is derived from the requirements for 4 star level maintainability (cf. ISO 25010). The concrete values of the thresholds used in BCH are documented also in our book Building Maintainability Software https://www.softwareimprovementgroup.com/resources/ebook-building-maintainable-software/.

Further, the methodology for arriving at the thresholds is documented in the papers you already point out. SIG performs the threshold calibration yearly on a proprietary data set to satisfy the requirements of TUViT to be a certified measurement model. However, please note that this data set cannot be shared due to non-disclosure agreements between SIG and its clients.

All in all, we argue that BCH's guidelines are a good proxy for building maintainable software. This is backed up by the observations of SIG in the field, as it was told us in personal communications:

> We saw the value of a tool like BCH in helping in writing clean and maintainable code and most importantly, it offers us so much more:
>
> Over time, the guidelines in BCH will become part of your coding standards, and the overall quality of your work will improve, slowly, but surely.

PAREI AQUI.

We clarified the difference between BCH and static analysis tools in the Section 1 (Introduction) and Section 3 (Methodology).

> Introduction
>
> ...
>
> Static analysis tools (SATs) have been built to automatically detect software vulnerabilities (e.g., FindBugs, Infer and more). Developers use those tools to locate the issues in the code. However, while performing the patches to those issues, SATs are not capable of providing information on the quality of the patch. Improving software security is not a trivial task and requires implementing patches that might affect software maintainability. Our hypothesis is that some of these patches may have a negative impact on the software maintainability and, possibly, even be the cause of

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

the introduction of new vulnerabilities—harming software reliability and introducing technical debt. Research found that 34% of the security patches performed introduce new problems and 52% are incomplete and do not fully secure systems[Li2017]. Therefore, in this paper, we present an empirical study on the impact of patches of vulnerabilities on software maintenance across open-source software. We arguee that tools that assess these type of metrics may complement SATs with valuable information to help the developer understand better the risk of its patch.
...

[Li2017] F. Li and V. Paxson. A Large-Scale Empirical Study of Security Patches. A Large-Scale Empirical Study of Security Patches in 2017.

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 already use. The SIG-MM model in Marco's paper is the model behind Better Code Hub.

We mention Marco's work in our related work:

> Recent work, proposed a new maintainability model to measure fine-grained code changes by adapting/extending the BCH model[diBiase2019]. The present work uses the same model (SIG-MM), considers more guidelines and focuses solely on evaluating the impact of security patches on software maintainability.

[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.

[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.

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.

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 and only uses 5 guidelines 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 assess 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 on software maintainability: software complexity and unit size.

Reviewer comment 2.3:

> Intro:
>
> - First sentence - quality is not ONLY related to cost but also to security and safety.
> - Provide a URL to Software Improvement Group and Better Code Hub.
> - Page 3, Line 1 - Application Security Verification Standard (ASVS).
> - Page 3, Line 7 - instead of a ”broad number of code metrics” - tell us exactly the number of metrics.
> - Page 3, Line 13 - ”suggest” sounds more scientific than ”hint at”
> - Page 3, Line 24 - ”we intend to highlight the need . . . ”. Is that the broad goal of your paper? The goal should be explicitly stated in the abstract and intro.

Response:
Thank you for reporting these issues. All of them were addressed in the new version of the paper. In the last point, where the reviewer asks “Is that the broad goal of your paper?”, we want to clarify that the goal of our study is to show the impact of security patches on software maintainability and highlight the need for solutions which are described in the end of the abstract, Section 1 (*Introduction*) and Section 5 (*Study Implications*).

Reviewer comment 2.4:

> Intro:
> You should check out this paper: Li, Frank and Paxson, Vern. A large-scale empirical study of security patches. ACM SIGSAC Conference on Computer and Communications Security in

2017.

Response:
Thank you for bringing our attention to this paper. We integrated this paper in the Related Work section and used it to support some of our claims and findings.

Reviewer comment 2.5:

Section 2

- The first paragraph is redundant with Section 1.
- Page 4, Line 15 - You define the OCSP acronym late in the paragraph.
- Page 5, Line 6 - how many new branch points?
- Page 6, Line 19 - I think you want to say ”In this study, maintainability is ....” Since it currently implies this information is available in the CWE.
- Page 6, line 47 - ”regular commits” non-security commits to be more clear; maybe a few more words to explain ”randomly collected.” You also call them ”baseline commits” in Figure 1 which gives a different phrase for the same concept. Pick one and use it everywhere.

Response:
Thank you for pointing out all of these issues. We addressed all of them. We decided to use regular changes/regular commits instead of baseline and non-security commits.

Reviewer comment 2.6:

Section 3

- Sometime you say 1330 patches (e.g. page 7, line 16) and sometimes 1300 (e.g. page 8, line 11, and the abstract). In science it's better to say the exact number and use it always and not round.
- You say the dataset had 1330 (or 1300) patches and 1282 commits though you also say one patch can have multiple commits assigned. So how did you end up with less than one commit/patch?

Response:
Thank you for raising this concern. The right number of security patches is 1300 (624 from Ponta et al. and 676 from Secbench). The 1330 value is a typo, we fixed the issue. Regarding the second point, 1282 commits is the equivalent to the 624 patches that integrate the Pontas et al. dataset where one patch can have multiple commits. The 1282 commits are referring to the Ponta et al. dataset and not to the combined dataset.

Reviewer comment 2.7:

Section 3

- I would really like a better picture of what SIG is to establish credibility. I was at first thinking it had something to do with ACM SIG's but now after going to the website, it seems like a commercial/consulting organization? I also feel irritated that I don't understand what you really mean by "running against the BCH toolset" (page 8, line 13) since you have repeatedly mentioned BCH like it was an industry standard - but you have not told me if it's a static analysis tool or what it is. The explanation and link to introduce SIG and BCH should be on Page 2 lines 41-42 rather than Page 7 - though an explanation of the BCH toolset is best a paragraph in the methodology.

Response:
Thank you for raising the concern regarding SIG's credibility. SIG is a company that has more than 20 years of experience and research in software quality production. Their models are scientifically proven and certified. We added this information and the links the reviewer refer in the comment to the Introduction section.

> As ISO does not provide any specific guidelines/formulas to calculate maintainability, we resort to Software Improvement Group (SIG[2])'s web-based source code analysis service Better Code Hub (BCH)[3] to compute the software compliance with a set of 10 guidelines/metrics to produce quality software based in ISO/IEC 25010 [Visser2016]. SIG has been helping business and technology leaders drive their organizational objectives by fundamentally improving the health and security of their software applications for more than 20 years. Their models are scientifically proven and certified [Alves2010, Alves2011, Baggen2012].

We rephrased "running against the BCH toolset" to "analyzed using the BCH toolset". We mean that we used BCH to calculate the metrics presented on Table 1.

[Visser2016] J. Visser. Building Maintainable Software, Java Edition: Ten Guidelines for Future-Proof Code. O'Reilly Media, Inc. in 2016.

[Alves2010] T. L. Alves and C. Ypma and J. Visser. Deriving metric thresholds from benchmark data. IEEE International Conference on Software Maintenance in 2010.

[Alves2011] T. L. Alves and J. P. Correia and J. Visser. Benchmark-Based Aggregation of Metrics to Ratings. Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement in 2011.

[Baggen2012] R. Baggen, J. P. Correia, K. Schill, J. Visser. Standardized code quality benchmarking for improving software maintainability. Software Quality Journal in 2012.

Reviewer comment 2.8:

[2]SIG's website: `https://www.sig.eu/` (Accessed on January 31, 2021)
[3]BCH's website: `https://bettercodehub.com/` (Accessed on January 31, 2021)

> Section 3
>
> - Page 7, line 4 - I don't think "resemble" is the word you are looking for - but I don't know what you are trying to say so I can't make a suggestion.

Response:
Thank you for raising this issue. In the sentence "The codebases that resemble to the commits of our datasets are collected before the analysis performed by BCH.", we mean that the codebases of each commit in our dataset were analyzed by BCH. We improved the text for:

> BCH evaluates the codebase available in the default branch of a GitHub project. We created a tool that pulls the codebase of each commit of our dataset to a new branch; it sets the new branch as the default branch; it runs the BCH analysis in the codebase; and, finally saves the BCH metrics results.

Reviewer comment 2.9:

> Section 3
>
> - Please explain your repeatable methodology to "clean the dataset and select the most relevant and compatible projects" (page 8, line 14) that brings the patches from 1282 to 969. From 969 to 866 patches - you just could not classify? Please explain.

Response:
As explained in comment 2.6, the 1282 commits refer only to the Ponta et al. dataset (a total of 624 patches). In total, we analyzed 1300 patches. However, BCH as some limitations such as analyzing projects with very large codebases and was uncapable to analyze some patches. In addition, we also detected floss-refactorings which we decided not to consider. All of this is explained in more detail later in Section 3.4.

> Security patches can be performed through one commit, several consecutive commits, or commit(s) interleaved with more programming activities (floss-refactoring). This study considers mainly single-commit patches. Yet, a small percentage of data points had more than one commit involved in the patch. We manually inspected these cases and 23 floss-refactorings were identified and disregarded since it would not be fair to measure the maintainability of these cases where other programming activities are involved.
>
> For projects with large codebases, the results retrieved for *Keep Your Codebase Small* were not viable because they retrieved values above the limit set by BCH (20 Person-years). The *Automated Tests* guideline was also not considered since the tool does not contemplate two of the most important techniques to security testing: vulnerability scanning and penetration testing. Instead, it only contemplates unit testing. Due to BCH limitations, we detected a few data points that retrieved incorrect overall calculations. Those data points were disregarded from the study.

Those are the cases that were tossed from the initial 1300 patches.

Regarding the second question, thank you for pointing out ... (still working on this)

Reviewer comment 2.10:

> Section 3
>
> - Page 8, line 33 ”regular commit” - pick baseline or regular (or non-security) commits and use it always. I don’t understand the sentence ”The baseline dataset is generated from the security commits dataset.” Do you mean you extract these commits from the projects in the security commits dataset?

Response:
Yes, for each security commit we collected a random regular change from the same project.

Reviewer comment 2.11:

> Section 3
>
> - Section 3.2 is confusing enough that I’m not sure of the analysis - I will need to review the revision that you would submit to gain confidence. Are you saying you want your non-security commits (regular commits) to be about the same size as a given security commit so you are using this criteria to choose your regular commits? Justify this reasoning and make the methodology clear. I would have expected that you randomly collected 1300 non-security commits to compare with the 1300 security commits. Why constrain to make them be of ”similar” size? Why not allow the characteristics of regular versus security commits to be what they are without such curating?

Response:
Yes, we are looking for regular changes with the same size as security commits. We have collected a baseline of random changes only (no size restrictions) in the past and performed the same evaluation. We now present the results for both baselines: *size-baseline*, where we consider size appproximation; and, *random-baseline*, where we consider all the regular changes characteristics.

Reviewer comment 2.12:

> Section 3
>
> - Since SIG seems to be ”just another consulting organization” - I’d like to see about the acceptance of the 10 guidelines in the software engineering discipline. I would guess you could find peer-reviewed references to support these guidelines, or even books by Martin Fowler. That has more credibility than the reference by the SIG consulting group. For example on Page 9, line 42 you cite McCabe Complexity – that has more credibility than a consulting organizations guidelines that they use to make money on their tool. Tell us

> more about the BCH data experience since that is your comparison point for compliance.

Response:
(I'll do this in the end.)

Reviewer comment 2.13:

> Section 3
>
> - Page 10, line 49 "mainly single-commit patches" ... "small percentage of data points" - give us the exact numbers for each of these. Page 11, line 38 - define "floss refactoring" and justify how these were objectively and repeatably identified in your set. You say "these cases" which makes it seems that you were including the set of patches that had more than one commit as your (only) floss candidates.

Response:

Reviewer comment 2.14:

> Section 3
>
> - Page 11, Line 42 - I think you are saying you removed these two guidelines from the analysis of all projects (as shown in Figure 4) but this paragraph seems to say this was only for projects with large code bases.

Response:
working on this

Reviewer comment 2.15:

> Section 3
>
> - Page 11 line 48. How many were disregarded?

Response:
Due to BCH limitations, 308 data points were disregarded.

Reviewer comment 2.16:

> Section 4
>
> - Page 13, Line 3 - 969 security commits and 969 baseline commits? It's hard to know how to parse that.

Response:

Reviewer comment 2.17:

> Section 4
>
> - Page 13, Line 31 ”overall patches . . .” rather than having us rely on looking at the Figure, tell us the number or percentage of when the impact is positive like you do with the negative in the next sentence. [You do provide the specifics starting on page 15 line 49. I wanted it here.]

Response:

Reviewer comment 2.18:

> Section 4
>
> - Page 19, Line 7 - for the unindoctrinated - I’d explain the concept of Research Concepts. Also explain your choice of 7a and 7b and how representative they are from the several hundred choices. Similar for your explanations in the second two paragraphs on this page. How far down the concept/vulnerability hierarchy tree are these CWE choices or are they lower level vulnerabilities chosen from the 700+ CWE types?

Response:

Reviewer comment 2.19:

> Section 4
>
> - Page 20, Line 42 - how many is ”a considerable number of cases”?

Response:

Reviewer comment 2.20:

> Section 4
>
> - Page 20, Line 45 - it seems choosing commits of similar size to the security commits ended up causing a limitation. I’m still wondering why you chose that methodology anyway.

Response:

Reviewer comment 2.21:

> Section 4
>
> - RQ3 is answered to succinctly and with too little detail to be convincing to me that the differences between security and regular changes is different.

Response:

Reviewer comment 2.22:

> Section 5
>
> - Line 32 - since RQ3 was not convincing, I don't feel a separate effort for maintainable security is justified as separate from general focus on maintainability.

Response:

Reviewer comment 2.23:

> Section 6
>
> - It might not be a threat - but I feel the curation of the non-security commits is a limitation to the generalizability of the characteristics of non-security commits

Response:

Reviewer 3

Reviewer comment 3.1:

> The paper addresses a relevant and up to date topic. The authors have analyzed a large set of project wrt to quality assessment focused on vulnerabilities in line with ISO25000 standard. There are some points worth considering: - authors should better motivate the choice for BCH as web based source code analysis service with respect to other options that are available in literature. Tools such as Kiuwan, sonar cloud, codify etc are also able to provide specific metrics on maintainability and security, vulnerability features of code. Furthermore Kiuwan is also an example designed based on the ISO25000 standard. Authors discuss the need for tools for risk assessment in the study implications (section5) but fail to assess why their choice falls on BCH.

> Furthermore all tools provide different conceptualization and evaluation criteria for calculating quality characteristics such as Vulnerabilities and Security. Authors should also explicit how these quality characteristics are measured in BCH.

Response:
We kindly thank you the reviewer for the comments. We added a sub Section to the paper called *Why Better Code Hub?*. We compare BCH with other tools that perform code analysis and motivate why we find BCH to be the best fit for us.

Reviewer comment 3.2:

> In section 3a better state/formulate the hypothesis authors analyze wrt to the Research questions in previous sections. In general the empirical study is not structured according to any of the guidelines or explicit states any of them.

Response:

Reviewer comment 3.3:

> The findings of the research should better emphasize what the lessons learned and what better practices should be put in place by developers to assure better quality of software. The take away message remains hindered.

Response: