Dear Editor and Reviewers,

We thank the reviewers for their comments. We have carefully revised the paper following these comments.

The major changes include:

1. In Section 1, we follow the 3rd reviewer’s suggestion by adding more detailed comparison with [HCW+19].

2. In Section 2, we follow the 1st reviewer’s suggestion by giving motivation examples to illustrate the motivation to define the semantics of AMASS.

3. In Section 6, we show the reason why we use “ValApp” to validate the semantics of AMASS, and move the source code to Appendix to keep a better readability.

4. In Section 8,

(1) we give the website of TaskDroid,

(2) we use cross-verification method involving 4 individuals to check manually the unbounded app,

(3) we add an experiment by using Monkey tool to randomly click the unbounded apps to show the relationship between abnormal behaviors and unbounded apps,

(4) we show the reason why TaskDroid can report more unbounded apps when using ICCBot\_{AMASS}.

~~-------------------------------------~~--------------------------------

Below are the answers to the reviewer’s questions and comments.

**Review 1**

**Q1**. The paper discusses the formal semantics introduced, but the practical implications and benefits of this approach remain unclear. Although it is mentioned that "the existing formal semantics is far from complete" and the treatment of activities as atomic objects abstracts away internal structures like fragments, there is a lack of intuitive understanding for the reader about the problems this abstraction causes in static analysis or the advantages the proposed semantics provide over it. I recommend that the authors precede the formal definition with a real-world code example demonstrating these advantages. For instance, the example could illustrate what errors cannot be detected by static analysis methods based on existing definitions, why these errors are missed, and what structures or semantics should be considered to rectify this. Introducing formal definitions directly can be abstract and challenging for readers to engage with without practical, contextual examples.

**A**. We give the motivation examples to show the models ATG, LHR17, AFTG report the negative task or fragment stack unbounded apps, and why AMASS dose not report these problems.

**Q2**. Beginning from line 1969, or Section 6.2, the exposition by the authors becomes puzzling. They suddenly mention the selection of ten applications from F-Droid but do not discuss the criteria used for choosing these ten applications or justify why only ten were chosen. Furthermore, the rationale behind creating a new application instead of experimenting on a larger pool of existing applications is not made clear. Is it to be inferred that it is challenging to find real-world applications that align with the specific semantics discussed? If so, what then is the real-world relevance of these semantics? It is crucial for the authors to clarify these points to enhance the credibility and applicability of their research findings.

**A**. We give the reason why we use “ValApp” to validate the semantics of AMASS, that is the intent flags may interfere with each other, the real world apps may not use the combination of intent flags which some flags will be ignored. However, we need to validate all possible combinations of intent flags to ensure the correctness of the semantics of AMASS, even though most combinations are not used in the real world apps.

**Q3**. Starting from line 2680, the authors describe their approach of selecting a sample of detected errors for manual verification to determine if the 100 apps are indeed subject to task/fragment-container unboundedness by checking the executability of the witness cycles reported by TaskDroid. This manual verification process is prone to introducing errors due to subjective interpretation and potential oversight. To mitigate these risks, it would be advisable for the authors to employ a cross-verification method involving multiple individuals to ensure the reliability of the findings. If such a procedure has already been implemented, it would be beneficial for the authors to explicitly state this in the paper to reinforce the validity of their verification process.

**A**. We use cross-verification method involving 4 individuals to check manually the unbounded app.

**Q4**. In line 2713, the authors describe using UIAutomator to execute the click-event sequence generated when simulating the witness cycle repeatedly until some abnormal behavior such as crashes or other malfunctions occurs. Based on these observations, they conclude that "These suggest that task unbounded and fragment-container unbounded apps can be potentially harmful to the Android system, highlighting the importance of such an analysis." I believe this argument is highly imprecise. The crashes and reboots of mobile applications can be caused by a variety of reasons, and it is common for mobile apps to experience such issues. The authors have not demonstrated a causal link between the observed crashes and the task unbounded and fragment-container unbounded conditions through code analysis or other technical proof. Consequently, the conclusion drawn lacks a solid foundation and can be misleading. It is crucial for the authors to provide a more rigorous analysis to substantiate their claims regarding the potential harm of such vulnerabilities.

**A**. To show the link between the observed crashes and the task unbounded and fragment-container unbounded conditions, we add an experiment by using Monkey tool to randomly click the unbounded apps as baseline. It is hard to show the link through code analysis, since we only can verify there exists such “witness cycles” in the code, but it is still hard to prove it will cause the observed crashes.

**Q5**. In lines 2905 to 2916, the authors note that the use of ICCBotAMASS results in the highest number of task unbounded apps being identified. However, they fail to establish that a greater number of identified apps necessarily correlates with superior detection. This could potentially indicate a higher rate of false positives rather than true positives. To substantiate the effectiveness of ICCBotAMASS over other methods, the authors should provide examples from real mobile applications, including an analysis of the code, to demonstrate that genuine task unbounded apps were indeed detected exclusively by ICCBotAMASS and not reported by other methodologies. This would provide a more concrete basis for their claims and help validate the precision and utility of ICCBotAMASS in identifying task unbounded apps.

**A**. The extra models constructed by ICCBot\_{AMASS} are extracted dynamically from the apps that cannot be decompiled, and these models cannot be extracted by the two other extractors. Moreover, we have confirmed the apps in Table 18 are indeed task unbounded apps, and these unbounded apps only reported when using ICCBot\_{AMASS}.

**Review 2**

**Q1**. Section 3.1 gives semantics of AMASS for Android 13.0, in comparison to that for Android 12.0 in FAC-2022-0049. However, no major changes are found. It is OK if the there is no change on Android's multitasking mechanism itself, but the paper should state this clearly.

**A**. The semantics of AMASS for Android 13.0 is the same as Android 12.0. We have stated this in Section 3.2.1.

**Q2**. Section 7 has been undertaken major changes with the details of Android OS auditing and app analysis. This part needs to be more concise though. In particular, the code snippets should move to appendix to keep a better readability.

**A**. We have moved the code snippets to appendix.

**Q3**. Evaluation on apps should include more that targets Android 13.

**A**. We add the experiments for Android 13.

**Q4**. In the revision, the authors should provide their response to the review of FAC-2022-0049.

**A**. We provide it in the end of this response.

**Q5**. The paper should conduct another literature review to study related work between the previous submission and this submission.

**A**. We have conducted the literature review, but there is no more related work between the previous submission and this submission.

**Review 3**

**Q1**. One suggestion is regarding the reproduction of the application tool. In particular, the application tool, Taskdroid, is not made available, which may impede reproducibility.

**A**. we give the website of TaskDroid.

**Q2**. A more detailed comparison with [HCW+19] would be even more helpful.

**A**. In Section 1, we add more detailed comparison with [HCW+19].

**The review of FAC-2022-0049.**

Dear Editor and Reviewers,

We thank the reviewers for their comments. We have carefully revised the paper following these comments.

The major changes include:

1. We follow the 1st reviewer’s suggestion by restricting the scope of this paper to the multitasking mechanism between activities and fragments.

2. We improve the presentation of Section 3 as follows.

(1) To simplify the definition of the semantics of AMASS𝐴𝐶𝑇,𝐼𝐹, we choose to skip the NOH flag in Section 3.1 (as a result, the definition of formal semantics is shortened from over 3 pages to around 1 page). The full semantics of AMASS𝐴𝐶𝑇,𝐼𝐹 is included in the appendix.

(2) Before the technical definition of the formal semantics, we use examples to help the readers get an intuitive understanding of the semantics first.

3. In Section 6, besides the semantics-validation experiments, we also audit the source code of the activity-fragment multitasking mechanism in Android OS to confirm the consistency of the formal semantics to the source code. The manual code audit and automated semantics-validation experiments both increase the confidence in the correctness of the formal semantics.

4. We add discussions on the limitations of the dynamic model extraction in Section 4. The effect of the dynamic model extraction on the performance of ICCBot\_{AMASS} is demonstrated by the experiment results in Tables 9 and 10.

5. We discuss the implications of their work in practice in the conclusion.

~~-------------------------------------~~--------------------------------

Below are the answers to the reviewer’s questions and comments.

**Review 1**

**Q1**. I still have doubts on the claim from the authors about the multi-tasking mechanisms in Android. The authors indicate that all of the high-level multitasking is implemented via (relatively) lower-level inter-component interaction/communication. But, for different types of high-level multitasking mechanisms for different components, the low-level implementation and modelling can be different. For example, the intent flags used for activities can be totally different from those used for starting Services. In addition, Android also supports other types of multi-threading such as AsyncTasks. However, in Section 2, the authors only use multi-tasking to refer to “activities and fragments, and the evolution of the back stack”, which, I think, are only a type of multitasking on Android. I think the authors should better restrict the scope of this paper. Instead of claiming that they are formalizing the multitasking mechanism of Android, they should explicitly specify that they are formalizing multitasking between activities and fragments.

**A**. We restrict the scope of this work to the Android activity-fragment multitasking mechanism. Parts of this paper, including the title, abstract, and introduction, have also been adapted accordingly.

**Review 2**

**Q1**. I thank the authors for adding more details regarding the cross-version difference in the mentioned aspects. However, the current Section 3 is a bit unorganized. The authors should consider organize them into subsections or tables for better readability.

**A**. We organize Section 3.2 (semantics of AMASS\_ACT for the other versions of Android) into subsections.

**Q2**. Please add discussion on the limitation of using dynamic testing for model extraction, and clarify how this could affect the performance of ICCBot\_{AMASS}.

**A**. We have added the discussions on the limitation of dynamic model extraction in Section 4 and clarified that the dynamic model extraction in ICCBot\_{AMASS} is slower than the static model extraction in general. The experiment results for the performance comparison can be found in Section 8 (Table 9-10).

**Review 3**

**Q1**. Although the authors claim to rewrite Section 3 completely, this section is still not understandable. As a consequence, the correctness of the developed formal semantics for Android system is still hard to judge. The authors' response "the precision/correctness of the model extraction is somehow not central to the goal of this paper" somehow sounds very strange to me.

The presentation of the paper, especially those technical parts, is not much improved. (Some examples: Section 3.1 - transition relation, Semantics of AMASS𝐴𝐶𝑇,𝐼𝐹 pp15-19.)

**A**. We improve the presentation of Section 3 as follows.

(1) To simplify the definition of the semantics of AMASS𝐴𝐶𝑇,𝐼𝐹, we choose to skip the NOH flag in Section 3.1 (as a result, the definition of formal semantics is shortened from over 3 pages to around 1 page). The full semantics of AMASS𝐴𝐶𝑇,𝐼𝐹 is included in the appendix.

(2) Before the technical definition of the formal semantics, we use examples to help the readers get an intuitive understanding of the semantics first.

**Q2**. The correctness of the proposed formal model is difficult to judge. The validation of the formal semantics in Section 6 is insufficient.

**A**. In Section 6, besides the semantics-validation experiments, we also audit the source code of the activity-fragment multitasking mechanism in Android OS to confirm the consistency of the formal semantics to the source code.

**Q3**. The authors should discuss what the implications of their work in practice.

**A**. We discuss the implications of the work in practice in the conclusion.

**Q4**. The authors should also discuss whether the proposed formal model is still relevant for later Android versions.

**A**. We define the semantics of AMASS for the latest Android version (i.e. Android 13.0) in Section 3 and also discuss the differences in the semantics for the other versions (i.e. Android 6.0-12.0).