RefleXGen:The unexamined code is not worth using

We sincerely thank the reviewers (Reviewer1, Reviewer2, and Reviewer3) for their thoughtful and constructive feedback. We have carefully revised and refined the manuscript to address the issues identified, particularly those concerning unclear expressions in the figures and text. The updated content is reflected in the revised version. Additionally, for the writing deficiencies and shortcomings mentioned by Reviewer1 and Reviewer3, we provide detailed explanations of the specific improvements made:

1. Regarding the term “also meets safety standards”—we clarify that “safety standards” specifically refer to the generated results being verified as secure through CodeQL security checks. In the current related research, many high-quality studies employ CodeQL, an objective tool, for security assessment. We have briefly mentioned this in Section 4.3. We will further clarify this point in the revised manuscript and highlight the corresponding content to ensure accuracy. (Reviewer 1:4) (Reviewer1:4)
2. In Figure 1, the font size of “RefleXGen” was indeed disproportionately large, which could distract from the overall visual balance of the figure. Additionally, certain symbols in Equation 4 (e.g., fb0) lacked sufficient explanation. These issues have been addressed in the revised version; we have added the relevant explanations to ensure clarity and readability. Furthermore, we have made additional corrections and improvements to address the incompleteness of the figures. (Reviewer1:1&3, Reviewer3:3)
3. Regarding our method’s steps and modules, it consists of two main steps: Step 1—Initial Code Generation, and Step 2—Reflection and Optimization. The overall method comprises three main modules: ①Initial Code Generation from Step 1, and ② Knowledge-Driven Safety Feedback, and ③Defect Repair and Knowledge Integration from Step 2. To provide clearer and more effective feedback, we have introduced corresponding symbols in the figure to better align the visual representation with the textual description. Additionally, we have further improved the graphics to enhance clarity and coherence. The iterative process of generating secure code, as depicted in the figure, is specifically realized through continuous reflection and iteration. Recognizing that this part was somewhat implicit, we have optimized the corresponding expressions to make these indications more apparent. (Reviewer1:2).
4. Reviewer3 highlighted concerns regarding the expressions of the motivation and limitations of existing works. We are pleased to have the opportunity to further elaborate on the innovative aspects of the proposed RefleXGen method. RefleXGen aims to improve the quality and security of code generated by large language models. Its core novelty lies in leveraging the models’ own reflective capabilities and knowledge accumulation to achieve self-improvement in code security. It does not rely on security optimization of the training set, does not require additional security fine-tuning of the model—which would consume significant computational resources—and does not need any external supplementary knowledge for model verification.

   RefleXGen enhances security solely through self-reflection during task execution and iterative accumulation of security knowledge.

   In our method, each iteration builds upon the results of the previous one, combining reflection on prior outputs to iteratively update a dynamic knowledge base. This dynamic knowledge base continuously evolves, enabling the system to more effectively identify and resolve potential security vulnerabilities over time. Specifically, the system uses the original pre-trained model to reflect and assess potential defects in the output. If the generated output is considered defect-free, the code is output directly; otherwise, the reflection process guides defect correction while updating the knowledge base to prevent similar issues in future iterations.

   This iterative feedback loop forms the foundation for our method’s ability to dynamically and adaptively enhance code generation security, and it can be conveniently applied to current open-source and closed-source models. Experiments have demonstrated that our method significantly improves both the quality and security of the generated code. (Reviewer3:3)

We sincerely thank the three reviewers for their valuable suggestions and insightful comments. We have carefully considered each of the reviewers’ remarks and have revised and improved the manuscript accordingly. We hope that these enhancements not only elevate the overall quality and clarity of the paper but also that the research findings of RefleXGen provide new insights and substantial contributions to the advancement of the field.