Qeios

Peer Review

Review of: "MVD: A Multi-Lingual Software Vulnerability Detection Framework"

Jiamou Sun[1]

1. Data61, The Commonwealth Scientific and Industrial Research Organisation, Canberra, Australia

Summary

The blog proposes a multi-language framework designed to harness the synergies across programming languages to detect vulnerabilities. Specifically, it uses CodeBERT and adopts a multi-language vulnerability dataset in the training process. Besides, it further leverages incremental learning to assist in detecting vulnerabilities in new languages. Their experiments show that the proposed method surpasses LineVul by 83.7% to 193.6% in PR-AUC.

Strengths

- Considering multi-language vulnerability detection
- Leveraging incremental learning to adopt new programming languages
- Good performance

Weaknesses

- Some claims are not rigorous and clear enough
- Needs to compare with more advanced baselines
- Needs better presentation for some tables

I'm glad to see the authors consider the fact that the software program can involve multiple languages, which is usually ignored by other works. The experiments comprehensively show the good

doi.org/10.32388/8XZoEW

performance of the proposed method, including the effectiveness of incremental learning in adopting new languages.

However, several issues significantly affect the quality of the report. First of all, some claims in the report are not rigorous and clear enough. In Section II-B, the authors mention three challenges in multi-language vulnerability detection, but it seems the first and second points (effectiveness of the combined model and effectiveness of CodeBERT in multi-language) are overlapping. In Section III-A, the authors claim CodeBERT is the SOTA model in vulnerability detection. Nevertheless, Unixcoder[1] and GraphCodeBERT[2] have been proved to have better performance in many SE tasks, not to mention that large language models (LLMs) often have more powerful performance. This also affects the rigor of the experiments, as lots of advanced baselines are missed. In Section V-A, the authors mention MVD has a 30.7% improvement compared to LineVul, but later they argue MVD has 137.9% better performance. I think the authors use different methods to do the calculation without clear description, which is confusing for me. Besides, in the same section, the authors claim training 5 individual LineVuls consumes more resources, which needs further clarification due to the fact that MVD is trained on a larger training set.

Second, as mentioned before, new advanced baselines should be considered, as LineVul was proposed in 2022 and some recent works have been proposed[3][4]. Third, I do not think the cross-comparison among different LineVuls in Table II is necessary. I believe comparing the MVD with LineVul trained in the corresponding language is enough.

Overall, although the report highlights some important issues, it needs further improvement to be more rigorous.

References

1. ^ *Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, et al. (2022). UniXcoder: Unified Cross-Modal Pre-training for Code Representation. doi:10.18653/v1/2022.acl-long.499.*
2. ^ *Guo et al.. (2021). GraphCodeBERT: Pre-training Code Representations with Data Flow. Arxiv.*

3. ^Moumita Das Purba, Arpita Ghosh, Benjamin J. Radford, Bill Chu. (2023). Software Vulnerability Detection using Large Language Models. doi:10.1109/issrew60843.2023.00058.
4. ^Ding et al.. (2024). Vulnerability Detection with Code Language Models: How Far Are We?. Arxiv.

Declarations

Potential competing interests: No potential competing interests to declare.