Reducing False Positives of Static Program Analysis in Industry

Anonymous Author(s)

I. INTRODUCTION

Static program analysis (SPA) is commonly used to improve software quality by analyzing code and detecting defects [1, 2] before the program runs. However, one of the major drawbacks of static program analysis is the high rate of false positives, which can lead to time and efforts waste as developers must manually review each report to determine its validity.

To mitigate this problem, researchers have proposed various approaches. For example, TASMAN [3], for instance, leverages symbolic execution to reduce false positives by tracking the data flow through the program. Phoenix [4] uses machine learning technique to detect potential vulnerabilities. Interactive static analysis is a promising technique to solve this problem, and EUGENE [5] is a typical representative of this approach. By providing limited amounts of user feedback which mark whether the report is true positive or false positive, EUGENE can significantly reduce the number of false positives.

II. CHALLENGES

However, EUGENE faces two main challenges when applied in industry:

Challenge one: User feedback cannot be reused in continuous integration process. A static program analysis algorithm normally consists of a series of rules that are used to identify program vulnerabilities. When a series of rules produces a false positive, other reports derived by the same series of rules may be also false positives. Therefore, when a small number of false positives are flagged by user, it may uncover more false positives derived from the same series of rules.

However, a false positive report conformed by users is associated to a specific code line, such as "there is a memory out-of-bounds error in line 68." After making code changes in a next version, it is challenging to determin whether the memory out-of-bounds error in the previous version still exists for two reasons:

- The code logic may have changed.
- The associated code line number may have changed.

Therefore, user feedback for a previous version cannot be reused directly in subsequent versions. We need to find new approaches to reuse user feedback in continuous integration process.

Challenge two: Unsorted reports lead to inefficient manual confirmation. A static program analysis tool typically outputs a large number of reports, and the labor resources are limited in practice. So manually confirming all reports is a time-consuming task. Existing tools output reports with a mixture of true positives and false positives making it difficult to pick out all the true positives by hand. Sorting the reports based on their probability of being true positives would be beneficial to improve the efficiency of manual confirmation.

REFERENCE

[1] Zhenjiang Dong, Hui Ye, Yan Wu, Shaoyin Cheng, and Fan Jiang. Android Apps: Static Analysis Based on Permission Classification[J]. ZTE Communications, 2013, 11(1): 62-66.

[2] TANG Kai. Risk Analysis of Industrial InternetIdentity System[J]. ZTE Communications, 2020, 18(1): 44-48.

[3] Arzt S, Rasthofer S, Hahn R, et al. Using targeted symbolic execution for reducing false-positives in dataflow analysis[C]// Acm Sigplan International Workshop on State of the Art in Program Analysis. ACM, 2015:1-6.

[4] Pistoia M, Tripp O, Lubensky D . Combining Static Code Analysis and Machine Learning for Automatic Detection of Security Vulnerabilities in Mobile Apps[M]. 2017.

[5] Mangal R, Zhang X, Nori A V, et al. A user-guided approach to program analysis[C]// Joint Meeting on Foundations of Software Engineering. ACM, 2015:462-473.