1. Evaluating and comparing memory error vulnerability detectors

Nong, Yu (1); Cai, Haipeng (1); Ye, Pengfei (1); Li, Li (2); Chen, Feng (3)
Source: *Information and Software Technology*, v 137, September 2021; **ISSN:** 09505849; **DOI:** 10.1016/j.infsof.2021.106614; **Article number:** 106614; **Publisher:** Elsevier B.V.
Author affiliation: (1) School of Electrical Engineering and Computer Science, Washington State University, Pullman; WA; 99164, United States (2) Faculty of Information Technology, Monash University, Melbourne; 3800, Australia (3) Department Of Computer Science, University of Texas, Dallas; TX; 75080, United States
Abstract: Context: Memory error vulnerabilities have been consequential and several well-known, open-source memory error vulnerability detectors exist, built on static and/or dynamic code analysis. Yet there is a lack of assessment of such detectors based on rigorous, quantitative accuracy and efficiency measures while not being limited to specific application domains. Objective: Our study aims to assess and explain the strengths and weaknesses of state-of-the-art memory error vulnerability detectors based on static and/or dynamic code analysis, so as to inform tool selection by practitioners and future design of better detectors by researchers and tool developers. Method: We empirically evaluated and compared five state-of-the-art memory error vulnerability detectors against two benchmark datasets of 520 and 474 C/C++ programs, respectively. We conducted case studies to gain in-depth explanations of successes and failures of individual tools. Results: While generally fast, these detectors had largely varied accuracy across different vulnerability categories and moderate overall accuracy. Complex code (e.g., deep loops and recursions) and data (e.g., deeply embedded linked lists) structures appeared to be common, major barriers. Hybrid analysis did not always outperform purely static or dynamic analysis for memory error vulnerability detection. Yet the evaluation results were noticeably different between the two datasets used. Our case studies further explained the performance variations among these detectors and enabled additional actionable insights and recommendations for improvements. Conclusion: There was no single most effective tool among the five studied. For future research, integrating different techniques is a promising direction, yet simply combining different classes of code analysis (e.g., static and dynamic) may not. For practitioners to choose right tools, making various tradeoffs (e.g., between precision and recall) might be inevitable. © 2021 (42 refs)
Main heading: C++ (programming language)
Controlled terms: Errors - Open systems
Uncontrolled terms: Benchmark datasets - Efficiency measure - Evaluation results - Overall accuracies - Performance variations - Precision and recall - Quantitative accuracy - Vulnerability detection
Classification Code: 723.1.1 Computer Programming Languages
Funding Details: Number: W911NF-21-1-0027, Acronym: ARO, Sponsor: Army Research Office;
Funding text: We thank the anonymous reviewers for their constructive comments. This research was sponsored by the Army Research Office, United States of America and was accomplished under Grant Number W911NF-21-1-0027. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.
Database: Compendex
Data Provider: Engineering Village