Software Vulnerability Detection by Fuzzing and Deep Learning

Yang Xiang[0000−0001−5252−0831]

Swinburne University of Technology, Australia
yxiang@swin.edu.au

Abstract. In this keynote talk, we introduce the current research and trend in the software vulnerability detection research. Then we present a series of novel approaches to deal with the vulnerability issues, such as the fuzzing, deep learning, and the combined approach to improve the effectiveness of the detection.

Keywords: Cybersecurity, Cyber-attacks, Software, Vulnerability, Deep learning, Fuzzing.

1 Introduction

Cybersecurity has become one of the top priorities in the research and development agenda globally today. New and innovative cybersecurity technologies that can effectively address this pressing danger are critically needed. Data-driven and system approaches to solve cybersecurity problems have been increasingly adopted by the cybesecurity research community. They have two areas of focus: detection and prediction of cyber threats. Recently, there have been significant efforts to detect soft-ware vulnerabilities. New methods and tools, consequently, must follow up to adapt to this emerging security paradigm.

Software lifecycles include development, deployment and usage, and mainte-nance phases. Current practice involves avoiding vulnerabilities during the develop-ment phase and removing vulnerabilities via patching during the deployment and usage phase. In fact, due to rampant security breaches in software, detecting vulner-abilities in different stages of software lifecycles has never become an easy job. Ex-isting methods face significant technical challenges caused by code representation, computational resource constraints, and algorithm bottlenecks.

2 Summary of the Talk

In this talk, we introduce the current research and trend in the software vulnerability detection research. Then we will present a series of novel approaches to deal with the vulnerability issues, such as the fuzzing, deep learning, and the combined approach to improve the effectiveness of the detection. The focus of the fuzzing approach is on automatic black-box fuzzing especially for firmware; and the focus of the deep learning approach is to reduce the training samples and improve the detection rate.

Existing techniques take much time to detect and verify vulnerabilities, which is too slow for massive code detection in the development phase. While a bug is an unintended code state, a vulnerability is a bug that attackers can exploit. The code states are usually vast, and the bug locations are unknown, making it difficult to detect bugs efficiently and effectively. In the maintenance stage, developers must reproduce, test, and fix bugs, which may require specific inputs and complex execu-tion states to trigger them. Both deep learning and fuzzing methods have pros and cons in detecting, verifying vulnerabilities, and performance. Current techniques face key challenges that urgently require practical solutions:

Fast detection of massive codes – From the software developer's perspective, one can use pre-trained model to fast detect the vulnerabilities automatically from time to time during the development process by scanning codes. From the user's perspec-tive, static analysis can fast scan the executable codes. Among the static analysis techniques, machine learning, especially deep learning has been used for software vulnerability detection. Deep learning-based static analysis techniques have been widely accepted given that it can automatically extract high feature representations and interpret complex non-linear structure. Thorough detection – One often wants to check every piece of code when it is un-known that whether a potential bug exists. The power schedule determines the num-ber of mutations assigned to each seed, and a seed is an input that makes process in code coverage. While these improvements have considerably decreased the time required to visit different parts of a target application, it is important to understand that code coverage alone is a necessary but not sufficient condition to discover bugs.

Solutions and future research directions will be discussed following above.