Abstract

Modern applications often run across multiple environments. A high-level language can invoke native extensions, typically written in C/C++ code, resulting in more efficient applications and increased productivity since legacy code can be reused. However, the use of native code introduces safety concerns that can lead to security breaches, potentially violating security protocols. In this work, we introduce a novel tool, HybridSE, to analyze Android applications with native code.

HybridSE distinguishes itself by integrating the strengths of established Dynamic Symbolic Execution (DSE) tools—SPF (Symbolic Pathfinder) and CORANA/API, which were originally designed for Java and ARM architectures, respectively. Enhanced with a specialized taint analysis module, HybridSE effectively addresses data leaks in real-world applications and malware, demonstrating a notably low false positive rate in our evaluations.

We assess the performance of HybridSE in two key aspects: control flow and data flow analysis. Regarding control flow, we utilize the generated graphs and apply graph similarity to two tasks: malware family classification and Android packer classification. The graphs generated by HybridSE, when used as features in classification tasks, yield results comparable to those of state-of-the-art classifiers.

In terms of data tracking and detecting data leakage, HybridSE demonstrates higher precision compared to other tools, effectively reducing false positives caused by over-approximation. Unlike static taint analysis tools, HybridSE avoids issues related to array handling and Java reflection. By generating accurate cross-environment control flow graphs for both Java bytecode (.dex) and native code (.so), our taint analysis method has successfully detected 139 data leaks in real-world Android malware. Through our analysis with HybridSE, we have made several observations on how data leaks occur, including a detailed examination of the Lotoor family, which remained active until 2022.

Keywords: Android mobile security, Symbolic execution, Taint analysis, Packer identification, Malware classification