氏　　名	NGUYEN THI VAN ANH		
学位の種類	博士（情報科学）		
学位記番号	博情第 532 号		
学位授与年月日	令和 6 年 9 月 24 日		
論文題目	CROSS-ENVIRONMENT DYNAMIC SYMBOLIC EXECUTION ON ANDROID/APK FILES AND ITS APPLICATION FOR MALWARE ANALYSIS		
論文審査委員	小川瑞史	北陸先端科学技術大学院大学	教授
	Jean-Yves Marion	University of Lorraine	教授
	吉岡克成	横浜国立大学	教授
	Nguyen Le Minh	北陸先端科学技術大学院大学	教授
	Razvan Beuran	同	准教授
	廣川 直	同	准教授

論文の内容の要旨

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

論文審査の結果の要旨

マルウェアは(1) 制御構造隠蔽, (2) 感染, (3) 悪意ある実行の３つのステップからなる。制御構造隠蔽手法は自己暗号化を用い、バイナリコードのパターンをみる構文的解析（Atni-Virus software）のマルウェア検出の成功率は 2015 年の時点で 45%以下といわれる。さらに動的解析（マルウェアの挙動解析）も Sandbox を検知し動作を停止する VMawareness や特定の日時・環境でのみ作動する trigger-based behavior が用いるステルス性の高いマルウェアの解析は困難である。動的記号実行（Dynamic Symbolic Execution）は、もっとも強力な隠蔽解除手法と言われており、x86/Windows に対しては、Angr, Mayhem, Klee-MC, BE-PUM などが知られている。ARM 上の記号実行実装例は限られており、さらに ARM/Android 上ではごく少ない。これは ARM への対応が限られていることに加え、Android/apk ファイルは複数の異なる言語記述（Java の変種である DEX bytecode とバイナリコード）から構成されているため、複数の環境を扱う困難のためである。Angr は x86/win のみならず ARM/Android も扱うが、そのアプローチは共通中間機械語への翻訳ののち、記号実行を扱う。X86/win の場合は VEX、ARM/Android は Jimple をターゲットとする。VEX は機械語レベルであるが、Jimple は Java を想定した bytecode のため、ARM からの変換は限界があり、そのため、Angr の適用範囲はごく限られる。本学位論文では、異なる環境課の動的軌道実行器を結合するフレームワークを設計し、さらに Java の動的記号実行器 SPF（NASA が開発）と ARM の動的記号実行器 CORANA（本研究室で過去に開発）を組み合わせ、ARM/Android 上の Apk ファイルの動的記号実行器を構成した。さらにその有効性の確認のため、DREBIN, AMD, AndroZoo などのデータセットから数万の Android マルウェアサンプルを用いて解析を行った。動的記号実行器の一つの利点は正確な制御フローグラフの生成であり、その Weisfeiler-Lehman グラフカーネル表現を用いてマルウェア分類とパッカー同定による評価を行い、Malscan や DREBIN などの最先端ツールに比肩する性能を確認した。特に制御構造グラフのグラフカーネル表現を用いた機械学習は、しばしば深層学習で問題となる aging / conceptdrift 問題（ある時点では高い性能を示しても、時間の経過とともに新しいデータ上で急速に精度が低下）に対し、上記既存ツールより良好な結果を示している。

以上、本論文は、Android/apk ファイルの動的記号実行器を実用規模で初めて実現し、学術的に貢献するところが大きい。よって博士（情報科学）の学位論文として十分価値あるものと認めた。