JNI를 이용하는 안드로이드 애플리케이션의 개인 정보 유출 탐지를 위한 정적 분석기

*김진섭[1], 김연어[1], 변석우[2], 우 균[1]
부산대학교 전기전자컴퓨터공학과[1], 경성대학교 소프트웨어학과[2]
e-mail : {*kimjinseob, yeoneo, woogyun*}*@pusan.ac.kr*[1], *swbyun@ks.ac.kr*[2]

A Static Analyzer for Detecting Personal Information Leaks in Android Applications Using JNI

*Jinseob Kim[1], Yeoneo Kim[1], Sugwoo Byun[2], Gyun Woo[1]
Dept. of Electrical and Computer Engineering Pusan National University[1],
Dept. of Computer Science Kyungsung University[2]

Abstract

The use of smartphones is increasing and the leakage of user personal information is frequently occurring. Taint analysis tools to prevent such leakage of personal information but they analyze Java byte code only, not the native libraries. Even a few tools including Gangjin's research and Argus-SAF, which can analyze the native libraries, do not report correct results depending on the NDK versions. To solve this problem, we converted the native libraries into the LLVM intermediate representations according to the NDK version, and the resulting intermediate representations are analyzed using the taint analysis method.

I. 서론

안드로이드 애플리케이션의 악성 코드를 검출하기 위해 다양한 분석 도구가 제안되었지만 네이티브 라이브러리에 악성 코드가 포함된 경우에는 이를 검출하지 못한다.

또한, 현재 네이티브 라이브러리를 고려하는 연구로 강진의 연구와 Argus-SAF가 있지만 NDK 버전에 의존적인 문제점을 가지고 있다. 따라서 본 논문은 Argus-SAF를 분석하고 문제점을 보완한 정적 분석기를 제안한다.

II. 본론

2.1 도구 분석

Argus-SAF는 모듈을 계속 확장하는 방식으로 구현되어있다. 시작은 자체 제작한 Jawa라는 중간표현 언어를 분석하는 모듈에서 안드로이드 애플리케이션 분석 모듈인 Amandroid, 그리고 네이티브 분석을 추가한 JN-SAF가 구현되었다.

이러한 Argus-SAF를 이용하여 샘플을 분석하였을 때 몇 가지 오류가 있음을 확인하였다. 첫 번째는 유출 변수의 타입을 변경하였을 때 이를 새로운 값이 덮어진 것으로 판단하여 유출을 판단하지 못하는 것이고. 두 번째는 리스너 내부에서 유출 함수를 호출하면 이를 검출하지 못하는 것이다.

또한, 네이티브 분석 모듈 사용 시에 벤치마크로 제공하는 애플리케이션에 대해서 실험한 결과 아무 문제없이 동작하였지만 동일한 애플리케이션의 소스 코드를 이용하여 직접 빌드한 애플리케이션에 대해서는 이전과 다른 결과가 출력되었다. 이는 빌드 시에 설정한 NDK(native development kit)의 버전에 따라 생성되는 네이티브 라이브러리가 다르고 Argus-SAF는 armeabi 아키텍처에 대해서만 정확한 분석을 하기 때문에 발생한 문제로 생각된다.

2.2 구현

그림 1은 제안 분석기의 구조이다. NDK 버전에 의존적이지 않는 네이티브 분석 모듈 구현을 위해, 네이티브 라이브러리를 LLVM 중간표현으로 변환하고 LLVM 중간표현에 대한 오염 분석을 하는 방식으로 구현하였다.

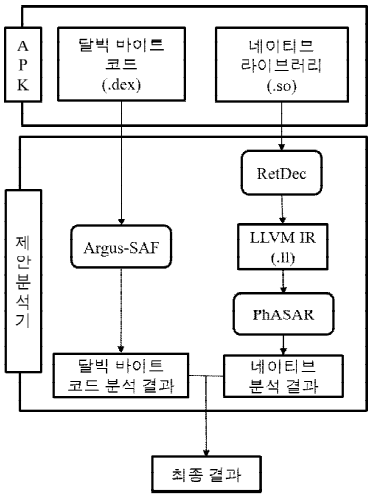

그림 1. 제안 분석기 구조

제안 분석기의 동작 방식은 먼저 입력으로 들어오는 애플리케이션에 대해 달빅 바이트 코드와 네이티브 라이브러리를 분류한다. 달빅 바이트 코드에 대해서는 기존의 Argus-SAF[1]를 사용하여 오염 분석을 진행한다. 추출한 네이티브 라이브러리를 RetDec을 이용하여 LLVM 중간표현으로 변환하고 수정한 PhASAR[2]을 이용하여 오염 분석을 진행한다. 마지막으로 Argus-SAF를 이용하여 달빅 바이트 코드에 대한 결과와 네이티브 분석 모듈을 이용하여 나온 네이티브 분석 결과를 통합하여 결과를 출력한다.

Ⅲ. 평가

실험은 총 9개의 안드로이드 애플리케이션으로 진행하였고 실험 결과 표 1과 같이 나타났다. Flowdroid의 경우 네이티브 라이브러리를 통해 개인 정보 유출이 발생하는 경우 탐지하지 못한 반면 Argus-SAF는 네이티브 라이브러리를 통해 개인 정보 유출이 발생하는 앱은 탐지할 수 있었지만 최신의 NDK 버전을 이용하여 빌드 된 애플리케이션을 탐지하지 못하였다. 하지만 제안 분석기는 네이티브 라이브러리나 NDK 버전에 상관없이 모든 애플리케이션의 개인 정보 유출을 탐지하였다.

표 1. 실험 결과

	Flowdroid	Argus-SAF	제안 분석기
nondk_mal_debug	O	O	O
nondk_mal_release	O	O	O
nondk_mal_version	O	O	O
ndk_mal_debug	X	X	O
ndk_mal_release	X	X	O
ndk_mal_version	X	O	O
native_complexdata-debug	X	X	O
native_complexdata-release	X	X	O
native_complexdata-version	X	O	O
총합	3/9	5/9	9/9

Ⅳ. 결론 및 향후 연구 방향

이 논문에서는 네이티브 라이브러리에 숨겨진 악성 코드를 검출하기 위한 오염 분석기를 설계하고 구현하였다. 구현한 오염 분석기는 Argus-SAF를 바탕으로 하였는데, 그 이유는 오픈 소스로 제공되며 안드로이드 애플리케이션에 대한 분석을 제공하기 때문이다.

분석기의 성능 평가를 위해 9개의 애플리케이션으로 실험을 하였고 실험 결과 NDK 버전에 상관없이 모든 애플리케이션의 개인 정보 유출을 탐지하였다. 따라서 제안 분석기를 이용하면 네이티브 악성 애플리케이션 탐지에 효과적임을 알 수 있다.

앞선 분석에서 Argus-SAF가 달빅 바이트 코드 분석에서 값 요약 문제와 리스너 내부에서 유출 함수 호출 시 검출이 안 되는 문제가 있었다. 이를 이용하여 오염 분석을 회피할 수 있어서 향후 연구로 이 문제를 해결하고자 한다.

ACKNOWLEDGMENT

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (2014-3-00035,매니코어 기반 초고성능 스케일러블 OS 기초연구 (차세대 OS 기초연구센터)).

*교신 저자 : 우균(부산대학교, woogyun@pusan.ac.kr).

참고문헌

[1] Fengguo Wei, Xingwei Lin, Xinming Ou, Ting Chen, and Xiaosong Zhang. JN-SAF: Precise and Efficient NDK/JNI-aware Inter-language Static Analysis Framework for Security Vetting of

Android Applications with Native Code, Conference on Computer and Communications Security, 2018

[2] Schubert, Philipp Dominik and Hermann, Ben and Bodden, Eric, PhASAR: An Inter-procedural Static Analysis Framework for C/C++, Springer, 2019