ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

안드로이드 모바일 단말에서의 실시간 이벤트 유사도 기반 트로이 목마 형태의 악성 앱 판별 메커니즘☆

Malicious Trojan Horse Application Discrimination Mechanism using Realtime Event Similarity on Android Mobile Devices

함 유 정[1] 이 형 우[1*]
You Joung Ham Hyung-Woo Lee

요 약

안드로이드 기반 모바일 단말 사용자가 증가함에 따라 다양한 형태의 어플리케이션이 개발되어 안드로이드 마켓에 배포되고 있다. 하지만 오픈 마켓 또는 3rd party 마켓을 통해 악성 어플리케이션이 제작 및 배포되면서 안드로이드 기반 모바일 단말에 대한 보안 취약성 문제가 발생하고 있다. 대부분의 악성 어플리케이션 내에는 트로이 목마(Trojan Horse) 형태의 악성코드가 삽입되어 있어 모바일 단말 사용자 모르게 단말내 개인정보와 금융정보 등이 외부 서버로 유출된다는 문제점이 있다. 따라서 급격히 증가하고 있는 악성 모바일 어플리케이션에 의한 피해를 최소화하기 위해서는 능동적인 대응 메커니즘 개발이 필요하다. 이에 본 논문에서는 기존 악성 앱 탐지 기법의 장단점을 분석하고 안드로이드 모바일 단말내에서 실시간 이용시 발생하는 이벤트를 수집한 후 Jaccard 유사도를 중심으로 악성 어플리케이션을 판별하는 메커니즘을 제시하고 이를 기반으로 임의의 모바일 악성 앱에 대한 판별 결과를 제시하였다.

☞ 주제어 : 안드로이드, 실시간 이벤트 수집, 이벤트 유사도, 트로이 목마 악성 앱, 판별 기법

ABSTRACT

Large number of Android mobile application has been developed and deployed through the Android open market by increasing android-based smart work device users recently. But, it has been discovered security vulnerabilities on malicious applications that are developed and deployed through the open market or 3rd party market. There are issues to leak user's personal and financial information in mobile devices to external server without the user's knowledge in most of malicious application inserted Trojan Horse forms of malicious code. Therefore, in order to minimize the damage caused by malignant constantly increasing malicious application, it is required a proactive detection mechanism development. In this paper, we analyzed the existing techniques' Pros and Cons to detect a malicious application and proposed discrimination and detection result using malicious application discrimination mechanism based on Jaccard similarity after collecting events occur in real-time execution on android-mobile devices.

☞ Keywords : Android, Real-time System Call, Event Similarity, Malicious Trojan Horse Application, Discrimination Mechanism.

1. 서 론

최근 모바일 단말 사용자가 급증하고 있으며 특히 안드로이드 플랫폼(Android platform)을 탑재한 상용 모바일 단말이 널리 보급되고 있다. 하지만 안드로이드 플랫폼의 보안 취약성이 계속적으로 발견되면서 악성 어플리케이션을 통한 공격 및 취약점이 급증하고 있다. 상용 모바일 단말을 대상으로 악의적인 공격자는 모바일 어플리케이션 내에 악성코드를 삽입한 후 이를 오픈 마켓 또는 인터넷을 통해 정상 어플리케이션인 것처럼 위장해 일반 사용자에게 배포하게 된다. 만일 사용자가 이를 자신의 단말에 설치한 후에 실행하게 되면 상용 모바일 단말 내에 저장된 SMS 송수신 정보, 전화번호부, 인터넷 접속 기록 등의 개인정보 뿐만 아니라 모바일 뱅킹 등에서 사용되는 모바일 공인인증서 등이 외부 서버로 유출되는 문제점이 발견되고 있다. 이와 같은 모바일 단말 보안위협에 능동적으로 대응하기 위해서는 악성 어플리케이션을 실

[1] School of Computer Engineering, Hanshin University, Gyeonggi, 447-791, Rep. of Korea.
* Corresponding author (hwlee@hs.ac.kr)
[Received 07 February 2014, Reviewed 22 February 2014, Accepted 19 March 2014]
☆ 본 연구는 2013년도 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2012R1A1A2004573)

시간 및 능동적 방법으로 탐지할 수 있는 메커니즘이 필요하다.

보다 구체적으로 살펴보면 안드로이드 기반 모바일 단말을 대상으로 단말 보안 취약성을 대상으로 한 악성 앱이 지속적으로 개발되고 있다. 만일 악성코드가 모바일 단말에 설치될 경우 단말 내 사용자 개인정보를 유출시키는 등의 문제점이 발생하고 있다[1]. 또한 정상적인 모바일 단말이 악성코드에 감염될 경우 공격자의 원격 명령에 의해 사용자의 단말이 좀비 스마트폰으로 제어될 경우 스마트폰을 통한 DDoS 공격 가능성도 증가하고 있다. 또한 악성코드가 삽입된 악성 앱에 의해 내부 리소스에 대한 권한 수정, 내부 정보에 대한 접근권한 불법 수집 및 내부 금융 정보 유출, 내부 저장 정보에 대한 불법적 외부 유출 등의 문제가 발생할 수 있다. 이와 같이 최근 급증하는 안드로이드 기반 모바일 단말 내 보안 취약성과 악성 어플리케이션의 공격에 능동적으로 대응하기 위한 기존 연구로 불법복제 앱 탐지 기법[2], 기계학습 기법을 통한 악성 앱 탐지 기법[3], 리패키징 여부를 탐지하는 기법[4], 유사 클래스 정보를 이용하여 악성 앱을 탐지하는 기법[5] 등의 악성 앱을 탐지하는 메커니즘이 제시되었다. 하지만 기존 연구인 경우 모바일 단말에 대한 실시간 이용 과정에서 발생하는 다양한 형태의 이벤트에 대한 특성 등을 이용하지 않은 방법이었다.

이에 본 연구에서는 시스템 이벤트 수집 도구를 이용하여 정상 어플리케이션과 악성 어플리케이션 실행시 발생하는 이벤트 특성을 분석하였다. 먼저, 정상 어플리케이션에 대한 이벤트 특성 도출을 위해 안드로이드 공식 마켓인 구글 플레이 스토어에서 수집된 200개의 정상 어플리케이션을 대상으로 실시간 이벤트를 추출하여 정상 어플리케이션 이벤트 집합을 구축하였고, Android MalGenome Project[6,7]에서 배포한 1,260개의 악성 앱 샘플 중에서 200개의 악성 어플리케이션을 대상으로 실시간 이벤트 집합을 구축한 후 정상 어플리케이션 이벤트 집합과의 비교 분석 및 특성 추출 과정을 수행하였다. 악성 어플리케이션에 대한 판별 정확도를 높이기 위해 Jaccard Coefficient 기법[8]을 응용한 실시간 이벤트 유사도 분석 알고리즘을 이용하여 모바일 단말을 대상으로한 트로이 목마(Trojan Horse) 형태의 악성 앱 판별 메커니즘 및 실험 결과를 제시하였다. 본 연구에서 제시한 이벤트 유사도 기반의 악성 앱 판별 알고리즘을 사용할 경우 기존에 연구했던 이벤트 패턴을 통한 악성 앱 모니터링 판별 기법[9,10]과 이벤트 기반 악성 앱 탐지 기법[11]보다 실시간으로 임의의 앱을 대상으로 한 정상 또는 악성 앱 판별 기능을 제공할 수 있었다.

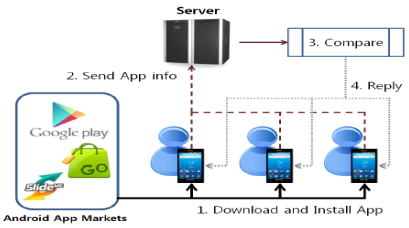
본 논문의 구성은 다음과 같다. 2장에서 기존 안드로이드 기반 악성 앱 판별 기법들에 대해서 분석하고 3장에서는 유사도 판별을 위해 strace 도구를 활용하여 정상과 악성 어플리케이션의 이벤트 Set을 구축한 결과에 대해 제시한다. 4장에서는 이벤트 유사도 기반의 악성 앱 판별 구조도와 알고리즘을 제시하고 임의의 앱을 대상으로 한 실험 결과를 통해 악성 앱 판별 결과를 제시한다. 마지막으로 5장에서는 본 연구의 결론을 제시하였다.

2. 기존 악성 앱 판별 기법

기존에 제시된 악성 앱 판별 및 탐지 메카니즘으로는 리패키징된 앱(repackaged application) 여부를 탐지하는 기법[4], 유사 클래스 정보를 이용하여 탐지하는 기법[5]이 있으며 위 기법들에 대한 주요 특성 및 문제점을 분석하면 다음과 같다.

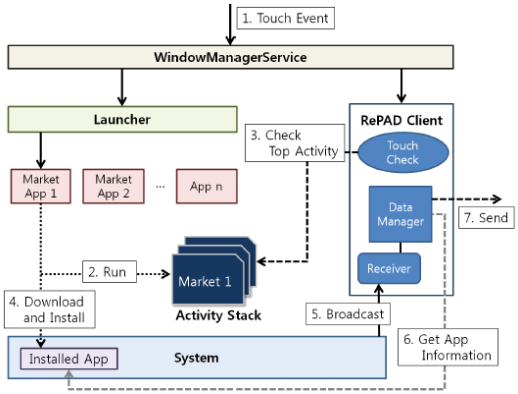
2.1 리패키징 앱 탐지 기법[4]

리패키징 앱 탐지(RePAD) 기법에서는 사용자가 설치한 모바일 앱의 내부 정보를 이용하여 리패키징 앱을 탐지하는 방법이다. RePAD 메커니즘을 이용하면 (그림 1)과 같이 사용자가 앱을 설치할 때 클라이언트 모바일 단말 내에 설치된 해당 앱에 관한 정보를 원격 서버로 전송하고, 서버는 수신된 앱 관련 정보를 이용하여 리패키징 여부를 판별하는 과정을 수행한다.

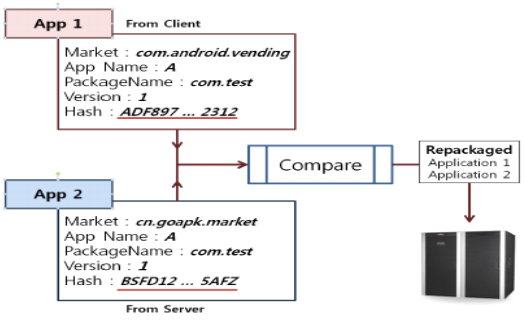


(그림 1) RePAD 시스템 구조
(Figure 1) RePAD System Structure

사용자가 오픈 마켓에서 앱을 다운받아 설치하면 클라이언트 단말에서는 사용자가 설치한 앱의 이름, 패키지 이름, 버전, 해시, 마켓 이름 등의 정보를 서버로 전송한다. 서버는 클라이언트로부터 전달 받은 정보를 저장하고, 이전에 저장된 앱의 정보와 함께 비교하여 리패키징 여부를 판단한 후 클라이언트로 분석 결과 메시지를 전송한다. 만일 패키지 이름은 같지만 마켓 출처 및 실행 파



(그림 2) RePAD 클라이언트의 구조와 동작
(Figure 2) RePAD Structure and Action



(그림 3) 리패키징 판별 예시
(Figure 3) Repacking Discrimination example

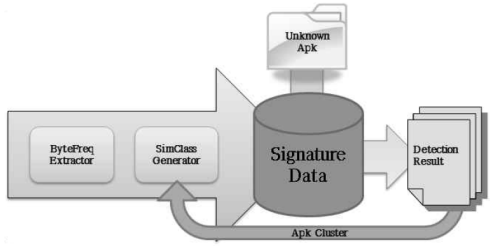
일의 시그니처(Signature) 또는 해시 값(Hash value)이 같지 않으면 해당 앱의 내부가 수정되었다고 가정하여 리패키징된 앱이라고 판단하게 된다.

RePAD를 구성하는 클라이언트의 동작 과정을 살펴보면 다음 (그림 2)와 같다. 오픈 마켓으로부터 임의의 앱을 다운로드하고 설치하게 되면, RePAD 클라이언트에 포함된 DataManager 모듈을 통해 설치된 앱의 정보를 추출한 후 앱의 이름 등과 앱 내부 정보 등을 서버로 전송하게 된다.

서버는 클라이언트에서 앱의 정보를 DB에 저장한 후 (그림 3)과 같이 해시 함수 등을 이용하여 모바일 앱의 리패키징 여부를 판별하는 역할을 수행한다. 리패키징 여부를 판별할 때 앱 이름 외에 동일한 버전을 가진 앱들을 서로 비교하는 과정을 수행한다. 또한 서버는 리패키징 앱이 악성 앱인지를 탐지하기 위해 리패키징 앱과 원본 앱에 대해 앱 내에 포함된 AndroidManifest.xml 파일로부터 접근 권한 정보를 분석하고, 보안 정보 유출 또는 과금을 유발할 때 사용되는 접근 권한이 리패키징 앱에 추가로 포함되었는지 여부를 확인한다.

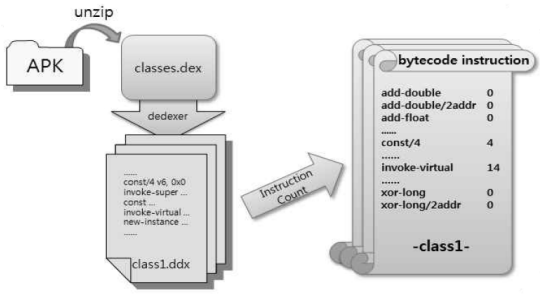
2.2 유사 클래스 정보 기반 악성 앱 탐지 기법[5]

유사 클래스 정보 기반 악성 앱 탐지 기법에서는 트로이 목마(Trojan) 형태로 구동하는 악성 어플리케이션들을 탐지하는 기능을 제공한다. 대부분의 안드로이드 어플리케이션은 자바 언어를 이용하여 개발되었기 때문에 (그림 4)와 같이 앱에 포함된 클래스를 대상으로 악성 어플리케이션을 판별하는 과정을 수행한다.



(그림 4) 안드로이드 악성 그룹 탐지 시스템
(Figure 4) Android Malicious Group Detection System

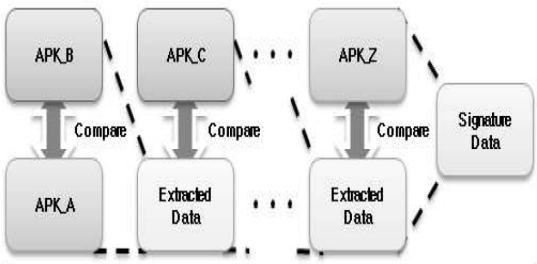
악성 클래스에 대한 시그니처 추출을 위해 어플리케이션을 역어셈블(reassembling)한 후에 클래스 단위로 Dalvik 바이트 코드 명령어에 대한 빈도수를 추출한다. 클래스 내부에 포함된 명령어 빈도수에 대한 비교 분석 과정을 통해서 유사한 클래스 정보들을 추출하여 악성 시그니처 집합을 구축한 후 악성 여부를 판별하는 과정을 수행하며 (그림 5)와 같이 악성 어플리케이션 탐지를 위해 ByteFreq Extractor, SimClass Generator, 악성 어플리케이션 탐지 모듈 및 Apk Cluster 모듈로 구성되어 있다.



(그림 5) ByteFreq Extractor 시스템 구조
(Figure 5) ByteFreq Extractor System Strucuture

APK 파일을 압축 해제하여 classes.dex 파일을 획득한 후 dedexer 프로그램을 통해서 역어셈블 과정을 수행하고 앱 내에 포함된 모든 클래스를 DDX 파일로 변환하여 저장한다. DDX 파일에는 해당 클래스에 대한 패키지 정보, 상속 정보, 메소드 정보, 그리고 Dalvik 바이트코드로 이루어진 명령어 정보 등이 존재하기 때문에 바이트코드 명령어들만 추출해서 어플리케이션에 대한 명령어 사용 빈도 정보를 저장하게 된다.

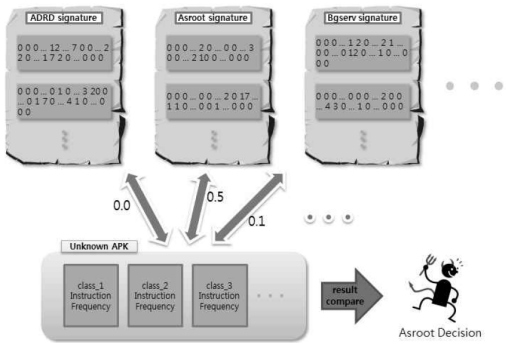
SimClass Generator는 악성 그룹을 탐지하는데 사용되는 시그니처 생성과정을 수행한다. SimClass Generator의 시스템 구조는 다음 (그림 6)과 같다. 시그니처 집합을 구축하기 위해 그룹에 존재하는 어플리케이션들을 대상으로 1:1 비교를 순차적으로 진행하여 각 어플리케이션에 공통적으로 포함된 클래스 정보를 최종 시그니처로 정의한다. 이 과정에서 2단계의 비교 기법이 진행되는데 먼저 정렬된 명령어 정보들에 대해 최상위 20개의 명령어를 해당 클래스 집합으로 저장하고, 저장된 집합을 서로 비교하여 모든 원소가 일치하는 집합들의 클래스에 대해 정렬된 명령어에 대한 유사도 비교 과정을 수행한다.



(그림 6) SimClass Generator 시스템 구조
(Figure 6) SimClass Generator System Strucuture

악성 어플리케이션 탐지를 위해 구축된 시그니처 집합을 이용하여 (그림 7)과 같이 임의의 어플리케이션에 대한 악성 여부를 판별하는 과정을 수행한다. 어플리케이션의 악성 여부를 판단하기 위해 ByteFreq Extractor를 통해 클래스 기반 명령어 빈도수를 추출한다.

이와 같은 전처리 과정 수행 후 SimClass Generator에서 사용한 비교 기법을 통해 시그니처의 클래스와 유사하게 판단되는 정보가 애플리케이션에 포함될 경우 그 개수를 모두 누적하여 시그니처에 존재하는 클래스 개수와의 비율을 계산하여 어플리케이션과 그룹간의 유사도 값으로 결정하는 방법을 사용하였다.



(그림 7) 어플리케이션 악성 여부 판단 과정
(Figure 7) Malicious decision process about application

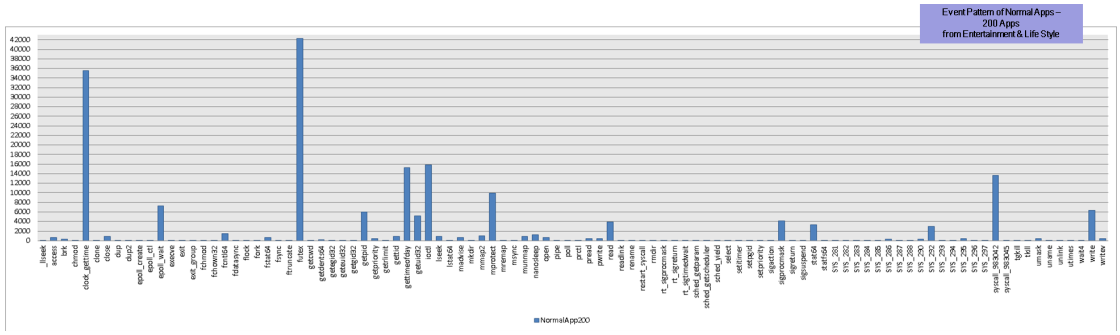
2.3 기존 기법에 대한 고찰 및 개선 방안

앞에서 제시한 두 가지 기법 등에 대한 장단점 분석 과정을 통해 효율적인 악성 앱 판별 메커니즘을 제시하고자 한다.

기존 기법에 대한 장단점을 분석하면 리패키징 앱 탐지 기법인 경우 스마트폰의 CPU와 메모리를 적게 사용하면서 사용자가 설치한 앱의 출처를 파악할 수 있고, 사용자가 설치하는 앱에 대해서만 악성 여부를 판별할 수 있다는 장점이 있다. 또한 다양한 마켓에 존재하는 리패키징 앱을 수집할 때 요구되는 인력과 시간을 절약할 수 있다는 장점이 있다. 하지만, 리패키징 앱에 특화된 기법이기 때문에 다른 형태의 악성 앱을 탐지하는 과정에서 탐지율이 떨어진다는 단점이 존재한다.

유사 클래스 정보 기반 악성 앱 탐지 기법인 경우 실제 악성 앱을 대상으로 악성 앱 내에 존재하는 유사 클래스 기반의 명령어 빈도수를 추출하여 정상 앱과 악성 그룹 사이의 유사도를 측정하고 이를 기반으로 악성 앱을 탐지하는 방법을 사용하였기 때문에 악성 앱 판별에 대한 신뢰성과 정확성이 높다는 장점이 있다. 하지만 실시간 탐지가 어려우며 사전 분석 및 시그니처 집합 구축에 많은 비용과 시간이 든다는 단점이 있고, 실험 결과에서 일부 낮은 비율의 탐지 성능을 나타내는 샘플들에 대한 보다 개선된 판별 방법이 필요하다는 단점이다.

이에 대한 개선 방안으로는 웹 페이지 문서에 포함된 단어에 대한 유사도를 측정하는데 널리 사용되는 Jaccard Coefficient 기법[8]을 응용하여 실시간으로 구동되는 모바일 앱 실행시 발생하는 이벤트에 대한 유사도 분석 메커니즘을 토대로 트로이 목마 형태의 악성 앱을 판별하는 메커니즘을 제시하고자 한다. 이를 위해서는 우선 안드로



(그림 8) 200개의 정상 어플리케이션 이벤트 Set
(Figure 8) Two hundreds of Normal Application Event Set

이드 기반 모바일 단말을 사용하는 과정에서 발생하는 각종 이벤트 정보를 수집하여야 한다. 그리고 수집된 정상 및 악성 이벤트 집합의 특성 정보를 토대로 판별하고자 하는 임의의 모바일 앱에 대한 판별 과정을 수행할 수 있다. 이에 본 논문에서는 안드로이드 기반 모바일 단말 내에서 발생하는 이벤트 정보를 수집하여 이벤트 유사도를 기반으로 악성 앱을 판별할 수 있는 메카니즘을 설계 및 구현하였다.

3. 유사도 판별을 위한 실시간 이벤트 집합 구축

본 논문에서는 기존에 연구했던 이벤트 기반의 악성 앱 모니터링 판별 기법[11]에서 더 나아가 안드로이드 기반 모바일 단말 내 유사도 기반의 악성 앱 탐지 판별을 위해 각각 정상 앱 중 200개, 악성 앱 200개를 대상으로 정상 이벤트 집합과 악성 이벤트 집합을 구축하였다. 200개의 정상 앱은 안드로이드 공식 마켓인 구글 플레이 스토어 내 엔터테인먼트와 라이프 스타일 카테고리에 포함된 앱을 설치한 후 실시간 이벤트를 수집하였고, 200개의 악성 앱은 Android MalGenome Project[6,7]에서 제공한 1,260개의 악성코드 샘플들 중 200개의 앱 파일을 설치해서 실시간 이벤트를 수집하였다.

3.1 정상 어플리케이션 이벤트 집합

본 연구에서 수집한 200개의 정상 앱은 안드로이드 공식 마켓인 구글 플레이 스토어 내 엔터테인먼트와 라이프 스타일 카테고리 중에 선별하여 해당 정상 앱을 설치한 후 strace 도구를 통해 실시간 이벤트를 수집하였다. (그림 8)은 200개의 정상 어플리케이션의 이벤트들을 수집한 정상 어플리케이션 이벤트 Set을 나타내고 있다.

(그림 8)을 보면 200개의 정상 어플리케이션 이벤트 Set의 수는 총 101개로 clock_gettime, futex, gettimeofday, ioctl, syscall_983042 등의 이벤트가 상대적으로 많이 발생하는 것을 확인할 수 있었다.

3.2 악성 어플리케이션 이벤트 집합

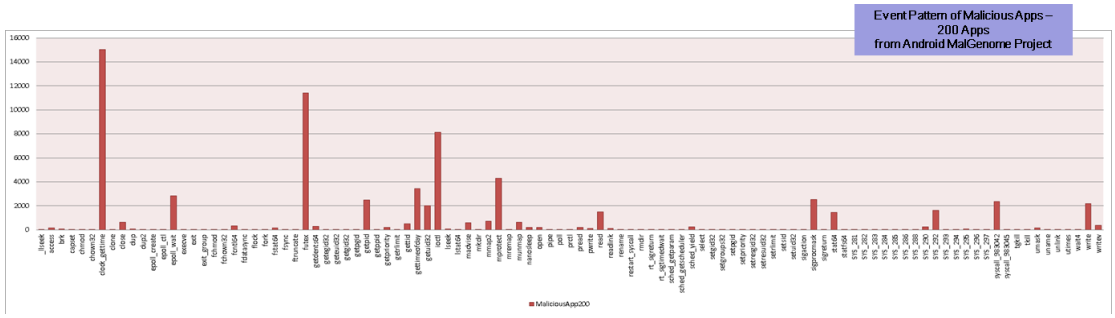
Android MalGenome Projcet[6,7]에 포함된 1,260개의 악성 앱 중 악성코드의 특성을 반영하여 200개의 악성 앱을 선정한 후 이를 대상으로 안드로이드 모바일 단말에서 발생하는 실시간 이벤트를 추출 및 수집하였다.

(그림 9)와 같이 200개의 악성 어플리케이션 이벤트 Set의 수는 총 105개로 정상 앱 실행시 수집된 이벤트 집합과 일부 공통적으로 발생하는 이벤트가 있으면서도 악성 앱에서 상대적으로 많이 발생하는 이벤트가 있다는 것을 확인할 수 있었다.

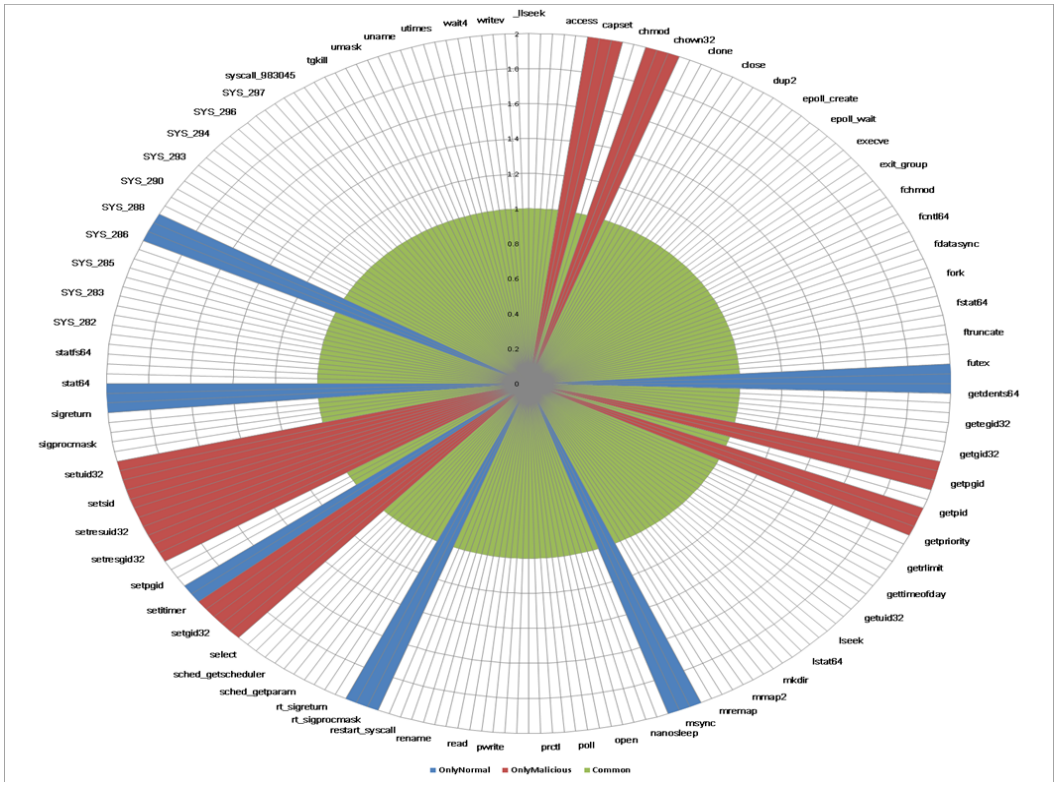
(그림 8)과 (그림 9)를 비교해봤을 때 각 정상과 악성 어플리케이션 이벤트 Set내 이벤트들은 크게 세 가지로 분류할 수 있다. 우선, 정상 어플리케이션 실행 시에만 발생하는 이벤트, 악성 어플리케이션 실행 시에만 발생하는 이벤트 및 정상과 악성 어플리케이션에 모두 존재하는 이벤트 집합으로 나눌 수 있으며 이를 대상으로 각 집합들에 대한 특성 및 유사도 분석 과정을 수행하였다.

3.3 정상 및 악성 앱 이벤트 Set 비교 분석

각 200개의 정상과 악성 어플리케이션 이벤트 Set을 비교분석하면 다음 (그림 10)과 같다. 115개의 이벤트들은 크게 (그림 10)에서 정상 어플리케이션 이벤트에서만



(그림 9) 200개의 악성 어플리케이션 이벤트 Set
(Figure 9) Two hundreds of Malicious Application Event Set



(그림 10) 200개의 정상 및 악성 어플리케이션 이벤트 집합 비교 분석
(Figure 10) Two hundreds of Normal and Malicious Application Event Set Comparison Analysis

발생하는 'OnlyNormal' 이벤트 집합과 악성 어플리케이션 이벤트에서만 발생하는 'OnlyMalicious' 이벤트 집합, 그리고 정상 및 악성 어플리케이션에서 모두 발생하는 'Common' 이벤트 집합으로 나눌 수 있었다.

OnlyNormal에 포함되는 이벤트의 수는 총 6개로 getcwd, msync, rt_sigprocmask, setitimer, sigsuspend, SYS_287의 이벤트가 존재한다. 반면 OnlyMalicious에 포함되는 이벤트의 수는 총 11개로 capset, chown32, getpgid, getppid, setgid32, setgroups32, setresgid32, setresuid32, setrlimit, setsid, setuid32의 이벤트가 존재하는 것을 확인할 수 있었

다. 이들 이벤트들은 정상 내지 악성 어플리케이션에만 존재하는 이벤트들로써 정상과 악성 어플리케이션을 구별할 수 있는 하나의 기준이 될 수 있다.

마지막으로 Common에 포함되는 이벤트 집합은 위의 17가지 이벤트들을 제외한 이벤트 들의 집합으로 구성되며 _llseek, access, clock_gettime, epoll_create, fchmod, fchown32, fcntl64, fdatasync, flock, fork 및 fstat64 등을 포함하여 최종적으로 총 98개의 이벤트로 구성되었다.

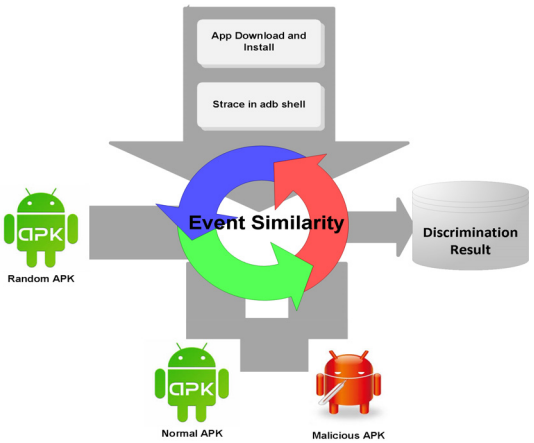
4. 이벤트 유사도 기반 악성 앱 판별

앞에서 분석한 결과를 토대로 본 장에서는 이벤트 유사도 기반의 악성 어플리케이션을 판별하기 위한 시스템 구조를 제시하였다. 그리고 이와 함께 안드로이드 모바일 단말용 앱에 적합한 실시간 유사도 측정 알고리즘을 제시하였다. 이를 기반으로 임의의 정상 및 트로이 목마(Trojan Horse) 형태의 악성 어플리케이션 10개를 대상으로 본 연구에서 제안한 알고리즘을 이용하여 정상 및 악성 어플리케이션에 대한 실시간 이벤트 유사도 측정 결과를 제시하고 악성 앱 판별 실험 결과를 제시하였다.

4.1 이벤트 유사도 기반 악성 앱 판별 시스템 구조

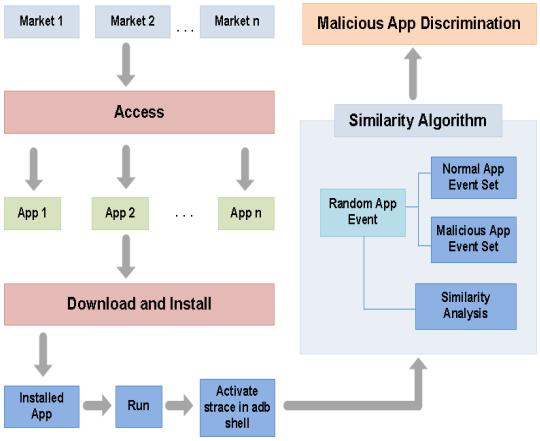
4.1.1 악성 앱 판별 시스템 구조

악성 어플리케이션에 대한 사용자의 피해를 최소화하기 위해 본 논문에서 제시하는 악성 앱 판별 시스템 구조는 다음 (그림 11)과 같다.



(그림 11) 악성 어플리케이션 판별 시스템 구조
(Figure 11) Malicious Application Discrimination Structure

본 논문에서 구현한 악성 어플리케이션 판별 시스템은 다음 순서와 같이 진행된다. 오픈 마켓으로부터 분석 대상인 어플리케이션을 다운로드 및 설치한 다음 adb shell를 통해 strace 모듈을 통해 어플리케이션 실행 시 모바일 단말에서 발생하는 실시간 이벤트를 수집한다. 이제 단말로부터 수집된 정상 및 악성 어플리케이션의 이벤트 집합을 기반으로 임의의 어플리케이션 실행시 발생하는 실시간 이벤트를 대상으로 정상 및 악성 이벤트와의 유사도 분석 과정을 통해 어플리케이션에 대한 악성 여부를 판별하는 것으로 세부 과정은 아래 (그림 12)와 같다.

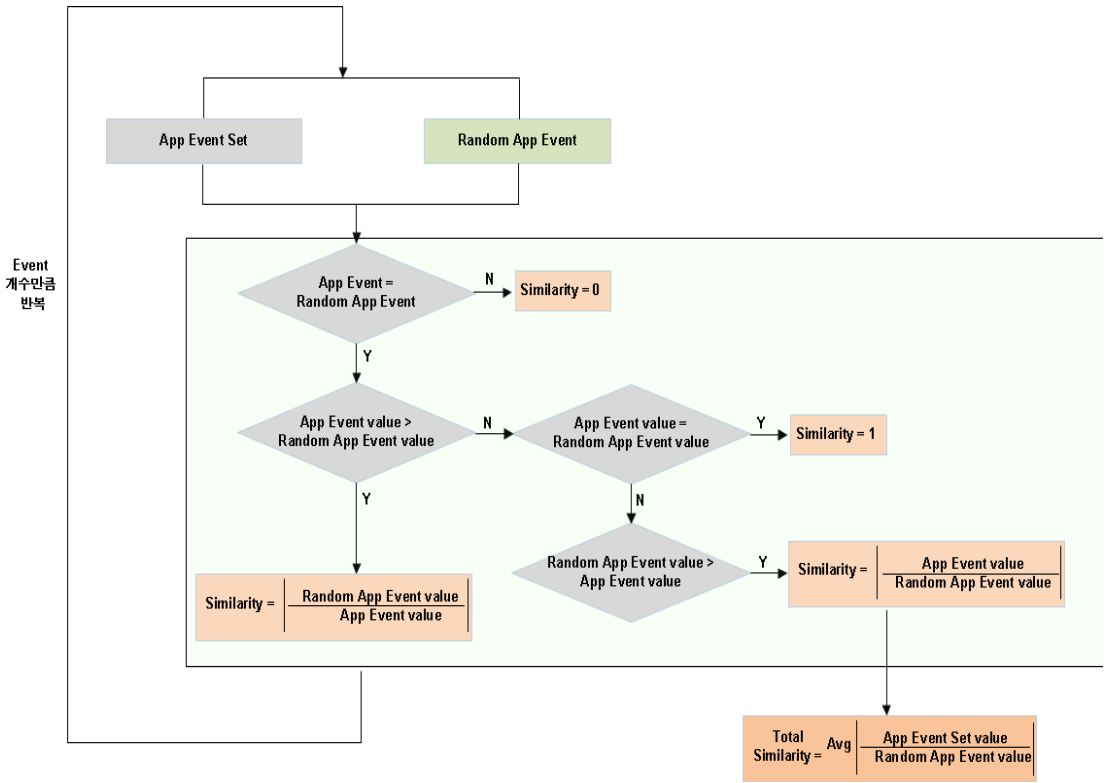


(그림 12) 악성 어플리케이션 판별 과정
(Figure 12) Malicious Apps Discrimination Processes

4.1.2 유사도 기반 악성 앱 판별 알고리즘

(그림 12)에 나타나있는 악성 어플리케이션을 판별하기 위한 유사도 알고리즘을 좀 더 자세하게 나타내면 (그림 13)과 같다. 본 논문에서 제안하는 유사도 알고리즘은 Jaccard Coefficient 기법[8]을 기반으로 안드로이드 모바일 단말에 적합하도록 변형하였으며 단계별 수행 과정은 다음과 같다.

① 정상 및 악성 어플리케이션 실행 결과 발생한 이벤트에 대한 평균 발생 빈도 값을 계산한다.
② 어플리케이션 내에 존재하는 이벤트가 임의의 어플리케이션 내 이벤트에도 동일하게 존재하는지를 비교한다.
③ 만약 임의의 어플리케이션 실행시 발생한 이벤트가 정상 및 악성 어플리케이션 이벤트 집합 내에 존재하지 않는다면 이벤트의 유사도 값은 0이 된다.



(그림 13) 악성 앱 판별을 위한 이벤트 유사도 알고리즘
(Figure 13) Event Similarity Algorithm for Malicious Application Discrimination

④ 반면 임의의 어플리케이션 이벤트가 정상 또는 악성 어플리케이션 이벤트 집합 내에 존재한다면 해당 이벤트의 발생 빈도 값의 크기를 비교한다.

⑤ 정상 또는 악성 어플리케이션 실행시 이벤트 발생 빈도값 EV_i^{App}과 임의의 어플리케이션 실행시 이벤트 발생 빈도값 EV_j^{AppRdm}에 대해서 $\frac{\min(EV_i^{App}, EV_j^{AppRdm})}{\max(EV_i^{App}, EV_j^{AppRdm})}$ 값을 계산한다.

⑥ ①부터 ⑤까지의 과정을 임의의 어플리케이션 실행시 발생한 이벤트에 대해 반복적으로 수행하여 계산하면 각 이벤트마다 유사도 값이 나타나게 되고 이를 바탕으로 다시 이벤트 전체에 대한 평균 유사도 값을 구한다.

⑦ 위와 같은 단계를 거쳐서 어플리케이션의 이벤트 Set과 임의의 어플리케이션 이벤트의 유사도를 나타내는 수식은 다음과 같다.

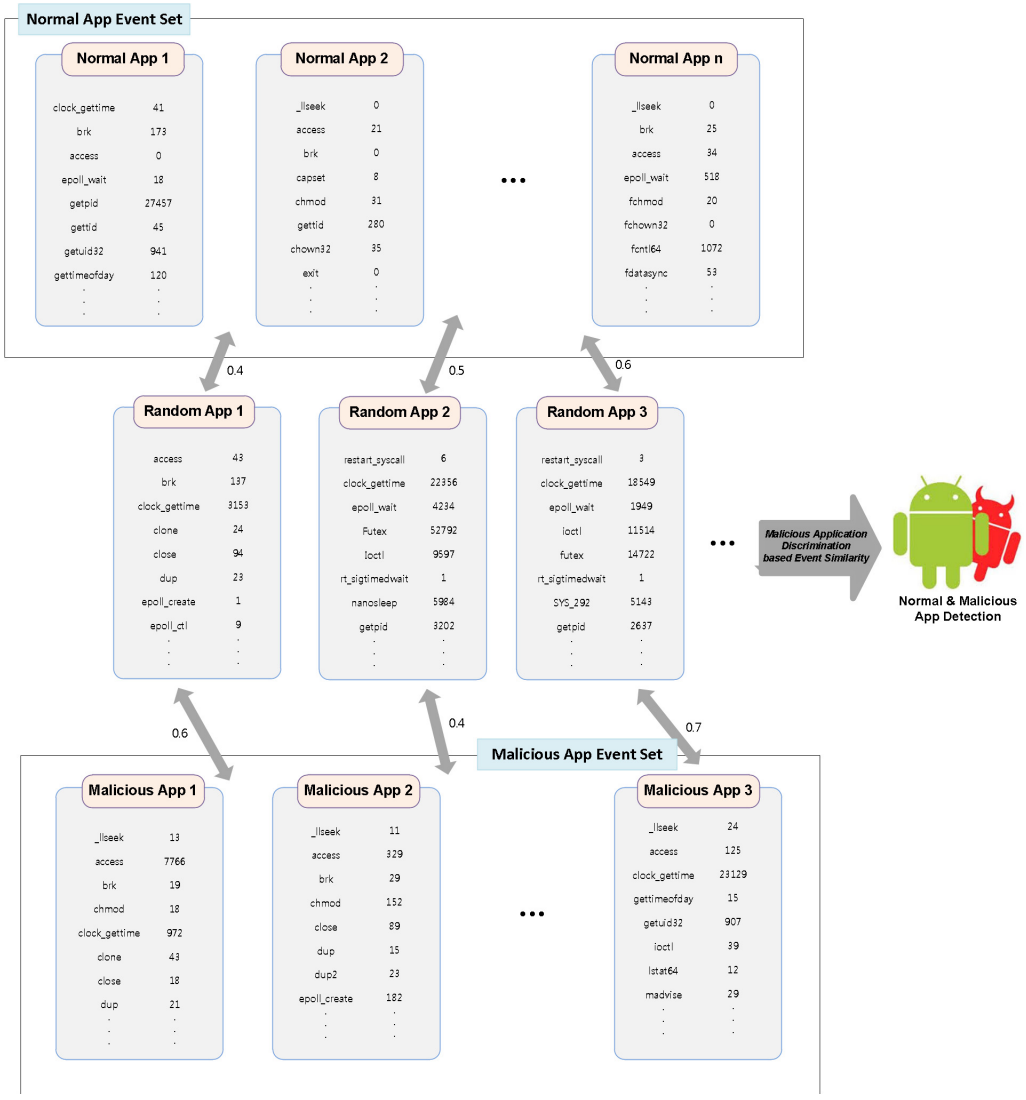
$$S = \frac{\left| \frac{\sum_{i=1}^{n} EV_i^{App}}{n} \right|}{EV_j^{AppRdm}}$$

(수식 1)
이벤트 유사도

(수식 1)에서 이벤트 유사도는 0에서 1사이의 값이 되며, EV_i^{App}는 어플리케이션에서 발생하는 i 번째 이벤트이고 EV_j^{AppRdm}는 판별 대상인 임의의 어플리케이션에서 발생한 이벤트 j 라고 정의할 경우 두 앱에서 발생하는 이벤트 유사도는 Jaccard Coefficient 기법을 응용하여 실험군에 속한 n 개의 앱에서 발생한 이벤트 EV_i^{App} 각각에 대한 발생빈도의 평균에 EV_j^{AppRdm}을 나눈 값으로 계산할 수 있다.

4.1.3 유사도 기반 악성 앱 판별 과정

정상 어플리케이션 이벤트 Set과 악성 어플리케이션 이벤트 Set에서 추출된 이벤트 유사도 특성에 기반하여

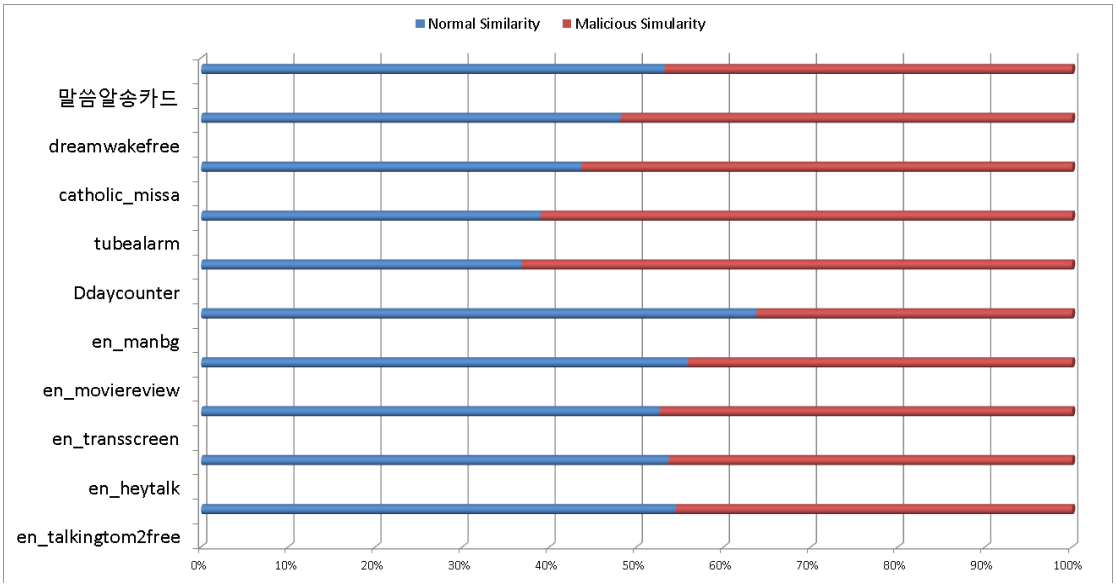


(그림 14) 이벤트 유사도 기반 악성 앱 판별 과정

(Figure 14) Malicious Application Discrimination Process based Event Similarity

악성 앱 판별 알고리즘을 적용할 경우 실험군에 포함되지 않은 임의의 어플리케이션 실행시 발생하는 이벤트 정보를 수집하여 최종적으로 정상/악성 앱을 판별하는 과정은 다음 (그림 14)와 같다.

우선 strace를 통해 추출한 정상 및 악성 어플리케이션 실행시 발생하는 두 이벤트 집합 내에 일치하는 n개의 정상 및 악성 어플리케이션 이벤트와 임의의 어플리케이션 이벤트의 값 사이의 유사도를 계산하여 임의의 어플리케이션에 대한 최종 유사도 값을 구하게 된다. 이때 정상 어플리케이션 이벤트 Set과 임의의 어플리케이션 이벤트의 유사도가 더 높게 나타나면 해당 임의의 어플리케이션은 정상 어플리케이션에 가깝다고 판별할 수 있고 악성 어플리케이션 이벤트 Set과 임의의 어플리케이션 이벤트의 유사도가 더 높게 나타나면 해당 임의의 어플리케이션은 악성 어플리케이션에 가깝다고 판별할 수 있다.



(그림 15) 이벤트 유사도 기반 10개의 임의의 정상 앱 실험 결과
(Figure 15) Random Unknown Application Decision Result based Event Similarity

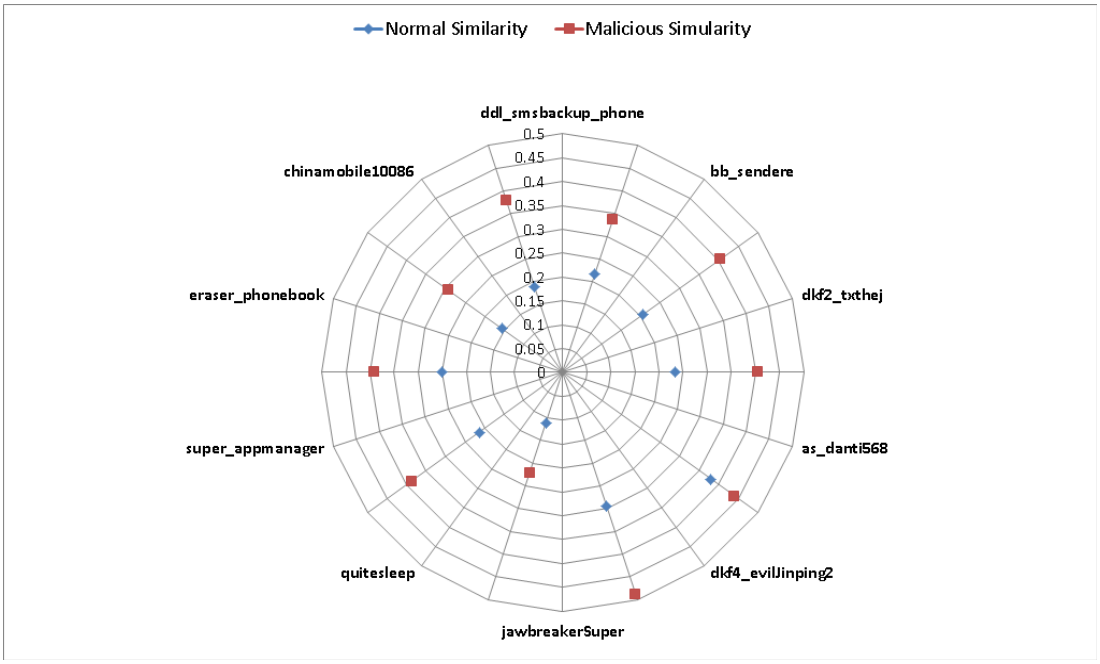
4.2 이벤트 유사도 기반 임의의 앱에 대한 악성 앱 판별 결과

이벤트 유사도를 기반으로 하여 악성 어플리케이션인지 아닌지를 판별하기 위해 본 논문에서는 각 10개의 임의의 정상 및 악성 어플리케이션을 실행하여 이벤트를 수집한 후 유사도 알고리즘을 바탕으로 각각 유사도를 도출하였다.

4.2.1 임의의 정상 앱에 대한 판별 실험

10개의 임의의 정상 어플리케이션의 이벤트와 각각의 정상 및 악성 어플리케이션 이벤트 Set을 대상으로 유사도를 비교해본 결과는 (그림 15)와 같다. 이때, 유사도는 0에서 1사이의 값으로 일반적으로 각각 정상 및 악성 어플리케이션 이벤트 Set과 비교해보았을 때 유사도가 0.5 이상이면 정상 및 악성 어플리케이션에 가깝다고 가정한다. 하지만 이는 상대적인 수치로 본 논문에서는 유사도 알고리즘을 기반으로 유사도를 계산한 결과가 임의의 정상 어플리케이션 이벤트와 정상 어플리케이션 이벤트 Set의 유사도가 악성 어플리케이션 이벤트 Set의 유사도보다 더 높을 때 해당 임의의 정상 앱이 정상 어플리케이션에 가깝다고 가정하였고 반대로 임의의 악성 어플리케이션 이벤트와 악성 어플리케이션 이벤트 Set의 유사도가 정상 어플리케이션 이벤트 Set의 유사도보다 더 높게 나타났을 때 해당 임의의 악성 앱이 악성 어플리케이션에 가깝다고 가정하였다.

(그림 15)를 보면 10개의 임의의 정상 어플리케이션 중 말씀알송카드, en_manbg, en_movieview, en_transscreen, en_heytalk, en_talkingtom2free 등 6개의 정상 어플리케이션이 정상 어플리케이션 이벤트 집합과의 유사도가 악성 이벤트 집합과의 유사도보다 높게 나타난 것을 확인할 수 있었다. 본 실험을 통해 기존 정상 형태의 어플리케이션을 대상으로 60%만이 정상 유사도가 높게 나온 이유를 분석해 보면, 정상으로 보이는 형태의 앱에도 악성과 유사한 형태의 의심되는 부분이 포함되어 있을 가능성이 높기 때문이다. 최근 언론 보도(전자신문 2014년 1월 28일 기사 : 미 NSA, 앵그리버드와 페이스북도 떨었다)를 통해 알려진 바와 같이 정상 앱으로 알려진 앵그리버드(Angry Birds) 모바일 앱에 악성 코드를 포함하여 정보 유출 앱(leaky Apps)으로 배포한 사실이 알려졌다. 따라서 정상 앱이라고 대부분 알고 있고 기존의 오픈-마켓을 통해서도 널리 배포되고 있는 앱 중에서 확인되지 않은 악성 코드를 포함하고 있는 정상 형태의 모바일 앱이 배포 및 설치되어 일반 사용자가 사용하고 있을 가능성이 매우 높다는 것을 확인할 수 있었다.



(그림 16) 이벤트 유사도 기반 10개의 임의의 악성 앱 실험 결과

(Figure 16) Ten of Random Malicious Application Experiment Result based Event Similarity

4.2.2 악성 앱에 대한 유사도 기반 탐지 및 판별 실험

트로이 목마 형태로 작동하는 악성 앱 중에서 실험군에 포함되지 않은 10개의 악성 어플리케이션을 대상으로 각각 정상 및 악성 어플리케이션 이벤트 Set과의 유사도를 측정한 결과는 (그림 16)과 같다.

(그림 16)을 보면 super_appmanager, bb_sendere, chinamobile10086, quitesleep, jawbreakSuper, dkf4_evilJinping2, as_danti568, ddl_smsbackup_phone 및 dkf2_txthej 등 총 10개의 악성 어플리케이션 모두가 실험군에 포함된 악성 어플리케이션 실행시 발생하는 이벤트 패턴과의 유사도가 높다는 것을 확인할 수 있다. 그 이유는 정상 어플리케이션 이벤트 Set에서는 존재하지 않고 악성 어플리케이션 이벤트 Set에만 존재하는 이벤트들 및 두 집합 모두에서 발견되는 이벤트들이 10개의 악성 어플리케이션 실행 과정에서도 유사하게 발생하기 때문이다. 그러므로 판별 과정에 사용된 10개의 악성 앱 이벤트와 악성 어플리케이션 이벤트 Set과의 유사도는 높게 나타났으며 결과적으로 본 논문에서 제시한 유사도 측정 방법을 이용할 경우 악성 어플리케이션을 모두 탐지 및 판별할 수 있었다.

4.3 기존 기법과의 비교 분석

이벤트 유사도를 기반으로 한 악성 앱 판별 결과를 바탕으로 2장에서 살펴본 악성 앱 판별과 관련된 기존 리패키징 앱 탐지 기법과 유사 클래스 기반 악성 앱 탐지 기법을 본 논문에서 제안하는 악성 앱 판별 기법과 비교 분석하면 다음 (표 1)과 같이 나타낼 수 있다.

(표 1) 기존 기법과의 비교분석

(Table 1) Comparative Analysis with existing method

구분	RePAD [4]	MFD [5]	제안기법
실험 대상 플랫폼	Android	Android	Android
어플리케이션 수집대상	n개의 단말	n개의 단말	n개의 단말
사용 도구 및 방법	Broadcast Receiver	dedexer	strace
악성 앱 판별시 사용하는 정보	앱 내부 정보 (해시/패키지 이름 등)	앱 내부에 포함된 클래스 정보	앱 실행시 발생하는 이벤트 정보
악성 앱 판별 알고리즘	-	DBSCAN Clustering	Jaccard Coefficient
악성 앱 판별 기능	△	○	○
정상 및 악성 앱 특징 분석	△	△	○

(표 1)에 제시한 바와 같이 세 가지 기법 모두 안드로이드를 플랫폼으로 하고 있으며 어플리케이션 수집 또한 다수의 상용 모바일 단말을 대상으로 하고 있다. 본 논문에서 제안하는 기법은 리눅스 기반의 이벤트 수집 도구인 strace를 기반으로 하여 어플리케이션 실행 시 발생하는 이벤트를 수집한다. 이러한 도구를 통해서 각각 악성 앱을 판별하기 위한 정보로 기본 RePAD 기법[4]은 어플리케이션 이름, 패키지 이름, 해시, 마켓 이름 등의 어플리케이션 정보를 획득하여 비교한 뒤 악성 앱을 판별한다. MFD 기법[5]은 dedexer로 추출해낸 DDX 파일의 클래스 정보를 활용하여 악성 앱을 판별하는데 반해 제안한 기법은 어플리케이션의 이벤트를 수집하여 이를 토대로 악성 앱을 판별하는 기능을 제공한다.

사용된 악성 앱 판별 알고리즘을 비교 분석해보면 우선 RePAD 기법은 사용된 악성 앱 판별 알고리즘이 없었음을 알 수 있었고 MFD 기법은 DBSCAN Clustering 기법을 통해 분류되지 않은 악성 앱을 다시 판별하는 것을 알 수 있었다. 하지만 본 논문에서 제시한 기법은 Jaccard Coefficient 기법[8]을 바탕으로 이벤트 간의 유사도를 측정해 악성 앱을 판별하도록 하였다.

악성 앱을 판별한 결과 및 성능을 비교해 보면 RePAD 기법은 악성 앱 중에서 Repackaging 형태의 악성 어플리케이션들을 분류해낼 수 있다는 장점이 있으나, 제안한 기법은 악성 앱의 형태 및 작동 방식과 상관없이 기존 악성 앱을 거의 100%에 가깝게 판별할 수 있다는 장점이 있다. 이는 본 논문에서 제안하는 기법이 악성 앱을 판별하기 위해 안드로이드 모바일 단말에서 실시간으로 발생하는 이벤트 정보를 중심으로 유사도를 측정하였기 때문이다.

5. 결 론

본 논문에서는 안드로이드 기반 모바일 단말을 대상으로 점점 증가하는 악성 앱에 대한 피해를 최소화하기 위해 다수 사용자 모바일 단말에 대한 이벤트 유사도 기반 악성 앱 판별 메카니즘을 제시하였다. 기존 악성 앱 탐지 기법을 분석하여 Jaccard Coefficient 기법을 이용한 이벤트 유사도 알고리즘을 제안하였으며, 정상 어플리케이션과 악성 어플리케이션를 실행한 후 발생하는 이벤트를 수집한 후 정상 어플리케이션 이벤트 Set과 악성 어플리케이션 이벤트 Set을 구축하였다. 뿐만 아니라 임의의 정상 및 악성 어플리케이션과의 유사도 실험 결과를 통해 악성 앱을 판별할 수 있는 기초적인 기능을 제공하였으며 이를 토대로 10개의 임의의 악성 앱 모두가 악성 어플리케이션에 가까운 결과를 도출할 수 있었다.

향후 추가적으로 다수의 사용자 단말을 대상으로 시스템 내부에서 어플리케이션 실행시 발생하는 커널 레벨의 이벤트 정보에 대한 연관성 분석을 수행하고 이를 통해 다수의 사용자 단말에서 수집된 이벤트 정보에 대한 그룹핑 방법과 이를 반영한 유사도 판별 메커니즘을 통해 오탐율을 줄일 수 있는 방법에 대한 연구가 필요하다.

참 고 문 헌(Reference)

[1] Hyung-Woo Lee, "Android based Mobile Device Rooting Attack Detection and Malicious Application Event Monitoring", *Review of Korean Society for Internet Information*, Vol. 13, No. 1, (2012), pp.30-38.

[2] Sungmin Kim, Eunhoe Kim, Jaeyoung Choi, "Illegally-copied App Detector on Android Platform", *Journal of Security Engineering,* Vol. 10, No. 1, (2013), pp.51-61.

[3] Seoungwook Min, Hyungjin Cho, Jinseop Shin, Jaecheol Ryou, "Android Malware Analysis and Detection Method Using Machine Learning", *Journal of the Korean Institute of Information Scientists and Engineers: Computing Practices and Letters*, Vol. 19, No. 2, (2013), pp.95-99.

[4] Youngnam Joun, Woohyun Ahn, "Detecting Repackaged Applications using the Information of App Installation in Android Smartphones", *Journal of Convergence Security,* Vol. 12, No.4, (2012), pp.9-15.

[5] Jungtae Kim, Eul-Gyu Im, "Malicious Family Detection Based on Android Using Similar Class Information", *Journal of Security Engineering,* Vol. 10, No. 4, (2013), pp.441-453.

[6] W. Zhou, X. jiang, Editors, "Dissecting Android Malware : Characterization and Evolution.", *Proceedings of the 33rd IEEE Symposium on Security and Privacy*, (2012). May 23-24, San Francisco, CA

[7] http://www.malgenomeproject.org, 2013. 4

[8] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn and Supachanun Wanapu, "Using of Jaccard Coefficient for Keywords Similarity",

Proceedings of the International MultiConference of Engineering and Computer Scientists 2013, Vol I, IMECS 2013, (2013), March 13-15, Hong Kong.

[9] You Jeong Ham, Daeyeol Moon, Hyung-Woo Lee, aedeok Lim, Jeong Nyeo Kim, “Activation Pattern Analysis on Malicious Android Mobile Applications”, *Proceedings of the 1st International Conference on Artificial Intelligence, Modelling and Simulation(AIMS)*, (2013) May 7-9, Sabah, Malaysia.

[10] You Jeong Ham, Hyung-Woo Lee, “Normal and Malicious Application Pattern Analysis using System Call Event on Android Mobile Devices for Similarity Extraction”, *Journal of Internet Computing and Services(JICS)*, Vol. 14, No. 6, (2013), pp.125-139.

[11] You Jeong Ham, Daeyeol Moon, Hyung-Woo Lee, Jaedeok Lim, Jeong Nyeo Kim, “Android Mobile Application System Call Event Pattern Analysis for Determination of Malicious Attack”, *International Journal of Security and Its Applications(IJSIA)*, Vol. 8, No.1, (2014), pp.231-246.

◐ 저 자 소 개 ◑

함 유 정

2012년 한신대학교 정보통신학과 졸업(학사)
2012년~현재 한신대학교 대학원 컴퓨터공학과 재학중(석사)
관심분야 : 정보보호, 스마트폰 보안, Smart Fuzzing.
E-mail : you86400@hanmail.net

이 형 우

1994년 고려대학교 컴퓨터학과 졸업(학사)
1996년 고려대학교 대학원 컴퓨터학과 졸업(석사)
1999년 고려대학교 대학원 컴퓨터학과 졸업(박사)
1999년~2003년 백석대학교 정보통신학부 조교수
2003년~현재 한신대학교 컴퓨터공학부 부교수, 정교수
관심분야 : 정보보호, 스마트폰 보안, 컴퓨터 포렌식스, 네트워크 보안, Fuzzing.
E-mail : hwlee@hs.ac.kr