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

안드로이드 모바일 정상 및 악성 앱 시스템 콜 이벤트 패턴 분석을 통한 유사도 추출 기법☆

Normal and Malicious Application Pattern Analysis using System Call Event on Android Mobile Devices for Similarity Extraction

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

요 약

안드로이드 기반 오픈 마켓의 개방성으로 인해 일반적인 정상 어플리케이션 뿐만아니라 공격자에 의해 개발된 악성 어플리케이션의 배포 역시 점차 증가하고 있는 추세이다. 악성 어플리케이션들의 확산으로 인한 피해를 줄이기 위해서는 상용 모바일 단말을 대상으로 보다 정확한 방법으로 정상 앱과 악성 앱을 판별할 수 있는 메커니즘이 개발되어야 한다. 이에 본 논문에서는 안드로이드 플랫폼 기반 모바일 단말을 대상으로 정상 앱과 악성 앱으로 부터 이벤트 패턴을 분석하기 위해 안드로이드 오픈 마켓에서 가장 사용자 이용도가 높은 게임 앱을 대상으로 정상 이벤트 패턴을 분석하였고, Android MalGenome Project에서 배포하고 있는 1,260개의 악성 샘플들 중에서 게임 앱 형태에 해당하는 악성 앱과 유사 악성 앱 등을 대상으로 악성 이벤트 패턴을 분석하였다. 이와 같이 안드로이드 기반 모바일 단말에서 정상 앱과 악성 앱을 대상으로 리눅스 기반 시스템 콜 추출 도구인 Strace를 이용해 정상 앱과 악성 앱의 이벤트를 추출하는 실험을 수행하였다. 정상 앱 및 악성 앱이 각각 실행되었을 때 발생하는 이벤트를 수집하여 각각의 이벤트 집합에 대한 연관성 분석 과정을 수행하였다. 이러한 과정을 통해 정상 앱과 악성 앱 각각에 대한 이벤트 발생 특징 및 패턴과 분포도를 분석하여 이벤트 유사도를 추출할 수 있었으며 최종적으로는 임의의 앱에 대한 악성 여부를 판별하는 메커니즘을 제시하였다.

☞ 주제어 : 안드로이드, 정상 및 악성 어플리케이션, 시스템 콜 이벤트, 패턴 분석, 유사도 분석

ABSTRACT

Distribution of malicious applications developed by attackers is increasing along with general normal applications due to the openness of the Android-based open market. Mechanism that allows more accurate ways to distinguish normal apps and malicious apps for common mobile devices should be developed in order to reduce the damage caused by the rampant malicious applications. This paper analysed the normal event pattern from the most highly used game apps in the Android open market to analyse the event pattern from normal apps and malicious apps of mobile devices that are based on the Android platform, and analysed the malicious event pattern from the malicious apps and the disguising malicious apps in the form of a game app among 1260 malware samples distributed by Android MalGenome Project. As described, experiment that extracts normal app and malicious app events was performed using Strace, the Linux-based system call extraction tool, targeting normal apps and malicious apps on Android-based mobile devices. Relevance analysis for each event set was performed on collected events that occurred when normal apps and malicious apps were running. This paper successfully extracted event similarity through this process of analyzing the event occurrence characteristics, pattern and distribution on each set of normal apps and malicious apps, and lastly suggested a mechanism that determines whether any given app is malicious.

☞ keyword : Android, Normal and Malicious Application, System call events, Pattern Analysis, Similarity Analysis

1 School of Computer Engineering, Hanshin University, Gyeonggi, 447-791, Rep. of Korea.
* Corresponding author (hwlee@hs.ac.kr)
[Received 14 October 2013, Reviewed 22 October 2013, Accepted 12 November 2013]
☆ 본 연구는 2013년도 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2012R1A1A2004573)

1. 서 론

최근 안드로이드 기반 상용 스마트워크 단말 사용자가 증가함에 따라 다양한 형태의 앱이 개발되어 안드로이드 마켓을 통해 배포되고 있다. 하지만 누구나 개발해서 배포할 수 있는 안드로이드 기반 오픈 마켓의 개방성으로 인해 일반적인 정상 어플리케이션뿐만 아니라 공격자에 의해 개발된 악성 어플리케이션의 배포 역시 점차 증가하고 있는 추세이다.

공격자는 안드로이드 플랫폼을 탑재한 상용 스마트워크 단말을 대상으로 어플리케이션 내에 악성코드를 삽입한 후 이를 일반 사용자에게 오픈 마켓 또는 인터넷을 통해 배포하게 된다. 이를 통해 일반 사용자의 스마트워크 단말 내 저장된 SMS, 전화번호부 등의 개인정보 뿐만 아니라 상용 스마트워크 단말 내에 저장된 공인인증서 등 금융정보 등을 외부 서버로 유출시키는 공격을 시도할 수 있다. 이와 같이 최근 국내외적으로 널리 보급되고 있는 안드로이드 기반 상용 스마트워크 단말 기반 이용자 환경에서 보안 문제점이 크게 대두되고 있다[1,2,3].

따라서 악성 어플리케이션들의 확산으로 인한 피해를 줄이기 위해서 모바일 단말에 대한 공격을 검출할 수 있는 기법[4,5] 등이 제시되었다. 하지만 상용 모바일 단말을 대상으로 보다 정확한 방법으로 정상 앱과 악성 앱을 판별할 수 있는 메커니즘이 개발되어야 한다. 이에 본 논문에서는 우선 안드로이드 기반 모바일 단말의 최신 보안 취약점을 통해 공격 형태를 분석하고 안드로이드 기반 이벤트 분석 도구인 Strace를 통해서 정상 앱과 악성 앱의 이벤트를 추출한 후에 이를 분석하여 정상 앱과 악성 앱의 특징 및 이벤트 패턴의 연관성을 분석하는 방법을 제시하고자 한다. 제시한 방법을 토대로 본 연구에서는 안드로이드 기반 정상 앱과 악성 앱에 대한 판별 기법을 제시하고자 한다.

구체적으로 본 논문에서는 안드로이드 플랫폼 기반 모바일 단말을 대상으로 정상 앱과 악성 앱으로부터 이벤트 패턴을 분석하기 위해 안드로이드 오픈 마켓에서 사용자 이용도가 가장 높은 게임 앱을 대상으로 정상 이벤트 패턴을 분석하였다. 뿐만 아니라 안드로이드 공식 마켓인 구글 플레이 스토어에서 게임 카테고리에 포함된 게임 앱인 경우 정상 앱으로 위장한 악성 앱의 형태가 가장 많이 나타나기 때문에 우선 게임 카테고리 내 상위 80개의 정상 게임 앱을 대상으로 정상 시스템 콜 이벤트 특성을 분석하였다. 또한 Android MalGenome Project에서 배포하고 있는 1,260개의 악성 샘플들 중에서 게임 앱 형태에 해당하는 악성 앱과 유사 악성 앱 등을 대상으로 악성 이벤트 패턴을 분석하였다.

이와 같이 안드로이드 기반 모바일 단말에서 정상 앱과 악성 앱을 대상으로 리눅스 기반 시스템 콜 추출 도구인 Strace를 이용해 정상 앱과 악성 앱의 이벤트를 추출하는 실험을 수행하였다. 우선 정상 앱 및 악성 앱이 각각 실행되었을 때 발생하는 이벤트를 수집하여 각각의 이벤트 집합에 대한 연관성 분석 과정을 수행하였다. 이러한 과정을 통해 정상 앱과 악성 앱 각각에 대한 이벤트 발생 특징 및 패턴과 분포도를 분석하여 이벤트 유사도를 측정할 수 있었으며 이를 통해 악성 앱 여부를 판별하는 과정을 수행하였다.

본 논문의 구성은 다음과 같다. 2장에서 기존에 존재하는 안드로이드 관련 최신 보안 취약점에 대해 분석하였다. 3장 및 4장에서는 정상 앱과 악성 앱에 대한 이벤트 패턴을 각각 분석하였다. 3장에서는 분석 대상인 정상 앱에 대해 카테고리와 퍼미션 중심으로 분류해서 각각의 앱 실행 시 발생하는 이벤트 패턴의 특징 및 연관성에 대해서 분석하였다. 4장에서는 주요 악성 앱의 작동 방식을 중심으로 분석한 후에 각각의 악성 앱 실행 시 발생하는 이벤트 패턴의 특징 및 연관성에 대해서 분석한다. 5장에서는 3장과 4장에서 분석한 결과를 토대로 정상 앱과 악성 앱의 이벤트 패턴의 특징을 비교 분석하였으며 이벤트 유사도를 기반으로 임의의 앱에 대해 악성 여부를 판별할 수 있는 기법을 제시하였고 6장에서는 결론을 제시하였다.

2. 안드로이드 최신 보안 취약점 분석

2.1 안드로이드 단말 보안 위협

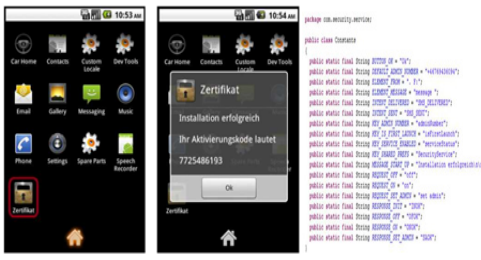
안드로이드 플랫폼 기반 스마트워크 단말을 대상으로 한 악성코드가 오픈 마켓을 통해 널리 확산되고 있다[6]. 안드로이드 기반 스마트워크 단말과 관련된 보안 위협을 토대로 다양한 형태의 취약점이 발견되고 있다.

안드로이드 플랫폼을 대상으로 한 보안 위협이 증가하고 있는 이유는 안드로이드 플랫폼이 개방성, 휴대성, 저성능 및 접근성의 기능을 제공하기 때문에 최근 공격 유형이 발생하고 있다. 결국 안드로이드 플랫폼의 보안 취약성으로 인해 다양한 형태의 공격이 발생하고 있으며 이를 해결하기 위해서는 사용자 중심의 능동적인 취약성 진단 과정과 개인정보에 대한 외부 유출 방지를 위한 보안성 강화 방안이 제시되어야 한다.

2.2 안드로이드 기반 악성 어플리케이션 보안 위협

공격자는 정상적인 형태로 보이는 모바일 앱 내에 악성코드를 포함하여 일반 사용자의 스마트워크 단말에 배포 및 설치하게 된다. 일반 안드로이드 단말 사용자는 앱 실행 후에 화면 내에 포함된 버튼 부분에 대한 단순 클릭 등의 과정으로 인해 사용자도 모르게 악성 코드가 구동되며, 이 과정에서 스마트워크 단말 내에 저장된 개인정보 등이 외부로 유출되는 문제점이 발생하게 된다. 악성코드가 실행되면 공격자는 사용자 단말 내 저장된 리소스에 대한 불법적인 접근권한을 획득하게 된다.

또한 공격자에 의해 배포된 악성 앱이 실행되면서 특정 서버로부터 악성코드를 다운로드 받아 설치되는 경우에는 기존의 일반적인 모바일 백신에 의해서 쉽게 검출되지 않는다는 문제점이 발생할 수 있어서 더욱더 큰 문제가 될 수 있다. 예를 들어 그림 1과 같이 일반 사용자에게 C&C 서버로부터 추가적인 악성코드를 다운로드하도록 유도하기 위해 공격자는 SMS 문자 등을 사용자 단말에 전송하고 이를 통해 일반 사용자 단말내 설치된 정상 앱을 악성코드가 포함되도록 변경할 수 있다. 이 경우 역시 사용자의 SMS 내역 등 개인정보를 공격자가 지정한 외부 서버로 전송하게 된다.



(그림 1) 사용자 개인정보를 외부로 유출하는 악성 앱
(Figure 1) Privacy Leakage Malicious App

이 외에도 정상적인 모바일 보안 어플리케이션으로 위장하여 C&C 서버와 통신하면서 온라인 뱅킹 공격 등을 수행하여 금융피해를 유발하는 악성 어플리케이션도 존재한다. 정상 어플리케이션으로 위장하여 사용자의 인증번호를 가로채는 등의 공격을 통해 스마트워크 단말으로 이용 가능한 온라인 뱅킹 과정에 피해를 유발시키는 공격을 수행하기도 한다.

표 1과 같이 안드로이드 플랫폼을 대상으로한 주요 악성코드를 분석해 보면 사용자를 속이거나 어플리케이션 설치 프로그램으로 위장하여 문자 전송 등의 과정을 수행하는 경우, 게임 또는 만화를 리패키징하여 악성코드를 추가하는 경우, 정상 유틸리티를 리패키징하여 악의적인 기능을 수행하는 형태 등으로 나눌 있다.

(표 1) 안드로이드 플랫폼을 대상으로한 주요 악성코드
(Table 1) Top 10 Malware on Android Platform

순위	악성코드 명	주요 증상
1	Android-Trojan/SmsSend	사용자를 속이거나 사용자 몰래 문자를 전송
2	Android-Trojan/FakeInst	어플리케이션 설치 프로그램으로 위장해 문자 전송 등으로 수익을 챙김
3	Android-Spyware/Geimini	정상 어플리케이션을 리패키징해 개인정보 탈취
4	Android-Exploit/Rootor	취약점을 이용해 시스템 권한 취득
5	Android-Trojan/LightDD	성인 어플리케이션으로 위장해 사용자 스마트폰 정보 유출
6	Android-Dropper/Anserver	정상 앱을 리패키징해 다른 악성코드를 설치
7	Android-Trojan/Gdream	게임, 만화(화보) 리패키징하여 악성코드를 추가
8	Android-Spyware/BgService	정상 앱을 리패키징해 악성코드를 삽입
9	Android-Trojan/Boxer	어플리케이션 설치 프로그램으로 위장하여 유료 문자 발송
10	Android-Spyware/Adrd	정상 유틸리티를 리패키징해 악의적인 기능 수행

이와 같은 악성코드를 검출하기 위해 모바일 백신 프로그램들이 개발되고 있다. AhnLab에서 개발된 'V3 Mobile' 백신과 이스트소프트 알약은 악성 앱에 대한 스미싱 공격을 탐지 및 차단하는 기능을 제공한다. 또한 Kaspersky Mobile Security는 악성 코드 및 바이러스 차단 기능, 실시간 검사 및 예약 검사 기능을 제공하며 통화 및 SMS 서비스에 대한 필터링 기능을 제공한다.

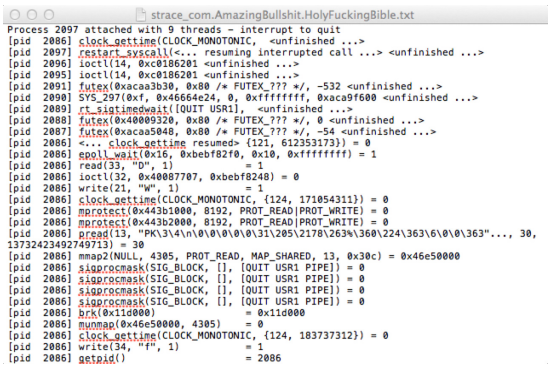
하지만 기존 백신 등은 최근 급증하고 있는 C&C 서버 기반 악성코드 다운로드 방식 또는 정상 형태의 앱에 악성코드를 추가하는 업데이트 방식의 앱에 대해서는 효과적으로 검출하지 못한다는 문제점이 있다.

결국 이러한 문제점을 해결하기 위해서는 안드로이드 기반 스마트워크 단말에서 실시간으로 구동되는 과정에서 실시간으로 발생하는 내부 시스템 콜 이벤트를 대상으로 정상과 악성 여부를 판별하는 과정이 필요하다는 것을 알 수 있다. 이에 본 논문에서는 실제 모바일 단말을 대상으로 앱 실행시 발생하는 시스템 콜 이벤트 패턴을 수집 및 분석하여 임의의 앱에 대한 악성 여부를 판별하는 방법을 제시하고자 한다.

2.3 안드로이드 시스템 콜 이벤트 수집

안드로이드 기반 모바일 단말 내에 설치된 앱을 실행하게 되면 커널을 통해 발생하는 시스템 이벤트를 확인할 수 있다. 아래 그림 2와 같이 안드로이드 플랫폼에서 구동되는 앱을 실행하게 되면 각 프로세스에 의해 호출

되는 시스템 콜을 추출하여 해당 앱의 내부 작동 방식과 특성에 대해 분석할 수 있다.



(그림 2) 모바일 앱 실행시 발생하는 시스템 콜 이벤트
(Figure 2) System Call Event Activated after Mobile App Execution

앱을 실행시 발생하는 시스템 콜 이벤트를 순차적으로 살펴보면 그림 3과 같은 패턴을 보인다. 가로축은 발생하는 이벤트 순서번호로 하고 세로축은 호출되는 이벤트에 대해 고유 번호를 부여하여 이를 그래프로 나타내었을 경우 각 앱마다 조금씩 다른 패턴의 시스템 콜 이벤트가 나타나는 것을 확인할 수 있다.

따라서 본 논문에서는 안드로이드 기반 모바일 단말에서 사용되는 정상 및 악성 앱으로부터 발생하는 시스템 콜 이벤트를 수집하여 각각의 특성을 분석하는 과정을 수행하였다. 이를 통해 정상 앱과 악성 앱 실행시 발생하는 이벤트 특성을 파악할 수 있으며 이를 통해 정상 및 악성 앱의 유사도를 측정할 수 있다.

3. 정상 게임 어플리케이션 이벤트 분석

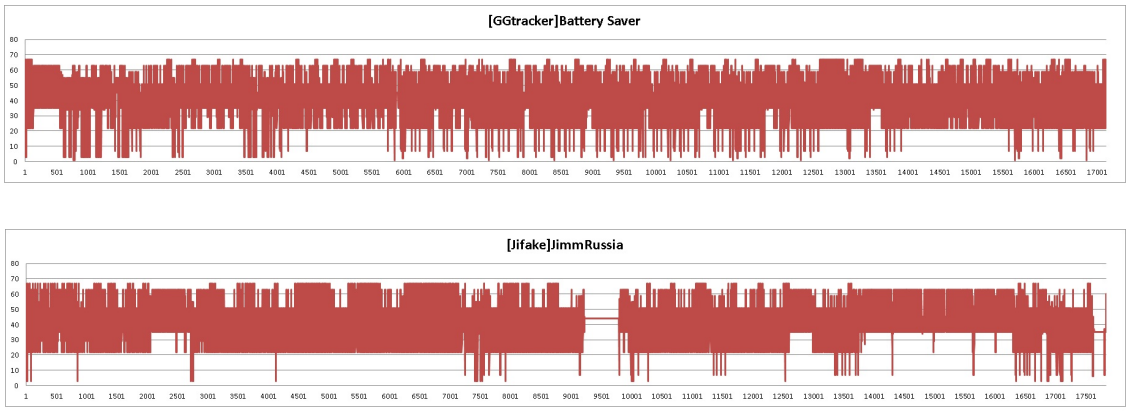
안드로이드 공식 마켓인 구글 플레이 스토어에서 게임 카테고리에 포함된 앱인 경우 정상 앱으로 위장한 악성 앱의 형태가 가장 많이 나타나기 때문에 우선 게임 카테고리 내 상위 80개의 정상 게임 앱을 대상으로 정상 시스템 콜 이벤트 특성을 분석하였다. 구체적으로 게임 카테고리 내 세부 7개 일부 항목을 분석하였고 이를 대상으로 이벤트 분석, 퍼미션 중심 분류 등을 수행하였고 이를 통해 정상 어플리케이션의 시스템 콜 이벤트 특징에 대해서 분석할 수 있었다.

3.1 카테고리 중심 분석

우선 정상 게임 어플리케이션을 카테고리 중심으로 분석하였다. 그 다음 정상 게임 어플리케이션 내 분류된 7가지 카테고리를 대상으로 정상 게임 어플리케이션 실행 시 발생하는 시스템 콜 이벤트와 특징을 분석하였다.

3.1.1 분석 대상 게임 앱

구글 플레이 스토어 내 게임 카테고리는 세부적으로 7가지로 분류되어 있는데 일반적으로 많이 사용되고 있는



(그림 3) 앱에서 발생하는 시스템 콜 이벤트 패턴
(Figure 3) System Call Event Pattern Derived from Mobile App

상위 80개의 정상 게임 앱을 대상으로 분류하면 다음 표 3과 같다. 우선 말달리자 for Kakao 등 약 25개의 어플리케이션과 같이 아케이드 카테고리에 포함되는 모바일 게임 어플리케이션을 대상으로 이벤트를 분석하였다. 그리고 모두의 마블 for Kakao 등 약 30개의 어플리케이션과 같이 캐주얼 게임 카테고리에 포함되는 어플리케이션 및 애니팡 for Kakao 등 약 11개의 어플리케이션이 포함된 두뇌게임 및 퍼즐 카테고리를 대상으로 이벤트 특성을 분석하였다. 이밖에도 자동차 경주, 스포츠 게임, 엔터테인먼트, 카드 게임 카테고리에 포함되는 앱으로 부터 정상 이벤트 패턴 및 특성을 분석하였다.

3.1.2 게임 앱 이벤트 분석

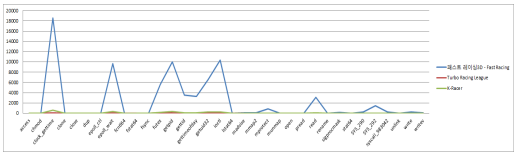
앞서 분류한 7가지의 카테고리의 정상 게임 어플리케이션을 바탕으로 각 카테고리에 있는 정상 게임 어플리케이션들의 이벤트를 기존 [8] 기법과 달리 Strace를 통해 분석하였다.

아케이드 카테고리 내 25개의 정상 게임 어플리케이션들을 대상으로 이벤트 분포를 분석한 결과 clock_gettime, epoll_wait, futex, getpid, ioctl, mprotect, read, SYS_292 등과 같은 8개의 이벤트가 다른 이벤트에 비해 상대적으로 많이 발생한 것을 확인할 수 있었다.

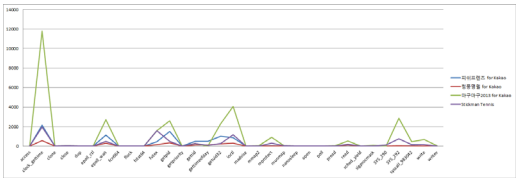
캐주얼 게임 카테고리 내 30개의 정상 게임 어플리케이션들을 대상으로 이벤트 분포를 분석한 결과 아케이드 카테고리에서 발생한 이벤트와 유사하게 clock_gettime, epoll_wait, futex, getpid, ioctl, mprotect, read, SYS_292, write 등과 같은 9개의 이벤트가 상대적으로 많이 발생한 것을 확인할 수 있었다. 이는 캐주얼 게임 카테고리에 포함된 앱과 아케이드 형태의 게임 앱이 유사한 작동 구조를 포함하고 있기 때문에 유사한 형태의 이벤트가 많이 발생한 것으로 생각된다.

두뇌게임 및 퍼즐 카테고리 내 11개의 정상 게임 어플리케이션들을 대상으로 이벤트 분포를 분석하면 앞서 제시한 두 카테고리의 앱에서와 유사한 부분도 있지만 futex, ioctl, gettimeofday, SYS_290 등과 같은 4개의 이벤트가 상대적으로 많이 발생한 것을 알 수 있다. 특히 두뇌게임 및 퍼즐 카테고리 내 어플리케이션들에서는 아케이드나 캐주얼 게임 카테고리 내 어플리케이션들과는 달리 gettimeofday, SYS_290 같은 시스템 콜 이벤트가 발생한 것을 알 수 있었다.

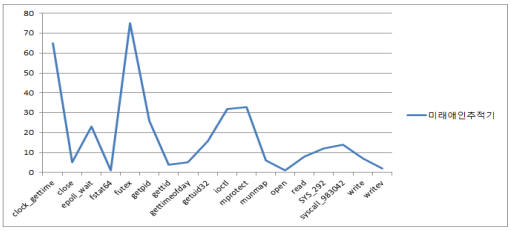
마지막으로 앞서 서술한 카테고리 외 기타 카테고리들인 자동차 경주, 카드 게임, 엔터테인먼트, 스포츠 게임 카테고리에 포함되는 정상 게임 어플리케이션의 콜 카운트 그래프를 제시한 것은 그림 4와 같다.



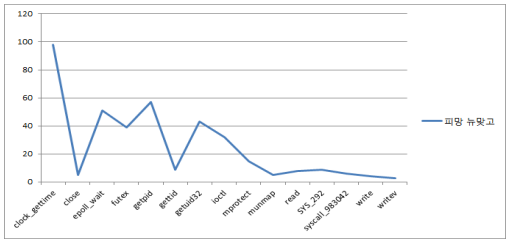
(a) 자동차 경주



(b) 스포츠 게임



(c) 엔터테인먼트



(d) 카드 게임

(그림 4) 앱 시스템 콜 이벤트 그래프

(Figure 4) App System Call Event Graph

자동차 경주. 스포츠 게임, 엔터테인먼트, 카드 게임 내 정상 게임 어플리케이션의 시스템 콜 이벤트들의 분포도 다른 카테고리 내 어플리케이션의 시스템 콜 이벤트 분포와 같이 주로 clock_gettime, epoll_wait, ioctl, getuid32, mprotect 등의 시스템 콜 이벤트가 주로 나타나는 것을 알 수 있었다.

자동차 경주와 스포츠 게임 앱인 경우 유사한 형태의 이벤트가 발생하는 것을 확인할 수 있었으며 카드 게임과 엔터테이먼트 관련 앱 역시 유시한 패턴을 보이는 것을 확인할 수 있었다.

3.2 퍼미션 중심 분석

안드로이드 공식 마켓인 구글 플레이 스토어 내에는 정상 앱으로 위장한 악성 앱들이 많이 분포하고 있다. 본 절에서는 이러한 악성 앱들에서 나타나는 악성 유사 퍼미션들을 분류하고 각 해당되는 퍼미션들을 포함하고 있는 정상 게임 어플리케이션 실행 시 발생하는 이벤트와 특징을 분석하였다.

3.2.1 퍼미션 분류

구글 플레이 스토어 내 게임 카테고리 앱 중 인기있는 상위 80개의 정상 게임 앱의 퍼미션을 분석한 결과 시스템에 악영향을 끼칠 수 있는 권한은 약 20여개로 이를 퍼미션 집합으로 분류하면 System, SMS, Contact, Location와 관련된 권한의 4가지 퍼미션 집합을 다음 표 2와 같이 나타낼 수 있다.

(표 2) 퍼미션 중심 분류
(Table 2) Permission based Classification

SYSTEM	SMS	CONTACT	LOCATION
GET_TASKS	RECEIVE_SMS	READ_CONTACTS	ACCESS_FINE_LOCATION
WRITE_SETTINGS	SEND_SMS	READ_CALL_LOG	ACCESS_COARSE_LOCATION
SYSTEM_ALERT_WINDOW		CALL_PHONE	
INSTALL_PACKAGES		READ_LOGS	
MOUNT_UNMOUNT_FILESYSTEMS		WRITE_CONTACTS	
UPDATE_DEVICE_STATE			
RESTART_PACKAGES			

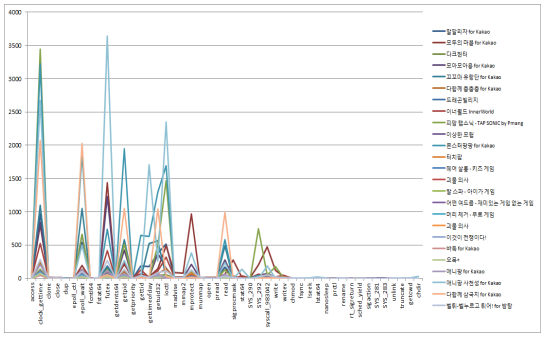
퍼미션 권한들을 크게 4가지의 카테고리로 분류하였는데 이를 상위 80개의 게임 정상 앱을 토대로 나누어보면 SYSTEM 카테고리에 포함되는 어플리케이션은 말달리자 for Kakao 외 24개, SMS 카테고리에 포함되는 어플리케이션은 다크 헌터 외 11개, Contact 카테고리에 포함되는 어플리케이션은 아이러브커피 for Kakao 외 12개, 마지막으로 Location 카테고리에 포함되는 어플리케이션은 쿠키런 for Kakao 외 7개의 어플리케이션이 존재한다.

3.2.2 퍼미션 별 앱 이벤트 분석

앞서 4가지의 퍼미션 카테고리로 정상 게임 어플리케이션을 분류한 결과를 바탕으로 각 카테고리에 포함되어 있는 정상 게임 어플리케이션들의 이벤트를 분석하였다.

3.2.2.1 System

System 카테고리에 포함되는 정상 게임 어플리케이션의 시스템 콜 이벤트 그래프를 제시한 것은 그림 5와 같다.

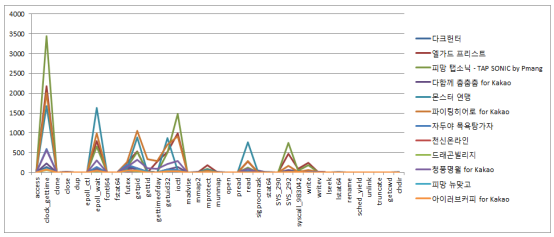


(그림 5) SYSTEM 퍼미션 중심 이벤트 그래프
(Figure 5) SYSTEM Permission oriented Call Count Graph

SYSTEM에 포함되는 정상 게임 어플리케이션에서는 clock_gettime, epoll_wait, futex, getpid, getuid32, ioctl, mprotect, read, SYS_292, syscall_983042 등 10개의 시스템 콜 이벤트에 카운트가 많이 분포되어있으며, 디렉토리를 읽는데 사용되는 시스템 콜 이벤트인 getdents64, 최근 작업 디렉토리를 가져오는 시스템 콜 이벤트인 getcwd 등 시스템과 관련된 시스템 콜 이벤트들이 상대적으로 발생한 것을 확인할 수 있었다.

3.2.2.2 SMS

SMS 카테고리에 포함되는 정상 게임 어플리케이션의 시스템 콜 이벤트 그래프를 제시한 것은 그림 6과 같다.



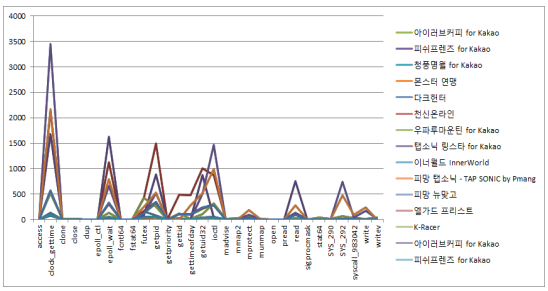
(그림 6) SMS 퍼미션 중심 이벤트 그래프
(Figure 6) SMS Permission oriented Call Count Graph

SMS에 포함되는 정상 게임 어플리케이션에서 발생하는 시스템 콜 이벤트에는 앞서 분석한 SYSTEM 카테고리에 포함되는 정상 게임 어플리케이션에서 발생한 시스템 콜 이벤트와는 다르게 SYSTEM과 관련있는 시스템 콜 이벤트들인 chmod, prctl, fysnc 등이 나타나지 않았다.

이는 SMS와 관련된 퍼미션을 가지고 있는 정상 게임 어플리케이션의 시스템 콜 이벤트를 나타냈기 때문이다. 하지만 정상 게임 어플리케이션이 SMS와 관련된 퍼미션을 가지고 있더라도 해당 퍼미션이 앱을 실행하는 도중에 활성화가 되지 않아 SMS와 관련있는 시스템 콜 이벤트인 recvmsg, recvfrom, send 등을 발견하지는 못하였다.

3.2.2.3 Contact

Contact 카테고리에 포함되는 정상 게임 어플리케이션의 시스템 콜 이벤트 그래프를 제시한 것은 그림 7과 같다.



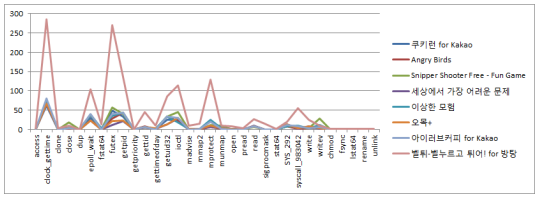
(그림 7) Contact 퍼미션 중심 이벤트 그래프
(Figure 7) Contact Permission oriented Call Count Graph

Contact에 포함되는 정상 게임 어플리케이션에서 발생하는 시스템 콜 이벤트에는 SYSTEM과 SMS 내 정상 게임 어플리케이션에 발생한 lseek, chmod, rename, chdir, getcwd, truncate, unlink 등의 시스템 콜 이벤트가 발생하지 않았다. 그 이유는 Contact과 관련된 퍼미션이 포함된 정상 게임 어플리케이션에 대한 시스템 콜 이벤트를 분석한 것이기 때문에 메시지나 시스템과 같이 파일과 관련있는 시스템 콜 이벤트들은 나타나지 않았다. 하지만 Contact 카테고리 역시 SMS 카테고리 내 포함된 정상 어플리케이션들과 마찬가지로 앱 실행 시 Contact과 관련된 퍼미션들이 활성화가 되지 않기 때문에 관련 있는 시스템 콜이 나타나지 않음을 확인할 수 있다.

3.2.2.4 Location

Location 카테고리에 포함되는 정상 게임 어플리케이션의 이벤트 콜 이벤트 그래프는 그림 8과 같다.

앞서 분석한 세 퍼미션 집합과 비슷하게 Location에 포함되는 정상 게임 어플리케이션들도 시스템 콜 이벤트의 분포가 형성된다. 하지만 Location에 포함되는 정상 게임 어플리케이션에서는 세 퍼미션 집합에는 거의 발생하지 않았던 close, open 등의 시스템 이벤트 콜이 발생하는 것을 그래프에서 확인할 수 있다. 이와 같이 게임 카테고리 및 퍼미션을 중심으로 분석한 결과를 종합적으로 분석한 결과를 제시하면 다음과 같다.



(그림 8) Location 퍼미션 중심 이벤트 그래프
(Figure 8) Location Permission oriented Call Count Graph

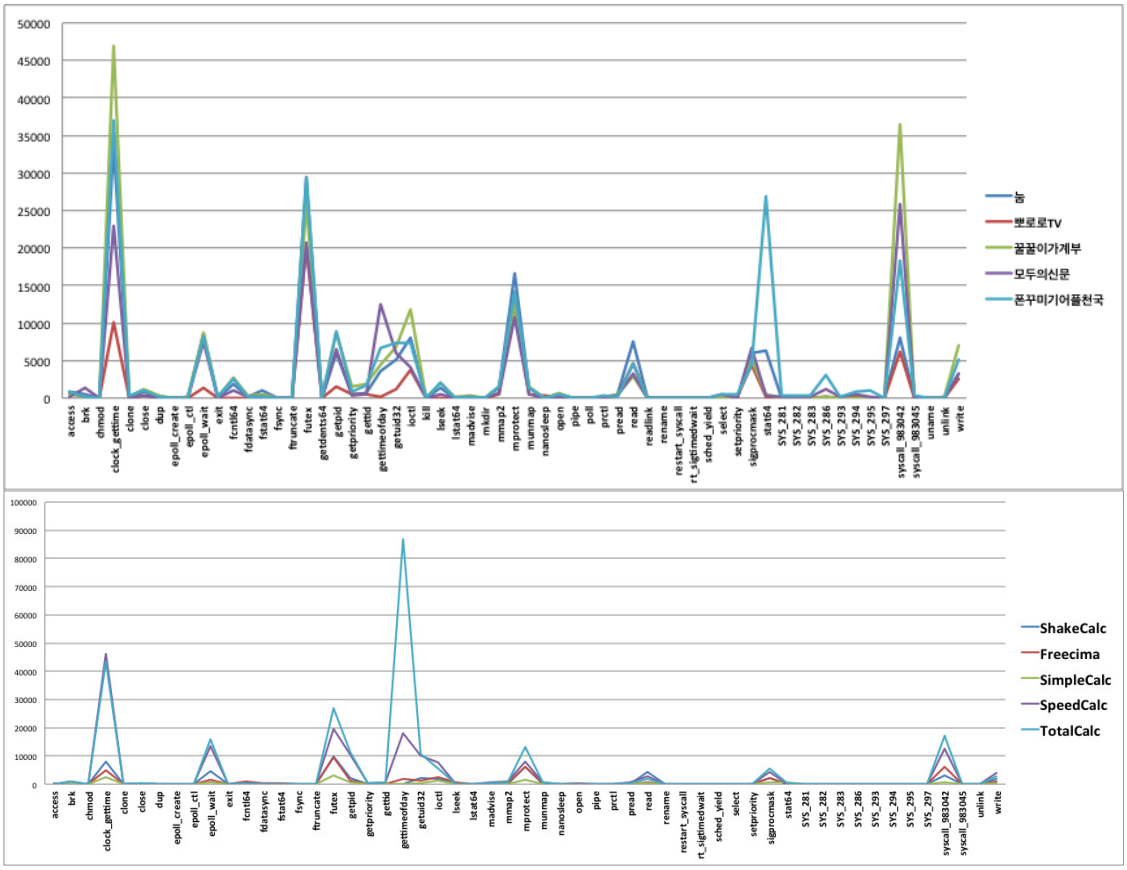
3.3 정상 게임 어플리케이션 이벤트 특징 분석

앞서 분석한 카테고리 중심 분석 및 퍼미션 중심 분석과 관련된 정상 게임 어플리케이션들은 각각의 특징을 가지고 있기도 하지만 그렇지 않은 것들도 존재하였다. 정상 게임 어플리케이션의 시스템 콜 이벤트 그래프는 다음과 같다. 정상게임 어플리케이션에는 clock_gettime, clone, close, epoll_wait, futex, getpid, gettid, gettimeofday, getuid32, ioctl, madvise, mmap2, munmap, read, SYS_292, syscall_983042, write, writev 등의 이벤트가 존재하는 것을 확인할 수 있었다. 이를 통해 정상 어플리케이션에는 주로 위와 같이 약 18개의 이벤트들이 주기적으로 나타난다고 추측할 수 있다.

그림 9와 같이 서로 다른 정상 앱 다섯 가지에서 발생하는 시스템 콜 이벤트 패턴을 보면 유사한 형태로 발생하는 것을 확인할 수 있었다. 또한 예를 들어 계산기와 같이 동일 기능을 제공하는 서로 다른 정상 앱을 구동하였을 경우 발생하는 시스템 콜 이벤트 패턴 역시 유사도가 높게 나타나는 것을 확인할 수 있었다.

4. 악성 게임 어플리케이션 이벤트 분석

본 장에서는 Android MalGenome Project에서 1260개의 악성 샘플을 배포받아 분석한 내용을 제시하였으며 Android MalGenome Project에서 작성한 관련 연구[9]를 토대로 악성 앱 분류 및 이벤트 분석과 악성 어플리케이션의 이벤트 특징에 대해서 제시한다.



(그림 9) 정상 앱 시스템 콜 이벤트 (상: 임의의 정상 앱, 하: 유사 계산 기능 정상 앱)
(Figure 9) Normal App System Call Event (Up: Random Normal App , Down: Calc Function)

4.1 작동 방식 중심 분류

Android MalGenome Project는 1260개의 악성 앱 샘플들을 특징에 따라서 크게 Malware Installation, Activation, Malicious Payloads, Permission Uses로 분류하고 있으며[9] 본 논문에서는 Malware Installation에 관한 내용을 분석하였다. Malware Installation는 Repacking, Update-Attack, Drive-by Download, Standalone으로 분류된다.

먼저 Repacking은 앱을 재포장하는 방법으로써, 악성 개발자가 안드로이드 공식 마켓 등에 등록한 앱을 다운받아 apk 또는 .jar 파일을 재수정해서 악성코드를 삽입한 다음 재배포하는 방식으로 정상 앱처럼 위장해서 사용자들의 개인정보 및 금융정보를 유출하는 사례가 빈번히 발생하고 있다.

두 번째로 Update Attack은 사용자가 앱을 업데이트할 때 다른 악의적인 앱이 설치되는 방식으로 사용자도 모르는 앱이 설치될 뿐만 아니라 사용자의 사생활 정보를 유출시키거나 과금을 유발시킬 수 있다. 또한 Update Attack은 자체 업데이트 기능을 가지며 크게 네 가지 기술로 분류할 수 있다[10]. 첫째, 원래 존재하는 어플리케이션에 사용자가 업데이트를 하는 방식이다. 두 번째, 안드로이드의 DexClassLoader 클래스를 사용해서 컴파일된 안드로이드 코드를 동적으로 로딩하는 것과 어플리케이션의 일부 설치되지 않은 코드의 실행을 허용하는 방식이다. 세 번째, Java의 Runtime 클래스를 사용함으로써 실행되어질 수 있는 네이티브 코드를 포함하는 실행 가능한 코드 또는 .so 라이브러리라고도 불리는 이진을 공유하는 객체 파일을 동적 로딩하는 방식이다. 마지막 방식으로는 악의적인 shellcode와 같이 과금 부과를 포함하는 mp3, jpg, flash, pdf 등 파일의 동적 로딩과 그것의 시스

템 라이브러리들 또는 파일 타입을 다루는 외부 어플리케이션 안에서 취약점을 탐지함으로써 실행하는 기술이 있다.

세 번째로 Drive-by Download 공격 기법은 사용자 모르게 악성코드를 자동으로 다운로드 받아 실행하는 원격 공격의 일종으로 주로 use-after-free 취약점을 Heap Sparying 기법으로 공격하는 사례 등이 발견되고 있다. Drive-by Download 공격 기법은 패치 사이클이 길기 때문에 해당 취약점이 패치되기 이전에 공격이 이루어져 사용자들의 사생활 정보를 유출시킬 수 있다. 이 Drive-by Download 공격 또한 Update Attack 공격과 마찬가지로 다른 악성 앱에 비해서 탐지가 어렵다는 단점이 존재한다.

마지막으로 Standalone은 다른 어떤 장치의 도움도 필요 없이 홀로 운영되는 앱의 종류를 말한다. 이 Standalone은 크게 네 가지 그룹으로 분류할 수 있다. 첫 번째는 어플리케이션 스스로 스파이웨어 앱으로 간주되는 그룹으로, 피해자의 스마트폰에 설치되는 것을 목적으로 한다. 두 번째 그룹은 합법적인 정상 앱인 것처럼 보이지만 사용자 몰래 SMS를 보내거나 사용자의 신원을 보증하는 정보 등을 가로챔으로써 악성 동작을 수행하는 가짜 어플리케이션을 포함하고 있다. 세 번째 그룹은 의도적으로 권한이 없는 SMS를 보내거나 자동으로 소소한 부가가치 서비스에 가입하는 등의 악의적인 기능을 가지는 어플리케이션들이 포함되어있다. 그러나 두 번째 그룹과 다른 점은 가짜로 위장하지 않았다는 점이다. 마지막으로, 네 번째 그룹은 기능들이 잘 작동하기 위해 루트 권한에 의존하는 어플리케이션들을 포함하고 있다. 하지만 이러한 어플리케이션들은 루트 권한을 부여하는 사용자에게 의견을 묻지 않고 내장된 보안 샌드박스를 우회하는 루트 공격을 활용한다[9].

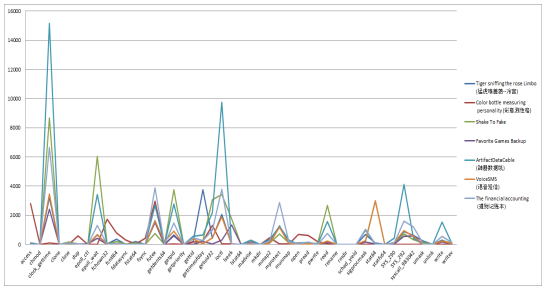
4.2 작동 방식 별 앱 이벤트 분석

본 절에서는 앞서 살펴본 Repacking, Update Attack, Drive-by Download, Standalone 네 가지의 작동 방식으로 분류한 것 중 모바일 백신에 상대적으로 검출되기 쉬운 Repacking이나 Standalone 대신에 상대적으로 탐지하기 어려운 Update-Attack과 Drive-by Download 방식을 대상으로 Strace를 통해 1260개의 악성 샘플들을 분석하였다.

4.2.1 Update Attack

Update Attack 방식에 포함되는 악성은 AnserverBot, BaseBridge, DroidKungFu Update, Plankton 네 가지로 되어있으며 각 187개, 122개, 1개, 11개의 apk 파일이 포함되어있다. 본 연구에서는 각 악성에 포함된 apk 파일 중 2개를 대상으로 악성 어플리케이션의 이벤트를 분석하였다.

Update Attack 방식에 포함된 4가지의 악성 중 7개의 악성 어플리케이션의 시스템 콜 이벤트를 그래프로 제시한 것은 다음 그림 10과 같다.



(그림 10) 악성 업데이트 공격 앱 시스템 콜 이벤트
(Figure 10) Malicious Update Attack System Call Event

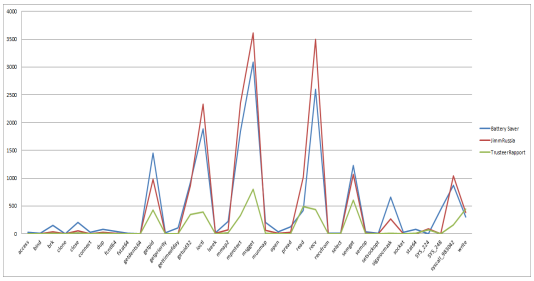
Update Attack 내 7개의 악성 어플리케이션들을 대상으로 이벤트 분포를 분석하면 앞서 정상 게임 어플리케이션의 시스템 콜 이벤트에는 상대적으로 발생하지 않았던 fchown32, fdatasync, mkdir, rmdir, statfs64, umask 등의 시스템 콜 이벤트가 나타났다. 이러한 시스템 콜 이벤트는 디렉토리를 생성 및 제거하거나 파일의 디스크에 있는 데이터 동기화, 파일 시스템 정보 획득, 파일 마스크의 생성 등의 기능을 하며 상대적으로 사용자의 모바일 단말 내 악영향을 미치게 된다.

4.2.2 Drive-by Download

Drive-by Download 방식에 포함되는 악성은 GGTraker, JiFake, Spitmo, Zitmo 네 가지로 되어있으며 각 1개의 apk 파일이 포함되어있다. 본 연구에서는 각 악성에 포함된 apk 파일을 대상으로 악성 어플리케이션의 이벤트를 분석하였다.

해당 악성 앱은 프리미엄 SMS 서비스를 전송하기 위해 원격 서버에 폰 번호를 전송하며 이러한 기능의 알림을 피하기 위해 들어오는 메시지를 차단한다.

Drive-by Download 방식에 포함된 4가지의 악성 중 3개의 악성 어플리케이션의 시스템 콜 이벤트를 그래프로 제시한 것은 다음 그림 11과 같다.



(그림 11) 악성 다운로드 유도 공격 앱 시스템 콜 이벤트
(Figure 11) Malicious Drive-by Download Attack System Call Event

Drive-by Download 내 3개의 악성 어플리케이션들을 대상으로 이벤트 분포를 분석하면 앞서 Update Attack에서 나타난 시스템 콜 이벤트와는 달리 bind, brk, connect, msgget, recv, recvfrom, select, semget, semop, setsockopt 등의 시스템 콜 이벤트가 발생하였다. 발생한 시스템 콜 이벤트는 프로세스의 데이터 세그먼트 크기를 변경하거나 메시지 큐의 식별번호를 반환, 소켓으로부터 데이터와 메시지를 읽어 들이는 기능을 한다. 이러한 기능으로 봤을 때 Drive-by Download는 특정 외부 서버로부터 지시 작업을 수행해서 사용자의 SMS 등 개인정보 및 금융정보를 유출시키거나 사용자를 악성 URL로 접속하게 해서 악성 어플리케이션을 다운로드 받도록 유도시킨다는 것을 유추할 수 있다.

다음 절에서는 Update Attack과 Drive-by Download에서 분석한 10개의 악성 어플리케이션의 이벤트의 특징을 분석한 것을 제시한다.

4.3 악성 어플리케이션 이벤트 특징 분석

앞서 Update Attack와 Drive-by Download에 포함된 악성 apk의 이벤트를 분석하면 다음과 같다. 악성 앱인 경우 정상 어플리케이션에서 주로 나타나는 clock_gettime, epoll_wait, futex, getpid, getuid32, ioctl, mprotect, read, SYS_292, write 등의 시스템 콜 이벤트가 발생하면서 동시에 정상 어플리케이션에서는 발생하지 않는 bind, brk, connect, fchown32, fdatasync, fsync, mkdir, msgget, recv, recvfrom, rmdir, select, semget, semop, setsockopt, statfs64, umask 등 17개의 시스템 콜 이벤트들이 나타나는 것을 확인할 수 있었다. 이를 통해 어플리케이션 내 위와 같이 17개의 시스템 콜 이벤트가 발생하면 해당 어플리케이션은 악성일 가능성이 있다고 추측할 수 있었다.

또한 그림 12와 같이 서로 다른 악성 앱 다섯 가지에서 발생하는 시스템 콜 이벤트 패턴을 보면 전체적으로 매우 유사한 형태인 것을 확인할 수 있었다. 이와 함께 유사 기능을 포함하고 있는 서로 다른 악성 앱을 구동하였을 경우 발생하는 시스템 콜 이벤트 패턴 역시 유사도가 높게 나타나는 것을 확인할 수 있었다.

결국 정상과 악성 앱을 대상으로 안드로이드 플랫폼을 대상으로 실시간으로 구동하였을 경우 추출되는 시스템 콜 이벤트는 정상과 악성을 구분 지을 수 있는 특성을 포함하고 있다는 것을 확인할 수 있었다.

5. 정상 및 악성 어플리케이션 이벤트 비교 분석 및 판별

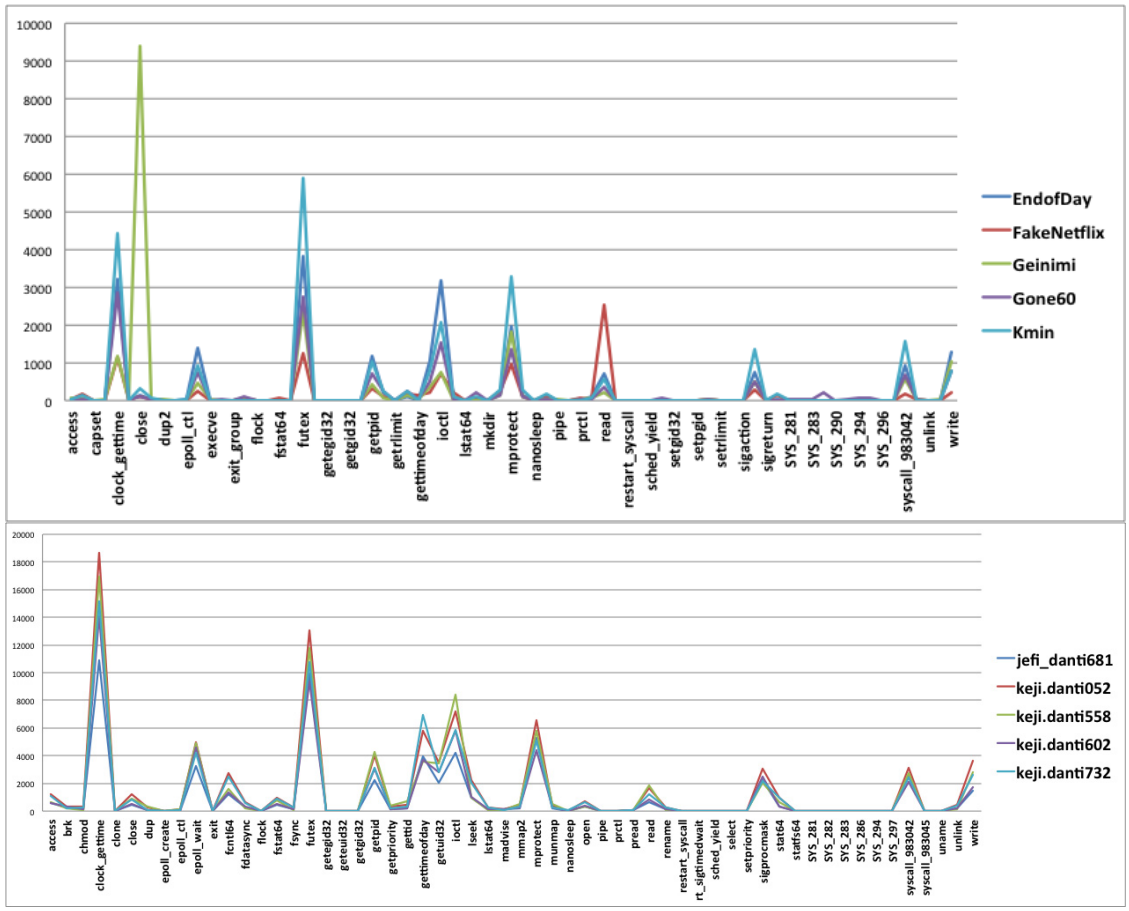
3장과 4장에서 각 정상 어플리케이션과 악성 어플리케이션의 이벤트 분석한 결과를 토대로 본 장에서는 정상 및 악성 어플리케이션의 이벤트를 비교분석한 결과를 제시한다. 정상 어플리케이션과 악성 어플리케이션에서 각각 나타나는 이벤트를 자세히 구분하기 위해 수식을 통해서 분석하였다.

5.1 정상 및 악성 앱 이벤트 비교 분석

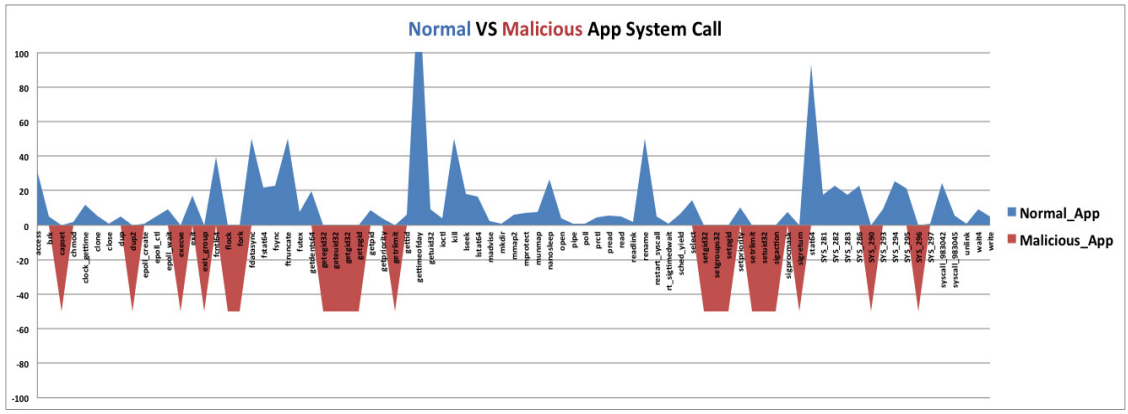
지금까지 분석한 80개의 정상 게임 어플리케이션과 10개의 악성 어플리케이션의 이벤트에는 악성 어플리케이션에만 발생하는 이벤트와 악성 어플리케이션과 정상 어플리케이션 모두 발생하는 이벤트, 정상 어플리케이션에만 발생하는 이벤트들이 존재했다. 그림 13과 같이 악성 어플리케이션에서만 발생한 시스템 콜 이벤트는 bind, brk, connect, fdatasync, mkdir, msgget, pwrite, recv, recvfrom, rename, rt_sigreturn, semget, semop, setsockopt, socket, statfs64, SYS_224, SYS_248 등 18개이며, 정상 어플리케이션에서만 발생한 시스템 콜 이벤트는 chdir, flock, getcwd, nanosleep, poll, prctl, rt_sigreturn, sigaction, SYS_281, SYS_283 등 11개라는 것을 알 수 있었다. 또한 정상 앱과 악성 앱에서 모두 발생한 시스템 콜 이벤트는 access, chmod, clock_gettime 등 40개인 것을 확인할 수 있다.

5.2 정상 및 악성 앱 판별 기법

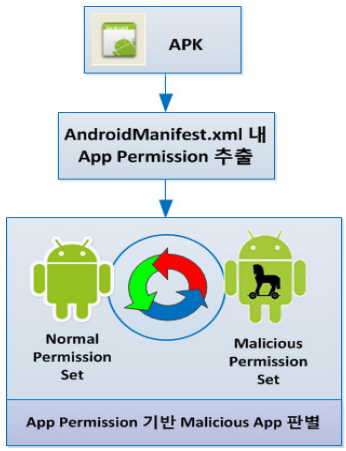
사용자가 임의의 앱을 다운받아 설치했을 경우 해당 앱이 악성인지 아닌지는 판별하기는 어렵다. 본 논문에서는 설치한 앱을 퍼미션 중심으로 판별하는 방법을 그림 14와 같이 제시하였다.



(그림 12) 악성 앱 시스템 콜 이벤트 (상: 주요 임의의 악성 앱, 하: 유사 기능 악성 앱)
(Figure 12) Malicious App System Call Event (Up: Major Malicious App , Down: Similar Function)



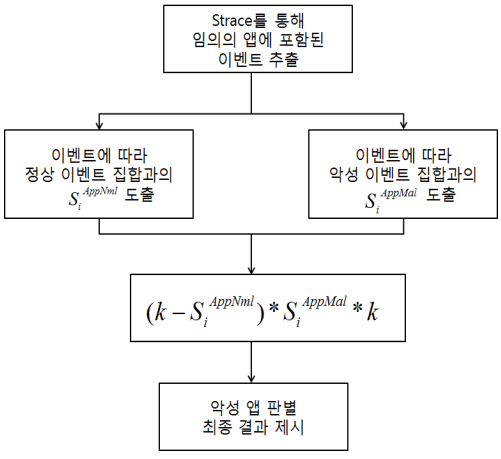
(그림 13) 정상 및 악성 어플리케이션 시스템 콜 이벤트 비교 분석 그래프
(Figure 13) System Call Event Comparison Graph between Normal and Malicious App



(그림 14) 이벤트 기반 악성 앱 판별 방법
(Figure 14) Malicious App Decision Method

사용자가 설치한 어플리케이션이 악성인지 정상인지를 판별하기 위해서는 모바일 단말 내 어플리케이션의 apk 파일을 분석하기 위해 apk를 압축해제해서 어셈블리 코드를 얻는다. 압축 해제한 폴더 내 AndroidManifest.xml에서 어플리케이션의 퍼미션을 추출한 뒤 정상 퍼미션 집합과 악성 퍼미션 집합을 중심으로 분석하면 해당 어플리케이션이 정상인지 악성인지를 판별할 수 있다.

임의의 앱에 포함된 퍼미션을 대상으로 정상 앱과 악성 앱의 이벤트 집합과의 유사도를 통해 악성 앱을 판별하는 알고리즘은 다음 15과 같다.



(그림 15) 이벤트 유사도를 통한 악성 앱 판별 알고리즘
(Figure 15) Event Similarity based Malicious App Decition Algorithm

임의의 앱을 설치한 후 해당 앱에 포함된 퍼미션을 획득한 뒤 Strace를 통해 이벤트를 추출해서 정상 이벤트 집합과 악성 이벤트 집합과의 유사도를 각각 도출한다. 이때 정상 이벤트 집합과 악성 이벤트 집합은 매 번 변경될 수 있다.

본 논문에서는 32개의 정상 이벤트 집합과 36개의 악성 이벤트 집합을 대상으로 정상 앱과 악성 앱에서 분석했던 앱을 각 5개씩 임의로 선택하여 정상 이벤트 집합과 일치하는 정상 앱의 이벤트 및 악성 앱의 이벤트, 악성 이벤트 집합과 일치하는 정상 앱의 이벤트 및 악성 앱의 이벤트를 분류하였다. 이를 통해 S_i^{AppNml}(정상 유사도) 및 S_i^{AppMal}(악성 유사도)의 특성을 시스템 콜 이벤트 정보를 중심으로 도출할 수 있었고 $(k - S_i^{AppNml}) * S_i^{AppMal} * k (i = 1,2,..,n)$의 수식으로 정상 및 악성 앱의 최종 판별 결과를 제시한다. 이때 S(유사도)는 0부터 1 사이의 값을 가지며 k는 이벤트의 가중치를 나타낸다.

정상 이벤트 집합은 clock_gettime, epoll_wait, getcwd, poll 등 32개로 앞서 살펴봤던 정상 앱에서만 나타나는 이벤트들과 정상과 악성 앱에서 모두 발생하는 이벤트 중 상대적으로 정상 앱에서 더 많이 나타나는 이벤트로 되어있다. 반면 악성 이벤트 집합은 bind, pwrite, rename, unlink 등 36개로 마찬가지로 앞서 살펴봤던 악성 앱에서만 존재하는 이벤트들과 정상과 악성 앱에서 모두 발생하는 이벤트 중 상대적으로 악성 앱에서 더 많이 나타나는 이벤트로 분류하였다.

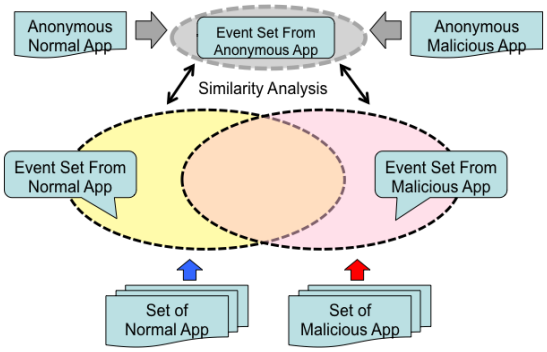
S_i^{AppMal}는 정상 이벤트 집합과 일치하는 정상 및 악성 앱의 이벤트 수/정상 이벤트 집합 수로 정상 앱이 악성 앱보다 상대적으로 높은 수치를 나타내고 있는 것을 확인할 수 있다. 그 이유는 정상 이벤트 집합이 정상 앱에서만 나타나는 이벤트와 정상과 악성 모두 발생하는 이벤트이지만 그 빈도수가 정상 앱에 더 많이 나타나기 때문이다. 반면 S_i^{AppMal}는 악성 이벤트 집합과 일치하는 정상 및 악성 이벤트 수/악성 앱의 이벤트 수로 악성 앱에만 존재하거나 악의적인 가진 이벤트들의 집합이기 때문에 상대적으로 정상 앱보다는 악성 앱에 정상 앱보다 훨씬 높은 수치로 나타나는 것을 알 수 있었다.

지금까지 분석한 정상 및 악성 유사도를 통해 악성 앱을 판별하는 유사도를 도출하기 위해 악성 이벤트 집합 중에서 악의적인 기능을 갖는 chmod, unlink, fchown32 등 14개의 이벤트에 k의 가중치를 부여해서 유사도 수식을 통해 임의의 정상 및 악성 앱에 대해서 악성 앱 판별을 할 수 있도록 제시하였다. 그림 15에서는 악성 이벤트의

집합을 많이 포함하고 있는 악성 앱의 이벤트가 정상 앱의 이벤트보다 더 많기 때문에 k만큼의 가중치를 곱한 S_i^{AppMal}는 상대적으로 S_i^{AppNml}보다 훨씬 큰 값을 갖게 된다. 이와 같이 임의의 악성 앱에 대해 악성 앱 판별 유사도가 높은 수치로 나타났기 때문에 이러한 수식을 다른 모바일 환경 내 임의의 앱에서 적용했을 때도 위와 같은 결과가 나올 것이라고 예상할 수 있다.

5.3 정상 및 악성 앱 이벤트 중심 유사도 추출

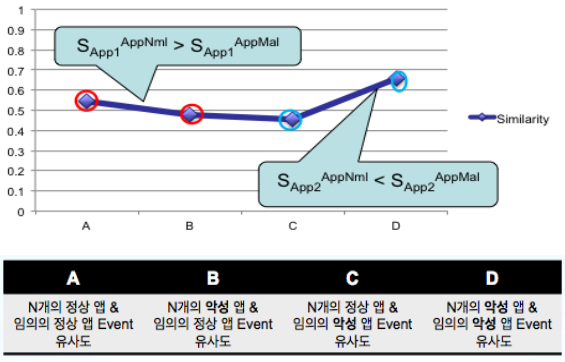
앞에서 제시한 바와 같이 안드로이드 플랫폼에서 구동되는 정상 및 악성 앱에 대한 판별을 위해서 본 논문에서는 단말내에서 실시간으로 앱을 구동하여 실행되는 시스템 콜 이벤트를 수집 및 분석하는 방법을 사용하였다. 이제 임의의 앱을 대상으로 정상 앱인지 아니면 악성 앱인지를 판별하는 과정을 구조화하면 다음 그림 16과 같다. 우선 안드로이드 마켓을 통해 다수의 정상 앱과 악성 앱 집합을 구축한다. 이제 전처리 과정으로 두 집합에 속한 앱을 단말에서 실행하여 각각 정상/악성 이벤트 집합 DB를 구축한다. 이제 마지막 유사도 분석 및 추출 단계로 임의의 앱 실행시 발생하는 이벤트 집합과의 유사도 측정 과정을 수행하여 해당 앱에 대한 악성 여부를 판별하게 된다.



(그림 16) 시스템 콜 이벤트 기반 유사도 분석 구조도
(Figure 16) Structure of Similarity Analysis Structure with System Call Event

아래 그림 17은 임의의 정상 앱 또는 임의의 악성 앱을 대상으로 실험을 통해 구축된 정상 및 악성 시스템 콜 이벤트 집합 DB와의 유사도 측정을 통해 최종 판별하는 과정을 보인다. 임의의 앱 *App_1*과 *App_2*에 대해 각각 정상 이벤트 집합 및 악성 이벤트 집합과의 유사도 분석 과정을 수행하였을 경우 결과를 제시하고 있다. 임의의 *App_1*에서 발생하는 이벤트에 대한 유사도 추출 과정을 수행하면 정상 유사도(S_i^{AppNml})가 악성 유사도(S_i^{AppMal})보다 큰 값을 보이기 때문에 결국 *App_1*은 정상 앱으로 판별하게 된다.

그러나, 임의의 *App_2*인 경우 악성 유사도가 정상 유사도 보다 큰 값을 갖는 것으로 분석되었기 때문에 최종적으로 *App_2*는 악성 앱으로 판별하게 된다.



(그림 17) 정상 및 악성 이벤트 중심 유사도 분석
(Figure 17) Similarity Evaluation with Normal and Malicious Events

6. 결 론

안드로이드 기반 어플리케이션은 안드로이드의 개방적인 특성에 따라서 오픈 소스(open source) 형태로 개발이 가능하다. 따라서 안드로이드 공식 마켓과 인터넷, 블랙마켓 등 여러 유통 경로를 통해 정상 앱인 것처럼 위장하여 악성코드를 삽입한 형태로 배포가 가능하다. 따라서 안드로이드 플랫폼을 기반으로 하는 상용 모바일 단말인 경우 안드로이드 단말 자체 뿐만 아니라 악성 앱을 통한 공격 위협에 노출되어 있다.

본 논문에서는 안드로이드 기반 상용 모바일 단말 환경에서 구글 플레이 스토어 내 게임 카테고리에서 인기 있는 상위 80개의 정상 게임 어플리케이션과 Android MalGenome Project에서 배포받은 1260개의 악성 샘플들을 대상으로 각각 분류하여 정상 및 악성 앱의 특징 및 이벤트를 분석하고 이벤트를 기반으로 해서 악성 앱을 효율적으로 판별하기 위한 기법을 제시하였다. 우선 최신 안드로이드 단말 위협 및 악성 앱의 공격 형태를 분석하였다. 또한 안드로이드 커널에서 시스템 콜 이벤트를

수집할 수 있는 Strace 모듈을 이용하여 정상 및 악성 앱에서 각각 발생하는 시스템 콜 이벤트 특성과 이를 비교 분석하였다. 이를 토대로 본 논문에서는 안드로이드 기반 상용 모바일 단말 내 정상 앱과 악성 앱에서 발생한 이벤트에 대한 유사도 분석 알고리즘을 이용하여 악성 앱을 판별할 수 있는 알고리즘을 제시하였다.

본 연구에서 제시한 기법은 정상 앱과 악성 앱에서 Strace를 통해 추출한 시스템 콜 이벤트를 바탕으로 시퀀스 분석을 통해 악성 앱에서는 공통적으로 존재하면서 빈도수가 높고 상대적으로 정상 앱에서는 빈도수가 낮은 시스템 함수를 분류해서 이벤트 기반의 악성 앱 판별 유사도를 나타내는 기법으로 이를 통해 정상 앱과 악성 앱 실행 시 발생하는 시스템 콜 이벤트의 특징과 일정한 패턴의 시스템 콜 함수 시퀀스가 존재하는 것을 확인할 수 있었다. 뿐만 아니라 이러한 기법을 이용할 경우 임의의 모바일 앱에 대해 악성 여부를 판단하는 방법에 적용될 수 있다. 또한, 악성 앱 판별 유사도 식을 통해 사용자가 임의의 앱을 설치하였을 때 해당 앱이 악성인지 아닌지를 효율적으로 판별할 수 있는 기준을 제시하였다.

향후 연구 과제로는 본 연구에서 제시한 기법을 보다 많은 형태의 모바일 앱에 적용하여 임의의 앱에 대한 악성 여부 판별의 정확성을 높이는 것이며 이를 통해 보다 안전한 상용 모바일 단말 사용자 환경을 제공하는 것이다.

참 고 문 헌(Reference)

[1] Woogryul Jeon, Jeeyeon Kim, Youngsook Lee, Dongho Won, "Analysis of Threats and Countermeasures on Mobile Smartphone," Journal of The Korea Society of Computer and Information, Vol. 16, No. 2, pp.153-163, 2011.

[2] W. Enck, M. Ongtang, P. McDaniel, "Understanding android security," IEEE Security & Privacy Magazine, Vol. 7, No. 1, pp. 50-57, 2009.

[3] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, "Google Android: A State-of-the-art Review of Security Mechanisms," Technical Report, Cornell University, 2009.

[4] 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, pp.30-38, 2012.

[5] Jae-woo Park, Sung-tae Moon, Gi-Wook Son, In-Kyoung Kim, Kyoung-Soo Han, Eul-Gyu Im, Il-Gon Kim, "An Automatic Malware Classification System using String List and APIs," Journal of Security Engineering, Vol.8, No.5, pp.611-626, 2011.

[6] More than 50 Android apps found infected with rootkit malware, http://www.guardian.co.uk/technology/blog/2011/mar/02/android-market-apps-malware.

[7] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, Y. Weiss, "Andromaly: a behavioral malware detection framework for android devices," Journal of Intelligent Information Systems, Vol. 38, No. 1, pp. 161-190, 2012.

[8] Y. Zhong, H. Yamaki, H. Takakura, "A Malware Classification method Based on Similarity of Function Structure," 12th International Symposium of Applications and the Internet(SAINT), pp.256-261, 2012.

[9] Yajin Zhou, Xuxian Jiang, "Dissecting Android Malware: Characterization and Evolution", 2012.

[10] L.Tenenboim-Chekina, O. Barad, A. Shabtai, D. Mimran, L.Rokach, B. Shapira, Y. Elovici, "Detecting Application Update Attack on Mobile Devices through Network Features", 2013.

◐ 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