https://doi.org/10.5392/JKCA.2018.18.04.305

안드로이드 환경에서의 효과적인 악성코드 탐지 메커니즘

An Effective Malware Detection Mechanism in Android Environment

김의탁*, 류근호**

충북대학교 전자계산학과*, 충북대학교 전기전자정보컴퓨터학부**

Eui Tak Kim(kingket9@hanmail.net)*, Keun Ho Ryu(khryu@chungbuk.ac.kr)**

요약

스마트폰의 폭발적인 증가와 효율성으로 개방형 모바일 운영체제인 안드로이드의 활용도가 점차 증가하고 있고, 모바일 기기, 가전제품의 운영체제, IoT 관련 제품들과 더불어 메카트로닉스의 분야에도 활용될 수 있는 가용성과 안정성이 증명되고 있다. 하지만, 사용성이 증가하면 증가할수록 안드로이드 기반의 악성코드 역시 기하급수적으로 증가하고 있는 추세이다. 일반 PC와 다르게 모바일 제품에 악성코드가 유일될 경우, 모바일 기기가 Lock됨으로 사용할 수 없고, 불필요한 과금과 더불어 수많은 개인의 연락처가 외부로 유출될 수 있으며, 모바일 기기를 활용한 금융서비스를 통해 막대한 손실을 볼 수 있는 문제점이 있다. 따라서, 우리는 이 문제를 해결하기 위하여 유해한 악성 파일을 실시간으로 탐지 및 삭제할 수 있는 방법을 제시하였다. 또한 이 논문에서는 안드로이드 기반의 어플리케이션 설치 과정 및 시그니처 기반 악성코드 탐지 방법을 통해 보다 효과적인 방법으로 악성코드를 실시간 감시하고 삭제할 수 있는 기법을 설계하였다. 우리가 제안하고 설계한 방법은 모바일 환경과 같이 제한적인 리소스 환경에서 악성코드를 효과적으로 탐지할 수 있다.

■ 중심어 : | 안드로이드 | 안티바이러스 | 악성코드 | 탐지 메커니즘 | 시그니처 |

Abstract

With the explosive growth of smart phones and efficiency, the Android of an open mobile operating system is gradually increasing in the use and the availability. Android systems has proven its availability and stability in the mobile devices, the home appliances's operating systems, the IoT products, and the mechatronics. However, as the usability increases, the malicious code based on Android also increases exponentially. Unlike ordinary PCs, if malicious codes are infiltrated into mobile products, mobile devices can not be used as a lock and can be leaked a large number of personal contacts, and can be lead to unnecessary billing, and can be cause a huge loss of financial services. Therefore, we proposed a method to detect and delete malicious files in real time in order to solve this problem. In this paper, we also designed a method to detect and delete malicious codes in a more effective manner through the process of installing Android-based applications and signature-based malicious code detection method. The method we proposed and designed can effectively detect malicious code in a limited resource environment, such as mobile environments.

■ keyword : | Android | Anti-Virus | Malware | Detection Mechanism | Signature |

* 이 논문은 2017년도 충북대학교 연구년제사업의 연구비 지원에 의하여 이루어진 연구임.

접수일자 : 2018년 03월 05일 심사완료일 : 2018년 03월 26일

수정일자 : 2018년 03월 25일 교신저자 : 류근호, e-mail : khryu@chungbuk.ac.kr

I. 서 론

최근 스마트폰을 비롯한 모바일 기기 시장의 급격한 확대로 2017년도 국내 스마트폰 사용자는 총 인구수의 87.8%를 차지하고[1], 이 중 안드로이드 사용자는 74.5%를 차지하고 있으며[2], 2018년도 1사분기 전 세계 스마트폰 운영체제 시장 점유율 중 안드로이드가 차지하는 비중은 85.0%에 달한다[3][4]. 이처럼 개방형 모바일 운영체제에 대한 활용도가 높아지고 있고, 개방형 모바일 운영체제 중, 안드로이드 기반의 운영체제는 구글의 지원으로 가전제품, IoT 제품 및 다양한 분야로 확산되고 있으며, 이러한 추세를 반영하듯, 안드로이드 기반 악성코드 또한 폭발적인 수치로 늘어나고 있는 것으로 파악되고 있다[5]. 안드로이드 기반 악성코드가 계속적으로 증가하는 이유는 첫째, 가장 많이 판매된 모바일 운영체제이기 때문에 악성코드 배포가 쉽다. 둘째, 안드로이드 마켓은 앱 등록시 검증을 받지 않는 구조로 되어 있어, 운영체제 내에서 실행되는 다양한 어플리케이션들 사이에 신뢰를 구축하기 위해 개발자 인증을 사용하지만, 제3자가 개발자의 허가를 받지 않고 키를 취할 경우 악의적으로 사용이 가능하다[6]. 셋째, 안드로이드 운영체제는 이식성이 높은 자바로 구현되어 편리함이 있으나, 역공학(Reverse Engineering)이 쉽기 때문에 어플리케이션을 분석 및 위변조한 뒤 악성코드 삽입이 가능하다[7]. 넷째, 악성 앱이 설치되어 있어도 사용자가 인지하지 못하는데 있다. 실제 악성 앱이 설치되어도 사용자는 악성 앱임을 의심하기 힘들며, 특별한 증상이 나타나지 않는다. 이외에, 안드로이드가 개방형 플랫폼이라는 것도 또 다른 이유가 될 수 있다[8].

이 같은 이유들로, 안드로이드 기반의 모바일 제품군에 대한 악성코드 탐지 및 실시간 감시 기능이 탑재된 보안제품군의 사용은 필수적인 조건으로 보인다. 안드로이드 스마트폰 사용자들은 신뢰할 수 없는 사이트로부터 앱을 다운받아 설치하는 행위를 자제하고, 악성코드의 감염 및 해킹에 악용될 수 있는 루팅(Rooting)을 하지 않으며, 보안취약점 점검 기능이 제공되는 스마트폰 보안 프로그램을 이용하여 보안 취약점 점검을 수행해야 한다. 또한, 보안 취약점이 해결된 최신 버전의 운영체제로 업데이트를 해야만 안전한 모바일 환경을 유지할 수 있다.

따라서, 우리는 이 문제를 해결하고 효율적인 안전한 모바일 환경을 유지하기 위하여, 유해한 악성 파일을 실시간으로 탐지 및 삭제할 수 있는 방법, 즉 안드로이드 기반의 모바일 환경에서 classes. dex를 이용한 시그니처 추출 및 앱의 설치 시점을 파악하여 실시간으로 악성 여부를 판단 및 삭제할 수 있는 메커니즘을 제시한다.

우리가 제안하고 설계한 방법은 모바일 환경과 같이 제한적인 리소스 환경에서 악성코드를 효과적으로 탐지할 수 있다. 이 논문을 효율적으로 전개하기 위하여 제2장에서는 기존의 모바일 환경에서 일반적인 안드로이드 앱의 설치 방법과 기존의 안드로이드 기반 악성코드 탐지 기법을 관련 연구로서 설명을 한다. 그리고 제3장에서는 우리가 제안하는 악성코드 탐지 메커니즘과 탐지 과정을 자세하게 설계 기술하고 제안하는 메커니즘을 평가하기 위하여 제안 방법의 장단점을 기술한다.

II. 관련 연구

2.1 안드로이드 기반 악성코드

안드로이드는 플랫폼은 기존 리눅스 운영체제의 커널을 사용하고 있기 때문에 리눅스에서 갖고 있는 취약점부터 안드로이드 플랫폼에 대한 취약점까지 모두 갖고 있다. 또한 안드로이드 구조 상 펌웨어 업그레이드를 해야 취약점을 제거할 수 있기 때문에 정기적인 보안 업데이트가 사실상 어려운 현실이다[9][10].

안드로이드에서 동작하는 악성코드는 모바일 기기에 다양한 피해를 주는 프로그램으로 최근 들어 랜섬웨어(Ransomware)가 급증하고 있는 추세이다. 대표적인 악성코드 프로그램은 Backdoor, Trojan, Exploit, Spyware, HackTool 등이 있다[11][12]. Backdoor는 공격자의 명령을 받아 모바일 기기에서의 악의적인 행위를 실행하는 악성코드이고, Trojan은 시스템 내에서 개인정보 및 다양한 정보를 외부 C&C로 보내는 악성코드, Exploit은 모바일 기기의 권한 상승을 획득하여 각

종 악성코드를 배포하거나 실행할 수 있는 환경을 제공하는 악성코드이고, Spyware는 사용자 동의 없이 개인정보를 수집하고 탈취하는 악성코드이며, HackTool은 모바일 기기를 악의적으로 사용할 수 있도록 한 해킹 툴이다[13]. 안드로이드 기반의 악성코드는 PC에서 사용되던 악성코의 특징을 따라가는 추세이고, 추가적으로 프리미엄 SMS 사용, 국제전화 도용 등을 수행함으로 사용자로 하여금 불필요한 과금을 할 수 있도록 하는 추가 기능도 제공한다[14]. 현재는 악성코드 제작자들이 모바일용 랜섬웨어를 많이 제작하여 유포하고 있어, 해당 악성코드에 감염될 경우, 스마트폰의 화면이 잠기거나 특정 확장자의 파일을 암호화함으로써 다양한 피해를 초래하고 있다.

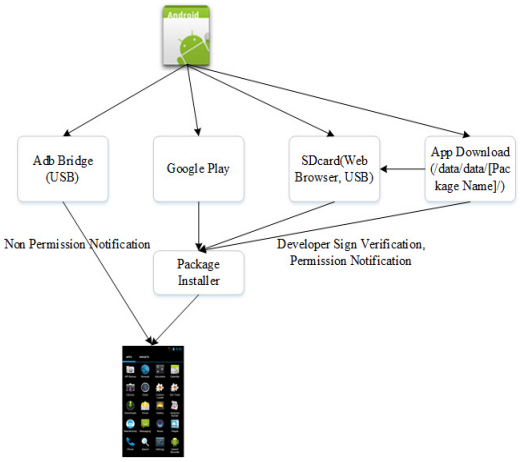

그림 1. 안드로이드 앱 설치 방법

2.2 안드로이드 APK 설치 과정

일반적인 안드로이드 앱의 설치 방법은 안드로이드 마켓, SDcard를 통한 설치와 Android Debug Bridge (ADB)를 이용하여 설치하는 방법이 있다. 웹브라우저를 통하여 설치를 하는 방법도 있지만, 결국 SDcard에 다운로드 받은 후 설치함으로 SDcard를 통한 설치와 같은 방법이라고 할 수 있다. 특이한 방법으로는 앱 자체에서 다운로드 기능을 구현하여 APK를 다운로드 하여 설치하는 방법도 있다. 이 경우는 앱 자신의 디렉터리에 파일을 저장하거나 SDcard에 저장을 한 후, Package Installer을 통해 설치를 진행하게 된다. ADB를 통하여 안드로이드 장치가 PC와 USB로 연결이 된 상태에서 “adb install file.apk” 형태의 명령옵션을 이용하여 설치한다[14][15]. [그림 1]은 안드로이드 기반 앱 설치 방법이다.

[그림 1]에서 안드로이드 마켓을 통하여 앱을 설치할 경우, 안드로이드 시스템의 기본 설정상태에서 설치할 수 있다. 하지만, SDcard와 앱 자체 다운로드의 경우에는 “Settings -> Applications ->Unknown sources” 옵션이 체크되어 기능이 활성화 되어야 한다. ADB를 통하여 앱을 설치하기 위해서는 “Settings -> Applications -> Development -> USB debugging” 옵션이 활성화되어 있어야 한다[9][16].

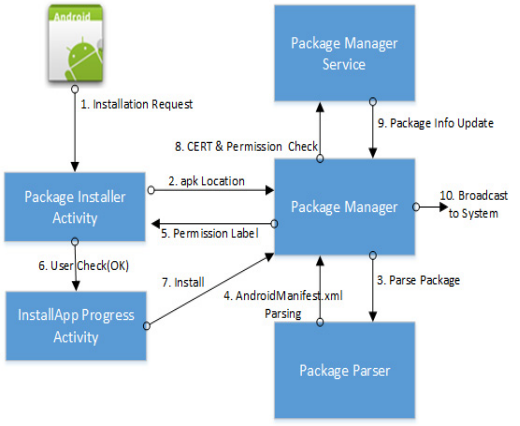

그림 2. APK 설치 과정

[그림 2]는 [그림 1]의 Package Installer[17][18] 부분을 보다 상세하게 나타내는 그림이다. 그림에 표시된 번호순서에 따라 APK 설치가 진행 된다.

2.3 시그니처 기반 악성코드 탐지

안드로이드 기반 악성코드 탐지 기법에는 대표적으로 시그니처 기반 탐지 기법을 사용한다. 시그니처 기반 탐지 기법은 악성코드로 분류된 파일의 특정 부분 또는 고유한 부분을 검사의 대상으로 하기 때문에 오탐(False Positive)과 미탐(False Negative)을 최소화하는 정확한 진단이 가능하다는 것과 파일 검사 시에 파일들

의 특징적인 부분들만 비교하기 때문에 빠른 스캐닝을 장점으로 들 수 있다[19]. 하지만 악성코드의 파일이 일부분만 바뀌어도 진단이 되지 않는 미탐이 발생함으로 새로운 형태의 알려지지 않는 악성코드에 대해서는 대응할 수 없고, 시그니처의 수가 많아지면, 시그니처의 업데이트 시간과 스캔 시간이 길어질 수 밖에 없는 단점을 가지고 있다[20]. 시그니처 기반 탐지 기법은 현재 PC, 서버 이외에 모바일 환경에서도 가장 많이 사용하는 방법으로 안전성과 효과성이 입증되어 안드로이드 기반 백신에서 필수적으로 사용하고 있는 방법이다[14].

시그니처 기반 탐지 기법에서 효과적으로 시그니처를 생성하여 비교를 통해 악성 어플리케이션을 찾아내는 것이 중요하다. 시그니처를 생성 및 비교하는 방법은 보통 APK 파일명을 비교하는 방법, APK 파일 내, 특정 스트링을 비교하는 방법이 주로 사용된다[14]. APK 전체 사용하는 방법은 비효율적인 방법이므로 대부분 일부의 특징적인 부분만 사용하여 시그니처를 생성하는 방법을 사용한다[20]. 하지만, 이러한 방법은 모든 APK 파일이 모바일 기기에 유입될 때마다 실시간으로 확인 후, 설치해야 하는 불편함이 있고, 이에 따른 과도한 탐지 모듈의 동작으로 리소스에 대한 영향을 줄 수 있다[21][22].

2.4 동적분석 기반 악성코드 탐지

시그니처 기반 악성코드 탐지 방법 이외에 동적분석 탐지 방법이 많이 사용된다. 동적분석 탐지 방법은 안드로이드 디버거(Debugger)나 에뮬레이터(Emulator)를 이용하여 프로그램을 수행시킴으로써 해당 어플리케이션의 행위를 분석하여 악성 여부를 판별하는 방식이다. 그러나, 모바일기기 상에서 디버거를 이용한 분석은 모바일기기에 발생 가능한 피해 외에도 디버거 탐지 및 우회 로직을 포함한 악성코드들을 분석하지 못하는 단점을 갖는다[22][23]. 이러한 이유로, PC 기반 악성코드의 동적 분석과 같이 안드로이 기반 악성코드의 동적 분석 역시 가상 머신이 활용되고 있으며, Honeynet Project[24]와 DroidScope[17]과 같은 가상 머신상에서 안드로이드 어플리케이션의 동적 분석을 돕기 위한 도구들이 개발되고 있다.

동적분석 방법은 악성코드의 특징 및 시그니처를 추출하기 위해 많이 이용되고 있으며, 코드 난독화가 수행된 악성코드 역시 분석 가능하다는 장점을 갖는다[19]. 그러나 모든 가능한 프로그램 실행 패스를 다 분석하지 못하는 문제점을 갖고 있다. 실제로 초기 동적분석 방법은 단지 하나의 프로그램 실행 패스만을 분석할 수 있었으며, 현재 동적분석 범위를 높이기 위한 목적으로 다중 실행 패스 분석 기법들이 연구되고 있다[22][25].

III. 악성코드 탐지방법 제안 및 평가

3.1 탐지방법 제안

APK 파일은 어플리케이션 기반 설치 파일로, 압축되어 있으며, META-INF, res, Android Manifest.xml, classes.dex, resources.arsc로 구성되어 있고, 이 중 실행파일은 classes.dex 파일이다. 이 파일은 최종 컴파일된 바이너리 파일로, 이 파일에 대한 해쉬 값을 추출하여 시그니처로 활용함으로써 특정 APK 파일임을 비교 및 탐지가 가능하다[17][26].

표 1. dex 파일 헤더 정보

Name	Format	Description
magic	ubyte[8] =DEX_FILE_MAGIC	magic value. See discussion above under "DEX_FILE_MAGIC" for more details.
checksum	uint	adler32 checksum of the rest of the file (everything but magic and this field); used to detect file corruption
signature	uByte[20]	SHA-1 signature (hash) of the rest of the file (everything but magic, checksum, and this field); used to uniquely identify files
file_size	uint	size of the entire file (including the header), in bytes
header_ size	uint = 0x70	size of the header (this entire section), in bytes. This allows for at least a limited amount of backwards/forwards compatibility without invalidating the format.

또한, classes.dex는 자체 파일 포맷을 가지고 있고, 파일 헤더에 dex에 대한 고유한 정보 및 파일 생성 시점의 dex에 대한 SHA1 시그니처를 생성하여 포함하고 있다. [표 1]은 dex 파일의 헤더 정보를 나타내고, [그림 3]은 dex 파일의 헤더 정보 중, “magic” 값을 나타낸다. “magic” 값은 해당 파일이 dex라는 것을 구분할 수 있는 값이다[17].

```
ubyte[8] DEX_FILE_MAGIC = { 0x64 0x65 0x78 0x0a 0x30 0x33 0x35 0x00 }
                        = "dex\n035\0"
```

그림 3. DEX_FILE_MAGIC

본 논문에서는 APK 파일을 일부만 사용하는 방법 중, APK 파일 구조를 통해 효과적으로 악성 APK를 탐지하고자, APK 파일 내의 classes.dex 파일을 이용한 시그니처 생성을 제안한다. 해당 파일로 시그니처를 생성할 경우, 앱 설치시 반드시 사용되는 Package Installer에서 앱의 권한 획득을 위해 사용하는 package manager를 호출하고, package manager는 할당된 APK 파일에 대한 권한을 체크하게 되는데, 만일, 이 시점에서 classes.dex 파일을 통해 생성된 악성코드 시그니처와 설치되는 APK의 classes.dex 파일을 비교함으로써 손쉽게 실시간 감시기 구현 및 악성코드 탐지를 수행할 수 있다. 모바일 환경과 같이 제한적인 리소스 상황에서 악성코드를 탐지하기 위해 탐지 모듈이 모든 파일과 행위를 감시한다는 것은 리소스 낭비가 심화되고, 사용자로 하여금 불편함을 초래할 수 있다. 따라서, 악성코드 탐지 메커니즘 설계시 악성코드 파일이 유입될 수 있는 경로를 파악하여 실시간 감시를 통해 악성코드 파일 설치 여부를 판단 및 탐지하는 것이 보다 효과적인 방법이라 할 수 있다. 앞 장에서 살펴본 바와 같이 안드로이드 기반의 앱 설치시 대부분의 파일은 Package Installer와 ADB를 통해 설치된다. 이처럼 앱이 설치되는 포인트를 classes.dex를 이용한 시그니처를 생성하여 실시간 감시함으로써 효과적으로 악성코드를 탐지가 가능하다. [표 2]는 기존 패턴 생성 방식과 제안하는 방식에 대한 차이점에 대한 비교이다.

표 2. 기존 패턴 생성 방식과 제안하는 방식 비교

Division	Traditional	Proposed
Pattern Compare	APK Hash	class.dex Hash(Suggested)
Pattern Size	8byte	8byte
Detection Time	Installing Package	Installing Package, Mounting SDCard, Creating SDCard File Modifying SDCard File
Detection Target	Installed Package, *.apk File, Ordinary File, Zip File	Installed Package, *.apk File, Ordinary File, Zip File

3.2 악성코드 실시간 감시

악성코드를 실시간 탐지하기 위해서는 안드로이드 앱의 컴포넌트 중 하나인 브로드캐스트 리시버(Broadcast Receiver)를 이용한다. 패키지 설치 작업이 완료된 후 패키지 관리자(Package Manager)는 시스템에 패키지가 추가되었다는 메시지를 브로드캐스트 한다. 악성코드 탐지 모듈은 실행될 때 “android.intent.action.PACKAGE_ADDED” 에 해당하는 메시지를 받을 수 있도록 브로드캐스트 리시버를 등록한다. 이와 같은 일련의 작업으로 인해 악성코드 탐지 모듈은 시스템에 전달된 패키지가 추가되었다는 메시지를 받고, 메시지를 받은 시점에 해당 패키지의 정보를 추출한 후 악성코드 시그니처를 이용하여 검사를 실시한다.

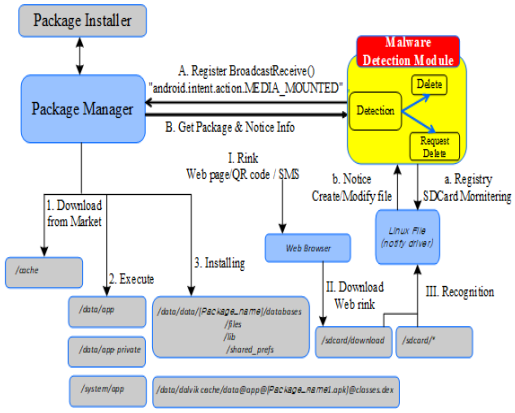

그림 4. 안드로이드 기반 악성코드 실시간 감시

이외, SDcard에 악성코드가 유입되었을 경우도 반드시 실시간 감시를 해야한다. SDcard 내에 검사대상은

APK, 일반파일, zip 파일이다. SDcard는 [표 2]에 나열된 퍼미션을 사용하여 접근할 수 있고, PC와의 네트워크나 USB 연결을 통해 APK나 파일이 유입될 수 있으므로 중요한 감시지점 중 하나이다.

SDcard가 안드로이드 시스템에 마운트 되어 있는 상태에서는 네트워크를 통하거나 로컬에서 앱이나 기타 프로그램에 의해 파일이 생성될 수 있다. 이러한 파일에서 발생할 수 있는 이벤트는 생성(Create), Delete(삭제), 수정(Modify), 쓰기(Write)로 안드로이드 SDK (Software Development Kit)인 "android.os.File Observer" Java API 또는 NDK(Native Development Kit)에서 제공하는 "inotify - monitoring file system events" C API를 통해 이벤트 발생 시점을 받을 수 있도록 하여 해당 시점에 이벤트가 발생한 파일에 대해 검사를 수행할 수 있다.

SDcard를 PC와 USB로 연결하여 안드로이드 시스템에 마운트할 때 마다 발생하는"android.intent.action.MEDIA_MOUNTED" 메시지를 브로드캐스트 리시브에 등록하여 해당 메시지를 받은 시점에, 악성코드 탐지 모듈이 SDcard에 대한 전체 검사를 실시하여 SDcard 내에 악성코드의 침입여부를 효과적으로 감시할 수 있다.

악성코드 실시간 감시와 관련된 코드는 아래 [표 3]과 같다.

표 3. 악성코드 실시간 감시 관련 코드

```
// SDCard 감시 시작
if (!Utility.amGetSDCardScanSetup(mContext)) {
      intentFilter = new IntentFilter();
      intentFilter.addAction(Intent.ACTION_MEDIA_MOUNTED);
       intentFilter.addAction(Intent.ACTION_MEDIA_UNMOUNTED);
      intentFilter.addDataScheme("file");
      mContext.registerReceiver(m_brSDCardMountWatching,
intentFilter);
       // SDCard 디렉토리 감시 시작
       if (Environment.getExternalStorageState().equals
       (Environment.MEDIA_MOUNTED)) {
      // Directory Watch Service Start
        realtimeDirectoryWatchStart();
       } else {
            Utility.amSetRealtimeServiceLock(mContext, false);
       }
} else {
     // 실시간 감시 Lock
      Utility.amSetRealtimeServiceLock(mContext, false);}
```

3.3 악성코드 삭제

안드로이드 운영체제는 샌드박스(Sandbox) 보안모델이 적용되어 있어, 권한이 없는 앱이 다른 앱에 접근하거나 시스템의 권한이 요구되는 행위를 할 수 없다. 이는 기본적으로 안전한 환경을 제공해주지만, 다른 앱의 삭제와 같은 필수적인 권한을 가질 수가 없어 악성앱을 탐지했을 경우 어플리케이션 레벨에서 해당 앱을 직접 삭제할 수 없다. 따라서, 안드로이드 SDK(Software Development Kit)에서 제공해주는 API (Application Programming Interface)를 통해 악성 앱이 탐지되었을 경우 이를 사용자에게 확인시킨 후, 해당 앱의 삭제를 유도하여 사용자가 직접 앱을 삭제하도록 해야 한다.

표 4. SDcard 내 파일을 핸들링하기 위한 퍼미션

Permission	Description
android.permission.READ_EXTERNAL_STORAGE	외장 저장소 읽기 허용
android.permission.WRITE_EXTERNAL_STORAGE	외장 저장소 쓰기 허용

안드로이드 시스템에 설치된 악성 프로그램은 직접적인 삭제가 불가능하지만, SDcard 내의 악성코드 탐지시 파일 삭제가 가능하다. SDcard 내에 검사대상은 APK, 일반파일, zip 파일이다. [표 4]는 SDcard 내의 파일을 삭제하기 위해 사용되는 퍼미션을 나열한 것이다.

[그림 5]는 악성코드의 삭제 과정을 나타내고 있다. 만약, 안드로이드에 설치된 악성 앱을 직접 삭제하기 위해서는 root 권한이 필요하다. 그러나, root 권한을 가질 수 있는 정상적인 방법은 단말 제조사를 통해 Built-in되어 "/system/app" 경로에 설치되거나 제조사에서 제공해 줄 수 있는 특별한 API를 사용할 수 있는 경우이고, 비정상적인 방법으로는 사용자가 시스템의 제약을 해제할 목적으로 시스템을 Unlock하는 루팅이 있다. 루팅이 된 시스템일 경우 앱은 root 권한을 가질 수 있지만 사용자가 알지 못하는 취약점이 발생할 수 있으므로 권장하지 않는다. 이외에 시스템을 강제 루팅하는 악성 앱과 같은 방법으로 시스템의 취약점을 이용하여 시스템을 루팅할 수 있지만 이 방법은 악성 앱과 다를 것이 없으므로 사용할 수 없는 방법이다.

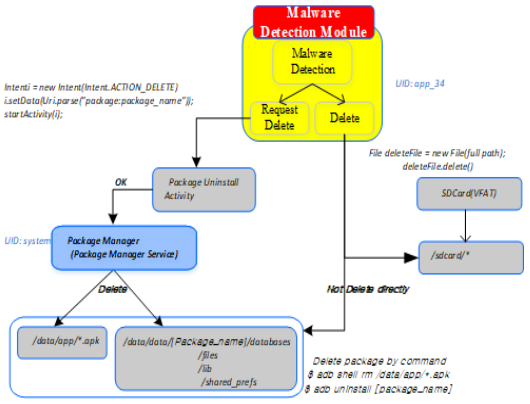

그림 5. 악성코드 삭제 과정

3.4 제안 방법의 평가

일반적으로 사용하는 APK 파일을 이용한 악성코드 탐지 패턴과 본 논문을 통해 제안하는 APK 파일에 대한 class.dex 파일을 통한 패턴 추출 방식에 대한 악성코드 탐지율에 대해 비교하였다.

탐지율 비교를 위한 테스트 샘플은 2017년 5월 23일자 AV-TES[27]에서 제공한 악성코드 샘플 2,339개를 활용하였다. AV-TEST에서 제공한 샘플은 Backdoor 388개, Trojan-SMS 1,803개, Trojan-Spy 127개, Trojan-Coinmine 10개, Trojan-Other 11개로 구성되어 있다. Backdoor 계열 샘플에는 사용자의 모바일기기를 원격제어가 가능한 Helir 악성코드, 프리미엄 SMS 서비스를 무단으로 사용가능한 Fobus 악성코드, 사용자 문자메시지를 감시하는 SMSSpy, 사용자 메일 환경을 이용한 Spambot 악성코드가 있고, Trojan 계열의 샘플에는 Boxer, Cova, Stealer, Androrat 등과 같이 원격제어와 더불어 전화번호, 문자 메시지, 각종 파일을 C&C 서버로 전송하는 악성코드가 있으며, Coinmine과 같이 전자화폐 채굴을 위한 악성코드로 안드로이드 기기의 자원을 소모시키는 기능이 포함되어 있다.

[표 5]에서 보는 바와 같이 악성코드 개수는 총 2,339개이고, 두 방식을 이용하여 탐지 및 미탐지 개수와 더불어 총 탐지율에 대해 실험해 보았다. 악성코드 2.339개는탐지율에 대한 비교 결과, 기존 APK 파일의 Hash 값을 이용한 탐지 패턴 대비, class.dex 파일의 파일 Hash 값을 활용한 탐지 패턴 생성시 각각 58.61%, 86.32%의 탐지율을 기록하였고, 제안하는 방식의 패턴 추출에 따른 탐지율이 27.71% 상향되었음을 확인할 수 있다.

표 5. 기존 방식과 제안하는 방식의 악성코드 탐지 성능

Division	APK Hash	class.dex Hash(Suggested)
Total File No.	2,339	
Detected File No.	1,317(58.61%)	2,019(86.32%)
Not Detected File No.	930(41.39%)	320(13.68%)
Detection Rate	58.61%	86.32%

Ⅳ. 결론

안드로이드는 오픈소스 정책으로 다양한 하드웨어와 단말제조사, 통신사들의 참여로 단시간 내에 스마트폰 시장에서 1위를 차지하고 있다. 하드웨어 성능이 PC와 비슷해짐에 따라 기업에서 업무용으로의 사용도 많아지고 있다. 이와 같이 사용자가 많아짐에 따라 안드로이드의 악성코드도 계속적인 증가 추세에 있고, 근래에 들어 악성코드 샘플 발생건수는 일일 평균 수천 건이 수집되고 있다. 이에 따라 안드로이드를 사용하는데 있어 안전한 환경을 제공할 필요가 있으며 사용자가 자각하지 못하는 부분을 악성코드 탐지 모듈을 통해 보완해 줄 수 있다.

본 논문에서는 APK 파일의 classes.dex 파일 정보를 이용하여 악성코드 시그니처를 추출하고, 악성코드가 유입되어 설치될 경우, 대부분의 파일은 Package Installer와 ADB를 통해 설치됨을 확인하였고, 해당 포인트에 대한 실시간 감시를 함으로써 악성코드 탐지를 보다 효율적으로 수행할 수 있음을 확인하였다. 또한, 실시간 감시를 통해 탐지된 악성코드를 삭제할 수 있는 방법들에 대한 방안에 대하여 기술하였다. 안드로이드 기반의 스마트폰 사용자 증가에 따라 안드로이드의 악성코드도 계속적인 증가함으로 우리가 제안한 기법은 안드로이드를 사용하는데 있어 안전한 환경을 제공할

수 있을 것을 기대한다. 아울러 사용자가 자각하지 못하는 부분을 악성코드 탐지 모듈을 통해 보완해줄 수 있을 것이라고 생각한다.

추가적으로, 본 연구와 더불어 악성코드 제작자들의 인증서를 활용하여 악성코드 탐지 패턴을 추가 제작할 경우, 탐지율은 보다 더 증가할 것으로 판단되며, 일일 수천 건이 넘는 악성코드 시그니처에 대한 축소 방안에 대한 부가적인 연구가 필요하다.

참 고 문 헌

[1] 한국인터넷진흥원, *2017년 인터넷이용실태조사 요약보고서,* p.10, 2018(1).

[2] https://www.koreahtm5.kr

[3] https://www.idc.com/

[4] http://www.gartner.com

[5] Roman Unuchek, "Mobile malware evolution 2016," Kaspersky lab, p.10, 2017(2).

[6] 신화윤, *모바일 애플리케이션 취약점 진단 도구 개발에 관한 연구,* 숭실대학교, 석사학위논문, pp.8-9, 2017(6).

[7] 유동훈, *모바일 스마트 플랫폼 원격, 로컬 취약점 공격 분석,* CodeEngn Conference 06, 2012.

[8] 양정수, 김은영, 김남국, *안드로이드 입문서,* 2010(2).

[9] 이우진, *안드로이드 뱅킹 어플리케이션의 안티바이러스 적용에 대한 개선방안 연구 : 안티바이러스 우회 취약점 중심으로,* 고려대학교, 석사학위논문, p.22, 2017(12).

[10] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically Rich Application-Centric Security in Android," 2008.

[11] Aubrey-Derrick Schmidt, Hans-Gunther Schmidt, Kamer Ali Yüksel, Osman Kiraz, Dr. Seyit Ahmet Camptepe and Prof. Dr. Sahin Albayrak, "Enhancing Security of Linux-based Android Devices," 2008.

[12] Jesse Burns, "DEVELOPING SECURE MOBILE APPLICATIONS FOR ANDROID", Version 1.0, 2008(10).

[13] Junuper Networks, *2011 Mobile Threats Report,* 2012(2).

[14] Korea Internet & Security Agency 연구보고서, *안드로이드 기반 모바일운영체제 보안기능 분석,* 2010(8).

[15] 김대청, *안드로이드 기반의 모바일 오피스 환경에서 앱 설치 제어 방법,* 충남대학교, 석사학위논문, pp.14-17, 2010(6).

[16] http://www.kandroid.org

[17] http://developer.android.com

[18] http://source.android.com

[19] 배근태, 김기영, "모바일 단말 보안 운영체제 기술 동향," 전자통신동향분석, 제23권, 제4호, pp.39-47, 2008(8).

[20] A. Shabtai, Y. Fledel, Y. Elovici, and Ben-Gurion University, "Securing Android-Powered Mobile Devices Using SELinux," IEEE Security & Privacy Magazine, Vol.8, No.3, pp.36-44, May/June, 2010.

[21] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and Ben-Gurion University, "Google Android: A State-of-the-Art Review of Security Mechanisms," 2009(12).

[22] L. K. Yan and H. Yin, "DroidScope:Seamlessly Reconstruction the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis," Proc 21 USENIX conf. Security Symp., Security, 2012.

[23] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security," IEEE Security & Privacy Magazine, Vol.7, No.1, pp.50-57, January/February 2009.

[24] http://www.honeynet.org

[25] A. Moser, C. Kruegel, and E. Kirda, "Detection of Mobile Malware in the Wild," Comput.,

Vol.45, No.9, 2012(9).

[26] Andrew Hoog, “Android Forensics: Investigation, Analysis and Mobile Security for Google Android,” 2011(6).

[27] https://www.av-test.org

저 자 소 개

김 의 탁(Eui-Tak Kim) 정회원

- 1997년 2월 : 대전대학교 컴퓨터공학과(공학사)
- 1999년 8월 : 대전대학교 컴퓨터공학과(공학석사)
- 2018년 현재 : 충북대학교 전자계산학과(공학박사)
- 2001년 ~ 2005년 : ㈜아이언마스크 기술이사
- 2005년 ~ 2010년 : 티에스온넷(주) 정보보호연구소 부장
- 2010년 ~ 현재 : ㈜하우리 기술연구소 연구소장

<관심분야> : 접근통제시스템, 악성코드 분석 및 탐지, 데이터 마이닝, 클라우드 컴퓨팅, 핀테크 보안, 인공지능

류 근 호(Keun Ho Ryu) 정회원

- 1976년 : 숭실대학교 전자계산학과(공학사)
- 1980년 : 연세대학교 공학대학원 컴퓨터공학(공학석사)
- 1988년 : 연세대학교 대학원 컴퓨터공학전공(공학박사)
- 1976년 ~ 1986년 : 육군 3군수 지원사 전산실(ROTC 장교), 한국전자통신연구원 (연구원), 한국방송통신대 전산학과(조교수)
- 1989년 ~ 1991년 : 미국 Univ. of Arizona Research Staff (TempIS 연구원, Temporal DB)
- 1986년 ~ 현재 : 충북대학교 전기전자정보컴퓨터학부(교수)

<관심분야> : 시공간 데이터베이스, 빅데이터 분석, 지식기반 정보검색, 데이터마이닝, 데이터보안, 바이오메디칼 및 바이오인포매틱스