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

안드로이드 플랫폼에서 odex 파일을 이용한 불법 복제 앱 탐지 방법☆

An Illegally-copied App Detecting Method by Using Odex File in Android Platform

조 득 연[1] 최 재 영[1*] 김 은 회[2] 강 기 두[3]
Dueckyoun Cho Jaeyoung Choi Eunhoe Kim Gi-Du Gang

요 약

모바일 환경의 변화에 따라 안드로이드 앱의 사용이 증가하면서 앱에 대한 관심도 증가하였다. 하지만 불법으로 복제된 앱의 사용도 증가하여 앱 마켓의 투명성과 신뢰성이 저하되고 앱 개발자의 저작권을 침해하는 피해 사례가 발생하고 있다. 불법으로 복제된 앱의 사용을 방지하는 다양한 기술이 연구되었지만, 이를 우회하는 방법이 있기도 하고 또한 최초 유포자에 대한 정보를 알아낼 수 없어서 법적으로 제재하기도 어렵다. 본 논문에서는 안드로이드 플랫폼에서 불법 복제 앱을 탐지하는 방법을 제안한다. 불법 복제 앱 탐지기는 앱 설치 과정에서 생성되는 odex 파일을 사용함으로 불법 복제 앱을 탐지하고, 포렌식 워터마크 기술을 사용함으로써 최초 유포자의 정보를 알아낼 수 있다. 제안하는 불법 복제 앱 탐지기는 시스템 서버에서 서비스 형태로 실행되어 사용자에게 노출되지 않는다. 실험 결과, 불법 복제 앱 탐지기는 평균 0.2초 이내로 불법 복제 앱을 탐지하고 삭제하는 것이 가능하다.

☞ 주제어 : 불법 복제 앱, 탐지기, 안드로이드 플랫폼, 실행 은닉, odex 파일

ABSTRACT

According to the changes of the mobile environments, the usage and interest of the Android apps have been increased. But the usage of illegally-copied apps has been also increased. And the transparency and dependability of the app markets has been decreased. Therefore there are many cases for the copyright infringement of app developers. Although several methods for preventing illegally-copied apps have been studied, there may exist possible ways to bypass the methods. Since it is difficult to find out the first distributors of the illegally-copied apps, it is not easy to punish them legally. This paper proposes the method of detecting illegally-copied apps. The proposed detector can detect the illegally-copied apps using odex file, which is created when the app is installed. The detector can also find out the information of the first distributors based on forensic watermark technique. Since the illegally-copied app detector is running as a service on the system server, it is granted that the detector hides from the users. As an experiment result, the illegally-copied app detector takes on average within 0.2 seconds to detect and delete an illegally-copied app.

☞ keyword : illegally-copied app, detector, android platform, execution hiding, odex file

1. 서 론

모바일 환경은 점차 대용량의 데이터를 고속으로 전송할 수 있도록 발전하였다. 이에 문자와 음성 데이터 위주의 모바일 통신 환경에서 사진, 동영상 등의 대용량의 데이터를 전송할 수 있는 스마트 모바일 환경으로 변화하였고, 대용량의 데이터를 처리할 수 있는 스마트 모바일 기기의 사용량도 폭발적으로 증가하였다. 정보통신정책연구원에서 발표한 '2013 한국 미디어 패널 조사'[1] 내용에 따르면, 2011년도에는 3G/LTE 스마트폰이 전체 휴대폰 사용량의 23.8%를 기록하고 있지만, 2013년도에는 72%로 2011년도에 비해 약 48.2%p 증가하였다. 반면 2G/3G 비 스마트폰의 경우, 2011년도에 75.4%를 기록하였지만 2013년도에는 27.9%로 2011년도에 비해 약 47.5%p 감소하였다.

[1] School of Computer Science and Engineering, Soongsil University, Seoul 156-743, Korea
[2] Department of Internet Information, Seoil University, Seoul 131-702, Korea
[3] School of Business Administration, Soongsil University, Seoul 156-743, Korea
* Corresponding author (choi@ssu.ac.kr)
[Received 24 December 2014, Reviewed 6 January 2015, Accepted 2 March 2015]
☆ 이 연구는 2013년도 숭실대학교 교내연구비 지원에 의해 수행되었습니다.

스마트 기기의 사용량의 증가는 스마트 기기에서 사용할 수 있는 앱에 대한 관심과 앱 마켓의 발전을 가져왔다. 특히 국내 모바일 플랫폼 사용률 중 약 92%를 차지하는 안드로이드 플랫폼의 대표 마켓인 구글 플레이 스토어에는 2013년을 기준으로 약 100만개 이상의 앱이 등록되고, 500억 번 이상의 다운로드 수를 기록하고 있다[2].

하지만 앱을 앱 마켓에서 정상적으로 구매하지 않고 불법으로 복제한 앱을 사용하는 사례가 증가하며, 이로 인한 피해 사례도 증가하고 있다. 불법으로 복제된 앱은 개발자의 창작물인 앱을 정상적이지 않은 방법으로 복제하였기 때문에 앱 개발자의 저작권을 침해하며, 판매로 인한 수익이 발생되지 않기 때문에 앱 개발자의 개발 의욕을 저하시키는 등, 앱 마켓에 대한 투명성과 신뢰성을 저하시킨다. 또한 앱을 불법으로 복제하는 과정에서 악성 코드가 삽입되어 불법으로 복제된 앱을 사용하는 기기에서 사용자의 정보가 유출되는 2차적인 피해도 발생될 수 있다.

본 논문에서는 안드로이드 플랫폼에서 불법으로 복제된 앱을 탐지하는 탐지기를 제안한다. 불법 복제 앱 탐지기는 앱이 설치된 직후, 설치된 앱의 실행 파일인 odex 파일을 이용하여 앱의 불법 복제 여부를 판별한다. 포렌식 워터마크 기술을 사용하여 불법으로 복제된 앱의 유포자를 추적할 수 있다는 장점이 있다. 불법 복제 앱 탐지기는 시스템 서버의 서버 쓰레드로 탑재되어 시스템 서비스 모듈 형태로 실행되며, 시스템 서버의 서비스 재실행 기능으로 항상 실행을 유지할 수 있다는 특징이 있다.

본 논문의 구성은 다음과 같다. 2장에서는 안드로이드 플랫폼에서 불법으로 복제된 앱을 방지하는 선행 연구에 대해 논한다. 3장에서는 제안한 odex를 사용하는 불법 복제 앱 탐지기의 불법 복제 앱 탐지 방법 및 운영 방법을 설명한다. 4장에서는 불법 복제 앱 탐지기의 실험 결과에 대해 논한다. 마지막으로 5장에서는 논문의 결론을 맺으며 향후 연구 내용에 대해 설명한다.

2. 관련 연구

불법으로 복제된 앱은 정상적으로 구매한 앱에서 다시 설치할 수 있는 apk 파일을 추출하여 블랙 마켓이나 P2P 사이트 등을 통해 다른 사용자들에게 유통한 앱 파일을 의미한다. 안드로이드에서 불법으로 복제된 앱을 방지하기 위하여 불법으로 복제된 앱의 실행을 차단하거나 앱의 불법 복제를 차단하는 기술들이 연구되었다.

구글에서 제공하는 LVL (License Verification Library) 기술[3]은 불법 복제 앱의 실행을 방지하는 기술로, 개발자 수준에서 적용 가능한 불법 복제 앱 방지 기술이다. 개발자는 안드로이드 앱을 마켓에 올리기 전에, 앱이 실행될 때마다 인증 작업을 수행할지 혹은 최초 1회 실행될 때에 인증 작업을 수행할지를 옵션으로 지정하여 LVL 기술을 적용한다. LVL 기술을 적용한 앱은 내부에 LVL 클래스가 유지되며, 이 클래스는 앱이 실행될 때에 안드로이드 라이센스 서버에 접속하여 정상 구매자 여부를 확인한다. 정상 구매자의 경우 클래스는 Success 값을 반환하여 앱을 정상적으로 실행하도록 하고, 정상 구매자가 아닌 경우에는 Fail 값을 반환하여 앱의 실행을 방지할 수 있다. 하지만 앱을 해킹하여 LVL 클래스를 수정하는 방식으로 LVL 기술이 적용된 앱 파일을 불법 복제하여 사용할 수 있고, 추가적인 네트워크 비용이 발생한다는 단점이 있다.

사용자 선호기반 강제 접근 제어 방식[4]은 앱의 불법 복제를 차단하는 기술로 안드로이드 커널 수준에서 제공하는 불법 복제 앱 방지 기술이다. 안드로이드 커널에서 프로세스들의 정보를 저장하는 테이블을 유지하여 테이블에 있는 프로세스들만 apk/dex 파일이 저장된 디렉터리에 접근할 수 있는 권한을 제공하고, 다른 프로세스들은 apk/dex 파일이 저장된 디렉터리에 접근할 수 없도록 하여 앱 파일의 추출을 방지한다. 안드로이드 앱을 실행하거나 삭제하는 등의 안드로이드 플랫폼 운영을 위한 프로세스들의 접근은 허용하고, 마켓에서 합법적으로 구매한 앱의 백업을 목적으로 하는 프로세스들도 접근이 허용된다. 기타 프로세스들에 대해서는 앱의 추출 등의 목적을 가지고 앱 파일에 접근한다고 간주하고 이를 강제적으로 방지하고 있다. 하지만 커널 코드의 수정이 필요하기 때문에 쉽게 적용할 수 없다는 문제점이 있다.

앱에 암호화 모듈을 삽입하는 기술[5]은 불법 복제 앱의 실행을 방지하는 기술로, 앱 마켓 서버에서 제공하는 불법 복제 앱 방지 기술이다. 앱의 일부 모듈을 암호화하여 저장해두고 구매 시점에서 사용자에 맞는 키 값을 제공한 뒤에 앱의 설치 및 실행 과정에서 제공된 키 값을 이용하여 암호화된 모듈을 복호화 및 재암호화를 수행한다. 정상적인 사용자의 경우, 설치나 실행이 정상적으로 되지만, 불법 복제 앱 사용자의 경우에는 암호화된 모듈을 정상적으로 복호화하는 키 값이 없기 때문에 앱의 설치나 실행이 정상적으로 되지 않는다. 하지만 암호화된 모듈을 복호화하고 앱 파일을 재구성하여 불법 복제함으로써 쉽게 우회할 수 있다는 문제점이 있다.

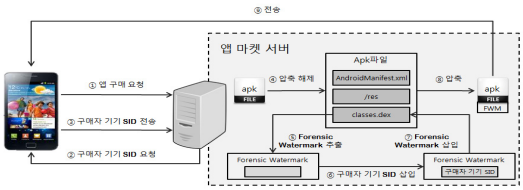
이와 같은 불법 복제 앱의 사용을 방지하는 기술 외에도, 불법 복제 앱 사용이나 유포에 대한 법적인 제재를 가하는 방법이 있다. 하지만 대부분의 불법 복제 앱 방지 기술을 우회할 수 있는 방법이 제공되고 있어서, 유출자를 완벽하게 추적하기가 힘들기 때문에 법적인 제제가 힘들다.

본 논문에서 제공하는 불법 복제 앱 탐지기는 앱 설치 시점에서 불법 복제 앱 사용을 방지한다. 앱 내부에 삽입된 포렌식 워터마크를 이용하여 불법 복제 여부를 판별하여 강제적으로 불법 복제 앱을 삭제한다. 삽입된 포렌식 워터마크에는 앱 파일의 구매 정보가 삽입되기 때문에 불법 복제 앱의 최초 유포자를 판별할 수 있다. 불법 복제 앱 탐지기는 시스템 서버에서 운용되는 시스템 서비스 모듈 형태로 안드로이드 플랫폼 수준에서 운영하며 사용자의 강제적인 접근으로 서비스가 종료되더라도 재실행된다. 안드로이드의 시스템 서버의 코드는 안드로이드의 버전이 업그레이드되어도 항상 유지되고 있으므로 시스템 서버의 서비스로 등록되어 다양한 안드로이드 버전에서 적용 가능하고 추가적인 네트워크 비용이 발생하지 않는 특성이 있다.

3. 불법 복제 앱 탐지기

3.1 포렌식 워터마크

워터마크 기술은 주로 콘텐츠의 저작권을 보호하기 위해 많이 사용되어 왔으며 사람이 인지할 수 없는 소유권자의 정보를 콘텐츠에 삽입하는 기술이다. 포렌식 워터마크 기술은 워터마크 기술에 추가적으로 구매자 정보 및 유통 경로, 사용자 정보 등을 삽입하는 기술로, 디지털 데이터가 불법적으로 무단 복제되었을 경우에 복사본의 원 구매자를 식별함으로써 불법적인 배포를 방지한다. 불법 복제 앱 탐지기는 앱의 불법 복제 여부를 확인할 뿐만 아니라 최초 유포자를 확인하기 위하여 포렌식 워터마크 기술을 사용한다[6].



(그림 1) 앱 마켓 서버에서 앱 파일에 포렌식 워터마크 기술 적용 과정

(Figure 1) Forensic Watermark Technology Application Process to App File In App Market Server

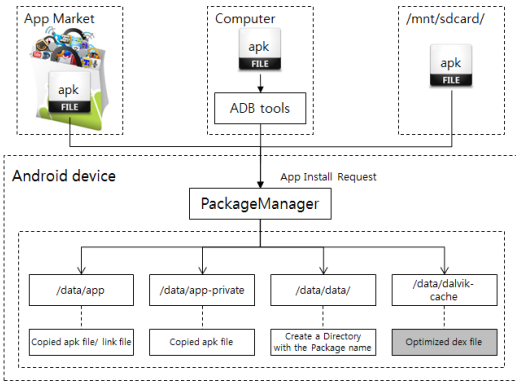
그림 1은 앱 마켓 서버에서 앱 파일에 포렌식 워터마크 기술을 적용하는 과정이다. 앱 구매 요청이 들어오면 먼저 구매자 기기의 SID(Subscribe IDentification) 값을 확인한다. 앱 마켓 서버는 apk 파일의 압축을 해제하고 내부의 dex 파일에 구매자 기기의 SID 값을 포렌식 워터마크로 삽입한 후에 앱 파일을 다시 압축하여 구매자 기기에 전송함으로써 구매 과정을 완료한다[7]. 불법 복제 앱 탐지기는 설치된 앱의 포렌식 워터마크를 추출하여 구매자 기기의 SID를 추출하고, 앱이 설치된 기기의 SID와 비교하여 서로 일치하면 정상적으로 구매가 이루어진 앱으로 판별하고, 서로 일치하지 않으면 불법 복제 앱으로 판별한다. 포렌식 워터마크는 앱의 dex 파일에 삽입되는데, 설치된 앱에 포렌식 워터마크로 삽입된 SID를 추출하기 위해서는 dex 파일이 필요하다. apk 파일은 앱이 설치된 이후 기기 내부에 저장되고, dex 파일은 apk 파일 내부에 압축되어 유지된다. 불법 복제 앱 탐지기는 저장된 apk 파일의 압축을 해제하여 dex 파일을 추출한 뒤에, dex 파일에서 다시 포렌식 워터마크를 추출하여 앱의 불법 복제 여부를 확인해야 한다. 하지만 apk 파일의 압축 해제에 소요되는 시간은 apk 파일의 크기에 비례하여 증가하기 때문에 설치된 앱 파일을 압축 해제하여 dex 파일을 추출한다면 상당한 시간이 소요된다. 따라서 불법 복제 앱 탐지기는 앱 설치 과정에서 생성되어 앱이 삭제될 때까지 유지되는 odex 파일을 이용하여 앱의 불법 복제 여부를 판별하는 방법을 제안한다.

3.2. 앱 설치 과정에서 생성되는 odex 파일

불법 복제 앱 탐지기는 설치된 앱의 불법 복제 여부를 판별하기 위하여 앱 설치 과정에서 생성되는 odex 파일을 이용한다. 그림 2는 앱의 설치 과정에서 생성되는 파일과 디렉터리를 나타낸다. 앱 실행 중에 생성되거나 사용할 수 있는 데이터를 저장 및 관리하기 위해, /data/data/ 디렉터리 하위에 패키지 이름을 갖는 디렉터리를 생성한다. 또한 apk 파일을 복사하여 /data/app/ 디렉터리에 저장한다. 만약 앱 개발자가 앱의 강제 추출 방지 옵션을 설정했다면 /data/app-private/ 디렉터리에 apk 파일을 저장하고, /data/app/ 디렉터리에 apk 파일의 심볼릭 링크를 저장한다. 그리고 앱의 실행 파일인 dex 파일을 검증 및 최적화하고, 최적화된 dex 파일(odex 파일, Optimized dex)을 /data/dalvik-cache/ 디렉터리에 저장한다.

안드로이드 플랫폼은 기본적으로 안드로이드 기기에 설치되는 모든 앱에 대하여 odex 파일을 생성하도록 설

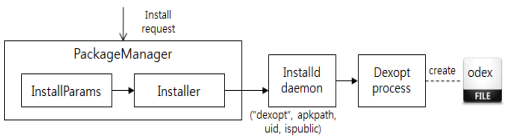
정되어 있다. odex 파일은 바이트코드 형태로 apk 파일 내부에 저장된 앱 실행파일인 dex 파일을 dexopt 프로세스에 의해 검증과 최적화 과정을 거쳐서 생성된다[8].



(그림 2) 앱 설치 과정에서 생성되는 정보
(Figure 2) The information generated by the app install process

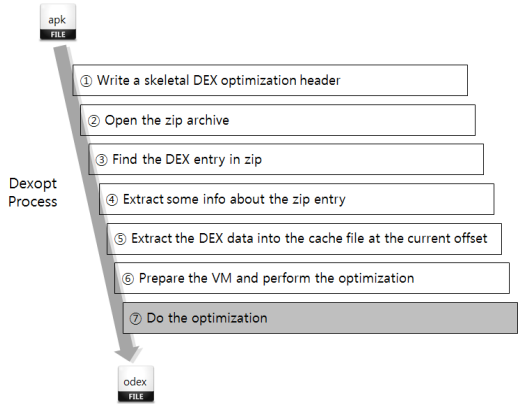
3.3 dexopt 프로세스

그림 3은 앱 설치 과정에서 dexopt 프로세스가 실행되기까지의 과정을 보여준다. 패키지 매니저는 앱 설치 요청이 들어오면 InstallParams에 설치할 앱의 정보를 넣고, Installer에서 설치 과정을 수행한다. Installer는 dexopt 프로세스를 생성하기 위해 ("dexopt", apkpath, uid, ispublic)으로 구성된 요청 정보, 즉 apk가 설치될 경로, uid, 강제 추출 방지 옵션을 Installd daemon으로 보낸다. 요청을 받은 Installd daemon은 dexopt 프로세스를 호출하고, dexopt 프로세스는 앱 파일에서 dex 파일을 추출한 뒤에 검증과 최적화 과정을 거쳐 dex 파일을 odex 파일로 변형한다.



(그림 3) 앱 설치 과정에서 dexopt 프로세스 실행 과정
(Figure 3) Dexopt Process excution process in app install process

앱에 대한 검증은 dex 영역의 모든 클래스의 모든 메소드가 가지는 명령에 대해서 잘못된 표현을 식별하여 앱 실행 중에 잘못된 명령을 실행하지 않도록 미리 확인하는 과정이다. dexopt 프로세스는 검증 과정을 마친 뒤에 그림 4의 과정을 통해 apk 파일에서 odex 파일을 생성한다. ① 우선 /data/dalvik-cache/ 디렉터리 내부에 dex 영역의 경로로 이름이 지정된 classes.dex라는 odex 파일을 생성하며, 비어있는 40Bytes의 odex header를 추가한다. ② apk 파일의 압축을 해제하여, ③ 내부에 저장된 dex 영역의 시작 주소 및 크기 등의 정보를 얻어온다. ④ apk 파일 전체에서 odex 파일 생성에 필요한 정보를 추출한다. ⑤ 본격적인 최적화 과정을 위하여 apk 파일에서 dex 영역을 추출한다. ⑥ 실행되는 다른 앱과의 자원 충돌로 인해 최적화가 제대로 진행되지 않을 수 있기 때문에, 앱이 실행되는 가상 머신이 아닌 별도의 가상 머신을 새로 생성하여 최적화 과정을 진행할 준비를 모두 마친다. ⑦ 마지막으로 최적화를 수행하여 odex 파일을 생성한다. dex 영역에서 추출된 정보는 클래스 검증 과정을 수행한 뒤에 앱이 설치되는 안드로이드 로컬 시스템에 맞게 Byte Reordering, Structure realigning, Bytecode optimization 과정을 거쳐서 odex 파일이 생성된다. 기기 정보에 따라 변경되는 방식이 다르기 때문에 기기마다 서로 다른 odex 파일이 생성된다[8].

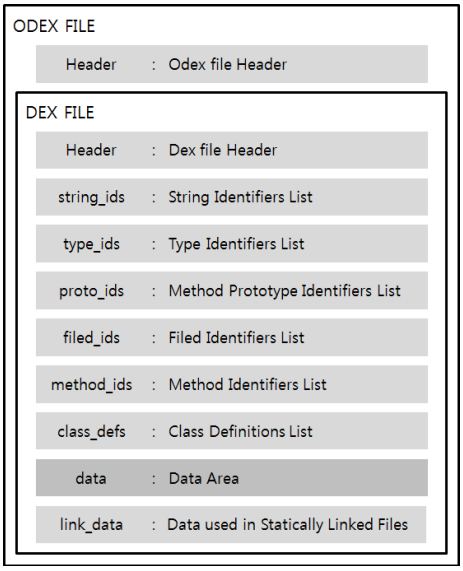


(그림 4) dexopt 프로세스에 의한 odex 파일의 생성 과정
(Figure 4) the creation of a odex file by dexopt process

3.4 odex 파일을 이용한 앱의 불법 복제 여부 판별

최적화를 마치고 생성된 odex 파일은 그림 5와 같은 구조를 갖게 된다[9]. dexopt 프로세스에 의한 odex 파일 생성 과정에서 첫 번째 단계로 40Bytes의 odex header 영

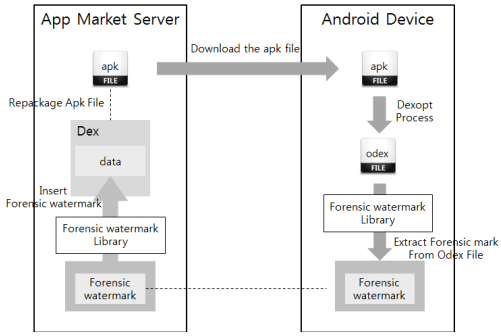
역을 생성한다. odex header 영역은 dexopt 프로세스의 단계가 지나면서 유용한 값들로 채워진다. 나머지 영역은 기존의 dex 파일의 구조와 동일한 구조를 갖는데, 기존의 dex 파일의 header인 Dex Header 영역, class의 이름이나 문자열 상수, 변수 이름 등과 같이 문자열의 개수와 시작 위치의 정보를 테이블 형태로 기술하는 string_ids, 안드로이드 프로그래밍에서의 함수의 리턴 값이나 인자 값의 형식을 테이블 형태로 기술하는 type_ids, 함수 원형에 대한 정보를 테이블 형태로 기술하는 proto_ids, 안드로이드 프로그램의 패키지의 이름과 클래스의 이름 등의 정보를 테이블 형태로 기술하는 field_ids, class에서 사용된 method의 정보를 테이블 형태로 기술하는 method_ids, 그리고 실제 모든 class의 정보를 기술하는 class_defs, 마지막으로 data와 link_data 영역은 앞에서 작성된 테이블과 연결되는 실제 데이터가 저장된 영역이다.



(그림 5) odex 파일의 구조
(Figure 5) The structure of odex file

이들 영역 중에 string_ids 영역부터 class_defs 영역까지는 실제 dex 파일 내부의 class나 method 등의 정보에 접근하기 위한 테이블을 유지하고 있으며, 실제 구현된 class 내용은 data 영역에 위치한다. dexopt 프로세스에 의해 dex 파일이 최적화되는 과정은 기기가 사용하는 시스템의 정보를 참조하며, 바이트 교체나 재정렬이 이루어지는 부분은 실제 class가 구현되어 있는 data 영역이 아닌, data 영역에 접근하기 위한 offset 정보가 저장되어 있는 테이블 부분이다. 따라서 dexopt에 의한 최적화 과정에서 dex 파일 내부의 data 영역은 변형되지 않고 odex 파일에 동일하게 저장된다. 따라서 apk 파일의 구매가 이루어질 때 dex 파일의 data 영역에 포렌식 워터마크를 삽입하여 앱이 설치된 후 생성되는 odex 파일에서 변형없이 추출 가능하게 한다. dex 파일의 data 영역에 구매자 기기의 SID를 포렌식 워터마크로 삽입하기 위하여 포렌식 워터마크 기술을 지원하는 AWL(Android Watermark Library)[6][7]을 사용한다.

그림 6은 포렌식 워터마크 라이브러리를 이용한 포렌식 워터마크 추출 과정을 나타낸다. 앱을 구매하면 앱 마켓에서 apk 파일을 압축 해제하여 dex 파일에 포렌식 워터마크를 삽입한 뒤, 재압축하여 apk 파일을 구성한다. 포렌식 워터마크 라이브러리는 dexopt 프로세스가 생성한 odex 파일에서 dex 파일에 삽입한 동일한 포렌식 워터마크를 추출할 수 있다. 포렌식 워터마크에 삽입된 구매자 기기의 SID와 앱이 설치된 기기의 SID를 비교하여 앱의 불법 복제 여부를 확인할 수 있고 최초 유포자 정보를 확인할 수 있다.

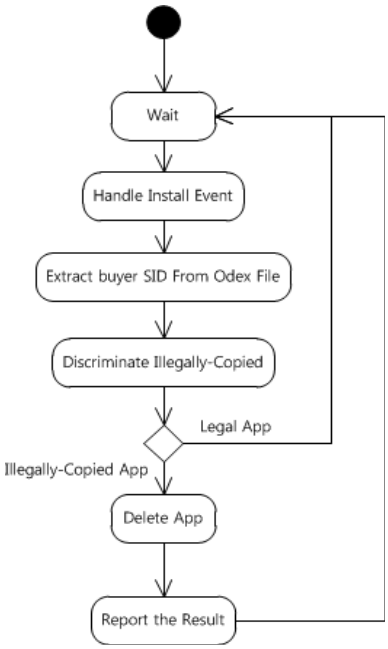


(그림 6) 포렌식 워터마크 라이브러리를 이용한 포렌식 워터마크 추출
(Figure 6) The extraction of forensic watermark using forensic watermark library

3.5 시스템 서비스 형태로 실행되는 불법 복제 앱 탐지기

그림 7은 포렌식 워터마크 기술이 적용된 불법 복제 앱 탐지기의 동작 과정을 보여준다. 불법 복제 앱 탐지기가 실행되면 패키지 매니저로부터 앱 설치 이벤트를 받을 때까지 대기한다. 패키지 매니저로부터 앱 설치 이벤

트를 받게 되면, 설치된 앱의 odex 파일에 삽입되어 있는 구매자 기기의 SID를 포렌식 워터마크 라이브러리를 통해 추출하고, 앱이 설치된 기기의 SID와 비교하여 앱의 불법 복제 여부를 확인한다. 불법 복제 앱으로 판별되면 해당 앱을 삭제하고 결과를 사용자에게 리포팅한다. 정상 앱으로 판별되면 불법 복제 앱 탐지기는 다시 대기 상태로 돌아와 패키지 매니저로부터 다음 앱 설치 이벤트를 받을 준비를 한다.



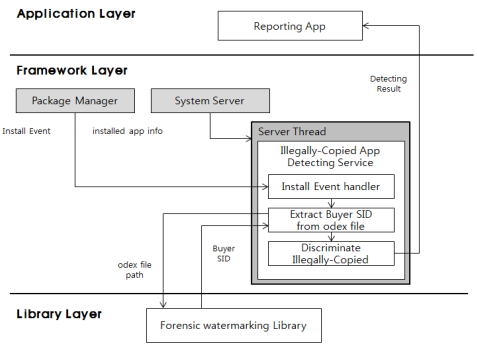
(그림 7) 불법 복제 앱 탐지기의 동작 과정
(Figure 7) The operation proecess of Illegally-copied App Detector

불법 복제 앱 탐지기는 사용자의 기기에 설치되는 불법 복제 앱을 탐지하여 사용을 방지하는 목적을 갖는다. 따라서 안드로이드 기기의 부팅과 동시에 실행되어 기기가 꺼질 때까지 실행이 유지되어야 하며, 사용자가 실행 여부를 파악하지 못하도록 사용자로부터 은닉되어 실행되어야 한다.

안드로이드 플랫폼에서 서비스는 특정 기능을 수행하기 위하여 실행되며, 프로세스의 일부로서 다른 서비스나 앱에서 호출하여 사용할 수 있는 프로그램이다. 기기가 부팅된 이후 모든 서비스는 컨텍스트 매니저라는 서비스 매니저에 등록되어 관리된다. 서비스는 바인더 인터페이스를 생성하므로 일반 프로세스에서 바인더 인터페이스를 사용하여 서비스를 호출하여 사용할 수 있다. 안드로이드 플랫폼은 앱 실행이나 기기의 운영에 필요한 필수적인 기능도 서비스로 구동하는데 앱의 생명주기나 패키지 관리, 앱 설치 및 삭제, 화면 제어 등의 플랫폼 운영을 관리하는 시스템 프로그램 수준의 서비스를 시스템 서비스로 구분한다. 시스템 서비스는 하드웨어 서비스, 네이티브 서비스, 코어 플랫폼 서비스로 구분하며 미디어 서버와 시스템 서버에서 운용한다. 이 중에서 코어 플랫폼 서비스는 하드웨어의 사용과 상관없이 안드로이드 운영을 위한 실질적인 요소들을 담당한다. 코어 플랫폼 서비스는 항상 메모리에서 실행되어야 하는 중요한 서비스로 구성되어 있고, 시스템 서버 프로세스에 포함되어 서버 쓰레드를 통해 구동된다[10].

불법 복제 앱 탐지기는 기기에 설치되는 불법 복제 앱을 탐지하여 삭제하는 기능을 수행하기 때문에 사용자에 의한 시작이나 종료가 불필요하며, 사용자로부터 은닉되어 실행되어야 하므로 기존 시스템 서비스와 같은 형태로 운용하는 것이 적합하지 않다. 불법 복제 앱 탐지기는 기존 시스템 서비스와 달리 컨텍스트 매니저에 등록하지 않고, 바인더 인터페이스를 생성하지 않게 만들어서 앱 수준에서 접근이 불가능하도록 설계하여 은닉성을 제공한다. 시스템 서버의 서버 쓰레드에 의해 실행되게 하여 루팅과 같은 불법 권한 획득 이후에도 강제적인 프로세스 종료가 불가능하도록 한다. 시스템 서버의 서버 쓰레드는 안드로이드 기기의 운영에 반드시 필요한 시스템 서비스들을 실행하기 때문에 에러 값을 인자로 전달하여 시스템 서비스를 강제적으로 종료하더라도 재실행하여 항상 실행을 보장한다.

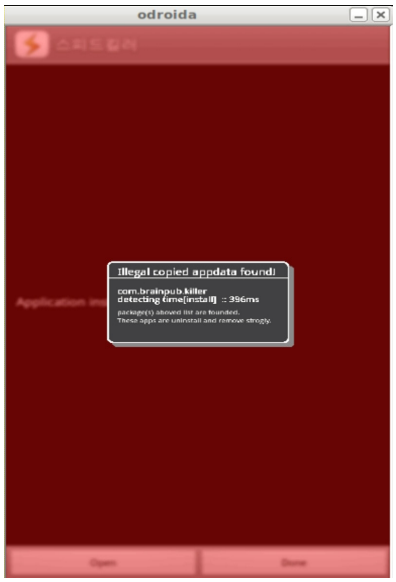
불법 복제 앱 탐지기는 그림 8과 같은 구조를 갖는다. 프레임워크 레이어에 위치한 불법 복제 앱 탐지 서비스는 시스템 서버에 의해 서버 쓰레드로 실행되며, 앱 수준에서 접근이 불가능하고 사용자가 프로세스 목록에서 확인할 수 없으며 강제적인 종료가 발생하도 재실행된다. 라이브러리 레이어에 위치한 포렌식 워터마크 라이브러리는 AWL을 사용하며, odex 파일에 삽입된 포렌식 워터마크를 추출한다[7]. 불법 복제 앱 탐지 결과는 어플리케이션 레이어에 위치한 리포팅 앱으로 전송되어 사용자가 탐지 결과를 확인할 수 있다. 불법 복제 앱 탐지기는 시스템 서버에서 운용되어 시스템 서버의 system 권한을 가지고 있으므로 탐지된 불법 복제 앱의 삭제에 대한 명령을 패키지 매니저에 보낼 수 있다.



(그림 8) 불법 복제 앱 탐지기의 구조
(Figure 8) The structure of Illegally-copied App Detector

3.6 불법 복제 앱 탐지 결과 화면

불법 복제 앱 탐지기는 탐지된 불법 복제 앱을 삭제하고 사용자에게 그 결과를 리포팅한다. 안드로이드 코어 플랫폼 서비스 중 화면을 제어하는 윈도우 매니저를 이용하여 그림 9와 같이 화면의 최상단에 결과 화면을 출력하며 화면에 출력할 다이얼로그 내용은 탐지된 앱의 패키지 이름과 탐지 시간으로 구성되어 있다. 결과 화면은 사용자가 확인한 뒤 화면을 터치할 때까지 유지된다.

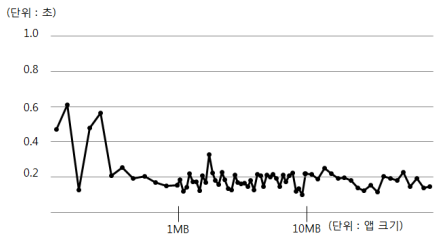


(그림 9) 불법 복제 앱 탐지 결과 리포팅
(Figure 9) Illegally-copied App detection Reporting

4. 실험 결과

불법 복제 앱 탐지기는 앱 설치 과정에서 생성되는 odex 파일을 이용하여 설치된 앱의 불법 복제 여부를 판별하고, 불법 복제 앱인 경우 기기에서 삭제하며 사용자에게 그 결과를 리포팅한다. 본 논문에서는 불법 복제 앱 탐지기의 불법 복제 앱 탐지 시간을 측정함으로써, 불법 복제 앱 탐지기의 성능을 분석한다.

실험은 총 100개의 상용 앱을 대상으로 하였으며, 각각의 앱은 최소 100KB부터 최대 50MB의 앱들로 구성된다. 20개의 앱은 포렌식 워터마크 기술을 적용하지 않았고 80개의 앱은 포렌식 워터마킹 기술을 적용하였다. 80개의 앱 중 9개는 실험 기기와 동일한 SID 값을 포렌식 워터마크로 삽입하였고, 나머지 71개의 앱은 실험 기기와 다른 SID 값을 삽입하여 불법 복제 앱으로 만들었다. 실험에 사용한 기기는 안드로이드 플랫폼 개발을 위한 디버깅 보드 중 하나인 OdroidA (hardKernel)이며, 사용한 안드로이드 플랫폼 버전은 진저브레드(Gingerbread)이다.



(그림 10) 불법 복제 앱 탐지기의 탐지 시간
(Figure 10) The detection time of Illegally-copied App detector

그림 10은 불법 복제 앱 탐지 시간을 나타낸다. 71개의 불법 복제 앱에 대해서 평균 199ms의 탐지 시간이 소요되었으며 최소 100.7ms, 최대 609ms의 시간이 소요되었다. odex 파일을 이용하여 불법 복제 여부를 확인하기 때문에 apk 파일의 압축 해제 과정이 필요 없으므로 앱 크기와 상관없이 평균 0.2초 이내의 탐지 시간을 보인다.

불법 복제 앱 탐지기는 시스템 서비스 형태로 백그라운드에서 항상 실행되고 있지만 앱 설치 이벤트를 받기 전까지는 대기 상태를 유지하며, 앱 설치 이벤트를 받은 후에 설치된 앱에 대한 불법 복제 여부를 확인하기 때문에 불법 복제 앱 탐지기의 실행으로 인한 오버헤드는 거의 발생되지 않으므로 좋은 성능 효율을 보인다.

5. 결 론

모바일 환경의 변화와 스마트 기기 사용량의 증가로 인해 앱에 대한 관심이 높아지고, 앱 마켓도 발전하였다. 하지만, 불법 복제 앱의 사용으로 인해 개발자의 저작권 문제와 개발 의욕 저하, 불법 복제 앱 사용자의 정보 유출 등의 피해가 증가하고 있다.

본 논문에서는 안드로이드 플랫폼에서 사용할 수 있는 불법 복제 앱 탐지기를 제안한다. 불법 복제 앱 탐지기는 odex 파일에서 포렌식 워터마크 형태로 삽입된 구매자 기기의 SID 정보를 추출하고, 앱이 설치된 기기의 SID 정보와 비교하여 앱의 불법 복제 여부를 판별할 수 있다. 또한 포렌식 워터마크 기술을 사용하므로 최초 유포자의 정보도 확인할 수 있는 장점이 있다. 불법 복제 앱 탐지기는 시스템 서버에서 운용되는 서버 쓰레드에 의해 시스템 서비스 형태로 실행되며, 일반적인 시스템 서비스와 달리 컨텍스트 매니저에 등록되지 않고 바인더 인터페이스를 생성하지 않는다. 따라서 사용자가 불법 복제 앱 탐지 서비스의 실행 여부를 알 수 없게 된다. 또한 시스템 서버의 서비스 재실행 기능에 의해 항상 실행을 유지할 수 있다. 불법 복제 앱 탐지기는 앱의 불법 복제 여부 판별 및 삭제, 사용자에게 리포팅하는 시간이 평균 0.2초로 좋은 성능을 보였다.

향후에 설치나 유포를 목적으로 안드로이드 기기 내부의 파일 시스템에 저장된 불법 복제 앱 파일까지 삭제하는 기술을 불법 복제 앱 탐지기에 적용하여, 불법 복제 앱의 실행뿐만 아니라 배포를 방지하는 연구를 진행할 예정이다.

참 고 문 헌 (Reference)

[1] Mincheol Kim, Jihyung Shin, Yunhwa Kim, Taelim Ha, Sun Shin, “2013 Korea Media Panel Research”, Korea Information society development institute, 2013. http://www.kisdi.re.kr/kisdi/fp/kr/board/selectSingleBoard.do?cmd=select SingleBoard&curPage=1&boardId=GPK_RND_DATA&seq=29270&reStep=8999&ctx=_

[2] Mashable, Available at : http://www.mashable.com

[3] Android LVL, Available at: http://developer.android.com/google/play/licensing/adding-licensing.html

[4] Hanbyul Baek, Eungyu Lee, Kanghee Kim, “A Mobile Application Anti-Piracy Technique Using Mandatory Access Control Based on User Preferences”, Journal of Security Engineering, Vol.10, No.2, pp.151-162, 2013. http://www.sersc.org/journals/JSE/vol10_no2_2013/4.pdf

[5] Sung-Ryul Kim, “Copy Protection System for Android App using Public Key Infrastructure”, Journal of Security Engineering, Vol.9, No.1, pp.121-134, 2012. http://www.sersc.org/journals/JSE/vol9_no1_2012/11.pdf

[6] H. Kim, ″The Technology Trend of Forensic mark″, Korea Copyright Commission, Sept. 2010. http://www.copyright.or.kr

[7] Cheol Jeon, Yookun Cho, ″A Robust Steganography-Based Software Watermarking″, RACS 2012, San Antonio, TX, USA, 2012. http://dl.acm.org/citation.cfm?id=2401675

[8] The Android Open Source Project. Dalvik optimization and verification with dexopt. Available at: http://www.netmite.com/android/mydroid/dalvik/docs/dexopt.html.

[9] Security Enginering Research Group, ″Analysis of Dalvik Virtual Machine and Class Path Library″, 2010. “http://www.academia.edu/4768292/Analysis_of_Dalvik_Virtual_Machine_and_Class_Path_Library_Constrained_Intents_Extending_Android_Security_for_Intent_Policies_EASIP_”

[10] Taeyoun Kim, Jihoon Park, Sangyup Kim, WangJae Yi, “Android Anatomy System Service”, Developers Happy World, 2011.

◐ 저 자 소 개 ◑

조 득 연 (Dueckyoun Cho)
2012년 2월: 숭실대학교 컴퓨터학부 졸업 (학사)
2014년 8월 : 숭실대학교 컴퓨터공학과 (석사)
2014년 ~ 현재: (주)아이디어웨어 연구원
관심분야: 분산/클라우드 컴퓨팅, 스마트모바일
E-mail : helloworld@ssu.ac.kr

최 재 영 (Jaeyoung Choi)
1984년: 서울대학교 제어계측공학과 (학사)
1986년: 미국 남가주대학교 컴퓨터공학 (석사)
1991년: 미국 코넬대학교 컴퓨터공학 (박사)
1992년~1994년: 미국 국립 오크리지연구소 연구원
1994년~1995년: 미국 테네시 주립대학교 연구교수
2001년~2002년: 미국 국립 수퍼컴퓨팅 응용센터 (NCSA) 초빙연구원
1995년~현재: 숭실대학교 컴퓨터학부 교수
관심분야: 시스템소프트웨어, 클라우드 컴퓨팅, 고성능컴퓨팅(HPC)
E-mail : choi@ssu.ac.kr

김 은 희 (Eunhoe Kim)
1989년 2월: 숭실대학교 전자계산학과 졸업(학사)
1996년 8월: 숭실대학교 컴퓨터학과(석사)
2006년 8월: 숭실대학교 컴퓨터학과(박사)
2007년~2009년: 숭실대학교 정보미디어기술연구소 전임연구원
2010년~2012년: 숭실대학교 지능형로봇연구소 전임연구원
2013년~현재: 서일대학교 인터넷정보과 조교수
관심분야: 분산처리, 유비쿼터스 컴퓨팅, RFID, 정보보호
E-mail : ehkimnet@ssu.ac.kr

강 기 두 (Gi-Du Gang)
1990년: 중앙대학교 경영학과 (학사)
1992년: 중앙대학교 경영학과 (석사)
1997년: 중앙대학교 경영학과 (박사)
2004년: 일리노이주립대학교 (박사)
2005년~현재: 숭실대학교 경영학부 교수
관심분야: 서비스 품질, 고객만족, 척도개발
E-mail : gikang@ssu.ac.kr