안드로이드 무료앱 관리의 취약점 분석

김재곤*, 박경용*, 김윤기*, 한성봉*, 조태남*1)
*우석대학교 정보보안학과
e-mail : amenous3@gmail.com

Analysis on Vulnerabilities of Management for Android Free Apps

Jaegon Kim*, Gyeongyong Park*, Yungi Kim*, Seongbong Han*,
Taenam Cho*
*Dept. of Information Security, Woosuk University

요 약

스마트폰의 보급률이 폭발적으로 증가하고 있으며, 모바일OS 중 가장 많이 사용되고 있는 것은 안드로이드이다. 안드로이드의 스토어에는 100만개가 넘는 앱이 존재하고, 그 중 대다수를 차지하고 있는 것이 무료 앱이다. 무료 앱은 장치 고유의 키를 사용하여 암호화 시켜 접근을 제어하는 유료 앱과 달리, 스마트폰에 설치 된 후 앱에 대한 접근 방지책이 없어 접근에 취약하다. 본 논문에서는 무료 앱 접근에 대한 취약성을 분석하고 무료앱의 파일을 보호할 수 있는 대응방안을 마련하였다.

1. 서론

스마트폰의 보급률이 기하급수적으로 증가하고 다양한 모바일 OS를 가진 스마트폰이 쏟아져 나오고 있다. 현재 시장에 나와 있는 모바일OS만 해도 안드로이드, iOS, 윈도우 폰, 심비안 등 수가지에 이른다. 그 중 가장 많은 비율을 차지하는 모바일OS는 안드로이드로서 2013년 3분기 현재 세계 스마트폰 시장의 81퍼센트 이상을 차지하고 있다. <표 1>은 안드로이드의 2012년 점유율과 2013년 점유율을 비교한 것으로서 약 6퍼센트 증가하였다.

<표 1> 세계 모바일OS 출하량과 점유율 [1]

Operating System	3Q13 Shipment Volumes	3Q13 Market Share	3Q12 Shipment Volumes	3Q12 Market Share	Year-Over-Year Change
Android	211.6	81.0%	139.9	74.9%	51.3%
iOS	33.8	12.9%	26.9	14.4%	25.6%
WinPhone	9.5	3.6%	3.7	2.0%	156.0%
BlackBerry	4.5	1.7%	7.7	4.1%	-41.6%
Others	1.7	0.6%	8.4	4.5%	-80.1%
total	**261.1**	**100%**	**186.7**	**100%**	**39.9%**

오픈소스기반의 안드로이드는 특유의 개방성으로 인해 앞으로도 모바일OS의 점유율이 계속 높아질 것으로 예상되고 수많은 사용자들의 개인성향에 맞춰 요구를 만족시킬 앱이 앞으로 더욱 많이 필요할 것이다. 그러나 점유율이 높아지면서 공격자들의 공격과 공격방식 또한 증가하고 있다. 현재 사용자들의 요구를 반영한 100만개가 넘는 수많은 앱들이 스토어에 올라와 있고, 일반 사용자들은 스토어에 올라와있는 앱을 다운받아 사용한다. 다운받아 사용할 수 있는 앱은 크게 유료앱과 무료앱으로 나눌 수 있다. 사용자들은 두 앱을 나누는 기준을 단지 결제의 유무로 생각할 수 있다. 그러나 앱이 다운로드 되어 스마트폰에 설치가 되고나면 유료앱과 무료앱에 대한 시스템 관리 방법이 달라진다. 안드로이드의 앱 관리는 Java기반으로써 각 앱마다 독립적인 공간이 주어지고 서로 접근을 제한하는 방식으로 이루어진다. 유료앱의 경우 설치 후, 기기 내부에서 해당 앱 저장소 안의 파일에 접근이 불가능 하도록 제한된다. 그러나 무료앱의 경우 아무런 제약 없이 해당 앱의 파일까지 접근이 가능하다. 이것은 결국 무료앱의 보안 체계가 취약하다는 것을 말하고 있다. 해당 파일에 접근이 가능하면 파일 안의 접근제어와 같은 안드로이드 시스템에 필요한 속성정보에도 접근할 수 있다. 이러한 안드로이드 시스템에서는 shared-id라는 속성을 통해 다른 앱의 데이터와 리소스등의 정보를 공유할 수 있다. 속성정보를 알아내면 shared-id를 공유하도록 대상 앱을 리패키징 하여 대상 앱의 정보를 공유할 수도 있다. 본 논문에서는 사용자들이 가장 많이 사용하고, 쉽게 공격에 노출될 수 있는 무료앱의 접근 취약성을 분석하고 파일을 보호할 수 있도록 취약성에 대한 대응방안을 제시한다.

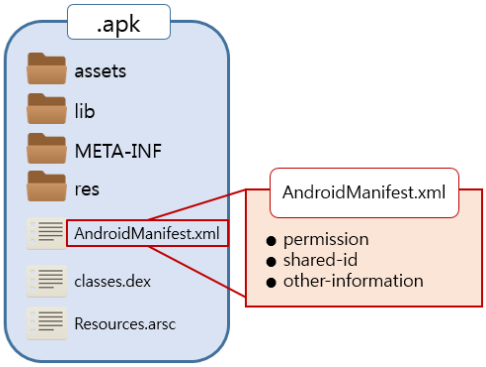
1) 교신 저자

* 이 논문은 2012년도 정부(미래창조과학부)의 재원으로 한국연구재단 일반연구자지원사업의 지원을 받아 수행된 연구임(No. 2012R1A1A3015030).

2. 안드로이드 파일구조

2.1 앱파일

안드로이드 앱은 확장자가 .apk인 압축파일로서 7-zip, Alzip등 압축 프로그램으로 내용을 확인할 수 있다. (그림 1)은 apk의 기본구조이다. assets폴더는 앱에서 사용하고자 하는 파일이 컴파일된 형태로 저장되는 폴더이다. lib 폴더는 라이브러리를 모아둔 폴더로써 각 프로세서에 맞춰서 컴파일 된 코드가 들어있다. res 폴더는 컴파일 되어 있지 않은 모든 리소스가 존재하는 폴더로서 레이아웃과 아이콘 등이 들어있다. META-INF 폴더는 코드서명과 관련되어 있는 데이터들의 폴더이다. 그 안에는 앱의 모든 파일의 해쉬값이 저장되어있는 MANIFEST.MF, 파일의 정보에 대한 해쉬값이 들어있는 CERT.SF, 그리고 실제 서명파일인 CERT.RSA 파일이 들어있다. AndroidManifest.xml은 앱을 관리하기 위한 파일로서, 퍼미션과 user-id, shared-id 등을 포함하고 있다.



(그림 1) .apk 파일구조

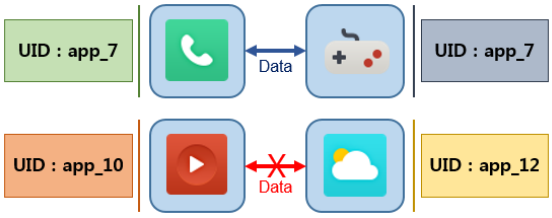
2.2 AndroidManifest.xml

AndroidManifest.xml 파일은 .apk 내 최 상위 폴더에 위치해 있다. AndroidManifest.xml은 응용 프로그램의 구성과 관련된 모든 정보를 담고 있는 파일로, 응용프로그램의 이름과 버전정보, 퍼미션, shared-id, 응용프로그램의 구성요소, 실행에 필요한 사용권한, 실행방식 등 다양한 정보를 정의한다. 그 중 퍼미션은 시스템에서 특정 파일이 특정 역할을 할 수 있게 권한을 주는 작업이다. 권한을 얻지 못하면 데이터나 시스템 서비스를 이용하지 못한다(3.3 참조). shared-id는 동일한 shared-id를 갖는 앱들을 상호 연동시켜 하나의 프로세스에서 동작시킨다. 이는 메모리와 CPU를 좀 더 효율적으로 사용할 수 있게 한다(3.4 참조).

3. 안드로이드 보안구조

3.1 샌드박스(Sandbox)

샌드박스는 리눅스의 사용자-기반 보호를 이용하여 앱 고유의 user-id(Uesr Identifier), group-id(Group Identifier)를 통해 접근제어를 한다. 앱마다 user-id가 다르기 때문에 각 앱 데이터는 서로 다른 퍼미션을 적용 받는다. 앱의 user-id가 다르면 서로의 데이터 영역에 접근할 수 없게 되고, 서로 다른 프로세스 상에서 동작을 하게 된다. 따라서 앱은 독립적인 상태에서 각각 실행된다. (그림 2)는 같은 user-id와 서로 다른 user-id를 갖는 앱들의 데이터 공유 관계를 나타낸 것이다[2].



(그림 2) 안드로이드 샌드박스

3.2 코드서명 (Code-signing)

사용자는 앱을 스토어에 등록하기 위해서 앱에 코드서명(전자 서명)을 하여 제출하여야 한다. 코드서명을 하기 위해서 개발자는 공개키/개인키 쌍을 생성해야 한다. 전자서명은 외부에 공개하는 공개키와 개인이 비밀로 소지하고 있는 개인키 한 쌍이 필요하다. 앱에 대한 서명에는 개발자가 비밀리에 소유한 유일한 개인키를 이용한다. 따라서 코드서명 값은 개인키와 앱에 유일한 값으로 생성되며, 다른 사람은 그 개발자의 서명을 생성할 수 없다. 서명의 검증을 위해서는 개발자의 공개키를 사용한다. 공개키는 대응되는 개인키와 수학적 특정 조건을 만족하는 공개되는 키로서 .apk파일의 META_INF 폴더 내에 CERT.RSA에 포함되어 배포된다. .apk 내의 공개키로 검증되는 앱은 개발자의 개인키로 서명되었음이 확인된 것이며, 무결성이 입증된 앱임을 말해준다.

3.3 퍼미션 (Permission)

안드로이드에서 다른 앱의 데이터나 시스템 서비스를 사용하기 위해서는 권한이 필요하다. 권한을 확보하는 방법은 AndroidManifest.xml에 필요한 권한을 요구하는 것이다. (그림 3)의 예에서는 네트워크를 사용하기 위한 퍼미션 ACCESS_NETWORK_STATE와 인터넷을 사용하기 위한 퍼미션 INTERNET을 요구하고 있다[3].

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.woosuk"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
```

(그림 3) AndroidManifest.xml 내 퍼미션 설정

3.4 Shared user ID

안드로이드는 백그라운드와 포어그라운드에서 여러 앱이 상호 연동하여 동작을 할 수 있도록 지원한다. 여러 개의 앱을 각각 실행하게 되면 프로세스가 너무 자주 그리고 많이 생성되어서 메모리와 CPU 사용 측면에서 비효율적이다. 그래서 상호 연동이 되는 앱들은 하나의 프로세스

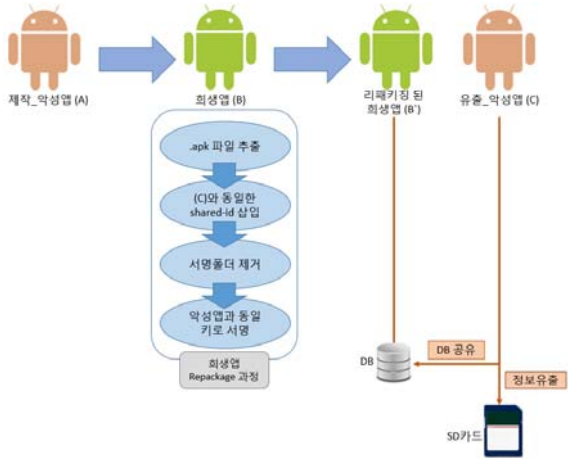
에서 동작하도록 지정하는 것이 좋다. 상호연동을 위해 AndroidManifest.xml에서 shared-id를 지정할 수 있다. shared-id 는 manifest 태그의 sharedUserId 속성을 사용한다. 보안을 유지하기 위해 동일한 shared-id를 가지는 앱만 동일한 user-id를 가질 수 있다. shared-id를 이용하면 서로 다른 앱과 같은 user-id를 공유할 수 있다. 같은 user-id를 공유하면 파일 등 데이터를 공유할 수 있고 프로세스도 공유할 수 있게 된다. 단, 앱들이 user-id를 공유하기 위해서는 그 앱들이 동일한 개인키로 서명 되어 있어야 한다[4].

4. shared-id를 악용한 악성앱

본 장에는 접근제어가 느슨한 무료앱과 shared-id를 공유하여 데이터를 유출시키는 앱을 작성함으로써 무료앱 관리의 위험성을 보인다.

4.1 악성앱 제작

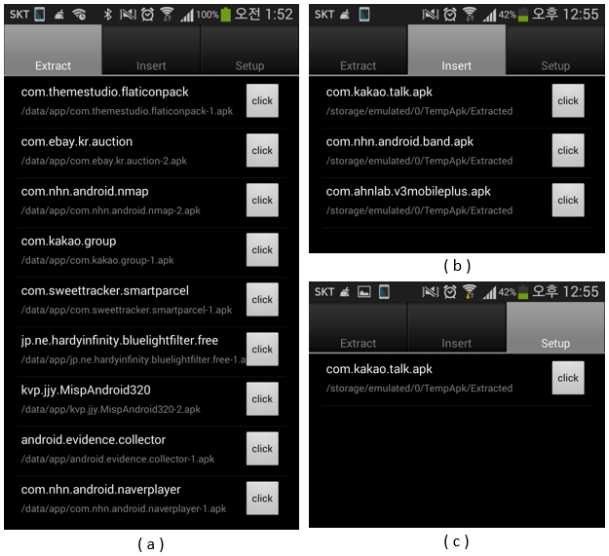
실제 악성앱을 사용자가 인지하지 못하도록 작동시키겠지만, 본 절에서는 그 과정을 보여주기 위해 사용자의 선택에 따라 희생 무료앱을 선택할 수 있도록 하는 앱을 작성하였다.



(그림 4) 악성앱 제작 과정

제작된 악성앱은 2개로서, 희생앱에 shared-id를 삽입하여 리패키징하는 제작_악성앱(A)과 리패키징된 앱으로부터 데이터를 유출시키는 유출_악성앱(C)이다. (그림 4)는 악성앱의 작성과정을 보여준다. 제작 악성앱(A)이 무료앱의 저장장소인 /data/app 폴더에 접근하여 희생앱(B)의 .apk파일을 추출한다. 희생앱(B)과 유출_악성앱(C)이 shared-id를 공유하도록 희생앱(B)의 manifest에 유출_악성앱(C)의 shared-id를 삽입한다. 희생앱(B)의 manifest 파일이 수정되었기 때문에 코드서명도 수정되어야 한다. 이를 위해 기존의 META-INF 폴더를 삭제한 후, 악성앱 개발자의 개인키로 서명함으로써 새로운 희생앱(B')으로 리패키징한다. 실제 리패키징된 앱과 shared-id를 공유한 유출_악성앱(C)은 리패키징된 희생앱(B')의 데이터를 공유할 수 있게 된다.

(그림 5)는 본 연구에서 작성한 악성앱이 다른 무료 앱 파일의 shared-id를 바꾸는 과정이다. (a) Extract탭에서 /data/app폴더 아래에 있는 모든 무료 .apk파일을 리스트로 보여준다. 사용자가 희생앱을 선택하면 해당 .apk파일을 추출한다. (b) Insert탭에서는 추출된 희생앱의 리스트를 보여준다. 그림에서는 3개의 앱이 추출되어 있다. 사용자가 앱을 선택하면 해당 .apk파일의 manifest에 악성앱의 shared-id를 삽입하고 재서명하여 리패키징한다. (c) 마지막으로 Setup탭은 리패키징이 완료된 희생앱의 리스트이다.



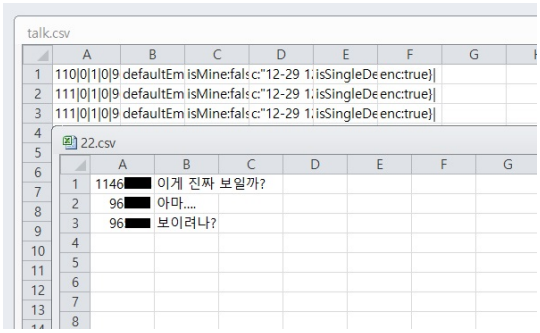
(그림 5) 악성앱 샘플

4.2 악성앱을 이용한 정보 유출

악성앱의 shared-id가 희생앱에 삽입되면 악성앱과 희생앱이 데이터와 퍼미션 등 여러 정보를 공유하게 되어 악성앱은 원하는 데이터를 유출할 수 있다. (그림 6)은 (그림 5)에서 리패키징된 희생앱을 이용한 대화 창이다.

(그림 6) 리패키징 된 희생앱 이용

(그림 7)은 (그림 5)의 리패키징된 희생앱(B′)과 shared-id를 공유한 유출_악성앱(C)이 B′의 DB로부터 대화내용을 추출하여 SD카드에 담은 내용을 복구한 화면이다. (그림 7)은 (그림 6)의 대화내용을 그대로 보여주고 있다.

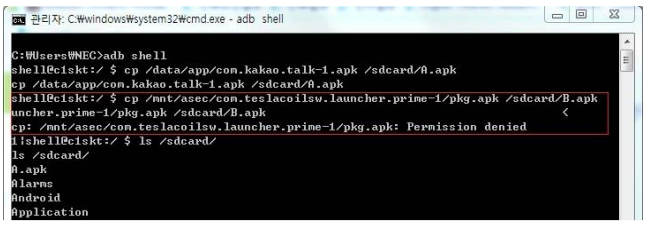


(그림 7) 공유한 DB로부터 추출된 데이터

5. 대응방안

4장에서 기술한 악성앱은 루팅 되지 않은 상태에서 작동된다. 이러한 악성앱의 제작은 안드로이드 시스템의 두 가지 취약점에 기인한다. 첫 번째는 무료앱에 대한 접근제어가 취약하다는 것이다. 현재 유료앱의 경우 안드로이드 4.1버전부터 설치할 때 유료 앱에 대한 보호 방법을 제공하고 있다. 유료앱은 기본 /data/app이 아닌 /mnt/asec에 설치, 저장이 되어 접근을 차단하기 때문에 무료앱의 저장장소보다 안전한 보호를 받고 있다(그림 8 참조). 무료앱의 경우 다운을 받아 설치했을 때 /data/app에 저장이 되어 접근에 제약을 받지 않는다. 두 번째는 악성앱이 희생앱을 리패키징 할 수 있다는 것이다[5].

두 가지 취약점에 대한 대응방안은 첫 번째, 모든 앱을 /mnt/asec에 설치하거나 통합적인 앱 설치 폴더를 만들어 앱에 대한 접근을 제한하여 유료앱 뿐만아니라 무료앱의 데이터도 안전하게 보호하도록 하는 것이다. 두 번째는 사용자 앱이 .apk 파일에 서명을 하거나 리패키징을 하지 못하도록 막는 것이다. 악성앱이 다른앱을 변경 하였을 경우, 변경된 앱을 사용자가 이용하도록 만들기 위해서는 리패키징을 필수적으로 행해야 하기 때문에 이를 원천적으로 차단해서 악성앱이 강제로 데이터를 공유하여 개인정보 유출을 하지 못하도록 하는 것이다. <표 2>는 취약점과 대응방안을 요약해 놓은 것이다.



(그림 8) /mnt/asec에 대한 접근 거부

<표 2> 취약점과 대응방안

취약점	대응방안
앱 내에서 다른앱이 shared-id 삽입, 서명 생성, 리패키징 허용	앱 내에서 서명 혹은 리패키징 불허
무료앱에 대한 접근 무제한 허용	무료앱에 대해서도 유료앱에 대한 보호기능 적용

6.결 론

안드로이드에서는 앱의 데이터 보호를 위해 각 앱마다 고유의 폴더를 사용하고 있으며, 다른 앱들은 이 폴더에 대한 접근이 제한된다. 그러나 앱 파일 자체에 대한 접근 제어는 유료앱과 무료앱에 대해 다른 방식을 취하고 있다. 저장되는 폴더가 다를 뿐 아니라, 유료앱의 경우에는 시스템 키로 암호화하여 다른 앱의 접근을 제한하고 있는 데 반해 무료앱의 경우에는 접근에 대한 제한이 없다.

본 논문에서는 이러한 약점을 이용하여 무료앱과 데이터베이스를 공유할 수 있도록 shared-id를 삽입하여 리패키징함으로써 앱을 감염시킬 수 있으며, 감염된 앱과 공유된 데이터베이스로부터 개인 정보를 유출할 수 있음을 보였다. 이러한 악용의 문제점을 해결하기 위한 방안으로서, 무료앱에 대한 접근 제어 강화와 다른 앱에 대한 리패키징 기능의 제한을 제시하였다.

참고문헌

[1] http://www.cio.com/article/743055/Android_Windows_Phone_Make_Major_Gains_in_Q3_2013.

[2] 안드로이드 보안 강화를 위한 새로운 앱 마켓 시스템/서울대학교/2013/전철/국회도서관.

[3] http://developer.android.com/reference/android/Manifest.permission.html.

[4] http://www.androidpub.com/10481.

[5] android-developers.blogspot.kr/2012/06/introducing-android-41-jelly-bean.html.