안드로이드 플랫폼의 권한 관련 보안 취약성 분석*

김익환, 김태현
서울시립대학교 기계정보공학과
e-mail: duo830210@uos.ac.kr, thkim@uos.ac.kr

Analysis of Security Vulnerabilities with Application Permissions in Android Platform

Ikhwan Kim, Taehyoun Kim
Dept of Mechanical and Information Engineering, University of Seoul

요 약

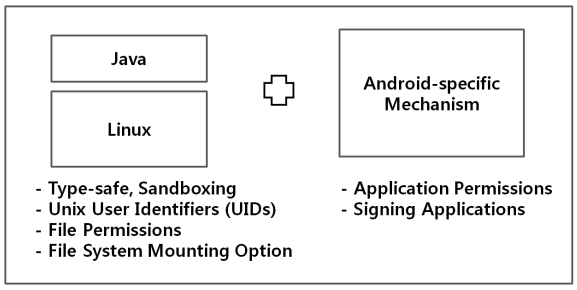
구글 안드로이드 플랫폼은 오픈소스 형태로 응용프로그램을 손쉽게 개발할 수 있는 환경을 제공하며 이러한 특징으로 인해 빠른 속도로 시장 점유율을 높이고 있다. 하지만 오픈 소스의 특징으로 인해 보안 취약점에 대한 우려가 증가하고 있다. 안드로이드 고유의 보안모델은 응용프로그램의 시스템자원에 대한 부적절한 접근을 제어하기 위한 권한을 중심으로 이루어진다. 본 연구에서는 안드로이드의 권한 기반 보안모델에 대한 취약성을 테스트 코드수행과 플랫폼 소스분석을 통해 알아보고 이에 대해 간단한 해결방안을 제시한다.

1. 서론

최근 주목받고 있는 구글 안드로이드는 오픈소스, 자바 기반의 손쉬운 응용프로그램 개발환경 제공 등의 강점으로 기존 스마트폰 플랫폼과의 경쟁에서 점차 시장 점유율을 높이고 있다. 하지만 오픈소스의 특징으로 인해 보안 취약점에 대한 우려가 높아지고 있다. 안드로이드 고유의 보안정책은 응용프로그램이 시스템자원 또는 개인정보 등에 대한 부적절한 접근을 제어하기 위한 권한을 중심으로 이루어져 있기 때문에 본 연구에서는 안드로이드 플랫폼의 기본 보안모델 중 응용프로그램 권한과 관련된 잠재적인 보안 취약점에 대한 분석을 수행하고 이에 대한 해결방안을 제시한다. 본 논문의 구성은 다음과 같다. 2장에서는 안드로이드 기본 보안모델중 하나인 응용프로그램 권한에 대해서 기술하며 3장은 권한에 기초한 보안정책에서 발생할 수 있는 잠재적인 보안 취약점에 대해 테스트 코드수행을 통해 분석한다. 마지막으로 4장에서 결론을 맺는다.

2. 안드로이드 보안모델

안드로이드 플랫폼의 기본 보안모델은 (그림 1)과 같이 기존의 자바 실행환경과 리눅스 플랫폼에서 채택하고 있는 보안정책과 응용프로그램 권한, 응용프로그램 서명 등과 같은 안드로이드 고유의 보안정책으로 구성된다. 안드로이드 고유의 보안정책에서 가장 큰 특징은 특정 응용프로그램이 내부의 다양한 데이터, 하드웨어 장치, 또는 다른 응용프로그램의 컴포넌트에 접근하려 할 때 적절한 권한(Permission)을 획득해야만 이를 허용하도록 하는 점이다 [1,2,3]. 권한은 권한취득을 통해 수행할 수 있는 작업의 특성 또는 권한을 통해 보호하고자 하는 내용을 잘 표현할 수 있도록 “android.permission.[권한명]”과 같이 특정한 문자열 형태로 정의되며, 시스템 차원에서 미리 정의된 권한과 응용프로그램 개발자가 자신의 컴포넌트 보호를 위해 새롭게 설정한 권한으로 구분된다.

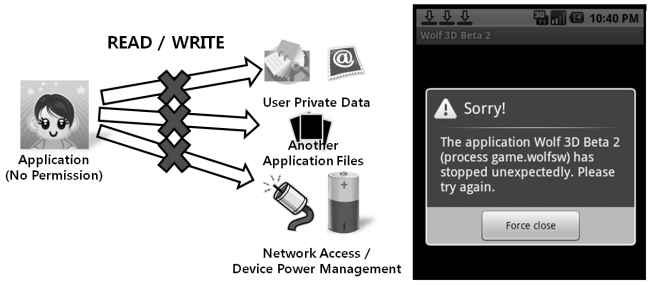


(그림 1) 안드로이드 보안모델

시스템에 의해 미리 정의된 권한들 중 응용프로그램 개발자가 사용할 수 있는 권한들은 안드로이드 공식 개발자 문서에서 확인할 수 있으며, 시스템 전반의 동작에 영향을 미칠 수 있는 중요 권한들로는 소켓을 생성하고 인터넷에 접근할 수 있는 권한을 부여하는 INTERNET, 장치의 전력관리 부분에 접근할 수 있는 DEVICE_POWER, 전화 연결을 가능하게 해주는 CALL_PHONE 등이 있다 [4]. 시스템에서 제공하는 하드웨어 자원에 접근하거나 다른 응용프로그램이 제공하는 데이터에 대한 사용권 등을 확보하기 위해서는 응용프로그램 별로 정의되는 일종의 메타 파

* 이 연구는 한국 인터넷 진흥원 정보보호 시스템 평가 사업 (2010)의 지원을 받아 수행됨.

일인 AndroidManifest.xml 파일 내에 <uses-permission> 태그를 이용해 권한명을 지정해야 한다 [4]. 만약 응용프로그램이 적절한 권한을 획득하지 못한 상태에서 시스템 자원이나 다른 응용프로그램의 컴포넌트를 접근하려고 시도할 경우, (그림 2)와 같이 보안 예외상황 (Security Exception)이 발생하여 정상적인 수행을 할 수 없다 [1,2,3].



(그림 2) 권한이 없는 응용프로그램의 보안 예외 상황

또한, 각 응용프로그램은 시스템에서 미리 정의한 권한이나 개발자가 직접 정의한 권한명들을 AndroidManifest.xml 파일의 <permission> 태그에 명시함으로써 내부에 정의된 컴포넌트나 데이터를 외부의 허가되지 않은 접근으로부터 보호할 수 있다 [4, 6]. 심지어 같은 응용프로그램에 속한 내부 컴포넌트로부터의 접근조차도 적절한 권한을 명시하지 않을 경우 허용되지 않는다. 특정 컴포넌트 혹은 데이터 접근을 위한 권한을 얻고자 하는 응용프로그램은 자신의 AndroidManifest.xml 파일 내의 <uses-permission> 태그에 해당 권한명을 명시하여야 한다. 따라서 사용자 정의 권한명의 경우, 이 권한의 이름을 모르는 다른 응용프로그램으로부터 접근은 원천적으로 차단된다 [1, 2, 3].

3. 권한 관련 보안 취약성 분석

본 연구에서는 권한과 관련된 안드로이드 플랫폼의 보안 취약성을 찾기 위해 소스코드 상의 보안관련 부분 분석과 안드로이드 SDK 버전 2.1을 탑재한 상용 단말에서의 샘플 프로그램 테스트를 수행하였다.

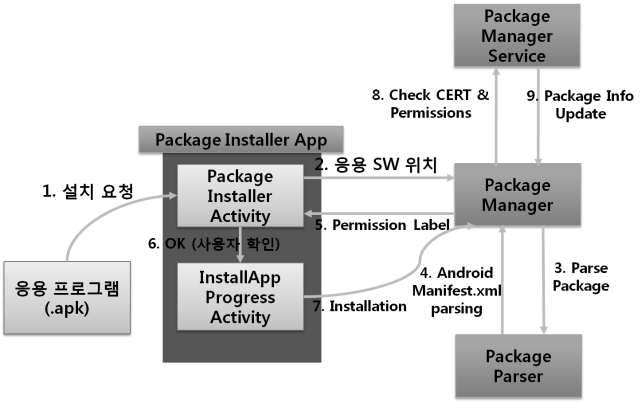
3.1 권한에 근거한 보안 점검 과정

응용프로그램이 갖고 있는 권한이 적절한지 판단하는 것은 <표 1>과 같이 응용프로그램이 설치될 때, 실행 중 다른 응용프로그램의 컴포넌트에 접근하거나 시스템 컴포넌트에 특정 작업을 수행 또는 요청할 때이다.

(그림 3)은 <표 1>의 권한 검사 지점 중 응용프로그램 설치 과정의 보안검사과정을 나타낸다. 응용프로그램이 설치될 때 사용자는 시스템 응용프로그램인 패키지 인스톨러(Package Installer)를 통해 응용프로그램이 요구하는 권한을 확인할 수 있다 [6]. 시스템 혹은 사용자가 응용프로그램이 요구하는 권한을 확인하고 설치를 허용하면 응용프로그램은 자신이 요구한 특정 기능에 대한 권한을 획득하게 되며, 패키지 매니저(Package Manager)라는 시스템 컴포넌트가 시스템 내부의 모든 응용에 대해 각각의 권한 정보를 유지한다.

<표 1> 권한확인 지점

이 름	설 명
응용프로그램 설치시	응용프로그램이 요구하는 권한이 적절한지 확인
액티비티 시작시	해당 응용프로그램이 다른 응용의 액티비티, 서비스, 브로드캐스트, 컨텐츠 제공자를 수행시킬 수 있는 권한이 있는지 확인
서비스 바인딩 또는 시작시	
브로드캐스트 송수신 시	
컨텐츠 제공자에 접근, 작업 수행 시	
시스템 내부로의 호출 시점	해당 응용프로그램이 시스템 컴포넌트나 자원을 사용할 수 있는 권한이 있는지 확인



(그림 3) 응용프로그램 설치 시 권한확인 과정

설치 후 응용프로그램 실행 과정에서 다른 응용프로그램의 컴포넌트 호출이 일어날 경우에는 시스템 서비스인 패키지 매니저의 checkPermission 메소드를 이용하여 시스템이 유지하고 있는 응용의 권한정보와 비교 후 코드 수행을 허용할지 여부를 판단하게 된다 [7].

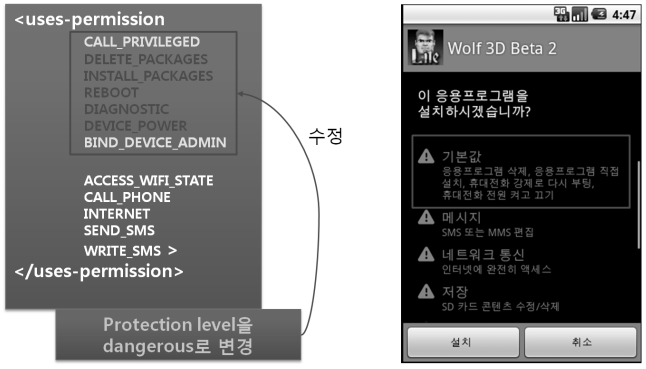
3.2 권한보호 수준 변경의 위험성

안드로이드에서는 시스템 내부에 정의된 모든 권한에 대해 시스템에 영향을 끼칠 수 있는 잠재적인 위험 수준을 나타내는 권한보호 수준(Permission Protection Level)에 따라 Normal, Dangerous, Signature, SignatureOrSystem의 4단계로 분류하고, 권한보호 수준을 응용프로그램 설치 과정에서 보안점검에 활용한다 [5,8]. Normal 레벨은 시스템에 적은 영향을 끼치는 권한에 부여된 레벨로, 설치과정에서 사용자에게 공지 없이 허용될 수 있지만, 사용자가 확인할 수 있다. Dangerous 레벨은 상대적으로 시스템에 높은 영향을 끼칠 수 있는 권한들을 나타내며 설치시 사용자에게 공지가 된다. Signature 레벨은 권한을 요구하는 응용프로그램이 이미 시스템에 설치된 해당 권한을 선언하고 있는 응용프로그램과 같은 개발자 서명이 되어 있는 경우에만 사용자에게 공지없이 권한이 부여되며

SignatureOrSystem 레벨은 하드웨어 제조사 또는 통신사에서 작성한 응용프로그램에 대한 권한에 해당된다.

본 연구에서는 특정권한이 속한 권한보호 수준 정의 내용이 변경될 경우 큰 보안 위협이 될 수 있다는 점을 소스분석을 통해 확인하고, 다음과 같은 테스트 시나리오를 통해 검증하였다. 먼저, 특정 응용프로그램의 AndroidManifest.xml 파일에 보안상 위협이 될 수 있는 강력한 권한들을 요구하는 부분을 추가한 후 실제 안드로이드 단말에 설치해 보았으며 (그림 4)는 이에 대한 실행결과이다. DDMS (Dalvik Debug Monitor Service) 툴을 사용해 (그림 4)의 실행결과에 대한 시스템 로그 메시지를 확인해 보면, 특정 권한의 경우 사용자에게 공지되지 않고 시스템레벨에서 허용이 되지 않은 것을 확인할 수 있다. 실제 소스코드 상에서 확인해 보면 시스템 컴포넌트인 패키지 매니저 서비스에서 권한보호 수준이 Normal 혹은 Dangerous 인 경우 사용자가 허가하면 해당 권한을 무조건 허용하지만, Signature와 SignatureOrSystem의 권한보호 수준을 갖는 권한들은 미리 정해진 규약에 따라 엄격하게 허용여부를 결정하도록 되어있음을 알 수 있다 [6].

(그림 4) 권한보호 수준 수정 전 응용프로그램 설치 화면



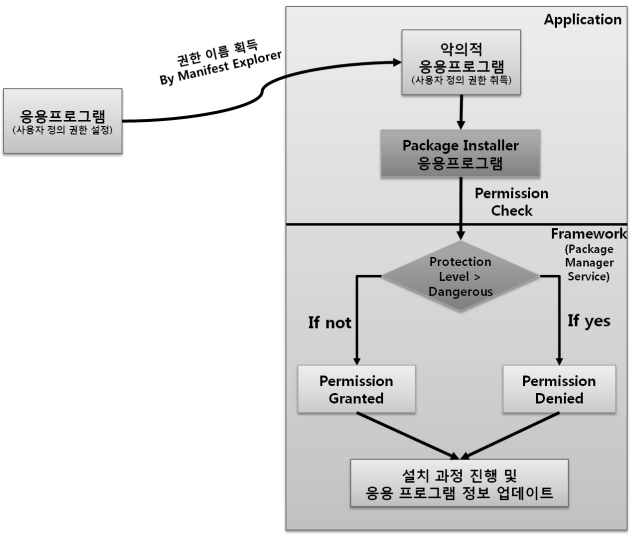
(그림 5) 권한보호 수준 수정 후 응용프로그램 설치 화면

위의 응용프로그램에 대해 안드로이드 플랫폼 소스코드를 수정하여 Signature 레벨 이상의 권한보호 수준을 가지는 강력한 권한들을 일부 Dangerous 레벨 이하로 변경할 경우의 실행결과는 (그림 5)와 같다. (그림 5)에서 볼 수 있듯이 권한보호 수준을 낮춘 것만으로 권한 허용 여부가 사용자의 판단으로 넘겨짐을 알 수 있다. 최근 시스템 최적화 및 사용자 인터페이스 레이아웃 변경을 목적으로 수정된 커스텀 펌웨어가 많이 배포되고 있는 상황에서 위와 같이 권한보호 수준이 수정된 배포된 커스텀 펌웨어을 사용하는 경우, 과도한 권한을 요구하는 응용프로그램이 권한에 전문지식이 없는 사용자에 의해 허용되어 설치될 가능성이 있다. 따라서 사용자는 응용프로그램이 요구하는 권한에 대해 관심을 가져야하며, 목적과 다른 권한을 요구하는 경우 설치에 주의를 해야 한다.

3.3 권한명 노출에 따른 문제점

권한에 관한 모든 정보는 AndroidManifest.xml 파일에 명시되어 있다. 이러한 방식의 문제점은 이 파일의 내용을 확인할 수 있다면 응용프로그램 개발자가 정의한 사용자 정의 권한명들도 쉽사리 확인할 수 있으며, 이 정보를 이용해 악의적인 응용프로그램이 권한획득을 시도할 가능성이 있다는 점이다. 실제로 Manifest Explorer [9]와 같은 툴을 사용하면, 시스템에 설치된 모든 응용프로그램들의 AndroidManifest.xml 파일의 정보를 확인할 수 있으며, 사용자가 정의한 권한들의 이름도 알 수 있게 된다.

네트워크, 배터리 등 하드웨어자원에 대한 직접적 접근이나 사용자 개인정보 등 민감한 내용에 대한 접근권한을 요구하는 응용프로그램이 사용자의 허가에 의하여 일단 설치되면 이후 추가적인 작업 없이 획득한 권한을 활용해 보안상 위협적인 동작을 수행할 가능성이 크다. 특히 획득하고자 하는 권한명만 알 수 있다면 응용프로그램이 권한을 요구하는 데는 제약이 없기 때문에 권한이란 개념에 관한 지식이 없거나 관심이 없는 사용자에게 있어서 보안 위협이 될 수 있다.



(그림 6) 권한 보호 수준 검사과정

하지만 3.2절에서 설명한 것과 같이 적절한 권한보호 수준 설정이 되어 있다면, 악의적인 응용프로그램이 다른 응용프로그램이나 시스템 서비스를 접근하기 위한 권한관련 정보를 획득하더라도 (그림 6)과 같이 시스템의 패키지 설치관리 부분에서 필터링이 된다. 따라서 패키지 보호의 목적으로 직접 권한을 정의할 경우 Signature 레벨 이상의 권한보호 수준을 가지도록 작성하는 것이 필요하다.

3.4 사용자 ID 공유의 문제점

안드로이드는 동일한 개발자가 작성한 응용프로그램 사이의 상호협업을 위한 사용자 ID 공유(Shared User ID)를 지원한다. 이는 동일한 시스템에 설치될 경우 같은 사용자 ID를 부여하고 같은 가상머신 상에서 구동을 하게 하는 기법이다. 하지만 사용자 ID 공유를 사용할 경우 각 응용프로그램이 가지고 있는 권한까지 공유되는 현상이 발생한다. 이는 안드로이드의 사용자 ID가 응용프로그램을 구분하는 식별자로도 사용되기 때문이다.

(그림 7)은 사용자 ID 공유에 따른 문제점을 보여 주는 테스트 시나리오의 실행결과이다. 실험에서는 먼저 아무런 권한이 없는 "suid"라는 응용프로그램을 만들어 설치를 한 후, 다양한 권한을 갖고 있는 프로그램 "suid2"라는 프로그램을 만들어 추가로 설치하였으며, 이 두 개의 프로그램은 사용자 ID 공유 옵션이 설정된 상태이다. 설치를 완료한 후에 응용프로그램의 정보를 살펴보면 아무런 권한이 없는 "suid"라는 프로그램에 "suid2"의 권한이 모두 공유된 것을 확인할 수 있다.

(그림 7) 사용자 ID 공유 후 응용프로그램 권한의 변화

만약 악의적인 개발자가 강력한 권한을 사용하는 정상적인 프로그램을 먼저 배포한 후 특별한 권한을 요구하지 않는 악성코드를 배포하여 사용자 ID 공유에 따른 사용자 정보 유출 등의 보안침해 시도를 한다면 일반 사용자 입장에서는 이를 알기 어렵다. 특히 설치과정에서 사용자 ID 공유가 이루어져 발생할 수 있는 보안 취약점에 대해 사용자에게 어떠한 공지도 없다는 점 또한 큰 문제점이다. 따라서 설치과정에서 사용자 ID 공유옵션이 설정된 응용프로그램에 대한 공지와 사용자 ID 공유에 따라 추가로 습득되는 권한에 대한 공지가 있도록 안드로이드 플랫폼을 수정하는 것이 유일한 대안으로 생각된다. 물론 사용자 편의성이 저하될 수 있고 일반 사용자가 ID 공유에 따른 보안 취약성에 대한 이해도가 낮다는 점에서 한계는 존재한다.

4. 결론

본 연구에서는 안드로이드 고유의 보안모델인 응용프로그램 권한의 취약성에 대해 플랫폼 소스코드와 테스트 코드를 상용 단말에서 수행하여 확인해 보았다. 응용프로그램에게 권한을 부여하는 것은 사용자의 선택에 따라 달려있지만 설치시 시스템 차원에서 권한보호 수준에 의해 악의적인 행동을 미연에 방지할 수 있다는 점을 확인했다. 하지만 사용자 ID 공유가 설정될 때 응용프로그램 간 권한이 공유되는 문제로 인한 보안취약성이 존재하므로 이 부분에 대한 플랫폼 수정이 필요하다. 추후 연구로는 사용자 공유 ID 설정시 권한이 사용자 공지 없이 부여되는 문제에 대해 플랫폼을 수정하는 방안을 고려하고 있다.

참고문헌

[1] J. Burns, "Developing Secure Mobile Applications for Android: An introduction to making secure Android applications", http://isecpartners.com/files/iSEC_Securing_AndroidApps.pdf, 2008.

[2] A. Shabtai, Y. Fledel and U. Kanonov, Y. Elovii, and S. Dolev, "Google Android: A State-of-the-Art Review of Security Mechanisms", IEEE Security and Privacy, vol. 8, issue 2, pp. 35-44, 2010.

[3] Google Android Developers Guide, "Security and Permissions", http://developer.android.com/guide/topics/security/security.html, Aug. 2010.

[4] Google Android Developers Official Site: References. "Manifest.permission", http://developer.android.com/reference/android/Manifest.permission.html, Aug. 2010.

[5] Google Android Developers Guide, "<permission>", http://developer.android.com/guide/topics/manifest/permission-element.html, Aug. 2010.

[6] Android SDK 2.1 Full Source, "Android Basic Application: Package Installer", <android_full_src>/packages/apps/PackageInstaller/src, March 2010.

[7] Android SDK 2.1 Full Source, "Android Application Framework: PackageManagerService", <android_full_source>/frameworks/base/services/java/com/android/server/PackageManagerService.java, March 2010.

[8] Android SDK 2.1 Full Source, "Android System Manifest File", <android_full_src>/framework/base/core/res/AndroidManifest.xml, March 2010.

[9] iSEC Partners, "Manifest Explorer", https://www.isecpartners.com/manifest_explorer.html, 2009.