안드로이드 플랫폼 인텐트 메커니즘 보안 취약성 분석*

박수용○ 김익환 김태현
서울시립대학교 기계정보공학과
sypark.moss@gmail.com, duo830210@uos.ac.kr, thkim@uos.ac.kr

Analysis of Security Vulnerabilities in Intent Mechanism of Android Platform

Sooyong Park○ Ikhwan Kim Taehyoun Kim
Department of Mechanical and Information Engineering, University of Seoul

최근 모바일 시장의 급성장과 모바일 네트워크 접속 기능을 갖춘 스마트폰에 대한 관심과 사용이 증가하고 있다. 이에 따라 스마트폰 보안에 대한 관심도 높아지고 있지만 기존 PC 사용 환경과 다른 특성으로 인해 다양한 보안 대책 또는 솔루션이 아직 부족한 상황이다. 본 연구에서는 최근 주목받고 있는 안드로이드 플랫폼 상에서 응용 프로그램 간 통신 수단인 인텐트(Intent)와 인텐트 필터(Intent Filter)의 동작 특성을 살펴보고 소스 레벨 분석과 테스트 시나리오 수행을 통하여 발견된 보안 취약점들과 이에 대한 대응 방안을 제시하였다. 안드로이드는 구글에서 발표한 리눅스 기반의 개방형 모바일 플랫폼이다[1,2]. 안드로이드 응용프로그램은 액티비티, 서비스, 컨텐트 프로바이더, 브로드캐스트 리시버의 네 가지 컴포넌트로 이루어진다. 그리고 각각의 응용프로그램은 AndroidManifest.xml 파일을 이용하여 각 응용프로그램의 컴포넌트의 특성과 보안 사항을 기술한다. 컴포넌트 간 통신에 사용되는 인텐트 메커니즘은 별도의 보안메커니즘 없이 안드로이드의 보안 메커니즘을 이용한다. 인텐트 메커니즘은 그림 1과 같이 컴포넌트 간 별도의 메소드를 이용하여 컴포넌트간 통신을 한다[3].

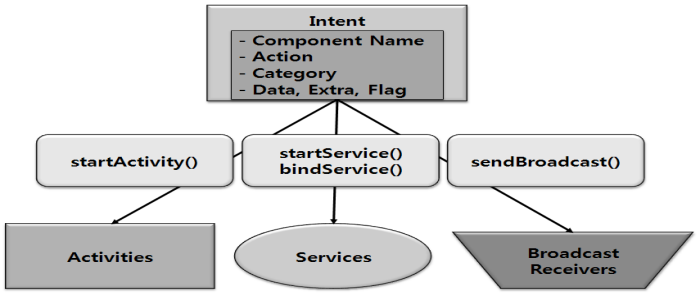

그림 1. 컴포넌트 종류에 따른 인텐트 전송 메소드

안드로이드 시스템은 인텐트를 전달할 때 인텐트 라우팅을 통하여 인텐트 수신자로 가장 적절한 컴포넌트를 찾게 된다. 라우팅 과정은 응용 프로그램 프레임워크 계층의 액티비티 매니저(Activity Manager)와 패키지 매니저(Package Manager)가 연관되어 있다[4,5]. 인텐트 라우팅은 인텐트의 종류에 따라 명시적 라우팅(Explicit Routing)과 암시적 라우팅(Implicit Routing)으로 구분할 수 있다. 명시적 라우팅은 인텐트 객체에 포함된 컴포넌트 이름을 이용하여 수신 컴포넌트를 찾는 것이다. 암시적 라우팅은 인텐트 객체에 포함된 액션과 데이터, 카테고리와 인텐트 필터와 비교를 통해 수신 가능한 컴포넌트를 찾는 것이다. 3가지 항목 비교를 전부 통과해야만 전달이 되며 만약 비교과정을 통해 적절한 수신 컴포넌트가 존재한다면 액티비티 매니저가 인텐트를 대신 전달하게 된다. 이 때 일종의 보안관련 체크를 하게 된다. 암시적 인텐트 라우팅의 경우 수신을 원하는 컴포넌트가 여러 개일 수 있다. 이러한 수신자 경합상황을 인텐트 리졸브(Intent Resolve)라 한다. 수신자 경합상황에서는 안드로이드 시스템에서 최종 수신 컴포넌트를 결정하지 않고 리졸버액티비티(ResolverActivity)를 구동시켜 사용자의 입력을 받아 최종 수신 컴포넌트를 결정한다.

일반적으로 수신자 경합상황이 발생되지 않는 경우에 안드로이드 보안 메커니즘은 정상적으로 작동한다. 권한을 설정한 컴포넌트는 적절한 권한을 보유 하지 않는 컴포넌트가 접근을 할 수 없으며, 컴포넌트 노출을 불허

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

(exported=false) 하게 설정한 컴포넌트는 외부에서 접근을 할 수 없다. 하지만 수신자 경합 상황이 발생하여 리졸버 액티비티가 구동되어 사용자 입력으로 수신 컴포넌트를 결정하는 경우는 안드로이드의 모든 보안 설정이 무시가 된다. 이러한 현상의 근본적인 원인은 리졸버액티비티를 통하여 인텐트가 전달될 때 그림 2와 같이 인텐트 송신자의 사용자 ID가 시스템 사용자 ID로 변경되어 전달되며, 이에 따라 인텐트 전송 과정의 보안 검사를 거치지 않기 때문이라는 것을 소스 분석을 통해 파악했다[4]. 수신자 경합상황에서는 비록 사용자 선택에 의해 수신자가 결정되더라도 내부적으로 리졸버액티비티라는 시스템 프로세스가 startActivity() 메소드를 이용하여 동일한 내용의 새로운 인텐트를 전송하기 때문이다.

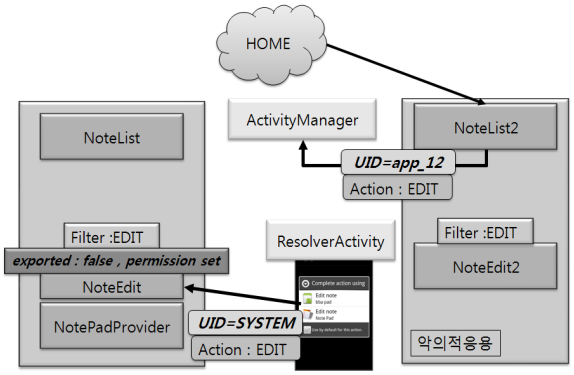

그림 2. ResolverActivity에 의한 인텐트 전달

서비스 컴포넌트의 인텐트 수신자 경합 상황은 액티비티의 경우와는 다른 문제가 있다. 서비스 컴포넌트의 인텐트 수신자 경합 상황에서는 사용자가 인지할 수 있는 별도의 확인 과정이 존재하지 않는다. 이러한 경합상황이 발생되면 무조건 먼저 설치되어 있는 서비스 컴포넌트에게 인텐트가 전달된다. 더욱 심각한 문제점은 서비스 컴포넌트가 오동작을 할 수 있음에도 불구하고 시스템 차원에서 사용자에게 공지하는 절차가 없다는 점이다.

위에서 제기한 문제점에 대한 대응 방안으로 사용자, 응용프로그램 개발자, 플랫폼 개발자 세 측면에서 고려할 부분이 있다. 사용자는 리졸버 액티비티를 통한 선택의 결과에 대한 최종 책임이 본인에게 있음을 인지하고 신중하게 대응을 해야 한다. 응용프로그램 개발자는 자신이 작성한 응용 프로그램에서 암시적 인텐트를 사용함에 있어 주의를 해야 한다. 액티비티 컴포넌트의 경우 충분히 보안사항을 설정하더라도 인텐트 수신 경합 상황에서는 보안 점검 과정이 생략될 수 있다는 점을 숙지해야 한다. 서비스 컴포넌트는 의도치 않은 서비스가 호출이 될 수 있음을 명시해야 하고 공유를 원치 않을 경우 권한 설정에 유의하고 AndroidManifest.xml 파일의 컴포넌트 노출과 관련된 속성인 exported를 false로 설정할 필요가 있다. 그리고 명시적 인텐트를 사용하는 것을 고려해야 한다. 마지막으로 플랫폼 개발자 입장에서는 리졸버 액티비티를 통한 인텐트 전송 과정에서 인텐트 송신자의 사용자 ID를 변경하지 않도록 수정하거나 리졸버 액티비티를 통한 인텐트 전송 시 원래 송신자의 정보를 확인해 보안 점검 과정을 거치도록 하는 방법이 있을 수 있다. 응용 프로그램 설치 과정에서 동일한 인텐트 필터를 가지는 서비스 컴포넌트가 존재하는 경우 사용자에게 공지하여 설치할 수 있도록 안드로이드 플랫폼을 수정하는 것도 한 가지 방법이 될 수 있다.

본 연구에서는 안드로이드의 인텐트와 인텐트 필터 메커니즘의 취약성을 소스 분석과 테스트 시나리오 실행을 통해 알아보았으며 그에 대한 대비책을 제안했다. 추후연구로는 대비책을 기반으로 플랫폼을 수정하여 근본적인 문제점을 방지하는 방안을 고려하고 있다.

참고문헌

[1] Google Android Developers Guide, "What is Android ?“, http://developer.android.com /guide/basics/what-is-android.html, Aug. 2010.
[2] 2008 Google I/O Session, "Anatomy and Physiology of an Android", http://sites.google.com/site/io/anatomy--physiology-of-an-android, 2008
[3] Google Android Developers Guide, "Application Fundamentals: Application Components", http://developer.android.com/guide/topics/fundamentals.html, Aug. 2010.
[4] Android SDK 2.1 Full Source, "Application Frameworks: ActivityManagerService", <android_full_source>/frameworks/base/services/java/com/android/server/am/ActivityManagerService.java,March. 2010.
[5] Android SDK 2.1 Full Source, "Android Application Framework: PackageManagerService", <android_full_source>/frameworks/base/services/java/com/android/server/PackageManagerService.java,March 2010.