http://dx.doi.org/10.5392/JKCA.2013.13.11.533

안드로이드 플랫폼을 위한 통합적인 사용자 인증 관리 모델

IU_AMDroid : An Integrated User Authority Manager Model for the Android Platform

남춘성*, 장경수**, 신동렬***
성균관대학교 인터랙션 사이언스 연구소*, 경인여자대학교 영상방송정보**, 성균관대학교 컴퓨터공학과***,

Choon-Sung Nam(namgun99@skku.edu)*, Kyung-Soo Jang(ksjang@kic.ac.kr)**,
Dong-Ryeol Shin(drshin@skku.edu)***

요약

안드로이드 플랫폼은 현재 단일 사용자 위주의 보호만을 지원하기 때문에 다중 사용자를 위한 보안 방법이 필요하다. 특히, 개인정보 및 금융정보가 담긴 어플리케이션과 같은 특정 앱 뿐만 아니라 콘텐츠 접근을 위한 권한 관리가 필요하다. 따라서 본 논문은 안드로이드 플랫폼 기반 디바이스를 대상으로 통합된 사용자 인증 매니저 모델을 제안한다. 이를 통해 어플리케이션의 실행, 설치, 삭제를 세분화하여 관리할 수 있는 인증 매니저 모델을 제시한다.

■ 중심어 : | 안드로이드 | 사용자 인증 | 다중 사용자 | 어플리케이션 |

Abstract

Currently, as the Android platform only supports single-user protection, it needs security solution for multi-users. Specially, it has to protect specific applications which have personal and financial information, and be available to support authority management for contents access. Thus, this paper proposes an integrated user authority manger model for the Android platform. It helps application authority which is capable to divide into three statuses: installation, execution, deletion with the help of information technology

■ keyword : | Android | User Authentication | Multi-users | Application |

I. 서 론

최근에 스마트폰(Smart Phone)은 전통적인 컴퓨터에서 할 수 있는 서비스 어플리케이션 (Service Applicaton) 및 사용자의 이동성을 고려한 서비스 어플리케이션을 사용자가 쉽게 설치 및 실행할 수 있는 기기로서 많이 사용되고 있고 개발되고 있다[1]. 가장 대중적인 플랫폼인 안드로이드(Android)[2]는 구글(Google)의 주도하에 개방 플랫폼을 지향한다. 또한, 안드로이드는 휴대폰 제조사 및 소비자들에게 비용절감 및 유연성을 제공하고 개발자들에게 편의를 도모하기 위한 OHA(Open HandSet Alliance)[3]를 출범하여 최초의 완전하고 개방적이면서 또한 무료인 모바일 플랫폼(Mobile Platform)이다. 안드로이드는 기본적으로 운영체제, 미들웨어(Middleware), 웹 브라우저(Web Browser), 전화다이얼, 알람, 연락처와 같은 중요 어플리케이션의 집합으로 이우러진 소프트웨어이다. 개발자들은 네트워크(Network) 가능 상태에서 데이터를 지

* 본 논문은 산업통상자원부 산업융합원천기술개발사업으로 지원된 연구결과입니다.[10043388, 인지 UX 기반 사용자 친화형 Natural 멀티터치/멀티포스 기술 개발]
접수일자 : 2013년 08월 01일
수정일자 : 2013년 10월 02일
심사완료일 : 2013년 10월 04일
교신저자 : 신동렬, e-mail : drshin@skku.edu

원 및 미디어 스토어 접속을 위한 GPS나 온라인 비디오(On-line Video)와 맵(Map)과 같은 추가적인 기능들을 제공하기 위해서 구글에서 제공하는 자바 기반의 개발 어플리케이션 프레임워크를 이용해야만 한다. 안드로이드 앱의 배포를 위해서 구글은 개발자들에게 모바일 디바이스의 기능을 확장할 수 있도록 어플리케이션 프로그램을 제작하여 배포할 수 있는 마켓을 제공한다. 즉, 서드파티(Third party)를 위한 안드로이드 마켓을 운영하고 있으며 이를 통해 안드로이드 기반의 모바일 디바이스 사용자들은 어플리케이션을 다운받고 설치할 수 있다. AndroLib[4]의 보고서에 의하면 현재 약 200,000개 이상의 어플리케이션들이 존재한다. 이러한 안드로이드 마켓은 개발자들에게 개방형 자료 개발 환경을 제공한다[5].

안드로이드 플랫폼 기반의 모바일 기기에서 개개인의 정보를 보호하기 위한 문제가 발생한다. 모바일 기기를 타인에게 빌려주는 경우 예를 들어 게임을 하기 위해 아이에게 넘겨주거나, 회사 업무용 대여폰을 지급하는 경우 등 실제 모바일 기기의 주인이 아닌 대여자가 임의로 앱 및 설정과 같은 개인화된 데이터를 접근할 수 있기 때문이다. 따라서 현재의 안드로이드 플랫폼은 단일 사용자 위주의 보호만을 지원하기 때문에 다중 사용자를 위한 보안 방법 즉, 다중 사용자와의 스마트폰 공유 시 발행할 수 있는 개인사용 정보 및 데이터를 보호할 수 있는 방법에 대해서 연구되고 있다[6][7]. 이러한 개인 정보는 사용자 별 개인의 신상 정보, 웹을 통한 접속 정보에 관해서 보호해야만 하고, 악의적인 어플리케이션 설치 혹은 중요 어플리케이션의 삭제와 같은 행위를 사단할 수 있어야 한다. 하지만 스마트폰 OS(Operating System)들은 오직 단일 사용자만을 지원하고, 개인 정보 보호를 위한 사용자 관리 권한에 대해서는 지원하지 않고 있다. 즉, 권한을 가진 사용자가 타 사용자에게 의해 중요한 어플리케이션 및 개인 정보 접근을 막기 위한 프로그램이 필요한 것이다. 따라서 우리는 기존의 데스크탑(Desktop) 및 서버(Server)의 OS에서 제공하는 전통적인 다중사용자 관리를 기반으로 안드로이드 플랫폼 가반에 적용 가능한 통합된 사용자 인증 매니저 모델을 제안한다.

본 논문의 구성은 2장에서는 안드로이드 플랫폼 기반에 사용되고 있는 기존의 관련연구에 대해서 기술하고, 3장에서는 제안하는 통합 사용자 권한 관리 모델을 제안한다. 4장에서는 제안하는 모델의 성능평가에 대해서 말하고, 5장에서는 결론을 맺는다.

II. 관련연구

안드로이드 플랫폼 기반 스마트폰 애플리케이션 접속제어 방법에 대한 연구는 악성앱 및 권한사용자를 가장한 접근 방법이다[8][9]. 즉, 거의 대부분의 접속제어 방법은 스마트폰 내에 설치된 어플리케이션에 대한 권한 할당 및 검증을 중점적으로 하고 있다. Apex[10][11]는 안드로이드의 자원사용 제어 모델이다. 이는 기존의 안드로이드 권한 프레임 워크를 확장한 포괄적인 정책 집행 메커니즘(mechanism)이다. 즉, 사용자가 애플리케이션으로 부터 중요한 자원 사용을 제한하는 방법으로 실행시간에 대한 제약을 지정함으로서 사용자 권한 관리 모델은 다른 사용자들로부터 개인정보와 같은 민감한 자원의 제한된 사용을 디바이스 사용자로 부터 보호할 수 있다. Apex 정책 메커니즘은 애플리케이션 권한이 실행되는 것을 체크할 때 악성 소프트웨어에서 훔치려하는 사용자 정보를 보호하지만 여전히 스마트폰의 사용자가 다수의 유저일 때 사용자의 권한 체크를 도와주지 못한다.

스마트폰 운영체제들은 스마트폰 주인인 한명의 사용자만 지원하고 기본적으로 스마트폰과 같은 모바일 디바이스의 근본적인 기능은 전화이다. AT&T 사는 Smart Limits for Wireless[12]라 불리는 서비스를 제공하는데, 어린이와 같은 어린 사용자의 스마트폰 자원 사용을 콘텐츠 필터로 제한하기 위해서 매달 스마트폰의 사용제한과 하루의 시간을 제한하는 것을 관리한다. 그러나 사용자들은 이러한 서비스를 위해 돈을 지불해야할 뿐 아니라 AT&T사의 서비스에만 가능하다. Xudong은 안드로이드 운영체제에서 DiffUser[6]라는 차별화된 사용자 접속 제어 모델을 발표했는데 그것은 다중사용자를 지원하는 것에 초점을 두었다. DiffUser

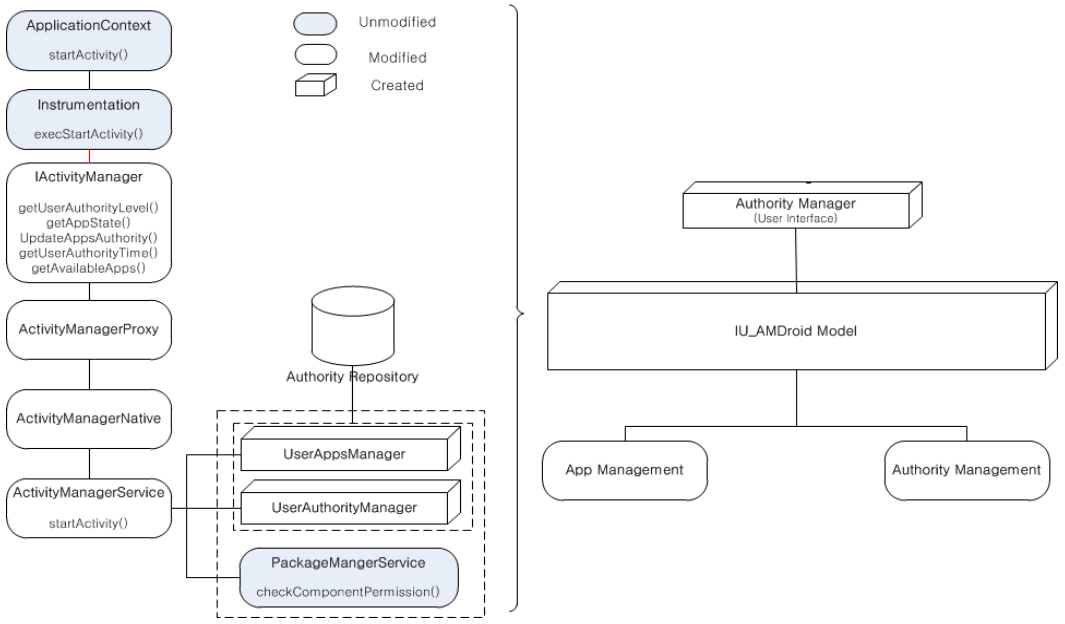

그림 1. 통합적 사용자 권한 관리 모델

는 몇몇의 클래스들(Class)로 스마트폰 유저들을 관리자, 일반 사용자, 게스트(guest), 기타 사용자들과 같이 사용자의 접속권한을 특정 사용자 그룹기반으로 분류하였다. 한 사용자가 그 시스템을 로그인할 때 홈 스크린(Home Screen) 정보는 다른 사용자의 다른 클래스를 참조하여 다른 접속 가능한 애플리케이션을 만들어 자동적으로 재설정할 수 있다. 이는 'Build-in permission-based security model'에서 작동하고, 홈 액티비티(Home Activity)를 단계에 맞게 제공하는 특징을 가지고 있다. 하지만, 최고 단계를 가진 사람 외에 어플리케이션을 설치, 제거, 실행하는 것에 구애를 받지 않는다. 이는 DifferUser는 사용자 단계별로 지정하기 위해서 각 사용자의 로그온(Log-on) 방식의 인증을 통해서만 이용 가능하다는 것이다. 따라서 DifferUser는 다중 사용자 환경을 제공할 수는 있지만, 어플리케이션의 삭제, 설치 및 실행에 대해서 모든 권한을 주기 때문에 특정 어플리케이션에 대한 보호가 불가능하다.

UAMDroid[7]는 사용자를 단계별로 분류하지 하되, 각 어플리케이션에 대해서 각각 인증을 받는 방법이다. 이는 'Build-in permission-based security model' 상위에서 작동하고, 각각의 단계별로 홈 액티비티를 맞춰줄 필요가 없다. 따라서 각 어플리케이션의 작동 및 설치, 제거를 인증받기 때문에 어플리케이션 및 보안에 대한 장점을 갖는다. 즉, UAMDroid는 어플리케이션 각각에 대해서만 인증이 가능하다는 것이다. 하지만 UAMDroid도 DifferUser에 비해 특정 어플리케이션에 대한 보호가 가능하지만, 지속적이거나 혹은 여러 개의 어플리케이션을 동시에 운용할 때 권한자도 항상 인증을 받아야 하는 불편함이 있고, 인증을 한 다음에 자신이 허락한 사용자가 사용할 경우에도 실행 중인 어플리케이션에 대한 인증을 꾸준히 받아야 하는 단점을 지닌다. 따라서 위와 같은 문제 때문에 사용자는 두 가지의 경우에 인증을 유동적으로 전환할 수 있어야만 한다. 첫째, 로그온(Log-on : 인증을 이미 받은 경우)시 인증에 대한 사용시간을 설정하거나 지속적인 인증을 위한 권한 시간을 설정할 필요성이 있다. 두 번째, 로그오프(Log-off: 인증을 미리 받지 못한 경우)시 어플리케이션의 첫 인증을 통해 어플리케이션 하나 혹은 다수의 인증을 설정할 수 있어야 하고, 인증 지속 시간 또한 설정해야 한다. 이는 시간설정 및 다수의 인증 그리고 권한자의 어플리케이션별 권한 설정이 필요하다. 따라서 본 논문은 위의 두 가지 경우를 충족할 수 있는 통합적

인 사용자 권한 관리 모델을 제안한다.

III. 통합적인 사용자 권한 관리 모델

안드로이드 플랫폼 디바이스 즉, 스마트폰을 권한관리자(소유자가) 다른 사용자에게 빌려 주는 경우 핵심 어플리케이션의 제거와 악성 어플리케이션의 설치를 제한 할 수 있어야 한다. 이를 위해 기존에 존재하는 안드로이드 보안 모델들[6][7]을 기반으로 장점을 보완하고 단점을 충족할 수 있는 보안 방법이 필요하다. 즉, 이를 둘 간의 장점을 위한 통합적인 사용자 권한 관리 모델이 필요하다. 이를 위해 우리는 인터페이스 수정 및 추가로 인한 IU_AMDroid의 아키텍처는 [그림 1]과 같이 제안한다.

안드로이드 어플리케이션은 필요에 의해 자신을 실행시킬 수 있는 컴포넌트들로 구성된다. 액티비티들은 사용자가 착수할 수 있는 비저블 인터페이스(Visible Interface)를 제공할 뿐 아니라 어플리케이션이 시작할 수 있는 엔트리 포인트(Entry Point)를 제공한다. 일반적인 어플리케이션 실행 과정 및 개념은 다음과 같이 어플리케이션 실행전과 어플리케이션 실행 후로 나뉠 수 있다.

1. 어플리케이션 실행 단계

1.1 어플리케이션 실행 전

어플리케이션은 AndroidManifest.xml 파일을 통해 어플리케이션 전체 구조 및 관련 정보를 확인할 수 있다. 특히, 권한 관리를 위해서 어플리케이션 컴포넌트 및 권한(<uses-permission/>)을 확인한다. 응용 프로그램의 구성과 관련된 모든 정보를 담고 있는 파일. 응용 프로그램의 이름과 버전 정보, 응용 프로그램의 구성요소들, 실행에 필요한 사용권한, 실행 방식 등 다양한 정보를 정의한 Manifest파일 내, 인터넷 필터(Intent Filter)는 안드로이드에게 인텐트 컴포넌트(Intent Component)가 다룰 수 있는 인텐트의 종류를 알려준다. 인텐트 필터내 Android.intent.action- MAIN의 액션(Action)과 Android.intent.category.- LAUNCHER의 카테고리(Category) 부분은 어플리케이션이 실행될 때 동시에 시작한다.

1.2 어플리케이션 실행 시

인텐트 오브젝트(Intent Object)는 액티비티가 시작될 때 ApplicationContext 클래스의 start Activity ()으로 필요한 정보와 함께 전송된다. 다시 인텐트 오브젝트는 인스트럼멘테이션(Instrumentation) 클래스의 execStartActivity()(시스템과 어플리케이션 사이의 모든 interaction을 모니터링)으로 전송된다. 이는 [그림 2]와 같다. 두 개의 독립적인 프로세스는 안드로이드에 의해 제공하는 특화된 IPC(Inter-Process Communication) 기법을 통해 실행 할 수 있다.

[그림 3]에서 ActivityManagerNative와 Antivity ManagerProxy 둘 다 Android IPC의 작업공간인 Binders와 Parcels의 개념을 사용한다. Binder는 동시 발생하는 메소드 호출 같은 간단한 기능을 제공하는 IBinder 인터페이스를 실행한다. Binder위의 Transact() 메소드를 호출하는 것은 클라이언트와 실행자 사이의 IPC 메시지를 전송하는 java.os.Parcel 인스턴스를 전송할 수 있다. ActivityManagerProxy는 목표된 인텐트에 대한 정보를 가진 파셀(Parcel)을 생성하고 그것을 ActivityManagerNative로 보낸다.

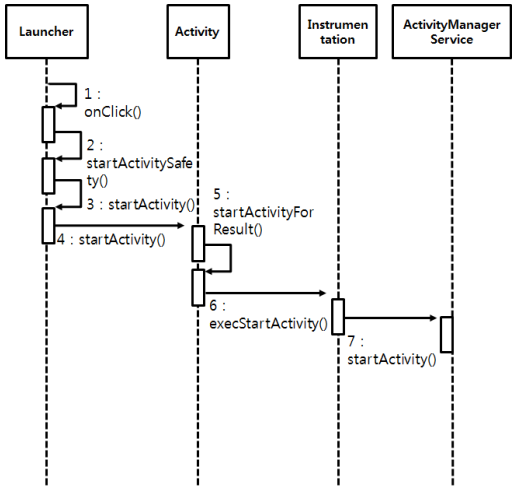

그림 2. ActivityMangerService 호출과정

그러면 파셀로부터 추출된 아규먼트(Argument)들은 액티비티를 관리하는 ActivityManagerService의 startActivity() 메소드로 이동한다. 우선 어플리케이션의 활동 같은 첫 번째 컴포넌트가 성공적으로 실행되면, ActivityManagerService는 디폴트(Default)로 기본 프로세스 이름과 같은 패키지 이름을 갖는 어플리케이션을 위한 프로세스를 생성한다. Activity-ManagerNative와 ActivityManagerService가 실행하는 IActivityManager는 클라이언트에게 노출되어진다. 그러므로 인스트럼멘테이션은 인텐트를 IPC 기법을 통해 간단히 메소드를 호출함으로써 Activity-ManageService로 이동시킬 수 있다.

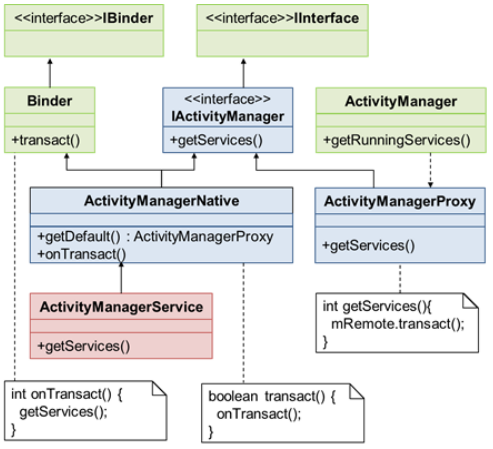

그림 3. ActivityManagerService 클래스 구조도

위와 같은 어플리케이션 실행에서 인증을 위한 어플리케이션 수정 방법은 기존의 UAMdroid[7]의 방법과 유사하게 설계 되었지만, 어플리케이션 타임라인 설정과 다수의 어플리케이션 권한 설정 과정이 포함되어 디자인을 할 수 있다. 응용프로그램의 실행을 관리하기 위해서 우리는 ActivityManagerService 클래스를 수정하고 start-Activity() 메소드 안에 우리의 UserAuthorityManager 및 UserAppsManager로 연결하는 고리를 위치시켰다.

UserAuthorityManager는 처음에 그 액션과 인텐트가 올라오는 카테고리가 ACTION_MAIN과 CATEGORY_LAUNCHER인지 각각 확인을 한다. 만약 이것이 맞는 다면, 권한 결정자(Resolver)를 호출하고 어플리케이션 결정자도 호출된다. 만약 맞지 않는다면, UserAuthorityManager는 어떠한 일도 하지 않지만, ActivityManagerService는 계속해서 자신의 작업을 수행한다. 그 외 AntivityManager- Service가 계속 자기 일을 하게 유지시키고, User- AuthorityManager 및 UserAppsManager는 다른 행동을 하지 않는다. 권한 결정자는 처음에 현재 사용자 권한과 목표 응용프로그램의 접근권한을 해결하고 비교한다. 만약 먼저 것이 나중 것보다 더 낮은 권한을 갖고 있다면, UserAuthorityManager는 시작권한이 Android IPC기법을 통해 어플리케이션 레이어(Layer)에게 거부되었다는 결과를 전파 할 것이다. 또한, 권한이 있다 하더라도 어플리케이션에 대한 상태를 권한을 따로 부여하여 각각의 실행을 제어할 수 있다.

표 1. 접근권한의 XML표

속성	내용
Name	다수 혹은 단수의 어플리케이션의 이름
Authority	접속권한을 얘기하며, 총 5단계의 접속권한을 가짐 (1단계 : 최고권한, 2단계 : 관리권한, 3단계 : 사용권한, 4단계 : 대리권한, 5단계 : 임시권한)
Category	어플리케이션이 명시된 시스템인지 검사
Authority time	권한 인증을 위한 최대 허용 시간
State	어플리케이션 설치, 제거, 실행에 대한 허가

```
<UserAuthority
name="com.android.browser, com.android.schedule"
authority="2nd level"
category="User_App"
authoritytime="12m"
state="Ex,In,Un" />
```

그림 4. 접근권한의 XML 예

2. 권한(Authority) 설정 방법

UserAuthorityManager과 UserAppsManager는 모든 설치된 응용프로그램들의 접근권한과 유저 권한 그리고 어플리케이션에 운영에 관한 관련된 정보들을 담는 Authority 저장소를 갖는다. 가볍고 효과적인 XML 직렬변환기 FastXmlSerializer(안드로이드가 제공하고 XmlPullParser 기반의 파서)를 사용하면서 우리는 Android 파일시스템의 SystemDir안에 만들어진

UserAuthorityDir안에 유저 권한 XML파일에 각각 응용프로그램의 모든 정보를 저장한다. UserAuthority로 명명된 파일이 사용자 권한을 명시하는 동안에 각 XML파일은 파일이름과 관련된 응용프로그램의 패키지 이름을 사용한다.

2.1 접근권한을 위한 속성

접근권한을 위해 우리는 다음 [표 1]과 같은 5가지의 속성을 지정하였다. 이를 사용하여 XML의 실제 예는 [그림 4]와 같다. 이 역시 사용자 인터페이스와의 상호작용을 위해 IActivityManager 인터페이스를 속성에 맞게 수정하고, ActivityManagerNative, Activity-MangerProxy, ActivityManagerService 클래스도 수정했다. 최고의 권한을 갖고 있는 사용자가 간단하게 우리의 UserAuthorityManager 클래스와 소통할 수 있고 특별한 응용프로그램의 접근 권한과 그것을 직접적으로 수정시킬 수 있게, 몇 가지 메소드들을 IActivityManager 인터페이스에 추가 및 수정하였다.

2.2 권한관리를 위한 추가적인 인터페이스

A) getUserAuthorityTime() : 앱(App) 에 대해 각 권한자가 사용할 수 있는 시간을 설정하여, 같은 앱들 혹은 다시 앱들을 호출 하는 경우 권한 인증을 추가로 하는 것을 방지한다. 또한, 기간 외 앱을 사용하였을 경우, 타 사용자의 접근을 방지하기 위해서 사용한다.

B) getAvailableApps() : 앱을 실행한 후에 지속적인 앱들의 사용이 빈번한 경우, 이러한 앱들이 모두 같은 권한자내에서 가능한지를 묻기 위한 앱들에 대한 추가적인 권한 인증을 위한 것이다.

2.3 권한관리를 위해 수정한 인터페이스

A) getUserAuthorityLevel() : 권한자의 등급을 5단계로 나눠, 로그오프 시 권한자가 사용할 수 있는 최종 앱들의 상태(State)를 설치, 제거, 실행을 위한 기본적인 권한을 제공하는 데 있다. 또한, 권한을 설정하기 위한 바탕이 되도록 수정하였다.

B) getAppState() : 앱들의 설치, 제거, 실행을 설정할 수 있는 상태를 두어, 권한자의 레벨에 따라서 앱에게 부여할 수 있는 설정을 하도록 돕도록 수정하였다.

C) UpdateAuthority() : 사용자의 등급 조절을 위한 권한자의 등급 설정 메소드를 수정하였다. 기존의 Installer와 uninstaller를 위한 클래스에 대한 인터페이스를 제공하는 대신에 UserAppsManager를 두어 사용자가 어플리케이션을 실행 및 제거하기 위해 Authority Repository에서 state값을 확인하여 이를 운용할 수 있도록 한다. 기본적은 In (Install : 설치), Un (Un-install : 제거), Ex (Execute : 실행)을 하도록 한다.

IV. 성능평가

IU_AMDroid를 개발하기 위한 환경으로 구글에서 제공하는 ADT(Andorid Developer Tools) 플러그인(Plug-In)을 탑재 한 이클립스(Eclipse) 3.5를 사용하였다. 운영체제는 리눅스 우분투 (Linux Obuntu) 10.04 버전을 기반으로 한다. 그리고 모든 테스트들은 안드로이드 에뮬레이터로 실험하였다. 또한 IDE에서 디바이스 혹은 안드로이드 에뮬레이터에 실행되는 애플리케이션을 연결하는 DDMS(Dalvik Debug Monitor Service), 안드로이드 에뮬레이터 인스턴스(Instance)혹은 스마트폰의 상태를 관리하는 ADB(Android Bridge)와 같은 디버그 툴(Debug Tool)을 사용하였다. 마지막으로 통합적인 사용자 권한 관리 모델에 대한 소스코드는 자바기반의 안드로이드 2.1업데이트 버전 1로 작성되었다. 어플리케이션 사용자 관리 모델을 적용하였을 때, 어플리케이션의 전체 동작시간을 측정하면 [그림 5]와 같다. [그림 5]에서 안드로이드 2.1의 경우에는 모든 어플리케이션의 동작시간이 약 240ms이하로 나타난다. 그리고 UAMDroid의 경우에는 전화의 경우를 제외하고는 260ms정도의 반응 시간을 보인다. 그리고 어플리케이션의 사용자 권한을 세분화하고 통합적으로 관리 가능한 I_UAMDroid는 모든 어플리케이션(전화: 250ms, 브라우저: 285ms, 음악: 284ms, 272ms)에서 반응속도에 큰 차이를 보이지 않았다. 이는 결국 보안이

강화된 상태에서도 어플리케이션의 차이가 거의 발생하지 않음을 뜻한다.

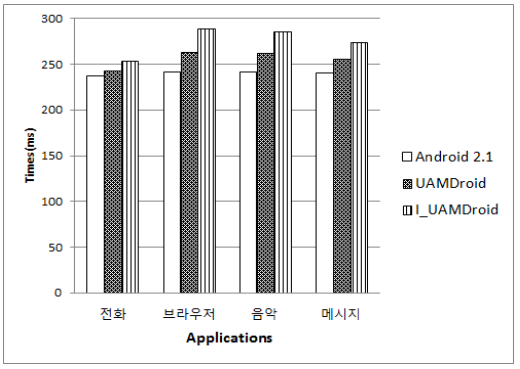

그림 5. 어플리케이션 동작 시간비교

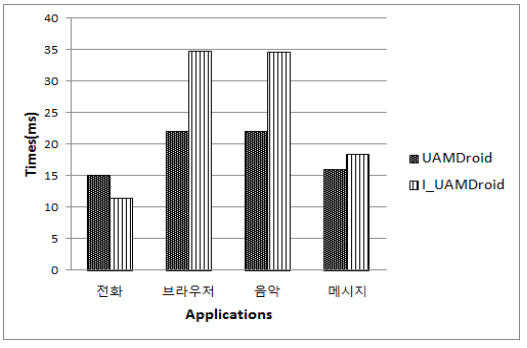

그림 6. 권한관리 확인 동작 시간

[그림 6]은 권한관리를 위해 각 어플리케이션이 권한 여부를 확인하는 시간을 나타낸다. 기존의 연구인 UAMDroid와 비교하였을 때, 기본적으로 권한을 모두 적용받을 수 있는 전화와 메시지와 같은 어플리케이션의 권한 여부 확인 시간은 전화의 경우 더 짧은 11ms의 속도를 보이고, 메시지와 같은 경우에는 단지 3ms의 속도의 차이만 보인다. 반면에 브라우저와 음악실행과 같은 개인적인 접근권한이 강화된 어플리케이션의 경우에는 약 45%이상의 확인 시간이 더 드는 것을 볼 수 있다. 즉, 개인화된 어플리케이션의 경우에는 권한의 등급, 시간, 상태와 같은 추가적인 확인 절차를 갖기 때문이다. 하지만, 그 차이도 20ms이내의 속도이기 때문에 보안을 강화한 I_UAMDroid의 장점이 높다고 할 수 있다.

V. 결 론

안드로이드 플랫폼에서 개개인의 정보를 보호하기 위한 문제가 대두되고 있다. 현재의 안드로이드 플랫폼은 단일 사용자 위주의 보호만을 지원하기 때문에 다중 사용자를 위한 보안 방법은 지원되지 않는다. 즉, 권한을 가진 사용자가 타 사용자에 의해 중요한 어플리케이션 및 개인 정보의 접근을 막기 위한 관리하기 위한 관리 프로그램이 필요하다. 따라서 본 논문은 안드로이드 플랫폼 기반의 디바이스를 위한 통합된 사용자 인증 매니저 모델을 제안하였다. 이전의 방법에 비해 사용자 권한에 따른 어플리케이션의 실행, 설치, 삭제에 대해서 세분화하고 다중의 사용자와 하나의 디바이스를 공유해 사용할 수 있는 인증 매니저 모델을 제시하였다. 또한, 이전의 연구와 비교하여 보안적인 요소는 강화되었지만, 속도에서 아주 적은 차이를 보임으로서 앞으로 안드로이드 기반 디바이스의 보안을 위한 방법을 제시하였다.

참 고 문 헌

[1] 이고은, 이종우, “스마트폰 상에서의 웹 응용프로그램 개발 환경 비교”, 한국콘텐츠학회논문지, 제10권, 제12호, pp.155-163, 2010.

[2] http://developer.android.com/guide/basics/what-is-android.html

[3] http://www.openhandsetalliance.com/

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

[5] http://www.android.com/market.html

[6] X. Ni, Z. Yang, X. Bai, A. C. Champion, and D. Xuan, "DiffUser: Differentiated User Access Control on Smartphones," International Conference on Mobile Adhoc and Sensor Systems, pp.1012-1017, 2009.

[7] Z. M. Liu, C. S. Nam, and D. R. Shin, "UAMDroid: A User Authority Manager Model for the Android Platform," International

Conference on Advanced Communication Technology, pp.1146-1150, 2011.

[8] 정윤식, 박영웅, 조성제, “안드로이드에서 개인정보 유출을 방지하기 위한 접근제어 및 디렉토리 명 사상 기법”, 정보과학회논문지, 제39권, 제6호, pp.366-372, 2012.

[9] 여세환, 이 진, 김정선, “안드로이드 플랫폼에서 인텐트 모니터링을 통한 유해 어플리케이션 차단 방법”, 정보과학회논문지, 제19권, 제5호, 2013.

[10] M. Nauman, S. Khan, and X. Zhang, "Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constrants," ACM Symposiuim on Information, Computer and Communications Security, pp.328-332, 2010.

[11] M. Nauman and S. Khan, "Design and Implementation of a Finegrained Resource Usage Model for the Android Platform," Interational Arab Joural of Information Technology, Vol.8, No.4, pp.440-448, 2010.

[12] http://www.wireless.att.com/learn/articles-resourcesD

저 자 소 개

남 춘 성(Choon-Sung Nam) 정회원

- 2005년 2월 : 상명대학교 소프트웨어학과(이학사)
- 2007년 2월 : 숭실대학교 컴퓨터학과(공학석사)
- 2011년 8월 : 성균관대학교 전자전기컴퓨터학과(공학박사)
- 2011년 9월 ~ 현재 : 성균관대학교 박사후연구원

<관심분야> : 센서 네트워크, 미들웨어, VANET, 안드로이드 프레임워크, 방향성 안테나

장 경 수(Kyung-Soo Jang) 정회원

- 1994년 2월 : 성균관대학교 전기공학과(공학사)
- 1998년 2월 : 성균관대학교 전기공학과(공학석사)
- 2005년 8월 : 성균관대학교 전자전기컴퓨터학과(공학박사)
- 2001년 9월 ~ 현재 : 경인여자대학교 교수

<관심분야> : 통신 네트워크, 유비쿼터스 컴퓨팅, 센서 네트워크

신 동 렬(Dong-Ryeol Shin) 정회원

- 1980년 2월 : 성균관대학교 전자공학과(공학사)
- 1982년 2월 : 한국과학기술원 전기 및 전자공학과(공학석사)
- 1992년 8월 : Georgia. Tech. 전기 및 전자공학과(공학박사)
- 1994년 3월 ~ 현재 : 성균관대학교 컴퓨터공학 교수

<관심분야> : 유비쿼터스 컴퓨팅, 센서 네트워크, 무선 네트워크