Fine-Grained 자원 접근 제어를 위한 향상된 안드로이드 권한 검사기법

김기원*, 김진수**
*성균관대학교 정보통신공학부
e-mail : kkekky@skku.edu
**성균관대학교 정보통신공학부
e-mail : jinsookim@skku.edu

Enhancing Android's Permission Enforcement for Fine-Grained Resource Access Control

Ki-Won Kim*, Jin-Soo Kim**
*School of Information and Communication Engineering, Sungkyunkwan University
** School of Information and Communication Engineering, Sungkyunkwan University

요 약

기존 안드로이드 응용프로그램에 대한 자원 접근 제어는 응용프로그램을 작성한 개발자가 XML 파일에 미리 선언한 권한을 검사하는 방식으로 이루어진다. 이 권한은 인스톨되는 시점에 사용자의 동의를 얻어 해당 응용프로그램에 수여되며 응용프로그램이 특정 권한을 의도적으로 사용자로부터 은닉하는 것을 방지할 수 있다. 하지만 사용자는 해당 응용프로그램에 허용하고자 하는 일부 권한만을 수여할 선택권이 없다. 선언된 모든 권한을 허락하거나 그렇지 않으면 아예 응용프로그램을 설치할 수 없게 된다. 또한, 응용프로그램에 부여한 권한을 변경할 수는 없고 박탈하는 방법은 응용프로그램을 삭제하는 방법밖에 없었다.

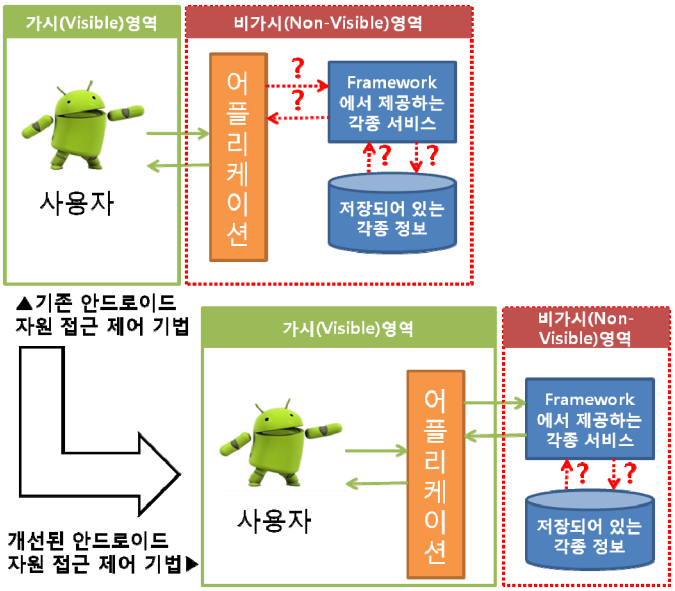
따라서 본 논문에서 제안하는 기법은 응용프로그램의 자원 접근을 사용자가 fine-grained 하게 제어할 수 있도록 하는 기법이다. 이를 위해 응용프로그램을 사용하는 시점에도 각 접근 권한을 '항상 허용', '항상 사용자가 확인' 그리고 '항상 거부'로 세부적으로 설정 및 변경이 가능하도록 하였다. 또한, MS Windows Vista 에서 적용하고 있는 '사용자계정컨트롤'(User Account Control)과 같이 응용프로그램이 요구하는 권한 중 '항상 사용자가 확인'으로 설정 된 권한은 요청 시점에 사용자에게 팝업을 띄워 접근 제어에 대한 가시성을 확보하여 사용자가 확인할 수 있도록 하였고 사용자가 이를 수락한 경우에만 해당 권한을 수여 받을 수 있도록 하였다.

1. 서론

최근 애플 앱스토어의 다운로드 횟수가 30 억 건을 돌파했고[5] MS, 안드로이드 마켓[6], SKT, 삼성 등 수만은 앱스토어가 개설되고 있고 그곳에는 수십만 건의 응용프로그램이 등록되어 있다[7]. 스마트폰의 확산과 멀티태스킹에 대한 사용자 요구의 반영은 모바일 응용프로그램의 폭발적 증가와 함께 스마트폰 사용자에 대한 개인정보 유출, 불법 과금 유발, 사기 등의 악의적 행위가 우려가 아닌 현실이 되도록 하고 있다. 하지만 개방성, 저성능, 휴대성과 같은 스마트폰 환경을 고려하여 PC 환경과는 다른 접근 방법이 필요하다[8]. 이러한 위협으로부터 스마트폰 사용자를 보호하기 위해 본 과제에서는 안드로이드 플랫폼을 대상으로 응용프로그램의 비정상적인 행동을 사전에 차단하는 향상된 응용프로그램 자원 접근 권한 제어 기법에 대해 제안하고자 한다.

기존 안드로이드 응용프로그램에 대한 자원 접근 제어는 응용프로그램을 작성한 개발자가 XML 파일에 미리 선언한 권한(SMS 에 접근하거나 GPS 에 접근하는 것과 같은 안드로이드 서비스를 요청할 권한을 의미한다)을 검사하는 방식으로 이루어진다[1]. 이 권한은 인스톨 되는 시점에 사용자의 동의를 얻어 해당 응용프로그램에 수여되며 응용프로그램이 특정 권한을 의도적으로 사용자로부터 은닉하는 것을 방지할 수 있다[3]. 하지만 스마트폰 사용자는 응용프로그램이 요청한 권한들 중에서 허용하고자 하는 일부 권한만을 수여할 선택권이 없다[2]. 선언된 모든 권한을 수여하거나 그렇지 않으면 아예 응용프로그램을 설치할 수 없게 된다[3].

본 논문에서 제시하고자 하는 기법은 사용자가 응용프로그램의 자원 접근을 fine-grained 하게 제어 할 수 있도록 하는 기법이다. 또한 응용프로그램이 악의적인 동작을 하여 피해를 입힌 후 이를 탐지하는 기법보다는 악의적인 행위를 사전에 차단하여 피해를 예방할 수 있도록 하는 기법이다. 그리고 기존의 권한검사 기법을 개선하여 사용하기 때문에 기존 응용프로그램에 대해 backward-compatibility 를 보장한다.



(그림 1)자원 접근에 대한 사용자의 가시영역의 변화

사용자는 응용프로그램이 요청하는 자원 접근 권한 중 개별 권한에 대해 '항상 허용', '항상 사용자가 확인', '항상 차단'중 한가지의 값으로 설정할 수 있다. 또한, '항상 사용자가 확인'으로 설정된 권한의 경우, MS 윈도우의 방화벽 차단 알림 기능 또는 '사용자계정컨트롤' 설정 시 알림 팝업 메시지[4]처럼 대화창을 띄워서 권한의 수락 및 거부를 결정할 수 있는 기반 환경을 제공한다. 이를 통해서 사용자는 응용프로그램이 언제, 어느 자원에 접근하려 하는지 확인 할 수 있고 사용 의도와 정황에 근거하여 응용프로그램의 자원 접근을 제어할 수 있게 된다.

현재 안드로이드의 권한관리 기법에서 응용프로그램에 수여한 권한을 빼앗는 방법은 응용프로그램을 삭제하는 방법밖에 없었다. 본 논문에서 제안하고자 하는 기법은 응용프로그램을 설치한 이후에도 이러한 권한 설정을 사용자가 변경할 수 있는 기법이다.

2. 관련연구

안드로이드를 포함한 스마트폰 응용프로그램의 보안관 관련된 선행연구는 다음과 간다.

우선 응용프로그램의 동작을 바탕으로 비정상적인 행위를 탐지하는 알고리즘이 있다[8][9][10]. 이 기법에서는 기계학습(machine-learning) 알고리즘을 활용하여 사용자의 사용 패턴을 분석하고 그것을 바탕으로 악성코드가 보이는 비정상적인 행위를 찾아내서 경고 알림을 보낸다[8]. 아직 시그니처가 등록되지 않은 유사한 동작행태를 갖는 변종 악성코드도 탐지할 수 있다는 장점이 있다[11]. 하지만 아직까지는 많은 악성코드가 출현하지 않아 동작 패턴 정보가 적다는 것과 사용자의 비 일상적인 사용 패턴에 대한 오탐지율이 높다는 해결 과제가 남아있다[12]. 또한 모니터링 및 시그니처 생성과 저장으로 인한 실행 시간 증가 및 배터리 소모를 초래한다. 따라서 본 논문을 통해 제안하려는 기법은 가능한 실행 중 간섭을 줄여 스마트폰 사용자의 사용성을 저하시키지 않도록 하였다.

다음으로는 시그니처 기반의 접근 방식을 사용하고 있는 Anti-virus 소프트웨어가 있다[11][14][15]. 이 기법에서는 검사 대상 응용프로그램으로부터 유도된 유일한 시그니처를 기존의 악성코드 데이터베이스에 있는 시그니처와 비교한다. 하지만 이는 자주 데이터베이스를 업데이트해야 하므로 무선 트래픽을 증가시키고 시기 적절한 때에 업데이트를 받지 못한다면 스마트폰의 개방성과 휴대성으로 인해 악성 코드의 급격한 확산에 적절하게 대응하지 못할 수 있다[16]. 또한 자원 제한 적인 모바일 환경에서는 효율적이지 않다[11]. 이와 유사하게 단말들이 수행기능(예, SMS, Bluetooth)과 자원 사용 정보(예, CPU, 메모리 사용량)를 중앙의 서버로 전송해서 통계기법을 활용하여 악성코드를 탐지하는 네트워크 기반의 악성코드 탐지 시스템도 있다[17][18].

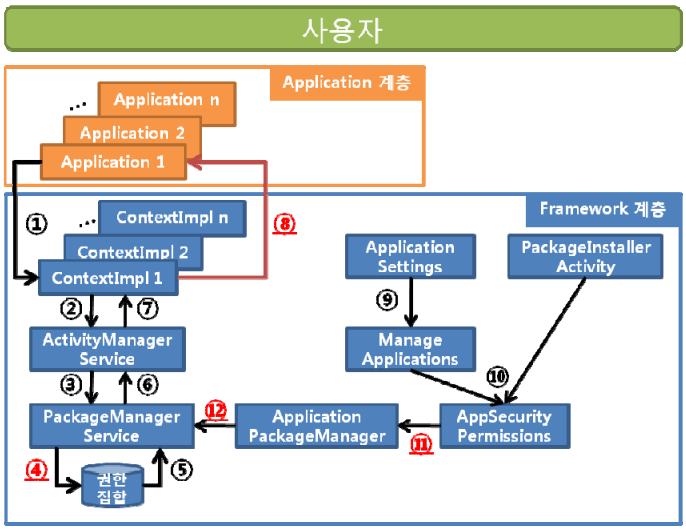
마지막으로 미리 정의한 정책을 바탕으로 개별 자원 접근 권한을 제어하는 기법이 있다[16][19][20]. 하지만 [16]의 경우 인스톨 시점에만 적용되어 실행 중 접근 권한의 제어를 하지 못한다. [19][20]의 경우에도 사용자가 해당 권한이 악의적인 목적으로 쓰이지는 않는지 사용 의도와 정황에 근거해 확인할 수 있는 방법은 제공하지 못한다. 그리고 느슨한 정책은 악의적인 행동을 탐지 못할 가능성이 있으며, 엄격한 정책의 경우 다양한 사용 패턴에 따라 오탐지가 발생할 가능성도 높다. 따라서 본 연구는 이를 보완해 사용자에게 응용프로그램의 자원 접근에 대한 가시성을 보장함을 지향한다.

3. 설계 및 구현

설계를 위한 시나리오는 아래와 같다.

<표 1> 권한관리기능의 설계를 위한 시나리오

B 씨는 외근이 잦은 영업사원이다. 따라서 필요한 은행업무를 이동 중에 스마트폰을 이용하여 처리하고 있다. 따라서 스마트폰 내에 모바일 뱅킹 관련 응용프로그램인 'BankXXX'을 설치하였는데 이 응용프로그램은 'SMS 보내기'와 관련된 권한을 가지고 있었다. B 씨는 혹시 내 결제 정보 혹은 은행의 계좌 정보가 SMS 를 통해 유출되는 것은 아닌지 걱정이 되었다. 하지만 은행에 갈 시간이 없는 B 씨는 다른 대안이 없었다. 그래서 응용프로그램 권한관리 화면을 이용하여 'BankXXX'의 SMS 발송 권한(SEND_SMS)[1]을 "항상 허용"에서 '항상 사용자가 확인'으로 변경하였다. 그리고 자신의 가계부 응용프로그램을 위해 SMS 를 발송할 때만 팝업 메시지를 통해서 수락 요청하는 것을 보고는 악성코드가 아님을 알고 안심이 되었다.



(그림 2) 안드로이드의 권한 검사 기법

위의 시나리오 1 과 그림 1 을 토대로, 제안하려는 기법은 크게 1) 응용프로그램의 실행 중 자원 접근을 제한하기 위해 권한을 검사하는 과정과 2) '항상 사용자가 확인'으로 설정된 자원 접근 권한을 사용자가 확인 후 수락 또는 거부하기 위한 팝업 다이얼로그의 구현부분, 그리고 3) 응용프로그램의 자원 접근 권한을 인스톨 이후 시점에도 변경이 가능하도록 하는 기능으로 나누어 질 수 있었다. 구현은 안드로이드 2.2 Froyo 를 대상으로 하였다.

1) 권한 검사 과정

각 응용프로그램은 안드로이드 프레임워크 계층에서 제공하는 서비스를 이용할 수 있도록 Context 객체를 하나씩 갖게 된다. 이를 이용해 특정 서비스를 요청하면(①) Context 객체는 ActivityManagerService 객체의 권한 검사 메소드 checkComponentPermission()을 호출하게 되고(②) 다시 PackageManagerService 객체의 checkUidPermission()을 호출한다(③). 이 메소드는 응용프로그램이 수여 받은 권한이 모여있는 Granted Permissions 객체 내의 grantedPermissions 해시집합에서 요청한 자원을 사용할 권한이 있는지 검사한다(④). 제안하는 기법은 권한을 수여하는 것뿐만 아니라 '항상 허용', '항상 차단', '항상 사용자가 확인'으로 각 권한을 다르게 설정할 수 있다. 따라서 granted Permission 해시집합 외에도 alwaysGrantedPermission, userDecidePermission, always DeniedPermission 해시 집합을 각각 만들어 설정 값에 따라 다른 해시집합에 저장되도록 변경하였다. 그리고 전원이 꺼지거나 설정 값이 변경되는 경우에 비 휘발성 메모리에 기록하고 꺼내기 위해 기존 writePackage()와 readGrantedPermissionsLP()에 앞서 언급한 해시 집합도 추가하였다.

권한이 모여있는 해시집합을 검사하여 존재하는 경우 PERMISSION_GRANTED 라는 상수를 반환하고 그렇지 않으면 PERMISSION_DENIED 라는 상수를 반환한다(⑤, ⑥, ⑦).

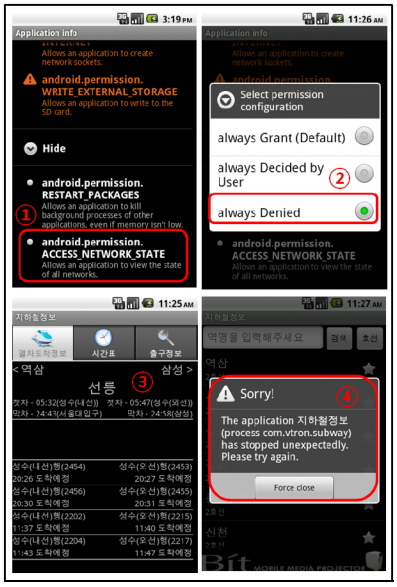
2) 사용자의 권한 확인을 위한 대화 창 구현

만약 위에서 반환된 값이 PERMISSION_DENIED 라면 SecurityException 을 발생시킨다(⑧). 하지만 userDecidePermission 해시 집합에 있는 권한의 경우 사용자로부터 수락 혹은 거부가 되는 시점까지 결정이 미뤄지게 된다. 기본적으로 안드로이드의 AlertDialog 는 Modal window[21]를 지원하지 않는다. 따라서 System window 를 사용하여 최상위 윈도우에 다이얼로그를 위치시켰고, Handler 를 이용하여 서로 다른 스레드간 통신을 통해서 Modal 을 구현하였다.

3) 인스톨 이후 개별 권한의 변경 기능

사용자가 장치의 여러 설정을 변경하기 위해 ApplicationSettings 액티비티를 실행시킨다(⑨). 그 중 설치된 응용프로그램의 설정을 변경하기 위해 ManageApplication 액티비티를 실행시킨다. 이 액티비티는 AppSecurityPermission 이라는 리스트뷰를 통해서 리스트 형태로 해당 응용프로그램이 가진 권한을 사용자에게 보여준다(⑩). 하지만 이 리스트는 권한을 그룹화해서 보여주기 때문에 개별 권한을 보여주도록 수정해야 했다. 그리고 이벤트 리스너를 추가해서 각 권한을 선택하면 '항상 허용', '항상 사용자가 확인', '항상 거부'를 선택할 수 있도록 하였다. 그리고 그 설정 값을 PackageManagerService 를 통해서 Granted Permission 객체내의 해시집합에 저장하도록 구현하였다(⑪, ⑫).

4) 권한 검사 기법의 시험

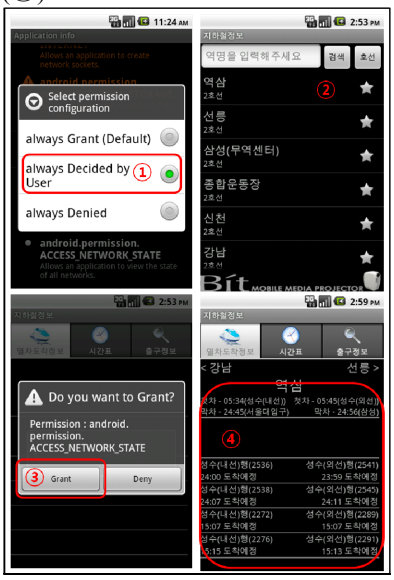


(그림 3) 권한 설정을 변경하는 기능 검증

안드로이드 SDK 에서 제공하는 에뮬레이터를 이용하여 실제 응용프로그램을 통해 제안된 기법을 실험해 보았다.

그림 3 은 설치된 응용프로그램의 권한 중

ACCESS_NETWORK_STATE(①)를 '항상 거부'로 변경 후(②) 응용프로그램을 실행한 화면이다(③). 수여 받지 않은 권한에 해당하는 자원 접근을 시도했을 때처럼 SecurityException 이 발생하여 응용프로그램이 강제 종료되었다(④).



(그림 4) 사용자가 권한 확인 후 수락하는 기능 검증

그림 4 는 설치된 응용프로그램의 권한 중 ACCESS_NETWORK_STATE 를 '항상 사용자가 확인'으로 변경 후(①) 응용프로그램을 실행한 화면이다(②). 해당 권한에 해당하는 자원 접근을 시도했을 때 사용자에게 대화 창을 띄우고 수락했을 때에만(③) 권한을 수락하게 된다(④).

4. 결론

지금까지 향상된 안드로이드의 권한검사 기법을 살펴 보았다. 그림 3 에서 볼 수 있듯이 이 기법은 응용프로그램 개발자가 SecurityException 에 대한 예외처리를 하지 않으면 예외로 인한 강제 종료 현상이 발생한다는 단점을 가지고 있다.

하지만 이 기법을 통해서 사용자는 사용하는 안드로이드 응용프로그램이 언제, 어떤 안드로이드 어플리케이션 프레임워크 계층의 자원 및 서비스를 사용하는지 직접 확인할 수 있게 되었으며 여기서 그치지 않고 개별 권한의 수여와 거부를 직접 제어할 수 있게 되었다.

참고문헌

[1] Android Developers, "Android Developers: References", http://developer.android.com/reference/android/Manifest.permission.html, 2010

[2] Wook Shin, Shinsaku Kiyomoto, Kazuhide Fukushima, Toshiaki Tanaka, "Towards Formal Analysis of the Permission-Based Security Model for Android," icwmc, pp.87-92, 2009 Fifth International Conference on Wireless and Mobile Communications, 2009

[3] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev, Chanan Glezer, "Google Android: A Comprehensive Security Assessment," IEEE Security and Privacy, vol. 8, no. 2, pp. 35-44, Mar./Apr. 2010

[4] Microsoft, "User Account Control ", http://technet.microsoft.com/en-us/library/cc731416(WS.10).aspx, May 11 2010

[5] Apple Application Store, http://www.apple.com/iphone/apps-for-iphone/, 2010

[6] Android Market, http://www.android.com/market, 2010

[7] 강동호 외, "스마트폰 보안 위협 및 대응 기술", ETRI 전자통신동향분석 제 25 권 제 3 호, 2010 년 6 월

[8] Liang Xie 외, "pBMDS: A Behavior-based Malware Detection System for Cellphone Devices", ACM WiSec10' pp. 37-48, March 2010

[9] A. Shabtai, U. Kanonov, and Y. Elovici, "Detection, Alert and Response to Malicious Behavior in Mobile Devices: Knowledge-Based Approach", Springer Berlin / Heidelberg RAID Lecture Notes in Computer Science pp.357-358, 2009.

[10] A. Shabtai, Y. Fledel, Y. Elovici, and Y. Shahar, "Using the KBTA Method for Inferring Computer and Network Security Alerts from Timestamped, Raw System Metrics" Journal in Computer Virology, 8(3): 267-298, 2009.

[11] Abhijit Bose 외, "Behavioral Detection of Malware on Mobile Handsets", ACM MobiSys pp. 225-238, June 2008

[12] Tansu Alpcan, Christian Bauckhage, Aubrey-Derrick Schmidt, "A Probabilistic Diffusion Scheme for Anomaly Detection on Smartphones", Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices Lecture Notes in Computer Science, 2010, Volume 6033/2010, 31-46, 2010

[13] Anlab, " 안철수 연구소 모바일 보안 제품: V3 Mobile", http://www.ahnlab.com/kr/site/product/productivew.do?prodSeq=19, 2010

[14] Hauri, "하우리 모바일 보안 제품: ViRobot Mobile for Android", http://www.hauri.co.kr/customer/support/hauri_notice_view.html?intSeq=203&page=1&keyfield=&key=, 2010

[15] William Enck 외, "On Lightweight Mobile Phone Application Certification", ACM CCS 09 pp. 235-245, November 9-13 2009

[16] Schmidt, Peter 외, "Monitoring Smartphones for Anomaly Detection", Mobile Networks and Applications Volume 14, Number 1, 92-106, 11 November 2008

[17] J.Cheng 외, "Smart Siren : Virus Detection and Alert for Smart phones", MobiSys 07 pp. 258-271, June 2007

[18] Machigar Ongtang, Stephen McLaughlin, William Enck, Patrick McDaniel, "Semantically Rich Application-Centric Security in Android," acsac, pp.340-349, 2009 Annual Computer Security Applications Conference, 2009

[19] Mohammad Nauman 외, "Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints", ASIACCS'10 pp. 328-332, April 13–16, 2010,

[20] Wikipedia, "Modal Window", http://en.wikipedia.org/wiki/Modal_window", Wikipedia, 30 August 2010