ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

모바일 앱 최소권한 사전검증에 관한 연구 - 금융, 안드로이드 운영체제 중심으로 -

A Study of Security Checks for Android Least Privilege - focusing on mobile financial services -

조 병 철[1] 최 진 영[1*]
Byung-chul Cho Jin-young Choi

요 약

안드로이드 운영체제의 보안체계는 샌드박스와 권한모델을 적용하고 있다. 특히 권한모델은 설치시점 확인과 all-or-nothing 정책을 운영하고 있기 때문에 안드로이드는 앱을 설치할 때 필요한 권한에 대해 사용자 동의를 요구하고 있다. 하지만 안드로이드 권한에 대한 사용자의 인식은 부족한 상황이다. 따라서 본 논문에서는 실제 모바일 앱을 대상으로 권한요구 실태를 조사하고 금융회사를 중심으로 모바일 서비스 제공자가 모바일 앱의 최소권한 정책에 위배되는 사항을 자체점검하고자 할 때 활용가능한 중점 점검항목과 방법을 제시하고 그 유용성에 대해 알아보고자 한다.

☞ 주제어 : 모바일 보안, 안드로이드 보안, 퍼미션, 최소권한

ABSTRACT

A security system in Android OS adopts sandbox and an permission model. In particular, the permission model operates the confirmation of installation time and all-or-nothing policy. Accordingly, the Android OS requires a user agreement for permission when installing an application, however there is very low level of user awareness for the permission. In this paper, the current status of permission requirement within mobile apps will be discovered, and the key inspection list with an appropriate method, when a mobile service provider autonomously inspects the violation of least privilege around financial companies, and its usefulness will be explored.

☞ keyword : mobile security, android security, permissions, least privilege

1. 서 론

시장조사 전문기관인 IDC 발표 자료에 따르면 2015년 2분기에 안드로이드 OS는 세계 스마트폰 OS 시장에서 82.8%의 점유율을 차지하였다[1]. 또한 2014년 1분기에는 새로운 보안위협(277개) 가운데 99% 이상이 안드로이드 플랫폼에서 발생했을 정도로 안드로이드에 대한 보안위협은 지속되고 있는 상황이다[2].

한편 안드로이드는 다양한 보안위협에 대응하기 위해 샌드박스, 권한모델 등의 보안정책을 운영하고 있다[3]. 특히 권한모델은 설치시점 확인과 all-or-nothing 정책을 적용하고 있기 때문에 자원접근에 필요한 권한에 대해 사용자 동의를 요구하고 있다. 하지만 Felt 등[4]은 온라인(308명)과 오프라인(25명) 실험을 통해서 안드로이드 권한(permission)에 대한 사용자들의 인식이 부족하다는 사실을 보여주었다.

방송통신위원회도 이와 같은 문제점을 인지하고 스마트폰 앱 개인정보보호 가이드라인을 통해 앱이 이용자의 스마트폰 정보에 접근할 수 있는 권한의 범위를 서비스에 필요한 범위 내로 최소화하도록 요구하였고[5] 금융감독원은 금융회사에게 개발단계부터 스마트폰 앱에 접근권한이 불필요한 항목이 포함되지 않도록 개발하여 배포하고 운영중인 앱에 대해서도 일제히 점검하여 불필요한 접근권한을 즉시 삭제하도록 요구하였다[6].

그러나 감독당국의 요구대로 모바일 서비스 제공자가 최소권한(least privilege) 원칙을 준수하기 위한 현실적인 대안은 마땅하지 않은 것이 현실이다. 이에 따라 본 연구에서는 안드로이드 권한시스템에 관한 몇 가지 의문점을 가지고 실태를 조사하였다.

[1] CIST(Center for Information Security Technologies), Korea University, Seoul, 136-713, Korea
[*] Corresponding author (choi@formal.korea.ac.kr)
[Received 7 November 2015, Reviewed 16 Novemver 2015, Accepted 3 December 2015]

- 사용자에게 동의를 요구하는 권한은 주로 어떠한 것들인가?
- 금융회사 업종에 따라 요구하는 권한별 특징이 분류되는가?
- 금융회사 등 모바일 서비스 제공자가 최소권한 원칙 준수여부를 자체점검하고자 할 때 활용할 수 있는 중점점검 권한 도출이 가능한가?
- 자체점검 시 활용 가능한 도구는 유용한가?

이를 위해 3장에서는 리버싱 기법을 통한 권한요구 현황 분석방법을 소개하고 4장에서는 실제 구글 플레이에 등록된 금융분야 모바일 앱을 대상으로 분석을 실시하고자 한다. 이를 통해 향후 모바일 서비스 제공자가 자율점검 수행 시 활용가능한 중점 점검항목을 도출하여 도구를 통한 점검방법의 효과성에 대해 알아보고 5~6장에서는 최소권한 원칙 준수를 위한 개선방안과 관련연구 및 향후계획에 대해 서술하고자 한다.

2. 연구 배경

2.1 안드로이드 보안 체계

안드로이드는 대표적으로 다음과 같은 보안기능을 제공한다[3].

1. 샌드박스* : 안드로이드에 설치된 응용프로그램은 기본적으로 일반 사용자 권한으로 실행된다. 각 앱은 고유의 사용자 아이디와 그룹 아이디를 부여받으며 앱간의 접근은 기본적으로 불가하다. 만약 다른 앱과 데이터 공유가 필요하다면 명시적인 정의가 필요하다.
2. 권한모델 : 안드로이드는 권한에 따라 단말기 자원에 대한 접근을 제한하도록 하고 있으며 이러한 제약은 사용자나 시스템에 대해 위험도가 높은 것으로 한정된다. 앱의 권한들은 안드로이드에 이미 내장되어 있으나 개발자가 새로운 권한을 설정할 수도 있다. 개발자가 사용할 수 있는 권한들은 안드로이드 공식 개발자 사이트에서 확인할 수 있다[7].
3. 서명 : 서명되지 않은 앱은 안드로이드 기기에 설치되지 못하도록 통제하고 안드로이드 마켓에 업로드할 때 반드시 개발자 개인키(private key)로 서명하도록 강제한다.

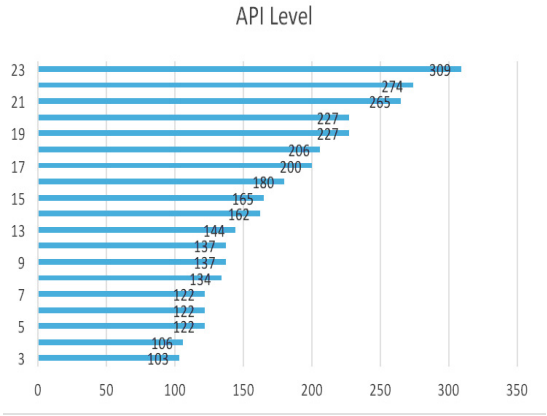
* 샌드박스 : 외부로부터 들어온 프로그램이 보호된 영역에서 동작해 시스템이 부정하게 조작되는 것을 막는 보안 형태이다.

2.2 안드로이드 권한

안드로이드는 단말의 자원에 접근이 필요한 모든 권한을 AndroidManifest.xml 파일에 정의해야 한다. 사용자는 앱을 설치할 때 요청된 권한목록을 확인하고 동의하는 절차를 거치게 되며 앱이 삭제되기 전까지 요청된 모든 권한은 유지된다. 다만, 의도한 기능이 필요하지 않을 때에도 자원에 접근이 허용된다는 단점을 가진다[3].

안드로이드는 2009년 API레벨1의 플랫폼 버전 1.0을 시작으로 현재 API레벨23의 6.0까지 출시되었으며 그림 1에서 보는 바와 같이 API레벨3의 권한은 103개였던 것이 가장 최근 버전인 API레벨23에서는 309개까지 증가하였다[8].

안드로이드 권한은 Protection Level과 Permission Group으로 구성된다.



(그림 1) API레벨의 권한 수
(Figure 1) Number of permissions for API level

2.1.1 Protection Level

각각의 권한들은 위험도에 따라 Normal, Dangerous, Signature, SinatureOrSystem의 4단계로 구분된다[3].

1. Normal : 위험이 낮은 권한으로 사용자의 승인없이도 자동적으로 권한이 허용되며 사용자는 허용된 권한에 대해 확인이 가능하다.
2. Dangerous : 위험이 높은 권한으로 사용자의 승인을 득한 경우에만 권한이 허용된다.
3. Signature : 가장 높은 단계의 권한으로 단말기 제조사의 인증서로 서명이 필요하며 사용자에게 별도

안내없이 설치된다.

4. SignatureOrSystem : 시스템 이미지에 있거나 시스템 이미지에 있는 동일 서명이 필요하며 사용자에게 별도 안내없이 설치된다.

2.1.2 Permission Group

안드로이드 시스템의 모든 dangerous 권한은 반드시 어느 하나의 권한 그룹과 연결되며 그룹 분류는 표1과 같다[9]. 만약 앱에서 dangerous 권한이 필요한 데 이미 동일 그룹에 속해 있는 권한에 대해 사용자 동의를 받았다면 별도 사용자 동의를 요구하지 않고 권한을 허용해준다. 만약 READ_CONTACTS 권한이 이미 허용된 상태라면 WRITE_CONTACTS 권한 요청도 같이 허용된다.

(표 1) 안드로이드 권한 그룹
(Table 1) Android Permission Group

Permission Group	Description
CALENDAR	Used for runtime permissions related to user's calendar.
CAMERA	Used for permissions that are associated with accessing camera or capturing images/video from the device.
CONTACTS	Used for runtime permissions related to user's contacts and profile.
LOCATION	Used for permissions that allow accessing the device location.
MICROPHONE	Used for permissions that are associated with accessing microphone audio from the device.
PHONE	Used for permissions that are associated telephony features.
SENSORS	Used for permissions that are associated with accessing camera or capturing images/video from the device.
SMS	Used for runtime permissions related to user's SMS messages.
STORAGE	Used for runtime permissions related to the shared external storage.

2.3 관련 연구

안드로이드 권한과 관련한 기존 연구는 다음과 같다. Felt 등은 모바일 앱을 분석하여 앱이 사용자 동의를 요구하는 권한과 실제 앱에서 사용되는 API가 요구하는 권한을 비교한 결과 940의 앱 중 1/3이상의 앱이 필요 이상의 권한을 요구한다는 사실을 확인하였고[16] 앱에서 불필요한 권한을 요구하는지를 점검하는 Stowaway 도구를 통해 안드로이드 API와 권한 간의 연관정보를 조사하였으나 더 이상 해당 도구는 현재의 앱들을 제대로 분석해내지 못한다[17].

Enck 등[18]은 앱이 설치될 때 해당 앱이 요구하는 권한과 권한규칙을 비교하여 앱의 보안을 평가하는 시스템인 Kirin을 제안하였다. 사용자가 앱을 설치할 때 간단하게 9가지 샘플규칙을 통해 악의적인 것으로 의심되는 권한정보를 제시하였으나 금융앱에서는 오탐에 해당하는 것이 많았다.

Vidas 등[19]은 안드로이드 문서를 검사하여 권한명세에 대해 조사하고 개발자가 필요한 권한과 불필요한 권한을 알 수 있도록 플러그인 형태의 개발자 지원도구를 제시하였지만 아직 개념검증용으로 실제 활용은 제한적이었다.

Wei 등[20]은 안드로이드 권한시스템의 발전과 사용에 대한 조사를 통해 보호수준이 dangerous인 앱이 증가하고 있으며 그것이 불필요한 권한을 요구하는 앱과 연관성이 있음을 확인하고 개발자에게 해당 보호수준을 사용할 때에는 더욱 주의해야 한다고 강조하였다.

앞서 살펴본 바와 같이 안드로이드 권한을 중심으로 다양한 연구는 계속되고 있으나 실제 업무에 활용가능한 안드로이드 최소권한 사전검증 방안에 대한 연구는 확인되지 않았다. 따라서 본 논문에서는 모바일 서비스 제공자가 서비스 시작 이전에 최소권한 원칙 준수여부를 자율적으로 점검하기 위한 방안에 대해 금융회사 업종에 따른 특징분류 관점에서 연구하고자 한다.

3. 권한요구 현황 분석 방법

3.1 개요

본 연구에서는 실제 모바일 앱에서 요구하고 있는 권한들에 대한 현황을 파악하기 위해 개발자가 앱에 명시적으로 정의한 권한정보와 실제 프로그램의 소스코드에서 사용되는 권한정보들을 리버싱 도구를 활용하여 분석을 수행하였다.

3.2 개발자가 정의한 권한정보 추출

앱 실행 시 요구되는 개발자 정의 권한정보를 추출하기 위해서는 AndroidManifest.xml 파일을 확인해야 한다. 먼저 APK의 압축을 해제하면 AndroidManifest.xml 파일을 추출할 수 있으나 이는 컴파일된 형태이기 때문에 다시 변환작업을 수행해야 한다.

이러한 일련의 작업은 apktool[10]을 활용하여 자동화된 방법으로 AndroidManifest.xml 파일을 추출할 수 있다. 또한 androaxml[11]도 AndroidManifest.xml 파일의 내용을 바로 화면에 출력해주는 기능을 제공하며 aapt[12]는 AndroidManifest.xml에 정의된 정보 이외에도 패키지정보, 권한정보 등을 화면상에 함께 출력해준다. 그림2는 Androaxml를 이용하여 앱에서 요구하고 있는 권한을 추출한 결과이다.

```
File  Edit  Tabs  Help
      <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE'
>
</uses-permission>

      <uses-permission android:name="android.permission.MODIFY_PHONE_STATE">
</uses-permission>

      <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE">
</uses-permission>

      <uses-permission android:name="android.permission.INSTALL_PACKAGES">
</uses-permission>

      <uses-permission android:name="android.permission.DELETE_PACKAGES">
</uses-permission>

</manifest>

jokun@jokun-VM:~/tools/apks/success$
```

(그림 2) Androaxml를 이용한 권한 조회
(Figure 2) Extracting permission from Androidmanifest using Androaxml

3.3 프로그램에서 사용되는 권한정보 추출

실제 앱에서 사용되는 권한정보를 추출하기 위해서는 소스코드의 API를 추출하고 각 권한과 API간의 연관관계를 알아야 한다.

소스코드의 API는 dex2jar[13]를 통해 APK 파일을 자바 클래스 파일로 변환하고 jad[14]를 통해 다시 자바 파일로 디컴파일하여 문자열 검색 등을 통해 확인할 수 있다. 또한 API에서 요구하는 권한은 pscout[15]에 제시된 mapping 자료를 통해 확인이 가능하다.

본 논문에서는 Androguard를 이용하여 APK 파일로부터 권한정보를 추출하였다. 그림3은 Androlyze를 이용하여 API와 권한정보를 추출한 결과이다.

```
File  Edit  Tabs  Help
1 Lcom/softsecurity/transkey/ITranskeyCommon;->GetUniqueID(Landroid/content/Cont
ext;)Ljava/lang/String; (0x16) ---> Landroid/telephony/TelephonyManager;->getSim
SerialNumber()Ljava/lang/String;
1 Lcom/softsecurity/transkey/ITranskeyCommon;->GetUniqueID(Landroid/content/Cont
ext;)Ljava/lang/String; (0x1e) ---> Landroid/telephony/TelephonyManager;->getDev
iceId()Ljava/lang/String;
1 Lkr/co/morpheus/client/library/common/b;->a(Landroid/content/Context;)V (0x31a
) ---> Landroid/telephony/TelephonyManager;->getDeviceId()Ljava/lang/String;
1 Lkr/co/morpheus/client/library/common/b;->a(Landroid/content/Context;)V (0x326
) ---> Landroid/telephony/TelephonyManager;->getDeviceSoftwareVersion()Ljava/lan
g/String;
1 Lkr/co/morpheus/client/library/common/b;->a(Landroid/content/Context;)V (0x36a
) ---> Landroid/telephony/TelephonyManager;->getLine1Number()Ljava/lang/String;
1 Lkr/co/morpheus/client/library/utils/r;->a(Landroid/content/Context;)Ljava/lan
g/String; (0x9a) ---> Landroid/telephony/TelephonyManager;->getDeviceId()Ljava/l
ang/String;
1 Lkr/co/morpheus/client/library/utils/r;->a(Landroid/content/Context;)Ljava/lan
g/String; (0x146) ---> Landroid/telephony/TelephonyManager;->getLine1Number()Lja
va/lang/String;
PERM :  ACCESS_FINE_LOCATION
1 Lbtworks/codeguard/engine/MainService$1;->A(Z Z)Ljava/lang/String; (0x70) --->
 Landroid/location/LocationManager;->getLastKnownLocation(Ljava/lang/String;)Lan
droid/location/Location;
jokun@jokun-VM:~/tools/apks/success$
```

(그림 3) Androlyze를 이용한 권한 조회
(Figure 3) Extracting permission from api using Androlyze

참고로 Androlyze는 그림4와 같이 API와 권한정보에 대한 연관관계를 별도 파일로 관리하기 때문에 오탐에 대한 수정 및 보완이 가능하다. 미탐으로 확인된 API에 대한 권한정보를 해당 파일에 등록하게 되면 추후 점검 시에는 점검결과에 포함되어 출력된다.

```
File  Edit  Tabs  Help
                ("F", "isBackoffStat", "()"),
                ("F", "isWimaxEnable", "()"),
        ],
},
"ACCESS_WIFI_STATE" : {
        "Landroid/net/sip/SipAudioCall;" : [
                ("F", "startAudio", "()"),
        ],
        "Landroid/net/wifi/IWifiManager$Stub$Proxy;" : [
                ("F", "getConfiguredNetworks", "()"),
                ("F", "getConnectionInfo", "()"),
                ("F", "getDhcpInfo", "()"),
                ("F", "getNumAllowedChannels", "()"),
                ("F", "getScanResults", "()"),
                ("F", "getValidChannelCounts", "()"),
                ("F", "getWifiApEnabledState", "()"),
                ("F", "getWifiEnabledState", "()"),
                ("F", "isMulticastEnabled", "()"),
        ],
        "Landroid/net/wifi/WifiManager;" : [
                ("F", "getConfiguredNetworks", "()"),
                ("F", "getConnectionInfo", "()"),
                ("F", "getDhcpInfo", "()"),
jokun@jokun-VM:/usr/share/androguard/androguard/core/bytecodes$
```

(그림 4) API와 권한 연결정보
(Figure 4) Mapping between api and permission

3.4 개발자 정의 권한과 실제 사용권한 비교

개발자가 정의한 권한정보와 프로그램에서 사용되는 권한정보를 비교하기 위해 그림5와 같이 점검결과 형식을 정규식을 통해 표준화하고 문자열 비교도구인 diff를 사용하여 권한을 비교하였다.

```
File  Edit  Tabs  Help
#!/bin/bash

for FILE in "$@"
do

    aapt d badging $FILE | grep "uses-perm" | grep -o "\.[^\.]*'" | sed "s/[\'\.]//g" | sort -u > $FILE.perm.tmp.1
    aapt d badging $FILE | grep "uses-implied-permission" | grep -o "\.[^'\.]*'" | sed "s/['.]//g" | sort -u > $FILE.perm.tmp.2
    grep -v -x -f $FILE.perm.tmp.2 $FILE.perm.tmp.1 > $FILE.perm.1
    rm $FILE.perm.tmp.1 $FILE.perm.tmp.2

    cat $FILE.perm | awk '{ print $1, $2 }' | sed "s/\[//g" | sed "s/'//g" | sed "s/,//g" | sed "s/ /|/g" | grep -o "\.permission\.[^\"]*" | sed "s/\.permission\.//g" | sort -u > $FILE.perm.2
    cat $FILE.method | grep PERM | awk '{ print $3 }' | sort -u > $FILE.method.1

    echo "" > $FILE.diff.1
    echo "[$FILE]" >> $FILE.diff.1
    echo "------------------------------------------------------------------------" >> $FILE.diff.1
    echo "       DEFINED PERMISSION             |          USED PERMISSION" >> $FILE.diff.1
    echo "------------------------------------------------------------------------" >> $FILE.diff.1
    diff -y -W 80 $FILE.perm.1 $FILE.method.1 >> $FILE.diff.1
    echo "------------------------------------------------------------------------" >> $FILE.diff.1
    cat $FILE.diff.1

done
~
                                                                    1,1           모두
```

(그림 5) 권한비교 프로토타입
(Figure 5) Prototype about comparing permissions

그림6은 개발자 정의 권한과 프로그램 사용 권한에 대한 비교 결과이다. 먼저 좌우측에서 공통적으로 확인된 권한으로 ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE , INTERNET, READ_CONTACTS, READ_PHONE_STATE, RESTART_PACKAGES는 개발자가 정의한 권한이 실제 프로그램에서 사용되는 것으로 올바른 권한 정의에 해당한다. 즉 사용자 동의를 요청한 모든 권한이 실제 프로그램 사용 시 필요한 권한임을 나타낸다.

한편 CALL_PHONE, CHANGE_NETWORK_STATE, MODIFY_PHONE_STATE, READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE 등과 같이 좌측에서만 보이는 권한들은 프로그램에서 사용되지 않으나 사용자 동의를 요구하는 것으로 불필요 권한일 가능성이 존재하는 것들이다.

반대로 ACCESS_FINE_LOCATION, CAMERA, FACTORY_TEST, NFC, READ_LOGS, RECORD_AUDIO, VIBRATE, WAKE_LOCK와 같이 우측에서만 보이는 권한들은 프로그램 소스에는 존재하지만 실제 프로그램 동작시에는 사용되지 않는 권한들로 보안적인 이슈를 떠나 향후에도 계속 사용되지 않는 코드에 해당하고 서드파티 라이브러리가 아닌 자체적으로 개발한 코드라면 해당코드에 대한 제거를 검토해 볼 수 있다.

```
File  Edit  Tabs  Help
------------------------------------------------------------------------
       DEFINED PERMISSION             |          USED PERMISSION
------------------------------------------------------------------------
                                      > ACCESS_FINE_LOCATION
ACCESS_NETWORK_STATE                    ACCESS_NETWORK_STATE
ACCESS_WIFI_STATE                       ACCESS_WIFI_STATE
C2D_MESSAGE                           | CAMERA
CALL_PHONE                            | FACTORY_TEST
CHANGE_NETWORK_STATE                  <
DELETE_PACKAGES                       <
GET_TASKS                             <
INSTALL_PACKAGES                      <
INTERNET                                INTERNET
KILL_BACKGROUND_PROCESSES             | NFC
MODIFY_PHONE_STATE                    <
READ_CONTACTS                           READ_CONTACTS
READ_EXTERNAL_STORAGE                 | READ_LOGS
READ_PHONE_STATE                        READ_PHONE_STATE
RECEIVE                               | RECORD_AUDIO
RESTART_PACKAGES                        RESTART_PACKAGES
USE_CREDENTIALS                       | VIBRATE
WRITE_EXTERNAL_STORAGE                | WAKE_LOCK
------------------------------------------------------------------------
jokun@jokun-VM:~/tools/apks/success$
```

(그림 6) 정의된 권한 vs 사용 권한
(Figure 6) Comparing permission between predefined and program required

4. 실험 및 평가

4.1 실험 데이터

실험은 구글 플레이의 금융 범주에 등록된 인기앱 가운데 448개를 대상으로 선정하여 분석을 진행하였다. 실험대상에는 스마트폰 뱅킹 등 전자금융거래를 위한 앱부터 가계부와 같은 일반 앱까지 다양하게 포함되어 있다.

4.2 실험 평가

4.2.1. RQ1 : 사용자 동의를 요구하는 권한

안드로이드는 보호수준이 dangerous에 해당하는 경우 사용자로부터 해당 권한에 대한 승인을 요구한다. 표2는 사용자 동의를 요구하는 권한 중 상위 20개를 나열한 것으로 70% 이상의 앱이 INTERNET(인터넷 등 네트워크 접속 허용), WRITE_EXTERNAL_STORAGE(외부 저장장치 쓰기 허용), READ_PHONE_STATE(폰 상태 읽기 허용, IMEI, IMSI, 폰번호, 통화중상태 등) 권한을 요구하고 있었다. 이외에도 전화걸기, 위치정보 접근, SMS 읽기/보내기, 카메라, 연락처 읽기 등 다양한 개인정보 관련 권한들이 10% 이상의 앱에서 요구되는 것으로 나타났다.

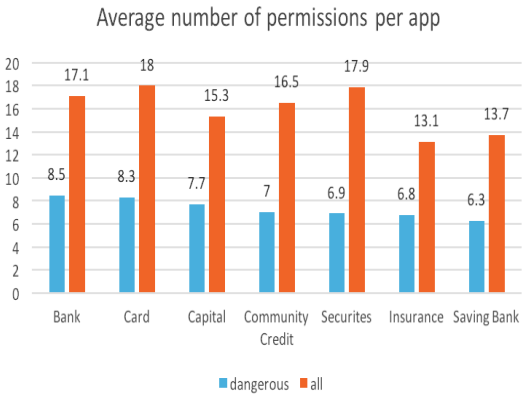
(표 2) 사용자 동의 요구권한 상위 20개
(Table 2) The top 20 most frequent dangerous permissions

Permissions	Usage(%)
INTERNET	98.4
WRITE_EXTERNAL_STORAGE	82.4
READ_PHONE_STATE	72.5
GET_TASKS	50.4
CALL_PHONE	37.5
CHANGE_WIFI_STATE	34.2
ACCESS_FINE_LOCATION	29.0
RECEIVE_SMS	26.6
CAMERA	24.6
ACCESS_COARSE_LOCATION	23.4
READ_CONTACTS	13.4
DISABLE_KEYGUARD	11.6
READ_SMS	10.9
SYSTEM_ALERT_WINDOW	9.8
SEND_SMS	9.4
USE_CREDENTIALS	8.9
NFC	7.8
ACCESS_MOCK_LOCATION	7.4
READ_CALL_LOG	6.3
RECORD_AUDIO	5.4

4.2.2. RQ2 : 금융회사 업종별 요구권한 특징

사용자 동의를 요구하는 권한을 대상으로 금융회사 업종별에 따라 분류해 본 결과 그림7과 같이 보호수준이 dangerous인 권한에 대해 앱별 평균 개수는 은행이 8.5개로 가장 많았고 카드, 캐피탈, 증권, 새마을금고, 증권, 보험 순이었다. 이에 반해 보호수준 전체를 대상으로 할 경우에는 카드가 18개로 가장 많았고 증권, 은행, 새마을금고, 캐피탈, 저축은행, 보험 순으로 확인되었다.

표3은 사용자 동의를 요구하는 권한의 세부 항목 전체를 대상으로 금융회사 주요 업종 중심으로 살펴보았다. 은행이 31개로 가장 많은 유형의 권한을 요구하였고 그 다음으로는 카드, 보험, 증권 순이었다. 특히 증권은 READ_SMS, READ_CONTACTS, WRITE_CONTACTS 등을 요구하지 않고 타업종에 비해 요구하는 권한 유형이 상대적으로 적었으며 카드는 연락처 쓰기(WRITE_CONTACTS)를 허용하지 않았다. 보험은 일정 읽기/쓰기(READ_CALENDAR / WRITE_CALENDAR)를 허용하지 않는 것으로 확인되었다.



(그림 7) 금융업종별 요구권한 앱별 평균 개수
(Figure 7) The average number of permissions by financial sector

4.2.3. RQ3 : 자체점검 시 중점점검 권한 목록

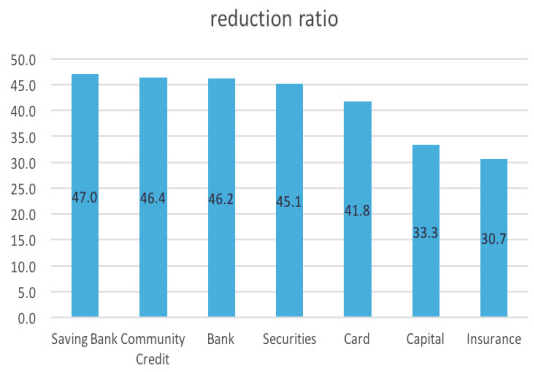
표3을 통해 금융회사 업종별로 차이가 있음을 확인하였고 이에 따라 은행과 카드는 31개, 보험은 28개, 증권은 19개의 요구권한 목록을 도출할 수 있었다.

금융회사 등 모바일 서비스 제공자는 해당 업종별 요구권한 목록을 기준으로 중점점검 권한 체크리스트를 작성하고 이를 토대로 향후 자체점검을 수행하고자 할 때 최소권한의 여부에 대해 화이트리스트 방식으로 효율적인 점검을 수행할 수 있다.

4.2.4. RQ4 : 자체점검 도구 및 방법

3장에서 소개된 방법에 따라 도구를 활용한 자체점검을 수행하여 개발자가 정의한 권한정보와 프로그램에서 요구하는 권한정보를 비교한 결과 그림8과 같이 저축은행이 47%로 가장 많은 감소효과가 있었고 새마을금고, 은행, 증권, 카드, 캐피탈, 보험 순으로 효과가 확인되었다. 업종에 상관없이 448개 전체 앱을 대상으로 평균적으로는 43.1%의 점검대상 권한목록을 줄이는 효과가 있었다. 100% 감소효과를 나타난 앱은 14개였고 50%이상은 195개였다.

점검대상 권한의 감소효과와 불필요 권한 검출에 대한 점검도구의 효과성을 확인하기 위해 사례연구로 생명보험사인 A사를 대상으로 추가적인 검토를 진행하였다. 앞서 살펴본 바와 같이 그림6은 A사의 모바일 앱을 대상으로 불필요 권한에 대한 점검을 수행한 결과이다.



(그림 8) 금융업종별 점검대상 감소율
(Figure 8) Reduction ratio of permissions for audit by financial sector

점검결과에서 좌우측에서 공통적으로 확인된 INTERNET 등 6개 권한은 실제 앱에서 사용되는 필요 권한으로 AndroidManifest.xml에 정의된 18개 권한 중 해당 권한에 대해서는 점검대상 권한목록에서 제외처리하였다. 이를 통해 A사의 경우 점검대상이 18개에서 12개로 33% 감소 효과가 있었다.

또한 나머지 12개 권한 중 점검대상 권한을 보호수준이 dangerous 인 것으로 한정하여 다시 점검대상의 범위가 4개로 축소되었다.

최종적으로 불필요 권한으로 의심되는 CALL_PHONE 등 4개의 권한에 대해서 개발자와 인터뷰를 진행한 결과 CALL_PHONE의 경우 앱에서 실제 고객센터 안내번호를 클릭하면 사용자가 전화번호를 별도로 누르지 않고 바로 전화를 연결하기 위해 필요하였고 WRITE_EXTERNAL_STORAGE의 경우 상품안내장 PDF 파일을 내려받는 기능에서 사용하기 위해 필요하였다. GET_TASKS와 USE_CREDENTIALS은 서드파티에서 사용하는 기능으로 확인되었다. A사의 경우는 4개가 모두 필요한 권한으로 확인되었고 도구를 통한 점검결과는 오탐으로 확인되었다.

(표 3) 금융앱에서 요구되는 권한
(Table 3) Permissions requested by financial sector

Permissions	Bank	Card	Insurance	Securities
INTERNET	57	34	36	45
WRITE_EXTERNAL_STORAGE	57	32	27	42
READ_PHONE_STATE	55	34	26	42
GET_TASKS	48	21	20	39
CHANGE_WIFI_STATE	30	15	17	28
ACCESS_FINE_LOCATION	25	12	19	19
CALL_PHONE	35	23	22	15
ACCESS_COARSE_LOCATION	16	11	17	15
DISABLE_KEYGUARD	5	9	1	12
SYSTEM_ALERT_WINDOW	7	5	2	11
ACCESS_MOCK_LOCATION	5	1	7	10
SEND_SMS	7	3	3	9
RECEIVE_SMS	21	17	11	7
CAMERA	33	13	11	4
READ_CALENDAR	2	1		3
WRITE_CALENDAR	2	1		3
USE_CREDENTIALS	3	8	4	2
BLUETOOTH	4	6	2	2
RECORD_AUDIO	2	4	2	2
READ_CONTACTS	16	2	4	
WRITE_CONTACTS	5		3	
BLUETOOTH_ADMIN	3	6	2	
NFC	8	11	1	
READ_SMS	10	2	1	
PROCESS_OUTGOING_CALLS	3	2	1	
CHANGE_WIFI_MULTICAST_STATE	2	2	1	
MANAGE_ACCOUNTS		1	1	
AUTHENTICATE_ACCOUNTS		1	1	
READ_CALL_LOG	12		1	
ACESS_MOCK_LOCATION			1	
BATTERY_STATS	2	1		
SMARTCARD		1		
READ_PROFILE		1		
SET_ANIMATION_SCALE		1		
WRITE_CALL_LOG	4			
WRITE_SMS	2			
RECEIVE_MMS	1			
합계	31	31	28	19

5. 개선방안

모바일 서비스 제공자는 개발단계부터 모바일 앱에 불필요한 접근권한 항목이 포함되지 않도록 하고 운영중인 앱에 대해서도 불필요한 접근권한을 즉시 삭제하도록 해야 한다.

이를 위해 4장에서 제시한 업종별 중점점검 권한 목록을 참조하여 해당 회사에 적합한 권한 체크리스트를 작

성하고 자체보안성 검토 등 모바일 앱에 대한 자체점검 시 이를 활용하여 점검을 수행해야 한다. 또한 서비스가 신규로 개발되거나 변경될 때 권한 체크리스트에 해당 사항을 반영하여 지속적으로 현행화해야 한다.

둘째, 모바일 앱에 대한 자체점검은 통제부서 뿐만 아니라 개발자도 3장에서 제시한 방법과 도구를 활용하여 사용자 동의를 요구하는 권한과 프로그램에서 실제 사용되는 권한에 대한 가시성을 확보해야 한다. 이를 통해 점검대상이 되는 권한의 범위를 축소하여 점검작업의 효율성을 제고할 수 있다.

셋째, 모의해킹 점검 시 점검 체크리스트에 안드로이드 권한에 대한 검토 활동을 추가하여 주기적인 검토가 수행될 수 있도록 절차화해야 한다.

6. 결 론

본 논문에서는 실제 구글 플레이에 등록된 안드로이드 앱을 대상으로 사용자 승인을 요구하는 권한에 대한 현황을 금융회사 업종에 따른 요구권한 특징분류 관점에서 조사하였다. 그리고 금융회사 등 모바일 서비스 제공자가 앱의 최소권한 원칙을 준수하는지 자체점검 수행 시 활용가능한 권한 점검항목을 제시하였다.

또한 실제 구글 플레이에 등록된 금융 관련 앱을 대상으로 오픈소스 도구를 통한 자체점검 방법을 수행한 결과 점검대상 권한목록이 평균적으로 43.1% 감소된 것으로 나타나 최소권한 여부를 효율적으로 점검하는 데 있어 보조도구로서의 유용성을 확인하였다.

그러나 A사의 사례에서 살펴본 바와 같이 3장에서 소개한 점검도구를 불필요 권한을 검출하기 위한 용도로 활용하기에는 오탐 등으로 인해 아직은 어려움이 있는 것으로 확인되었다.

향후에는 Androguard 도구 등을 통한 자체점검 시 점검대상 범위 효율성 제고를 목표로 하여 특히 금융분야 모바일에서 보편적으로 사용되는 다양한 서드파티(3rd party) 라이브러리를 중심으로 안드로이드 API와 권한 간의 연결정보에 대해 연구하고자 한다.

참 고 문 헌 (Reference)

[1] IDC, “Smartphone OS Market Share, 2015 Q2”, “http://www.idc.com/prodserv/smartphone-os-market-share.jsp,” 2015.

[2] Q1 2014 MobileThreatReport, “http://www.f-secure.com/weblog/archives/00002699.html,” 2014

[3] KISA, “Analysis of Android Mobile Platform Security Model”, Aug. 2010.

[4] Adrienne Porter Felt, Elizabeth Hay, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. ″Android permissions: User attention, comprehension, and behavior,″ In Proceedings of Symposium on Usable Privacy and Security, 2012.

[5] Korea Communication Commission, “Guidelines for privacy protection on smart phone application,” Aug. 2015.

[6] Financial Supervisory Service, “Required implementation of privacy information leak prevention for smartphone app,” Aug. 2015.

[7] Google Android Developers Official Site, “http://developer.android.com/reference/android/Manifest.permission.html,” 2015.

[8] Android Full Source Android Manifest File, “https://android.googlesource.com/platform/frameworks/base/+/.../core/res/AndroidManifest.xml,” 2015.

[9] Android-defined Permission Category, “http://developer.android.com/reference/android/Manifest.permission_group.html,” 2015

[10] A tool for reverse engineering Android apk files, “http://ibotpeaches.github.io/Apktool”

[11] Reverse engineering, Malware and goodware analysis of Android applications, “https://code.google.com/p/androguard/wiki/RE#Permissions”

[12] Android Asset Packaging Tool, “http://elinux.org/Android_aapt”

[13] Tools to work with android .dex and java .class files, “http://sourceforge.net/projects/dex2jar”

[14] Jad Decompiler, “http://www.javadecompilers.com/jad”

[15] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, “Pscout: analyzing the android permission specification,” In Proceedings of the 2012 ACM conference on Computer and communications security, pp 217-228. Oct. 2012.

[16] A. P. Felt, K. Greenwood, and D. Wagner. “The effectiveness of application permissions,” in Proceedings ofthe USENIX Conference on Web Application Development, 2011.

[17] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. “Android permissions demystified,” In Proceedings of the18th ACM conference on Computer andcommunications security, pages 627–638. ACM, 2011.
[18] W. Enck, M. Ongtang, and P. McDaniel. “On lightweight mobile phone application certification,” in Proceedings of the 16th ACM conference on Computer and communications security, 2009.
[19] T. Vidas, N. Christin, and L. Cranor. “Curbing androidpermission creep,” In Proceedings of the Web, volume 2, 2011.
[20] Xuetao Wei, Lorenzo Gomez, Lulian Neamtiu, Michalis Faloutsos, “Permission Evolution in the Android Ecosystem,” Dec, 2012.

◐ 저 자 소 개 ◑

조 병 철 (Byung-chul Cho)
2005년 숭실대학교 컴퓨터학부(학사)
2014~현재 고려대학교 정보보호대학원 금융보안학과 석사과정
관심분야 : 전자금융보안, 정보보호정책, SW개발보안, 금융IT감리, etc.
E-mail : jokun@naver.com

최 진 영 (Jin-young Choi)
1982년 서울대학교 컴퓨터학과(학사)
1986년 Drexel University Dept. of Mathematics and Computer(석사)
1993년 University of Pennsylvania Dept. of Computer and Information Science(박사)
1993～1996년 : Research Associate. University of Pennsylvania
1996~1999년: 고려대학교 컴퓨터학과 조교수
1999~2004년: 고려대학교 컴퓨터학과 부교수
2004~현재 : 고려대학교 컴퓨터·전파통신공학부 교수
관심분야 : 정형기법, 임베디드 실시간시스템, 프로그래밍언어, 프로세스 대수, 소프트웨어공학, etc.
E-mail : choi@formal.korea.ac.kr