안드로이드 데이터 공유 취약점 연구

김재형*, 조혁주*, 서승현**, 조태남*
*우석대학교 정보보안학과
**한국인터넷진흥원 인터넷침해대응센터
e-mail: 88rlawogud@hanmail.net, happydolosi@naver.com
seosh@kisa.or.kr, tncho@ws.ac.kr

A Study on Vulnerabilities in the Android Data Sharing

Jae-Hyeong Kim*, Hyeok-Ju Cho*, Seung-Hyun Seo**, Taenam Cho*
*Dept. of Information Security, Woosuk University
**Internet Incidents Response Division, KISA &
Dept. of Computer Science, Perdue University

요 약

스마트폰의 대중화와 보편화가 이루어지면서 스마트폰 개발 환경도 많은 발전을 이루었다. 스마트폰 운영체제들은 각각 고유의 특성을 가지고 다양한 기능들을 제공한다. 안드로이드에서는 서로 다른 어플리케이션끼리 데이터를 공유하기 위해 ContentProvider를 사용한다. 그러나 공격자가 역공학 기술을 이용할 경우 다른 어플리케이션이 데이터베이스에 접근하여 불법적 정보유출을 할 수 있다는 취약점을 가지고 있다. 본 논문에서는 ContentProvider를 통한 정보유출의 취약점을 분석하였다.

1. 서론

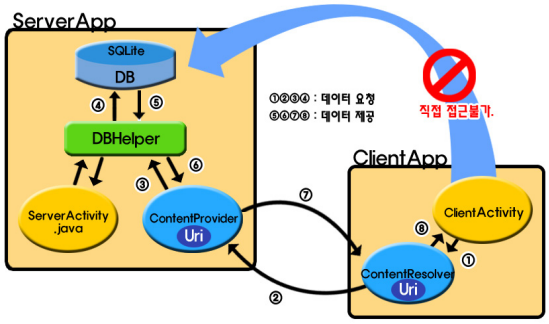
스마트폰의 편리한 기능으로 인해 스마트폰이 대중화되면서 스마트폰 개발 환경도 많은 발전을 이루었다. 애플의 IOS, 구글의 안드로이드와 같은 다양한 개발환경에서 수많은 어플리케이션들이 개발되었다. 특히 안드로이드는 IOS와는 달리 오픈 소스를 제공하기 때문에 어플리케이션 개발환경이 자유로우며 제약이 적지만, 많은 취약점들을 가지고 있다[1]. 이를 악용하여 금전적 손실이나 개인정보 유출, 스마트폰 DDoS 등을 유발시키는 악성코드들이 많다[2]. 악성코드들의 기능이나 동작유형, 그리고 악용하고 있는 취약점들은 매우 다양하다.

본 논문에서는 서로 다른 어플리케이션이 데이터를 공유할 수 있도록 제공되는 안드로이드 특성의 취약점을 연구하였다. [3]에서는 데이터 공유 방식을 이용하여 2~3개의 어플리케이션으로 나누어 악성코드를 작성함으로써 정보를 유출시킬 수 있음을 보였다. 본 연구에서는 데이터 공유를 허용한 정상적인 기존의 어플리케이션 데이터베이스로부터 정보를 유출하는 악성 어플리케이션 작성 기법과 위험성에 대해 연구하였다.

2. 관련연구

2.1 ContentProvider

안드로이드 시스템은 기본적으로 어플리케이션간의 데이터 공유 및 간섭을 허용하지 않는다. 하지만 어플리케이션의 효율성을 위해 관련 어플리케이션간의 공유가 필요하다. 데이터 공유를 위해 안드로이드는 ContentProvider 컴포넌트를[4] 제공한다. 데이터 공유를 위해 자신의 데이터베이스를 다른 어플리케이션에게 공유시키고자 하는 어플리케이션(Server)은 ContentProvider를 사용해야 한다. ContentProvider를 사용함으로써 데이터베이스 엑세스를 관리하고 데이터를 제공한다. ContentProvider는 자신을 유일하게 식별할 수 있는 uri를 가지고 있다. 다른 어플리케이션의 데이터베이스에 접근하고자하는 어플리케이션(Client)은 ContentProvider의 uri를 찾아주는 ContentResolver 객체를 이용하여 해당 데이터를 제공받는다. [그림 1]은 ContentProvider와 ContentResolver를 이용한 어플리케이션간의 데이터베이스 공유 체계를 나타낸 것이다.



[그림 1] ContentProvider와 ContentResolver를 이용한 어플리케이션 공유 체계

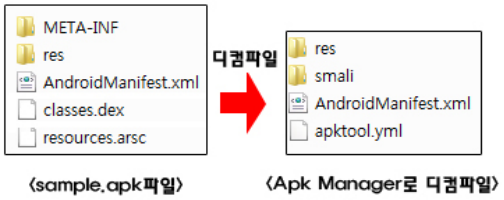
Client 어플리케이션은 데이터베이스를 엑세스하기 위해서 ContentResolver에 접근한다. ContentResolver는 CotnetProvider를 유일하게 식별할 수 있는 uri를 가지고 ContentProvider에게 데이터를 요청한다. ContentProvider는 uri를 확인한 후 데이터베이스에 접근하여 요청 데이터를 추출하고 ContentResolver에게 데이터를 제공한다.

2.2 역공학 기술

안드로이드 어플리케이션은 java 기반으로 개발되어 컴파일하면 classes.dex 파일로 만들어진다. 실행파일은 classes.dex 파일과 필요한 리소스들, 그리고 어플리케이션을 실행하기 위해 안드로이드 시스템에게 필요한 정보를 제공하는 AndroidManifest.xml 등으로 이루어진 압축파일로서 .apk라는 확장자를 갖는다. 역공학 기술은 apk 파일로부터 소스를 복원하는 기술로서 공격자들에게 이용되고 있다. 가장 많이 사용되는 툴로는 ApkManager와 Dedexer가 있다.

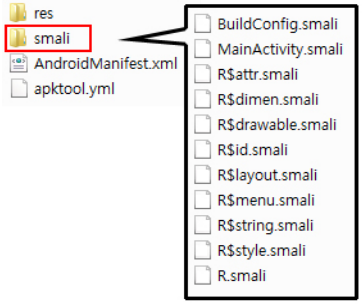
(1) ApkManager

ApkManager 툴은 apk파일을 디컴파일, 컴파일할 수 있는 도구로서 classes.dex 파일로부터 Smali 언어로 된 소스프로그램을 추출하는 기능과 Smali 언어로 구현된 소스파일을 classes.dex파일로 컴파일하는 기능을 제공한다. [그림 2]는 ApkManager를 이용하여 sample.apk 파일로부터 Smali 언어로 작성된 소스코드를[5] 추출한 결과이다.



[그림 2] apk 파일 원본 및 디컴파일 된 폴더

ApkManager 로 추출했을 경우 classes.dex 파일이 사라지고 smali 폴더가 추가되었으며 그 안에는 [그림 3]과 같이 Smali 언어로 된 소스파일들이 생성되었다.

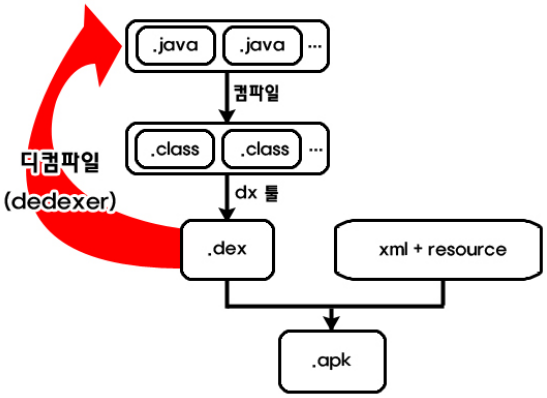


[그림 3] 추출된 Smali 파일

smali 폴더에는 java로 코딩되었던 프로그램들이 smali 언어로 복구되어 있다. Smali 언어를 수정하여 다시 컴파일하면 apk 파일구조가 생성되며 스마트폰에서 실행시킬 수 있다.

(2) Dedexer

apk 파일에 포함된 classes.dex 파일은 안드로이드 가상머신인 dalvik이 인식할 수 있도록 .class파일을 바이트 코드로 변환한 파일이다. 안드로이드 SDK에 포함된 dx라는 툴은 .java 파일들을 classes.dex 포맷으로 변환시킨다[6]. [그림 4]는 java 소스코드로부터 apk 파일이 생성되는 과정으로서 Dedexer 기능과의 관계를 보여주고 있다.



[그림 4] Dedexer를 이용한 classes.dex 디컴파일

classes.dex를 Dedexer 툴을 이용하여 sample apk 파일을 디컴파일 했을 때 [그림 5]와 같이 .java 파일들이 생성된다.

[그림 5] Dedexer 툴을 이용한 java 소스 추출

2.3 Proguard

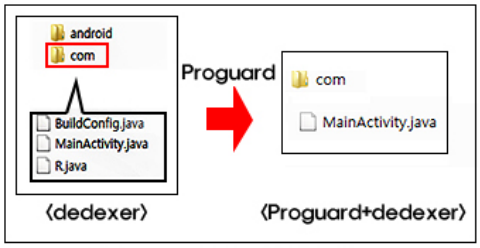
Proguard는 역공학 기술로 인해 소스코드가 노출되는 것을 줄이기 위해 안드로이드가 제공하는 기능이다. 필요 없는 소스를 삭제, 최소화 및 최적화하고 변수 이름을 바꿈으로써 난독화를 해준다[7]. 최신 ADT가 설치된 Eclipse에서 안드로이드 프로젝트를 생성하면 프로젝트의 루트 폴더에 proguard-project.txt와 project.properties 파일이 생성된다. Proguard를 적용하기 위해서는 [그림 6]과 같이 project.properties 안에 표시된 문장을 추가해야한다.

```
# This file is automatically generated by Android Tools.
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!
#
# This file must be checked in Version Control Systems.
#
# To customize properties used by the Ant build system edit
# "ant.properties", and override values to adapt the script to your
# project structure.
#
# To enable ProGuard to shrink and obfuscate your code, uncomment this (available pr
proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt

# Project target.
target=android-8
```

[그림 6] project.properties에서 Proguard 적용

Proguard를 적용한 후 다시 디컴파일 하게 되면 [그림 7] (a)와 같이 소스의 불필요한 부분은 제거되어 파일의 개수가 줄어들고, [그림 7]의 (b)와 같이 변수명이 변경된 것을 볼 수 있다.



(a) Proguard 적용 후 디컴파일

```
(Proguard 적용전)
public void onCreate(Bundle bundle)
{
    super.onCreate(bundle);
    setContentView(0x7f030000);
    Button button = (Button)findViewById(0x7f070002);
    id1 = (EditText)findViewById(0x7f070000);
    pw1 = (EditText)findViewById(0x7f070001);
    button.setOnClickListener(new _cls1());
}

(Proguard 적용후)
public void onCreate(Bundle bundle)
{
    super.onCreate(bundle);
    setContentView(0x7f030000);
    Button button = (Button)findViewById(0x7f070002);
    a = (EditText)findViewById(0x7f070000);
    b = (EditText)findViewById(0x7f070001);
    button.setOnClickListener(new b(this));
}
```

(b) Proguard 적용 후 소스 변화

[그림 7] Proguard로 인한 파일 축소 및 변수명 변경

3. 취약점 분석 및 실험

3.1 취약점 분석

ContentProvider는 다른 어플리케이션이 데이터베이스를 필요로 할 때, ContentProvider를 식별할 수 있는 uri를 통하여 접근을 허용한다. uri는 AndroidManifest.xml과 ContentProvider를 사용하는 java 소스에 기술된다. Proguard를 사용한 경우에도 데이터베이스를 제공하는 어플리케이션을 ApkManager 와 Dedexer 툴을 이용하여 디컴파일 해보면 [그림 8]과 같이 데이터베이스에 대한 내용과([그림 8](a)) uri가 추출된다([그림 8](b)). 따라서 uri로 추출된 데이터베이스 내용에 접근/수정/삭제가 가능하다. 본 논문에서는 이런 취약점을 이용한 악성 어플리케이션을 구현하여 실험하였다.

```
# virtual methods
.method public onCreate(Landroid/database/sqlite/SQLiteDatabase;)V
    .locals 1
    .parameter "db"

    .prologue
    .line 103
    const-string v0, "CREATE TABLE addresstable (_id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT,tel TEXT,addr TEXT);"

    invoke-virtual {p1, v0}, Landroid/database/sqlite/SQLiteDatabase;->execSQL(Ljava/lang/St

    .line 104
    return-void
.end method
```

〈ApkManager 툴을 이용한 데이터베이스 추출〉

```
class DBHelper extends SQLiteOpenHelper
{

    public DBHelper(Context context)
    {
        super(context, "addressbook.db", null, 1);
    }

    public void onCreate(SQLiteDatabase sqlitedatabase)
    {
        sqlitedatabase.execSQL("CREATE TABLE addresstable (_id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT, tel TEXT,addr TEXT);");
    }

    public void onUpgrade(SQLiteDatabase sqlitedatabase, int i, int j)
    {
        sqlitedatabase.execSQL("DROP TABLE IF EXISTS addresstable");
        onCreate(sqlitedatabase);
    }
}
```

〈Dedexer 툴을 이용한 데이터베이스 추출〉

(a) 데이터베이스 추출

```
<provider android:name=".CP"
    android:authorities="com.addressbook.data"
/>
```

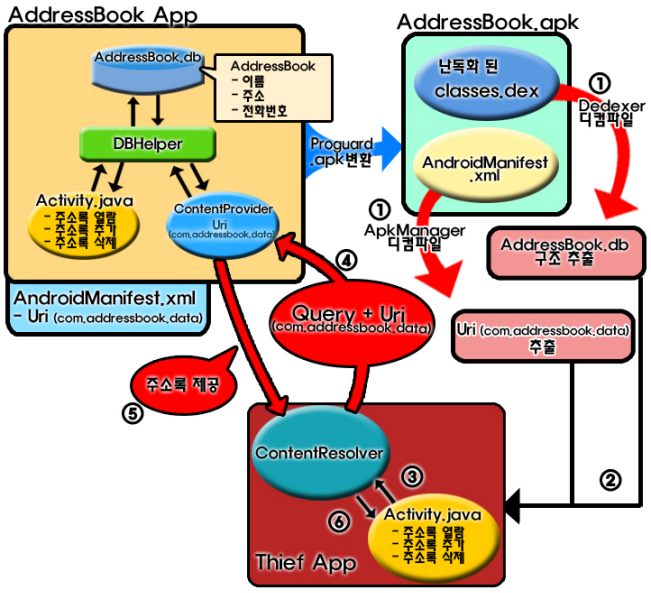
〈ApkManager 툴을 이용한 AndroidManifest.xml에서 uri추출〉

(b) uri 추출

[그림 8] 역공학 기술을 이용한 데이터베이스 내용과 uri 추출

3.2 실험

본 논문의 실험에서 사용한 주소록 어플리케이션(AddressBook)은 데이터베이스에 주소의 정보를 저장하고 다른 어플리케이션이 필요로 할 때 주소록의 정보를 제공하기 위해 ContentProvider를 사용한다. [그림 9]는 AddressBook 어플리케이션을 디컴파일 하여 데이터베이스의 내용과 uri를 추출한 후 악성 어플리케이션(Thief)을 작성하여 AddressBook 어플리케이션의 정보를 접근/수정/삭제하는 과정이다.



[그림 9] 정보 유출 어플리케이션

[그림 10] (a)의 AddressBook 어플리케이션으로부터 (b)와 같이 Thief가 데이터베이스를 수정하면 (c)와 같이 AddressBook 어플리케이션에 반영되는 것을 볼 수 있다. 유사한 방법으로 Theif는 AddressBook의 정보를 외부로 유출시킬 수 있다.

[그림 10] Thief 어플리케이션을 통한 AddressBook 어플리케이션 데이터 변조

4. 결론 및 향후 과제

본 논문에서는 데이터베이스를 기반으로 하는 안드로이드 어플리케이션이 다른 어플리케이션과 데이터를 공유하기 위해 사용하는 ContentProvider의 취약점을 분석하였다. 이 취약점을 이용하여 공격자가 ContentProvider의 uri와 어플리케이션의 데이터베이스 정보를 추출하여 타 어플리케이션의 데이터베이스에 접근/수정/삭제 할 수 있음을 샘플 악성 프로그램을 통해 보였다.

ContentProvider를 저장하는 어플리케이션에서는 퍼미션을 통해 동일한 서명을 갖는 어플리케이션끼리만 데이터를 공유하도록 설정할 수 있다. 그러나 많은 개발자들은 기정치를 사용함으로써 적절한 퍼미션 설정을 하지 않을 수 있다. 또한 적절한 퍼미션을 사용한 경우에는 기존 악성 어플리케이션과 같이 공격자가 퍼미션을 바꾼 후 공격자의 서명으로 리패키징하는 방법을 사용할 수 있을 것이다.

Proguard를 이용하면 변수 이름의 변경 등 난독화가 이루어지지만, 데이터베이스에 대한 정보는 난독화 되지 않는다. 향후에는 본 논문에서 분석한 취약점을 해결하기 위한 방법으로서 데이터베이스에 대한 난독화를 제공하는 향상된 Proguard에 대해 연구할 예정이다.

참고문헌

[1] 조태남, 문남미, "스마트폰 응용 프로그램 개발 및 코드 서명," 정보보호학회지, 제21권 제1호, pp.19-25, 2011. 2.

[2] Seung-Hyun Seo, Dong-Guen Lee, Kangbin Yim, "Analysis on maliciousness for mobile application," IMIS 2012, pp.126-129, 2012. 7.

[3] 전철, 조유근, 홍지만, "안드로이드 어플리케이션의 협력적인 개인 정보 유출 탐지," 한국컴퓨터종합학술대회 논문집, 제39권 제1호, pp.92-94, 2012. 5.

[4] Android Developer - ContentProvider , http://developer.android.com/intl/ko/reference/android/content/ContentProvider.html.

[5] http://code.google.com/p/smali/.

[6] 위키피디아 달빅 가상 머신, http://ko.wikipedia.org/wiki/%EB%8B%AC%EB%B9%85_(%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4).

[7] Android Developer - Proguard, http://developer.android.com/tools/help/proguard.html.