<http://www.voidcn.com/article/p-qlbghrpd-bbb.html>

# ****关于Activity的****getReferrer()****之一：如何在Activity中获取调用者？****

## ****0. 为什么要在Activity中获取调用者？****

主要是**安全**和**定制**两个方面的需求吧。

* **安全需求：**   
  一般Activity如果已经对外开放了（即exported为true，或者加了Intent-filter），那么对Activity的保护就会降低，形成Activity攻击面，引入了风险。如果只对某几个app开放其Activity，则可以获取调用者的信息，并进行控制。
* **定制需求：**   
  如果被启动的Activity想对特定的调用者进行定制操作，则需要知道调用者是谁。

## ****1. 在Activity中获取调用者的方法****

注：先列出所有可能的方法，后面会给出哪些方法可行。

1. **Binder.getCallingUid()** 和 **Binder.getCallingPid()**，然后根据uid，pid查找到包名
2. Activity的**getCallingPackage()** 和 **getCallingActivity()**
3. Activity的**getReferrer()** **【注意：Android 5.1（Api level 22）中才引入的】**
4. 反射的方式获取Activity的**mReferrer**: **reflectGetReferrer()**【注：自定义函数，目的是获取到**android.app.Activity**类的**mReferrer**的值，也需要Api level 22（含）之后才能使用】

这里先给出结论：   
方法1：不能在调用者startActivity()的时候获取到调用者的包名，只能用于Activity用到的Binder同步调用的地方。

方法2： 在特定情况下可以使用**getCallingPackage()** 和 **getCallingActivity()**，即如果Activity是通过**startActivityForResult**启动的，则可以使用。

方法3： Activity的**getReferrer()**是不可靠的，因为调用者可以自己设置referrer的值。

方法4：是对方法3的改进，消除**getReferrer()**可能返回的不可靠的值，直接获取可靠的**mReferrer**值（目前来看是可靠的）。

## ****2. 测试代码：验证上面的各种方法****

### ****2.1 场景和测试代码****

**场景：**   
有2个app，一个包名为**com.galian.mainapp**，app名为**MainApp**；另一个包名为**com.galian.secondapp**，app名为**SecondApp**。

SecondApp启动MainApp的MainActivity，在MainApp的MainActivity中检查调用者是谁。

**需要说明的是，本文中有referrer和mReferrer两种说法，这两种是有区别的。**   
其中**referrer**是指Activity的**getReferrer()**的返回值。   
**mReferrer**是Activity类中的成员变量。   
**getReferrer()**的代码后面会给出。

* **被调用者：MainApp**

[***MainApp的全部代码***](javascript:void())

MainApp的界面：   
其实这是从Launcher app启动MainApp的情况，可以看到Launcher的包名（com.sec.android.app.launcher）。

MainActivity

主要代码：

*// 测试各种方法，并显示*

private void checkCallingApp() {

StringBuffer stringBuffer = new StringBuffer();

stringBuffer.append("\n\n");

*// Binder.getCallingUid(), Binder.getCallingPid()*

stringBuffer.append("1. caller uid = ").append(Binder.getCallingUid()).append(", pid = ")

.append(Binder.getCallingPid()).append("\n\n");

*// getReferrer(), Activity自带的获取referrer的方法，Api Level22之后才可以用*

Uri referrer = getReferrer();

stringBuffer.append("2. caller (referrer uri) : ");

if (referrer != null) {

stringBuffer.append(referrer.toString());

} else {

stringBuffer.append("null");

}

stringBuffer.append("\n\n");

*// reflectGetReferrer()反射的方式获取mReferrer*

String referrerStr = reflectGetReferrer();

stringBuffer.append("3. caller (reflect mReferrer) : ").append(referrerStr).append("\n\n");

*// Activity的getCallingPackage()*

String callingPkg = getCallingPackage();

stringBuffer.append("4. callingPkg: ").append(callingPkg).append("\n\n");

*// Activity的getCallingActivity()*

ComponentName componentName = getCallingActivity();

stringBuffer.append("5. caller componentName: ");

if (componentName != null) {

stringBuffer.append(componentName.toString());

} else {

stringBuffer.append("null");

}

mTextView.setText(getString(R.string.caller\_name, stringBuffer.toString()));

}

反射的方式获取mReferrer

*// 反射的方式获取mReferrer*

private String reflectGetReferrer() {

try {

Class activityClass = Class.forName("android.app.Activity");

Field refererField = activityClass.getDeclaredField("mReferrer");

refererField.setAccessible(true);

String referrer = (String) refererField.get(MainActivity.this);

return referrer;

}

}

* **调用者：SecondApp**

[***SecondApp的全部代码***](javascript:void())

SecondApp的界面：

SecondMainActivity

SecondApp的代码在各种情况中给出。

### ****2.2 情况1: 调用者没有修改referrer****

SecondApp启动MainApp的**MainActivity**：

*// 测试getReferrer()： 没有修改referrer的情况*

Intent intent = new Intent();

intent.setClassName("com.galian.mainapp", "com.galian.mainapp.MainActivity");

startActivity(intent);

MainApp的显示结果：

这里写图片描述

结果分析：

* uid为10189，pid为29089，这是MainApp自身。

shell@c7ltechn:/ $ ps | grep 29089

u0\_a189 29089 2366 1414876 66788 SyS\_epoll\_ 0000000000 S com.galian.mainapp

* getReferrer()的返回值为**android-app://com.galian.secondapp**，此值**可以得到调用者的包名**。
* 反射方式获取到的**mReferrer**为：**com.galian.secondapp**，此值**可以得到调用者的包名**。
* getCallingPackage()和getCallingActivity()都返回null

### ****2.3 情况2：调用者通过****Intent.EXTRA\_REFERRER****修改referrer的值****

SecondApp启动MainApp的**MainActivity**：

*// 测试`getReferrer()`：修改referrer，通过Intent.EXTRA\_REFERRER*

Intent intent = new Intent();

intent.setClassName("com.galian.mainapp", "com.galian.mainapp.MainActivity");

*// case 1*

intent.putExtra(Intent.EXTRA\_REFERRER, Uri.parse("android-app://com.test.app"));

startActivity(intent);

MainApp的显示结果：

这里写图片描述

结果分析：

* uid为10189，pid为29089，这是MainApp自身。
* getReferrer()的返回值为**android-app://com.test.app**，此值**不是**真正的调用者的包名。
* 反射方式获取到的**mReferrer**为：**com.galian.secondapp**，此值**可以得到调用者的包名**。
* getCallingPackage()和getCallingActivity()都返回null

### ****2.4 情况3：调用者通过****Intent.EXTRA\_REFERRER\_NAME****修改referrer的值****

SecondApp启动MainApp的**MainActivity**：

*// 测试`getReferrer()`：修改referrer，通过Intent.EXTRA\_REFERRER\_NAME*

Intent intent = new Intent();

intent.setClassName("com.galian.mainapp", "com.galian.mainapp.MainActivity");

*// case 2*

intent.putExtra(Intent.EXTRA\_REFERRER\_NAME, "android-app://com.example.app");

startActivity(intent);

MainApp的显示结果：

这里写图片描述

结果分析：

* uid为10189，pid为29089，这是MainApp自身。
* getReferrer()的返回值为**android-app://com.example.app**，此值**不是**真正的调用者的包名。
* 反射方式获取到的**mReferrer**为：**com.galian.secondapp**，此值**可以得到调用者的包名**。
* getCallingPackage()和getCallingActivity()都返回null

### ****2.5 情况4：测试getCallingPackage()和getCallingActivity()****

同时调用者通过**Intent.EXTRA\_REFERRER**修改referrer的值。

SecondApp通过**startActivityForResult()**启动MainApp的**MainActivity**：

*// 测试getCallingPackage() 和 getCallingActivity()*

Intent intent = new Intent();

intent.setClassName("com.galian.mainapp", "com.galian.mainapp.MainActivity");

*// case 1*

intent.putExtra(Intent.EXTRA\_REFERRER, Uri.parse("android-app://com.test.app"));

intent.putExtra("GET\_DATA", true);

startActivityForResult(intent, REQUEST\_CODE\_TEST);

MainApp的显示结果：

这里写图片描述

结果分析：

* uid为10189，pid为29089，这是MainApp自身。
* getReferrer()的返回值为**android-app://com.test.app**，此值**不是**真正的调用者的包名。
* 反射方式获取到的**mReferrer**为：**com.galian.secondapp**，此值**可以得到调用者的包名**。
* getCallingPackage()返回**com.galian.secondapp**，getCallingActivity()返回**ComponentInfo{com.galian.secondapp/com.galian.secondapp.SecondMainActivity}**。**可以得到调用者的包名**。

## ****3. 结论****

从上面的测试结果可以得到下面的结论：

1. 通过反射的方式（**reflectGetReferrer()**）获取到的**mReferrer**，是调用者的包名，目前来看是可靠的，但是需要在Android5.1(Api level 22)以及之后才能用。
2. **getCallingPackage()** 和 **getCallingActivity()**只有在**startActivityForResult()**的时候才可以得到调用者的包名。
3. Activity的**getReferrer()**是不可靠的，因为调用者可以自己设置referrer的值。所以不能依赖此值来判断调用者。
4. **Binder.getCallingUid()** 和 **Binder.getCallingPid()**一般用在同步调用中，在这几个情况中并不适用。

## ****4. 关于****mReferrer****的细节****

### ****4.1 Activity的****getReferrer()

**需要注意的是，此方法是在Android 5.1 （Api level 22）中引入的，Android 5.1之前是不能使用的。**

Intent.java

public static final String EXTRA\_REFERRER

= "android.intent.extra.REFERRER";

public static final String EXTRA\_REFERRER\_NAME

= "android.intent.extra.REFERRER\_NAME";

Activity.java

public Uri getReferrer() {

Intent intent = getIntent();

*// 优先从Intent的Intent.EXTRA\_REFERRER数据获取Uri，作为referrer*

Uri referrer = intent.getParcelableExtra(Intent.EXTRA\_REFERRER);

if (referrer != null) {

return referrer;

}

*// 如果之前没有获取到，则从intent的Intent.EXTRA\_REFERRER\_NAME数据获取，并转换成Uri*

String referrerName = intent.getStringExtra(Intent.EXTRA\_REFERRER\_NAME);

if (referrerName != null) {

return Uri.parse(referrerName);

}

*// 如果上面都没有获取到，则将mReferrer转换成android-app://的形式*

if (mReferrer != null) {

return new Uri.Builder().scheme("android-app").authority(mReferrer).build();

}

return null;*// 都没获取到的话，返回null*

}

### ****4.2 反射的方式获取Activity的****mReferrer

**需要注意的是，此方法是基于getReferrer()（mReferrer）的，所以也必须在Android 5.1 （Api level 22）及 5.1 之后才能用。**

自定义方法：

private String reflectGetReferrer() {

try {

Class activityClass = Class.forName("android.app.Activity");

Field refererField = activityClass.getDeclaredField("mReferrer");

refererField.setAccessible(true);

String referrer = (String) refererField.get(MainActivity.this);

return referrer;

} catch (ClassNotFoundException | IllegalAccessException | NoSuchFieldException e) {

e.printStackTrace();

return "No referrer";

}

}

### ****4.3 Activity中的****mReferrer****是如何传递过来的？****

由于内容较多，请参考另一篇博文[***《关于Activity的getReferrer()之二：调用者的包名是如何传给mReferrer的，兼谈startActivity的详细流程》***](http://www.voidcn.com/article/p-bwimkokd-bbb.html)

## ****5. 参考****

1. [***https://developer.android.com/guide/topics/manifest/uses-sdk-element.html***](javascript:void())   
   查看 API level 与 Android版本 的对应关系
2. 查看Android各个版本的代码：[***http://androidxref.com/***](javascript:void())
3. **getReferrer()**的官方介绍： [***https://developer.android.com/reference/android/app/Activity.html#getReferrer()***](javascript:void())

## Android中Activity/Service获取调用者的信息（FIDO UAF Client获取调用者的信息）

*  2016年12月9日

*  [Android](http://sickworm.com/?cat=39), [FIDO UAF](http://sickworm.com/?cat=53)

实现UAF协议的时候，Client需要获取调用者的信息（获得其APK的签名）。用中文查了半天没查到获取Activity的方法，用英文一下就搜出来了（主要还是看英文累脑子）

Activity：

getCallingActivity()  
getCallingPackage()

注意：  
只有调用者使用的是startActivityForResult()，且Intent不设置NEW\_TASK时才可以获取，调用startActivity()得到的是null。

Service：

Binder.getCallingUid()  
Binder.getCallingPid()

注意：  
当AIDL使用的是oneway（异步）声明时，Binder.getCallingPid()返回的是0 。

获取到Uid之后，使用getPackageManager().getPackagesForUid(uid)获取到对应的包名。如果多个apk使用了shareUserId的话，返回值将会是多个包，这时候就没办法知道具体是哪个package调用的了。不过使用shareUserId的前提是使用相同的签名文件签名，而UAF要求是得到调用者apk签名的hash，这样的话哪个package调用结果都一样了。

版权所有，转载请注明出处：  
<http://sickworm.com/?p=270>