jiangwei0910410003的专栏

目录视图

Android中实现静态的默认安装和卸载应用

2014-07-02 22:43 15200人阅读 评论

分类： Android（96）

目录(?) [+]

最近好长时间都没有写blog了，主要是因为最近工作上的事以及下载Android源码的事耽误的(下载源码这件事会在后续的blog中呀~~)，那么今天来写点什么呢？主要的灵感来自于早上看新闻看到一篇文章说有一款应用在后台中卸载用户

手机中的所有浏览器的app，不会被用户察觉，但是最后百度浏览器还是用反侦察技术找到这个邪恶的应用然后将其告上法庭了。

够实现应用的静态安装和卸载呢？就是不让用户知道，下面就来一步一步的介绍一下实现步骤：

一、访问隐藏的API方式进行静态的默认安装和卸载

1.系统安装程序

android自带了一个安装程序---/system/app/PackageInstaller.apk.大多数情况下，我们手机上安装应用都是通过这个ap

的。代码使用也非常简单：

```java
/* 安装apk */
public static void installApk(Context context, String fileName) {
    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_VIEW);
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    intent.setDataAndType(Uri.parse("file://" + fileName),"application/vnd.android.package-archive");
    context.startActivity(intent);
}

/* 卸载apk */
public static void uninstallApk(Context context, String packageName) {
    Uri uri = Uri.parse("package:" + packageName);
    Intent intent = new Intent(Intent.ACTION_DELETE, uri);
    context.startActivity(intent);
}
```

通过发一个Intent,把应用所在的路径封装整uri.之后默认启动了PackageInstaller.apk来安装程序了。

但是此种情况下，仅仅是个demo而已，很难达到开发者的需求。如：

1).界面不好

2).被用户知晓

3).什么时候安装完了，卸载完了呢？

当然这里关于第三点的话，为了达到自己的需求，相信很多人都会接着来监听系统安装卸载的广播，继续接下来的代码逻辑。

监听系统发出的安装广播

在安装和卸载完后，android系统会发一个广播

android.intent.action.PACKAGE_ADDED（安装）

android.intent.action.PACKAGE_REMOVED（卸载）

咱们就监听这广播，来做响应的逻辑处理。实现代码：

```java
public class MonitorSysReceiver extends BroadcastReceiver{

    @Override
    public void onReceive(Context context, Intent intent){
        //接收安装广播
        if (intent.getAction().equals("android.intent.action.PACKAGE_ADDED")) {
            //TODO
        }
        //接收卸载广播
        if (intent.getAction().equals("android.intent.action.PACKAGE_REMOVED")) {
            //TODO
        }
    }
}
```

AndroidManifest.xml文件中的配置：

```html
<receiver android:name=".MonitorSysReceiver">
          <intent-filter>
           <action android:name="android.intent.action.PACKAGE_ADDED" />
           <action android:name="android.intent.action.PACKAGE_REMOVED" />
          </intent-filter>
</receiver>
```

到此，确实安装卸载的整体流程都知道了，但是这个效果肯定是无法达到项目的需求。

一般这种应用商店类(豌豆荚)的项目，肯定是会要自定义提示框效果的安装卸载功能，而不是调用系统的安装程序。

那咱就要想法子实现静默安装、卸载咯。

下面这种调用系统隐藏api接口来实现静默安装卸载，是比较大众靠谱的，实现自定义的提示界面

(关于这个调用系统隐藏的api接口，我们在之前讲到如何获取手机电量的一篇文章中介绍过了

http://blog.csdn.net/jiangwei0910410003/article/details/25994337)

2.系统隐藏的api

隐藏api,顾名思义，普通情况下肯定是调用不到的。翻翻源码\frameworks\base\core\java\android\content\pm目录下

PackageManager.java，应该发现在注释行里有加上@hide声明。调用的安装下载接口如下：

安装接口：

```java
public abstract void installPackage(Uri packageURI,IPackageInstallObserver observer, int flags,String i
```

卸载接口：

```java
public abstract void deletePackage(String packageName,IPackageDeleteObserver observer, int flags);
```

并且都是抽象方法，需要咱们实现。

看参数里IPackageInstallObserver observer一个aidl回调通知接口，当前目录中找到这接口：

```java
package android.content.pm;

/**
 * API for installation callbacks from the Package Manager.
 * @hide
 */
oneway interface IPackageInstallObserver {
    void packageInstalled(in String packageName, int returnCode);
}
```

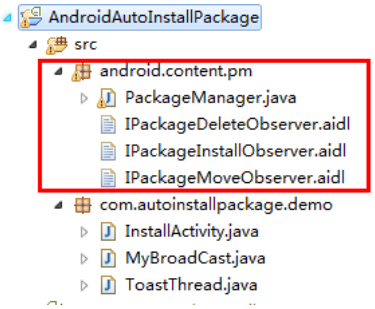
好吧，这里有现成的干货，咱拿过来直接用呗（当然如果没有源码的那就算了，那能实现的只是demo）。具体步骤：

从源码中拷贝要使用的aidl回调接口：IPackageInstallObserver.aidl、IPackageDeleteObserver.aidl当然完全可以把

录，这样就不会报错了。

作者项目里面用到了pm,所以把PackageManager.java以及涉及到的一些文件也拷贝过来了，不然eclipse报找不到PackageManage

对象。结构如下：



注：此处的包名**android.content.pm**一定要和源码目录结构一致，不然源码里编译会提示找不到aidl接口。一切朝源码编译

此处有2种方式实现：

1.直接只取IPackageDeleteObserver.aidl和IPackagerInstallObserver.aidl、IPackageMoveObserver.aidl等要使用的接口，

bindService来和系统连接服务，然后直接调用接口即可（这种没有方式作者没试过，不过原理上来说应该是可行的，除非系统

这个Service实现这个接口。有需求的可以深究下）

2.作者此处的方法是直接拷贝了源码PackageManager.java等文件过来，不过靠过来之后eclipse会提示一些接口错误，但这里作

上面那几个.java文件都放空了，因为用不到，只是为了编译过才拷贝了那么多文件。最简单的就是直接拷贝4个文件即可：

PackageManager.java

IPackageDeleteObserver.aidl

IPackagerInstallObserver.aidl

IPackageMoveObserver.aidl

然后把PackageManager.java中报的异常的接口都注释掉即可

实现回调接口，代码如下：

安装的回调接口：

```java
/*静默安装回调*/
class MyPakcageInstallObserver extends IPackageInstallObserver.Stub{

    @Override
    public void packageInstalled(String packageName, int returnCode) {
        if (returnCode == 1) {
            Log.e("DEMO","安装成功");
            new ToastThread(InstallActivity.this,"安装成功").start();
        }else{
            Log.e("DEMO","安装失败,返回码是:"+returnCode);
            new ToastThread(InstallActivity.this,"安装失败,返回码是:"+returnCode).start();
        }
    }
}
```

安装的方法：

```java
```

```
String fileName = Environment.getExternalStorageDirectory() + File.separator + "baidu"+File.separator +
Uri uri = Uri.fromFile(new File(fileName));
int installFlags = 0;
PackageManager pm = getPackageManager();
try {
PackageInfo pi = pm.getPackageInfo("com.UCMobile",PackageManager.GET_UNINSTALLED_PACKAGES);
if(pi != null) {
installFlags |= PackageManager.INSTALL_REPLACE_EXISTING;
}
} catch (NameNotFoundException e) {
}
MyPakcageInstallObserver observer = new MyPakcageInstallObserver();
pm.installPackage(uri, observer, installFlags, "com.UCMobile");
```

从代码中可以看到我们想安装的是从SD卡中的baidu文件夹中的UC.apk

(所以在测试程序的时候需要将UC.apk拷贝到SD卡中的baidu文件夹中，或者你可以自定义一个文件的存放目录)

卸载原理是一样的

卸载的回调接口：

[java]

```
/* 静默卸载回调 */
class MyPackageDeleteObserver extends IPackageDeleteObserver.Stub {

    @Override
    public void packageDeleted(String packageName, int returnCode) {
        if (returnCode == 1) {
            Log.e("DEMO","卸载成功...");
            new ToastThread(InstallActivity.this,"卸载成功...").start();
        }else{
            Log.e("DEMO","卸载失败...返回码:"+returnCode);
            new ToastThread(InstallActivity.this,"卸载失败...返回码:"+returnCode).start();
        }
    }
}
```

卸载的功能：

[java]

```
PackageManager pm = InstallActivity.this.getPackageManager();
IPackageDeleteObserver observer = new MyPackageDeleteObserver();
pm.deletePackage("com.UCMobile", observer, 0);
```

这里我们一定要传一个包名。

自此，静默安装卸载代码实现。最后在AndroidManifast.xml中要注册权限和添加为系统用户组

[html]

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.autoinstallpackage.demo"
    android:versionCode="1"
    android:versionName="1.0.19"
    android:sharedUserId="android.uid.system">

    <uses-sdk android:minSdkVersion="4"/>

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.INSTALL_PACKAGES" />
    <uses-permission android:name="android.permission.DELETE_PACKAGES" />
    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    ...

</manifest>
```

这里要注意的地方有两个：第一个必须要添加：

[html]

```
android:sharedUserId="android.uid.system"
```

这个是将其应用注册成系统用户组中，如果不加这行代码的话，会出现报错，但是加了这行代码之后，如果还报错的话，重新cl

是Eclipse的一个bug吧

还要添加安装和卸载的权限：

```
[html]
01. <uses-permission android:name="android.permission.INSTALL_PACKAGES" />
02. <uses-permission android:name="android.permission.DELETE_PACKAGES" />
```

这时候上面的工作都做完之后，那么我们就来用Eclipse运行一下吧：

```
AndroidAutoInstallPackage] Installation error: INSTALL_FAILED_SHARED_USER_INCOMPATIBLE
AndroidAutoInstallPackage] Please check logcat output for more details.
AndroidAutoInstallPackage] Launch canceled!
```


好吧，貌似不是那么的顺利，出现一个安装错误，其实这个信息去百度一下之后会找到很多的相关资料，因为这个问题有很多人

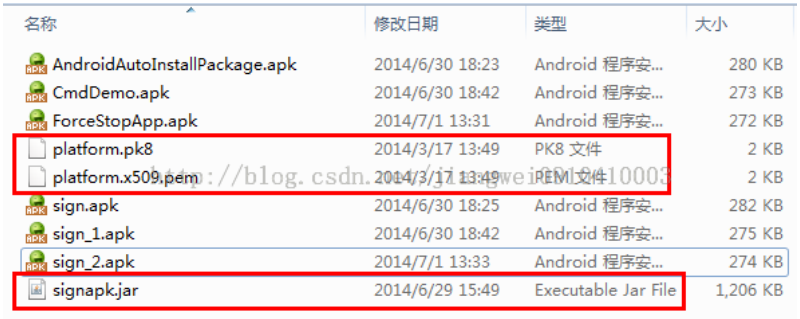
到过，所以解决的方法也是很多的，就是需要将我们的应用签名成系统级的即可，那么我们该怎么办呢？

网上普遍的两种方法：

第一种是：得到platform.pem,platform.x509.pem,signapk.jar这三个文件进行签名：签名的命令很简单：

java -jar signapk.jar platform.x509.pem platform.pem 需要签名的apk 签名之后的apk

下图是我签名的操作：



名称	修改日期	类型	大小
AndroidAutoInstallPackage.apk	2014/6/30 18:23	Android 程序安...	280 KB
CmdDemo.apk	2014/6/30 18:42	Android 程序安...	273 KB
ForceStopApp.apk	2014/7/1 13:31	Android 程序安...	272 KB
platform.pk8	2014/3/17 13:49	PK8 文件	2 KB
platform.x509.pem	2014/3/17 13:49	PEM 文件	2 KB
sign.apk	2014/6/30 18:25	Android 程序安...	282 KB
sign_1.apk	2014/6/30 18:42	Android 程序安...	275 KB
sign_2.apk	2014/7/1 13:33	Android 程序安...	274 KB
signapk.jar	2014/6/29 15:49	Executable Jar File	1,206 KB


```
C:\Users\weijiang204321>cd C:\Users\weijiang204321\Desktop\sign

C:\Users\weijiang204321\Desktop\sign>java -jar signapk.jar platform.x509.
pem platform.pk8 AndroidAutoInstallPackage.apk sign.apk

C:\Users\weijiang204321\Desktop\sign>
```

第二种方法：很麻烦的，为什么说麻烦呢？因为需要下载源代码，然后将我们的应用拷贝到指定的目录即可：

1.如果你有系统源码，最简单的就是将eclipse里面的应用拷贝到系统里面，然后编译系统，会生

成out/target/product/generic/system/app/abc.apk，这个应用就是经过系统签名了。具体方法如下：

将应用文件夹复制到源码目录：packages/experimental/project_name/里面，并且只保留src、res、libs、androidmanifest.x

一个Android.mk文件，内容如下：

```
LOCAL_PATH:= $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE_TAGS := optional

LOCAL_SRC_FILES := $(call all-subdir-java-files)

LOCAL_PACKAGE_NAME := abc

LOCAL_CERTIFICATE := platform

include $(BUILD_PACKAGE)
```

然后进行编译即可

其实我们在将第一种方法的时候有一个问题就是那三个签名的文件到哪里能搞到，这个吗？可以到网上去搜索的，但是本人在下

下来然后进行签名，结果总是失败，于是，我就放弃了从网上去下载，那么还有什么方法可以得到呢？好吧，还得去下载Androi

码，编译之后可以在指定的目录中找到这三个文件

但是还没有完，不管是第一种方法还是第二种方法，我在试验的过程中还是失败，原因是什么呢？因为我们如果用这三个文件进
签名或者把工程放到源码中进行编译的话，都必须要注意一点：就是签名的文件的系统版本必须和安装到设备的系统的版本一样
说白一点：现在假如签名的那三个文件是从下载的编译之后的源码中得到的，那么这三个文件的版本就是这个Android源码的版
比如是4.4版本的，那么我们用这三个文件进行签名之后的apk包只能安装到系统版本为4.4的机子上，同理如果我们将工程直接
Android源码中进行编译的话，假如Android源码的系统版本是4.4,那么编译之后的apk也只能安装到系统版本为4.4的机子上。

所以我们从网上下载到的签名的三个文件进行签名之后总是失败，所以这种方式总是不靠谱的，因为我们从网上下载的签名文件
定不知道他的版本号，但是我们可以将签名之后的包安装到每个版本的系统中进行尝试，所以最靠谱的方式还是我们下载源码然
得到这三个签名的文件即可，因为我们自己下载的源码肯定知道版本的。

这三个文件对应的目录是：

signapk.jar: 源码目录/out/host/linux-x86/framework/signapk.jar

platform.pk8和platform.x509.pem: 源码目录/build/target/product/security/platform.pk8&platform.x50

当我们使用第一种方法的时候，我们需要用Eclipse中导出一个未签名的包，这个很简单了，这里就不介绍了~~

运行的结果：

因为手头上没有4.4系统的机子，又不想刷机，所以就用模拟器了。在这个过程中安装和卸载可能要一段时间，所以要等待一会，

那么我们就介绍了使用隐藏的api来进行默认的安装和卸载app,

Demo工程下载地址：

http://download.csdn.net/detail/jiangwei0910410003/7584423

导入工程之后，需要做的几步：

在SD卡中拷贝一个UC.apk到baidu文件夹(需要新建的)

导出未进行签名的包

下载签名的工具：

http://download.csdn.net/detail/jiangwei0910410003/7584441

按照上面的签名步骤进行操作

然后找到一个系统是4.4版本的测试机或者模拟器进行安装包(因为我的签名文件的版本是4.4)

二、通过命令进行静态的默认安装和卸载

1、需要签名文件，不需要root权限

下面我们再来看一下如何使用命令的方式来实现默认的安装和卸载app，我们知道可以使用命令：

adb install apk文件

adb uninstall 包名

进行安装和卸载操作。

我们在代码中可以使用Runtime类进行操作：

```java
Runtime.getRuntime().exec("adb uninstall com.UCMobile");
```

或者ProcessBuilder：

```java
new ProcessBuilder().command("adb","uninstall","com.UCMobile").start();
```

看到这两个类的区别在于，第一种是将命令一起写完，第二种是将命令用空格分隔，然后将其当做参数进行传递的，其实commar

方法的参数是一个String可变参数类型的

但是我们执行上面的代码发现总是执行失败，结果才发现，这些命令执行的目录不是在shell层的，但是手机中执行的命令

要在shell层中的，所以会有问题的，但是我们可以使用pm命令，比如我们使用PC端的命令行进行安装文件：

>adb shell

>pm install apk文件

其实上面的两行命令和

>adb install apk文件

的效果一样

但是如果想在手机上执行安装文件只能执行命令：pm install apk文件

(可以把pm命令的作用想成是脱离shell层来执行命令的)

不多说了，下面就来看一下代码吧：

```java
/*
* m命令可以通过adb在shell中执行，同样，我们可以通过代码来执行
*/
public static String execCommand(String ...command)  {
    Process process=null;
    InputStream errIs=null;
    InputStream inIs=null;
    String result="";

    try {
        process=new ProcessBuilder().command(command).start();
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        int read = -1;
        errIs=process.getErrorStream();
        while((read=errIs.read())!=-1){
            baos.write(read);
        }
        inIs=process.getInputStream();
        while((read=inIs.read())!=-1){
            baos.write(read);
        }
        result=new String(baos.toByteArray());
```

```
        if(inIs!=null)
            inIs.close();
        if(errIs!=null)
            errIs.close();
        process.destroy();
    } catch (IOException e) {
        result = e.getMessage();
    }
    return result;
}
```

调用这个方法：

```
[java]
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final String path = Environment.getExternalStorageDirectory() + File.separator + "baidu"+File.s

        Button btn1 = (Button) findViewById(R.id.btn1);
        btn1.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                //安装apk，filePath为apk文件路径，如/mnt/sdcard/ApiDemos.apk
                String result = execCommand("pm","install","-f",path);
                Toast.makeText(MainActivity.this, "安装结果:"+result, Toast.LENGTH_LONG).show();
            }});

        Button btn2 = (Button) findViewById(R.id.btn2);
        btn2.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                //卸载apk，packageName为包名，如com.example.android.apis
                String result = execCommand("pm","uninstall", "com.qihoo360.mobilesafe");
                Toast.makeText(MainActivity.this, "卸载结果:"+result, Toast.LENGTH_LONG).show();
            }});

    }
```

在AndroidManifest.xml中的配置：

```
[html]
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.xieyuan.mhfilemanager"
    android:versionCode="1"
    android:versionName="1.0"
    android:installLocation="internalOnly"
    android:sharedUserId="android.uid.system" >
```

也有：

```
[html]
android:sharedUserId="android.uid.system"
```

这个在前面已经说过了，所以我们现在做的还是将其进行系统签名，具体方法就不多说了，同上

下面是运行的结果图：

Demo工程下载地址：http://download.csdn.net/detail/jiangwei0910410003/7584429

下载之后的工程导入之后需要进行的几步操作和前面的是一模一样的，这里就不做解释了~~

但是我们从上面的方法看到有一个弊端，还是需要系统签名，这个对于一般app来说是个限制，现在机型很多，系统的签
以很难做到这种方式能够适配到所有的机型。

那么还有另外的一种方法：

2、不需要签名文件，但是需要root权限

这种方式很简单和上面的没有太大的区别，就是将上面的命令放到su下面去执行就可以了。

从这里我们可以看到，现在很多需要系统签名的或者是命令不能执行的，都会想到su环境中去执行以下。

这里不多解释了，直接上代码：

```java
public static boolean RootCommand(String command){
        Process process = null;
        DataOutputStream os = null;
        try{
            process = Runtime.getRuntime().exec("su");
            os = new DataOutputStream(process.getOutputStream());
            os.writeBytes(command + "\n");
            os.writeBytes("exit\n");
            os.flush();
            process.waitFor();
        } catch (Exception e){
            Log.d("*** DEBUG ***", "ROOT REE" + e.getMessage());
            return false;
        } finally{
            try{
                if (os != null){
                    os.close();
                }
                process.destroy();
            } catch (Exception e){
            }
        }
```

```
        Log.d("*** DEBUG ***", "Root SUC ");
        return true;
    }
```

然后我们调用这个方法来执行命令即可：

```
[java]
new Thread(){
    @Override
    public void run(){
        RootCommand("pm uninstall com.tencent.mm");
    }
}.start();
```

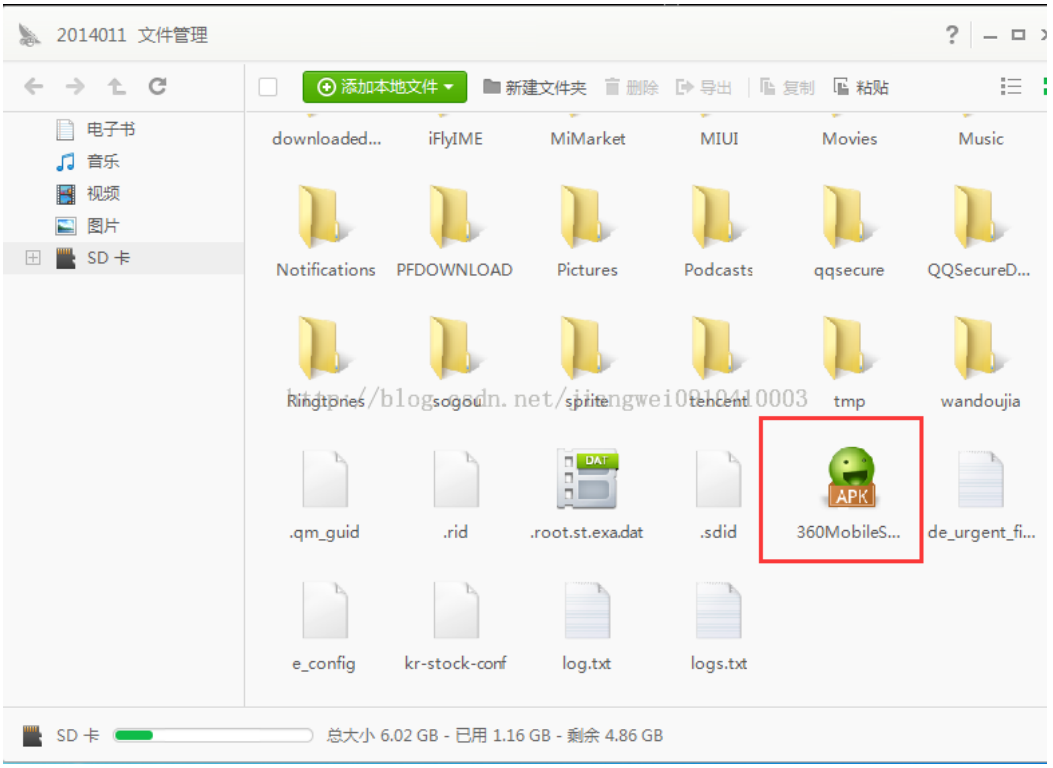
这里就可以执行了，这种方式可以适配所有的机型，但是需要root权限，这个就比较麻烦了，但是现在很多安全应用大

三、拷贝/删除apk文件实现安装和卸载

这个方式其实很简单，就是直接拷贝和删除文件即可。我们知道Android中安装的apk都是放在/data/app/目录下面的，所以我apk放到这个目录下即可。

注：这种方式是可以接收到安装和卸载的系统广播

首先我们在设备的SD卡中存放一个待安装的apk：360MobileSafe.apk



这里的目录可以随便选择的，我这里选择了SD卡的根目录

这里为了看到明显的效果，我们在写一个Demo程序，来接收安装的广播

后台监听广播的Service：

```
[java]
package com.example.installpackage;

import android.app.Service;
import android.content.Intent;
import                tent.IntentFilter;
import                IBinder;

public                MyService extends Service {

    @Override
    public void onCreate() {
        super.onCreate();
        IntentFilter filter = new IntentFilter(Intent.ACTION_PACKAGE_ADDED);
```

```
        filter.addAction(Intent.ACTION_PACKAGE_REMOVED);
        filter.addAction(Intent.ACTION_PACKAGE_CHANGED);
        filter.addDataScheme("package");
        filter.setPriority(Integer.MAX_VALUE);
        registerReceiver(new InstallReceiver(), filter);
    }

    @Override
    public IBinder onBind(Intent paramIntent) {
        return null;
    }

}
```

用动态方式进行注册

广播广播接收器：

```
[java]
package com.example.installpackage;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;

public final class InstallReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(final Context context, Intent intent) {
        Log.i("TTT","action:"+intent.getAction());
    }
}
```

开启服务：

```
[java]
package com.example.installpackage;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Intent i = new Intent(this, MyService.class);
        startService(i);
    }

}
```

还要记得在AndroidManifest.xml中进行注册服务：

```
[html]
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.installpackage"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.installpackage.MainActivity"
```

```
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service android:name=".MyService"></service>

    </application>

</manifest>
```

我们在PC端可以使用adb push XXX.apk /data/app/ 实现拷贝，但是在设备上这个命令不能执行。因为Android是底层是L

贝命令：

1、安装操作

1)、拷贝命令cp

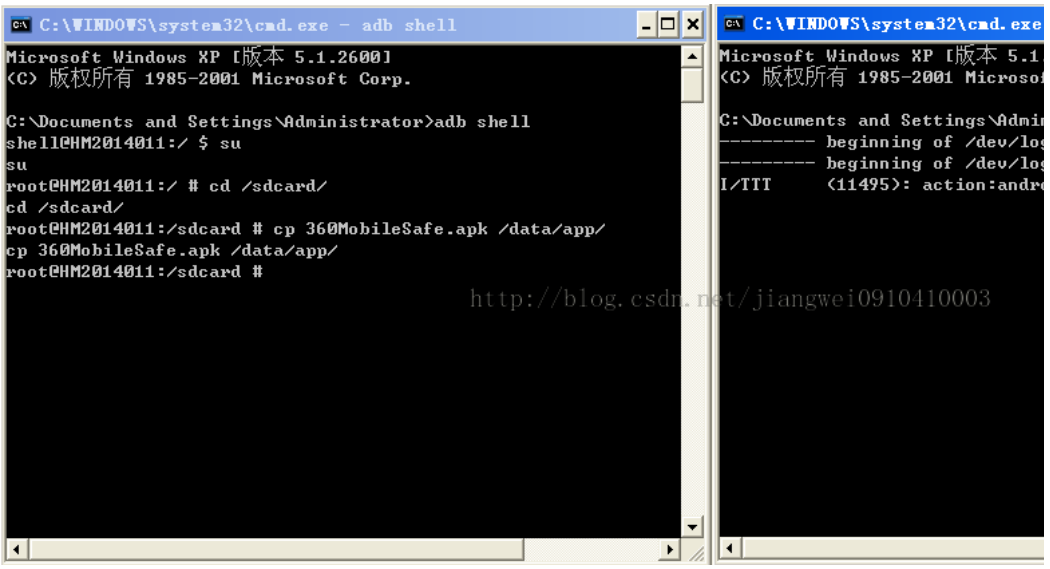
=>adb shell

=>su

=>cd /sdcard/

=>cp 360MobileSafe.apk /data/app

如图效果：



在这个过程中，我们通过打印log信息可以看到，是有点耗时的。

当然除了这种方式还有另外的一种方式：

2)、使用cat命令实现

我们知道cat命令有输出重定向的功能

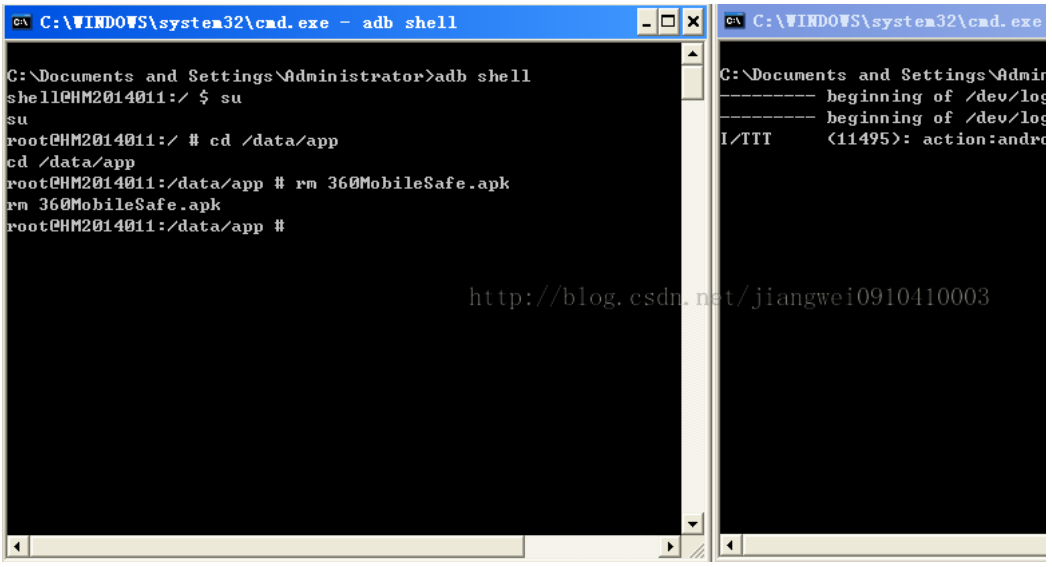
=>adb shell

=>su

=>cd /sdcard/

=>cat 360MobileSafe.apk > /data/app/com.qihoo.360.apk

运行结果：



2、卸载操作

卸载也很简单，我需要删除/data/app目录下指定的apk文件，但是我们知道，一个app安装好之后，不仅在这个目录中有数据，还

的：

/data/davlik-cache/ :存放释放的dex文件

/data/data/com.qihoo.360.mobilesafe/ :存放app的数据

/data/system/packages.list :这个文件是记录安装app的信息的

/data/system/packages.xml :这个文件是记录安装app的详细信息(权限，uid等)

命令：

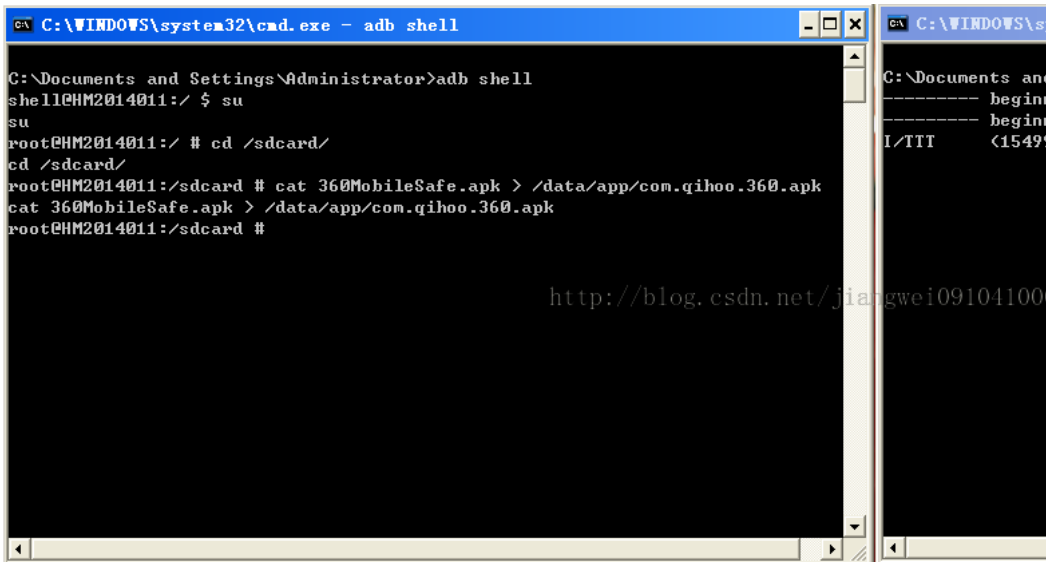
=>adb shell

=>su

=>cd /data/app/

=>rm 360MobileSafe.apk

运行结果：



总结

上面就介绍了实现静态安装和卸载的功能，前面两种方法不需要root权限的，但是需要签名文件，这个对于不同机型，有不同的

文件，(在第二种方法中的两种实现不同的，不需要签名文件，但是需要root权限)这个是件很头疼的事，第三种是不需要签名文

合，但是需要root权限的，现在很多安全软件在实现静

默安装和卸载的时候也都是需要root权限才能进行操作的，而且每个机型上都是可以的。所以他们应该采用的是第二种方式中的

方式。

注：前两种方式是不可以接受系统的安装和卸载广播的，第三种是可以接受系统的安装和卸载广播的。

待解决的问题

1. 上面我们可以看到我们使用的是4.4版本进行签名的，只能安装到系统是4.4版本的机子上，这个有待考证，因为我们开发的

可能只能装到指定的版本中，必须是所有的系统都能进行安装(比如豌豆荚就可以)，所以我就做了一个假设，是不是低版本的签

件进行系统签名之后的apk可以安装到高版本系统中，那么这样就好办了，我们可以下载Android2.2(因为现在市面上最低的版本

2.2了)源码，得到他的签名文件，或者从网上搜索到这个版本的签名文件(至今未找到，如果有找到的请分享，谢谢！！)

2. 还有一个问题就是上述的测试环境都是在手机或者模拟器都是root过了，有待考证的是没有root的手机会不会安装时出现问

上述的blog是纠结了我几个月的时间才解决的，所以我真心想说的一句话就是：做事千万不要放弃，办法总比问题多，一定要坚

~~，其实上面遇到的最大的问题就是在下载源码过程(没有技术可言，就是考验我们的耐心，同时网络也是一个重要的原因，这

于某些原因，我下载的是4.4版本的源码，共13G,这里就不公开下载地址了，如果真心需要请留言，我可以共享一下~~)，还有就

编译过程，我花费了一周的时间，因为在这个过程中可以说是千难万险，什么问题都有，我重新编译好几次~~

好吧，这里有现成的干货，咱拿过来直接用呗（当然如果没有源码的那就算了，那能实现的只是demo）。具体步骤：

从源码中拷贝要使用的aidl回调接口：IPackageInstallObserver.aidl、IPackageDeleteObserver.aidl当然完全可以

因为手头上没有4.4系统的机子，又不想刷机，所以就用模拟器了。在这个过程中安装和卸载可能要一段时间，所以要等待一会，

那么我们就介绍了使用隐藏的api来进行默认的安装和卸载app,

Demo工程下载地址：

http://download.csdn.net/detail/jiangwei0910410003/7584423

导入工程之后，需要做的几步：

在SD卡中拷贝一个UC.apk到baidu文件夹(需要新建的)

导出未进行签名的包

下载签名的工具：

http://download.csdn.net/detail/jiangwei0910410003/7584441

按照上面的签名步骤进行操作

然后找到一个系统是4.4版本的测试机或者模拟器进行安装包(因为我的签名文件的版本是4.4)

- 上一篇 J2EE学习篇之--JQuery技术详解
- 下一篇 数据结构和算法设计专题之---单链表的逆序

我的同类文章

Android（96）

- Android中的 Multiple dex files def… 2016-04-07 阅读 5226
- Android中如何修改编译的资源ID值(默… 2016-03-07 阅读 3798
- 如何使用Ant脚本编译出Jar和Apk包 2016-02-26 阅读 1010
- Android逆向之旅---解析编译之后的De… 2016-02-15 阅读 1307
- Android逆向之旅---解析编译之后的An… 2016-01-23 阅读 2086
- Android中自定义视图View之---进阶篇… 2016-03-
- Android关于Dex拆分(MultiDex)技术详解 2016-03-
- Android解析编译之后的所有文件(so,d… 2016-02-
- Android逆向之旅---解析编译之后的Re… 2016-02-
- Android签名机制之---签名验证过程详解 2016-01-

更多文章

查看评论

workspacewade 18楼 20
楼主，CmdDemo成功安装到了模拟机，点击安装提示failed[INSTALL_FAILED_INVALID_URI],怎么解决，谢谢

布丁西西 17楼 20
赵四。。。。。。。。。

colin_wj 16楼 20
hello 你的demo直接下载运行时没问题的，但是我在用你的工具该过签名过后，用adb安装依然不行 Failure [INSTALL_FAILED_SHARED_USER_INCCOMPATIBLE]

freemindh 15楼 20
我有不需要签名与root权限的方法

余烬岛游戏 Re: 20
回复freemindh：能介绍一下吗？是不是利用android的那个辅助功能 来帮助用户点击屏幕？

freemindh Re: 20
回复余烬岛游戏： 不是，是利用android 底层命令 am 调起应用的

尼古拉斯_赵四 Re: 20
回复余烬岛游戏：http://blog.csdn.net/jiangwei0910410003/article/details/48895153,应该使用辅助功能的，这篇文章我说到了~~

余烬岛游戏 Re: 20
回复尼古拉斯_赵四：多谢啦 看你的博客收货多多啊

qq_22703355 14楼 20
粗略看了一下，刚绝不错，谢谢楼主，加油！！！

LFHhua104 13楼 20
确实是很不错的技术干货，谢楼主慷慨分享！

安卓菜鸟yes 12楼 20

很有帮助，谢谢

chenzujie 11楼 20

看了下文章，意味着要实现静默安装或者卸载旧必须要么是用系统签名文件给apk签名要么就是得有root权限，这两个方法对于分发在app市场给用户下载的apk来说基本都不是完美的解决方案，现在我有一个需求，apk在运行一段时间后要把自己从手机里删除，看来是没有适配所有手机的办法了？

chenzujie 10楼 20

看了下文章，意味着要实现静默安装或者卸载旧必须要么是用系统签名文件给apk签名要么就是得有root权限，这两个方法对于分发在app市场给用户下载的apk来说基本都不是完美的解决方案，现在我有一个需求，apk在运行一段时间后要把自己从手机里删除，看来是没有适配所有手机的办法了？

u014333177 9楼 20

楼主我下载你的用命令安装demo 为什么我的安装不了呢！我有一个疑问，你那个l路径后面success 是在哪里获取到的？

hyy044101331 8楼 20

楼主写得不错，但有一些瑕疵：
1. adb shell 通常是在pc的shell中执行，是通过cs模式和手机通讯的，而pm是直接在手机的shell中执行的，故程序里面应该使用pm

尼古拉斯_赵四 Re: 20

写的再细，没你看的详细哈。。。文章篇幅太长了，真心懒得改了~~

liuqiuyue3166 7楼 20

可否给个扣扣号 这边很着急 拜托拜托

尼古拉斯_赵四 Re: 20

自己看一下代码，打印一下log看一下问题。

liuqiuyue3166 6楼 20

我用了你的代码，不知道为什么卸载失败，好奇怪 直接把你的项目下载 手机运行 卸载失败 求解

Living_in_Java 5楼 20

弱弱的问一句，系统签名文件都是通用的吗？模拟器上的和真机上的一样不？我研究的方法和楼主一样，过程也一样艰辛，在模拟器上也测过了，但是觉得手机上的系统签名可能是每个厂商都不公开的

尼古拉斯_赵四 Re: 20

是的，还是不能解决这样的问题，如果你要是解决了这样的问题，记得告诉我，非常感谢，如果我要是解决了，我也会及时告诉你的，相互学习吗？

Living_in_Java Re: 20

回复尼古拉斯_赵四：我开发的软件在定制的机器上跑，所以厂商能给提供系统签名文件。所以这个问题对我来说不是问题。但是不能推广到其他机器。

Android极致工程师 4楼 20

这个必须赞一下
看了网上很多文章 基本没有demo 也没有能运行的
楼主的 demo 确实能完美运行 测试机 小米2 版本4.4.4
如果安装不上 有的4.4 手机或许是因为手机本身没有root权限
写代码遇到 错误提醒 可以clean 一下就好了 谢谢楼主提醒

gqdy365 3楼 20

1、如果有root权限，91市场是可以静默安装卸载的，所以我想是可以通用的。估计用的是低版本的签名；
2、上述install安装和直接remove data/app目录下的apk的区别楼主知道不，按理说调用系统install接口是最安全，最直接的方法；

尼古拉斯_赵四 Re: 20

remove /data/apk这种方式太暴力了，而且你删除之后，如果你的应用在桌面有快捷方式的话，不会删除的，但是用户点击是没有反应的，所以这样一来对于用户体验特不好~~

xue081801 2楼 20

蝴蝶眨几次眼
才学会飞行

尼古拉斯_赵四 Re: 20

哥，你这个评论是我见过最有含义的，求解释。哈哈~~

xue081801 Re: 20

回复尼古拉斯_赵四：没事，技术交流的话请加QQ：850950269

gujianhunmeng 1楼 20

非常好的文章，要是能把标题上拼错的Android改过来，文章就完美了。现在很难搜到你的文章

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VPN
IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apache
HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails
Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App Spring
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angula
Redis Scala Django Bootstrap