## 302.java.lang.ClassNotFoundException

【出错堆栈】

http://bugly.qq.com/share/3ceWJ5

【出错代码】

public class MainActivity extends Activity {  
 MBroadCastReceiver receiver;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 FileUtil.CopyAssertApkToFile(MainActivity.this, "app-release.apk", "app-release.apk");  
  
 }  
  
 public void onClick(View v) {  
 System.out.println("bug start");  
  
 ArrayList list = new ArrayList();  
 list.add(getApkObject());  
  
 Intent it = new Intent();  
 it.setAction("com.cgy.action\_aa");  
 it.putExtra("se", list);  
 sendBroadcast(it);  
 System.out.println("sendBroadcast");  
 }  
  
 private Object getApkObject() {  
 File file = new File(Environment.getExternalStorageDirectory()  
 .toString() + "/cgytest" + File.separator + "app-release.apk");  
 final File optimizedDexOutputPath = getDir("temp", Context.MODE\_PRIVATE);  
 DexClassLoader classLoader = new DexClassLoader(file.getAbsolutePath(),  
 optimizedDexOutputPath.getAbsolutePath(), null,  
 getClassLoader());  
 Object obj = null;  
 try {  
 Class iclass = classLoader.loadClass("com.cgy.buglyproject.OtherSer");  
 obj = iclass.newInstance();  
  
 Field[] fields = iclass.getDeclaredFields();  
  
 for (Field field : fields) {  
 // 对于每个属性，获取属性名  
 String varName = field.getName();  
 boolean access = field.isAccessible();  
 if (!access) field.setAccessible(true);  
  
 //从obj中获取field变量  
 Object o = field.get(obj);  
 System.out.println("变量： " + varName + " = " + o);  
  
 if (!access) field.setAccessible(false);  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return obj;  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 receiver = new MBroadCastReceiver();  
 registerReceiver(receiver, new IntentFilter("com.cgy.action\_aa"));  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 unregisterReceiver(receiver);  
 }  
  
  
 class MBroadCastReceiver extends BroadcastReceiver {  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 System.out.println( "action:" + intent.getAction());  
 intent.getIntExtra("se",3);  
 }  
 }

【出错原因】

查看出错堆栈：

dalvik.system.BaseDexClassLoader.findClass java.lang.ClassLoader.loadClass

java.lang.ClassLoader.loadClass

java.lang.Class.classForName

java.lang.Class.forName

android.os.Parcel[SPV].resolveClass java.io.ObjectInputStream.readNewClassDesc java.io.ObjectInputStream.readClassDesc java.io.ObjectInputStream.readNewObject java.io.ObjectInputStream.readNonPrimitiveContent java.io.ObjectInputStream.readObject java.io.ObjectInputStream.readObject android.os.Parcel.readSerializable

android.os.Parcel.readValue

android.os.Parcel.readArrayMapInternal android.os.BaseBundle.unparcel

android.os.BaseBundle.getInt

android.content.Intent.getIntExtra ... android.app.LoadedApk$ReceiverDispatcher$Args.run android.os.Handler.handleCallback android.os.Handler.dispatchMessage

android.os.Looper.loop

android.app.ActivityThread.main

java.lang.reflect.Method.invoke

java.lang.reflect.Method.invoke com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run com.android.internal.os.ZygoteInit.main

先从下往上分析，由

android.content.Intent.getIntExtra

android.app.LoadedApk$ReceiverDispatcher$Args.run

这两句实践可知是在BroadcastReceiver中的getIntExtra（）方法中产生的错误，再结合错误ClassNotFound我们猜想是因为BroadcastReceiver接收到数据解析时ClassLoader找不到数据中的类造成的。

定位到源码ObjectInputStream.readNewClassDesc中

private ObjectStreamClass readNewClassDesc(boolean unshared)  
 throws ClassNotFoundException, IOException {  
 // So read...() methods can be used by  
 // subclasses during readClassDescriptor()  
 primitiveData = input;  
 int oldHandle = descriptorHandle;  
 descriptorHandle = nextHandle();  
 ObjectStreamClass newClassDesc = readClassDescriptor();  
 registerObjectRead(newClassDesc, descriptorHandle, unshared);  
 descriptorHandle = oldHandle;  
 primitiveData = emptyStream;  
  
 // We need to map classDesc to class.  
 try {  
 newClassDesc.setClass(resolveClass(newClassDesc));

进入resolveClass方法

protected Class<?> resolveClass(ObjectStreamClass osClass)  
 throws IOException, ClassNotFoundException {  
 // fastpath: obtain cached value  
 Class<?> cls = osClass.forClass();  
 if (cls == null) {  
 // slowpath: resolve the class  
 String className = osClass.getName();  
  
 // if it is primitive class, for example, long.class  
 cls = PRIMITIVE\_CLASSES.get(className);  
  
 if (cls == null) {  
 // not primitive class  
 // Use the first non-null ClassLoader on the stack. If null, use  
 // the system class loader  
 cls = Class.forName(className, false, callerClassLoader);  
 }  
 }  
 return cls;  
}

我们可以看到Class.forName方法的参数callerClassLoader并不是固定的，这个异常就是由于使用的classloader无法找到指定的类造成的，即数据类使用的classloader和解析时使用的classloader不是同一个。

我们在demo中验证这个bug,思路是使用一个额外的apk，将之保存到本地，然后使用DexClassLoader来加载其中的类，添加到ArrayList中，然后通过intent将ArrayList发送到BroadcastReceiver，在intent.getIntExtra()中看到抛出了异常，与崩溃堆栈相吻合。

【解决方案】

因为错误是由sendBroadCast中的Intent包含的数据类解析使用的classloader和BroadCastReceiver收到数据解析时使用的classLoader不一致导致，所以基本思路就是统一使用的classloader，可以在BroadCastReceiver中intent.get前添加一句

Intent.setExtrasClassLoader(loader)（loader为数据类使用的classloader）;这样就不会出现该错误了。如果无法获得数据类使用的classloader，可以使用try{}catch(){}捕获异常。