方法文档

一、核心思想

现有的GUI测试工具大多都是从主Activity开始进行GUI测试，现有工具Fax通过静态分析获取每个在Manifest文件中声明的Activity所需要的启动上下文，通过给每个Activity发送intent信息来直接启动Activity，而不需要经过复杂的路径才能启动某个Activity。但Fax在分析过程中，只考虑了Activity之间会启动Activity，忽略了其他三大组件Service、Broadcast Receiver和Content Provider发送intent信息启动Activity的情况，同时现在具体的安卓应用程序中，Fragment的使用也十分常见，Fax也没有考虑到这些Fragment的情况，此外，由于得到的intent信息都是静态分析得来的，给具体的某个属性赋值一般使用模糊的值，这降低了能够启动Activity的成功率。

在这里，我使用更先进的ICC检测工具ICCBot，它考虑到了其他三大组件和Fragment发送intent信息启动Activity的情况，可以得到更加精确的intent信息，然后我通过soot插桩、Monkey的GUI测试得到待测试apk的Activity转换之间的动态ICC信息，将这些ICC信息整合进通过ICCBot得到的静态ICC信息中，可以提高直接启动Activity成功率。

对于成功启动的Activity，进行多轮GUI测试。

二、具体工作

1. 预处理工作
    1.1. 对待测试的apk首先将其Manifest里声明的Activity均设置为暴露状态，以便dummyAPK给其发送intent信息
    1.2. 对待测试的apk使用InsDal进行插桩，以便后续收集测试的覆盖率数据。
2. 利用先进的ICC检测工具获取ICC信息

使用ICCBot得到待测试apk的信息后，通过三个途径对待测试apk的Activity构建静态的启动上下文：

1.apk的Manifest文件

apk的Manifest文件包含了所有可以直接启动的Activity，包括对应的intent信息，如下图这个com.integreight.onesheld.plugin.action.ActionActivity，可以提取出该Activity的一个基本属性action，构建出一条简单的intent信息用于启动该Activity。

```
<activity android:configChanges="orientation" android:exported="true" android:icon="@drawable/ic_launcher"
android:label="@string/plugin_name" android:name="com.integreight.onesheeld.plugin.action.ActionActivity"
android:screenOrientation="portrait" android:theme="@android:style/Theme.Light" android:uiOptions="
splitActionBarWhenNarrow" android:windowSoftInputMode="adjustResize">
    <intent-filter>
        <action android:name="com.twofortyfouram.locale.intent.action.EDIT_SETTING"/>
    </intent-filter>
</activity>
```

2.ICCBot的结果文件CTG.xml

CTG.xml包含了apk中的组件之间的转换信息，更准确地说，包含了一个Activity可能接收到的来自其他组件或者Fragment的intent信息，可以通过这一点来构建一条intent来启动一个Activity。

```
<source name="com.integreight.onesheeld.push.PushMessagesReceiver" type="Service">
  <destination ICCType="implicit" name="android.intent.action.VIEW, " edgeType="NonAct2Class" method="&lt;
  com.integreight.onesheeld.push.PushMessagesReceiver: void showNotificationWithUrl(
  android.content.Context,java.lang.String,java.lang.String,java.lang.String)&gt;" instructionId="20" unit="
  virtualinvoke r4.&lt;android.support.v4.app.NotificationCompat$Builder:
  android.support.v4.app.NotificationCompat$Builder setContentIntent(android.app.PendingIntent)&gt;($r7)" action="
  android.intent.action.VIEW" extras="" flags="FLAG_ACTIVITY_NEW_TASK "/>
</source>
```

3.ICCBot的结果文件componentInfo.xml

componentInfo.xml文件则包含了一个Activity可能接收到的所有的属性信息（虽然一次Activity转换不一定会触发所有的属性获取路径），从这里构建一个intent信息来启动Activity寄希望于尽可能地多触发执行路径。

4.对于得到的静态intent信息，根据其对应的数据类型，给它赋一个模糊值，比如int类型的属性字段，就给他赋一个0、Integer.MAXVALUE或者Integer.MINVALUE等。通过这些步骤可以得到一条启动Activity的静态intent信息，举例如：com.fsck.k9.activity.setup.AccountSetupIncoming null;;null;;null;;null;;boolean–>makeDefault–>false,String–>account–>abcde，给这个Activity的intent中包含以下信息：

```
基本属性:
    action=null
    category=null
    data=null
    type=null
额外属性:
 boolean makeDefault = false
 String account = abcde
```

3. 获取Activity的动态ICC信息

3.1. soot插桩

获取动态的icc信息首先需要在apk的GUI测试期间能够得到Activity的执行路径上触发的属性获取信息。因此我使用soot对Activity的所有属性获取语句下插入Log语句。

3.2. 前期GUI测试

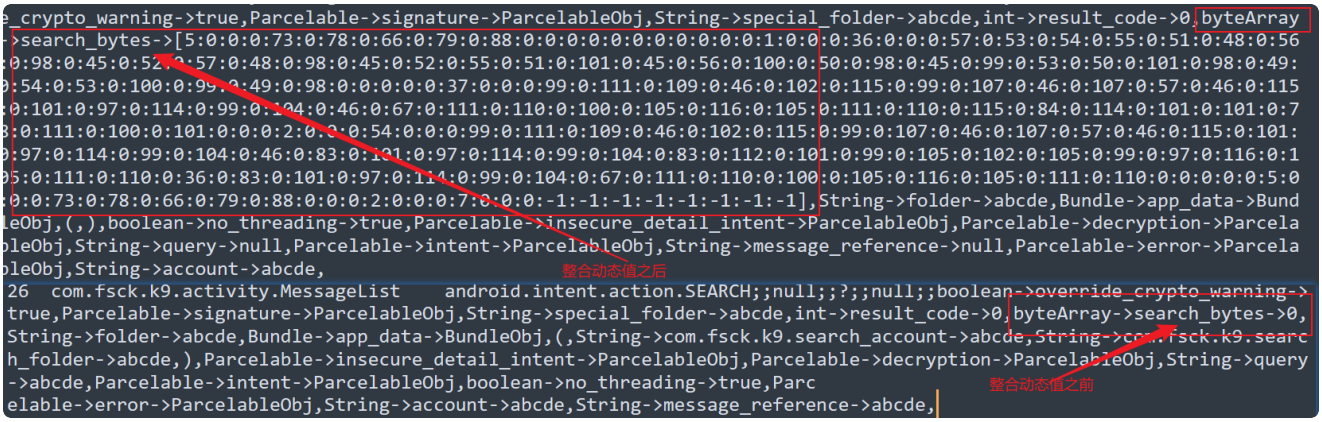
前期的GUI测试期间，可以通过命令`adb logcat  -s "q-tag" -s AndroidRuntime:E`得到Activity的执行路径上获取属性的具体值，并存入<appName>.logcat文件中。

3.3. 整合动态ICC信息

得到了Activity的动态ICC信息后，将Activity所获取的具体的值整合进通过ICCBot得到的静态ICC信息中。

举例：

应用K9Mail通过ICCBot得到这样一条intent信息，其中有一个extra额外属性，是Byte数组，没有动态的icc数据之前，我们是给了它一个模糊值，0，有了动态icc数据后，直接将动态的值结合进去。



4. 构建dummyAPK

由于给Activity发送的intent信息不仅仅包含基本的属性值，还可能会包含Bundle对象，因此需要创建一个小的apk，将一个intent信息存入apk中的一个Activity，专门用于给待测试的apk发送intent信息。

具体例子：这是一个构建好的一个用来给待测试apk发送intent信息的Activity。在这个Activity中，构建了一个intent，内部包含信息如代码所见。然后在这个Activity里，设置了一个按钮，点击按钮即向待测试的apk发送intent信息。

```java
package qiu.com.fsck.k9;

import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import java.util.List;
import android.content.ComponentName;
import android.os.Parcelable;
import java.util.ArrayList;
import android.net.Uri;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class Activity_25 extends Activity
{
    public void launch(){
        Intent intent = new Intent();
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        ComponentName cn=new ComponentName("com.fsck.k9","com.fsck.k9.activity.MessageList");
        intent.setComponent(cn);
        intent.setAction("android.intent.action.SEARCH");
        intent.setData(Uri.parse("mSheme:mAuthority/?"));
        Bundle app_data = new Bundle();
        intent.putExtra("account", "abcde");
        Parcelable error = new MyParcelable();
        intent.putExtra("error", error);
        Parcelable intent1 = new MyParcelable();
        intent.putExtra("intent", intent);
        Parcelable decryption = new MyParcelable();
        intent.putExtra("decryption", decryption);
        Parcelable insecure_detail_intent = new MyParcelable();
        intent.putExtra("insecure_detail_intent", insecure_detail_intent);
        intent.putExtra("no_threading", true);
        intent.putExtra("app_data", app_data);
        intent.putExtra("folder", "abcde");
        byte[] search_bytes = new byte[]{5,0,0,0,73,0,78,0,66,0,79,0,88,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,36,0,0,0,57,0,53,0,54,0,55,0,51,0,48,0,56,0,98,0,45,0,52,0,57,0,48,0,98,0,45,0,52,0,55,0,51,0,101,0,45,0,56,0,100,0,50,0,98,0,45,0,99,0,53,0,50,0,101,0,98,0,49,0,54,0,53,0,100,0,99,0,49,0,98,0,0,0,0,0,37,0,0,0,99,0,111,0,109,0,46,0,102,0,115,0,99,0,107,0,46,0,107,0,57,0,46,0,115,0,101,0,97,0,114,0,99,0,104,0,46,0,67,0,111,0,110,0,100,0,105,0,116,0,105,0,111,0,110,0,115,0,84,0,114,0,101,0,101,0,78,0,111,0,100,0,101,0,0,0,2,0,0,0,54,0,0,0,99,0,111,0,109,0,46,0,102,0,115,0,99,0,107,0,46,0,107,0,57,0,46,0,115,0,101,0,97,0,114,0,99,0,104,0,46,0,83,0,101,0,97,0,114,0,99,0,104,0,83,0,112,0,101,0,99,0,105,0,102,0,105,0,99,0,97,0,116,0,105
```

```
,0,111,0,110,0,36,0,83,0,101,0,97,0,114,0,99,0,104,0,67,0,111,0,110,0,100,
0,105,0,116,0,105,0,111,0,110,0,0,0,0,0,5,0,0,0,73,0,78,0,66,0,79,0,88,0,0
,0,2,0,0,0,7,0,0,0,-1,-1,-1,-1,-1,-1,-1,-1};
        intent.putExtra("search_bytes", search_bytes);
        intent.putExtra("result_code", 0);
        intent.putExtra("special_folder", "abcde");
        Parcelable signature = new MyParcelable();
        intent.putExtra("signature", signature);
        intent.putExtra("override_crypto_warning", true);
        startActivity(intent);
        //android.intent.action.SEARCH;;null;;mSheme:mAuthority/?;;null;;b
oolean-&gt;override_crypto_warning-&gt;true,Parcelable-&gt;signature-&gt;ParcelableOb
j,String-&gt;special_folder-&gt;abcde,int-&gt;result_code-&gt;0,byteArray-&gt;search_byte
s-&gt;[5:0:0:0:73:0:78:0:66:0:79:0:88:0:0:0:0:0:0:0:0:0:0:1:0:0:0:36:0:0:0:
57:0:53:0:54:0:55:0:51:0:48:0:56:0:98:0:45:0:52:0:57:0:48:0:98:0:45:0:52:
0:55:0:51:0:101:0:45:0:56:0:100:0:50:0:98:0:45:0:99:0:53:0:50:0:101:0:98:
0:49:0:54:0:53:0:100:0:99:0:49:0:98:0:0:0:0:0:37:0:0:0:99:0:111:0:109:0:4
6:0:102:0:115:0:99:0:107:0:46:0:107:0:57:0:46:0:115:0:101:0:97:0:114:0:99:
0:104:0:46:0:67:0:111:0:110:0:100:0:105:0:116:0:105:0:111:0:110:0:115:0:8
4:0:114:0:101:0:101:0:78:0:111:0:100:0:101:0:0:0:2:0:0:0:54:0:0:0:99:0:11
1:0:109:0:46:0:102:0:115:0:99:0:107:0:46:0:107:0:57:0:46:0:115:0:101:0:97:
0:114:0:99:0:104:0:46:0:83:0:101:0:97:0:114:0:99:0:104:0:83:0:112:0:101:0:
99:0:105:0:102:0:105:0:99:0:97:0:116:0:105:0:111:0:110:0:36:0:83:0:101:0:9
7:0:114:0:99:0:104:0:67:0:111:0:110:0:100:0:105:0:116:0:105:0:111:0:110:0:
0:0:0:0:5:0:0:0:73:0:78:0:66:0:79:0:88:0:0:0:2:0:0:0:7:0:0:0:-1:-1:-1:-1:-
1:-1:-1:-1],String-&gt;folder-&gt;abcde,Bundle-&gt;app_data-&gt;BundleObj,(,),boolean-
&gt;no_threading-&gt;true,Parcelable-&gt;insecure_detail_intent-&gt;ParcelableObj,Parc
elable-&gt;decryption-&gt;ParcelableObj,String-&gt;query-&gt;null,Parcelable-&gt;intent-&gt;
ParcelableObj,String-&gt;message_reference-&gt;null,Parcelable-&gt;error-&gt;Parcelabl
eObj,String-&gt;account-&gt;abcde,
    }
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        launch();
        Button button1=(Button)findViewById(R.id.button1);
        button1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                launch();
        }});
    }
}
```

5. GUI测试（粗略版）

GUI测试部分，我打算参照Fax作者的方法进行多轮探索测试。

大概的过程：

1.从先前启动的结果中，过滤掉启动失败的Activity，以剩余的成功启动的Activity为初始集合。

2.对这个初始集合的Activity，根据Activity和启动失败的Activity之间的距离、Activity的方法总数和一个常数权重值得到每个activity的权重，根据每个activity的权重给它分配相应的不同的Monkey脚本GUI事件数。（详细版本和Fax一样，看后续有没有可改进的）

3.每轮GUI测试都根据这个权重计算和分配事件数。