- **组件无需暴露**

  在 AndroidManifest.xml 为组件加上属性 android:exported=”false”

- **组件需要暴露**

  分两种情况：

  1. 仅暴露给自己或同一公司的其他应用

  修复方案 I：

  在被调用组件所属 app 的 Android-Manifest.xml 中自定义一个权限：

```
<permission android:name=”myapp.permission.CALL_MYSERVICE”
android:label=”@string/permlab_myService”
android:description=”@string/permdesc_myService”
android:permissionGroup=” android.pernission-group.SYSTEM_TOOLS”
android:protectionLevel=”signature”
/>
```

  并为被调用组件添加该权限限制：

```
<service android :name=”.MyService”
android :permission=”myApp.permission.CALL_MYSERVICE” />
```

  修复方案 II：

  在组件的实现代码中使用 Context.checkCallingPermission()检查调用者是否拥有这个权限。

  2. 需暴露给第三方应用

  仔细审查组件的代码，保证组件功能可控，避免出现能力泄露或权限重委派（获得额外权限的攻击）等风险。

- **Intent 无需隐式调用**

  修改隐式调用为显式调用：

```
Intent intent = new Intent();

// 下面是三种显式调用
intent.setClass(getApplicationContext(), theNewActivity.class);
intent.setClassName("com.secmobi.App", "com.secmobi.App.theNewActivity");
intent.setComponent(new Component("com.secmobi.App", ".theNewActivity"));
```

- **Intent 需隐式调用**

  用 sendBroadcast(Intent, String)替代 sendBroadcast(Intent)，且确保权限进过严格校验，如签名校验；或者采用本地广播实现。

  其他情况需仔细审查被调用组件的代码，保证组件功能可控，避免出现能力泄露或权限重委派（获得额外权限的攻击）等风险。