AndroidManifest.xml文件安全探索

simeon / 2017-08-23 02:54:00 / 浏览数 3972 新手 入门资料 顶(0) 踩(0)

---

最近在做一些apk的安全检测，对AndroidManifest.xml文件进行了研究和探讨，介绍AndroidManifest.xml文件的作用和架构，并研究了AndroidManifest.xml配置文件存

## 0X00 AndroidManifest.xml文件作用

AndroidManifest.xml文件的作用非常重要，应该说是缺一不可。在android官方介绍文档中（https://developer.android.com/guide/topics/manifest/manifest-intro.h
System，这些信息就存在AndroidManifest中。AndroidManifest.xml 存放在 app/src/main/
目录下。在反编译APK文件后，其文件是以乱码格式存在，需要进行转换才能正常查看。

## 0X01主要功能

命名应用程序Java包，软件包名称作为应用程序的唯一标识符

描述了应用程序的组件，其中包括构成应用程序的活动，服务，广播接收器和内容提供者；它还命名实现每个组件并发布其功能的类，例如Intent可以处理的消息。这些

决定哪些processes主持application

宣告这个App有哪些权限，它声明应用程序必须拥有的权限才能访问API的受保护部分并与其他应用程序交互。它还声明其他人为了与应用程序的组件交互而需要的权限

5. 它列出了Instrumentation在应用程序运行时提供概要分析和其他信息的类。这些声明仅在应用程序正在开发中才会存在，并在应用程序发布之前被删除。
6. 它声明了应用程序需要的最低级别的Android API。
7. 它列出了应用程序必须链接的库。

## 0X02 Manifest架构

允许的元素，蓝字是预设常见的元素，其中的<manifest>与<application>是必要且只能出现一次。每个元素有各自的属性，属性数量不一定，每个属性有其默认值，可视

1.预设的AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

■    package="com.example.bmi" ■■■■

■    android:versionCode="1"   ■■■■■■■■■■

■    android:versionName="1.0" >  ■■■■■■■■■■

■    <uses-sdk

■        android:minSdkVersion="8" ■■■■SDK■■

■        android:targetSdkVersion="21" /> ■■■■■■■■■■■■■■■minSdkVersion

■    <application

■        android:allowBackup="true" ■■■■■■■

■        android:icon="@drawable/ic_launcher" App Icon

■        android:label="@string/app_name" App■■

■        android:theme="@style/AppTheme" > App■■

■        <activity activity, service, receiver, provider■■■application■4■■■■■■

■            android:name=".MainActivity" activity■■■■■■manifest package■■■■

■            android:label="@string/app_name" > APP■■■■■■■■■■■■■■■

■            <intent-filter> activity■■■■■
```

```
■                <action android:name="android.intent.action.MAIN" /> .MAIN■■activity■APP■■■
■                <category android:name="android.intent.category.LAUNCHER" /> ■■■■■■■■■
■            </intent-filter>
■        </activity>
■    </application>
</manifest>
```

2.标准的AndroidManifest.xml文件样例。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
■    <!-- ■■■■ -->
■    <uses-permission />
■    <permission />
■    <permission-tree />
■    <permission-group />
■    <instrumentation />
■    <uses-sdk />
■    <uses-configuration />
■    <uses-feature />
■    <supports-screens />
■    <compatible-screens />
■    <supports-gl-texture />
■    <!-- ■■■■ -->
■    <application>
■        <!-- Activity ■■ -->
■        <activity>
■            <intent-filter>
■                <action />
■                <category />
■                <data />
■            </intent-filter>
■            <meta-data />
■        </activity>
■        <activity-alias>
■            <intent-filter> . . . </intent-filter>
■            <meta-data />
```

```
■        </activity-alias>

■           <!-- Service ■■ -->

■        <service>

■            <intent-filter> . . . </intent-filter>

■            <meta-data/>

■        </service>

■        <!-- Receiver ■■ -->

■        <receiver>

■            <intent-filter> . . . </intent-filter>

■            <meta-data />

■        </receiver>

■        <!-- Provider ■■ -->

■        <provider>

■            <grant-uri-permission />

■            <meta-data />

■        </provider>

■        <!-- ■■■■■■ -->

■        <uses-library />

■    </application>

 </manifest>
```

## 0x03 文件约定及语法

□
从上面的代码中，我们可以看出Android配置文件采用XML作为描述语言，每个XML标签都不同的含义，大部分的配置参数都放在标签的属性中，下面我们便按照以上配置文

1.元素（Elements）

在所有的元素中只有<manifest>和<application>是必需的，它们各自必须存在，且只能出现一次。如果一个元素包含有其他子元素，必须通过子元素的属性来设置其值。

1、一个<activity-alias> 元素必须遵循 <activity>它是一个别名。</activity></activity-alias>

2、<application> 元素必须是里面的最后一个元素 <manifest>
的元素。换句话说</manifest></application>结束标签必须在</manifest>结束标签之前立即出现。

2.属性

□
正常来讲，所有的属性都是可选的，但是有些属性是必须设置的。以便元素可以实现其目的，除了根元素<manifest>的属性之外，所有其他元素属性的名字都是以android

定义类名：所有的元素名都对应其在SDK中的类名，如果你自己定义类名，必须包含类的数据包名，如果类与application处于同一数据包中，可以直接简写为“.”；

3.声明类名

许多元素对应于Java对象，包括应用程序本身（<application>元素）的元素及其主要组件：活动（<activity>），服务<service>），广播接收器（<receiver>）和内容提

如果你定义一个子类，如同你总是会为组件类（Activity，Service，
BroadcastReceiver和ContentProvider）子类是通过name属性来声明，该名称必须包括完整的包装名称。例如，一个Service子类可能被声明如下：

```
<manifest . . . >
```

```
■    <application . . . >
■        <service android:name="com.example.project.SecretService" . . . >
■            . . .
■        </service>
■        . . .
■    </application>
</manifest>
```

4.多个值

如果某个元素有超过一个数值，这个元素必须通过重复的方式来说明其某个属性具有多个数值项，且不能将多个数值项一次性说明在一个属性中；例如一个intent-filter可以保护多个action：

```
<intent-filter . . . >
■    <action android:name="android.intent.action.EDIT" />
■    <action android:name="android.intent.action.INSERT" />
■    <action android:name="android.intent.action.DELETE" />
■    . . .
</intent-filter>
```

5.资源值

 某些属性具有可显示给用户的值，例如一个活动的标签和图标。这些属性的值应该从资源或主题进行本地化和设置。资源值以下列格式表示：

@[package:]type/name

如果资源与应用程序在同一个软件包中，则可以省略软件包名称。该类型是一种资源，例如字符串或可画的对象，名称是特定资源的标识名称。例如：

<activity android:icon="@drawable/smallPic" . . . >

主题的值使用类似地表达，但以初始值"？"代替"@"：

?[package:]type/name

注意：资源或主题包的值必须是"android"或应用程序包的名称。

6.字符串值

在属性值为字符串的地方，必须使用双反斜杠（\）来转义字符，例如\n换行符或\uxxxx 表示Unicode字符。

7.意图过滤器

应用程序的核心组件，如活动，服务和广播接收器由意图（Intent）激活。意图是Intent描述所需动作的一组信息（对象），包括要执行的数据，应该执行该操作的组件的类
Intent对象。

组件通过意图过滤器通知他们可以响应的意图类型。由于Android系统必须了解组件在启动组件之前可以处理的意图，因此在清单中将intent过滤器指定为
<intent-filter>
元素。组件可以具有任意数量的过滤器，每个过滤器描述不同的功能。显式命名目标组件的意图激活该组件，因此过滤器不起作用。没有通过名称指定目标的意图可以仅在组

8.图标和标签

许多元素都有图标和标签属性，可以向用户显示一个小图标和文本。一些还有一个更长的描述属性，也可以在屏幕上显示。例如，该 <permission>
元素具有所有这三个属性，以便当询问用户是否授予已请求它的应用程序的权限时，一个图标代表权限，许可的名称以及它所需要的描述都会呈现给用户。</permission>

在每种情况下，在包含元素中设置的图标和标签将成为所有容器的子元素的默认值 icon和label设置。因此，
<application>元素中设置的图标和标签是每个应用程序组件的默认图标和标签。类似地，为组件（如<activity>
元素）设置的图标和标签是每个组件<intent-filter> 元素的默认设置 。如果一个 <application>
元素设置了一个标签，但是一个活动和它的意图过滤器没有，应用程序标签将被视为活动和意图过滤器的标签。</application></intent-filter></activity></application

为意图过滤器设置的图标和标签表示当组件呈现给用户并满足由过滤器发布的功能时的组件。例如，带有android.intent.action.MAIN和
android.intent.category.LAUNCHER设置的过滤器将活动通告为启动应用程序的活动。也就是说，应该在应用程序启动器中显示。在过滤器中设置的图标和标签显示在启动

9.权限

权限是限制的代码的一部分，或者在设备上的数据的访问的限制。限制是为了保护可能被误用以扭曲或损坏用户体验的关键数据和代码。

每个权限都由唯一标签标识。标签通常表示限制的动作。以下是Android定义的一些权限：

```
android.permission.CALL_EMERGENCY_NUMBERS

android.permission.READ_OWNER_DATA

android.permission.SET_WALLPAPER

android.permission.DEVICE_POWER
```

功能只能通过一个权限来保护。如果应用程序需要访问受权限保护的功能，则它必须声明它需要使用<uses-permission> 清单中的元素的权限
。当应用程序安装在设备上时，安装程序将通过检查签署应用程序证书的机构以及在某些情况下询问用户来确定是否授予所请求的权限。如果许可被授予，应用程序就可以使

应用程序也可以通过权限保护自己的组件。它可以使用由Android定义的任何权限，如android.Manifest.permission由其他应用程序列出
或声明的。它也可以自己定义。<permission> 元素声明了新的权限 。例如，活动可以如下保护：</permission>

```
<manifest . . . >

■    <permission android:name="com.example.project.DEBIT_ACCT" . . . />

■    <uses-permission android:name="com.example.project.DEBIT_ACCT" />

■    . . .

■    <application . . .>

■        <activity android:name="com.example.project.FreneticActivity"

■                android:permission="com.example.project.DEBIT_ACCT"

■                . . . >

■            . . .

■        </activity>

■    </application>

</manifest>
```

请注意，在这个例子中，DEBIT_ACCT权限不仅仅是使用<permission> 元素来声明的 ，所以它也使用了 <uses-permission>
元素。为了启动受保护的活动，您必须要求使用该应用程序的其他组件，即使应用程序本身也施加了保护。</uses-permission></permission>

## 0x04权限属性值意义

ACCESS_CHECKIN_PROPERTIES：允许对checkin数据库中的表"properties"进行读/写访问，以更改上传的值。

ACCESS_COARSE_LOCATION：允许应用访问大概位置。

ACCESS_FINE_LOCATION：允许应用访问精确位置。

ACCESS_LOCATION_EXTRA_COMMANDS：允许应用程序访问额外的位置提供程序命令。

ACCESS_NETWORK_STATE：允许应用程序访问有关网络的信息。

ACCESS_NOTIFICATION_POLICY：希望访问通知政策的应用程序的标记权限。

ACCESS_WIFI_STATE：允许应用程序访问有关Wi-Fi网络的信息。

ACCOUNT_MANAGER：允许应用程序调用AccountAuthenticator。

ADD_VOICEMAIL：允许应用程序将语音邮件添加到系统中。

ANSWER_PHONE_CALLS：允许应用接听来电。

BATTERY_STATS：允许应用程序收集电池统计信息

BIND_ACCESSIBILITY_SERVICE：必须由a要求AccessibilityService，以确保只有系统可以绑定到它。

BIND_APPWIDGET：允许应用程序告诉AppWidget服务哪个应用程序可以访问AppWidget的数据。

BIND_AUTOFILL_SERVICE：必须由a要求AutofillService，以确保只有系统可以绑定到它。

BIND_CARRIER_MESSAGING_SERVICE：这个常量是在API层面弃用23. BIND_CARRIER_SERVICES代替

BIND_CARRIER_SERVICES：允许绑定到运营商应用程序中的服务的系统进程将具有此权限。

BIND_CHOOSER_TARGET_SERVICE：必须由a要求ChooserTargetService，以确保只有系统可以绑定到它。

BIND_CONDITION_PROVIDER_SERVICE：必须由a要求ConditionProviderService，以确保只有系统可以绑定到它。

BIND_DEVICE_ADMIN：必须由设备管理接收器要求，以确保只有系统可以与其进行交互。

BIND_DREAM_SERVICE：必须由a要求DreamService，以确保只有系统可以绑定到它。

BIND_INCALL_SERVICE：必须由a要求InCallService，以确保只有系统可以绑定到它。

BIND_INPUT_METHOD：必须由a要求InputMethodService，以确保只有系统可以绑定到它。

BIND_MIDI_DEVICE_SERVICE：必须由a要求MidiDeviceService，以确保只有系统可以绑定到它。

BIND_NFC_SERVICE：必须要求HostApduService 或OffHostApduService确保只有系统可以绑定到它。

BIND_NOTIFICATION_LISTENER_SERVICE：必须由a要求NotificationListenerService，以确保只有系统可以绑定到它。

BIND_PRINT_SERVICE：必须由a要求PrintService，以确保只有系统可以绑定到它。

BIND_QUICK_SETTINGS_TILE：允许应用程序绑定到第三方快速设置图块。

BIND_REMOTEVIEWS：必须由a要求RemoteViewsService，以确保只有系统可以绑定到它。

BIND_SCREENING_SERVICE：必须由a要求CallScreeningService，以确保只有系统可以绑定到它。

BIND_TELECOM_CONNECTION_SERVICE：必须由a要求ConnectionService，以确保只有系统可以绑定到它。

BIND_TEXT_SERVICE：必须由TextService要求

BIND_TV_INPUT：必须通过a TvInputService 来确保只有系统可以绑定它。

BIND_VISUAL_VOICEMAIL_SERVICE：链接必须要求，VisualVoicemailService以确保只有系统可以绑定到它。

BIND_VOICE_INTERACTION：必须由a要求VoiceInteractionService，以确保只有系统可以绑定到它。

BIND_VPN_SERVICE：必须由a要求VpnService，以确保只有系统可以绑定到它。

BIND_VR_LISTENER_SERVICE：必须由a要求VrListenerService，以确保只有系统可以绑定到它。

BIND_WALLPAPER：必须由a要求WallpaperService，以确保只有系统可以绑定到它。

BLUETOOTH：允许应用程序连接到配对的蓝牙设备。

BLUETOOTH_ADMIN：允许应用程序发现和配对蓝牙设备。

BLUETOOTH_PRIVILEGED：允许应用程序在没有用户交互的情况下配对蓝牙设备，并允许或禁止电话簿访问或消息访问。

BODY_SENSORS：允许应用程序访问用户用来衡量身体内发生的情况的传感器的数据，例如心率。

BROADCAST_PACKAGE_REMOVED：允许应用程序广播应用程序包已被删除的通知。

BROADCAST_SMS：允许应用程序广播短信收据通知。

BROADCAST_STICKY：允许应用程序广播粘性意图。

BROADCAST_WAP_PUSH：允许应用程序广播WAP PUSH收据通知。

CALL_PHONE：允许应用程序发起电话而不通过拨号器用户界面供用户确认通话。

CALL_PRIVILEGED：允许应用程序呼叫任何电话号码，包括紧急号码，而无需通过Dialer用户界面，用户确认呼叫正在被放置。

CAMERA：需要能够访问相机设备。

CAPTURE_AUDIO_OUTPUT：允许应用程序捕获音频输出。

CAPTURE_SECURE_VIDEO_OUTPUT：允许应用程序捕获安全视频输出。

CAPTURE_VIDEO_OUTPUT：允许应用程序捕获视频输出。

CHANGE_COMPONENT_ENABLED_STATE：允许应用程序更改应用程序组件（不是自己的）是否启用。

CHANGE_CONFIGURATION：允许应用程序修改当前配置，如区域设置。

CHANGE_NETWORK_STATE：允许应用程序更改网络连接状态。

CHANGE_WIFI_MULTICAST_STATE：允许应用程序进入Wi-Fi组播模式。

CHANGE_WIFI_STATE：允许应用程序更改Wi-Fi连接状态。

CLEAR_APP_CACHE：允许应用程序清除设备上所有已安装应用程序的缓存。

CONTROL_LOCATION_UPDATES：允许启用/禁用收音机的位置更新通知。

DELETE_CACHE_FILES：允许应用程序删除缓存文件。

DELETE_PACKAGES：允许应用程序删除软件包。

DIAGNOSTIC：允许应用程序RW到诊断资源。

DISABLE_KEYGUARD：允许应用程序禁用键盘保护程序，如果它不安全。

DUMP：允许应用程序从系统服务检索状态转储信息。

EXPAND_STATUS_BAR：允许应用程序展开或折叠状态栏。

FACTORY_TEST：作为制造商测试应用程序运行，以root用户身份运行。

GET_ACCOUNTS：允许访问帐户服务中的帐户列表。

GET_ACCOUNTS_PRIVILEGED：允许访问帐户服务中的帐户列表。

GET_PACKAGE_SIZE：允许应用程序找出任何包使用的空间。

GET_TASKS：这个常数在API级别21中已被弃用。不再强制执行。

GLOBAL_SEARCH：该权限可用于内容提供商，以允许全局搜索系统访问其数据。

INSTALL_LOCATION_PROVIDER：允许应用程序将位置提供程序安装到位置管理器中。

INSTALL_PACKAGES：允许应用程序安装软件包。

INSTALL_SHORTCUT：允许应用程序在Launcher中安装快捷方式。

INSTANT_APP_FOREGROUND_SERVICE：允许即时应用创建前台服务。

INTERNET：允许应用程序打开网络套接字。

KILL_BACKGROUND_PROCESSES：允许应用程序调用 killBackgroundProcesses(String)。

LOCATION_HARDWARE：允许应用程序在硬件中使用位置功能，例如geofencing api。

MANAGE_DOCUMENTS：允许应用程序管理对文档的访问，通常是文档选择器的一部分。

MANAGE_OWN_CALLS：允许通过自我管理的ConnectionServiceAPI 管理自己的呼叫的呼叫应用程序 。

MASTER_CLEAR：不适用于第三方应用程序。

MEDIA_CONTENT_CONTROL：允许应用程序知道正在播放哪些内容并控制其播放。

MODIFY_AUDIO_SETTINGS：允许应用程序修改全局音频设置。

MODIFY_PHONE_STATE：允许修改电话状态 - 开机，mmi等

MOUNT_FORMAT_FILESYSTEMS：允许将文件系统格式化为可移动存储。

MOUNT_UNMOUNT_FILESYSTEMS：允许安装和卸载文件系统以进行可移动存储。

NFC：允许应用程序通过NFC执行I / O操作。

PACKAGE_USAGE_STATS：允许应用程序收集组件使用统计信息，声明权限意味着使用API，设备的用户可以通过“设置”应用程序授予权限。

PERSISTENT_ACTIVITY：此常数在API级别9中已被弃用。此功能将在以后删除; 请不要使用。允许应用程序使其活动持续。

PROCESS_OUTGOING_CALLS：允许应用程序在呼出期间查看正在拨打的电话号码，并选择将呼叫重定向到其他号码或完全中止呼叫。

READ_CALENDAR：允许应用程序读取用户的日历数据。

READ_CALL_LOG：允许应用程序读取用户的通话记录。

READ_CONTACTS：允许应用程序读取用户的联系人数据。

READ_EXTERNAL_STORAGE：允许应用程序从外部存储器读取。

READ_FRAME_BUFFER：允许应用程序进行屏幕截图，更一般地，可以访问帧缓冲区数据。

READ_INPUT_STATE：此常数在API级别16中已被弃用。使用此权限的API已被删除。

READ_LOGS：允许应用程序读取低级别的系统日志文件。

READ_PHONE_NUMBERS：允许读取设备的电话号码。

READ_PHONE_STATE：允许只读访问电话状态，包括设备的电话号码，当前的蜂窝网络信息，任何正在进行的呼叫的状态以及PhoneAccount在设备上注册的任何列表。

READ_SMS：允许应用程序读取短信。

READ_SYNC_SETTINGS：允许应用程序读取同步设置。

READ_SYNC_STATS：允许应用程序读取同步统计信息。

READ_VOICEMAIL：允许应用程序读取系统中的语音信箱。

REBOOT：需要重新启动设备。

RECEIVE_BOOT_COMPLETED：允许应用程序收到ACTION_BOOT_COMPLETED在系统完成启动后广播的应用程序 。

RECEIVE_MMS：允许应用程序监视传入的彩信。

RECEIVE_SMS：允许应用程序接收短信。

RECEIVE_WAP_PUSH：允许应用程序接收WAP推送消息。

RECORD_AUDIO：允许应用程序录制音频。

REORDER_TASKS：允许应用程序更改任务的Z顺序。

REQUEST_COMPANION_RUN_IN_BACKGROUND：允许随播应用在后台运行。REQUEST_COMPANION_USE_DATA_IN_BACKGROUND：允许随播应用在后台使用数

REQUEST_DELETE_PACKAGES：允许应用程序请求删除包。

REQUEST_IGNORE_BATTERY_OPTIMIZATIONS：许可申请必须持有才能使用 ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS。

REQUEST_INSTALL_PACKAGES：允许应用程序请求安装软件包。

RESTART_PACKAGES：此常数在API级别8中已弃用restartPackage(String) 。不再支持API。

SEND_RESPOND_VIA_MESSAGE：允许应用程序（电话）向其他应用程序发送请求，以处理来电期间的响应通过消息动作。

SEND_SMS：允许应用程序发送短信。

SET_ALARM：允许应用程序广播Intent为用户设置闹钟。

SET_ALWAYS_FINISH：允许应用程序控制是否在后台放置活动时立即完成。

SET_ANIMATION_SCALE：修改全局动画缩放因子。

SET_DEBUG_APP：配置应用程序进行调试。

SET_PREFERRED_APPLICATIONS：这个常数在API级别7中已被弃用。不再有用， addPackageToPreferred(String) 有关详细信息。

SET_PROCESS_LIMIT：允许应用程序设置可以运行的最大数量（不需要的）应用程序进程。

SET_TIME：允许应用程序设置系统时间。

SET_TIME_ZONE：允许应用程序设置系统时区。

SET_WALLPAPER：允许应用设置壁纸。

SET_WALLPAPER_HINTS：允许应用程序设置壁纸提示。SIGNAL_PERSISTENT_PROCESSES：允许应用程序请求将信号发送到所有持久进程。

STATUS_BAR：允许应用程序打开，关闭或禁用状态栏及其图标。

SYSTEM_ALERT_WINDOW：允许应用使用类型创建窗口 TYPE_APPLICATION_OVERLAY，显示在所有其他应用程序的顶部。

TRANSMIT_IR：允许使用设备的红外发射器（如果有的话）。

UNINSTALL_SHORTCUT：不再支持此权限。

UPDATE_DEVICE_STATS：允许应用程序更新设备统计信息。

USE_FINGERPRINT：允许应用使用指纹硬件。

USE_SIP：允许应用程序使用SIP服务。

VIBRATE：允许访问振动器。

WAKE_LOCK：允许使用PowerManager WakeLock来防止处理器进入睡眠状态或屏幕变暗。

WRITE_APN_SETTINGS：允许应用程序写入apn设置。

WRITE_CALENDAR：允许应用程序写入用户的日历数据。

WRITE_CALL_LOG：允许应用程序写入（而不是读取）用户的通话记录数据。

WRITE_CONTACTS：允许应用程序写入用户的联系人数据。

WRITE_EXTERNAL_STORAGE：允许应用程序写入外部存储。

WRITE_GSERVICES：允许应用修改Google服务地图。

WRITE_SECURE_SETTINGS：允许应用程序读取或写入安全系统设置。

WRITE_SETTINGS：允许应用程序读取或写入系统设置。

WRITE_SYNC_SETTINGS：允许应用程序写入同步设置。

WRITE_VOICEMAIL：允许应用程序修改和删除系统中现有的语音信箱。

## 0x05 apk文件获取AndroidManifest.xml文件

1.解压apk文件

首先需要下载apk文件，使用压缩软件直接解压缩即可，解压成功后会在apk目录中生存一个AndroidManifest.xml文件，如图1所示。使用记事本或者IE等打开该文件后，其

图1 AndroidManifest.xml文件

图2文件内容为乱码

2.使用androguard进行转码

androguard可以下载最新版本，也可以下载1.9版本。

https://github.com/androguard/androguard/archive/1.9.zip

将AndroidManifest.xml文件复制到androguard目录，我使用的是PentestBox-with-Metasploit-v2.2平台。到E:\Tools\测试平台\PentestBox-with-Metasploit-v2.2\bin

androaxml.py -i AndroidManifest.xml -o new.WoCloud.AndroidManifest.xml

即可解码内容。

## 0x06.apktool反编译apk

前面通过压缩文件直接解压会导致部分文件未经过编码，因此会出现乱码，经过编译的文件可以很好的进行查看，下面介绍使用apktool进行反编译apk程序，执行效果如下

1.下载apktool.jar

https://bitbucket.org/iBotPeaches/apktool/downloads/apktool_2.2.4.jar

2.将一下脚本保存为apktool.bat

```
@echo off
```

```
if "%PATH_BASE%" == "" set PATH_BASE=%PATH%

set PATH=%CD%;%PATH_BASE%;

java -jar -Duser.language=en "%~dp0\apktool.jar" %*
```

3.反编译程序

（1）直接用java进行反编译：java -jar apktool.jar d test.apk

（2）使用bat脚本进行编译：apktool -f d test.apk //覆盖已有的反编译程序及其目录

apktool d test.apk

注意：apktool.bat和apktool_2.2.4.jar在同一个目录，且下载的apktool_2.2.4.jar需要重命名为apktool.jar

## 0x07. AndroidManifest.xml 默认设置漏洞

1.配置文件中的默认设置allowBackup风险

（1）安全风险描述

Android API Level 8及其以上Android系统提供了为应用程序数据的备份和恢复功能，此功能的开关决定于该应用程序中 AndroidManifest.xml文件中的 allowBackup属性值，其属性值默认是True。当allowBackup标志为true时，用户即可通过adb backup和adb restore来进行对应用数据的备份和恢复，这可能会带来一定的安全风险。当设置该属性值为true，adb backup容许任何一个能够打开USB调试开关的人从Android手机中复制应用数据到外设，一旦应用数据被备份之后，所有应用数据都可被读取；同时adb restore容许用户指定一个恢复的数据来源（即备份的应用数据）来恢复应用程序数据的创建。因此，当一个应用数据被备份之后，用户即可在其他 Android 手机或模拟器上安装同一个应用，以及通过恢复该备份的应用数据到该设备上，在该设备上打开该应用即可恢复到被备份的应用程序的状态。

对于目前大多数手机来说，一旦存在该漏洞，容易导致个人通讯录、微信、QQ聊天信息、短信等敏感信息泄露；通过将备份程序在模拟手机上恢复后，可以直接进行店家扫

（2）影响范围

Android API 等级8（Android 2.2 - 2.2.3）以及以上系统，目前绝大部分系统都受影响。下面给出Android API等级对应按照系统以及名称对应的图标名称：

API等级1： Android 1.0

API等级2： Android 1.1 Petit Four 花式小蛋糕

API等级3： Android 1.5 Cupcake 纸杯蛋糕

API等级4： Android 1.6 Donut 甜甜圈

API等级5： Android 2.0 Éclair 松饼

API等级6： Android 2.0.1 Éclair 松饼

API等级7： Android 2.1 Éclair 松饼

API等级8： Android 2.2 - 2.2.3 Froyo 冻酸奶

API等级9： Android 2.3 - 2.3.2 Gingerbread 姜饼

API等级10：Android 2.3.3-2.3.7 Gingerbread 姜饼

API等级11：Android 3.0 Honeycomb 蜂巢

API等级12：Android 3.1 Honeycomb 蜂巢

API等级13：Android 3.2 Honeycomb 蜂巢

API等级14：Android 4.0 - 4.0.2 Ice Cream Sandwich 冰激凌三明治

API等级15：Android 4.0.3 - 4.0.4 Ice Cream Sandwich 冰激凌三明治

API等级16：Android 4.1 Jelly Bean 糖豆

API等级17：Android 4.2 Jelly Bean 糖豆

API等级18：Android 4.3 Jelly Bean 糖豆

API等级19：Android 4.4 KitKat 奇巧巧克力棒

API等级20：Android 4.4W KitKat with wearable extensions 奇巧巧克力棒

API等级21：Android 5.0-5.0.2 Lollipop 棒棒糖

（3）测试流程（以sina.weibo为例）

 测试环境：Windows 7，ADB 调试工具；物理接触目标手机1，连接手机1到 PC 端

手机1和手机2均未被 ROOT，开启 USB 调试；不用安装其它应用，不启动被测试的应用。连接安装开启USB调试手机1到PC端，在PC自动（也可以提前）安装好手机驱动后，启动命令行界面输入以下命令：

l adb devices

#显示已连接的设备列表，测试手机是否正常连接

l adb backup -nosystem -noshared -apk -f com.sina.weibo.abcom.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

l 点击手机1确认备份界面的“备份我的数据”

l 等待备份完成，至此微博客户端数据成功备份为 com.sina.weibo.ab 文件

l 断开手机1的连接

l 连接手机2 ，在命令行界面下输入以下命令：

l adb kill-server #关闭ADB

l adb devices #重新启动ADB，检测手机2是否成功连接

l adb restore com.sina.weibo.ab

l 点击手机2确认恢复界面的“恢复我的数据”

l 等待恢复完成

l 打开手机2中新安装的微博客户端，测试可正常登录手机1中帐号执行各种操作，且长期有效。

（4）安全防护

显示设置android:allowBackup=false，使用android:restoreAnyVersion的默认值。

（5）检测漏洞

使用apktool等工具反编译apk后，查看AndroidManifest.xml文件，查找allowBackup，如果其值为ture，则表示存在漏洞，如下图所示。

2.Debuggable默认设置风险

原理：android:debuggable属性用于指定应用程序是否能够被调试，如果设置其为true，那么其将能够被java调试工具（jdb）调试，信息和代码都将可能会被获取和修改。

防护：系统默认其为false，使用系统默认设置。


上一篇：极验验证码破解—超详细教程（三） 下一篇：聊一聊当年应急过的案例
1. 5 条回复


bigfacecat 2017-08-23 07:33:37

所以，你这文章有什么用？

0 回复Ta

---


simeon 2017-08-27 12:19:57

这个文件中包含了apk权限的参数。
可以将apk直接打包，然后还原。

0 回复Ta

---


simeon 2017-08-27 12:20:35

apk如果存在安全问题，可以越权，可以插入恶意代码，直接控制对方的手机。后续还有文章跟大家分享。

0 回复Ta

---


ih0cker 2017-09-04 02:23:56

分享是一种美德

0 回复Ta

---


hades 2017-09-04 03:10:49

回帖支持一下也是一种美德~

0 回复Ta

---

登录 后跟帖

先知社区

---

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板