Android 系统签名实现的三种方式

晓涵涵 于 2019-06-22 17:23:35 发布

在项目开发时，如果需要使应用具有系统权限，例如可以支持静默安装和卸载APK，此时就需要使用系统签名。
常用的系统签名方式包括在 ubuntu 环境下、手动签名和在AndroidStudio环境配置，三种方式中，实现最简单的是通过AndroidStudo方式，该方式的签名实现与正常的APK签名相同，唯一不同的就是签名文件是通过系统生成的。
注意，无论采用何种签名方式，如果想实现具有系统权限的应用，在APK生成时，都需要在AndroidManifest.xml中配置android:sharedUserId=“android.uid.system”，如下所示

```
<manifest  xmlns:android="http://schemas.android.com/apk/res/android"
            package="com.xxxx.xxxx"
           android:sharedUserId="android.uid.system">

</manifest>
```

1. ubuntu环境编译

该方式需要在安卓源码编译的环境下，在将APK打包至安卓系统升级包时，需要配置Android.mk文件，通过在配置LOCAL_CERTIFICATE 参数时，将其设置为PRESIGNED，如果需要改APK具有系统权限，使用系统签名，则需要设置为platform

```
LOCAL_CERTIFICATE := platform 或 shared 或 media

```

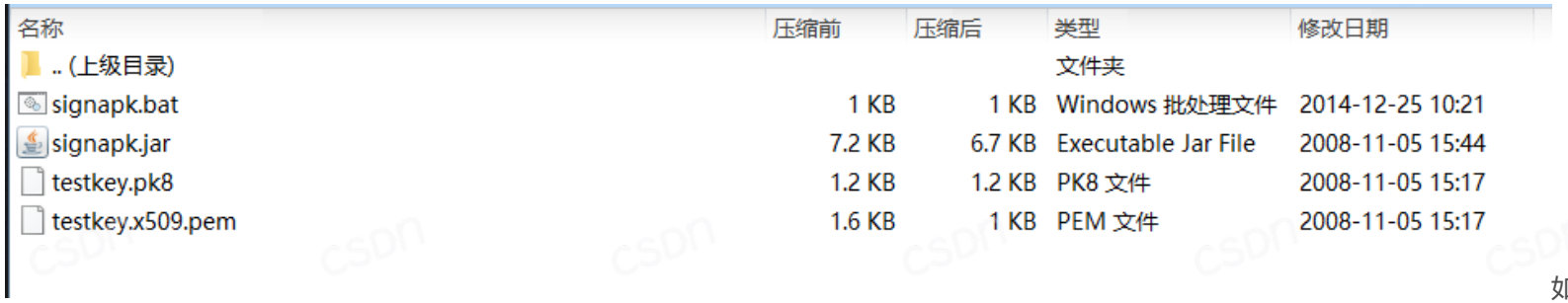
在设置好 Android.mk 文件后，在ubuntu环境下执行系统的编译流程。
如果之前在ubuntu环境下已经执行好了系统的编译流程，则可以直接cd到系统的package/apps目录下，进入到需要编译的APK文件目录下执行mm指令，或在其他目录执行mmm /package/apps/APK所在的文件夹名称。

2.手动签名

手动系统签名与Android的APK重签名不同，重签名是之前APK已经签名完成，在实际使用时，需要更改签名文件，例如一些特殊的算法处理对于系统的包名和签名都有绑定操作，如更换则无法使用。
手动系统签名是对于通过AndroidStudio的build生成的无签名的APK文件，进行手动系统签名操作。具体执行过程如下。

2.1 下载SignApk.jar

首先下载SignApk.jar包，如下图所示



名称	压缩前	压缩后	类型	修改日期
.. (上级目录)			文件夹	
signapk.bat	1 KB	1 KB	Windows 批处理文件	2014-12-25 10:21
signapk.jar	7.2 KB	6.7 KB	Executable Jar File	2008-11-05 15:44
testkey.pk8	1.2 KB	1.2 KB	PK8 文件	2008-11-05 15:17
testkey.x509.pem	1.6 KB	1 KB	PEM 文件	2008-11-05 15:17

如果本地有安卓系统的源码，可以直接在本地的build\tools\signapk目录下查找到该jar包。

2.2 查找security文件

拿到系统定制厂商提供的security文件，不同编译下生成的security文件内容不同，需针对该安卓系统下的编译生成的security文件，因此这也限定了该系统签名后的应用只能在该系统下运行，在其他系统下运行就不具有系统权限。
在security文件中找到media.pk8和platform.x509.pem两个文件。

2.3 执行系统签名操作

将2.1中的SignApk.jar和2.2中的media.pk8和platform.x509.pem文件一起复制到包含需要签名的APK文件夹中，然后执行如下语句

```
1 java -jar signapk.jar  platform.x509.pem platform.pk8  old.apk new.apk
```

生成的new.apk文件就是系统签名后APK文件。
该方式的操作，可参考安卓签名工具SignApk.jar使用教程，其已将签名的流程制作成Window下的.exe工具，只需一次配置，每次点击更换需要签名的文件即可。

3 AndroidStudio方式

通常对于APK的签名文件是通过AndroidStudio的New Key Store方式自定义实现，但是如果需要使用系统签名文件需要结合security文件中的media.pk8和platform.x509.pem两个文件，通过keytool-importkeypair实现，下载成功后，将media.pk8和platform.x509.pem两个文件放置在包含 keytool-importkeypair目前下，执行以下语句。

```
1 ./keytool-importkeypair -k ./platform.keystore -p android -pk8 platform.pk8 -cert platform.x509.pem -alias platform
```

需要注意的是：

- 该语句的执行是在ubuntu环境下执行的
- platform.keystore为系统签名文件
- android为签名密码
- platform为签名的别名(alias)

生成系统签名后，在AndroidStudio中配置Signing签名信息，配置成功后在modle的buid.gradle中可以查看如下配置信息。

```
signingConfigs {
    releaseConfig {
        keyAlias 'platform'
        keyPassword 'android'
        storeFile file('.........platform.keystore')//签名文件路径
        storePassword 'android'
    }
}
```

4.总结

从以上的分析中可以看出，无论三种那种方式的实现，都离不开系统源码中的security目录下的media.pk8和platform.x509.pem两个文件，该两个文件是保证应用具有系统签名的前提，如果使用其他系统的文件，则在该系统中，无法具有系统权限。
对比以上方法，分为具有ubuntu和没有两种环境下。

- 具有ubuntu的编译环境：则使用第一种比较简单，第三种生成的系统签名文件，可用于其他APK的签名使用，在AndroidStudio中配置后，方便调试使用，不必每次都执行命令行来生成签名后的APK文件。
- 不具有ubuntu的编译环境：该方式只能通过第二种方式实现，且每次调试使用时都需要替换APK生成新的系统签名后的APK，操作比较繁琐。当然在2.3中也提到，可借助其他同学制作的小工具方便签名文件的使用。

参考

安卓签名工具SignApk.jar使用教程
让Android Studio支持系统签名
keytool-importkeypair 使用