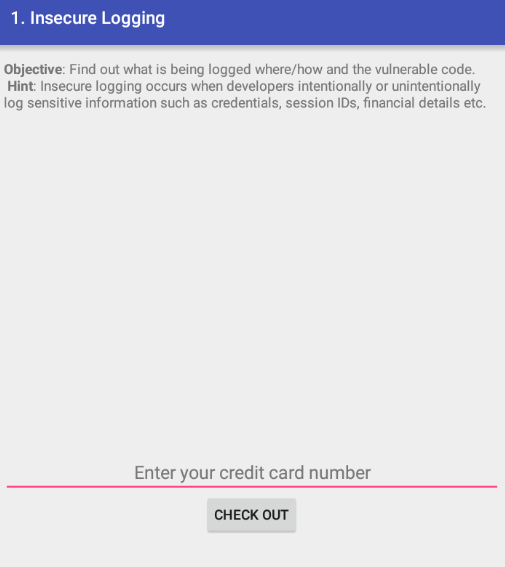
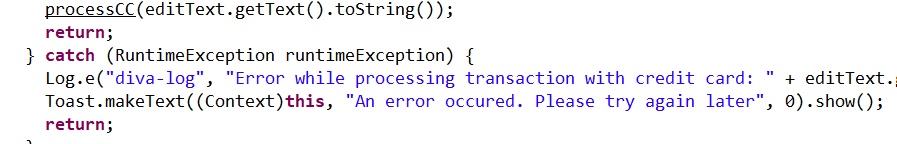
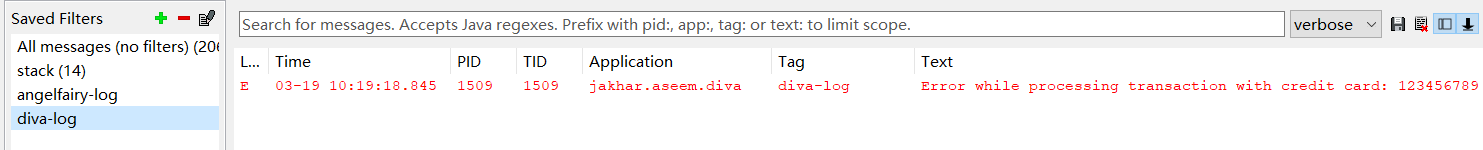
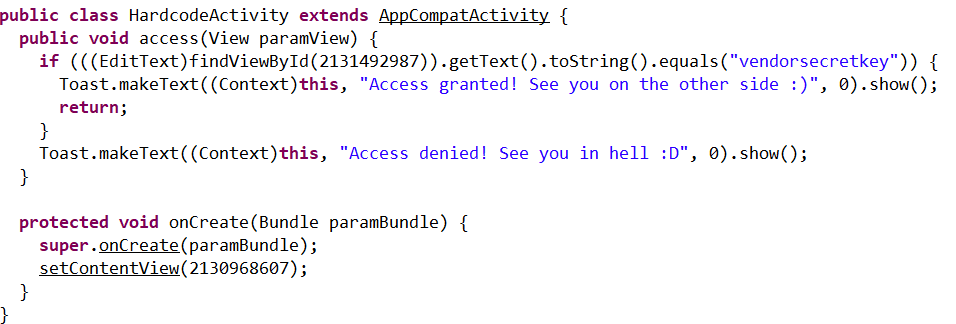
# 1 不安全的日志输出



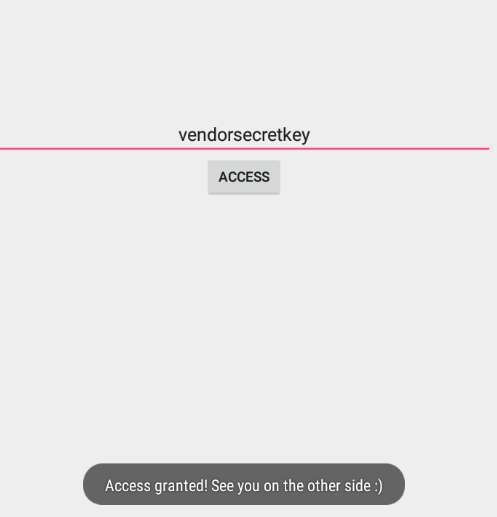




# 2硬编码1（class源文件）



攻击者只需要在app中输入秘钥vendorsecretkey就可以访问成功，如图：



# 3 不安全的存储1（shared\_prefs/xxx.xml）

不安全的数据存储也是App常见的安全问题之一，主要有三种方式：

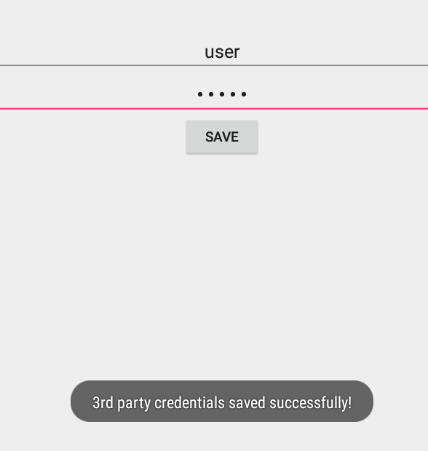
1，将敏感数据保存到配置文件中；

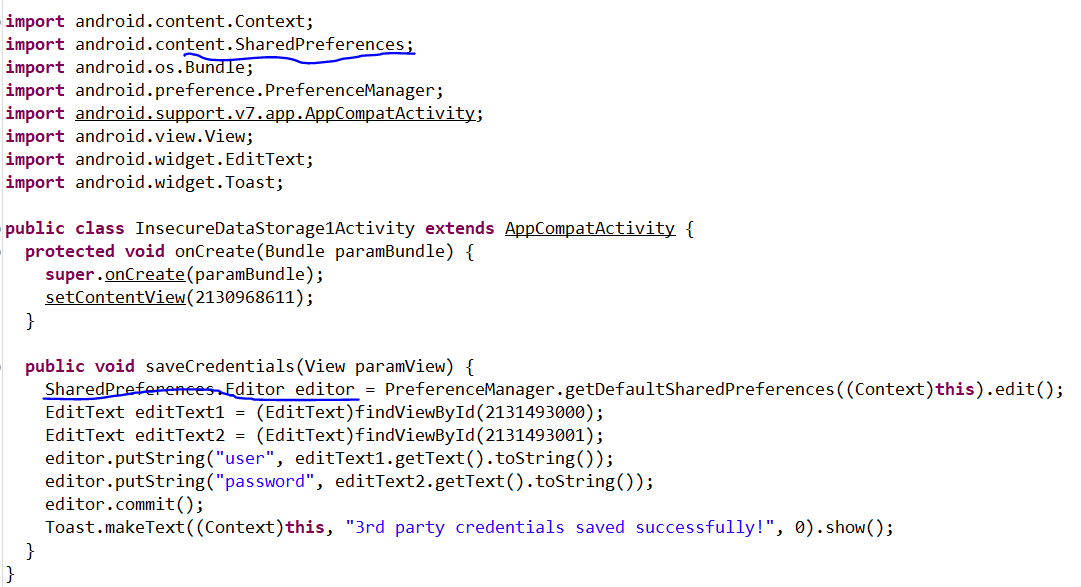
2，将敏感数据保存在本地的sqlite3数据库中；

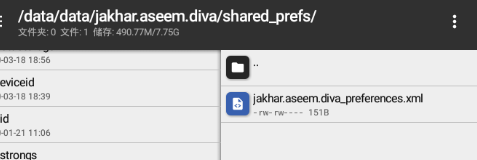
3，将敏感数据保存在临时文件或者sd卡中。

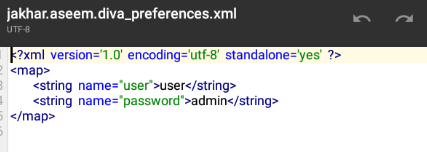
SharedPreferences类存储的数据会以.xml的形式存储在

/data/data/apppackagename/shared\_prefs目录下。









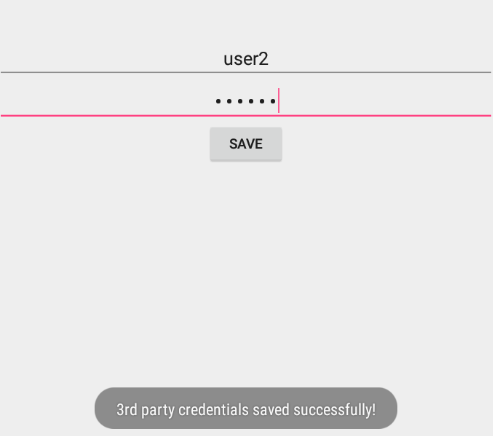
# 4 不安全的存储2（databases/xxx.db）

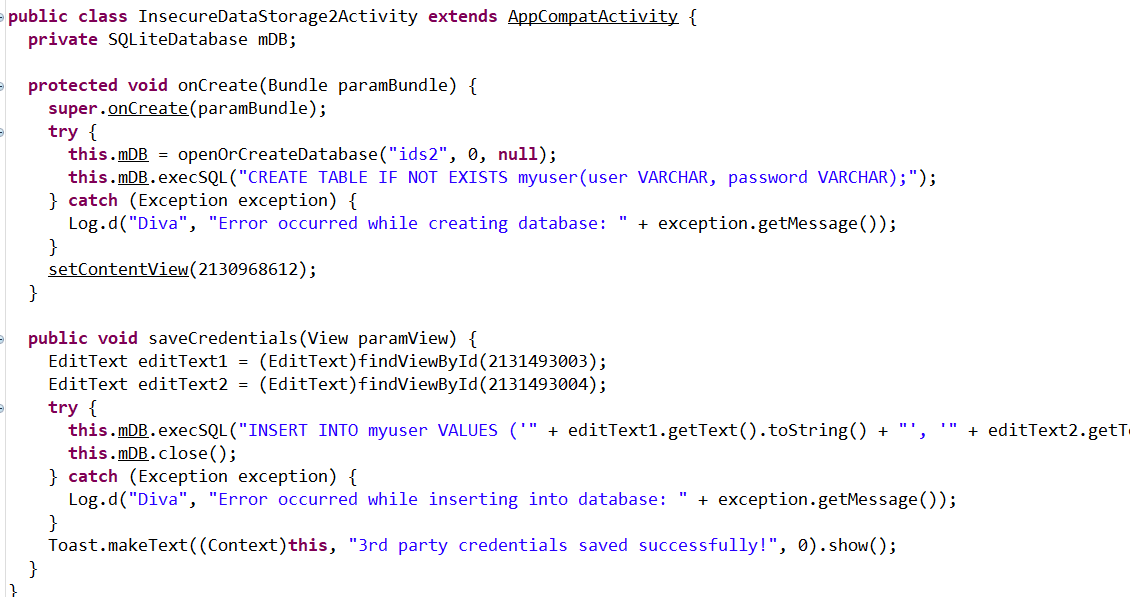
用户的敏感信息存储到本地的数据库中，一般app对应的数据库目录:

/data/data/apppackagename/databases

本例中是：

/data/data/jakhar.aseem.diva/databases

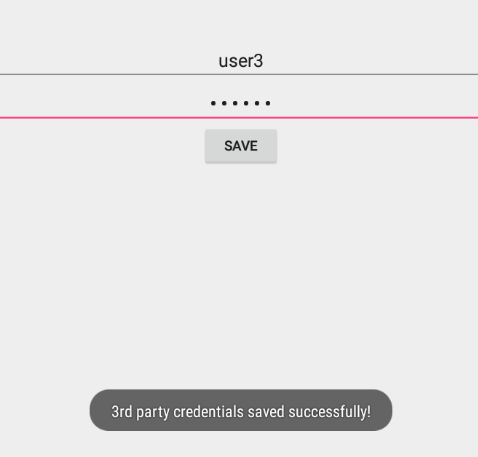






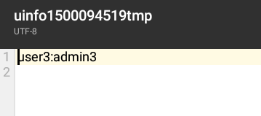


# 5 不安全的存储3（临时文件）

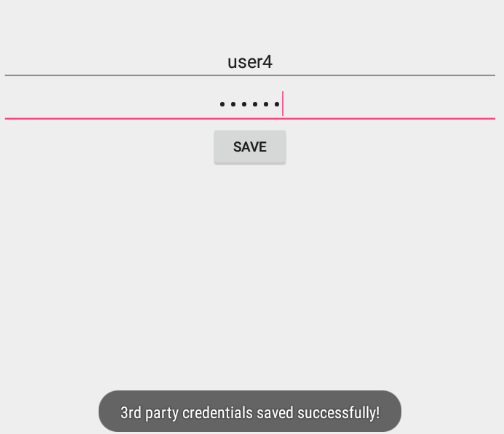




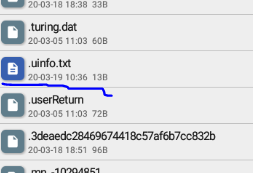


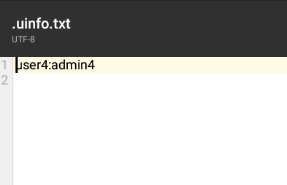


# 6 不安全的存储4（SD卡）





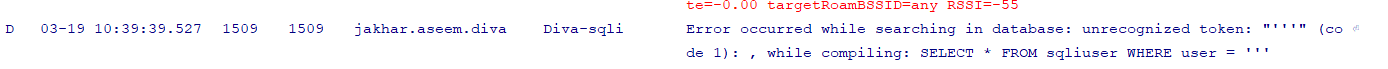




# 7 不安全的输入1（sqli）



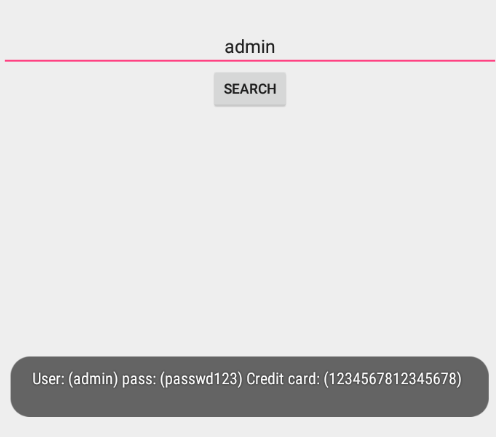
输入’，出现如下错误：



输入一个单引号，在日志中可以看到报错信息，存在sql注入。

输入' or '1'='1，可返回所有用户密码。

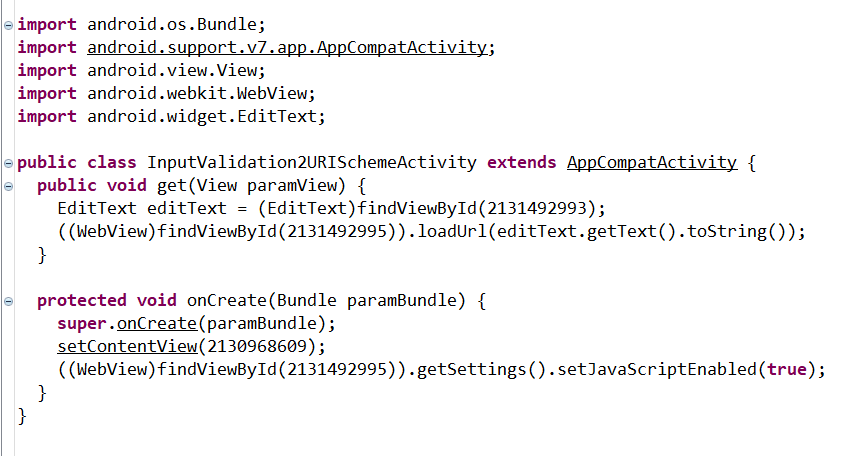
  
验证一下：

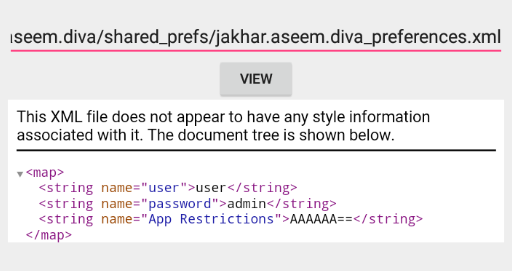


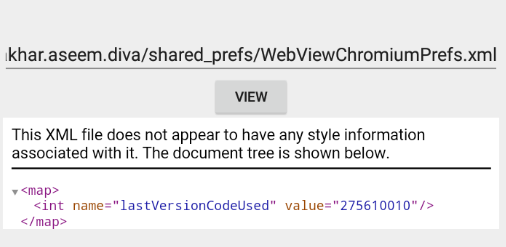
# 8不安全的输入2（读取本地文件）

可访问本地的任意文件，输入

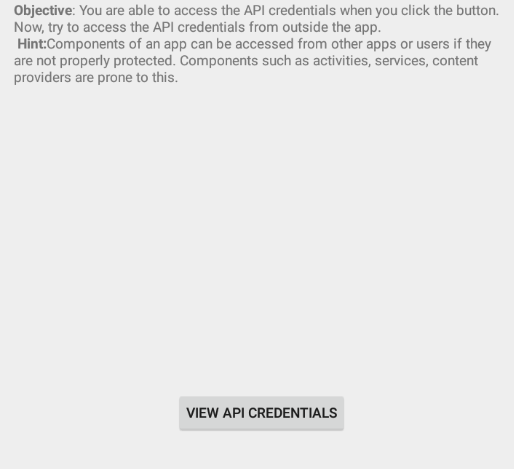
file:///data/data/jakhar.aseem.diva/shared\_prefs/jakhar.aseem.diva\_preferences.xml



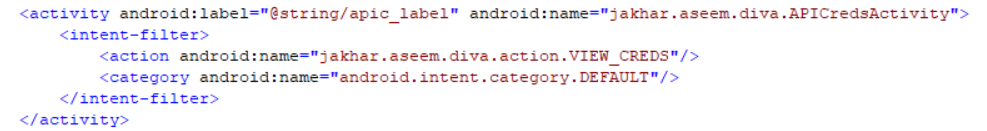


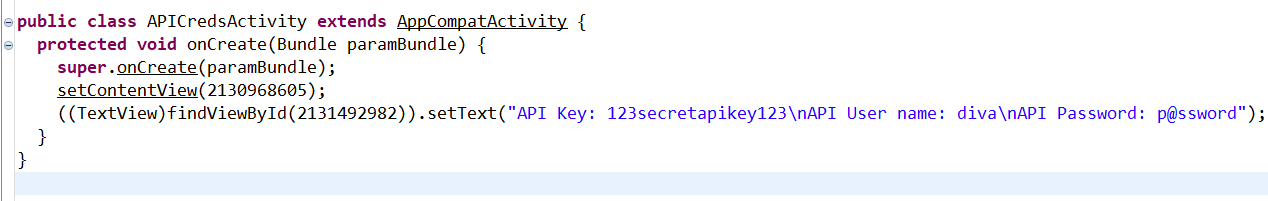


# 9访问控制1（Activity 1）

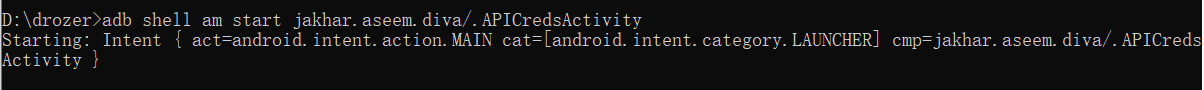


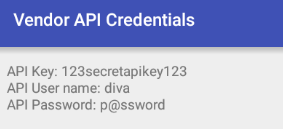
查看AndroidManifest.xml文件中暴露的activity组件：



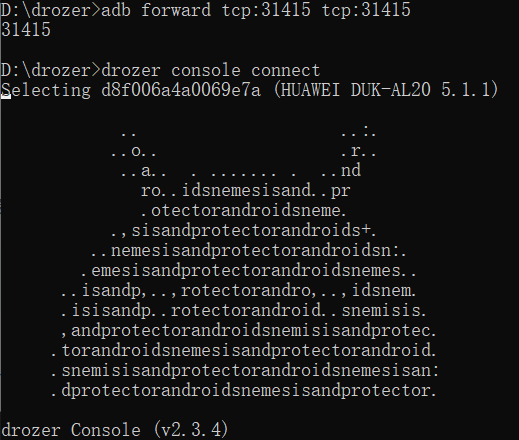


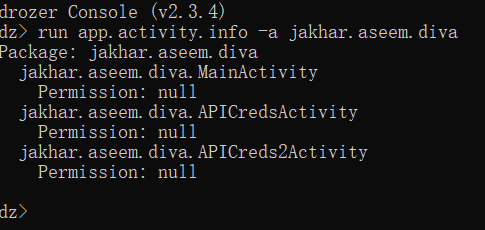
利用am(Activity Manager tool)启动暴露的组件，来绕过权限控制：

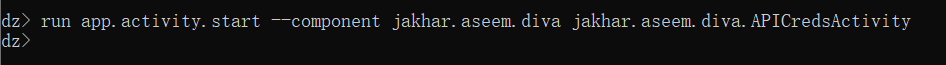


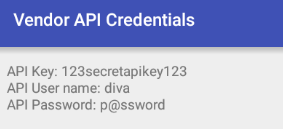


同时也可以使用drozer来完成：

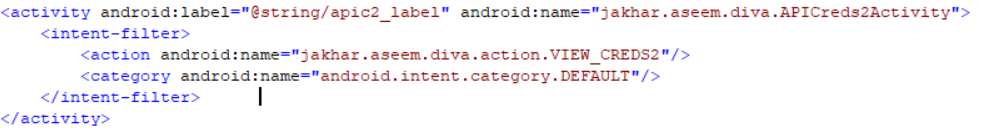




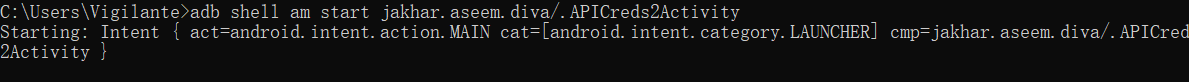


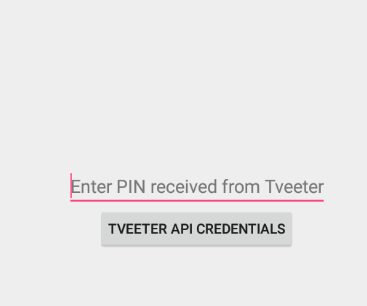


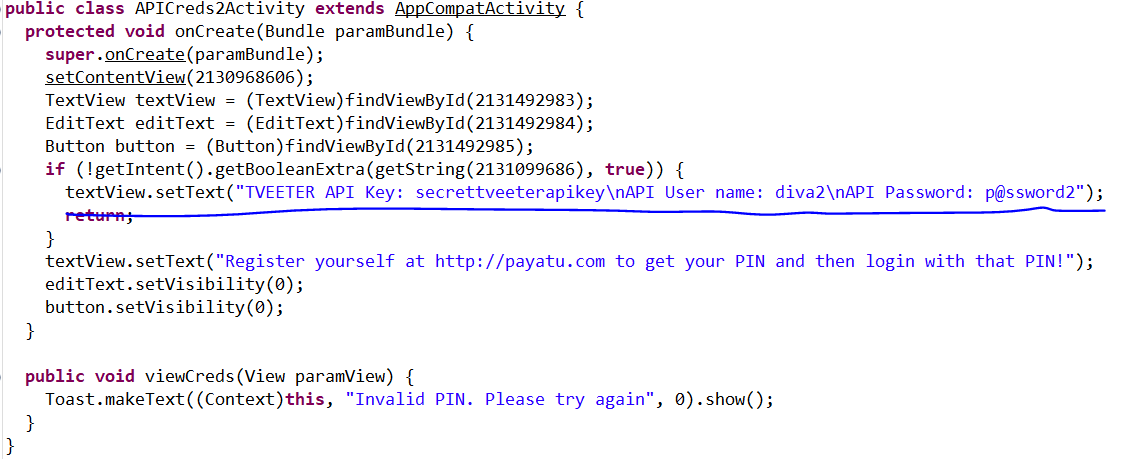
# 10 访问控制2（Activity 2）



输入 adb shell am start jakhar.aseem.diva/.APICreds2Activity，并没有出现PART9中的情况，添加了验证PIN码。

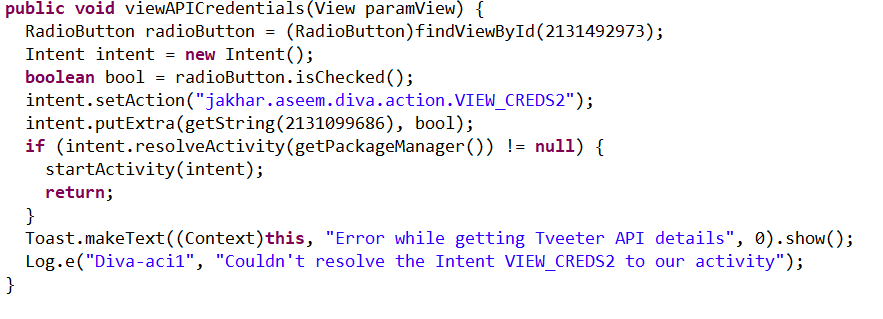




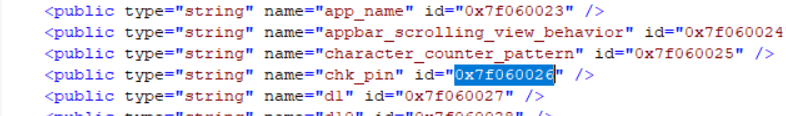


从Intent中获取到上个activity传来的boolean值（2131099686），当这个值为false时就视为用户已注册，再看看上个activity。

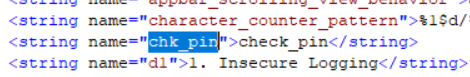
AccessControl2Activity.class



这个类中bool值是通过单选项来决定的，而且把值传给下个activity。这时我们可以使用–ez来传递一个boolean键值对，还要获取到2131099686值对应的key值是什么，一般情况下使用apktool反编译的字符串文件存放在res/vaules/strings.xml文件下，string文件中所有的字符串资源都会被R.java的String类中被标识，每个字符串中都有唯一的iint类型索引值，在反编译的情况下所有的索引值都保存在strings.xml同目录下的public.xml文件中，所以，我们现在public中查看16进制的2131099686（0x7f060026 ）对应的name值

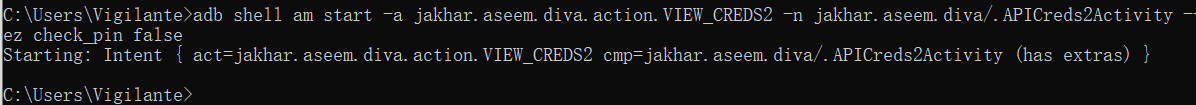


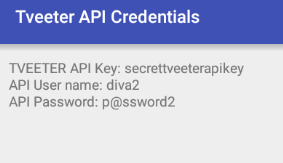
然后再strings文件下查找chk\_pin，最后得到key 为check\_pin。



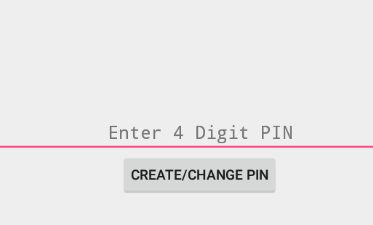
最后的命令：

$ adb shell am start -a jakhar.aseem.diva.action.VIEW\_CREDS2 -n jakhar.aseem.diva/.APICreds2Activity --ez check\_pin false





# 11 访问控制3（Content Providers ）



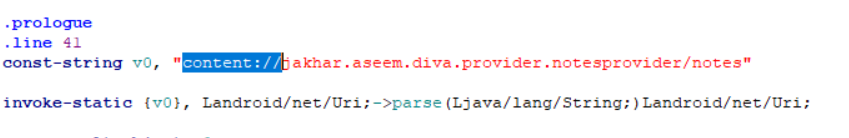
这是个私人的笔记app，一开始需要设置密码才能使用，我们的目标是不设置密码就开始使用。

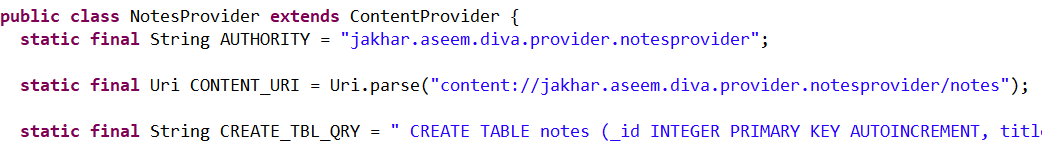


这里使用了ContentProvider，android:enabled表示是否能由系统初始化，android:exported表示是否能被其他应用使用，android:authorities：标识这个ContentProvider，调用者可以根据这个标识来找到它，具体内容可以查阅其他资料。

看到2个值都为true，我们就可以使用content://访问里面的数据了，搜索包含content://的字符串文件：

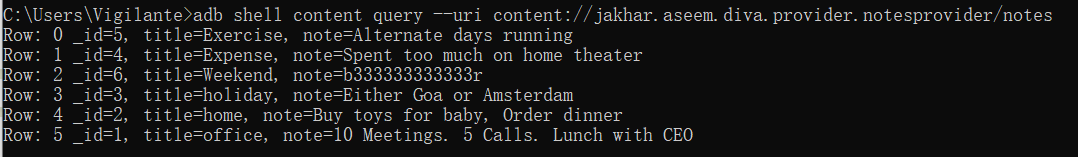
smali/jakhar/aseem/diva/NotesProvider.smali



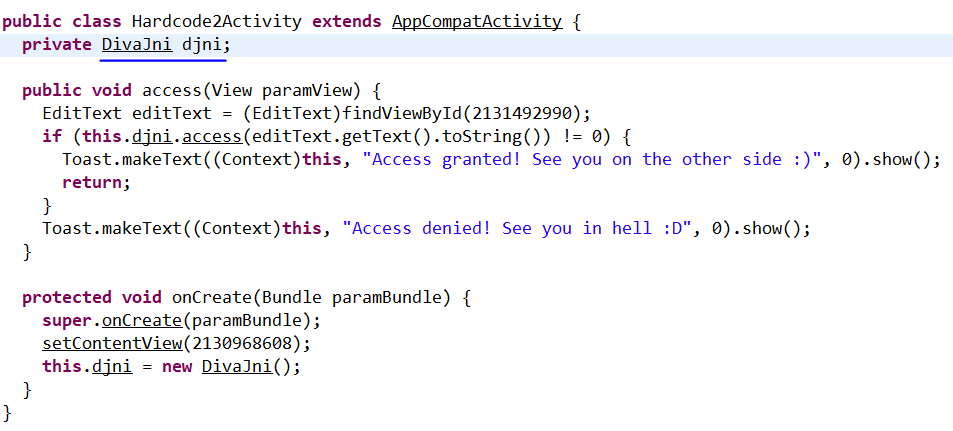


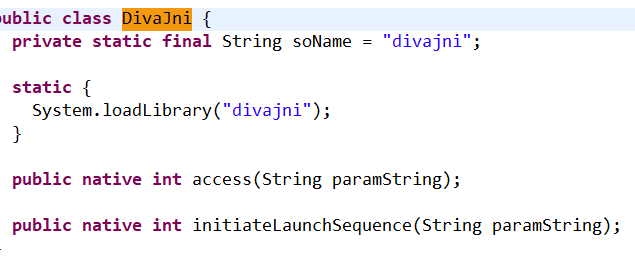
我们可以使用以下命令随意的访问该uri：

$ adb shell content query –-uri content://jakhar.aseem.diva.provider.notesprovider/notes

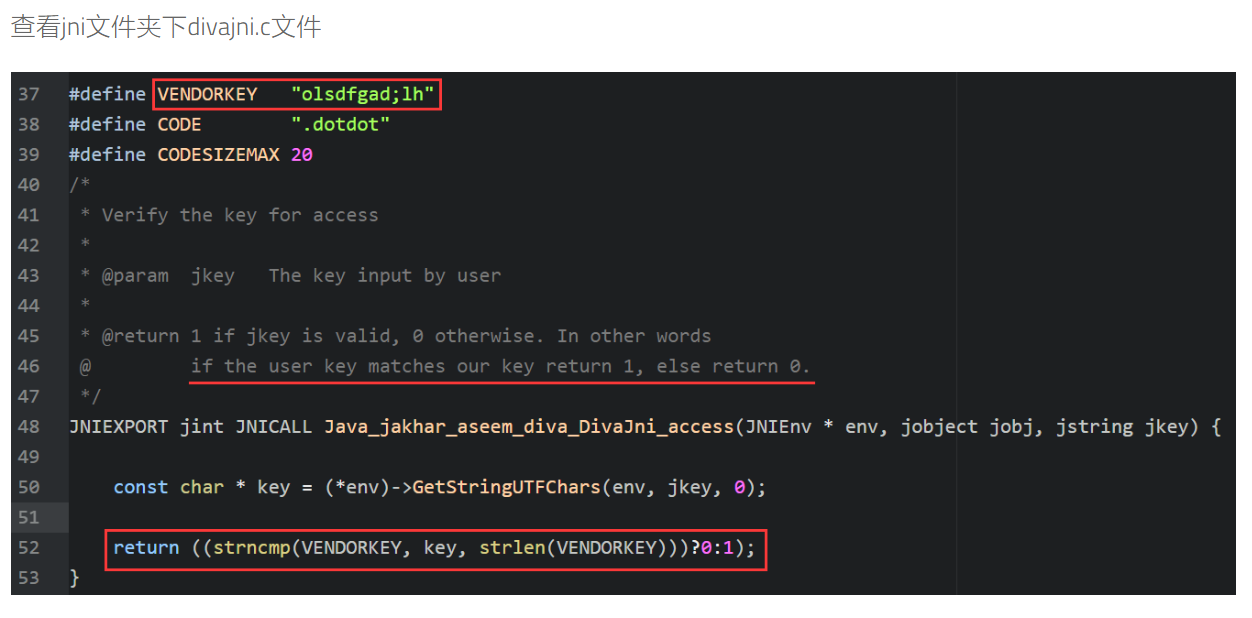


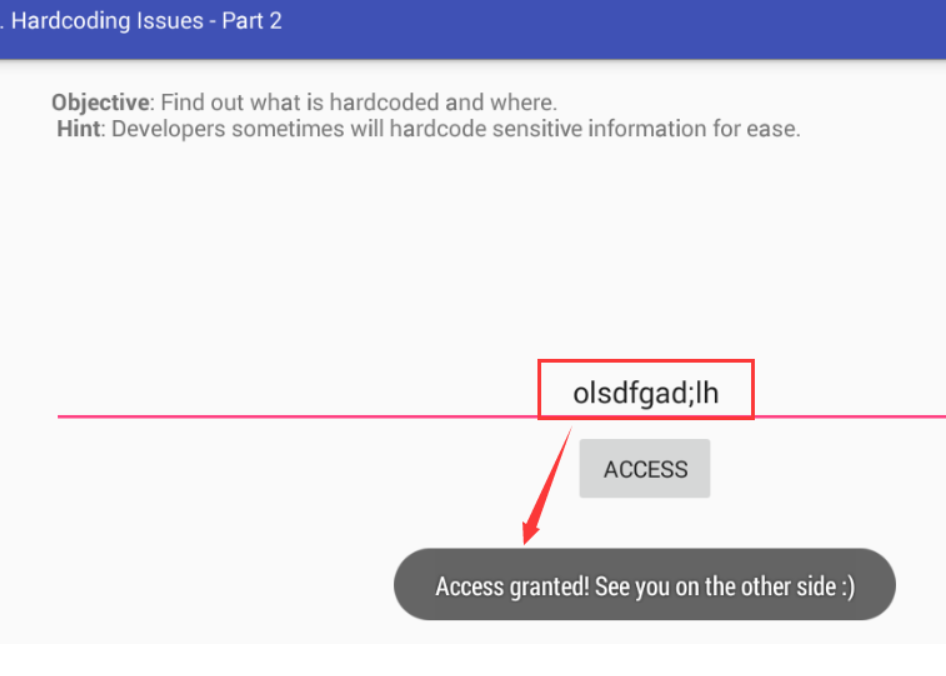
# 12硬编码2(so层)





源码中：<https://github.com/payatu/diva-android/blob/master/app/src/main/jni/divajni.c>





# 13不安全的输入3（字符串溢出）

字符串溢出。。。

<http://www.atomsec.org/android/diva%e7%b3%bb%e5%88%97%e6%95%99%e7%a8%8b-part-7-%e8%be%93%e5%85%a5%e9%aa%8c%e8%af%81%e9%97%ae%e9%a2%983/>