安全小课堂第139期【APP安全之四大组件漏洞】

JSRC 京东安全应急响应中心 5月27日

Android系统由于其开源的属性，市场上针对开源代码定制的ROM参差不齐，在系统层面的安全防范和易损性都不一样，Android应用市场对app的审核相对iOS来说也比较宽泛，为很多漏洞提供了可乘之机。市场上一些主流的app虽然多少都做了一些安全防范，但由于大部分app不涉及资金安全，所以对安全的重视程度不够；而且由于安全是门系统学科，大部分app层的开发人员缺乏安全技术的积累，措施相对有限。

据了解：反编译 Androidapk 现象的发生主要原因，在于开发商投入市场的Android apk包未经任何加固保护。总之就是现在的移动APP安全测试的需求迅速扩大，相关测试技能对于大家的日常工作来说是必不可少的。

JSRC小课堂第**139期**，邀请到**Lemon师傅（资深安全专家，从事网络安全工作五年，擅长漏洞挖掘和APP渗透测试）**，就APP安全之四大组件漏洞为大家进行分享，同时感谢白帽子们的精彩讨论。

APP面临的主要风险存在哪几个点?

APP面临的主要风险可以分为客户端风险和服务端风险。客户端风险又分为传统逆向分析类（反编译、调试、四大组件漏洞、加密/签名破解...）和用户已经中招类（输入记录、导出组件、进程注入...）。服务端风险又分为系统组件类（MS12-020、ShellShock、心血、ST2...）和业务应用类（注入跨站越权执行上传下载弱口令...）。

APP客户端的测试点以及工具有哪些?

测试工具环境准备：

JDK：因为Android应用都包含J*A外壳，所以J*A环境是必要的。没有J*A，就没有apktool、Eclipse、jarsigner;

ADT：ADT中除了神器adb（AndroidDebug Bridge）和Monitor，还包含了Emulator虚拟机和Eclipse开发环境；另有较为新锐的替代品Android Studio;

GDB：作为大名鼎鼎的跨平台调试工具，GDB在实际测试中主要用于DUMP内存，从而发现敏感信息、解密数据、脱壳等等；

NDK：NDK可以在Windows上编译Android(arm)使用的Native层可执行文件（C/C++），各项Native层测试工具都可用NDK制作；

安卓设备：已root的真实手机/Emulator，不推荐x86虚拟机（VirtualBox/Genymotion）；

x86虽然运行快，但是兼容性不佳，尤其客户APP涉及Native层时有时会出现莫名其妙的异常;

网络流量处理工具:

BurpSuite/Fiddler：针对HTTP进行各种操作的利器;

Wireshark：偶尔遇到不走HTTP的APP时，定位代码用;

APK文件处理工具:

ApkTool：功能众多，主要用来解包和打包Apk文件;

SignApk：对Apk文件进行签名，否则无法安装运行;

Dalvik反编译工具:

Dex2Jar：将Dex文件转换为Jar文件，便于反编译J*A;

Smali2Java：直接从APK中反编译J*A;

通用逆向分析工具:

JD-GUI/Luyten：可以反编译Class/Jar文件，在Dex2Jar之后使用;

IDA：当APP存在Native层代码时，用于进行逆向分析;

Xposed框架：

XposedBridge：JAR文件，Xposed开发所必需的接口库;

XposedInstaller：APK文件，Xposed运行环境，注意5.0前后版本不同;

JustTrustMe:

Xposed模块，能够解除绝大多数常见SSL函数库的证书校验;

注：通过修改本地客户端的方法实现的通信加密攻击不能记为风险;

BlockSecureFlag：

Xposed模块，能够解除所有APP的FLAG_SECURE设置;

注：适用于截屏时提示“内存不足XXXX”的场景;

Surrogate：

Xposed模块，能够手动配置修改任意函数的返回值，多用于固定化随机密钥;

注：灵活性不如XPOSED开发，但使用方便，适合开发基础不深的同学;

上面提到的组件漏洞能简单介绍下么?

安卓APP以组件为单位进行权限声明和生命周期管理;
安卓系统的四大组件:
Activity:呈现可供用户交互的界面,是最常见的组件;
Service:长时间执行后台作业,常见于监控类应用;
ContentProvider:在多个APP间共享数据,比如通讯录;
BroadcastReceiver:注册特定事件,并在其发生时被激活
权限声明:
如果一个APP或组件在没有声明权限的情况下就调用相关API,会被拒绝访问;但如果声明了相关权限,安装的时候就会有提示;这样一来,用户就可以评估使用该APP可能带来的风险。
组件导出的危害:
因为权限声明是以组件为单位的,A组件调用B组件的功能来访问操作系统API时,适用于B组件的权限声明。如果B作为导出组件,没有进行严格的访问控制,那么A就可以通过调用B来访问原本没有声明权限的功能,构成本地权限提升。
四大组件漏洞分别为Activity组件漏洞、Service组件、BroadcastReceiver导出漏洞、Content Provider组件漏洞。
Activity是Android组件中最基本也是最为常见用的四大组件之一,是一个负责与用户交互的组件。Activity组件中存在以下常见的漏洞。(1)activity绑定browserable与自定义协议activity设置“android.intent.category.BROWSABLE”属性并同时设置了自定义的协议android:scheme意味着可以通过浏览器使用自定义协议打开此activity。可能通过浏览器对app进行越权调用。(2)ActivityManager漏洞
ActivityManager类中的killBackgroundProcesses函数,用于杀死进程,属于风险API。还有通过ActivityManager被动嗅探intent。Intent嗅探脚本首先调用一个Context.getSystemService()函数,并传给它一个ACTIVITY_SERVICE标志的标识符,该函数返回一个ActivityManager类的实例,它使得该脚本能够与activitymanager进行交互,并通过这个对象调用ActivityManager.getRecentTasks()方法。最后把intent相关的信息格式化成字符串返回出来。
Service具有和Activity一样的级别,只是没有界面,是运行于后台的服务。其他应用组件能够启动Service,并且当用户切换到另外的应用场景,Service将持续在后台运行。另外,一个组件能够绑定到一个service与之交互(IPC机制),例如,一个service可能会处理网络操作,播放音乐,操作文件I/O或者与内容提供者(content provider)交互,所有这些活动都是在后台进行。从表面上看service并不具备危害性,但实际上service可以在后台执行一些敏感的操作。Service存在的安全漏洞包括:权限提升,拒绝服务攻击。没有声明任何权限的应用即可在

没有任何提示的情况下启动该服务，完成该服务所作操作，对系统安全性产生极大影响。BroadcastReceiver导出漏洞：当应用广播接收器默认设置exported='true'，导致应用可能接收到第三方恶意应用伪造的广播，利用这一漏洞，攻击者可以在用户手机通知栏上推送任意消息，通过配合其它漏洞盗取本地隐私文件和执行任意代码。Android 可以在配置文件中声明一receiver或者动态注册一个receiver来接收广播信息，攻击者假冒APP构造广播发送给被攻击的receiver，是被攻击的APP执行某些敏感行为或者返回敏感信息等，如果receiver接收到有害的数据或者命令时可能泄露数据或者做一些不当的操作，会造成用户的信息泄漏甚至是财产损失。ContentProvider为存储和获取数据提供统一的接口。可以在不同的应用程序之间共享数据。

（1）读写权限漏洞Content Provider中通常都含有大量有价值的信息，比如用的电话号码或者社交帐号登录口令，而确认一个content provider是否有能被攻击的漏洞的最好的办法，就是尝试攻击它一下。可以用drozer来寻找一些不需要权限的contentprovider:dz>runapp.provider.info –permission null这条命令能列出所有不需要任何读写权限的Content Provider，然后找到相对应的包，去访问给定包存放在它的ContentProvider中的数据。如果一些Content Provider的URI不需要读权限，那就可以通过drozer工具提取其中的数据。在某些情况下，设置和执行读写权限不当，也会将ContentProvider中的数据暴露给攻击者。除了提取数据，对于写权限管理不当的ContentProvider还可以向其中写入数据，使得攻击者可以将恶意数据插入到数据库中。

（2）Content Provider中的SQL注入漏洞和Web漏洞类似，安卓APP也要使用数据库，那就也有可能存在SQL注入漏洞。主要有两类，第一类是SQL语句中的查询条件子语句是可注入的，第二类是投影操作子句是可注入的。使用drozer可以很容易的找出查询条件子句可注入的content provider。dz> runapp.provider.query [URI] –selection “1=1” 也可以使用其他恒为真的值，例如 “1-1=0” ， “0=0” 等等。如果APP存在SQL注入漏洞，那么输入这行指令后就会返回数据库中的整张表。

（3）Provider文件目录遍历漏洞当Provider被导出且覆写了openFile方法时，没有对Content Query Uri进行有效判断或过滤。攻击者可以利用openFile()接口进行文件目录遍历以达到访问任意可读文件的目的。

关于四大组件漏洞要如何测试?

组件测试工具-drozer，Drozer是MWRLabs开发的一款Android安全测试框架。是目前最好的Android安全测试工具之一。

环境准备：

1.手机获得root权限

2.adb.exe、配置android环境变量

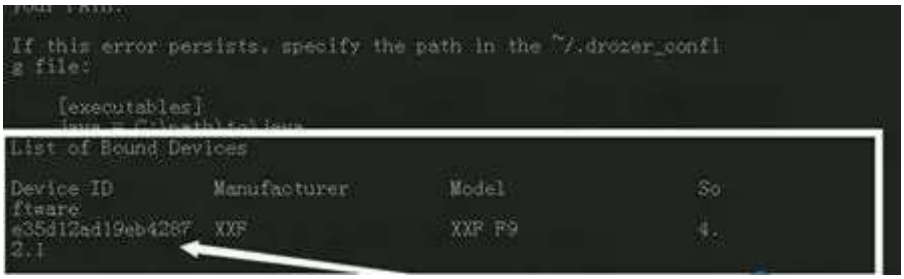
3.手机usb连接开启debug模式(在设置>关于手机>连续点击多次版本号,即可开启开发者模式)

4.Window下安装drozer

5.安装完drozer后在其目录下把agent.apk安装到手机

连接准备：

drozerconsole devices



启动drozer：

adb forward tcp:31415 tcp:31415 //将pc端31415的所有数据转发到手机上的31415端口 drozer console connect //使用drozer console 连接agent

获取手机上所有安装的app包名：runapp.package.list 加上”-f [app关键字]”查找某个app，如runapp.package.list -f sieve

获取sieve的基本信息run app.package.info–a com.mwr.example.sieve

可以看到sieve的版本信息，数据存储目录，用户ID，组ID，共享库，权限等信息

run app.package.attacksurface com.mwr.example.sieve

进一步获取每个组件的攻击面信息，如activity

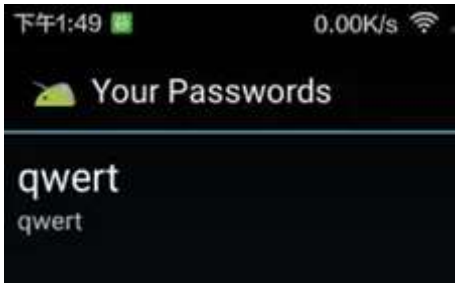
run app.activity.info这条命令将导出你设备上的所有的activity

run app.activity.info -a com.mwr.example.sieve

其中. MainLoginActivity是app启动时的主界面，必须可以导出，但其他两个activity正常情况下是不能导出的

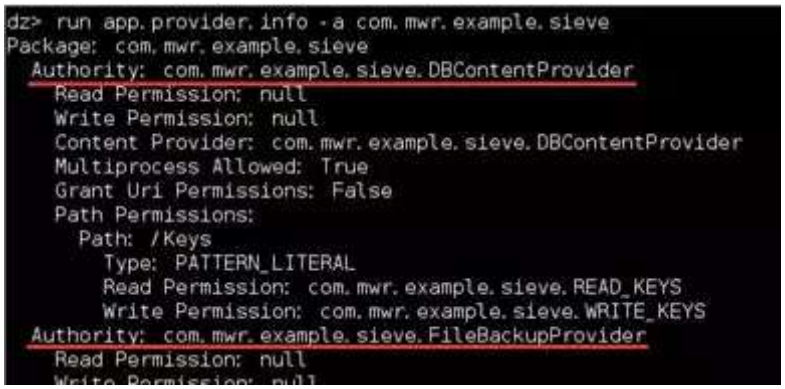
用drozer来启动可导出且不需要权限的activity

run app.activity.start --component com.mwr.example.sieve com.mwr.example.sieve.PWList



获取content provider的信息

run app.provider.info -a com.mwr.example.sieve



结合上面查看攻击面的信息，这2个content provider都可导出，

com.mwr.example.sieve.DBContentProvider/Keys 是需要读写权限的

如何查看四大组件的安全配置呢?

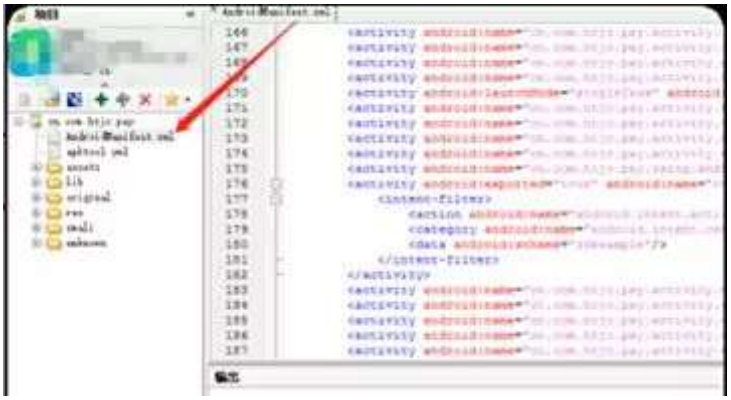
下面就是我们再AndroidManifest.xml中定义的组件的常见形式。

```
<activity
    android:name="com.xxx.helloworld.MainActivity"
```

```
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```

我们主要介绍的是activity通过exported属性来对activity进行的安全控制。首先讲下默认情况，如果activity没有设置intent-filter，则exported默认的属性为false,就是这个组件仅仅能被自身内部程序调用。如果activity设置了intent-filter,则android:exported默认为true。这时如果我们对这个值进行控制就会导致一系列安全问题。所以组件安全也主要针对配置了意图过滤的组件。

这时我们把组件分为公有组件和私有组件，公有组件就是activity组件可以被外部程序调用，私有组件就是不能被其他程序启动或调用。因此在创建组件时，如果是私有的组件，android:exported属性一律设置为false.如果是公有的，就设置android:exported为true。不管公有的还是私有的组件，处理接收的intent时都应该进行验证的数据验证。公有组件防止信息泄露和接收外部数据时进行严格的处理。如果对私有组件没有进行相应的配置，可能导致组件被其他程序调用，敏感信息泄露，拒绝服务器攻击和权限绕过等漏洞。



如图所示AndroidManifest.xml是Android应用的入口文件，它描述了package中暴露的组件（activities, services, 等等），他们各自的实现类，各种能被处理的数据和启动位置。除了能声明程序中的Activities, ContentProviders, Services, 和In android:allowClearUserData（‘true’ or ‘false’）

用户是否能选择自行清除数据，默认为true，程序管理器包含一个选择允许用户清除数据。当为true时，用户可自己清理用户数据，反之亦然。

AndroidManifest.xm一些特定属性的介绍，可自行百度。下面只是几个举例：

android:allowTaskReparenting（‘true’ or ‘false’）

是否允许activity更换从属的任务，比如从短信息任务切换到浏览器任务

android:debuggable

这个从字面上就可以看出是什么作用的，当设置为true时，表明该APP在手机上可以被调试。默认为false,在false的情况下调试该APP，就会报以下错误：

Device XXX requires that applications explicitely declarethemselves as debuggable in their manifest. Application XXX does not have the attribute ‘debuggable’ setto TRUE in its manifest and cannot be debugged.

android:exported是Android中的四大组件 Activity，Service，Provider，Receiver四大组件中都会有的一个属性。

总体来说它的主要作用是：是否支持其它应用调用当前组件。

默认值：如果包含有intent-filter 默认值为true;没有intent-filter默认值为false。

这是JSRC小课堂陪伴你的

第3年76天

如果有你希望出现在安全小课堂内容暂时未出现

欢迎留言告诉我们

如果有所收获欢迎将它分享

让更多的人加入JSRC安全小课堂

交流

开课时间：周五下午15:30

QQ开课群：464465695

留言：针对本期主题内容，你还有什么疑问吗？欢迎留言交流~

JSRC