Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

Android 应用安全检测

硕士研究生 李师伟

2017年10月19日

内容提要

- 基础知识
  - 基本概念
  - 基本步骤
- APK文件反编译
  - 反编译流程
  - 相关介绍
- 检测原理与方法
  - 检测类别
  - 组件暴露
- 检测实例
  - 本地拒绝服务
  - 远程代码执行
  - 同源策略绕过
  - Drozer

基础知识

基础知识

- 安卓APP是一个APK压缩包，其中包含了APP运行所需要的所有信息
  - 代码、资源、配置文件……
- 安卓APP安装的过程就是一个APK复制、解压缩的过程
- Dalvik是Google公司专为Android平台设计的虚拟机，可支持Java平台被转换成紧凑的Dalvik可执行格式(.dex)的应用程序，适合内存和处理器速度受限的系统

基础知识

- APK
  - Android安装包，本质是一个zip文件，运行时首先需要UnZip。
- Dex
  - Dalvik VM executes的简称，由APK文件解压直接得到，Dalvik虚拟机通过加载Dex文件来运行APP程序。
- Smali
  - 由APK文件经过反编译之后生成的文件，类似于汇编语言的底层计算机语言，涉及寄存器的直接操作，是Android虚拟机所使用的寄存器语言。
- Jar
  - 由Dex文件反编译得到的压缩文件，是开发过程最终编译生成的执行程序，其中包含了多个.class文件，每一个.class是一个类，实现一项特定功能。

基础知识

- 基本的检测步骤
  - 反编译
  - 检测
    - 静态检测源码
    - 动态检测运行中的应用

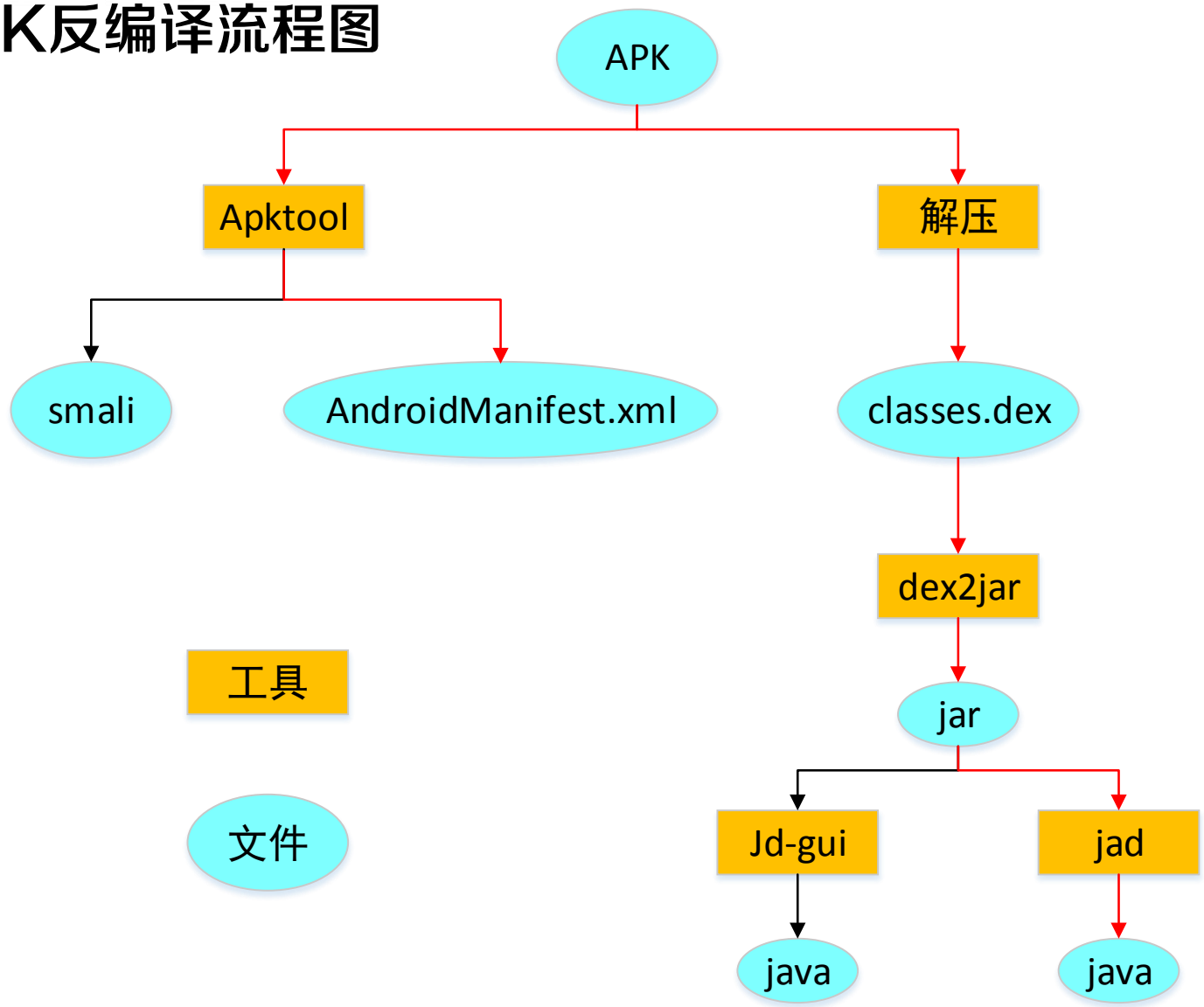
APK文件反编译

APK结构

- APK文件结构
  - AndroidManifest.xml
  - classes.dex
  - assets
  - lib
  - META-INF
  - res
  - resources.arsc

APK反编译流程

- APK反编译流程图



AndroidManifest.xml

- 反编译前后对比


```
<?xmlversion="1.0"encoding="utf-8"?>
<manifest>
    <uses-sdk/>
    <uses-configuration/>
    <uses-feature/>
    <uses-permission/>
    <permission/>
    <permission-tree/>
    <permission-group/>
    <instrumentation/>
    <supports-screens/>
    <application>
        <activity>
            <intent-filter>
                <action/>
                <category/>
            </intent-filter>
        </activity>
        <service>
            <intent-filter></intent-filter>
            <meta-data/>
        </service>
        <receiver>
            <intent-filter></intent-filter>
            <meta-data/>
        </receiver>
        <provider>
            <grant-uri-permission/>
            <meta-data/>
        </provider>
    </application>
</manifest>
```

AndroidManifest.xml

- 需要关注的属性
  - allowBackup
  - debuggable
  - sharedUserId
  - activity
  - service
  - receiver
  - Provider
  - uses-permission
  - permission: protectionLevel
    - Normal
    - Dangerous
    - Signature
    - signatureOrSystem

反编译效果对比

- Demo反编译效果对比

源代码

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this,
                    "you clicked button", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

反编译后

```
package com.example.guolin.androidtest;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity
  extends AppCompatActivity
{
  protected void OnCreate(Bundle paramBundle)
  {
    super.onCreate(paramBundle);
    setContentView(2130968601);
    ((Button)findViewById(2131492944)).setOnClickListener(new View.OnClickListener()
    {
      public void onClick(View paraAnonymousView)
      {
        Toast.makeText(MainActivity.this,"you clicked button",0).show();
      }
    });
  }
}
```

检测原理与方法

检测原理与方法

- 目前常做的检测类别
  - 用户隐私
  - 文件权限
  - 网络通讯(TLS/SSL)
  - 运行时解释保护
  - Android组件权限保护
  - 升级
  - 3rd库

四大组件简介

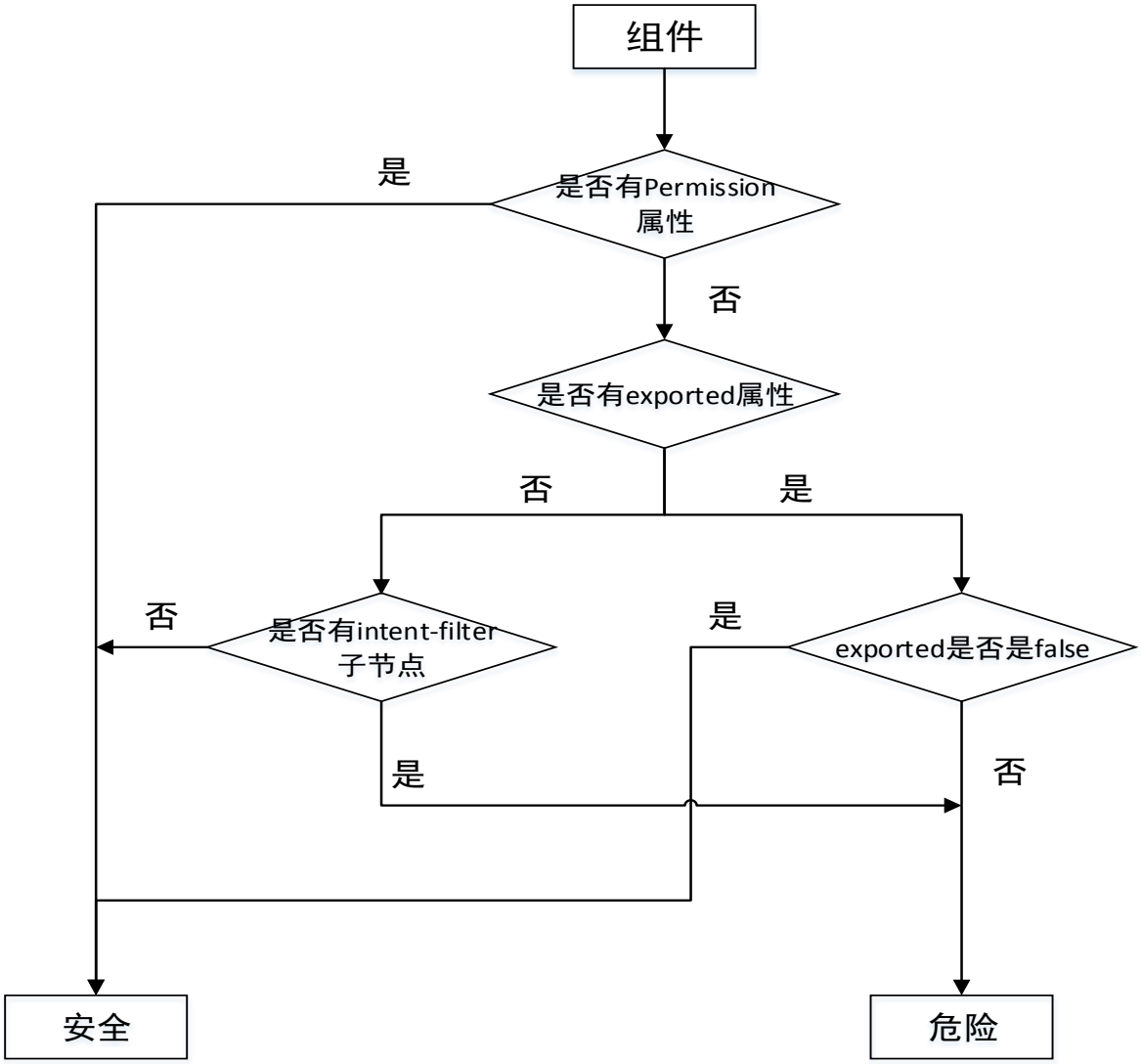
- Activity
  - 一个Activity通常就是一个单独的屏幕（窗口）
  - Activity之间通过Intent进行通信
  - 每一个Activity都必须要在AndroidManifest.xml配置文件中声明，否则系统将不识别也不执行该Activity
- Service
  - Service分为两种：started(startService)/bound(bindService)
  - Service通常位于后台运行，它一般不需要与用户交互，因此Service组件没有图形用户界面。Service组件需要继承Service基类。Service组件通常用于为其他组件提供后台服务或监控其他组件的运行状态

四大组件简介

- Content provider
  - 其他应用可以通过ContentResolver类从该内容提供者中获取或存入数据
  - 只有需要在多个应用程序间共享数据时才需要内容提供者
  - ContentProvider实现数据共享。ContentProvider用于保存和获取数据，并使其对所有应用程序可见，这是不同应用程序间共享数据的唯一方式
- Broadcast receiver
  - 对外部事件进行过滤，只对感兴趣的外部事件进行接收并做出响应
  - 两种注册方法：程序动态注册和AndroidManifest文件静态注册

组件暴露判断

- 检测组件是否暴露



检测实例

安全检测实例

- 组件本地拒绝服务
  - 特征函数：Intent.getXXX()
  - 触发条件
    - 定位到Intent.getXXX()方法的位置
    - 判断是否为导出的组件【exported属性为true】
    - 判断是否进行异常捕获【try...catch】
  - 漏洞原理
    - 导出的组件在处理Intent附加数据的时候，没有进行异常捕获，攻击者可通过向应用发送空数据、异常或畸形数据等，导致应用程序崩溃

安全检测实例

- WebView组件远程代码执行漏洞
  - 特征函数：addJavascriptInterface()
  - 触发条件
    - 使用addJavascriptInterface方法注册可供JavaScript调用的Java对象
    - Android系统版本低于4.2
  - 漏洞原理
    - WebView组件中的addJavascriptInterface方法用于实现本地Java和JavaScript的交互，但是该函数并没有对方法调用进行限制，导致攻击者可以调用任何JAVA类，最终导致JavaScript代码对设备进行任意攻击

安全检测实例

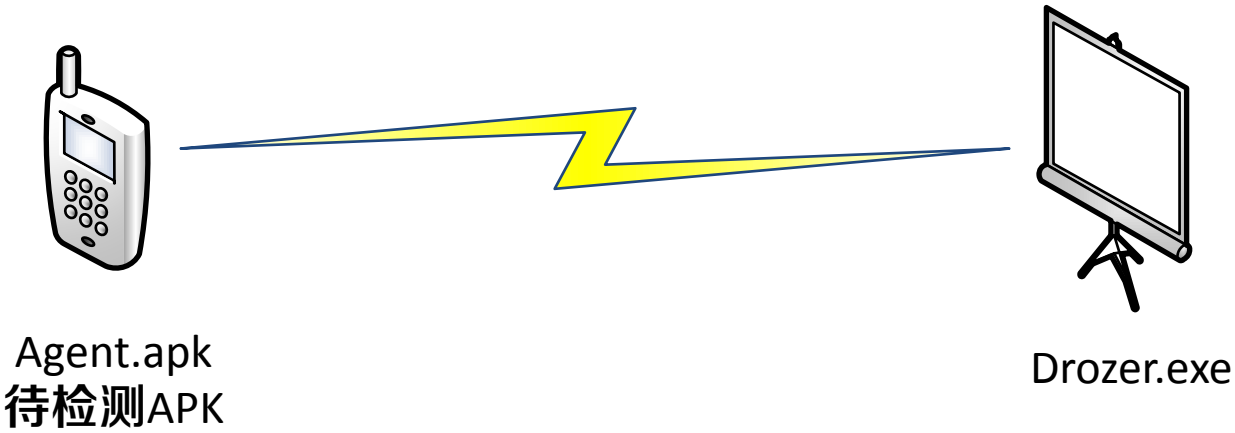
- WebView File域同源策略绕过漏洞
  - 特征函数：setAllowFileAccess()/setJavaScriptEnabled()
  - 触发条件
    - 调用WebSettings.setAllowFileAccess方法，设置setAllowFileAccess(true)或没有显示设置(默认为true)
    - 调用WebSettings.setJavaScriptEnabled方法，设置　setJavaScriptEnabled(true)
  - 漏洞原理
    - 应用程序一旦使用WebView，同时支持File域，并打开了对JavaScript的支持，就能利用JavaScript的延时执行，绕过file协议的同源检查，并能够访问应用程序的私有文件，导致敏感信息泄露

安全检测实例

- 检测的代码中的非常见情况
  - 一个函数体全部编写在一行
  - 在符合编译器规则情况下，每行只有一个符号
  - 对于多个参数的函数，此多个参数原则上可以任意放置在任何行

Drozer

Drozer是一款针对Android系统的安全测试框架。Drozer可以通过与Dalvik VM，其他应用程序的IPC端点以及底层操作系统的交互，避免Android应用程序和设备暴露出不可接受的安全风险。



Drozer

- Drozer框架结构
  - Drozer可以看成是C/S结构的工具。PC端是用Python写的，Agent端是用java写的。利用Google Protocol Buffer协议作为消息载体。很重要的一点就是利用反射，很大一部分工作都是由Agent代理完成，最后由PC端处理解析。

- 通信协议
  - 这种协议是一种轻便高效的结构化数据存储格式，可以用于结构化数据串行化，目前提供了C++/Java/Python三种语言的API。
    - SYSTEM_REQUEST/ SYSTEM_RESPONSE
    - REFLECTION_REQUEST/

Drozer测试实例

```
dz> run app.package.info -a com.xhl.cq
Package: com.xhl.cq
  Application Label: 閱趄簡
  Process Name: com.xhl.cq
  Version: 01.00.0038
  Data Directory: /data/data/com.xhl.cq
  APK Path: /data/app/com.xhl.cq-1.apk
  UID: 10044
  GID: [3003, 1028, 1015, 1023, 3001, 3002]
  Shared Libraries: null
  Shared User ID: null
  Uses Permissions:
  - android.permission.INTERNET
  - android.permission.ACCESS_COARSE_LOCATION
  - android.permission.ACCESS_FINE_LOCATION
  - android.permission.ACCESS_FIND_LOCATION
  - android.permission.ACCESS_MOCK_LOCATION
  - android.permission.ACCESS_NETWORK_STATE
```

```
dz> run app.package.attacksurface com.xhl.cq
Attack Surface:
  10 activities exported
  5 broadcast receivers exported
  2 content providers exported
  2 services exported
    is debuggable
```

```
dz> run app.provider.info -a com.xhl.cq
Package: com.xhl.cq
  Authority: com.xhl.cq.sp
    Read Permission: null
    Write Permission: null
    Content Provider: org.zywx.wbpalmstar.base.ACEContentProvider
    Multiprocess Allowed: False
    Grant Uri Permissions: False
'ReflectedNull' object has no attribute 'split'
```

```
dz> run app.provider.finduri com.snda.youni
Scanning com.snda.youni...
Could not find dx.
Ensure that dx is installed and on your PATH. If this error persists pleas
the path to your .drozer_config.
Could not find javac.
Ensure that javac is installed and on your PATH. If this error persists pl
dd the path to your .drozer_config.
content://com.snda.youni.providers.DataStructs/room
content://mms/drafts/
content://com.snda.youni.providers.DataStructs/mypurchase/
content://sms/conversations
content://sms
content://mms/part
```

谢谢！

大成若缺，其用不弊。大盈若冲，其用不穷。大直若屈。大巧若拙。大辩若讷。静胜躁，寒胜热。清静为天下正。