Android逆向入门篇--java层静态分析

yong夜 / 2019-07-01 06:02:00 / 浏览数 4730 安全技术 移动安全 顶(1) 踩(0)

[TOC]

概述

了解了编译、打包、签名、安装apk文件后，正式开始逆向的基础，静态分析

java层

apk包内的dex文件是dalvik虚拟机可识别的可执行文件，我们主要也是对dex文件进行逆向，分析其代码逻辑、更改其逻辑做一些分析、破解之类的行为

工具

apktool

androidkiller

- jeb
- jadx
- GDA
- smali/baksmali
- ....

破解流程

1. 反编译apk
2. 定位关键代码
3. 功能分析
4. smali修改
5. 重打包、签名、安装

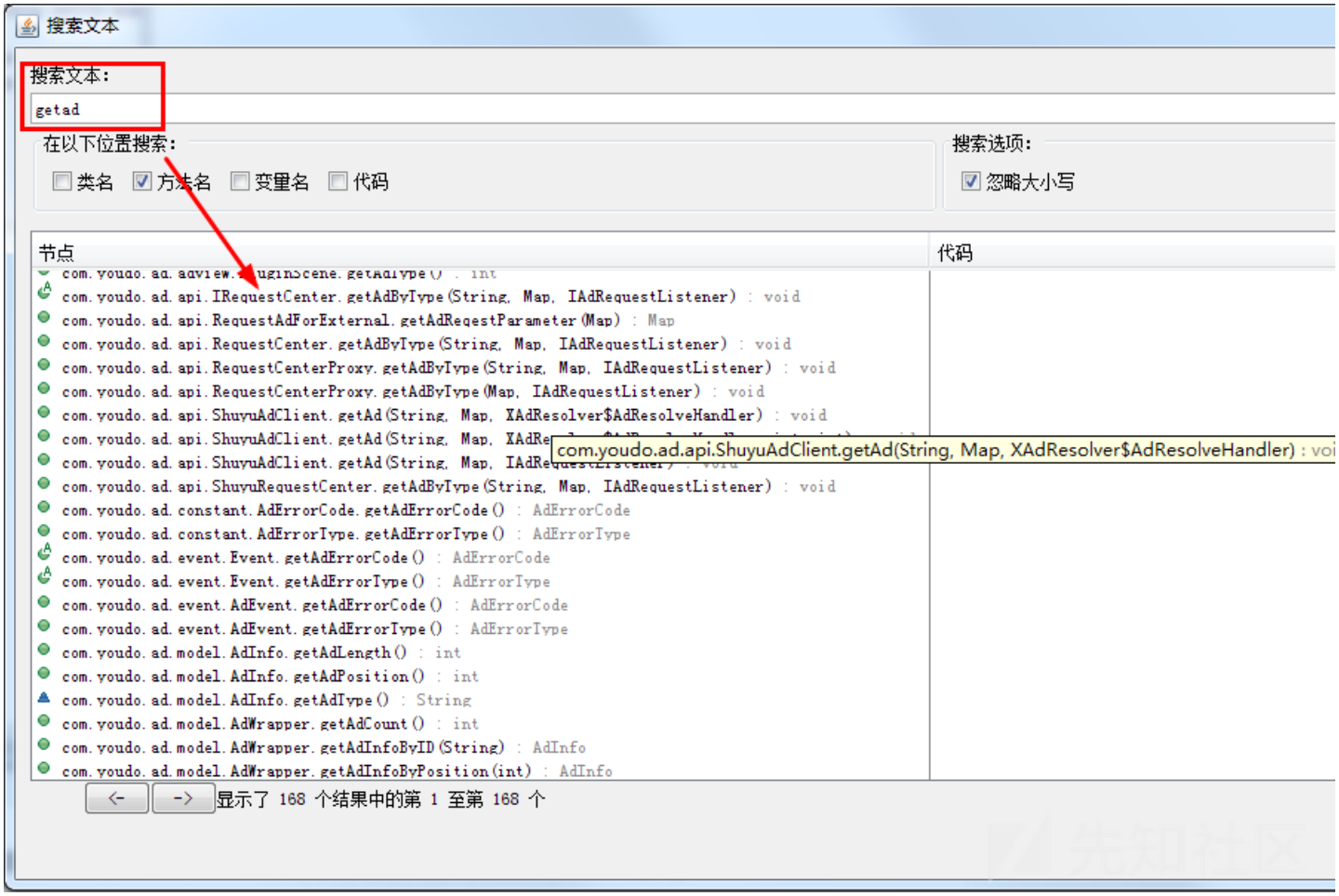
例子1，广告破解

这里我们用一个去广告的例子，简单的过一下流程

- 反编译

```
java -jar apktool.jar d xxx.apk -o out
```

定位关键代码的方法很多，这里我用的是一个开源的小工具，原理是■■+■■■，我们可以先用字符串大法先随意拼接以下ad字符串大致定位一下，可能会在哪个文件夹中




```
python3 inject_log.py -r .\out\smali\com\youdo
```

由于我们的目的是去广告，所以我们需要打开一个视频，查看日志中，广告开始播放时调用了哪些方法，这里我们就这样不断的去缩小我们过滤的范围，这里不做太多的
v0, 0x0,让返回始终为null

```
const/4 v0, 0x0
return-object v0
```

CTF-CrakeMe流程

首先针对不同逆向，我们需要清楚逆向的目的，在crakeme赛题中，我们主要目的是找到flag，具体需要我们找到打印输出的地方，在其周围找到判断逻辑，然后就是最主要

1.反编译

2.定位关键代码

3.逆向算法

例子2

反编译后，查看AndroidManifest.xml找到入口 点

```
<activity android:label="@h/a" android:name="ctf.bobbydylan.M">
          <intent-filter>
              <action android:name="android.intent.action.MAIN"/>
              <category android:name="android.intent.category.LAUNCHER"/>
          </intent-filter>
```

进入这个Activity类中，我们根据执行逻辑，如果有static{}和构造方法，我们可以看一眼里面除了初始化是否还有别的东西，这里并没有这两个调用，我们直接去onCreate方

分析代码：主要有个开启服务的方法，进入这个P服务类，扫一眼生命周期中用到的方法，并没有值得留意的东西，直接看onStartCommand方法，服务启动时执行的方法

```
public void onCreate(Bundle bundle) {
       super.onCreate(bundle);
       setContentView(R.layout.main);
       startService(new Intent(this, P.class));
```

```
        ((Button) findViewById(R.id.button)).setOnClickListener(new a(this, (TextView) findViewById(R.id.et)));
    }

public int onStartCommand(Intent intent, int i, int i2) {
        try {
            if (this.a == null) {
                this.a = MediaPlayer.create(this, R.raw.bobdylan);
                this.a.start();
                this.a.setLooping(true);
            }
        } catch (Exception e) {
        }
        return super.onStartCommand(intent, i, i2);
    }
```

按钮控件设置监听方法，在Jadx-Gui中右键->Find
Usage找到方法定义的地方，可以从中文字符看出我们找到了主要逻辑，我们的目的就是打印出正确二字，从这个逻辑中可以看出，我们必须保证this.b.check不能抛出异

```
a(M m, TextView textView) {
        this.b = m;
        this.a = textView;
    }

    public void onClick(View view) {
        try {
            this.b.check(this.a.getText().toString());
            new Builder(this.b).setMessage("■■").setNeutralButton("OK", null).create().show();
        } catch (Exception e) {
            new Builder(this.b).setMessage("■■").setNeutralButton("OK", null).create().show();
        }
    }
```

check

我们必须保证不抛出异常

满足下面注释的两个条件即可

```
public void check(String str) {
        int i = 0;
        //■■1■■■■■■■■■■■■16
        if (str.length() != 16) {
            throw new RuntimeException();
        }
        String str2 = "";
        try {
                str2 = get■ey();
        } catch (Exception e) {
            str2 = getKey();
            System.arraycopy(str2, 0, str, 5, 5);
        }
        int[] iArr = new int[16];
        iArr[0] = 0;
        iArr[12] = 14;
        iArr[10] = 7;
        iArr[14] = 15;
        iArr[15] = 42;
        try {
            iArr[1] = 3;
            iArr[5] = 5;
            System.out.println();
        } catch (Exception e2) {
            iArr[5] = 37;
            iArr[1] = 85;
        }
        iArr[6] = 15;
        iArr[2] = 13;
        iArr[3] = 19;
        iArr[11] = 68;
        iArr[4] = 85;
```

```
        iArr[13] = 5;
        iArr[9] = 7;
        iArr[7] = 78;
        iArr[8] = 22;
        //■■2■iArr■■&255■■■■■■■■■■■■
        while (i < str.length()) {
            if ((iArr[i] & 255) != ((str.charAt(i) ^ str2.charAt(i % str2.length())) & 255)) {
                throw new RuntimeException();
            }
            i++;
        }
    }

public String getKey() {
        return "bobbydylan";
    }
```

根据面的分析写出第一版算不出结果的解密脚本，然后我用androidkiller打开这个apk文件，看汇编代码发现这个get□ey不是调用M类中的方法，因为是个乱码，实际上是调

```
iArr = [0, 3, 13, 19, 85, 5, 15, 78, 22, 7, 7, 68, 14, 5, 15, 42]
chArr = ['b', 'o', 'b', 'b', 'y', 'd', 'y', 'l', 'a', 'n']
for i in iArr:
    number = 0
    while(True):
        if iArr[i] == number ^ ord(chArr[i%len('bobbydylan')]):
            print(chr(number))
            if i == len(iArr):
                print("end")
                exit(0)
            break
        number += 1
```

```
:try_start_0
invoke-virtual {p0}, Lctf/bobbydylan/M;->get□ey()Ljava/lang/String;
:try_end_0
.catch Ljava/lang/Exception; {:try_start_0 .. :try_end_0} :catch_0

move-result-object v0
```

解密算法，结果blow,in the winD

```
iArr = [0, 3, 13, 19, 85, 5, 15, 78, 22, 7, 7, 68, 14, 5, 15, 42]
chArr = ['b', 'o', 'b', 'd', 'y', 'l', 'a', 'n']
for i in range(len(iArr)):
    number = 0
    while(True):
        if iArr[i] == number ^ ord(chArr[i%len('bobdylan')]):
            print(chr(number))
            if i == len(iArr):
                print("end")
                exit(0)
            break
        number += 1
```

小结

【1】当java层代码不能给我们正确答案是，可以阅读smali代码

病毒分析

根据这几篇文章的逆向流程来分析病毒即可

分析流程

1. 大致阅览安装包内有哪些文件。资源目录raw、lib等
2. 看AndroidManifest.xml文件，看有哪些注册组件，主要看server组件和receiver组件，详细分析组件行为
3. 从入口类触发看逻辑

4. 汇总

【1】zoopark https://www.freebuf.com/articles/system/184286.html

【2】锁屏病毒 https://www.freebuf.com/articles/others-articles/199515.html

小结

上面从具体逆向的一些分支的实现中来了解一下java层的具体实现，难度其实没那么大，但是需要对Android开发一些基础知识有一定的掌握

参考

【1】批量注入栈跟踪小工具

【2】crakeme赛题 https://bbs.pediy.com/thread-251787-1.htm

上一篇：扫描器开发笔记-404页面识别 下一篇：分析某旺ActiveX控件Imag...

1. 0 条回复
   - 动动手指，沙发就是你的了！