



移动互联网安全

第七章 移动终端应用逆向分析

黄 玮



内容提纲

- 软件逆向分析基础之静态分析
- 软件逆向分析基础之动态分析
- 通信协议逆向分析基础



静态分析基础

- 静态分析是指在不运行代码的情况下，采用词法分析、语法分析等各种技术手段对程序文件进行扫描从而生成程序的反汇编代码，然后阅读反汇编代码来掌握程序功能和理解程序原理的一种技术
- 反汇编（或反编译）代码生成工具简称反汇编（反编译）工具

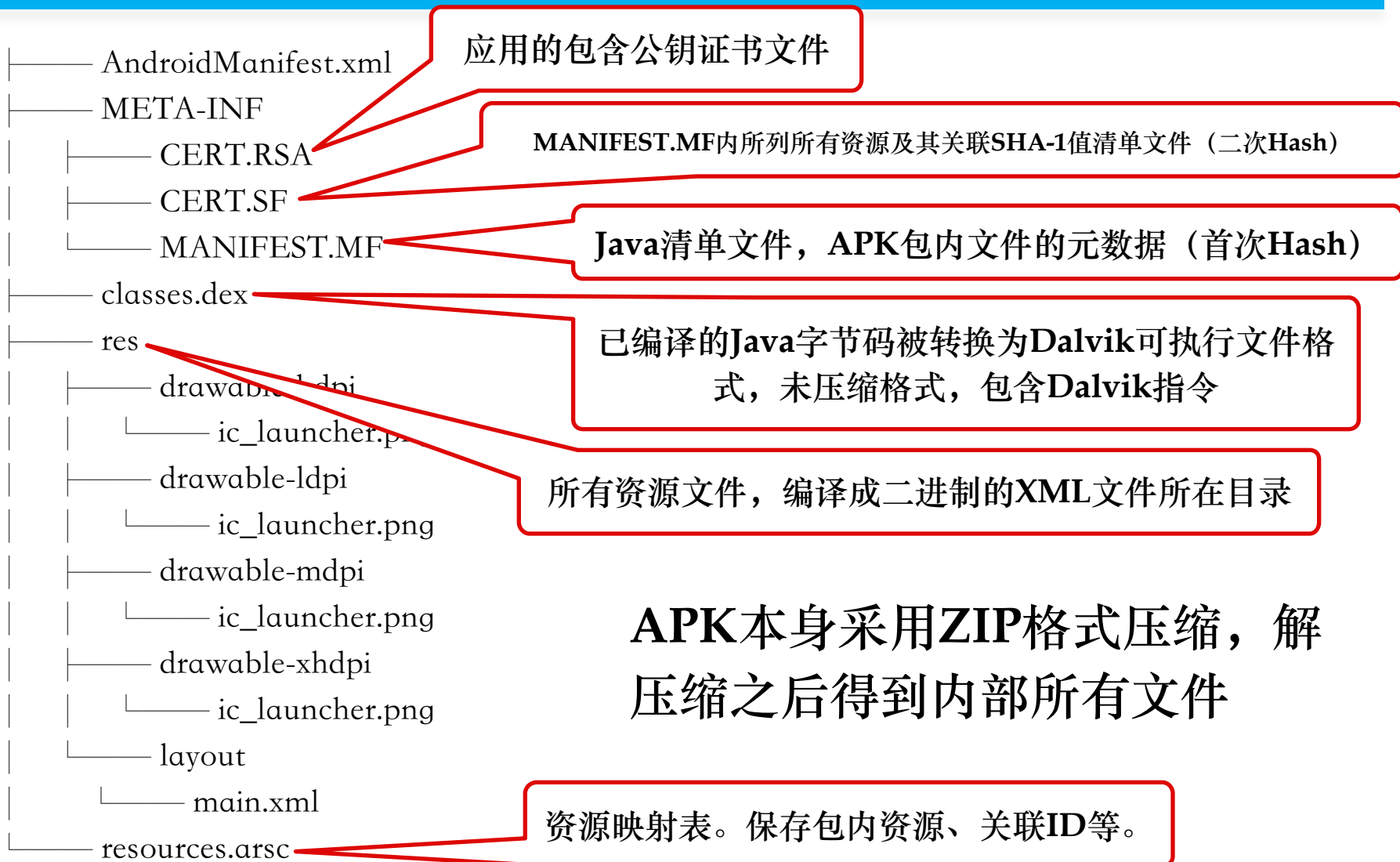


APK反编译工具集

- 反编译瑞士军刀 - apktool
 - APK解压缩，二进制XML解码还原为纯文本
 - 资源修改和重打包
- 反编译Java源代码
 - zip格式解压缩工具，例如: unzip
 - 将dex格式文件转换回jar格式文件 - dex2jar
 - Java字节码反编译工具 - jd-GUI
- Dalvik字节码包文件 (.dex) 打包和解包工具
 - backsmali (解包) VS. smali (打包)



解剖APK——以HelloWorld程序为例



APK本身采用ZIP格式压缩，解压缩之后得到内部所有文件



MANIFEST.MF VS. CERT.SF

```
2. vim META-INF/MANIFEST.MF (Python)

1 Signature-Version: 1.0
2 Created-By: 1.0 (Android)
3 SHA1-Digest-Manifest: kEXh13FgMgn2b11gCtHKED8EVEI=
4
5 Name: res/layout/main.xml
6 SHA1-Digest: F5AuAcTrr8L3NZFdfpTtjRLwaGI=
7
8 Name: res/drawable-xhdpi/ic_launcher.png
9 SHA1-Digest: HJWVdRCXqdgplGGDT6Uzb5hkqQE=
10
11 Name: AndroidManifest.xml
12 SHA1-Digest: 9nbRTM4kA3w0+93CJFE76qxPiGw=
13
14 Name: res/drawable-mdpi/ic_launcher.png
15 SHA1-Digest: 73TZoa+3ApYPkY+wbmr1XiAso58=
16
17 Name: res/drawable-hdpi/ic_launcher.png
18 SHA1-Digest: 0oRDYl4wU0m0bMq8eVLJG09zkTg=
19
20 Name: resources.arsc
21 SHA1-Digest: Y6uD+HGUCKlMt5nJySavdNG4jkg=
22
23 Name: classes.dex
24 SHA1-Digest: I0XjTDSqQwuJlyfL09LyA10MPyA=
25
26 Name: res/drawable-ldpi/ic_launcher.png
27 SHA1-Digest: 2U03QT0JW0jA6L4fx4GV0P+TKag=
28

1 Manifest-Version: 1.0
2 Created-By: 1.0 (Android)
3
4 Name: res/drawable-xhdpi/ic_launcher.png
5 SHA1-Digest: UCzYT6RE8m1+z99KNVBkhnL38jY=
6
7 Name: res/layout/main.xml
8 SHA1-Digest: nFUY65FAqp4tsJ7ZwOQeELCwaJ8=
9
10 Name: AndroidManifest.xml
11 SHA1-Digest: w3Lr3VHXLi6riDu8d7HzT6ON2A=
12
13 Name: res/drawable-mdpi/ic_launcher.png
14 SHA1-Digest: I3MP0NXnINH3Gb4a/IxI+nMF2mw=
15
16 Name: res/drawable-hdpi/ic_launcher.png
17 SHA1-Digest: RQ09RDJf3yWYEKY0avo2ED4Yaz0=
18
19 Name: resources.arsc
20 SHA1-Digest: qDw+u9sDTeCHwlcBxa3fauaZlYg=
21
22 Name: classes.dex
23 SHA1-Digest: eSYcIgpwcrUK0GwWhtV78JyFaP4=
24
25 Name: res/drawable-ldpi/ic_launcher.png
26 SHA1-Digest: CepliFtAgOUV73Bk6BbHeZHAxS=
27

~
~

NORMAL >> META-INF/CERT.SF          << 100% : 28: 0  META-INF/MANIFEST.MF          mf 85% : 23: 1
"META-INF/CERT.SF" [dos] 28L, 725C
```



MANIFEST.MF

- MANIFEST.MF中的资源文件SHA-1值计算方法

```
→ inside_apk openssl dgst -binary -sha1 classes.dex | openssl base64  
eSYcIgpwcrUK0GWwHtV78JyFaP4=  
→ inside_apk cat META-INF/MANIFEST.MF | tail -n 6  
Name: classes.dex  
SHA1-Digest: eSYcIgpwcrUK0GWwHtV78JyFaP4=
```



- SHA1-Digest-Manifest的计算方法

```
→ inside_apk openssl dgst -binary -sha1 META-INF/MANIFEST.MF | openssl base64  
kEXh13FgMgn2b11gCtHKED8EVEI=  
→ inside_apk cat META-INF/CERT.SF  
Signature-Version: 1.0  
Created-By: 1.0 (Android)  
SHA1-Digest-Manifest: kEXh13FgMgn2b11gCtHKED8EVEI=
```




- 清单文件上每个文件的校验和计算方法
 - 由MANIFEST.MF文件中对应的部分先SHA-1再BASE64处理后的结果
 - 注意不能漏了：每个条目结尾的0d0a0d0a

```
→ inside_apk hexdump -C classes.dex.manifest
00000000  4e 61 6d 65 3a 20 63 6c  61 73 73 65 73 2e 64 65  |Name: classes.de|
00000010  78 0d 0a 53 48 41 31 2d  44 69 67 65 73 74 3a 20  |x..SHA1-Digest: |
00000020  65 53 59 63 49 67 70 77  63 72 55 4b 30 47 57 77  |eSYcIgpwcrUK0Gww|
00000030  48 74 56 37 38 4a 79 46  61 50 34 3d 0d 0a 0d 0a  |HtV78JyFaP4=....|
00000040

→ inside_apk openssl dgst -binary -sha1 classes.dex.manifest | openssl base64
I0XjTDSqQwuJlyfL09LyA10MPyA=

→ inside_apk cat META-INF/CERT.SF | tail -n 6
Name: classes.dex
SHA1-Digest: I0XjTDSqQwuJlyfL09LyA10MPyA=
```



CERT.RSA

- 用ASN.1语法解析CERT.RSA
 - `openssl asn1parse -in CERT.RSA -inform DER -i -dump`
- 查看证书信息
 - `openssl pkcs7 -in META-INF/CERT.RSA -inform der -print_certs -text`



APK内其他常见目录和文件

- lib 依赖于特定型号处理器的编译后代码目录
 - armeabi 兼容所有ARM处理器的编译后代码
 - armeabi-v7a 兼容所有ARMv7及以上版本ARM处理器的编译后代码
 - x86 兼容所有x86架构处理器的编译后代码
 - mips 兼容所有MIPS架构处理器的编译后代码
- assets **AssetManger**管理的资源文件目录
- AndroidManifest.xml 二进制XML格式，可以使用**AXMLPrinter2.jar**转换为人类可读的文本XML格式



以APKTOOL为例

中国传媒大学



APKTool的安装 (Windows)

- Apktool 1.x (Versions prior to 1.5.2)
 - Java 1.6+
- 解压缩文件之后把以下3个文件放在同一个目录
 - aapt.exe apktool.bat apktool.jar
- 将包含上述3个文件的目录全路径添加到系统环境变量PATH



APKTool基本使用

- 反汇编
 - `apktool d [OPTS] <file.apk> [<dir>]`
 - 对于某些经过加壳或安全加固的APK反汇编功能会失效
- 重打包
 - `apktool b [OPTS] [<app_path>] [<out_file>]`
 - Windows系统上不支持对中文目录名重打包



程序入口——主Activity

- 注册了名为
android.intent.action.MAIN
的Action对应的Activity

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2     package="com.droider.downloadmanager"
3     android:versionCode="1"
4     android:versionName="1.0" >
5
6     <uses-sdk
7         android:minSdkVersion="8"
8         android:targetSdkVersion="15" />
9     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
10    <uses-permission android:name="android.permission.INTERNET"/>
11
12    <application
13        android:icon="@drawable/ic_launcher"
14        android:label="@string/app_name"
15        android:theme="@style/AppTheme" >
16        <activity
17            android:name=".MainActivity"
18            android:label="@string/title_activity_main" >
19            <intent-filter>
20                <action android:name="android.intent.action.MAIN" />
21
22                <category android:name="android.intent.category.LAUNCHER" />
23            </intent-filter>
24        </activity>
25        <receiver android:name=".DownloadReceiver">
26            <intent-filter>
27                <action android:name="com.droider.download"></action>
28            </intent-filter>
29        </receiver>
30    </application>
31
32 </manifest>
```



Dalvik虚拟机——DVM

- 采用Apache 2.0授权协议
- Android 5.0以前的Java代码核心虚拟机引擎
- .dex格式是专为Dalvik设计的一种压缩格式
- Android 2.2开始，DVM支持JIT
 - just-in-time compiler
- DVM占用更少用户态内存
- 系统架构设计为嵌入式和智能设备优化，代码执行性能比JVM好



DVM VS. JVM

- JVM运行的是Java字节码，DVM运行的是Dalvik字节码
- JVM基于堆栈架构，DVM基于寄存器架构
- DVM可执行文件体积更小



- DVM的一种新的运行时模式
 - Ahead-of-time (AOT) compilation
 - Android 4.4开始系统支持ART模式（设置-开发者选项）
 - Android 5.0开始系统全局默认启用ART模式
 - 应用安装时，ART使用设备上的dex2oat工具，将DEX文件转换为目标机器的ART专用字节码
 - 目前对于大量APP可能会存在兼容性问题



- ART 模式发行的应用在用户安装时就进行预编译操作，将原本在程序运行中的编译动作提前到应用安装时
 - 应用的运行效率更高，牺牲掉的是更多的存储空间占用
 - 应用安装时间变长
 - 应用安装后无法备份



定位关键代码的方法

- 信息反馈法
- 特征函数法
- 顺序查看法
- 代码注入法（动态调试和分析）
- 栈跟踪法（动态调试和分析）
- Method Profiling（动态调试和分析）



实验一：使用信息反馈法破解认证

实验



实验二：特征函数法

实验



内容提纲

- 软件逆向分析基础之静态分析
- 软件逆向分析基础之动态分析
- 通信协议逆向分析基础



Java代码与反汇编Smali代码

```
→ Desktop adb -s emulator-5554 logcat | grep 'tag-here'
V/tag-here( 2175): message here: hello world!
V/tag-here( 2225): message here: hello world!
V/tag-here( 2225): message here: hello world!
V/tag-here( 2225): message here: hello world!
```

```
9 public class MainActivity extends Activity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_main);
15         Log.v("tag-here", "message here: hello world!");
16     }
17 }
```

```
32 .line 15
33 const-string v0, "tag-here"
34
35 const-string v1, "message here: hello world!"
36
37 invoke-static {v0, v1}, Landroid/util/Log;->v(Ljava/lang/String;Ljava/lang/String;)I
```




实验四：代码注入法

实验



内容提纲

- 软件逆向分析基础之静态分析
- 软件逆向分析基础之动态分析
- 通信协议逆向分析基础



通信数据监听方法

- 模拟器

```
→ MobileInternetSecurity android list avd
Available Android Virtual Devices:
  Name: nexus4
  Device: Nexus 4 (Google)
  Path: /Users/huangwei/.android/avd/nexus4.avd
  Target: Android 4.1.2 (API level 16)
  Tag/ABI: default/x86
  Skin: 768x1280
  Sdcard: 512M
-----
  Name: nexus5
  Device: Nexus 5 (Google)
  Path: /Users/huangwei/.android/avd/nexus5.avd
  Target: Google APIs (x86 System Image) (Google Inc.)
  Based on Android 4.4.2 (API level 19)
  Tag/ABI: default/x86
  Skin: 1080x1920
  Sdcard: 512M
```

```
→ MobileInternetSecurity emulator -tcpdump emulator-20141212.pcap -avd nexus4
```

```
→ MobileInternetSecurity wireshark emulator-20141212.pcap
```



通信数据监听方法

- 真机和模拟器通用(需要root手机)
 - 从<http://www.tcpdump.org/> 下载libpcap和tcpdump源代码，交叉编译出tcpdump的目标CPU架构可执行程序(静态链接，真机普遍使用ARM，模拟器可以使用x86或ARM)
 - 将该tcpdump上传到真机的/data/local目录
 - 先上传到SD卡（例如： /mnt/sdcard），再su后cp到/data/local

```
1|shell@android:/data/local # ./tcpdump --list-interfaces
1.wlan0 [Up, Running]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.p2p0 [Up]
5.nflog (Linux netfilter log (NFLOG) interface)
6.nfqueue (Linux netfilter queue (NFQUEUE) interface)
shell@android:/data/local # ./tcpdump -i wlan0 -w /mnt/sdcard/2014-12-12.pcap
tcpdump: listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C5 packets captured
6 packets received by filter
0 packets dropped by kernel
```



通信数据监听方法——HTTP协议

- PC机上开启HTTP代理软件
 - charles
 - burpsuite
- Android手机上开启无线热点的代理设置-高级选项
- 手机无需ROOT



通信数据监听方法——自建SoftAP

- 参见第3章课件——Evil Twins一节说明
- 具体操作方法见该节实验的补充材料
- 手机无需ROOT



通信数据监听的旁门左道——『投毒』

- DHCP服务器指定内网任意可抓包主机作为无线网络的默认网关
- DHCP服务器指定内网DNS服务器，在DNS解析记录里配置所有解析记录指向开启透明正向HTTP代理的可抓包主机
 - dnsspoof
 - Android系统里的/etc/hosts文件
 - 需root权限访问和修改



通信协议逆向的核心

- 熟练使用网络协议分析工具
 - wireshark的显示过滤规则和辅助工具
- 结合软件逆向分析结果
 - 源代码级别
 - 反汇编代码级别
 - 注入代码并输出关键变量运行时状态值
 - adb logcat



参考资料

- 丰生强, Android软件安全与逆向分析
- Reverse Engineering Android: Disassembling Hello World



延伸阅读

- [Monitoring Android Network Traffic Part I: Installing the Toolchain](#)
- [Monitoring Android Network Traffic Part II: Cross Compiling TCPDUMP](#)
- [Monitoring Android Network Traffic Part III: Installing & Executing TCPDUMP](#)
- [Monitoring Android Network Traffic Part IV: Forwarding To Wireshark](#)
- [如何在Android手机上实现抓包](#)