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

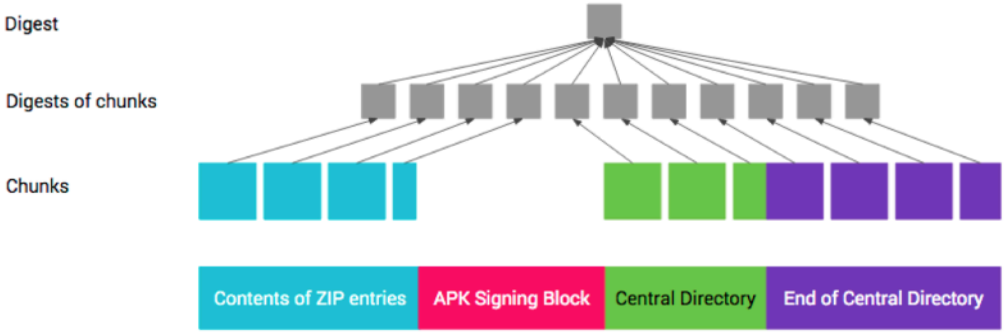
- 静态分析是指在不运行代码的情况下，采用词法分析、语法分析等各种技术手段对程序文件进行扫描从而生成程序的反汇编代码，然后阅读反汇编代码来掌握程序功能和理解程序原理的一种技术
- 反汇编（或反编译）代码生成工具简称反汇编（反编译）工具

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）

重要说明

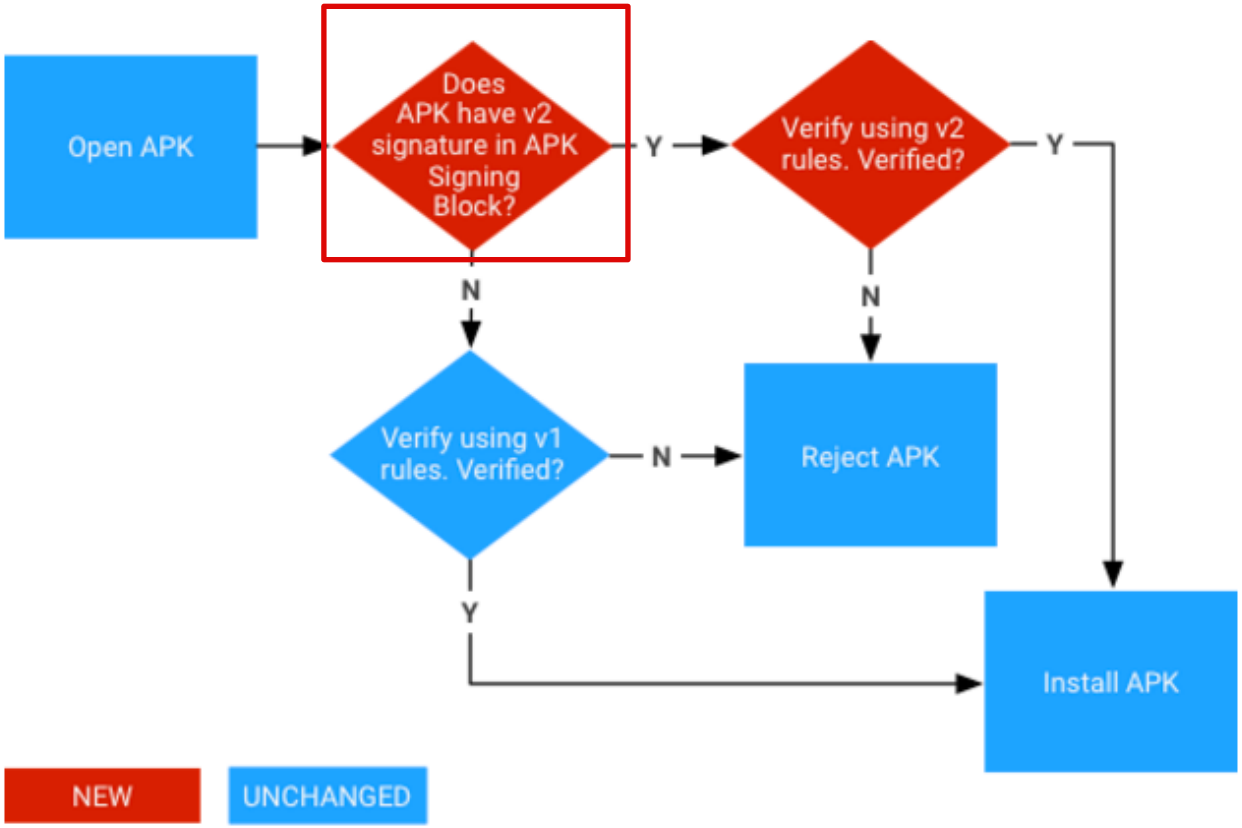
- Android 7.0开始使用APK Signature Scheme v2



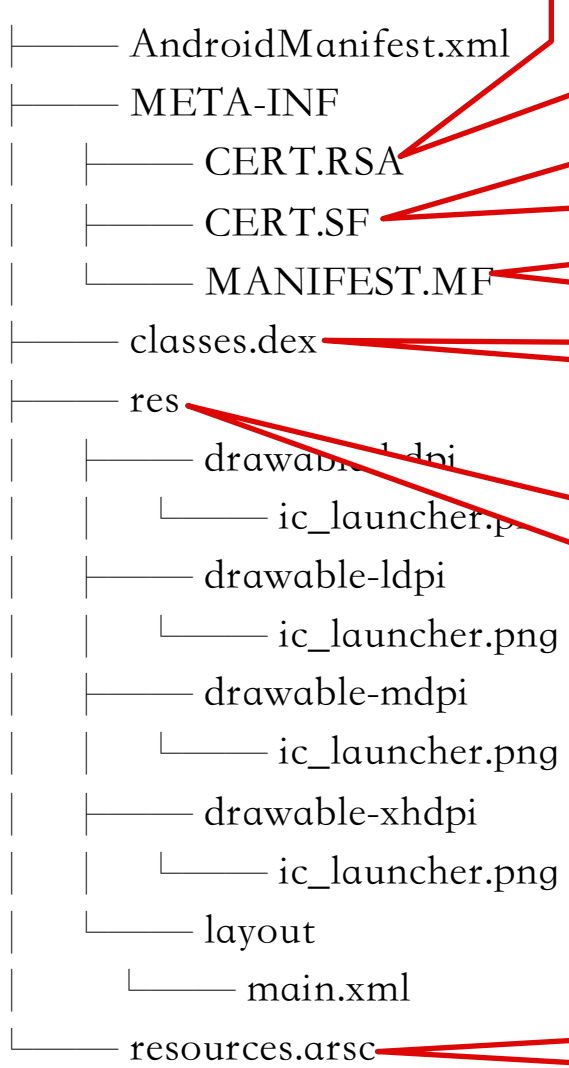
- 在APK Signature Scheme v1的基础之上
  - 增加对整个APK文件的签名
  - 提高了签名验证的速度和强度
- APK Signature Scheme v1直接使用JAR文件签名算法和工具

重要说明

- APK Signature Scheme v2的验证过程



解剖APK——以HelloWorld程序为例



应用的包含公钥证书文件

MANIFEST.MF内所列所有资源及其关联SHA-1值清单文件（二次Hash）

Java清单文件，APK包内文件的元数据（首次Hash）

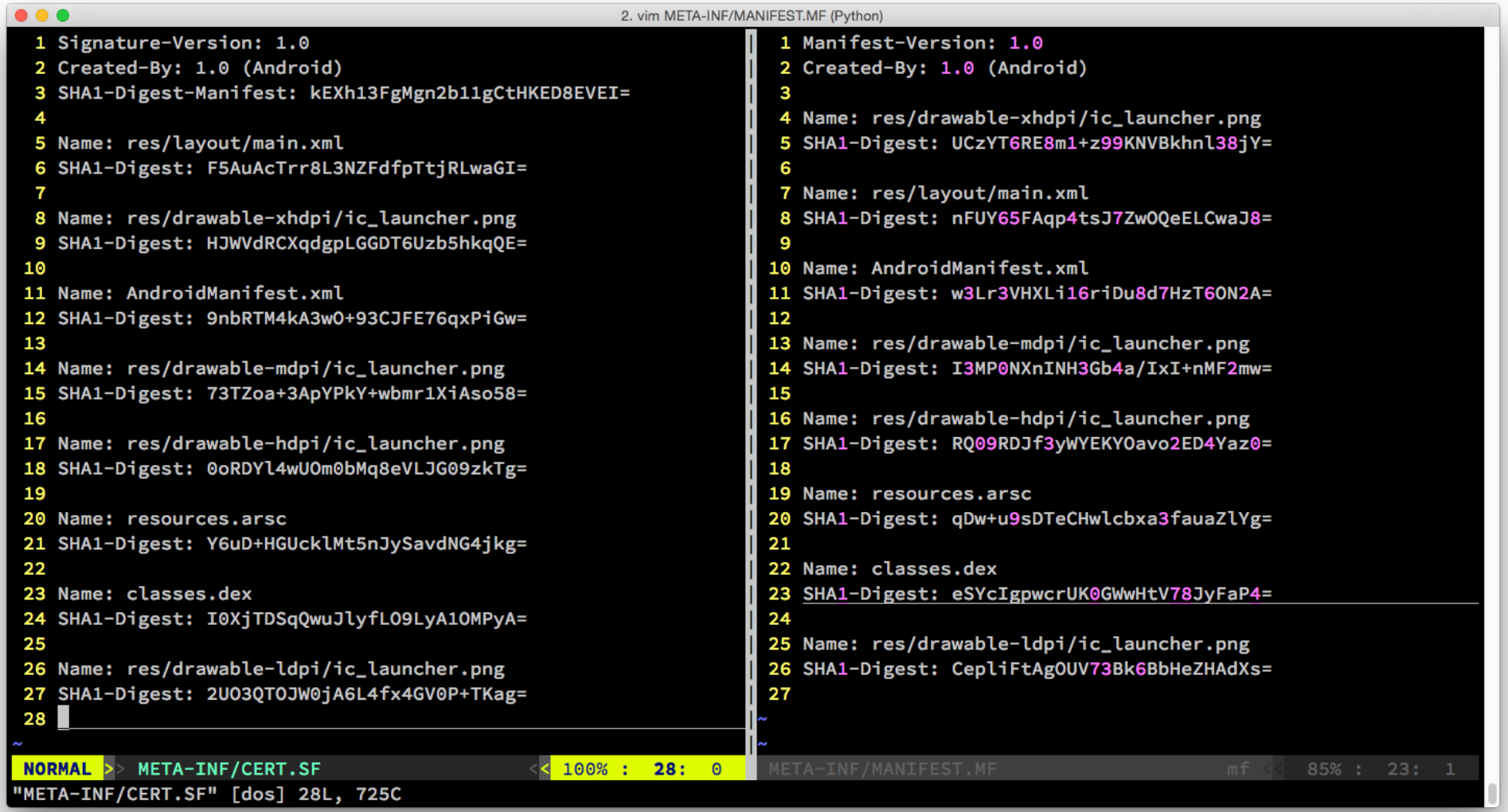
已编译的Java字节码被转换为Dalvik可执行文件格式，未压缩格式，包含Dalvik指令

所有资源文件，编译成二进制的XML文件所在目录

APK本身采用ZIP格式压缩，解压缩之后得到内部所有文件

资源映射表。保存包内资源、关联ID等。

MANIFEST.MF VS. CERT.SF



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=
```

CERT.SF

- 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=
```

CERT.SF

- 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
I0XjTDSqQwuJlyfLO9LyA1OMPyA=
➜  inside_apk  cat META-INF/CERT.SF | tail -n 6
Name: classes.dex
SHA1-Digest: I0XjTDSqQwuJlyfLO9LyA1OMPyA=
```

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

关于jarsigner（man jarsigner）

- 支持的签名算法
  - —Digital Signature Algorithm (DSA) with the SHA1 digest algorithm
  - —RSA algorithm with the SHA256 digest algorithm
  - —Elliptic Curve (EC) cryptography algorithm with the SHA256 with Elliptic Curve Digital Signature Algorithm (ECDSA)

```
root@KaliRolling:/media/sf_Downloads/app-debug-v2/original/META-INF# head CERT.SF
Signature-Version: 1.0
X-Android-APK-Signed: 2
SHA-256-Digest-Manifest: P8t1GVNZ42LmapcjaTsiu/6ygULJydaZPF9YEFbHNto=
Created-By: 1.0 (Android)

Name: res/drawable-xhdpi-v4/abc_scrubber_control_to_pressed_mtrl_005.p
 ng
SHA-256-Digest: KZU72AGd2Wt3ATKbmiA8/2NGBRlUm5zKml63T5sbCTA=
```

基于JAR签名策略v1的APK文件完整性保护

- <signer>.(RSA|DSA|EC) →<signer>.SF → MANIFEST.MF →压缩包内每个文件内容（完整性）

- jarsigner -verify -certs -verbose -keystore <path_to_keystore> <path_to_apk>

```
smk     9490 Thu Dec 15 09:13:20 CST 2016 res/mipmap-xxxhdpi-v4/ic_launcher.png

      X.509, CN=Wei Huang, OU=SEC, O=CUC, L=Chaoyang, ST=Beijing, C=CN (misdemo)
      [certificate is valid from 12/8/16 2:49 PM to 12/2/41 2:49 PM]

smk   217048 Thu Dec 15 09:13:20 CST 2016 resources.arsc

      X.509, CN=Wei Huang, OU=SEC, O=CUC, L=Chaoyang, ST=Beijing, C=CN (misdemo)
      [certificate is valid from 12/8/16 2:49 PM to 12/2/41 2:49 PM]


  s = signature was verified
  m = entry is listed in manifest
  k = at least one certificate was found in keystore
  i = at least one certificate was found in identity scope

jar verified.
```

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

APKTool

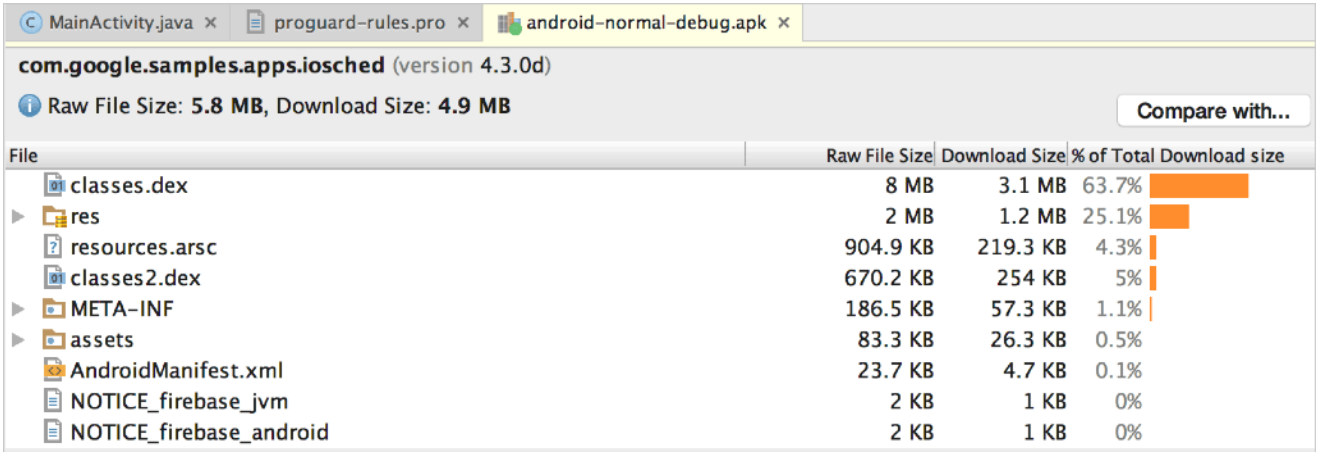
- 特性
  - 反汇编恢复原始二进制资源(包括 resources.arsc, classes.dex, 9.png. 和 XMLs)
  - 重打包解码出的资源到APK/JAR
  - 组织和处理 依赖框架资源的APK
  - Smali 调试 (Removed in 2.1.0 in favor of IdeaSmali)
  - 辅助重复任务

APKTool基本使用

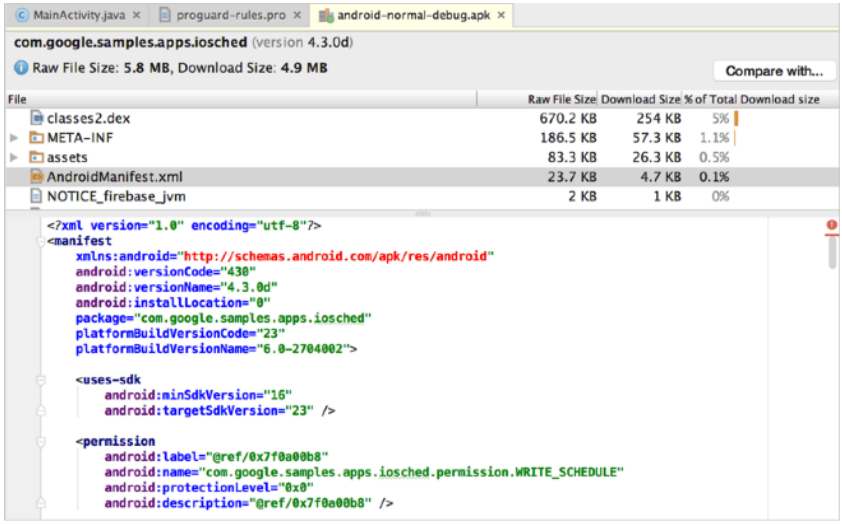
- 反汇编
  - apktool d [OPTS] <file.apk> [<dir>]
  - 对于某些经过加壳或安全加固的APK反汇编功能会失效
- 重打包
  - apktool b [OPTS] [<app_path>] [<out_file>]
  - Windows系统上不支持对中文目录名重打包

Android Studio的APK Analyzer

- 查看文件内部结构和资源占用信息

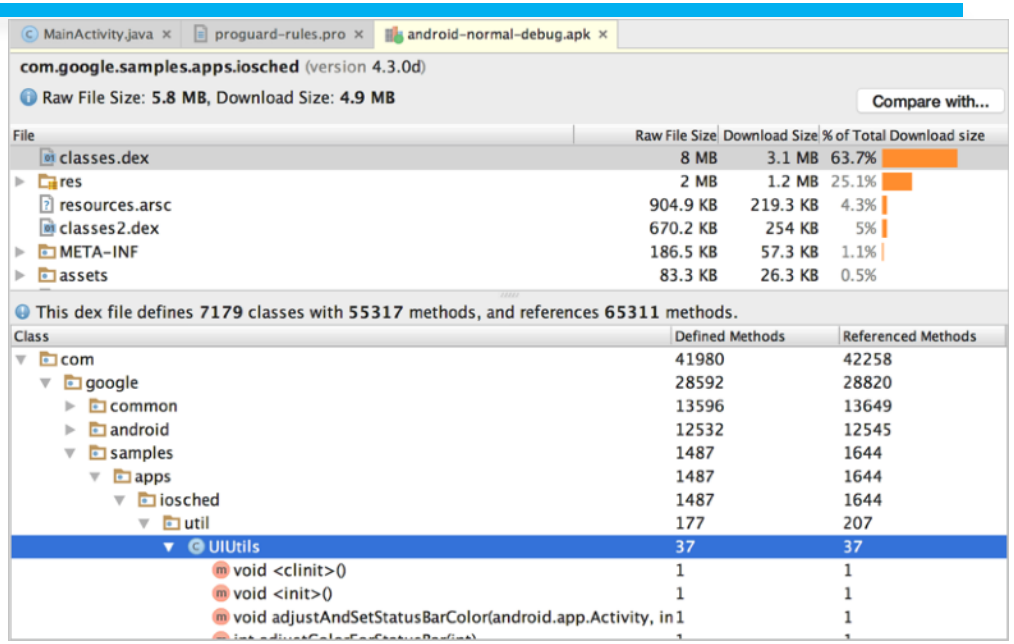


- 查看AndroidManifest.xml

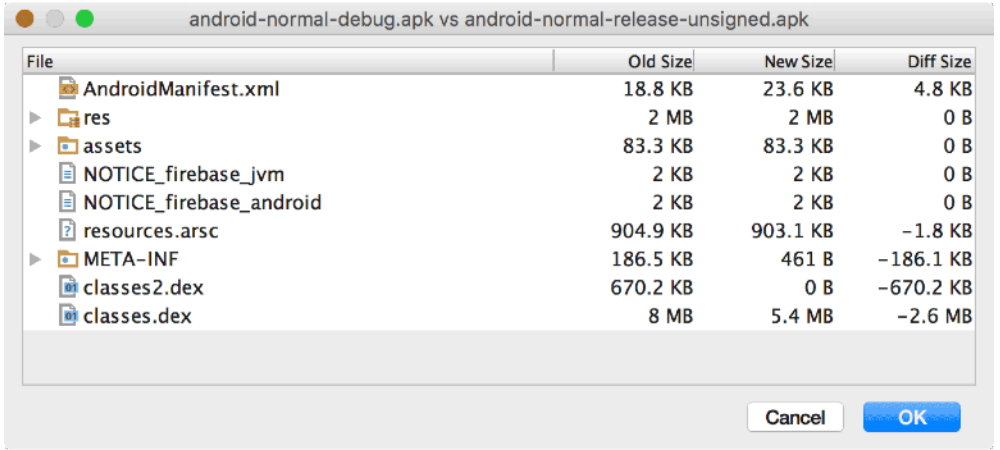


Android Studio的APK Analyzer

- 查看（反汇编）DEX



- APK比较



程序入口——主Activity

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

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.droider.downloadmanager"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".DownloadReceiver">
            <intent-filter>
                <action android:name="com.droider.download"></action>
            </intent-filter>
        </receiver>
    </application>

</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

ref: http://en.wikipedia.org/wiki/Dalvik_(software)

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

ART

- 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

ref: http://source.android.com/devices/tech/dalvik/art.html

ART

- 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时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

ref: http://source.android.com/devices/tech/dalvik/art.html

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）

信息反馈法

- 让目标程序运行起来，然后根据程序运行时给出的反馈信息作为突破口寻找关键代码
  - 例如“无效用户名或注册码”，这就是一个很典型的程序反馈输出信息
  - 带着这个关键词去程序代码或程序反汇编之后的代码中进行关键词查找，快速定位关键代码和上下文
  - 如果程序使用了资源字符串则还需要根据关键词对应的资源ID继续在反汇编代码中进行查找

特征函数法

- 与信息反馈法类似
  - 无论程序给我们“反馈”了什么信息，“反馈”方式的实现都是通过调用特定系统API完成
  - 根据系统API关键词进行检索查找，定位关键代码和上下文
    - 例如：Toast.MakeText().show()方法是常用的“吐司”提示实现手段

顺序查看法

- 从软件的入口/启动代码开始，逐行向下分析（在大脑中模拟代码运行），掌握软件的执行流程
- 不是孤立的方法，可以和其他静态、动态分析方法相结合

  —从“疑似”关键代码处开始顺序查看和分析

代码注入法

- “注入”在程序分析领域又被称为插桩 (Instrumentation)
  - 静态插桩：发生在程序执行前，重写目标代码或执行代码阶段
  - 动态插桩：发生在运行阶段。可以通过程序或者另外进程将分析代码“注入”到客户进程
- 例如：手动修改apk文件的汇编代码，加入Log输出，配合LogCat查看程序执行到特定点时的（变量）状态数据
  - 解密程序数据时的惯用手段

实验一：使用信息反馈法破解认证

实验

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      }
```

```
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
```


ref: http://www.kandroid.org/online-pdk/guide/tcpdump.html

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』

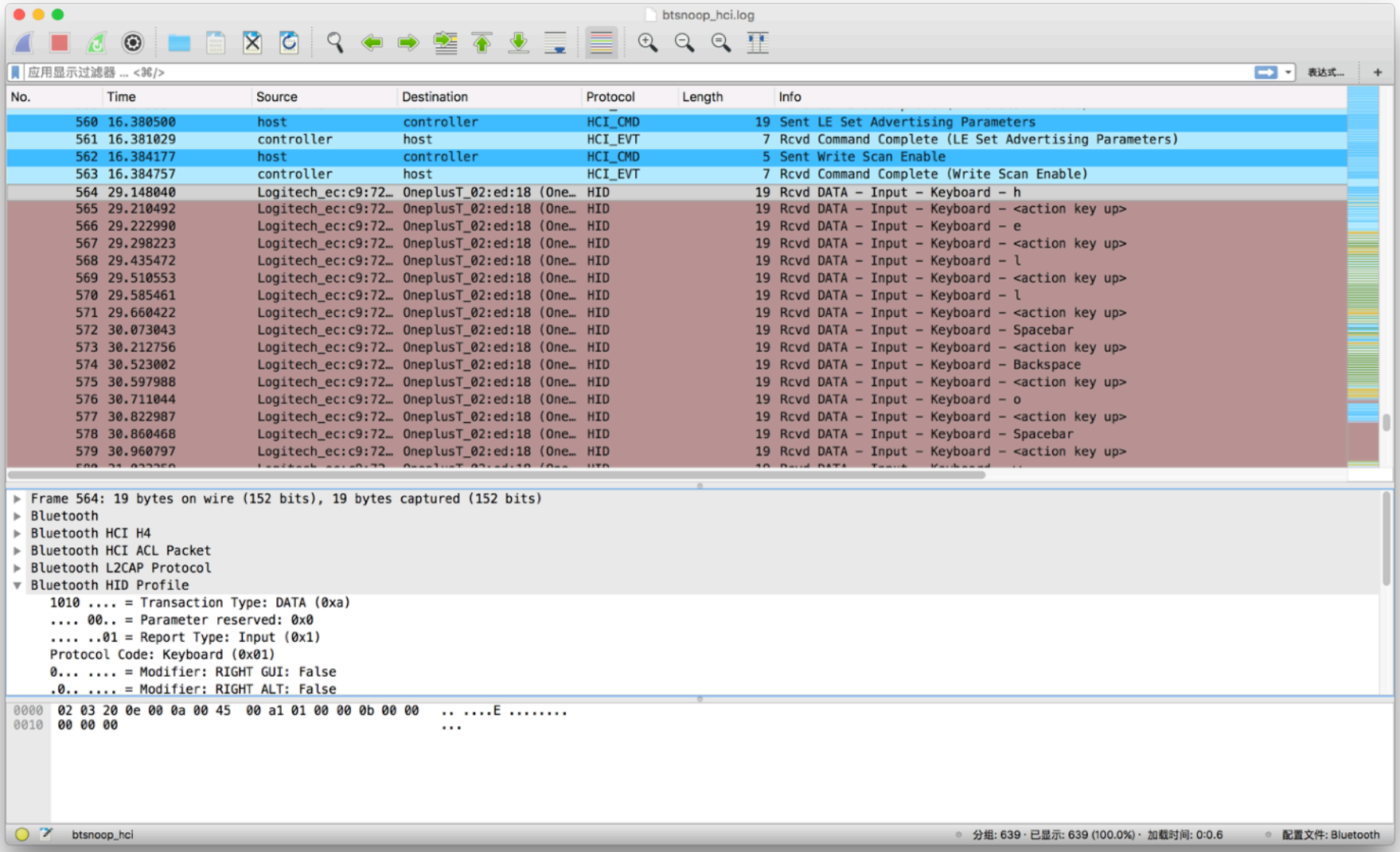
- DHCP服务器指定内网任意可抓包主机作为无线网络的默认网关
- DHCP服务器指定内网DNS服务器，在DNS解析记录里配置所有解析记录指向开启透明正向HTTP代理的可抓包主机
  - dnsspoof
  - Android系统里的/etc/hosts文件
    - 需root权限访问和修改

监听蓝牙协议通信数据

- Android 4.4+可以通过开发者选项中的“启用蓝牙HCI信息收集日志”功能捕获蓝牙设备与手机之间的点对点通信数据和手机的蓝牙广播数据
  - 默认抓包结果写入到手机上的/sdcard/btsnoop_hci.log
  - 可以用wireshark读取并解析
- 无线监听其他非本手机蓝牙点对点通信数据需要专门的硬件设备

https://www.nowsecure.com/blog/2014/02/07/bluetooth-packet-capture-on-android-4-4/

蓝牙通信实例——无线键盘


btsnoop_hci.log
应用显示过滤器 ... <⌘/>
表达式...
No. Time Source Destination Protocol Length Info
560 16.380500 host controller HCI_CMD 19 Sent LE Set Advertising Parameters
561 16.381029 controller host HCI_EVT 7 Rcvd Command Complete (LE Set Advertising Parameters)
562 16.384177 host controller HCI_CMD 5 Sent Write Scan Enable
563 16.384757 controller host HCI_EVT 7 Rcvd Command Complete (Write Scan Enable)
564 29.148040 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - h
565 29.210492 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - <action key up>
566 29.222990 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - e
567 29.298223 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - <action key up>
568 29.435472 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - l
569 29.510553 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - <action key up>
570 29.585461 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - l
571 29.660422 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - <action key up>
572 30.073043 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - Spacebar
573 30.212756 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - <action key up>
574 30.523002 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - Backspace
575 30.597988 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - <action key up>
576 30.711044 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - o
577 30.822987 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - <action key up>
578 30.860468 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - Spacebar
579 30.960797 Logitech_ec:c9:72… OneplusT_02:ed:18 (One… HID 19 Rcvd DATA - Input - Keyboard - <action key up>
Frame 564: 19 bytes on wire (152 bits), 19 bytes captured (152 bits)
Bluetooth
Bluetooth HCI H4
Bluetooth HCI ACL Packet
Bluetooth L2CAP Protocol
Bluetooth HID Profile
1010 .... = Transaction Type: DATA (0xa)
.... 00.. = Parameter reserved: 0x0
.... ..01 = Report Type: Input (0x1)
Protocol Code: Keyboard (0x01)
0... .... = Modifier: RIGHT GUI: False
.0.. .... = Modifier: RIGHT ALT: False
0000 02 03 20 0e 00 0a 00 45 00 a1 01 00 00 0b 00 00 .. ....E ....
0010 00 00 00 ...
btsnoop_hci
分组: 639 · 已显示: 639 (100.0%) · 加载时间: 0:0.6
配置文件: Bluetooth

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