android 竞品分析工具对比

原创

MrCheChe 于 2018-03-02 10:56:43 发布 3470 收藏 1
分类专栏： 项目经验总结 文章标签： 分析工具 逆向

本文链接：https://blog.csdn.net/ZLMrche/article/details/79420732

最近一段时间因为琐事（有公有私）太多，加上搬家后，家里断网了一周，虽然有很多东西想写，但却迟迟没有动手。好在目前基本事情都搞完，又可以愉快的写东西了，所以，接下来博客更新的频率将会有所提高。写博客的好处是很多的哈，一方面总结可以分享也方便自己回顾，另一方面是当作者对知识点理解可能存在局限（或者错误）时，读者也会帮你指出。

前言

本文整理了一些自己在开发过程中经常会用到的竞品分析工具，这些工具可以帮助分析竞品。让我们得以了解竞品相应的一些技术信息，例如：代码质量、某种业务的实现方式、用了什么第三方库等。除此之外，也有一些高端玩家会玩起 HOOK，更有甚者是通过修改代码然后进行二次打包。当然这些损害开发者利益的事情，是不值得提倡的。但如果只是出于学习的目的，我是十分建议多折腾的。

提前声明：

- 本文只对工具做简要功能介绍，要求面面俱到讲解每个工具使用，本人表示能力有限啊；
- 下文所介绍的工具，都会附上这些工具的官方地址以及相应的使用教程链接（如果有）；
- 有童鞋对下文提到的工具已经用得出神入化，欢迎写成文章，可以的话，也欢迎给个链接让我补充进本文，顺带学习一下；
- 本文所有提到的工具只做分析学习使用，**请不要拿去做损害他人利益的事情**；

Apk 内部结构

为了方便介绍工具，需要先简单科普一下 Apk 的内部结构，已经很熟悉的童鞋可以忽略此章节。需要注意的是，这里介绍的 Apk 结构并不包含加固的情况，虽然很多厂家推出了加固服务用于对抗反编译，但是加固也有诸多的问题存在，另外基本上分析的大厂应用都没有发现有加固的，可能也是考虑到加固后安装包存在的诸多问题吧。

直接使用 Android Studio 创建一个 HelloWorld 的 Moudle，然后打个 release 的 Apk 安装包，并修改后缀 apk 为 zip 后进行解压，可以看到下面一个标准的结构：

- META-INF： 存放签名文件签名信息的目录，用于系统签名校验
- res： 存放资源文件的目录，包含项目中的 xml 和 图片资源等
- AndroidManifest.xml： Android项目中的配置文件
- classes.dex： 由Java产生的字节码文件打包生成为虚拟机可以解读的字节码文件，所有的源码都在其中
- resources.arsc： 资源文件的ID索引表，如：layout、drawable、mipmap都会在R文件生成相应的ID资源
- 其他目录：开发者自行添加的目录，如：存放资源的 asserts 、存放依赖包的 lib 目录等

上面介绍完了一个最基本的 Apk 解压后的目录结构，下面直接拿微信作为示例，看看大厂应用的结构是怎样的：

我们可以看到微信除包含了上面提到的，还有 asserts、lib、r 这三个自行添加的目录，至于前两个目录是干嘛的上面已经提到，r 目录里面主要存放了一些 svg 和 xml 文件，有兴趣可以自行试试。如果要问为什么微信有3个dex文件的话，只能说它超了 Android 系统设定 65k 方法的限制，所以有多个dex包。

OK，关于 Apk 的目录结构介绍基本到此，这有助于我们去理解下面即将要介绍的工具！

Apktool

工具名：Apktool

官网：http://ibotpeaches.github.io/Apktool/

源码：

- Github地址：https://github.com/iBotPeaches/Apktool
- Bitbucket地址：https://bitbucket.org/iBotPeaches/apktool/downloads

配置文档：http://ibotpeaches.github.io/Apktool/install/ ， 教你如何配置 Apktool 的使用环境

使用文档：http://ibotpeaches.github.io/Apktool/documentation/ ， 教你各种基本的命令的使用

功能：

- 还原 Apk 中所包含的 resources.arsc, classes.dex, 9.png 和 xml 等文件；
- 对 Apk 进行重新（二次）打包；
- 反编译依赖于 Framework 的 apk 文件；
- 运行调试 Smali 文件；

下面以微信的Apk为例简要介绍每个功能。

1、还原 Apk 中所包含的 resources.arsc, classes.dex, 9.png 和 xml 等文件。通过DOS窗口去到Apktool目录下，执行如下命令

```
apktool d XXX.apk
```

可以看到上图的反编译过程中打印信息日志，以及反编译成功后得到的微信目录。反编译成功后，进入微信目录可以看到如下的目录结构

2、对 Apk 进行重新（二次）打包。通过DOS窗口去到Apktool目录下，执行如下命令：

```
apktool b 目录名
```

本想对微信做重新打包，结果发现微信果然做了处理了，打包不成功，大厂考虑得好周到。于是狡猾的拿了另一个对象试手，果然一下子搞定了

可以去到反编译生成目录下的dist目录看到我们重新打包好后的 Apk

这个功能主要有两个应用场景，一个是对应用进行本地化处理，例如反编译后对应用进行汉化。另一个是无耻的对应用进行广告植入，重新打包发放市场并以此牟利（非常强烈的不赞同大家去这么做）。

3、反编译依赖于 Framework 的 apk 文件。这个功能我并没有用过，但是也做了大致的了解。举个例子（摘自网络的栗子）：一般来说，你在使用 Apktool 进行反编译前不需要做其他的事情，然而由于有的厂商，如HTC，三星等，他们定制了 Framework 文件并且在他们的系统应用中使用了这些文件，这时，为了能正常的反编译这些Apk文件，你就必须从你的设备中拷贝出framework文件并且安装到apktool中。对这个功能感兴趣的童鞋，可以直接查看这篇文章：http://bbs.gfan.com/android-2657915-1-1.html

4、运行调试 Smali 文件。首先需要解释一下什么是Smali，它是 Android 虚拟机所使用的一种机器语言，差不多可以理解成汇编语言那样子。利用 Apktool 反编译后，classes.dex 会被翻译成 Smali 源代码

去到下面随便找个目录打开，可以看到都是后缀为 smali 的文件，

随便找个文件打开你会发现， smali 的语法还是比较好理解的

有兴趣想要了解 smali 语法的，可以看看这位作者写的文章：http://blog.isming.me/2015/01/14/android-decompile-smali/。看到这里你可能会想问，了解 smali 的语法有什么用呢？哈，非常简单粗暴，如果懂语法，就能读懂人家的应用里面的业务逻辑，还可以加以修改等。 再加上 Apktool 提供的调试 Smali 功能，可以玩得很爽。但是在 2.0.3 的版本上，作者已经宣布废弃此功能，到了 2.1 时此功能已经完全移除掉了。这也并不意味这我们无法玩转 smali 文件。感兴趣的童鞋可以直接玩耍下面这个 smali 调试插件：

SmaliIdea：https://github.com/JesusFreke/smali/wiki/smalidea

反正我一看到 SmaliIdea 是震惊了。目前 SmaliIdea 最新的版本是 0.03，喜欢玩耍的童鞋可以到下面的下载地址中：

> https://bitbucket.org/JesusFreke/smali/downloads

找到最新版的 SmaliIdea 进行安装使用

对了，这个插件是基于 IntelliJ IDEA / Android Studio 的，还在 Eclipse 下撸码的童鞋可以忽略。

使用 Apktool 需要注意还需要注意以下几点：

1、下载完 Apktool 后，记得要把 jar 文件改成 apktool.jar，与 apktool.bat 同名；

2、如果之前使用过旧版的 apktool ，记得到自己对应的用户目录下将 1.apk 文件删除，否则会造成反编译失败；

Clock 是我在 Windows 下的用户目录，其他童鞋自己换成自己对应的用户目录即可。

3、不要拿去使用 Apktool 去干违法盗版的事情，这也是工具的作者特别强调的；

dex2jar

工具名：dex2jar

官网：https://sourceforge.net/projects/dex2jar/

源码：

- Github地址：https://github.com/pxb1988/dex2jar
- Bitbucket地址：https://bitbucket.org/pxb1988/dex2jar

配置文档：很简单，去官网链接直接下载到本地即可

使用文档：同样比较简单，直接看下面示例，或者 Google 一下

功能：

- 将 dex 文件还原成为 jar 文件；
- 将 dex 文件还原成为 smali 文件；
- 其他一些命令行很少用，喜欢折腾的童鞋，同样自行摸索；

继续拿微信开刀，解压 Apk 去除一个 dex 包，对其进行如下命令行操作

```
d2j-dex2jar.bat dex文件名
```

可以看到生成一个对应的 jar 包，这样就可以结合下一个工具很方便的查看源代码了。如果想要将 dex 转换成 smali ，同样可以执行如下操作：

```
d2j-dex2smali.bat dex文件名
```

classes-out 目录中就可以看到由 dex 包生成的所有 smali 文件了。想必大家也留意到所有的文件都存在同名但是后缀不同的两份

bat 后缀的是 Windows 用户可以调用的批处理文件，sh 后缀的则是 Linux 用户的 Shell 脚本。根据自己的环境去选择用 bat 还是 sh 即可。

jd-gui

工具名：jd-gui

官网：http://jd.benow.ca/

源码：https://github.com/java-decompiler/jd-gui

功能：

- 查看 dex 文件还原成为 jar 文件代码；
- 将 jar 文件中所有的 class 文件转换成为 java 文件；

使用很方便，下载后，直接把用 dex2jar 生成的微信 jar 包拖曳进去即可查看到对应的源代码。

也可以去到工具栏选择 File -> Save 或者 File -> Save All Sources 将单个 class 文件或者整个 jar 包保存成为 java 文件。可惜的是，这个工具的作者从 2015 年开始就停止了维护，所以迟早寻求一个替代品。网上有童鞋说，jadx 不错，可以取代 jd-gui 。我自己还没玩过，童鞋们可以自行试试：

jadx：https://github.com/skylot/jadx

enjarify

工具名：enjarify

源码：https://github.com/google/enjarify

配置文档：下载源码到本地，并安装 Python3 的环境，同时配置好环境变量

使用文档：可以参考上面的源码地址中的说明，或者乌云平台的文章
http://wiki.wooyun.org/android:tools:enjarify

功能：可直接将 apk 文件还原成为 jar 文件，也可以和 dex2jar 一样，直接操作某个dex

这是 Google 出品的一个逆向分析工具，从反编译成 jar 包的流程来说，要比 dex2jar 方便得多。下载完工具到本地并配置好 Python 环境后，需要修改 enjarify 目录下的 enjarify.bat 文件，将 python3 改为 python

再通过DOS窗口去到其目录下，运行如下命令：

```
enjarify.bat XXX.apk
```

这里同样需要注意，一定要将 apk 拷贝到和 enjarify.bat 同级目录，如果使用 apk 的全目录的话，会报错。

enjarify 相较于 dex2jar 做了更多的优化处理，去避免在反编译得到 jar 包的过程中出错。虽然我在使用 dex2jar 的过程中还没遇到过出错的情况，但是在使用 enjarify 时，确实感受到相对更加方便的地方。例如：可以直接对apk进行操作，而不需要先解压提取classes.dex，特别是针对那些本来就不止一个dex包的大型应用。而 enjarfy 也并非万能的，它目前存在的一些限制，如暂时还无法逆向获得源文件的属性、行数和注释等。当然，相信 Google 的工程师也会尽力去优化改善 enjarfy 的功能，还是很值得期待的。

Procyon

工具名：Procyon

源码：https://bitbucket.org/mstrobel/procyon

配置文档：通过下载链接 https://bitbucket.org/mstrobel/procyon/downloads ， 下载 jar 包到本地即可

使用文档：https://bitbucket.org/mstrobel/procyon/wiki/Java%20Decompiler

功能：将反编译得到 jar 包还原成 java 文件，同时能够更强的还原代码的逻辑结构

下面直接对比一下，即可看到 Procyon 的优势所在了。直接去到 Procyon 的目录下，并将 enjarify 反编译好的微信 jar 文件放置到同级目录，同时执行如下命令反编译整个 jar 包：

```
java -jar procyon-decompiler-0.5.30.jar -jar XXX.jar -o 目录名
```

即可看到目录下指定生成的目录，和控制台窗口相应的反编译 log 信息。为了验证 Procyon 的亮点，我同时将 dex2jar 反编译微信的某个类拿出来做对比（类名是MCacheItem）得到结果如下：

相同的一个方法体的代码逻辑，图一是 dex2jar 的，图二是 Procyon 的，哪个能够更好的读懂相信大家自然心中有数，像这样的栗子是很多的，这里就不一一例举。除了上面反编译这个 jar 文件的命令外，Procyon 还有下面两个文件使用比较高频：

```
java -jar procyon-decompiler-0.5.30.jar 获取帮助命令
```

```
java -jar procyon-decompiler-0.5.30.jar xxx.class 反编译单个class文件
```

ClassyShark

工具名：ClassyShark

官网：http://classyshark.com/

源码：https://github.com/google/android-classyshark

配置文档：通过下载链接 https://github.com/google/android-classyshark/releases ， 下载 jar 包到本地即可

使用文档：

- Github markdown版本：https://github.com/borisf/classyshark-user-guide
- Github PDF版本：https://github.com/google/android-classyshark/blob/master/CommandLine.pdf
- 简书版本：http://www.jianshu.com/p/8e8b88ea2197

功能：可直接浏览 Apk，支持对.dex, .aar, .so，.apk, .jar, .class等文件的操作。

ClassyShark 同样是 Google 的亲儿子，利用它我们可以查看一些 Apk 使用了什么技术，方便作为我们开发的参考进行使用。同样以别人家的 Apk 作为分析的栗子，这回不拿微信了，因为从 Apk 分析上看，微信大部分基本上都是用自己的技术，参考价值可能不太大，换一个目标下手（常常逛知乎，就拿知乎了）。操作同样很傻瓜化，去到 ClassShark 的目录下执行如下命令即可：

```
java -jar ClassyShark.jar -open XXX.apk
```

可以看到如下界面

我们可以看到如下界面，左边包含了 xml 文件、 classes 文件以及 lib 目录及其所包含的 so 库。直接点击某个 xml 文件即可看到其文件内容，非常方便，不过这个工具的重点不在这里。直接点击 classes 目录下某个 dex 文件

从上面两张图中你可以看到知乎两个 dex 包所包含的方法数是多少（这个信息对我们可能没啥用），还有就是它使用了 Facebook 开源的图片加载库 Fresco、开源数据库 Realm、还有腾讯和新浪的一些SDK。

如果我切换到另外一个 Tab，还可以看到更多的信息

它分别列出了 Apk 下每个包中的方法数，并且以直观的扇形图展示在右边。透过这里也可以看到知乎还用了 Apache 的开源库、Twitter 的移动开发检测平台 Fabric、 Rxjava、 Retrofit 以及 Square 出品的 OKIO 库。发现大厂引入的开源技术我们也可以进行相应的学习，并权衡考虑引入使用。

最后补充一点，ClassyShark 能显示包中方法数的这个功能对我们开发自家产品还是有所帮助的，假设你引入大量的第三方库而导致了 Apk 超过了 65K 方法的限制，可以通过它来查找是哪些第三方库的方法过多导致。网上很多童鞋都推荐什么计算方法数的 AS 插件，个人觉得还不如用 ClassyShark 方便一些。

TcpDump

工具名：TcpDump

官网：http://www.androidtcpdump.com/

下载地址：http://www.androidtcpdump.com/android-tcpdump/downloads

使用文档：

- Trinea版本：http://www.trinea.cn/android/tcpdump_wireshark/
- MrPeak版本：http://mrpeak.cn/blog/tutorial-tcpdump/

功能：对手机进行网络抓包，**前提是手机已经获取 Root 权限**。支持 HTTP 和 HTTPS ，还支持更多其他协议。

Tcpdump 的使用就不介绍太多，具体怎么玩可以看上面 Trinea 和 MrPeak 的文章，已经写得很完善了。TcpDump 没有GUI，要查看抓包日志可以通过控制台窗口，但明显很鸡肋，为了方便还是把所有网络日志的抓包信息放到 pcap 文件里面，然后结合下面要介绍的 WireShark 进行操作。

WireShark

工具名：WireShark

官网：https://www.wireshark.org/

下载地址：https://www.wireshark.org/#download

使用文档：http://www.cnblogs.com/TankXiao/archive/2012/10/10/2711777.html （非常齐全的文档）

功能：

- 配置 Tcpdump 生成打 pcap 文件，进行分析；
- 或者可以让电脑建立热点，手机连上电脑创建的热点进行抓包，这种方式处理起来更加方便灵活；

WireShark 支持的协议也是非常多，可以看到我抓包出来的结构也非常详细，WireShark 的基本使用自行参考上面提供的链接即可，如果要玩得很深的话，需要对网络协议有一定深度的了解，童鞋们可以选择性自由发挥。

Fiddler

工具名：Fiddler

官网：http://www.telerik.com/fiddler

下载地址：https://www.telerik.com/download/fiddler

使用文档：直接参考 Trinea 的文章 http://www.trinea.cn/android/android-network-sniffer/ 即可。

功能：支持对 HTTP 和 HTTPS 两种协议进行抓包。

相比 Tcpdump ，它的好处在于手机不需要 Root，且有自己的 GUI 。不过它只支持 HTTP 和 HTTPS 两种协议。另外在使用时，手机也需要连接 WIFI 并设置好代理，具体查看 Trinea 的文章好了。抓包的过程还是能够获得不少信息的，我在抓了新浪微博和微信的包后感受是，微信的防抓包工作比以前做得好了，微博好像就没发现有什么变化，不知道是不是错觉。

Fiddler 是本文介绍的最后一个工具，需要补充一下，请教了一些童鞋，它们说 Mac 下使用 Charles 会更爽，额，我是 Windows 用户，Mac 下的童鞋就自行折腾吧。

总结

以上就是目前自己常用的一些竞品分析工具，有针对 Apk 反编译的，也有对应用进行网络抓包的。竞品分析并非只是帮助我们去了解竞争对手的产品，很多时候我只是出于对某些应用的技术实现感兴趣，而对其进行一番挖掘，挖掘的过程中你会发现某种新技术你可以开始学习了。例如像知乎使用 realm 数据库一样，其实 realm 正式版发布也没多久，不过既然大厂已经投入使用那有空也需要适当的跟进学习，说不定哪天 realm 真的干掉来了 SQLite 呢。

当然有时候除了学新技术，还派上了另一番用场，诸如某次同事对一个 App 的某些数据信息感兴趣，于是乎专门做了爬虫到网上爬取数据，后来我反编译了 App 后大致捋了一遍，发现该 App 在 raw 目录下其实已经放着一个本地数据库了，对比了数据基本差别无几，直接可以拿来用了。或者可以拿着这些工具来对自家应用动手，找找自家的产品哪里的安全性还不够，可以进行完善弥补，未尝不是一件好事。

文章总结到此，本人水平有限，无法做过多深入讲解，感兴趣竞品（逆向）分析技术的童鞋可以考虑看看**《Android软件安全与逆向分析》** 或者逛逛**看雪论坛**膜拜一些大神的奇技淫巧！