美团Android资源混淆保护实践

2015/10/23 • 技术分享 • 2 评论 • android, 工具

前言

Android应用中的APK安全性一直遭人诟病，市面上充斥着各种被破解或者汉化的应用，破解者可以非常简单的通过破解工具就能对一个APK进行反编译、破解、汉化等等，这样就可以修改原有代码的逻辑、添加新代码、添加或修改资源、或者更有甚者植入病毒等等，从而破坏原有APK的安全和用户体验，最终伤害到用户和原有的开发者。

而事物都是有两方面的，有矛就有盾，针对Android应用安全的各种方案应运而生，大家比较熟悉一般是各类加壳加固的工具，我们可以使用这些工具来保护我们的APK，加壳加固是另外一个话题了，我们这里不对加壳加固进行介绍，后续如果有机会会单独开一个话题讨论，我们在开发过程中可以通过ProGuard或者DexGuard来保护我们的代码，从而实现相对的代码安全，但我们的资源呢？我们往往忽略对资源文件的保护，那这里将要分享的是如果采用常规方式对APK中的资源文件进行保护。

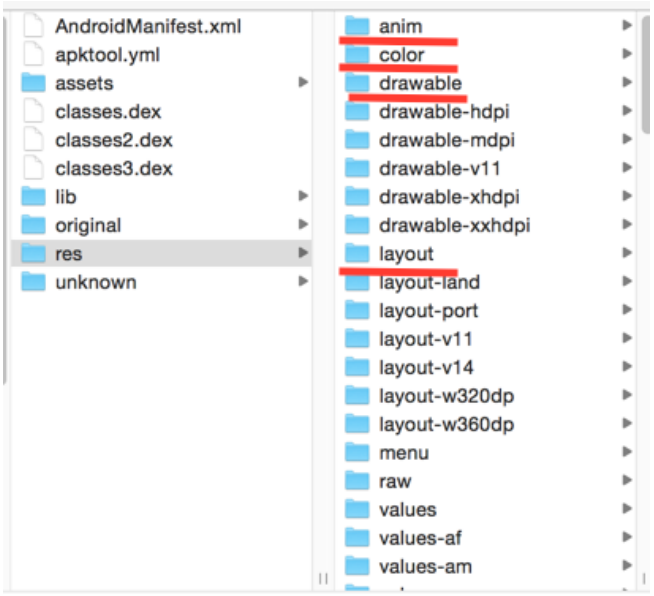
资源安全

资源安全这个话题目前大家关注度不算太高，相比较而言大家更关注代码安全，目前市面上各类APP基本都使用了ProGuard来保护代码的安全，但对资源文件的保护力度都不大，其实资源文件是存在比较大的安全隐患，那资源会有哪些安全隐患呢？下面我们通过一个比较简单的例子来说明下保护资源文件的重要性。

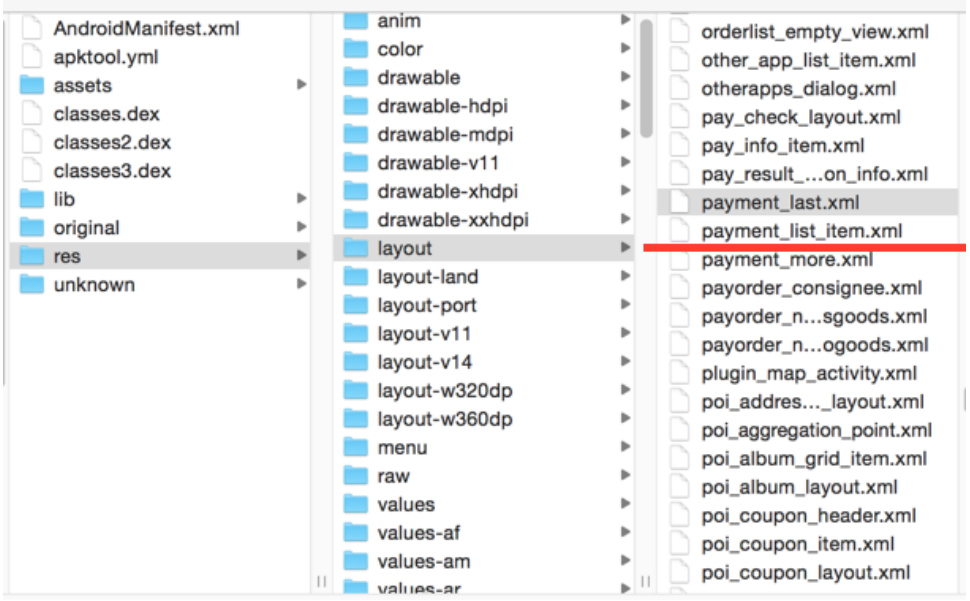
我们先用最常见的apktool工具来反编译一个应用来看看，通过运行下面命令就能进行反编译：

```Java
1 apktool d -s xxx.apk
```

反编译成功后我们来看下反编译得到的文件结构（见下图）：



通过上图中的目录结构，我们可以看到这个应用的资源文件大概有：anim、drawable、layout、menu、values等等，我们可以通过修改这些文件夹下的资源文件，并通过apktool进行回编译（apktool b 命令）就能创建一个经过修改过的APK应用，例如我们修改下图中红色横线所标示的layout文件，就能往原有APK的支付信息（根据资源名称猜测这个layout的意图）中添加一些我们自己的东西；

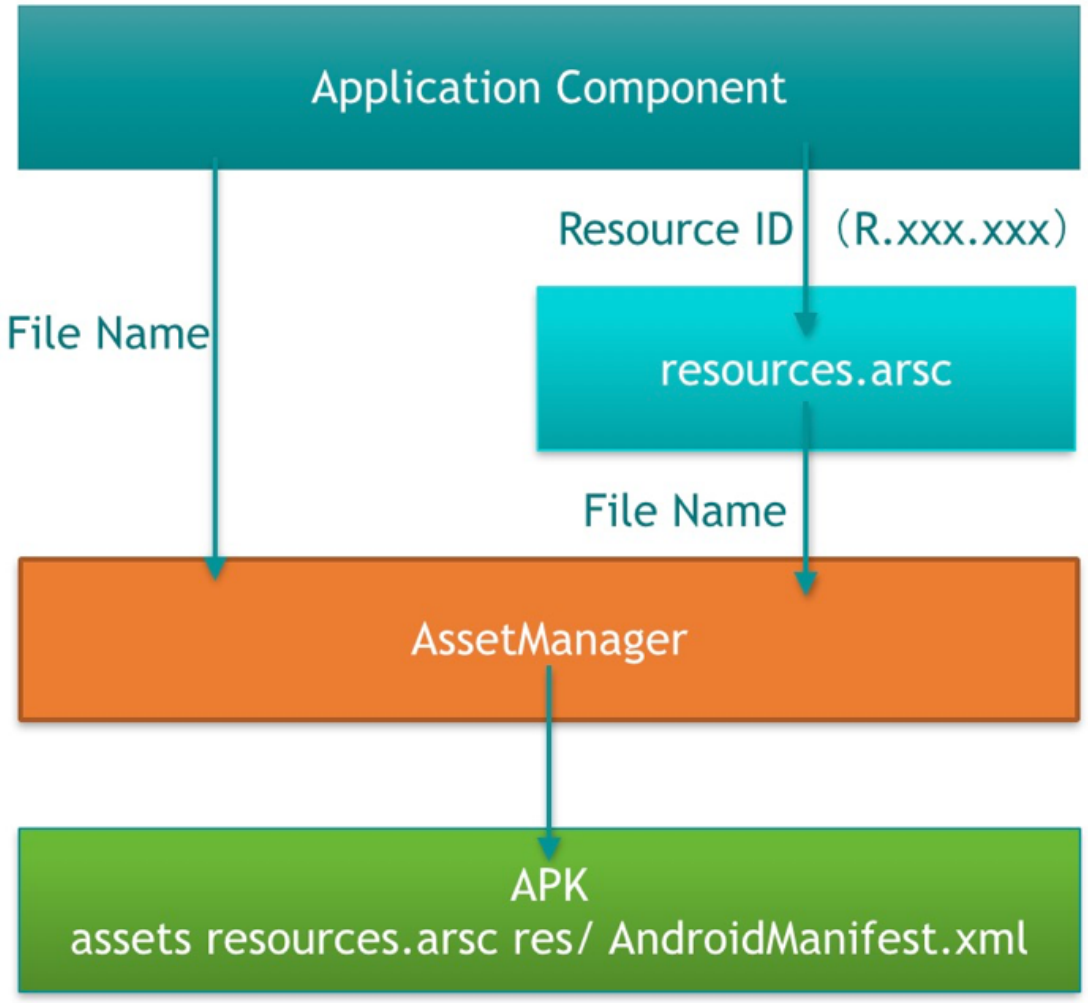


这个问题主要是因为我们在开发过程中倡导命名的规范性，一般都要求在命名时做到见名知意，这样能够方便我们自己的理解和维护，但同时这也方便了破解者，破解者可以轻松的根据文件名称来猜测这个文件的意图和作用，从而做破坏性的修改。

通过这个例子我们可以看出目前资源安全的重要性，那如何做到资源安全呢？安全都是相对的，没有绝对的安全，我们接下来要讨论的是类似Proguard方式的对我们的资源进行保护。我们主要是通过修改AAPT工具来对资源进行保护，为了方便理解，下面先讲一下Android应用是怎么查找资源的。

Android查找资源的流程

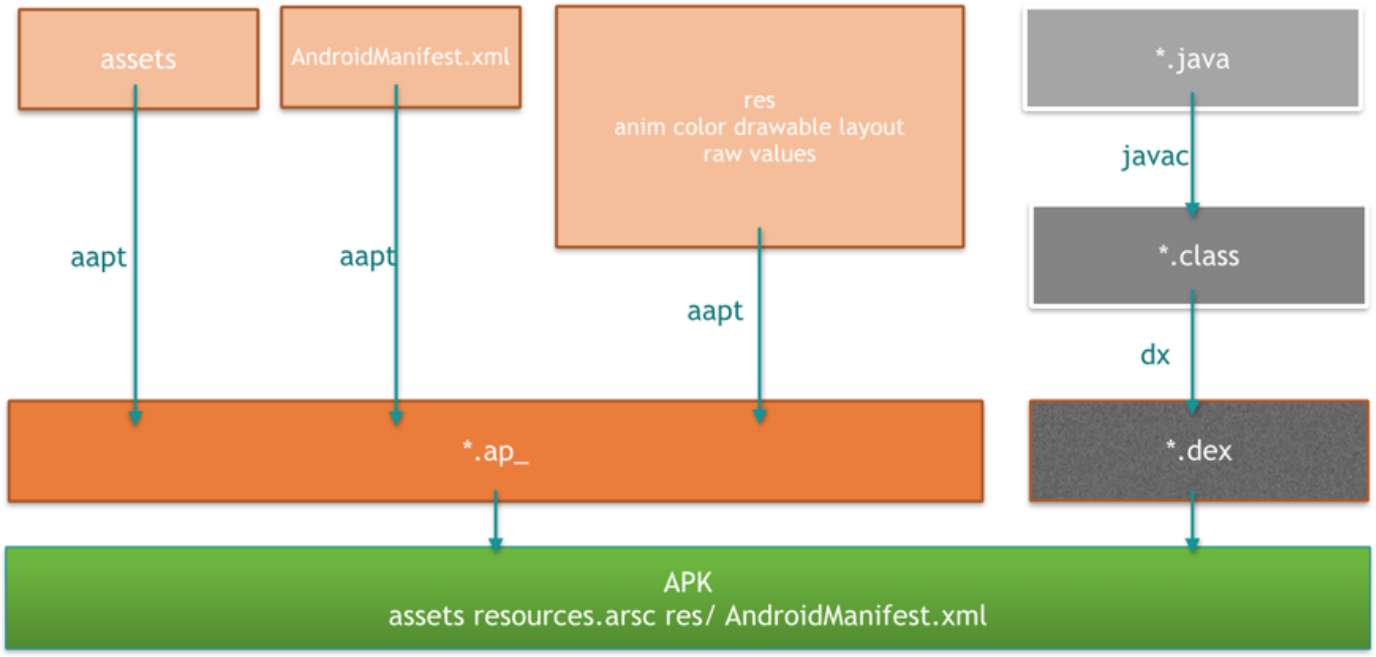
在Android系统中，每一个应用程序一般都会配置很多资源，用来适配不同密度、大小和方向的屏幕，以及适配不同的国家、地区和语言等等。这些资源是在应用程序运行时自动根据设备的当前配置信息进行适配的。这也就是说，给定一个相同的资源ID，在不同的设备配置之下，查找到的可能是不同的资源。
这个查找过程对应用程序来说，是完全透明的，这个过程主要是靠Android资源管理框架来完成的，而Android资源管理框架实际是由AssetManager和Resources两个类来实现的。其中，Resources类可以根据ID来查找资源，而AssetManager类根据文件名来查找资源。事实上，如果一个资源ID对应的是一个文件，那么Resources类是先根据ID来找到资源文件名称，然后再将该文件名称交给AssetManager类来打开对应的文件的。基本流程如下图：



通过上图我们可以看到Resources是通过resources.arsc把Resource的ID转化成资源文件的名称，然后交由AssetManager来加载的。
而Resources.arsc这个文件是存放在APK包中的，他是由AAPT工具在打包过程中生成的，他本身是一个资源的索引表，里面维护者资源ID、Name、Path或者Value的对应关系，AssetManager通过这个索引表，就可以通过资源的ID找到这个资源对应的文件或者数据。

AAPT介绍

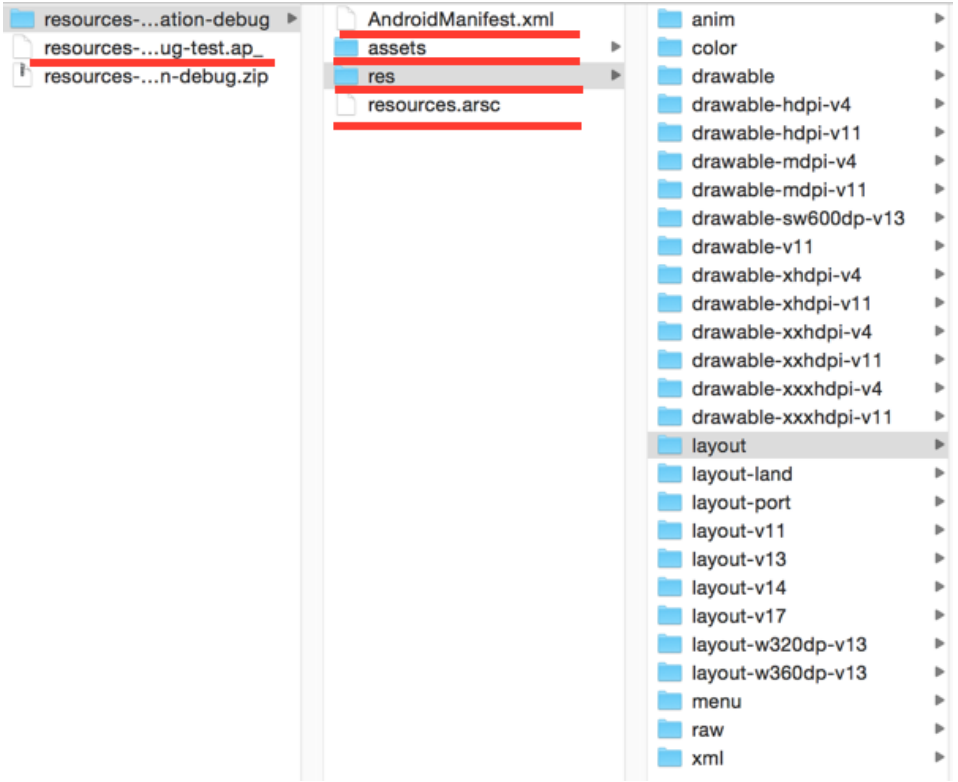
AAPT是Android Asset Packaging Tool的缩写，它存放在SDK的tools/目录下，AAPT的功能很强大，可以通过它查看查看、创建、更新压缩文件(如 .zip文件，.jar文件，.apk文件)，它也可以把资源编译为二进制文件，并生成resources.arsc，AAPT这个工具在APK打包过程中起到了非常重要作用，在打包过程中使用AAPT对APK中用到的资源进行打包，这里不对AAPT这个工具做过多的讨论，只看一下AAPT这个工具在打包过程中起到的作用，下图是AAPT打包的流程：



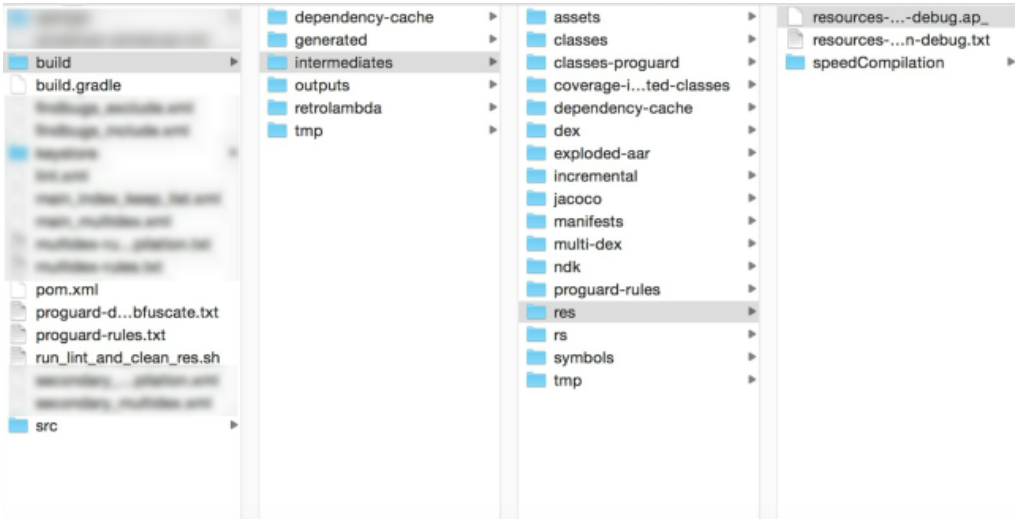
AAPT这个工具在打包过程中主要做了下列工作：

1. 把”assets”和”res/raw”目录下的所有资源进行打包（会根据不同的文件后缀选择压缩或不压缩），而”res/”目录下的其他资源进行编译或者其他处理（具体处理方式视文件后缀不同而不同，例如：”.xml”会编译成二进制文件，”.png”文件会进行优化等等）后才进行打包；
2. 会对除了assets资源之外所有的资源赋予一个资源ID常量，并且会生成一个资源索引表resources.arsc；
3. 编译AndroidManifest.xml成二进制的XML文件；
4. 把上面3个步骤中生成结果保存在一个*.ap_文件，并把各个资源ID常量定义在一个R.java中；

.ap_这个文件会在生成APK时放入APK包中， .ap 这个文件本身是一个ZIP包，他里面包含resources.arsc、AndroidManifest.xml、assets以及所有的资源文件，下图是UNZIP后的截图：



可以看出*.ap这个文件中包含的内容，这个文件存放在build/intermediates/res的目录下，下图是这个文件存放的路径截图：



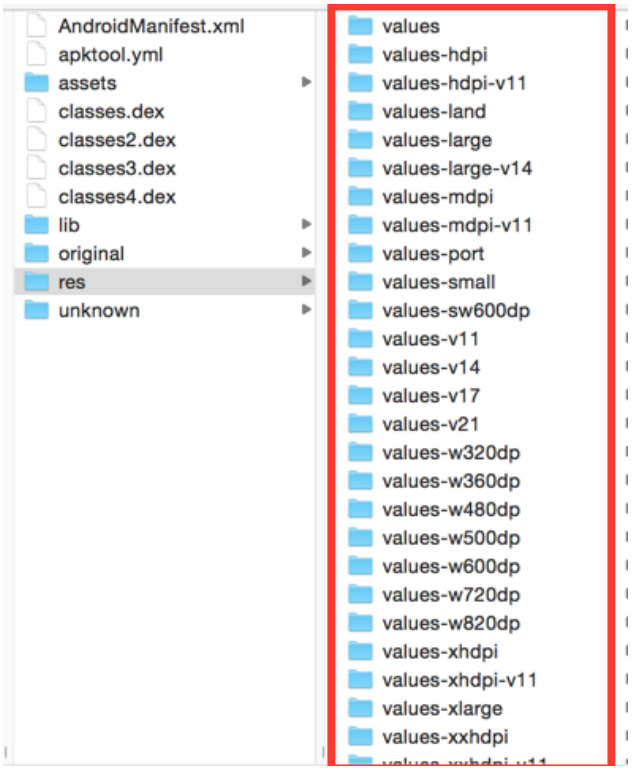
资源保护

我们这里参考Proguard Obfuscator方式，对APK中资源文件名使用简短无意义名称进行替换，给破解者制造困难，从而做到资源的相对安全；通过上面分析，我们可以看出通过修改AAPT在生成resources.arsc和*.ap_时把资源文件的名称进行替换，从而保护资源。
通过阅读AAPT编译资源的代码，我们发现修改AAPT在处理资源文件相关的源码是能够做到资源文件名的替换，下面是Resource.cpp中makeFileResources()的修改的代码片段：

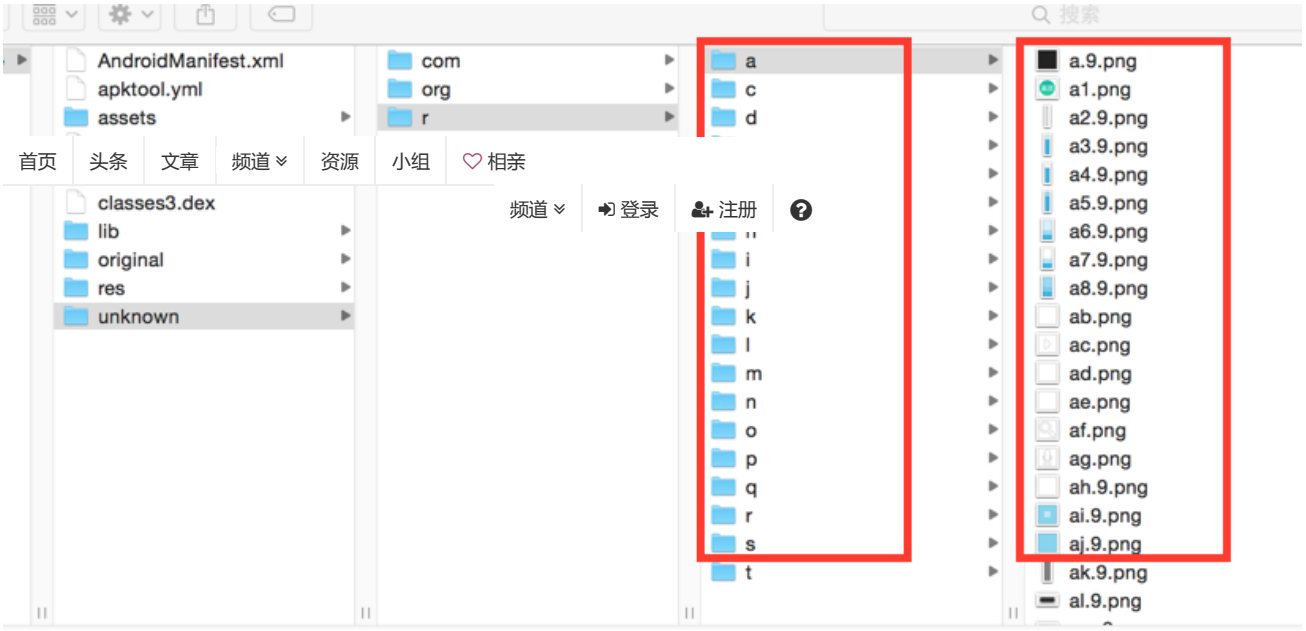
Java

```
static status_t makeFileResources(Bundle* bundle, const sp<AaptAssets>& assets,
                                  ResourceTable* table,
                                  const sp<ResourceTypeSet>& set,
                                  const char* resType)
    {
        String8 type8(resType);
        String16 type16(resType);

        bool hasErrors = false;

        ResourceDirIterator it(set, String8(resType));
        ssize_t res;
        while ((res=it.next()) == NO_ERROR) {
            if (bundle->getVerbose()) {
                printf("    (new resource id %s from %s)\n",
                       it.getBaseName().string(), it.getFile()->getPrintableSource().string());
            }
            String16 baseName(it.getBaseName());
            const char16_t* str = baseName.string();
            const char16_t* const end = str + baseName.size();
            while (str < end) {
                if (!((*str >= 'a' && *str <= 'z')
                        || (*str >= '0' && *str <= '9')
                        || *str == '_' || *str == '.')) {
                    fprintf(stderr, "%s: Invalid file name: must contain only [a-z0-9_.]\n",
                            it.getPath().string());
                    hasErrors = true;
                }
                str++;
            }
            String8 resPath = it.getPath();
            resPath.convertToResPath();

            String8 obfuscationName;
            String8 obfuscationPath = getObfuscationName(resPath, obfuscationName);

            table->addEntry(SourcePos(it.getPath(), 0), String16(assets->getPackage()),
                            type16,
                            baseName, // String16(obfuscationName),
                            String16(obfuscationPath), // resPath
                            NULL,
                            &it.getParams());
            assets->addResource(it.getLeafName(), obfuscationPath/*resPath*/, it.getFile(), type8);
        }

        return hasErrors ? UNKNOWN_ERROR : NO_ERROR;
    }
```

上述代码是在ResourceTable和Assets中添加资源文件时， 对资源文件名称进行修改，这就能够做到资源文件名称的替换，这样通过使用修改过的AAPT编译资源并进行打包，我们再用上面讲到的apktool这个工具进行反编译，下图是反编译后的截图：



发现什么变化了吗？在res目录下熟悉的layout、drawable、anim、menu等文件夹不见了，那他们去哪了呢？因为apktool工具把它们放到了unknown文件夹下了，见下图：



让我们来看一下unknown文件夹，你会发现资源文件名已经被简短无意义名称进行替换了，这样会给反编译者制造理解上的困难，反编译者需要消耗一定的时间来搞清楚这些资源文件的作用，资源混淆带来的另外一个好处是能明显减小APK的大小，资源混淆既能保护资源文件的安全又能减小安装包的大小，那我们何乐而不为呢？

这样通过修改AAPT，我们可以在代码零修改的基础下就能做到相对的资源安全，当然安全是相对的，没有绝对的安全。

- Android开发中的MVP架构
- Android数据库高手秘籍(2)：创建表和LitePal的基本用法
- Android的消息机制之ThreadLocal的工作原理
- Android中如何计算App的启动时间？
- Android应用启动优化:一种DelayLoad的实现和原理(下篇)
- 面试时，问哪些问题能试出一个Android应用开发者真正的水平？
- Android性能优化之被忽视的优化点

可能感兴趣的话题

- CentOS6.5装DoNetCore
- Python开发爬虫需要用Scrapy框架吗 • 4
- “hello world”用各语言怎么写，一起来接龙 • 20
- 学习iOS，但是想多多了解一些关于算法方面的问题,应该看什么书比较好 • 10
- HTML中的select下拉框内容显示不全的解决办法 有什么方法给下拉框带个小浮窗
- 程序员都会破解qq号密码吗？ • 57
- 如何看待快播事件 • 68
- 滴滴出行2016研发工程师笔试题(亮灯问题) • 19
- 为什么对朋友说我毕业后准备去北漂，然后换来的是各种嘲笑呢？ • 93
- github上收到了第一个pull request

« 延迟加载 Dex 文件
Android的消息机制之ThreadLocal的工作原理 »

登录后评论 新用户注册

直接登录

最新评论

风 君 子 （ ） 2015/11/02

可是在哪里修改呢

赞 回复

天海 （ 1 ） 2015/12/19

有机会，试试！

赞 回复

安卓 ▼
输入搜索关键字
搜索

小组话题 **我有新话题**

想必大家都有用过淘宝的APP，一些...
栗子3.0..... 发起 • 2 回复

安卓APP程序猿都用什么debug?
M. 发起 • 3 回复

插件式开发？
Abduweli 发起 • 1 回复

什么值得买 “再按一次退出”的toas...
Tiegeda 发起 • 1 回复

最近发觉手机好像中毒或被人动了手脚...
' ScorpioMan 发起 • 3 回复

求做优化伙伴指点～关于耗电排行的
黄明明 发起

- 本月热门安卓文章
- 年度热门
- 热门标签

0 给 App 提速：Android 性能优化...

1 Android Studio 2.0：速度提升...

2 Android 项目重构之路：架构篇

3 对Android开发有用的技术栈（一）架...

4 快讯：Android Studio 2.0 预览...

5 Android退出应用最优雅的方式

6 Android 项目重构之路：界面篇

7 Android 项目重构之路：实现篇

8 Android样式的开发：shape篇

9 最棒的开源 Android 应用：聊天、...

业界热点资讯 **更多 »**

面对虚假广告 谷歌是如何做的
2 天前 • 5 • 1

龙芯之痛：国产芯片陷烧钱怪圈
5 天前 • 5 • 6

英特尔的Management Engine被发现在设备休眠时接受数据包
4 天前 • 3

Skype将默认屏蔽IP，保护游戏玩家
4 天前 • 2

科技行业的五大巨头
4 天前 • 2

安卓工具资源 更多资源 »

Guice：Google轻量级依赖注入框架
依赖注入

Jersey：你值得拥有的Java RESTful框架
Android

Google code

Google Java Style：Google的Java编程规范
Android, Java, 知名网站

Manymo：在线安卓系统模拟器工具
Android, 设计, 设计工具

Android-Xbmcremote：官方远程Android版 XBMC
Android, 开发库

最新评论

- Re: Android应用中MVP最佳实践
  mark 一下。
- Re: Android应用启动优化:一种Delay...
  index++
- Re: 从零开始搭建架构实施Android项目
  感谢分享！

- 
  Re: 从零开始搭建架构实施Android项目
  好文，收藏了。
- 
  Re: Android网络请求心路历程
  太牛！
- 
  Re: Android应用中MVP最佳实践
  用过MVP一段时间。总的来说，MVP的最大的特点就是，V与P互相绑定，一个V对应一个P。这样的好处是...
- 
  Re: 从零开始搭建架构实施Android项目
  好文，收藏了，工程结构那块也看了。
- 
  Re: 一个优秀的Android应用从建项目...
  好文好文收藏收藏谢谢谢谢

关于安卓频道

安卓频道分享Android开发文章，精选工具和安卓相关的行业动态。

快速链接
问题反馈与求助 »
安卓工具资源 »
安卓技术话题 »

关注我们

新浪微博：@安卓开发频道
RSS：订阅地址
推荐微信号

合作联系
Email：bd@Jobbole.com
QQ： 2302462408 （加好友请注明来意）

更多频道

小组 - 好的话题、有启发的回复、值得信赖的圈子
头条 - 分享和发现有价值的内容与观点
相亲 - 为IT单身男女服务的征婚传播平台
资源 - 优秀的工具资源导航
翻译 - 翻译传播优秀的外文文章
文章 - 国内外的精选文章
设计 - UI,网页，交互和用户体验
iOS - 专注iOS技术分享
安卓 - 专注Android技术分享
前端 - JavaScript, HTML5, CSS
Java - 专注Java技术分享
Python - 专注Python技术分享

 首页 博客 资源 小组 相亲 反馈 本站由 UCloud 赞助云主机