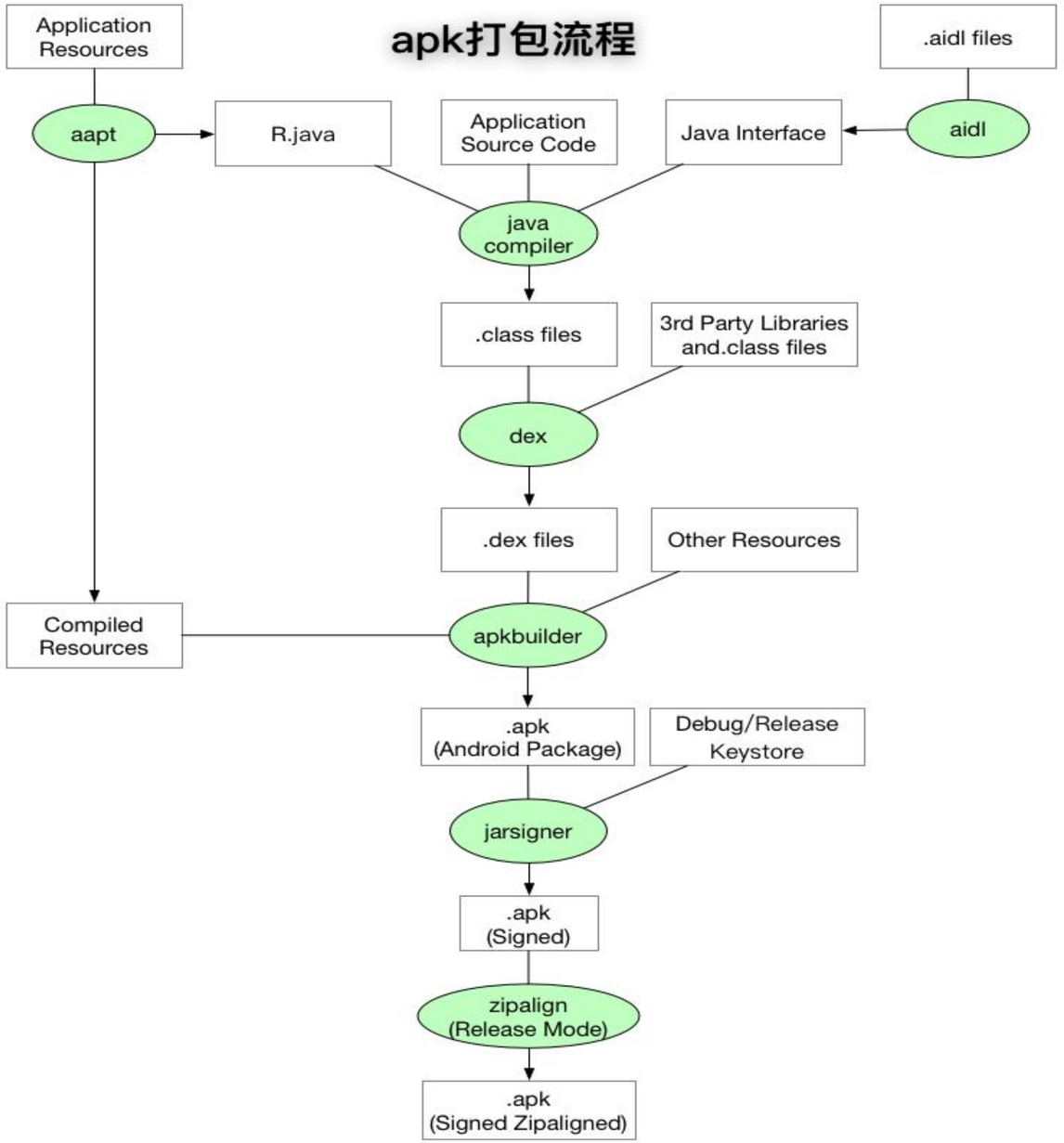
防反编译利器技术框架

APK构建流程图

- 打包流程



R. Java文件

R. Java结构图

```
public final class R {
    private R() {}

    public static final class anim { … }
    public static final class attr { … }
    public static final class bool { … }
    public static final class color { … }
    public static final class dimen { … }
    public static final class drawable { … }
    public static final class id { … }
    public static final class integer { … }
    public static final class layout  { … }
    public static final class string  { … }
    public static final class style  { … }
}

R.string.appname = 0x7f07006b
```

MutiDex方案配置

修改Build.gradle文件

```
//minSdkVersion 设置为 21 或更高值

android {
    defaultConfig {
        ...
        minSdkVersion 21
        targetSdkVersion 28
        multiDexEnabled true
    }
    ...
}
```

```
//minSdkVersion 设置低于 21时

android {
    defaultConfig {
        ...
        minSdkVersion 15
        targetSdkVersion 28
        multiDexEnabled true
    }
    ...
}

dependencies {
    compile'com.android.support:multidex:1.0.3'
}
```

MutiDex方案配置

修改Application代码

```
//继承MultiDexApplication

public class MyApplication extends
MultiDexApplication { ... }
```

```
//调用 MultiDex.install(this) 启用 分包

public class MyApplication extends SuperApplication {
 @Override
 protected void attachBaseContext(Context base)                {
   super.attachBaseContext(base);
   MultiDex.install(this);
 }
}
```

MutiDex加载原理

```
/**
* Installer for platform versions 14, 15, 16, 17 and 18.
*/
private static final class V14 {
    private static void install(ClassLoader loader, List<File> additionalClassPathEntries,
            File optimizedDirectory){
        // 扩展ClassLoader实例的"pathList"字段。
        Field pathListField = findField(loader, "pathList");
        Object dexPathList = pathListField.get(loader);
        expandFieldArray(dexPathList, "dexElements", makeDexElements(dexPathList,
                new ArrayList<File>(additionalClassPathEntries), optimizedDirectory));
    }
    private static Object[] makeDexElements(
            Object dexPathList, ArrayList<File> files, File optimizedDirectory)
                    throws IllegalAccessException, InvocationTargetException,
                    NoSuchMethodException {
        Method makeDexElements =
                findMethod(dexPathList, "makeDexElements", ArrayList.class, File.class);
        return (Object[]) makeDexElements.invoke(dexPathList, files, optimizedDirectory);
    }
}
```

MutiDex加载原理

BaseDexClassLoader

```
protected Class<?> findClass(String name) {
    List<Throwable> suppressedExceptions     = new
ArrayList<Throwable>();
   Class c = pathList.findClass(name,
suppressedExceptions);
   …
   return c;
 }
```

DexPathList

```
public Class<?> findClass(String name, List<Throwable>
suppressed) {
      for (Element element : dexElements) {
         Class<?> clazz = element.findClass(name,
definingContext, suppressed);
         if (clazz != null) {
            return clazz;
         }
      }
      …
}
```

Proguard配置

```
android {
    compileSdkVersion 23
    buildToolsVersion "24.0.1"
    defaultConfig {
    }
    buildTypes {
        release {
            //主要看这部分:
             zipAlignEnabled true
             minifyEnabled true
             proguardFiles getDefaultProguardFile('proguard-android.txt'),
      'proguard-rules.pro'
        }
    }
}
```

build.gradle配置

混淆配置实例

规则实例

```
-keep public class com.droidyue.com.widget.**


-keepclassmembers public class * extends android.view.View
{
   void set*(***);
   *** get*();
}

-keepclassmembers class **.R$* {
    public static <fields>;
}


-keepclasseswithmembernames class * {
    native <methods>;
}

-dontwarn android.support.**
```

反编译工具对抗

- 花指令

在原始程序中插入一组无用的字节，但又不会改变程序的原始逻辑，程序仍然可以正常运行，然而反编译工具在反编译这些字节时会出错，造成反汇编工具失效，提高破解难度。

例如下面的dalvik指令：

```
0003bc: 1250                |0000: const/4 v0, #int 5 // #5
0003be: 2900 0400           |0001: goto/16 0005 // +0004
0003c2: 0001                |0003: <Junkbytes>
0003c4: 0000                |0004: <Junkbytes>
0003c6: d800 000            |0005: add-int/lit8 v0, v0, #int 1 // #01
0003ca: 0f00                |0007: return v0
```

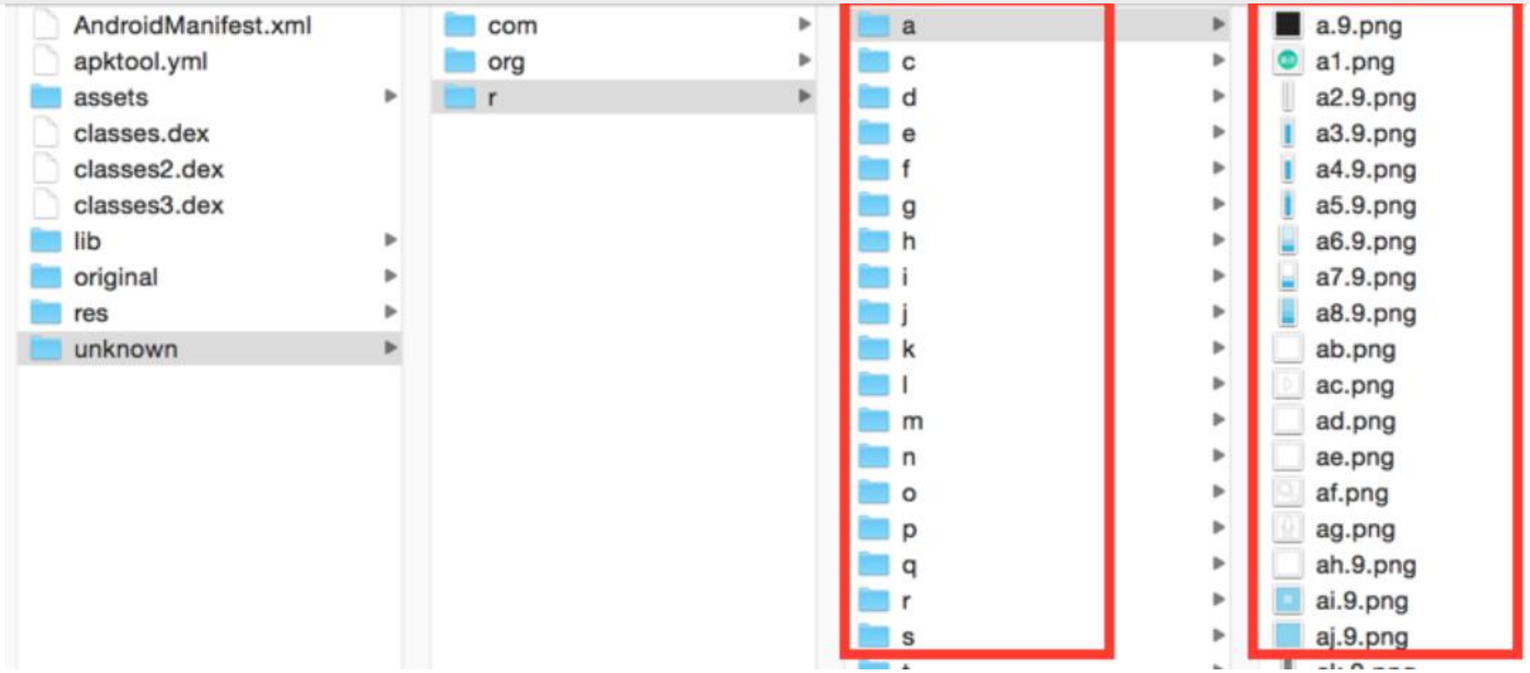
如果反编译工具采用线性扫描算法，会错误识别花指令导致出错

```
0003bc: 1250                |0000: const/4 v0, #int 5 // #5
0003be: 2900 0400           |0001: goto/16 0005 // +0004
0003c2: 0001 0000 d800 0001 |0003: packed-switch-data (4 units)
0003ca: 0f00                |0007: return v0
```

资源混淆

- 修改aapt

修改aapt处理资源文件相关的源码，参考proguard 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。



资源混淆

- 修改resources.arsc

根据resources.arsc文件格式，修改资源名与路径的映射关系。

工具	描述
table stringblock	改变文件指向路径，例如res/layout/test.xml，改为res/layout/a.xml
资源文件名	修改资源的文件名，即将test.xml重命名为a.xml
specsname stringblock	旧的specsname除了白名单部分全部废弃，替换成所有混淆方案中用到的字符。由于重复使用[a-z0-9_]，specsname的总数量会大大减少。
entry中指向的specsname中的id	例如原本test.xml它指向specsname中的第十项，我们需要用混淆后的a项的位置改写。
table chunk的大小	修改table chunk的最后大小