文章编号：2096-1472(2019)-06-08-04
DOI:10.19644/j.cnki.issn2096-1472.2019.06.003

Android APP加固方案的研究

彭守镇
(广东理工学院信息技术学院，广东 肇庆 526100)

摘　要：Android应用程序在未经任何加固处理的情况下极容易受到反编译攻击，存在较大的安全隐患。APP加固技术为应用程序提供了有效的防护措施，增加了反编译的难度，本文针对目前常用的反编译手段提出了一种APP加固方案，该方案首先针对需要保护的资源文件进行加密处理；然后针对DEX文件的反编译，采用混淆代码技术加固处理，针对二次打包，本文采用签名校验技术。实验表明，本文提出的加固方案能够有效地防患APK被反编译，加大了二次打包的难度。

关键词：APP反编译；DEX加固；加壳保护；二次打包
中图分类号：TP309.5　**文献标识码**：A

Research on the Reinforcement Scheme for Android Apps

PENG Shouzhen
(*Guang Dong Polytechnic College*,*Zhaoqing* 526100,*China*)

Abstract: Android applications without any reinforcement are extremely vulnerable to de-compilation attacks,with significant security risks.App reinforcement technology provides effective protection for applications and increases the difficulty of de-compilation.In this paper,an APP reinforcement scheme is proposed against the commonly used de-compilation methods.The scheme first encrypts the resource files to be protected;Then,for the DEX file recompilation,confusion code technology is used to reinforce the DEX file.For the secondary packaging,the signature verification technique is adopted in this paper.Experimental results show that the proposed reinforcement scheme can effectively prevent APK from being decompiled and thus increases the difficulty of secondary packaging.

Keywords: App de-compilation;DEX reinforcement;shell protection;secondary packing

1 引言(Introduction)

Android程序在编译后形成字节码文件，经过打包、优化、签名等工具处理后最终形成可发布的APK文件。然而，未经加固处理的APK文件极容易被反编译工具进行逆向工程处理，暴露程序的实现细节，甚至被植入恶意代码[1]。Android APP加固技术能够有效地防止程序被反编译、破解、二次打包、注入等，提高程序的安全性，从而为维护知识版权增加了一道“防火墙”。

目前，Android APP加固技术[2]较多，常用的加固方法有混淆代码、DEX加固、虚拟机加固、加密等。爱加密通过对APP进行漏洞分析并采用DEX加壳保护、内存和资源文件等的保护技术能够有效地防止APP被二次打包。NAGA IN[3]通过推出的手机客户端安全检测控件为银行、基金、券商等手机软件提供了有效的安全保障。360加固保[4]、阿里聚通过签名校验、代码加密压缩、内存动态跟踪等技术为移动应用提供专业性安全保护。本文在研究当前各种加固技术的基础上提出了一种Android APP加固方案，方案通过加密、混淆代码、签名校验等多种方法对目标代码进行加固操作，经实验验证，本文提出的加固方案能够很好地对抗反编译、二次打包、注入等攻击。

2 APP反编译概述(Overview of APP de-compilation)

2.1 Android APK结构

APP反编译无非是对Android应用程序编译打包后的APK进行逆向工程，然后分析APK的目录结构和应用程序的行为，并篡改应用程序代码、注入恶意代码。不同的应用程序APK结构不同，但每一个APK中都包含基本目录结构，也是逆向工程攻击的目标。基本目录结构如下：

(1)classes.dex文件。该文件是JAVA源码经JDK编译和

基金项目：广东省教育厅大学生创新创业项目(项目编号：201713720011)；广东理工学院大学生创新创业项目(项目编号：CXCY2017045).

DEX优化后的DEX字节码文件，可以直接运行在Dalvik虚拟机上。DEX文件包括文件头和文件主体两部分，DEX文件头主要包括校验字段、签名字段偏移地址和长度信息等，DEX文件的主体部分包括索引区和数据区。图1是DEX文件的一个实例。

图1 DEX文件实例

Fig.1 Instance of DEX file

DEX文件的构成如图2所示。

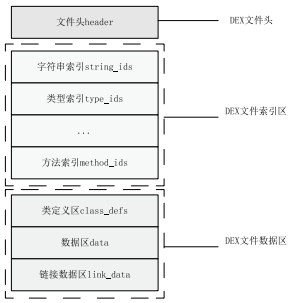


图2 DEX文件构成

Fig.2 Composition of DEX file

(2)AndroidManifest.xml文件。该文件为应用程序的配置文件，应用程序建立时存在。配置文件描述了应用程序的配置信息，如应用的名称、图标、版本、权限的授予、引用的库信息等，在打包时配置文件经过了压缩。

(3)META-INF目录。该目录存放签名文件，如CERT.RSA、CERT.SF和MANIFEST.MF。

(4)res目录和assets目录。这两个目录为应用程序的资源文件目录，目录下的部分资源将被打包进APK中。

(5)resources.arsc文件。该文件为二进制资源文件的索引文件，程序运行时引用资源ID，通过该索引文件的映射找到对应的资源。

2.2 反编译工具

当前，网络上有许多反编译之类的免费工具，Android APP反编译常用的工具有：

(1)APKTool：APKTool[5]是GOOGLE提供的APK编译工具，利用该工具能够获取APK中的图片、布局等资源文件和目录结构。

(2)dex2jar[6]：该工具能够将APK中的classes.dex文件转换成JAR文件。

(3)JD-GUI：是反编译工具之一，利用该工具能够查看dex2jar生成的JAR文件，显示反编译后的JAVA源代码。

利用以上三个工具能够轻松地将APK反编译成JAVA源代码，其实现步骤如下：

步骤一：利用APKTool工具对APK实施第一轮反编译，反编译后得到APK的目录结构及资源文件。

步骤二：利用dex2jar工具将APK目录下的classes.dex文件转换成JAR文件，生成的JAR文件无法直接查看。

步骤三：利用JD-GUI工具查看将DEX文件转换后的JAR文件，源代码直接在该工具下查看，反编译结束。

3 反编译分析(Analysis of de-compilation)

针对以上反编译工具的使用我们不难发现，没有经过加固的APK极容易被反编译，应用程序代码的安全性存在巨大的隐患。下面本文分析隐患存在的原因：

(1)APK本身存在一定的开放性[7]。我们将APK文件的后缀名手工改成压缩文件的后缀名，如.rar或.zip，其目录结构完全暴露出来。虽然部分文件如AndroidManifest.xml、classes.dex等文件经过了编译，但其目录下的部分资源文件却是原封不动地被打包进APK，如res、assets目录下的资源文件。

(2)JAVA字节码易被反编译[8]。JAVA代码经编译后形成字节码保存在class文件中，而字节码本身是一种中间代码，是解析后的带有丰富语义的跨平台编码，这些语义字节码直接表达了程序的行为和动机，经过极为简单的反编译就能破解其中的语义信息。

(3)Android APP的签名机制能够被伪造[9]。在JDK 1.7版本前APK的签名能够轻易地被伪造，从而将反编译后的文件进行二次打包。虽然在后来的版本中使用原APK的签名进行二次打包是无效的，但二次打包工具却能够为反编译后的文件进行重新签名，从而诞生了一个全新的伪造的应用程序。

4 加固方案(Reinforcement scheme)

经过对反编译过程进行分析我们不难发现，如果需要提高应用程序的安全性，增加APK的反编译难度，就必须针对反编译使用的手段采取相应的措施来加固APK。本文在研究反编译技术的基础上提出一种加固Android APP的方案，该方案增加了反编译的难度，能够有效地防止APK被反编译。

4.1 加固方案描述

针对反编译工具能够轻易地提取APK中的资源文件，本文提出抽取需要保护的资源文件，并对其进行加密处理；针对DEX文件的反编译，本文采用混淆代码加固处理；针对二次打包，本文采用签名校验技术防患重签名，从而加大了二次打包的难度。加固方案的基本操作流程如图3所示。

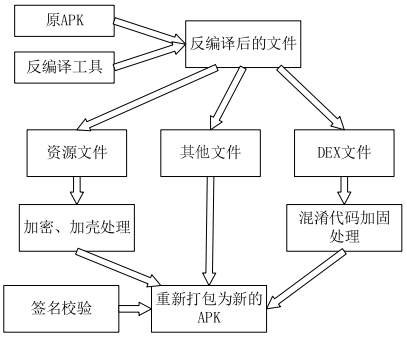


图3 加固方案操作流程

Fig.3 Flow chart of reinforcement scheme operation

4.2 加固方案的实现

本方案的实现分三个步骤进行：

第一步：对资源文件进行加密、加壳处理。处理过程如下 ：

(1)将APK文件的后缀名改成.zip或.rar，并将修改后的APK文件解压至一文件下。

(2)抽取需要保护的资源文件。编写JAVA文件读取程序，由文件读取程序以文件流方式读取需要保护的资源文件。

(3)对抽取的文件采用加密处理，并隐藏加密文件。在文件读取的过程中采用MD5加密算法对文件流进行加密处理，并设置文件的属性为隐藏。

(4)编写资源解密恢复程序，并将代码放置Application类的入口。在Application类的入口处编写MD5解密算法，在程序正式执行前调用解密算法，恢复加密的资源文件，以保证程序能够正常访问资源文件。

经过以上步骤处理，APK中的资源文件已经被加密，反编译工具得到的将是加密后的资源文件。而APK在正式执行前已经调用解密算法恢复了被加密的资源文件，加密方案并不影响程序对资源的正常访问。

第二步：对classes.dex进行代码混淆加固处理。处理过程如下：

(1)利用反编译工具对classes.dex文件实施反编译。常用的反编译工具dex2jar和JD-GUI能够轻松地对DEX文件进行反编译，反编译后生成jar文件，从JD-GUI工具中能够查看该jar文件中的源码文件。以新建一个“HelloWorld”应用程序为例，将该程序APK中的classes.dex文件经反编译后在JD-GUI中打开，我们能看到，未经代码混淆处理的APK经反编译后，其源代码暴露无遗，图4是“HelloWorld”应用程序经反编译后的结果。

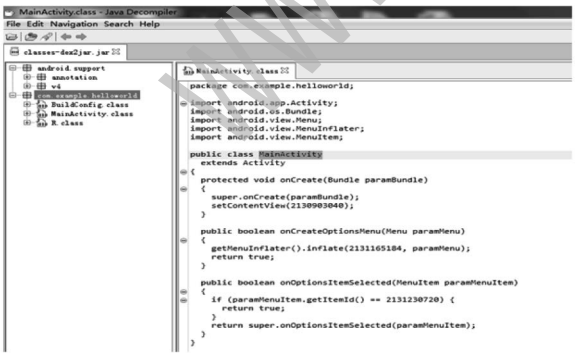


图4 反编译“HelloWorld”

Fig.4 Decompile "HellWorld"

(2)利用ProGuard工具对反编译后的应用程序实施代码混淆。在Android SDK中集成了一个ProGuard免费工具，也可使用其他版本的ProGuard工具，利用该工具可以对反编译后的jar文件实施代码混淆。ProGuard的执行过程由六个步骤组成：

第一步：输入：将原DEX文件反编译后的jar文件作为输入值。

第二步：压缩：检测并移除代码中无用的类、字段、方法等。

第三步：优化：移除无用指令，对字节码进行优化。

第四步：混淆：使用a、b、c这样的简短字母对类、字段和方法进行重命名。

第五步：预检：在Java平台上对处理后的代码进行预检，确保加载的class文件是可执行的。

第六步：输出：最后将混淆后的代码以jar文件形式输出。

ProGuard工具执行过程如图5所示。

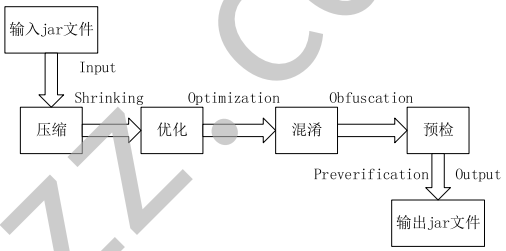


图5 ProGuard执行过程

Fig.5 Execution process of ProGuard

第七步：重新打包和签名。处理过程如下：

(1)新建文件夹，将反编译后的所有工程文件夹和文件放置在该文件夹下。

(2)利用打包工具生成目标APK，并利用重签名工具对目标APK实施重签名。打包工具可以使用apktool，重签名工具可以使用signapk工具。

5 实验验证(Experimental verification)

为了验证本方案是否可行，本文以未经任何加固处理的APK为实验对象，采用本方案提供的操作步骤，对实验对象实施加固处理。

(1)实验目的

通过方案提供的操作步骤验证采用本方案能否对目标APK实施加固处理，并通过对加固处理后的APK实施反编译，验证反编译能否成功。

(2)实验预期结果

如果对加固后的APK实施反编译不能成功，表明本方案的加固操作能够有效地防患APK被反编译，加大了二次打包的难度。

(3)实验过程

本实验以名为“Helpme.apk”的APK为实验对象，该APK未经任何加固处理，其源代码目录结构如图6所示。

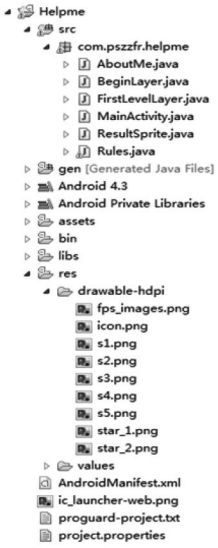


图6 目标APK源代码目录

Fig.6 Directory of target APK's source code

为了对APK中的需要保护的资源文件进行加密处理，本文选择APK中的图片资源作为重要的保护资源，并对其实施MD5加密处理。加密过程调用事先编写好的方法，以其中一张图片“s4.png”为例，加密前后对比的结果如图7所示。

(a)加密前原图能够正常打开

(b)加密后图片无法正常打开

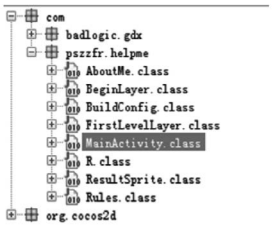
图7 图片加密前后对比

Fig.7 Image comparison before and after encryption

接下来利用反编译工具dex2jar对classes.dex文件实施反编译。在cmd命令窗口定位到dex2jar所在的文件夹，并输入如下命令：

d2j-dex2jar.bat classes.dex

命令执行完毕后在dex2jar文件夹下新生成一个classes-dex2jar.jar文件，该文件为反编译后生成的压缩文件，不能直接打开，可以利用JD-GUI工具查看该文件的目录结构和文件，JD-GUI工具下查看的目录结构如图8所示。



(a)DEX文件反编译后的目录结构

(b)DEX文件中某文件源代码

图8 DEX文件反编译结果

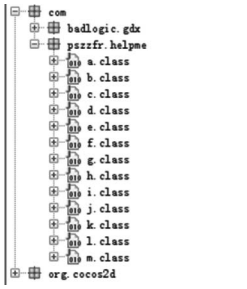
Fig.8 Result of de-compilation for DEX file

下面利用ProGuard工具对反编译后的jar文件实施代码混淆，为了代码混淆的实施能够顺利进行，并且能够看到混淆后的结果，在实施混淆之前需要对相关参数进行设置。

支持库参数：添加三项支持库，即：rt.jar、cocos2d-android.jar和android.jar。

输出参数：设置输入文件为classes-dex2jar_pro.jar。

利用ProGuard工具实施代码混淆的结果如图9所示。



(a) 代码混淆后的目录结构

(b) 代码混淆某文件源代码

图9 实施代码混淆的结果

Fig.9 Result of implementing code confusion

从图8和图9的对比可以看出，实施代码混淆后的目录结构发生了变化，原目录结构下的文件被不能表达任何语义的简短字母代替，源代码中的类、字段和方法等也被简短字母代替，从而增加了反编译的难度。

虽然我们对DEX文件实施了代码混淆，但利用相关工具依然能够对实施代码混淆后的DEX文件实施二次反编译，为此，本文利用signapk工具进行签名校验，由于篇幅所限，本文不再对签名校验的过程做详细说明。

本文最后将加密后的资源文件和代码混淆后的文件进行重新打包，并使用签名校验技术实施重签名，将新打包的APK再次实施反编译，最后实验结果表明，反编译过程出现错误，由此证明了本文提出的APP加固方案能够有效地防患APK被反编译，加大了二次打包的难度。再次对加固后的APK文件实施反编译实验结果如图10所示。

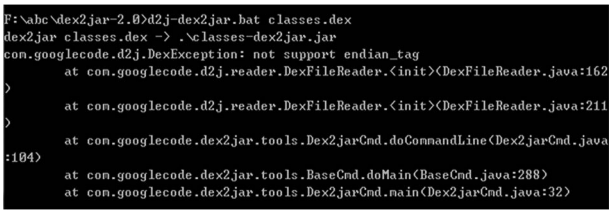


图10 对加固后APK实施反编译的结果

Fig.10 Result of de-compilation for the reinforced APK

6 结论(Conclusion)

本文系统地介绍了一种Android APP加固方案，方案为了对APP中的资源进行有效的保护，采用了加密方式对重要资源进行加密处理，为了防止DEX文件被反编译，采用了混淆代码技术加固处理，为了防止APK被二次打包，采用了签名校验技术。最后通过实验验证了本方案的可行性。

参考文献(References)

[1] 吴敬征,武延军,武志飞,等.基于有向信息流的Android隐私泄露类恶意应用检测方法[J].中国科学院大学学报,2015,32(06):807-815.

[2] 宋言言,罗森林,尚海,等.函数Native化的Android APP加固方法[J].浙江大学学报(工学版),2019,53(03):555-562.

[3] 梆梆安全.梆梆加固[EB/OL].http://blog.csdn.net/justfwd/article/dct-ails/51164281,2018-02-10.

[4] 赵跃华,刘佳.安卓APP安全加固系统的分析与设计[J].计算机工程,2018,44(02):187-192.

[5] 郑兴生.Android应用程序反编译工具研究与设计[D].北京理工大学,2016.

[6] 梁丹.基于动态字节码注入的Android沙盒模型[D].上海交通大学,2015.

[7] 章宗美,桂盛霖,任飞.基于N-gram的Android恶意检测[J].计算机科学,2019,46(02):145-151.

[8] 刘奥,过辰楷,王伟静,等.Android应用Activity启动环研究[J/OL].计算机学报,2019:1-18.

[9] 汪润,唐奔宵,王丽娜.DroidFAR:一种基于程序语义的Android重打包应用抗混淆检测方法[J].武汉大学学报(理学版),2018,64(05):407-414.

作者简介：

彭守镇(1979-)，男，硕士，讲师.研究领域：计算机应用技术，信息安全，软件技术.

(上接第21页)

[4] 钱晓捷,张路一.融合评分结构特征与偏好距离的协同过滤推荐算法[J].计算机工程,2017,43(5):185-190.

[5] 李强,何兴盛,傅忠谦.基于用户与物品间差异的推荐算法研究[J].中国科学技术大学学报,2016,46(2):113-119.

[6] Zhou X,He J,Huang G,et al.A Personalized Recommendation Algorithm Based on Approximating the Singular Value Decomposition (ApproSVD)[C].Ieee/wic/acm International Conferences on Web Intelligence and Intelligent Agent Technology.IEEE,2013:458-464.

[7] 郑鹏,王应明,梁薇.基于信任和矩阵分解的协同过滤推荐算法[J].计算机工程与应用,2018,54(13):34-40.

[8] Ar Y,Bostanci E.A genetic algorithm solution to the collaborative filtering problem[J].Expert Systems with Applications,2016,61:122-128.

[9] 臣健美,孙亚军.基于用户近邻的N维张量分解推荐算法[J].计算机工程,2017,43(11):193-197.

[10] Koren,Yehuda.Collaborative filtering with temporal dynamics[C].ACM,2009:447-456.

[11] Bakir C.Collaborative Filtering with Temporal Dynamics with Using Singular Value Decomposition[J].Tehnicki Vjesnik,2018,25(1):130-135.

[12] 孙光福,吴乐,刘淇,等.基于时序行为的协同过滤推荐算法[J].软件学报,2013,24(11):2721-2733.

[13] 孙雀,卢剑波,张凤凤,等.植物物种多样性与岛屿面积的关系[J].生态学报,2009,29(5):2195-2202.

[14] 郭新明,弋改珍.混合模型的用户兴趣漂移算法[J].智能系统学报,2010,5(2):91-94.

作者简介：

李明秀(1998-)，女，本科生.研究领域：推荐系统.

王淑军(1996-)，男，硕士生.研究领域：分布式系统，知识图谱.

贾 如(1982-)，女，博士，讲师.研究领域：推荐系统，数据挖掘.

陈立荣(1980-)，女，讲师，博士生.研究领域：电子商务信誉与信任.本文通讯作者.