

基于 Android 平台的软件加固方案的设计与实现

巫志文, 李炜

(北京邮电大学网络技术研究院, 北京 100876)

摘要: 随着 Android 智能手机的普及, 其软件安全问题逐渐引起人们的关注。由于 Android 软件使用的编程语言 Java 极易被反编译, 导致 Android 软件比其他使用编译型语言的程序更容易被逆向破解。为了解决这个问题, 本文在深入分析 Android 的系统架构、软件结构及其执行机制的基础上, 提出和实现了一种基于 classes.dex 文件动态加载的 Android 软件加固方案, 实现软件关键代码的隐藏, 可以有效地对抗各种针对 Android 软件的逆向工程攻击。

关键词: 软件加固; 系统架构; 软件结构; Android

中图分类号: TP309

Design and Implementation of Software Reinforcement Scheme Based on Android Platform

Wu Zhiwen, Li wei

(Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract: With the popularity of Android smart phones, its software security problem attracted people's attention gradually. As Android software use Java as its programming language, which is very easy to decompile, so it is easier to reverse break than other program that use compiled language. In order to solve this problem, this paper deeply analyzed Android's system architecture, software structure and its execute mechanism, proposed and implemented an Android software reinforcement scheme base on classes.dex file's dynamic loading. This scheme can hide software's key code and can effectively resist reverse attack against Android software.

Key words: software reinforcement; system architecture; software structure; Android

0 引言

Android 是 google 公司于 2007 年推出的开源手机操作系统, 由于其强大的功能和灵活的定制能力, 使其在短短几年内便跃居智能手机操作系统市场份额的首位, 据著名市场研究机构 IDC 的最新数据表明, 2014 年第二季度, Android 系统智能手机的市场份额高达 84.7%, 远超其他系统^[1]。

虽然 Android 系统在设计之初便充分考虑到了安全问题, 但随着其广泛应用, 诸多潜在的安全问题还是逐渐地暴露出来, 对其安全性的研究开始受到人们的广泛关注。由于 Android 平台软件使用的编程语言是 Java, 而 Java 源代码编译后的二进制代码极易被反编译, 导致其破解难度远小于其他使用编译性语言编写的程序。虽然 Android 2.3 以后加入了代码混淆机制, 但通过逆向工程, 其 API 级别上的代码仍是难以隐藏, 对攻击者来说, 破解的可能性仍是很大。一旦软件的关键部位被破解或注入恶意代码, 黑客就有机会通过特殊手段拿到用户数据。而这对于一些诸如银行客户端、支付客户端来说, 后果将会是十分严重的。

本文详细分析了 Android 的系统架构、程序结构、程序执行机制以及 Android 平台软件面临

作者简介: 巫志文 (1987 年 9 月), 男, 现北京邮电大学网络技术研究院网络与交换国家重点实验室硕士, 主要研究方向: 移动安全

通信联系人: 李炜, 男, 副研究员, 博士, 北京邮电大学网络技术研究院副教授, 主要研究方向为业务网络智能化、移动增值业务系统等. E-mail: liwei@ebupt.com

的逆向威胁，提出和实现了一种基于 classes.dex 文件动态加载的 Android 软件加固方案。

1 Android 系统架构

Android 是基于 Linux 内核的开放式操作系统，采用层次化系统架构。如图 1 所示，Android 自底向上分为四个主要功能层^[2]，分别是：Linux 内核层、系统运行库层、应用程序框架层和应用程序层。

Android 4.0 版本以前，其内核为 Linux 2.6 内核，4.0 及之后的版本使用 Linux 3.X 内核。Linux 内核服务实现诸如内存管理、进程管理、网络协议栈和驱动模型等核心系统功能^[3]。系统运行库层位于 Linux 内核层之上，由系统类库和 Android 运行时构成，是应用程序框架的支撑，为 Android 系统中的各个组件提供服务。应用程序框架层提供开发 Android 应用程序所需的一系列类库，包含 4 种基本组件，丰富的控件、资源管理器、内容提供者、活动管理器等，使开发人员可以快速地进行应用程序的开发，方便地重用组件，也可以通过继承实现个性化的扩展。应用层包括各类系统自带或开发者开发的应用程序。位于应用层上的各种各样的应用程序为用户提供了丰富的服务，是用户操作 Android 设备的入口。



图 1 Android 系统架构图

2 Android 软件结构及其执行机制

本软件加固方案是在对 Android 软件结构及其执行机制进行深入研究和分析的基础上提出的，下面对相关知识进行介绍。

每个要被安装到 Android 系统中的应用都要被编译打包成为一个后缀名为.apk 的单独的文件，其中包含了应用程序的二进制代码、资源及配置文件等。apk 文件实际上是一个 zip 格式的压缩文件，解压后，可以看到其文件结构及功能描述如表 1 所示：

表 1 Android 应用程序文件结构

文件名	说明
META-INF	存放签名信息
res	存放资源文件的目录
AndroidManifest.xml	程序全局配置文件
classes.dex	java 源码编译后生成的 Dalvik 字节码文件
resources.arsc	编译后的二进制资源文件

其中，classes.dex 是 java 源码编译后生成的 Dalvik 二进制字节码文件，可以直接在 Dalvik 虚拟机中运行。classes.dex 文件由文件头、索引区和数据区三大部分组成^[4]，表 2 给出了各部分包涵的内容及相关说明：

表 2 classes.dex 文件结构

区块	字段名称	说明
文件头	Header	文件头
	string_ids	字符串索引
	type_ids	类型索引
	proto_ids	方法原型索引
	field_ids	域索引
索引区	method_ids	方法索引
	class_defs	类的定义区
	data	数据区
	link_data	链接数据区

其中，文件头部分主要包括校验和以及其他结构的偏移地址和长度信息。其部分结构定义如表 3 所示：

75

表 3 classes.dex 文件头部分结构定义

字段名称	说明
checksum	校验码
signature	SHA-1 签名
file_size	Dex 文件的总长度
data_size	数据段的大小，必须以 4 字节对齐
data_off	数据段基地址

80

Dalvik 虚拟机在执行应用的时候，首先要解压 apk 文件并校验完整性，然后执行 classes.dex 文件中的字节码^[5]。在执行 classes.dex 文件时，Dalvik 虚拟机首先会检查其文件头部分的 checksum 字段和 signature 字段，确保 classes.dex 文件没有被损坏或篡改，然后根据文件头里定义的其他结构的偏移地址和长度信息进行寻址解析与执行。这也是本软件加固方案的理论基础所在。

3 基于 classes.dex 文件动态加载的 Android 软件加固方案

85

虽然 Android 平台采用了多层次的安全保护机制，但由于 Android 平台自身的缺陷，使用易于反编译的 Java 语言，其软件仍然受到来自各方面的威胁，其中最重要的威胁便是逆向攻击。即通过逆向工程，破解软件的关键部位，从而获取软件信息或实施一系列恶意攻击。逆向 Android 软件的一般步骤是：首先是对其进行反编译，然后阅读反汇编代码，如果有必要还会对其进行动态调试，找到突破口后注入或直接修改反汇编代码，最后重新编译软件进行测试。整个过程可分为反编译、静态分析、动态调试、重编译等 4 个环节。逆向防护技术也是从这四个方面进行的，包括对抗反编译工具、对抗静态分析、对抗动态调试和防止重编译等^[6]。

90

95

为了对抗来自逆向工程的攻击，结合前文分析的 Android 软件结构与执行机制，本文提出了一种基于 classes.dex 文件动态加载的 Android 软件加固方案。本方案首先将 Android 软件分为两部分：软件主体部分与核心功能部分。软件主体部分是实际安装在 Android 系统中的部分，核心功能部分是软件主体部分调用的需要重点保护的功能代码。本方案将核心功能部分编译成一个独立的 apk 文件，加密后将其隐藏于软件主体部分的 classes.dex 文件体内，需要使用时再进行实时的代码分离与动态加载，从而达到隐藏软件关键代码，软件加固的效果。

3.1 方案实现

本软件加固方案由两大部分组成，分别是软件加固模块和动态加载模块。下面对两个模

100 块的具体实现进行介绍。

3.1.1 软件加固模块

根据前文对 classes.dex 文件的文件结构和执行机制的研究，我们知道，Android 系统在解析执行 classes.dex 文件的时候，如果发现其文件头的校验码字段与 SHA-1 签名字段无误，则认为此文件未损坏或未被篡改，是可以执行的。因此，可以猜想，如果在 classes.dex 文件
105 后面增加一些内容，同时在增加这些内容后，修改 classes.dex 文件的校验码、SHA-1 签名及文件大小（修改文件后，该字段也会相应发生变化）字段，classes.dex 仍然能够正确执行。经试验，该方法是可行的。基于这个原理，本方案把部分软件的关键代码编译成一个独立的文件，添加到 classes.dex 文件后面，然后在程序运行时动态地分离出这些代码，再通过反射机制对这部分关键代码进行动态加载。同时，可以对这些关键代码进行加密处理，需要执行
110 时再进行解密。经过这样处理后，逆向工程逆向出来的代码便只有软件的主体部分，而核心部分被隐藏。而且，即使核心部分被发现了，也会由于代码被人为加密而无法得到解密后的源码，从而很好地保护了核心代码。整个加固的实现流程如下：

1、解压软件主体部分 apk 文件，提取其 classes.dex 文件。

2、将核心功能部分代码编译成独立的 apk 文件，并对其进行加密。

115 3、把加密后的 apk 文件写入到软件主体部分的 classes.dex 文件的末尾，并在文件尾部添加加密数据的大小，用以解密时找到核心功能部分代码的起始位置。

4、重新计算 classes.dex 文件的 checksum、signature 和 file_size 字段的值。其中，checksum 字段为 classes.dex 文件第 12 字节以后的内容的循环冗余校验码，存储在 classes.dex 文件的第 8 到 12 字节位置，共 4 个字节；signature 字段为 classes.dex 文件第 32 字节以后的 SHA-1 签名值，存储在第 12 到 32 字节位置，共 20 个字节；file_size 字段为 classes.dex 文件的文件长度，存储在第 32 到 36 字节位置，共 4 个字节。分别计算这几个字段变化之后的值，替换原位置的内容即可。
120

5、将修改后的 classes.dex 文件放回软件主体部分 apk 包中，使用 Android SDK 中提供的签名工具对程序进行签名。

125 整个加固流程如图 2 所示，加固后的 classes.dex 文件结构如图 3 所示。



图 2 APK 加固流程图



图 3 加固后的 classes.dex 文件结构

3.1.2 动态加载模块

经过加固的软件安装到 Android 系统中后，需要调用被隐藏的核心功能部分的时候，需要对其进行动态分离、解密和加载，这个过程是与软件加固过程一一对应的。整个动态加载的实现流程如下：

- 1、软件主体部分从自身的 apk 文件中读取 classes.dex 文件，在 classes.dex 文件尾部得到加密数据的长度，根据加密数据长度计算出加密数据的起始位置，从而读取得到加密数据。
- 2、运行解密方法。解密得到核心功能部分 apk。
- 3、通过 Android API 提供的 DexClassLoader 类，对核心功能部分代码进行反射调用，从而实现核心功能部分代码的动态加载。
- 4、调用完成后，删除核心功能部分 apk 文件，从而避免核心功能部分代码暴露在系统

130

135

140

内部存储之中，被攻击者得到。

整个动态加载的流程如图 4 所示：



图 4 apk 动态加载流程图

145

3.2 方案效果

为了验证效果，本方案建立了两个 Android 工程，分别是软件主体部分工程与核心功能部分工程。包名分别是 com.willen.apkshellandroid 与 com.willen.apkLoad。对软件主体部分 apk 文件使用本方案的加固方式进行加固后，分别使用目前最流行的 Android 逆向工具 dex2jar 与 apktool 进行逆向分析。

150

dex2jar 实现将 classes.dex 文件反编译成一个 jar 包，图 5 是使用 dex2jar 逆向之后形成的 jar 包在 jar 包查看工具 jd-gui 下显示的效果。从图中可以看出，逆向出来的代码只有软件主体部分 (com.willen.apkshellandroid 包的代码)，核心功能部分的代码 (com.willen.apkLoad 包的代码) 被隐藏了，无法查看。

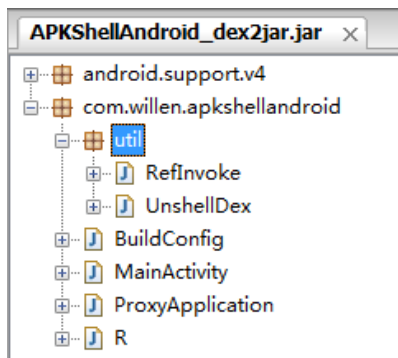


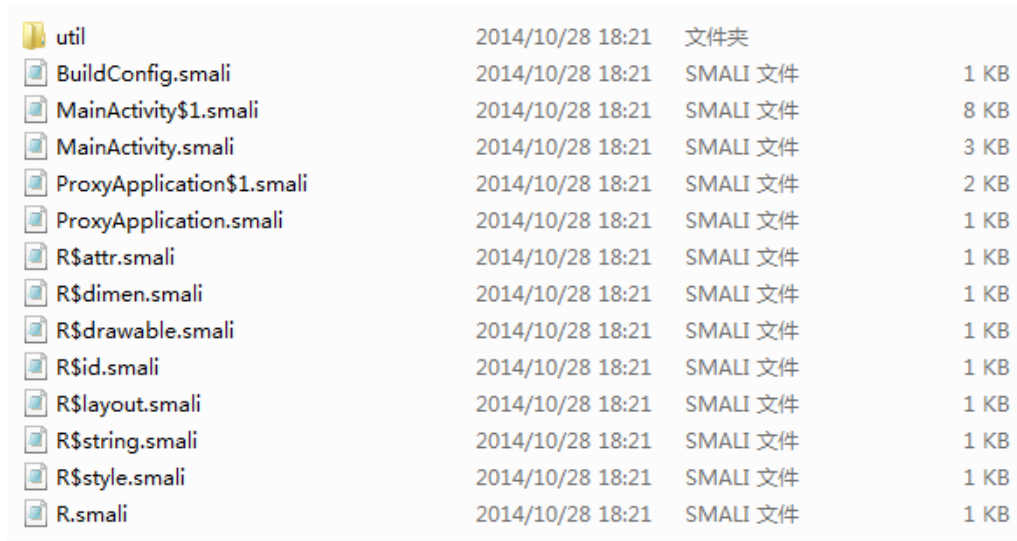
图 5 通过 dex2jar 逆向后的代码

155

apktool 实现将 classes.dex 文件反编译成 smali 汇编代码，图 6 是通过 apktool 反编译后

的代码结构。从图中可以看出，逆向出来的代码中仍然是只有软件主体部分代码，核心功能部分的代码被隐藏。

160



util	2014/10/28 18:21	文件夹	
BuildConfig.smali	2014/10/28 18:21	SMALI 文件	1 KB
MainActivity\$1.smali	2014/10/28 18:21	SMALI 文件	8 KB
MainActivity.smali	2014/10/28 18:21	SMALI 文件	3 KB
ProxyApplication\$1.smali	2014/10/28 18:21	SMALI 文件	2 KB
ProxyApplication.smali	2014/10/28 18:21	SMALI 文件	1 KB
R\$attr.smali	2014/10/28 18:21	SMALI 文件	1 KB
R\$dimen.smali	2014/10/28 18:21	SMALI 文件	1 KB
R\$drawable.smali	2014/10/28 18:21	SMALI 文件	1 KB
R\$id.smali	2014/10/28 18:21	SMALI 文件	1 KB
R\$layout.smali	2014/10/28 18:21	SMALI 文件	1 KB
R\$string.smali	2014/10/28 18:21	SMALI 文件	1 KB
R\$style.smali	2014/10/28 18:21	SMALI 文件	1 KB
R.smali	2014/10/28 18:21	SMALI 文件	1 KB

图 6 通过 apktool 逆向后的代码

经过上述验证可以看出，本软件加固方案可以有效对抗各种针对 Android 软件的逆向工程攻击，提高 Android 软件的安全性。

165

4 结论

作为目前智能手机市场占有率最高的操作系统，Android 系统的安全性备受瞩目。本文首先分析了 Android 系统的系统架构、软件结构与执行机制，在此基础上，提出并实现了一种基于 classes.dex 文件动态加载的软件加固方案，实现了 Android 软件关键代码的隐藏。采用此方案，可以极大程度地提高 Android 软件的安全性。

170

[参考文献] (References)

- [1] IDC：Android 手机市场份额高达 84.7%，中低端手机成主力军 [OL]. [2014-8-15]. <http://bbs.dgtele.com/thread-204503-1-1.html>
- [2] 乜聚虎;周学海;余艳玮;吴志忠. Android 安全加固技术[J]. 计算机系统应用, 2011, 20 (10): 74-77.
- [3] 吴倩, 赵晨啸, 郭莹. Android 安全机制解析与应用实践[M]. 北京: 机械工业出版社, 2013
- [4] Dalvik Executable Format[OL]. [2014-10-21]. <http://source.android.com/devices/tech/dalvik/dex-format.html>
- [5] 陈昱, 江兰帆. 基于 Google Android 平台的移动开发研究[J]. 福建电脑, 2008, 11: 156-157.
- [6] 丰生强. Android 软件安全与逆向分析[M]. 北京: 人民邮电出版社, 2013

175