基于 smali 的 Android 软件敏感 API 调用日志模块嵌入系统

吕晓庆，邹仕洪
（北京邮电大学网络与交换技术国家重点实验室，北京 100876）

摘要：在恶意软件动态分析中，需要用到软件运行时的 API 调用日志，特别是敏感 API 的调用日志，而 Android 系统的日志仅提供了有限的信息。为获取丰富的日志信息，本文设计了基于 smali 的 Android 软件敏感 API 调用日志模块嵌入系统，该系统能够针对指定敏感 API，自动在 Android 软件中嵌入调用日志模块，以达到在软件运行时记录敏感 API 的输入和输出等详细信息的目的。实验证明，本系统是有效可行的。

关键词：Android; 动态分析; smali
中图分类号：TP311

A system for embedding sensitive API call logging module in Android software based on smali

Lv Xiaoqing, Zou Shihong
(State Key Laboratory of Networking and Switching, Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract: In dynamic software analysis, runtime sensitive API call log is needed. However, log system of Android platform only provides finite information. To obtain enough log information, a system of embedding sensitive API call logging module in Android software based on smali is designed. The system could automatically embed call logging module of appointed sensitive API in Android software, so that when the software runs, detailed information of the API, such as input, output and so on, is recorded. Experiments show that this system is effective.

Key words: Android; dynamic analysis; smali

0 引言

Android 由 Google 主持开发，具有显著的开放性，源码开放使其拥有更多的开发者，商业模式开放，使其得到更多的手机厂商支持，并且使得第三方应用商店蓬勃发展。据 Gartner 统计，2012 年第三季度，Android 已经占据了智能操作系统 72.4%的份额[1]。庞大的用户数以及开放性，使得 Android 平台成为黑客关注的重点，据《网秦 2012 年第三季度全球手机安全报告》，2012 年三季度查杀到手机恶意软件 23375 款[2]，手机恶意软件的分析检测成为安全研究人员研究的新课题。

通常，恶意软件的实现需要调用特定 API 来完成，如恶意计费软件会调用发送短信 API，隐私窃取软件会调用访问通讯录 API，此类 API 被称为敏感 API。对恶意软件的分析检测，通常通过静态分析和动态分析两种方式，静态分析指通过反汇编等静态方法来定位、分析软件的恶意代码；动态分析是指通过运行软件并跟踪监控其运行时的行为，分析程序对数据的处理方式，能够更准确地捕捉到软件实际发生的恶意行为。在 Android 平台上，静态分析与传统 PC 平台静态分析并无不同；而对于动态分析，在 Android 平台上，程序运行时系统会输出部分 API 调用日志，但是日志记录信息过少，不足以满足动态分析的要求。为此，需

作者简介：吕晓庆(1988-)，女，硕士研究生，主要研究方向：移动信息安全
通信联系人：邹仕洪(1977-)，男，副教授，主要研究方向：移动信息安全. E-mail: zoush@bupt.edu.cn

要为待分析程序增加敏感 API 调用日志。

Android 软件主要使用 Java 进行开发，反编译较为容易，可以首先将 Android 软件反编译，然后在反编译得到的代码中加入 API 调用日志记录模块，最后再将加入日志记录模块的代码编译成 Android 程序。由于 Android 程序在开发过程中通常会使用混淆等方式，反编译为 Java 代码容易出现代码片段丢失；而 Google 针对 Android 平台提供了名为 baksmali 的反编译器和名为 smali 的编译器，可以实现 Android 软件与 smali 代码的转换[3]。据此，本文设计了基于 smali 的 Android 软件敏感 API 调用日志模块嵌入系统，该系统将待分析程序反编译为 smali 代码，借助 smali 代码完成敏感 API 调用日志模块的嵌入。

本文首先介绍基于 smali 的 Android 软件敏感 API 调用日志模块嵌入系统的实现原理，然后介绍了系统框架以及各模块的功能和实现方案，最后通过实验结果对本系统的作用进行了总结。

1 实现原理

本系统利用 Android 软件自身的格式特点，以及 baksmali 和 smali 这一对反编译器和编译器，设计了能够自动为待分析 Android 软件中针对敏感 API 调用嵌入日志模块的方法和流程。

1.1 Android 软件格式

Dalvik 虚拟机是 Android 平台的核心组成部分之一[4]，负责加载运行 Android 软件。Android 软件主要使用 Java 语言开发，生成的安装包格式为 APK（全称为 Android Package），APK 实际是 ZIP 格式，可以直接解压，其中文件最主要的是 classes.dex，dex 全称为 Dalvik Executable，classes.dex 由软件对应工程的所有 Java 类编译生成，运行在 Dalvik 虚拟机上。

1.2 smali 与 baksmali

针对 dex 格式文件，Google 提供了 smali 和 baksmali 两个工具，分别是 dex 文件的编译器和反编译器，baksmali 可以将 dex 文件反编译成 smali 文件，smali 可以将 smali 文件编译成 dex 文件。而 smali 语法完整的实现了 dex 的所有功能，包括注释、调试信息等等。借助 baksmali 和 smali 可以实现对一个 dex 文件完全无损的反编译和再次编译。

1.3 smali 语法简介

smali 语法是一种宽松式的 Jasmin/dedexer 语法，下面我们简单介绍 smali 中如何表示数据类型、方法、字段[5]以及寄存器[6]。

smali 数据类型包括基本数据类型和引用数据类型，对象和数组属于引用数据类型，其他都属于基本数据类型。smali 中各种基本元素的表示与解释，如下表 1 所示：

表 1 smali 基本元素

Tab. 1 Elements in smali

元素	smali 语法	对应的 Java 语法或解释
基本数据类型	V	void - can only be used for return types
	Z	boolean
	B	byte
	S	short
	C	char
	I	int

	J	long (64 bits)
	F	float
	D	double (64 bits)
对象	Lpackage/name/ObjectName;	等价于 Java 中的 package.name.ObjectName。L 表示这是个对象类型，package/name/表示对象所在的包，ObjectName 是对象的名称，“;”表示对象的结束
数组	单维数组用类似与[I 表示，多维数组每增加一维添加一个[符号	[I 表示 int[]，[[C 表示 char[][]
方法	采用类似于 Lpackage/name/ObjectName;->MethodName(III)Z 的形式	对象调用了 boolean MethodName(int, int, int)方法
变量	采用类似于 Lpackage/name/ObjectName;->FieldName:Ljava/lang/String;	表示对象的类型为 String 名称为 FieldName 变量
寄存器	V 命名方式和 P 命名方式。P 命名方式中的 P0 是方法中的第一个参数寄存器	指令.registers 指定方法中寄存器的总数，指令.locals 表明方法中非参寄存器的数量。当一个方法被调用的时候，方法的 K 个参数被置于最后 K 个寄存器中，而非静态方法中的第一个参数总是调用该方法的对象。

1.4 处理流程

由于 smali 语法完整地实现了 dex 的所有功能，Java 代码中的敏感 API 与相应要嵌入的 API 调用日志模块，都可以找到对应的 smali 代码表示；而 dex 文件可以和 smali 文件进行无损的互相转换，那么可以在 smali 代码层实现敏感 API 的定位以及对应的 API 调用日志模块的嵌入。具体过程为如下图 1 所示：可以首先获取 dex 文件，然后将使用 baksmali 工具将 dex 文件转化为 smali 文件，在 smali 文件中查找敏感 API 对应的 smali 代码，一旦匹配即可嵌入对应的 API 调用日志模块的 smali 代码，然后使用 smali 工具将新的 smali 文件编译成 dex 文件，最后打包生成 APK 文件。

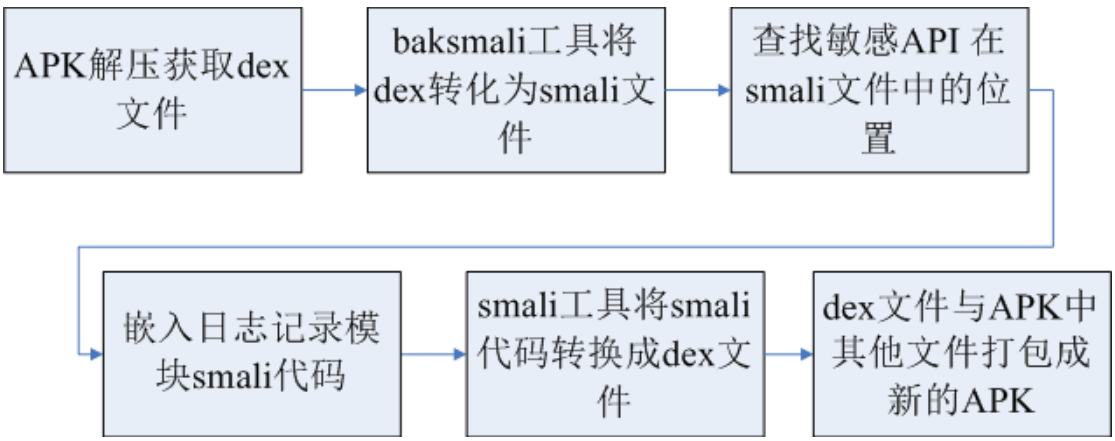

图 1 嵌入 API 调用日志模块流程图

Fig. 1 The flow of embedding call logging module

2 系统设计

2.1 系统框架

基于 smali 的 Android 软件敏感 API 调用日志模块嵌入系统如下图 2 所示，主要由以下五部分组成：APK 反编译模块，敏感 API 代码库，API 调用日志记录代码库，日志嵌入模块，以及 APK 编译模块。

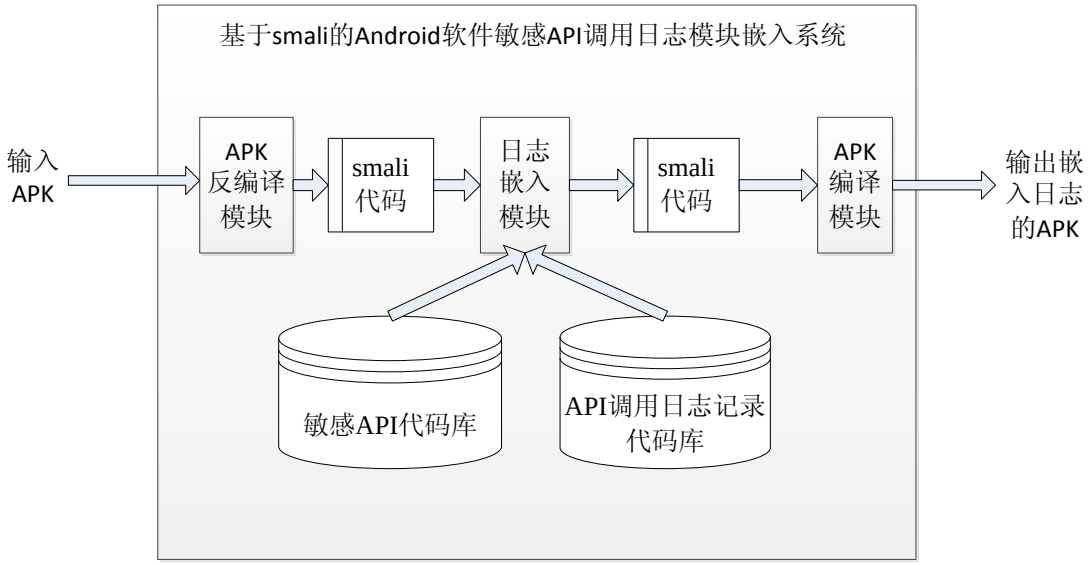

图 2 系统总体架构图

Fig. 2 Architecture of the system

2.2 各模块描述

下面分别对系统五个部分进行详细描述。

2.2.1 APK 反编译模块

该模块负责实现将 APK 中的 classes.dex 文件反编译成 smali 文件[7]的功能，分两步进行：首先，将输入 APK 中的 classes.dex 文件抽取出来，然后将 classes.dex 反编译为 smali 文件。由于 APK 文件是 ZIP 格式文件，本模块通过调用解压工具（WinRAR，7-Zip 等工具）将输入的 APK 解压到某一文件夹下，然后从中选取出 classes.dex 文件，其他文件留存备用；接着本模块通过调用 baksmali 工具，将刚才取出的 classes.dex 反编译为 smali 文件，供日志嵌入模块使用。

2.2.2 敏感 API 代码库

本库存放的是敏感 API 及其对应的 smali 代码，供后续的日志嵌入模块在 APK 反编译模块生成的 smali 文件中定位敏感 API 使用。

敏感 API 来源于对已知恶意软件分析的经验，以及对已知恶意软件和已知正常软件 API 统计的差异。最终选定了可以用来完成发送短信、删除短信、拨打电话、删除通话记录、修改联系人、执行 shell 脚本等功能的 API。

根据 smali 语法中函数的表示方法，得到敏感 API 对应的 smali 语法形式表示。以发送短信为例，通常通过调用 android.telephony.SmsManager 类的 void sendTextMessage(String, String, String, PendingIntent, PendingIntent)方法来实现，在敏感 API 代码库中的记录如表 2 所示：

表 2　sendTextMessage 在敏感 API 代码库的记录

Tab. 2　sendTextMessage in sensitive API codes table

敏感 API ID	功能描述	敏感 API 函数	对应的 smali 语法表示
1	发送短信	android.telephony.SmsManager 类中的 void sendTextMessage(String, String, String, PendingIntent, PendingIntent)函数	Landroid/telephony/SmsManager;->sendTextMessage(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Landroid/app/PendingIntent;Landroid/app/PendingIntent;)V

2.2.3　API 调用日志记录代码库

本库存放的是与敏感 API 调用日志的记录代码相应的 smali 代码。

对于不同的 API，对应的调用日志记录代码有不同的地方，例如对发送短信 API 调用的日志需要记录短信的收件人号码以及短信内容，而对修改联系人号码 API 调用的日志需要记录修改联系人的姓名、修改前的号码和修改后的号码；也有相同的部分，比如 API 所表示的行为的名称，API 调用时间等等，以及为记录信息需要实现的写文件操作部分。为此，对每一个要监控的 API，除了各 API 可通用的调用日志代码部分，还有专门针对该 API 的调用日志代码。

对于 API 通用的调用日志代码部分，设计两个公用函数，第一个函数 wrtieCommonLog 实现对 API 名称、API 调用时间等等公共部分的记录；另一个函数 writeLog 实现对某参数的记录。创建一个工程实现这两个公用函数，然后编译该工程生成 APK，利用 APK 反编译模块将该 APK 中的 dex 文件反编译成 smali 文件，从中提取出两个公用函数对应的 smali 代码。

而每个 API 独特的部分是该 API 的输入和输出，根据 API 函数原型，以及 smali 语法中的寄存器命名原则，获取该 API 需要记录参数在 smali 语法中对应的寄存器编号。如程序中调用 void sendTextMessage(String, String, String, PendingIntent, PendingIntent)发送短信，在对应的 smali 代码中收件人号码参数和短信内容参数对应的寄存器为 V1 和 V3，在 API 调用日志记录代码库中的记录如表 3 所示：

表 3　sendTextMessage 在 API 调用日志记录代码库中的表示

Tab. 3　sendTextMessage in API call log codes table

敏感 API ID	敏感 API 函数	要记录的寄存器
1	android.telephony.SmsManager 类中的 void sendTextMessage(String, String, String, PendingIntent, PendingIntent)函数	V1，V3

函数 writeLog 对应 smali 代码将寄存器值作为输入进行记录，于是，公用函数代码与每个 API 要记录参数对应的寄存器共同构成了 API 调用日志记录代码库，组合可实现对各个 API 调用的日志记录。

2.2.4　日志嵌入模块

本模块负责在 APK 反编译模块生成的 smali 文件中的敏感 API 位置嵌入相应的日志记录代码。首先本模块扫描 APK 反编译模块生成的 smali 文件，与敏感 API 代码库中的 smali 进行比对；如果匹配成功，根据匹配到的敏感 API 在 API 调用日志记录代码库获取对应记

录代码的 smali 代码，并将其加入到 smali 文件的相应位置；接着从该位置开始进行下一轮比对，直至扫描完全部 smali 文件。

2.2.5 APK 编译模块

本模块负责将嵌入日志模块的 smali 文件编译为新的可用 APK。首先本模块通过调用 smali 工具将生成的 smali 文件编译为新的 classes.dex，然后通过调用压缩工具将新的 classes.dex 文件与 APK 反编译模块留存的其他文件压缩生成新的 APK，并对其签名。

3 验证测试

为验证方案的设计效果，在 Windows 7 上对其进行实现，并构造测试用例进行检验。为保证本系统的正常运行，安装了 Python2.3，JRE7，Eclipse，Android SDK，7-Zip 等必要的工具和环境。程序框架使用 Python 实现，APK 反编译模块中的解压程序以及 APK 编译模块中的压缩程序均使用 7-Zip，敏感 API 调用日志输出格式为 XML 文件，输出位置为 SD 卡。选取 Android 软件 SendMessage.apk，该软件一旦运行会自动在后台发送短信，通过检验该软件经过本系统处理后能否成功监控其中发送短信的敏感 API，来验证本系统的有效性。

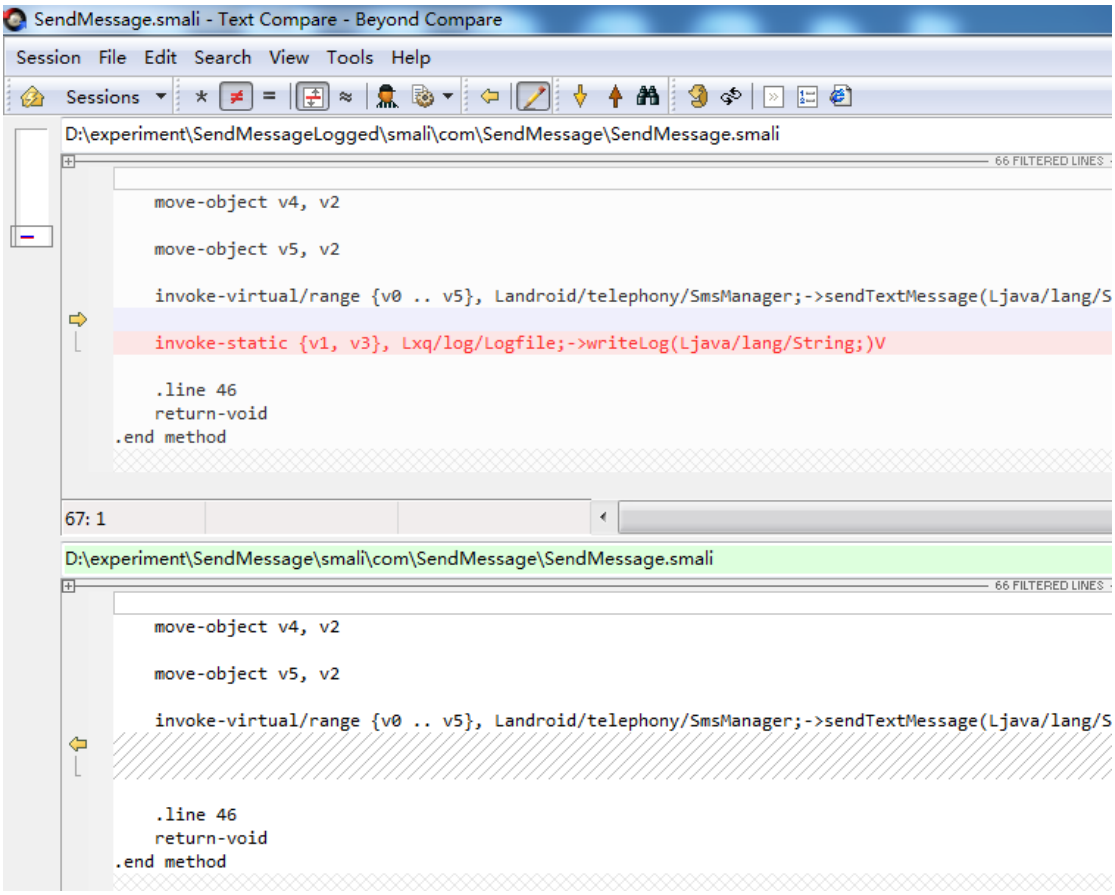

图 3 日志插入前后 smali 文件对比图

Fig. 3 Comparison between smali files after log module embedded

运行脚本 main.py，对 SendMessage.apk 生成新的 Android 软件 SendMessageLogged.apk。对比 SendMessage.apk 中 classes.dex 对应的 smali 文件与 SendMessageLogged.apk 中的 classes.dex 对应的 smali 文件，发现在 sendTextMessage 函数后，后者比前者多了一条短信内容记录 smali 代码，如图 3 所示。

将 SendMessageLogged.apk 安装到手机上，运行，在手机 SD 卡上生成了 behaviourLog.xml，如图 4 所示。

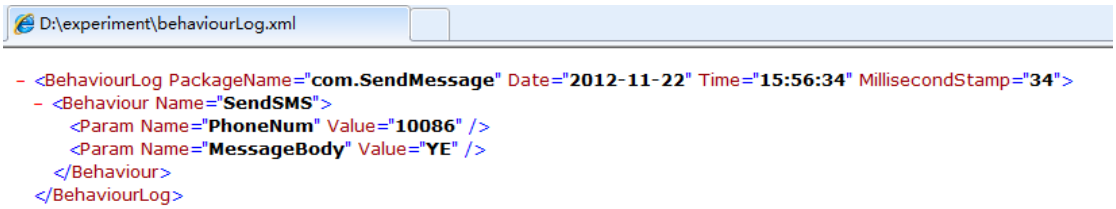

图 4　发送短信的敏感 API 的日志图

Fig. 4　Log of API of sending message

证明调用日志模块被成功嵌入，并且成功地生成了日志。

4　结论

在 Android 平台对恶意软件进行动态分析时，为解决 Android 系统提供的敏感 API 调用日志信息不足的问题，本文利用 Android 文件特点以及 Google 提供的 baksmali 和 smali 工具，设计并实现了基于 smali 的 Android 软件敏感 API 调用日志模块嵌入系统，实验结果证明，经过本系统嵌入日志模块的软件，在运行时能够成功记录软件敏感 API 调用的各种信息，为动态分析恶意软件提供了丰富的输入。

[参考文献] (References)

[1] Gartner. Gartner Says Worldwide Sales of Mobile Phones Declined 3 Percent in Third Quarter of 2012; Smartphone Sales Increased 47 Percent[OL]. [2012-11-14]. http://www.gartner.com/it/page.jsp?id=2237315
[2] 北京网秦天下科技有限公司. 网秦 2012 年第三季度全球手机安全报告[OL]. [2012-11-1]. http://cn.nq.com/neirong/2012Q3.pdf
[3] Google. smali[OL]. [2009-5-9]. http://code.google.com/p/smali/
[4] 杨丰盛. Android 技术内幕[M]. 北京：机械工业出版社，2011.
[5] Google. TypesMethodsAndFields[OL]. [2009-5-9]. http://code.google.com/p/smali/w/list
[6] Google. Registers[OL]. [2009-5-9]. http://code.google.com/p/smali/wiki/Registers
[7] Jeff Six. Application Security for the Android Platform[M]. Beijing: O'Reilly Media, 2011.