网络出版时间：2016-03-18 15:31:52
网络出版地址：http://www.cnki.net/kcms/detail/42.1658.N.20160318.1531.012.html

DOI：10.13245/j.hust.160312

抗混淆的 Android 应用相似性检测方法

王兆国[1] 李城龙[2] 关 毅[1] 薛一波[3]
(1 哈尔滨工业大学计算机科学技术学院，黑龙江 哈尔滨 150006；2 国家计算机网络应急技术处理协调中心，北京 100029；3 清华大学清华信息科学与技术国家实验室，北京 100084)

摘要 为了对抗混淆问题，更好地应对相似性检测需求，基于抗混淆的7种应用代码特征，以及抗混淆的音频和图片文件特征，提出一种新型的抗混淆相似性检测和评估方法. 并基于该方法实现了可满足大数据分析环境的原型系统. 通过该系统对1 259款恶意应用和12个应用商店的288款应用进行分析，结果表明：该方法可有效对抗混淆攻击，完成同源性分析，依据同源性分析结果可对零日恶意应用进行识别. 实验证明该方法可有效对抗代码混淆给相似性检测带来的问题，特征选取具有全局性，效果优于同类方法.

关键词 Android应用；抗混淆特征；相似性检测；恶意软件分析；声纹特征；代码特征
中图分类号 TP391 **文献标志码** A **文章编号** 1671-4512(2016)03-0060-05

Anti-obfuscation method for detecting similarity of Android application

Wang Zhaoguo[1] *Li Chenglong*[2] *Guan Yi*[1] *Xue Yibo*[3]
(1 School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150006, China; 2 National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China; 3 Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract In order to confront the code obfuscation problem, and better respond to the needs of similarity detection, a new method based on anti-obfuscation characteristics was designed, which included 7 kinds of anti-obfuscation code characteristics, audio characteristics and image characteristics. The method was proposed for detecting similarity and evaluating. Based on these characteristics and the methods, a prototype system was implemented and made suitable for large-scale scenario. The detections of 1259 malware and 288 APPs from 12 APP markets show that the proposed method has obvious advantages over the state-of-the-art methods against code obfuscation with high detection performance.

Key words Android application; anti-obfuscation; detecting similarity; malware analysis; voiceprint characteristic; code characteristic

移动互联网产业的蓬勃发展，推动电子商务、金融、服务等行业发生巨大变革，其市场潜力巨大、产业发展前景可观. 海量应用服务带来便捷的同时，盗版应用与Android恶意应用问题严重，侵害了开发者和用户的利益. 据360报告显示[1]，国内应用盗版现象非常严重，平均每项正版应用有26.3项盗版. 另一份报告显示[2]，2014全年Android用户感染恶意程序 3.19×10^9 人次，其中嵌入已知恶意代码和重打包应用为主要威胁. 针对目前嵌入已知恶意代码和重打包应用带来的安全问题，国内外研究者提出了一系列解决方法，其中通过检测应用相似性来识别嵌入已知恶意代码应用和重打包应用是目前研究的主流方向. Android应用相似性检测方法主要有基于静态特征码[3]的

收稿日期 2015-08-06.
作者简介 王兆国(1986-)，男，博士研究生，E-mail：wangzhaoguo@tsinghua.edu.cn.
基金项目 国家科技支撑计划资助项目(2012BAH46B04).

检测（如 n-gram[4]）、基于行为的检测及基于文件的检测[5-6]. 这些相关研究采用反编译工具或者动态行为分析工具得到应用软件不变的特征序列（即胎记特征），以达到对该软件或该软件家族的识别，若两项软件的胎记特征之间具有较大的相似度，则其中一项软件很可能是另外一项软件的盗版，或者属于同一家族. 但是行为序列特征的提取容易受到代码混淆技术的影响，导致代码重用、剽窃和重打包等恶意行为无法被有效检测. 为了克服代码混淆给软件相似性检测带来的困扰，实现 Android 平台下大规模应用的相似性自动化检测，本课题研究了 Android 应用混淆时软件特征的变化规律，提出一种抗混淆的应用软件相似性检测方法，该方法选取抗混淆的资源文件与代码特征，通过计算高维空间距离，进而判别应用软件的家族. 该方法可以有效抗代码混淆完成相似度分析，实现检测的自动化，大幅度提高检测效率.

1　抗混淆的相似性检测方法

1.1　相关定义

为了减少二义性，这里首先明确代码混淆、软件特征和抗混淆特征的相关定义.

定义 1　代码混淆. 基于 Java 的 Android 应用，其 class 文件中包含多种源码信息，如变量名、方法名等信息. Android 应用很容易被反编译工具逆向获取源码，修改后在应用市场发布盗版软件. 为了对抗盗版，通过使用 Java 混淆器将 Java 字节码中的所有变量、函数、类的名称用简短的无意义英文字母代替，且保证执行结果也与混淆前一样，这项混淆过程称为代码混淆.

定义 2　软件特征. 软件特征即提取软件中的不变特征（或关键特征），如方法类继承关系、字节码、操作码及图片信息等. 通过软件特征之间的相似度计算，达到对该软件或该软件家族的识别. 软件特征可用于检测软件盗版或阻止恶意软件的破坏.

定义 3　抗混淆特征. 通过混淆器可改变软件代码结构和代码内容，抗混淆特征即在代码混淆条件下，选取具有较高的鲁棒性、不易受到混淆攻击影响的特征，用于应用软件的相似性检测.

1.2　抗混淆相似性检测方法

1.2.1　抗混淆特征选择

如今应用软件种类繁多且更新频繁，特征的选取和检测效率尤为关键. 为了准确、有效地通过文件内容分析应用相似性，并符合实际的检测需求，抗混淆特征的选取应满足如下三个方面的要求：具有合适的抗混淆特征，以解决代码混淆下的软件相似性检测问题；适应大数据运算，能快速应对 Android 应用种类繁多、更新频繁问题，并满足检测准确性；可适用于高效的特征提取的准确的相似性计算算法.

基于上述特征选取的要求，通过大量分析 Android 应用软件资源组成，得到 Android 应用资源，主要包含可执行字节码文件、证书文件和资源文件（如 XML 文件、图片文件、音频文件等）. 文献[7]对大量 Android 应用进行了统计分析，选取应用软件签名文件、布局文件和图片文件作为抗混淆特征，但其忽略了可执行字节码文件. 然而盗版软件和恶意程序主要针对代码层面的复制、植入恶意程序，因此抗混淆特征必须包含代码层面的特征.

为了更好地展示代码混淆技术对应用软件相似性检测的影响，选取抗混淆的相似性特征，通过反编译工具查看混淆前后代码，其函数名称和变量名被混淆为无意义字符. 进一步分析函数的调用关系得知：由于受到代码混淆影响，因此代码变量名被无意字符替换，使得传统相似性计算方法不再适用. 通过对大量代码混淆工具进行调研可知：若代码混淆技术只替换字串或加入少量无用字串，则对函数之间的调用关系改变甚微.

基于上述分析，为了在代码混淆条件下满足特征的全局性原则，选取文件特征和字节码特征，文件特征选取音频文件和图片文件. 字节码特征从函数的调用关系入手，以函数为节点、函数之间的调用关系为边，构建 Android 应用的函数调用关系图. 将字节码特征相似性计算问题，转化为图的相似性计算分析. 依据图的属性特征，选取关系图的节点数、边数量、平均度（average degree）、网络直径（average path length）[8]、模块化（modularity）[9]、连接组件（connected components）及平均路径长度等属性作为代码层面的抗混淆特征，每个特征经过特征提取算法计算后为浮点数值.

1.2.2　特征文件提取与相似度计算

Android 应用安装文件（Android package，APK）为资源打包文件，通过解压缩可以得到应用中包含的音频、图片文件和 dex 文件. 将每个 Android 应用称为样本，每个样本包含的 p 项特征，即 p 项指标，由此构建高纬度距离空间，抽象每个 Android 应用为 p 维空间里的点，n 项应用就组成 p 维空间中的 n 项点，通过高维空间的距离计算方法度量 n 项样本间的接近程度.

用 d_{ij} 表示第 i 项样本与第 j 项样本之间的明氏距离,一切距离应满足如下条件:

a. 对于一切 i 和 j,$d_{ij}\geqslant 0$;

b. $d_{ij}=0$,等价于 i 样本和 j 样本指标相同;

c. 对于一切 i 和 j,$d_{ij}=d_{ji}$;

d. 对于一切 i,j 和 k,$d_{ij}\leqslant d_{ik}+d_{kj}$.

用样本集合提取的特征构建 $n\times p$ 维矩阵

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \cdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix},$$

式中 X_{pn} 为第 p 个样本的第 n 个特征.则明氏距离为

$$d_{ij}=\left(\sum_{l=1}^{p}|x_{il}-x_{jl}|^{k}\right)^{1/k}. \tag{1}$$

为了简化计算,设定 $k=1$,将距离计算转化为明氏绝对距离

$$d_{ij}=\left(\sum_{l=1}^{p}|x_{il}-x_{jl}|\right). \tag{2}$$

通过明氏绝对距离计算样本之间的距离,从而完成应用之间的相似性检测,而样本距离依据选取的特征属性计算获得.实际检测过程中不同属性在样本空间的权重不一样,即它们与类别的关联度不同,因此有必要对选取的属性赋一定的权重.依据 weka[10] 工具中 InfoGain AttributeEval评价模型,应用关联度学习模型和 Ranker 搜索策略方法完成代码层面的特征权重分配计算.InfoGainAttributeEval 与决策树构造原理相似,通过分析计算将拥有的信息量大的节点赋予较高的权重.本研究提出的抗混淆特征来源于代码、图片和音频文件,三类特征权重不同,按照样本间距离最大化原则赋予不同的权重,权重分配比例为

$$U=R_{\mathrm{A}}\left[\sum_{i=1}^{2}(R_{u_i}u_i)\right]+R_{\mathrm{B}}\left[\sum_{i=3}^{9}(R_{u_i}u_i)\right],$$

式中:R_{A} 和 R_{B} 分别为文件特征和代码特征权重,且 $R_{\mathrm{A}}+R_{\mathrm{B}}=1$;$u_i$ 为第 i 项特征;R_{u_i} 为第 i 项特征权重,且 $\sum_{i=1}^{2}R_{u_i}=1$,$\sum_{i=3}^{9}R_{u_i}=1$.抗混淆特征实验中选取图片特征、音频特征、节点数、边数量、平均度、网络直径、模块化、连接组件和平均路径长度作为相似性检测特征.这里采用特征权重排序算法来可提高检测的正确性.通过 InfoGainAttributeEval 算法计算 R_{B} 的权重分配.根据大量分析统计实验,在保证样本间距离最大原则前提下,用户相似性计算的权重分配如下所示:$R_{u_1}=0.6$,$R_{u_2}=0.4$,$R_{u_3}=0.162\ 1$,$R_{u_4}=0.121\ 8$,$R_{u_5}=0.157\ 0$,$R_{u_6}=0.147\ 0$,$R_{u_7}=0.080\ 0$,$R_{u_8}=0.132\ 4$,$R_{u_9}=0.199\ 1$.

2 抗混淆系统实现

提出一种抗混淆的 Android 应用相似性检测方法,其系统架构如图 1 所示.本系统基于在线检测和线下训练更新特征库两个部分.线上系统由用户提交应用,系统自动提取特征文件,并依据上文相似度计算方法完成相似性比对,并生成报告.

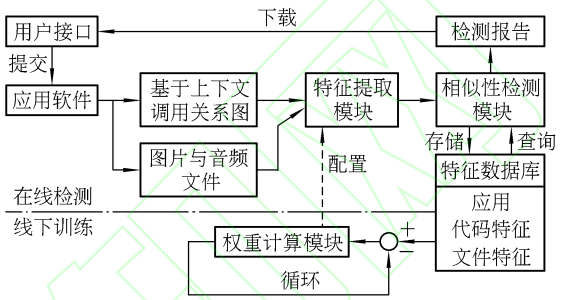

图 1 抗混淆相似性检测方法系统架构

系统实现中抗混淆特征的提取是本研究重点,下面讨论代码特征提取、图片特征提取以及音频特征提取.

2.1 代码特征提取

代码混淆技术的目的是改变代码的特征以增加逆向工程师对代码阅读的难度.盗版者和恶意代码的利用者都可以利用该技术,基于已有样本完成代码修改和重打包工作,这无疑给检测恶意软件和防盗版工作增加了难度.这里通过分析主流的代码混淆技术,寻求抗混淆的代码特征.该特征选取须忽略代码名称,并且保留代码宏观上下文的调用关系,用于代码级别的相似性检测.首先系统自动构建 Android 应用的 API(应用程序编程接口,application programming interface)调用关系图,再利用 Gephitoolkit[11] 选取 API 调用关系树的节点数、边数量、平均度、网络直径、模块化、连接组件和平均聚类系数等属性,作为抗混淆特征.

2.2 图片特征提取算法

Android 应用软件的图片重打包主要目的是盗版或者诱骗用户安装恶意软件,图片修改主要包含替换图片、删除图片、修改原图片、改变图片清晰度或颜色等方面.图片相似度计算研究广泛,但是大部分图片处理算法具有较高的时间和空间复杂度,尤是其特征的选择直接影响算法匹配的效率.基于图片的相似度计算,采用感知哈希算法[12]完成图片的相似度计算,具体流程如下.

a. 除开图片细节,仅保留结构、明暗等基本

信息以对抗图片尺寸变换攻击，将图片缩小到8×8像素；

b. 为了消除图片颜色混淆而影响相似度检测，将其转为 64 级灰度；

c. 计算平均灰度值；

d. 将每个像素的灰度与平均值进行比较，大于等于平均值则记为 1，否则记为 0，由此构建图片 64 bit 二进制特征值；

e. 依据上述方法提取其本应用中其他图片特征，得到该应用的图片的特征.

根据前人研究经验，将差异阈值小于 5 标记为相似图片.

2.3　声纹特征提取算法

移动智能业务极大推动了声纹识别技术的发展，基于声纹的相似性检测应运而生. 在 Android 应用中包含的音频文件通常受到替换、文件截取、音质变化等混淆攻击. 为了满足抗混淆特性，且适用于大数据环境，基于 echoprint 算法[13]完成音频的相似性检测. 该算法可以有效地抗混淆，仅需几秒钟的音频段即可完成相似性检测. 主要算法流程：通过时间滑动窗口截取音频段；通过对音频的傅里叶变换得到频域特性；选取频域极值与其对应的时间节点作为特征；比较特征进行相似性判断.

3　实验与评估

3.1　实验环境与实验组成

为了验证基于隐私窃取行为链的检测方法的有效性和正确性，这里对比了其他 Android 恶意应用检测方法. 实验在内存 8 GB，i3-2120CPU@3.3 GHz 的机器上完成，检测算法和爬虫由 Java 语言实现. 实验样本使用文献[14]提供的 1 259 款典型恶意应用，以及国内外 12 个 Android 应用商店中的 TOP288 应用. 实验由代码混淆验证、恶意应用同源性分析、有效性分析、可用性分析几部分组成.

3.2　实验评估

3.2.1　代码混淆验证

选取主流的三种混淆工具对目标应用进行混淆，验证本文方法的抗代码混淆效果，其相似性检测结果如表 1 所示，表中 n_1 和 n_2 分别为正确识别数量和检测数量.

ProGuard[15]为一个压缩、优化和混淆 Java 字节码文件的免费工具，可以删除无用的类、字段、方法和属性，也可以使用简短的无意义名称来重命名已经存在的类、字段、方法和属性. DashO-Pro[16]用于对 Java 平台应用程序的代码混淆进行保护. APKProtect[17]用于混淆 APK 文件中字符串、流程、类名等，从而保护 APK 文件，使之不易分析. 本实验选取了 12 个应用商店，即 360，googleplay，official、百度手机助手、联通沃商店、木蚂蚁应用市场、搜狗商店、淘宝手机助手、腾讯应用宝、天翼空间、豌豆荚和小米应用商店，并分别抓取腾讯 QQ、新浪微博、淘宝、微信、酷我音乐盒、百度地图、爱奇艺和墨迹天气 8 款软件，作为 8 类应用.

表 1　抗混淆验证实验相似性检测结果

应用分类	ProGuard		DashO-Pro		APKProtect	
	n_1	n_2	n_1	n_2	n_1	n_2
腾讯 QQ5.4.1	11	12	11	12	12	12
新浪微博 5.1.2	12	12	12	12	12	12
微信 6.1.0.66	11	12	9	12	10	12
淘宝 5.2.6	11	12	11	12	11	12
酷我音乐盒 8.0.2.2	12	12	11	12	12	12
百度地图 8.4.0	12	12	11	12	12	12
爱奇艺 4.0.0.32	11	12	10	12	11	12
墨迹天气 1.3.1.2	12	12	10	12	12	12

由表 1 可见：将 12 个平台下载的 8 种软件，分别用三种代码混淆工具混淆，生成混淆后的软件共 288 款. 结果表明本文方法正确率为93.4%，由此证明本文方法可以有效抗代码混淆.

3.2.2　恶意应用同源性分析

由文献[1]知 3.19×10^9 次恶意感染程序主要来源于嵌入已知恶意代码和重打包应用. 这里提出应用同源性分析实验，通过提取待检测应用特征并比对样本库中应用的相似性，完成 0-day 应用的检测和识别.

为了验证本文方法相似性检测同源性分析结果，选取 11 类常见的恶意代码族应用，分别为 ADRD，AnserverBot，BaseBridge，DroidKungFu1，Geinimi，KMin，Pjapps，SndApps，YZHC，zHash 和 Zsone[14]，设计交叉验证实验，每个应用依次与同族应用进行相似性计算，完成同源性分析实验，如表 2 所示.

将本研究检测结果与文献[14]进行对比可知：11 类恶意软件集合共 599 个应用，正确识别 517 项，其检测结果的正确率超过 86.3%. 进一步人工校正分析表明：漏检应用中绝大部分为不同软件，包含相同小部分恶意代码，按照本文方法检

测出来的结果为不相似.

表 2 恶意代码族应用同源性分析与检测耗时结果

应用	总数量	正确数量	平均耗时/ms
ADRD	22	14	239.4
AnserverBot	187	181	407.1
BaseBridge	122	105	114.0
DroidKungFu1	34	26	292.7
Geinimi	69	53	169.2
KMin	52	42	143.0
Pjapps	58	43	994.3
SndApps	10	10	80.5
YZHC	22	20	147.4
zHash	11	11	746.8
Zsone	12	12	252.8

3.2.3 有效性分析

APP应用相似性检测效率是衡量检测系统的重要指标,这里通过分析音频、图片与代码文件特征验证本文方法的有效性,具体为:**a.** 音频文件相似性检测有效性分析. 基于echoprint算法对音频文件生成特征文件与音频时长成线性关系,如某APP中时长143 s,大小1.34 MB的MP3音频格式生成指纹编码56 KB,用时3.781 s;**b.** 图片文件相似性检测有效性分析. 基于感知哈希算法对图片文件完成相似性检测,通过实验对多个APP中200个图片文件,提取特征时间为9.8 s,指纹匹配效率为1×10^5次耗时约10 ms,检测效率极高;**c.** 代码文件相似性检测有效性分析. 基于Androguard工具中androgexf.py脚本来生成APK的GEXF格式的图形文件,基于生成的GEXF文件提取应用的代码特征,对上节11类恶意软件特征进行提取的平均耗时结果如表2所示.

3.2.4 可用性分析

将本文方法与文献[4-7]进行比较,分析其算法特点,这里从是否抗混淆分析、是否为上下文语义特征、是否可以同源性分析、是否考虑代码混淆及是否包含布局文件分析五个维度进行评价,其结果如表3所示.

表 3 本文方法与文献[4-7]的比较

项目	方法				
	文献[4]	文献[5]	文献[6]	文献[7]	本文
抗混淆分析	否	否	否	是	是
上下文语义	否	否	否	否	是
同源性分析	否	否	是	是	是
考虑代码混淆	否	否	否	否	是
包含布局文件	否	否	否	是	否

4 结语

本文提出了一种基于文件内容特征和代码特性的抗混淆Android应用相似性检测方法,该方法选取具有全局性的属性特征,并提出满足大数据环境下相似性检测系统框架,以及特征提取算法与相似性计算算法. 通过1 259款恶意应用与12项平台288个Android应用的实验验证,证明本文方法可以在大量应用中有效识别重打包应用和嵌入已知恶意代码的应用,有效地抵抗已知的文件混淆技术,在运算效率和准确性上优于现有解决方案. 然而本文方法也有其局限性. 这里基于逆向工具完成Android应用解压特征文件提取,未考虑到高级加壳技术或本地C代码调用导致的Android应用软件无法分析的情况. 并且本文方法依赖于已知样本空间覆盖,为未知样本空间无法涉及. 在恶意软件同源性分析实验中,当两款不同软件均嵌入小部分相同代码而代码结构未改变时,按照本文方法检测的效果不佳.

参 考 文 献

[1] 360公司. 2014年Q2移动互联网App分发行业报告[EB/OL]. [2015-03-01]. http://dev.360.cn/da/hybgq2.html.

[2] 360公司. 2014年中国手机安全状况报告[EB/OL]. 2015-03-01. http://ad8n468m37,17,yunpan.cn/1k/cyCT3CtHPnITT.

[3] Mahmood Y, Sarwar S, Pervez Z, et al. Method based static software birthmarks: a new approach to derogate software piracy[C]// Proc of 2nd International Conference on Computer, Control and Communication. Karachi: IEEE, 2009: 149-155.

[4] Lu B, Liu F, Ge X, et al. Feature n-gram set based software zero-watermarking[C]// Proc of International Symposiums on Information Processing. Moscow: IEEE, 2008: 607-611.

[5] Zhou W, Zhou Y, Jiang X, et al. Detecting repackaged smartphone applications in third-party android marketplaces[C]// Proc of 2nd ACM Conf on Data and Application Security and Privacy. New York: ACM, 2012: 317-326.

[6] Li S. Juxtapp and DStruct: detectiong of similarity among Android applications[EB/OL]. [2015-01-20]. http://www.xiph.org/ogg/.

[7] 罗养霞,房鼎益. 基于聚类分析的软件胎记特征选择[J]. 电子学报, 2013, 41(12): 2334-2338.

(下转第76页)

public key encryption scheme[C]// Proc of Post-Quantum Cryptography. Berlin: Springer-Verlag, 2014: 229-245.

[8] Wang H Z, Shen C X, Xu Z Q, et al. Multivariate public key encryption scheme based on error correcting codes[J]. China Communications, 2011, 8(4): 23-31.

[9] Niederreiter H. Knapsack-type cryptosystems and algebraic coding theory[J]. Problems of Control and Information Theory Problem, 1986, 15(2): 159-166.

[10] Gaborit P, Ruatta O, Schrek J, et al. New results for rank based cryptography[M]. Berlin: Springer, 2014.

[11] Gaborit P, Murat G, Ruatta O, et al. Low rank parity check codes and their application to cryptography[C]// Proc of International Workshop on Coding and Cryptography (WCC 2013). Bergen: Selmersenteret, 2013: 168-180.

[12] Heyse S, Von M I, Güneysu T. Smaller keys for code based cryptography: QC-MDPC McEliece implementations on embedded devices[M]. Berlin: Springer-Verlag, 2013.

[13] Baldi M. QC-LDPC code-based cryptography[M]. Berlin: Springer, 2014.

(上接第64页)

[8] Brandes U. A faster algorithm for betweenness centrality[J]. Journal of Mathematical Sociology, 2001, 25(2): 163-177.

[9] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks[J]. Journal of Statistical Mechanics: Theory and Experiment, 2008(10): 10008-10020.

[10] Hall M, Frank E, Holmes G, et al. The WEKA data mining software: an update[J]. ACM SIGKDD Explorations Newsletter, 2009, 11(1): 10-18.

[11] Gephi. Gephi toolkit[EB/OL]. [2015-03-01]. http://gephi.github.io/toolkit/.

[12] 张慧,张海滨,李琼,等. 基于人类视觉系统的图像感知哈希算法[J]. 电子学报, 2008, 36(S1): 30-34.

[13] EchoNest. Echoprint[EB/OL]. [2015-03-01]. http://echoprint.me/.

[14] Zhou Y, Jiang X. Dissecting android malware: Characterization and evolution[C]// Proc of 2012 IEEE Symposium on Security and Privacy (SP). Oakland: IEEE, 2012: 95-109.

[15] ProGuard. ProGuard software version5.2[EB/OL]. [2015-03-01]. http://gephi.github.io/toolkit/.

[16] DashO. PreEmptive protection[EB/OL]. [2015-03-01]. http://www.preemptive.com/products/dasho/overview/.

[17] SharkTeam. ApkProtect[EB/OL]. [2015-03-01]. https://github.com/SharkTeam/ApkProtect.