(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104778409 B
(45)授权公告日 2018.01.12

(21)申请号 201510179551.3

(22)申请日 2015.04.16

(65)同一申请的已公布的文献号
申请公布号 CN 104778409 A

(43)申请公布日 2015.07.15

(73)专利权人 电子科技大学
地址 611731 四川省成都市高新区(西区)西源大道2006号

(72)发明人 陈瑞东 张小松 牛伟纳 戴中印 鲍凯 漆艳梅 于洲 王东 刘小垒

(74)专利代理机构 电子科技大学专利中心 51203
代理人 李明光

(51)Int.Cl.
G06F 21/56(2013.01)

(56)对比文件
CN 104484607 A,2015.04.01,
CN 104091121 A,2014.10.08,
CN 103440459 A,2013.12.11,
CN 103473346 A,2013.12.25,
赵幸.Android平台恶意应用程序行为分析与研究.《中国优秀硕士学位论文全文数据库信息科技辑》.2014,正文第2.1.2,2.1.3,2.2.3,2.3,3.2,4.1,4.2.1-4.2.4节,图2-2,3-1,3-2,3-4,4-2,4-3,4-5.

审查员 陈玲

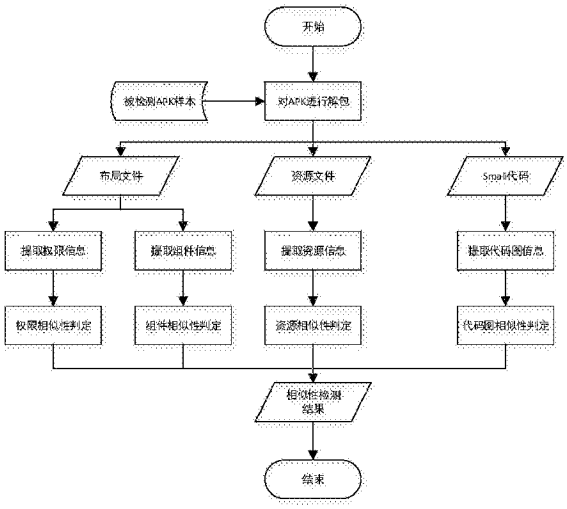
权利要求书2页 说明书4页 附图5页

(54)发明名称

一种Android应用软件相似性的检测方法及装置

(57)摘要

本发明提供一种针对移动平台Android系统下应用软件相似性的检测方法及装置,属于计算机安全领域,具体涉及从应用程序权限、组件、资源及代码图四个方面对通过对正常应用软件使用重打包技术注入恶意代码或广告代码的软件进行检测的方法及装置。所述的方法包括Android应用软件解包、反编译,提取权限信息、组件信息、资源信息与代码图信息,并根据上述信息进行相似性的判定;所述的装置包括标准软件库模块、爬虫模块、信息提取模块、相似性判断模块。本发明提供的检测方法及装置,可以快速准确地检测出重打包的Android应用软件,保护Android应用软件开发者与用户的利益。



CN 104778409 B

1.一种Android应用软件相似性的检测方法,包括以下步骤:

步骤1.将待检测的应用软件安装包进行解包反编译,获得代码、布局与资源文件;

步骤2.从解包反编译后的布局文件中提取应用程序的权限信息,并将该待检测应用软件的权限与目标软件的权限进行相似性检测;

步骤3.从解包反编译后的布局文件中提取应用程序的组件信息,并将该待检测应用软件的组件与目标软件的组件进行相似性检测;

步骤4.从解包反编译后的资源文件中提取应用程序的资源信息,并将该待检测应用软件的资源与目标软件的资源进行相似性检测;

步骤5.从解包反编译后的代码文件中提取代码图信息,并将该待检测应用软件的代码图与目标软件的代码图进行相似性检测;

步骤6.若所述待检测应用软件的权限、组件、资源及代码图均与目标软件相应的权限、组件、资源及代码图相似,则判定该待检测应用软件由所述的目标软件重打包而得。

2.根据权利要求1中所述的Android应用软件相似性的检测方法,其特征在于,所述解包反编译过程具体包括解包过程和反编译过程;所述解包过程指从应用软件安装文件中得到编译后的代码文件classes.dex、编译后的资源文件resources.arsc、编译后的布局文件AndroidManifest.xml;所述反编译过程指从上述已编译的文件获得相应的未编译的文件,即获得smali代码文件、各xml资源文件以及AndroidManifest.xml布局文件。

3.根据权利要求2中所述的Android应用软件相似性的检测方法,其特征在于,所述的提取权限信息是指从AndroidManifest.xml文件中提取出<uses-permission>节点的字符串,该字符串描述了应用软件所申请的权限,将提取的字符串构建成该待检测应用软件的权限集合;

所述的权限相似性检测过程具体如下:

记目标应用软件的权限集合为P1,待检测应用软件的权限集合为P2,若满足关系$P1 \subseteq P2$,则判定待检测应用软件的权限与目标应用软件的权限相似。

4.根据权利要求2中所述的Android应用软件相似性的检测方法,其特征在于,所述的提取组件信息是指从AndroidManifest.xml文件中提取出<activity>、<service>、<receiver>节点的字符串,上述三个字符串分别描述了应用软件中的Activity组件、Service组件与BroadcastReceiver组件,将提取的字符串构建该待检测应用软件的组件集合;

所述的组件相似性检测过程具体如下:

记目标应用软件的组件集合为C1,待检测应用软件的组件集合为C2,若满足关系$C1 \subseteq C2$,则判定待检测应用软件的组件与目标应用软件的组件相似。

5.根据权利要求2中所述的Android应用软件相似性的检测方法,其特征在于,所述的提取资源信息是指从各xml文件中提取出字符串、布局、图片信息,字符串信息位于strings.xml文件,布局信息位于layout目录下的各xml文件,图片信息位于drawable目录,将提取的资源信息构建该待检测应用软件的资源集合;

所述的资源相似性检测过程具体如下:

记目标应用软件的资源集合为R1,待检测应用软件的资源集合为R2,若满足关系

$R1 \subseteq R2$，则判定待检测应用软件的资源与目标应用软件的资源相似。

6.根据权利要求2中所述的Android应用软件相似性的检测方法，其特征在于，所述的提取代码图信息是指从smali代码文件中提取类、方法、域以及方法与类的包含关系、域与类的包含关系、方法与方法的引用关系、域与方法的引用关系，以方法与域作为图的两种节点，方法与域所属的类作为该节点的属性，方法与方法、方法与域的引用关系作为节点之间的边，形成一个描述了代码布局与关系有向图作为代码图；

所述的代码图相似性检测过程具体如下：

记目标应用软件的代码图为G1，待检测应用软件的代码图为G2，若G1为G2的子图，则判定待检测应用软件的代码图与目标应用软件的代码图相似。

7.一种实现如权利要求1所述的Android应用软件相似性检测方法的装置，包括标准软件库模块、爬虫模块、信息提取模块、相似性判断模块；

其特征在于，所述标准软件库模块用于存储Android平台各应用软件的官方版本，作为相似性检测的基准；

所述爬虫模块用于通过网络从各应用软件的官方网站下载软件的最新版本至标准软件库，保证标准软件库模块中的应用软件最新最全；

所述信息提取模块用于提取被检测样本与标准软件的权限、组件、资源以及代码图信息；

所述相似性判断模块用于对信息提取模块提取的各种信息与标准软件库模块中各软件官方版本的相应信息进行相似性判断并输出相似性检测结果：若所述待检测应用软件的权限、组件、资源及代码图均与标准软件库中某一标准软件相应的权限、组件、资源及代码图相似，则判定该待检测应用软件由所述的标准软件重打包而得。

一种Android应用软件相似性的检测方法及装置

技术领域

[0001] 本发明属于计算机安全技术领域，具体涉及一种针对移动平台Android系统下应用软件相似性的检测方法及装置。

背景技术

[0002] 随着移动互联网的发展，Android平台的新应用也层出不穷。智能移动设备的便利性使其成为了很多人生活、工作和学习不可获取的一部分，例如：购物、导航等。但是近几年针对Android应用软件使用重打包技术来对其注入恶意代码或广告代码的行为使我们不得不更多地考虑Android应用软件的安全问题。根据《腾讯移动安全实验室2014年上半年手机安全报告》，2014年上半年，全国Android病毒感染用户数达到8923.52万，是2012年全年Android手机染毒用户的3.68倍；2014年上半年Android手机染毒用户数是2013年上半年的2.28倍，同比增长128％。而这些Android恶意软件绝大多数是通过重打包技术注入到正常应用软件中的。

[0003] Android的安装时权限模型决定了程序一旦发布其权限就必然固定，可以在任何时刻进行审查，固定的权限也决定了程序的功能已经固定。在Android操作系统中，并没有传统意义上的进程，而是系统预定义的各种组件，这些组件实际上都是Android系统的回调模块。为了让系统回调这些模块，Android应用软件需要首先注册这些组件，而绝大部分都要求是显示的静态注册。Android应用软件的资源包括了字符串、图片、布局等。Android应用软件的代码布局可以使用图来表示，节点为方法与域，节点包含的属性为方法与域所属的类，边为方法与方法、方法与域的引用关系。

[0004] Android平台的应用软件使用Java语言编写，但Java语言的特性导致Android应用软件容易被逆向和破解，通过ApkTool等工具可以很容易获得Android应用软件的代码与资源文件。同时，Android平台允许应用程序的开发者使用自己的证书对安装包进行签名，也允许系统从第三方应用市场中安装应用。因此，当Android应用软件被逆向后，可以修改其代码与资源等文件，重新签名生成新的安装包，发布到第三方应用市场供用户安装使用。很多恶意软件与广告软件开发者利用该方法，将正常软件中植入恶意代码或广告代码，达到更加广泛的传播并欺骗用户安装使用的目的，以获取更多利益。

[0005] 由于恶意代码与广告代码都是一个相对独立的模块，为保证原应用软件的正常使用，重打包过程通常不会对原应用软件做大量修改，这使得对重打包植入恶意代码或广告代码的软件进行检测成为了可能。

发明内容

[0006] 本发明要解决的技术问题在于克服现有技术针对传统的以API作为关键词或对代码进行模糊哈希的方法来进行相似性检测的不足，提供一种基于权限相似性、组件相似性、资源相似性以及代码图相似性结合的Android应用软件相似性检测方法及装置，有效地检测通过重打包技术植入恶意代码或广告代码的Android应用软件，保护Android应用软件开

发者与用户的利益。

[0007] 本发明具体采用如下技术方案：

[0008] 一种Android应用软件相似性检测方法，用于判断待检测应用软件是否由目标软件重打包而得，其流程如图1所示，包括以下步骤：

[0009] 步骤1.将待检测的应用软件安装包进行解包反编译，获得代码、布局与资源文件；

[0010] 步骤2.从解包反编译后的布局文件中提取应用程序的权限信息，并将该待检测应用软件的权限与目标软件的权限进行相似性检测；

[0011] 步骤3.从解包反编译后的布局文件中提取应用程序的组件信息，并将该待检测应用软件的组件与目标软件的组件进行相似性检测；

[0012] 步骤4.从解包反编译后的资源文件中提取应用程序的资源信息，并将该待检测应用软件的资源与目标软件的资源进行相似性检测；

[0013] 步骤5.从解包反编译后的代码文件中提取代码图信息，并将该待检测应用软件的代码图与目标软件的代码图进行相似性检测；

[0014] 步骤6.若所述待检测应用软件的权限、组件、资源及代码图均与目标软件相应的权限、组件、资源及代码图相似，则判定该待检测应用软件由所述的目标软件重打包而得。

[0015] 步骤1中所述的对安装包进行解包反编译，具体包括解包过程与反编译过程；Android应用程序安装包即APK格式文件，是一种ZIP格式的压缩文件；所述解包过程指从应用程序安装包文件中得到编译后的代码文件classes.dex、编译后的资源文件resources.arsc、编译后的布局文件AndroidManifest.xml；所述反编译过程指从上述已编译的文件获得相应的未编译的文件，即获得smali代码文件、各xml资源文件以及AndroidManifest.xml布局文件。

[0016] 步骤2中所述的提取权限信息是指从AndroidManifest.xml文件中提取出<uses-permission>节点的字符串，该字符串描述了应用软件所申请的权限，将提取的字符串构建成该待检测应用软件的权限集合；所述的权限相似性检测如图4所示，记目标应用软件的权限集合集合为P1，检测应用软件的权限集合为P2，若满足关系$P1 \subseteq P2$，则判定待检测应用软件的权限与目标应用软件的权限相似。

[0017] 步骤3所述的提取组件信息是指从AndroidManifest.xml文件中提取<activity>、<service>、<receiver>节点的字符串，上述三个字符串分别描述了应用软件中的Activity组件、Service组件与BroadcastReceiver组件，将提取的字符串构建该待检测应用软件的组件集合；所述的组件相似性检测如图5所示，记目标应用软件的权限集合为P1，待检测应用软件的权限集合为P2，若满足关系$P1 \subseteq P2$，则判定待检测应用软件的权限与目标应用软件的权限相似。

[0018] 步骤4所述的提取资源信息是指从各xml文件中提取出字符串、布局、图片信息，字符串信息位于strings.xml文件，布局信息位于layout目录下的各xml文件，图片信息位于drawable目录，将提取的资源信息构建该待检测应用软件的资源集合；所述的资源相似性检测如图6所示，记目标应用软件的资源集合为R1，待检测应用软件的资源集合为R2，若满足关系$R1 \subseteq R2$，则判定待检测应用软件的资源与目标应用软件的资源相似。

[0019] 步骤5所述的提取代码图信息是指从smali代码文件中提取类、方法、域以及方法与类的包含关系、域与类的包含关系、方法与方法的引用关系、域与方法的引用关系，以方

法与域作为图的两种节点，方法与域所属的类作为该节点的属性，方法与方法、方法与域的引用关系作为节点之间的边，形成一个描述了代码布局与关系有向图作为代码图；所述的代码图相似性检测如图7所示，记目标应用软件的代码图为G1，待检测应用软件的代码图为G2，若G1为G2的子图，则判定待检测应用软件的代码图与目标应用软件的代码图相似。

[0020] 本发明还提供了一种实现上述Android应用软件相似性检测方法的装置，其结构如图2所示，包括标准软件库模块、爬虫模块、信息提取模块、相似性判断模块。

[0021] 所述标准软件库模块用于存储Android平台各应用软件的官方版本，作为相似性检测的基准；

[0022] 所述爬虫模块用于通过网络从各应用软件的官方网站下载软件的最新版本至标准软件库，保证标准软件库模块中的应用软件最新最全；

[0023] 所述信息提取模块用于提取被检测样本与标准软件的权限、组件、资源以及代码图信息；

[0024] 所述相似性判断模块用于对信息提取模块提取的各种信息与标准软件库模块中各软件官方版本的相应信息进行相似性判断并输出相似性检测结果：若所述待检测应用软件的权限、组件、资源及代码图均与标准软件库中某一标准软件相应的权限、组件、资源及代码图相似，则判定该待检测应用软件由所述的标准软件重打包而得。

[0025] 本发明的有益效果是：

[0026] 1、本发明从权限、组件、资源、代码图四个方面进行相似性判定，使得结果更加准确；

[0027] 2、本发明使用代码图，包含了代码中类、方法、域的关系，不受代码混淆的影响；

[0028] 3、本发明能够获得最新最全的标准软件，保证检测结果的有效性。

附图说明

[0029] 图1是本发明Android应用软件相似性检测方法流程图；

[0030] 图2是本发明Android应用软件相似性检测装置结构示意图；

[0031] 图3是本发明Android应用软件相似性检测方法具体实施流程图；

[0032] 图4是权限相似性示意图；

[0033] 图5是组件相似性示意图；

[0034] 图6是资源相似性示意图；

[0035] 图7是代码图相似性示意图。

具体实施方式

[0036] 下面结合附图1-7和具体实施方式对本发明一种Android应用软件相似性检测方法与装置作进一步的说明。

[0037] 实施例

[0038] 本实施例具体采用如下技术方案：

[0039] 一种Android应用软件相似性检测方法，其流程如图3所示，具体包括以下步骤：

[0040] S1.采用Android应用软件的包名与版本号作为Android应用软件的唯一标识；

[0041] S2.使用爬虫从各知名软件官网与官方应用市场中爬取下载Android应用软件原

始版本安装包，构成集合$Set_{APK}=\{APK_1,APK_2,APK_3,\cdots,APK_n\}$；

[0042] S3.从S2中获得的Android应用软件原始版本安装包中，提取各原始版本的包名Pkg与版本号Ver，构成APK的唯一ID=<Pkg,Ver>，对于每一个APK_m，都有对应的ID_m；

[0043] S4.将S2中获得的Android应用软件原始版本安装包解包，得到AndroidManifest.xml、classes.dex、resources.arsc文件，对其进行反编译，获得布局文件、smali代码文件与资源文件；

[0044] S5.从S4得到的布局文件、smali代码文件与资源文件中提取权限信息P、组件信息C、资源信息R与代码图G，共同作为Android应用软件相似性检测特征Sig=<P,C,R,G>；

[0045] S6.针对每一个Android应用软件安装包，都存在其对应的唯一ID与特征，将S2中的安装包与其对应的S3中的唯一ID和S5中的特征作为一个元组，构成标准软件库L的一个条目Item，对于APK_m，其条目为$Item_m=<APK_m,ID_m,Sig_m>$，其中$ID_m=<Pkg_m,Ver_m>$，$Sig_m=<P_m,C_m,R_m,G_m>$；

[0046] S7.对于提交的被检测样本APK_s，同样使用S3-S5所述方法，获得其ID_s与Sig_s；

[0047] S8.遍历标准软件库中的所有条目，若存在$Item_n\in L$，使得$P_n\subseteq P_s$，$C_n\subseteq C_s$，$R_n\subseteq R_s$，G_n为G_s的子图，则被检测样本APK_s为标准样本库中APK_n重打包得到的。

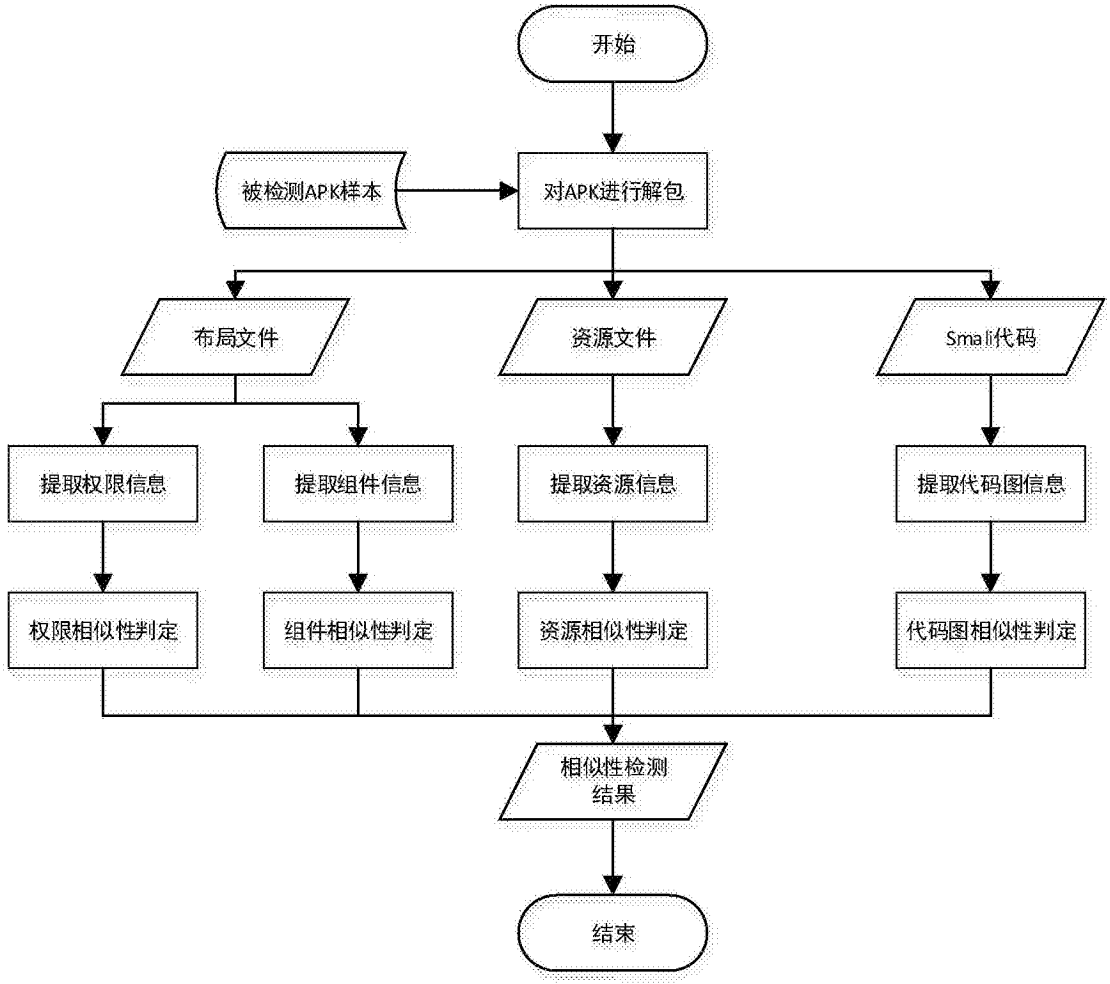

开始
被检测APK样本
对APK进行解包
布局文件
资源文件
Smali代码
提取权限信息
提取组件信息
提取资源信息
提取代码图信息
权限相似性判定
组件相似性判定
资源相似性判定
代码图相似性判定
相似性检测结果
结束

图1

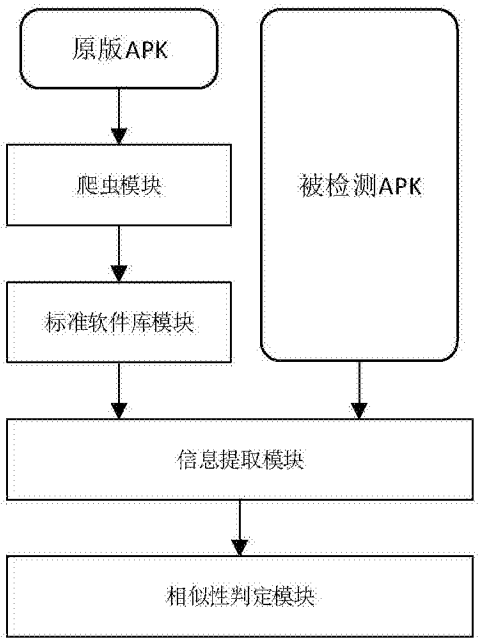


图2

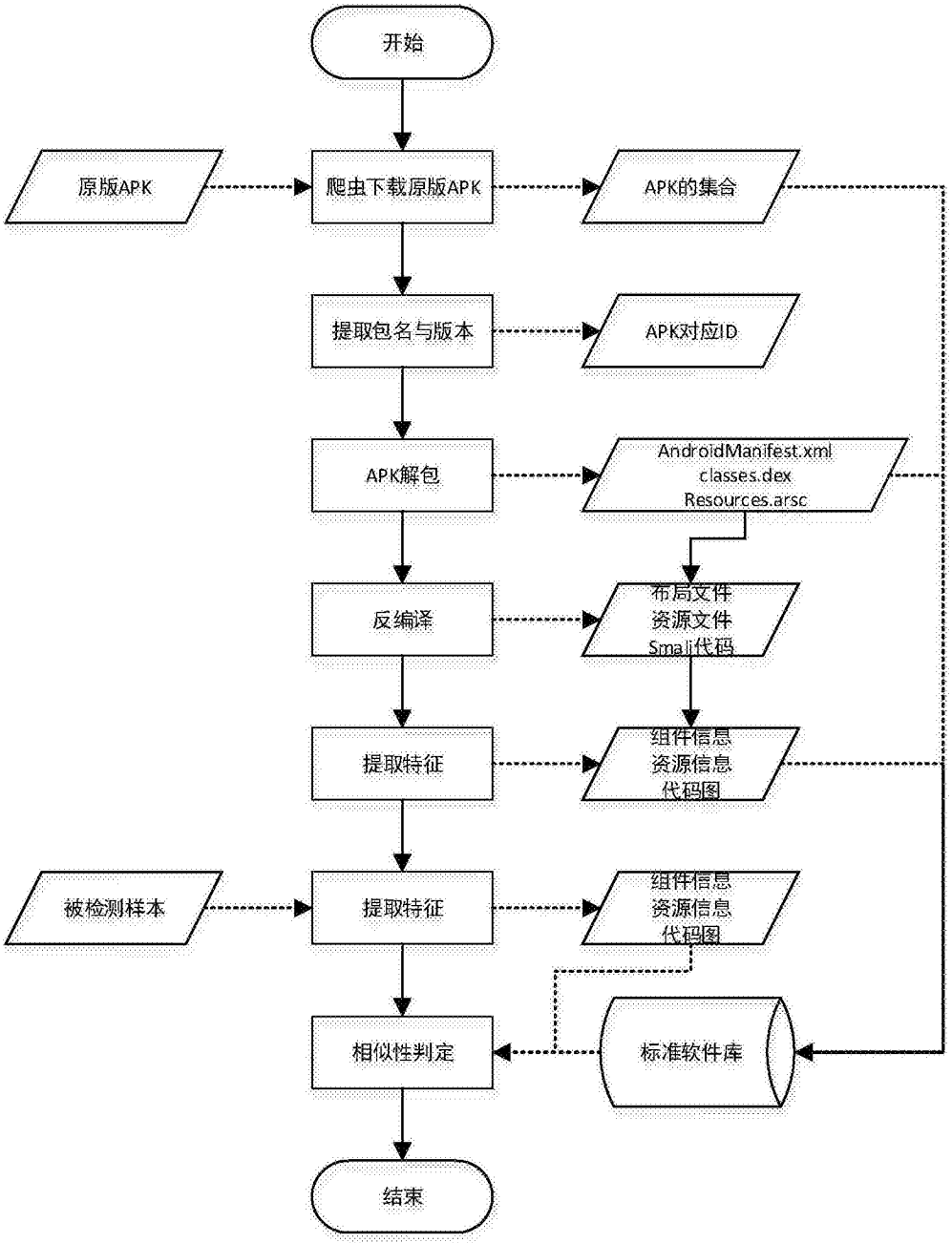

开始
原版APK
爬虫下载原版APK
APK的集合
提取包名与版本
APK对应ID
APK解包
AndroidManifest.xml
classes.dex
Resources.arsc
反编译
布局文件
资源文件
Smali代码
提取特征
组件信息
资源信息
代码图
被检测样本
提取特征
组件信息
资源信息
代码图
相似性判定
标准软件库
结束

图3

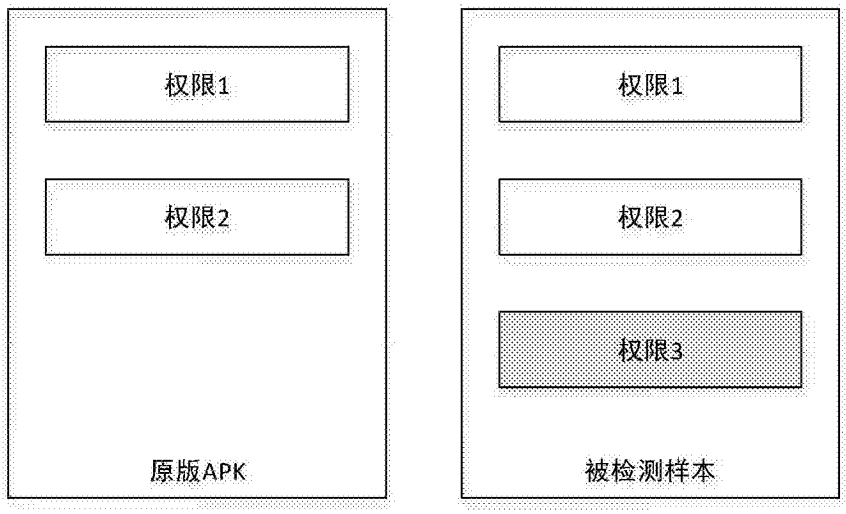

权限1
权限2
原版APK
权限1
权限2
权限3
被检测样本

图4

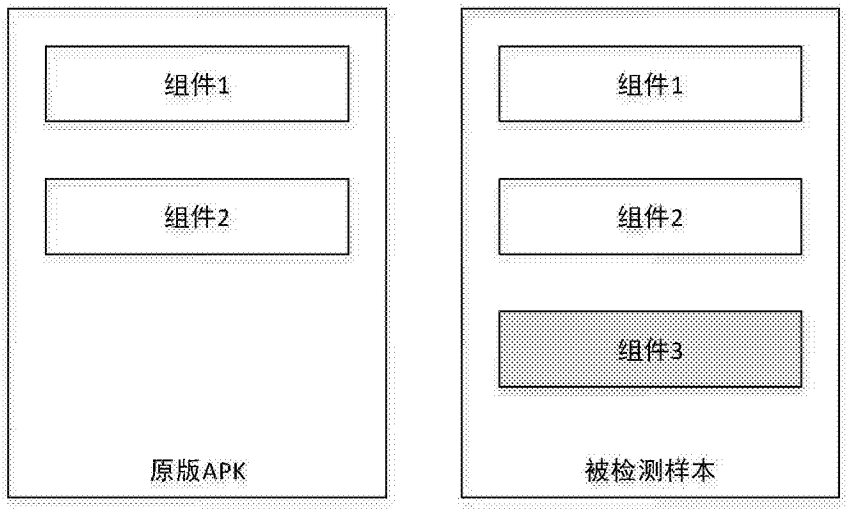

组件1
组件2
原版APK
组件1
组件2
组件3
被检测样本

图5

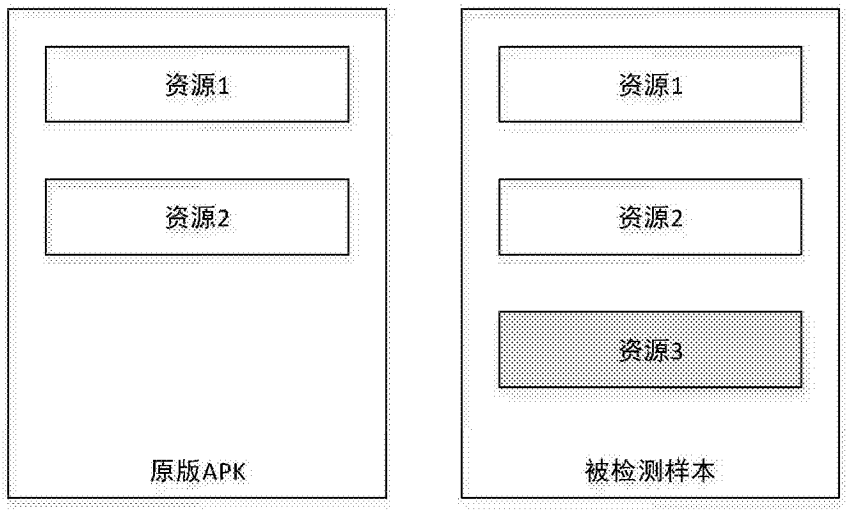

资源1
资源2
原版APK
资源1
资源2
资源3
被检测样本

图6

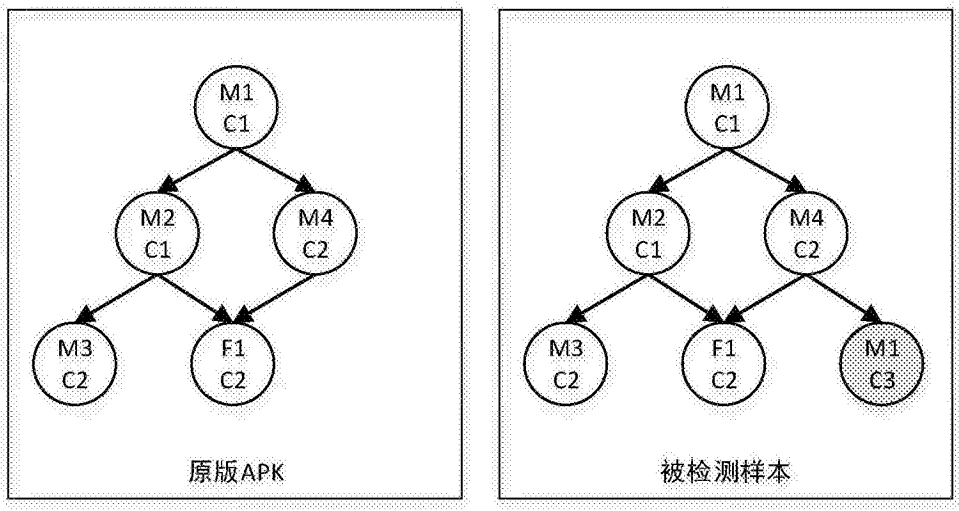

M1
C1
M2
C1
M4
C2
M3
C2
F1
C2
原版APK
M1
C1
M2
C1
M4
C2
M3
C2
F1
C2
M1
C3
被检测样本

图7