doi:10.3969/j.issn.1000-436x.2013.z1.014

基于权限频繁模式挖掘算法的 Android 恶意应用检测方法

杨欢[1]，张玉清[1,2]，胡予濮[1]，刘奇旭[2]

（1. 西安电子科技大学 综合业务网理论及关键技术国家重点实验室，陕西 西安 710071；
2. 中国科学院大学 国家计算机网络入侵防范中心 北京 100190）

摘 要：Android 应用所申请的各个权限可以有效反映出应用程序的行为模式，而一个恶意行为的产生需要多个权限的配合，所以通过挖掘权限之间的关联性可以有效检测未知的恶意应用。以往研究者大多关注单一权限的统计特性，很少研究权限之间关联性的统计特性。因此，为有效检测 Android 平台未知的恶意应用，提出了一种基于权限频繁模式挖掘算法的 Android 恶意应用检测方法，设计了能够挖掘权限之间关联性的权限频繁模式挖掘算法—PApriori。基于该算法对 49 个恶意应用家族进行权限频繁模式发现，得到极大频繁权限项集，从而构造出权限关系特征库来检测未知的恶意应用。最后，通过实验验证了该方法的有效性和正确性，实验结果表明所提出的方法与其他相关工作对比效果更优。

关键词：频繁模式；数据挖掘；恶意应用检测；权限特征；Android 系统

中图分类号：TP 393.08 **文献标识码**：A **文章编号**：1000-436X(2013)Z1-0106-10

Android malware detection method based on permission sequential pattern mining algorithm

YANG Huan[1], ZHANG Yu-qing[1,2], HU Yu-pu[1], LIU Qi-xu[2]

(1. State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China;
2. National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing 100190, China)

Abstract: The permissions requested by Android applications reflect the behavior sequence of the application. While a generation of malicious behavior usually requires the cooperation of multiple permissions, so mining the association between permissions can effectively detect unknown malicious applications. Most researchers concerned the statistical properties of a single permission, and there was little researchers studying the statistical properties of the association between permissions. In order to detect unknown Android malwares, an Android malware detection method based on permission sequential pattern mining algorithm was proposed. The proposed method design a permission sequential pattern mining algorithm PApriori to dig out permissions association. PApriori algorithm could discover permission sequential pattern from 49 malware families and build the permissions association dataset to detect malware. The experiment results prove that it performs better than other related work in efficiency and accuracy.

Key words: sequential pattern mining; data mining; malware detection; permission feature; Android OS

1 引言

根据市场调研机构 Gartner 发布的最新统计报告显示[1]，2012 年第三季度全球智能手机销量超过 4.28 亿部。其中，Google Android 和 Apple iOS 操作系统的智能手机市场份额分别为 72.4%和 13.9%，合计达到了 86.3%。由此可见，Android 在智能手机市场份额中占主导地位。Android 系统是完全开

收稿日期：2013-07-03

基金项目：国家自然科学基金资助项目（61272481）；中国博士后科学基金资助项目（2011M500416，2012T50152）；北京市自然科学基金资助项目(4122089)；国家发改委信息安全专项基金资助项目（发改办高技[2012]1424）

Foundation Items: The National Natural Science Foundation of China (61272481); China Postdoctoral Science Foundation (2011M500416, 2012T50152); The Natural Science Foundation of Beijing (4122089); The Foundation of National Development and Reform Commission (NDRC) Special Information Security(High-Tech Development and Reform Commission [2012]1424)

放的操作系统，而开放性越高风险就越大。因此，Android 也成为了众多恶意代码开发者的活跃地盘。Google 对于 Android 应用缺乏严密的安全审核机制，恶意应用程序开发者对国内外热门应用增加恶意代码后发布到各大论坛和应用商店，这些恶意应用被大量用户下载和安装，造成了一定的危害。

许多用户发现在下载和安装 Android 应用（app）的时候，第三方会要求读取用户的通讯录、短信、相册以及地理位置等隐私信息。根据中国互联网络信息中心调查的数据显示[2]，44.4%的人在下载安装应用软件的过程中会仔细查看授权说明，40.7%的人不会仔细查看，14.9%的人表示“不好说”，也就是说大部分人存在着盲目授权的风险。金山手机毒霸的反病毒工程师李铁军认为，盲目授权成为用户信息泄露的主要原因[3]。这使得对 Android 应用申请的权限信息进行检测成为亟待解决的问题。

与 PC 平台检测技术类似，手机上的恶意应用检测技术也分为 2 类，分别为基于特征码和基于行为的检测技术。基于特征码的检测方法应用比较广泛，大部分反病毒软件都采取此类方法，通过提取恶意应用样本的特征码来进行检测，误报率小，但是只能检测特征库中存在的恶意应用，不能检测防御恶意应用变种及未知恶意应用。基于行为的检测方法提取恶意应用各种行为信息，如 API 使用、系统调用、函数调用、内存情况等来检测未知恶意应用，但是算法实现比较复杂。

目前，由于数据挖掘技术可以从大量数据中挖掘出有意义的信息[4,5]，有研究者使用数据挖掘技术提取恶意应用的行为信息，检测未知恶意应用。数据挖掘技术在 PC 平台的恶意代码检测上的应用已经取得了良好的效果。并且已经有研究者将频繁模式挖掘算法应用到 PC 平台上的恶意代码检测。在智能手机平台上，研究者大多还是使用传统的分类、聚类等数据挖掘算法对恶意应用的行为进行检测。

基于以上提出的问题，本文使用基于权限行为的静态分析方法，并结合频繁模式挖掘算法进行 Android 恶意应用检测。首先，使用基于行为的静态特征提取方法对 Android 应用进行自动分析，得出该应用程序所申请的权限信息，构建权限特征库。然后，对每个恶意应用家族使用权限频繁模式挖掘算法挖掘出权限之间的相互依赖性，即频繁模式构建多个恶意应用家族的权限关系特征库。最后，对待检测应用程序提取权限信息匹配权限关系特征库，来判断该应用程序是否为恶意应用。本文设计了多个实验对 1 260 个恶意应用、Google Play 上的 2 000 个非恶意应用以及 2012 年~2013 年的 90 个最新恶意应用进行检测，证明了本文提出方法的有效性和正确性。

本文主要贡献如下。

首次将频繁模式挖掘算法应用到智能手机的恶意应用检测，设计并实现了权限频繁模式挖掘算法 PApriori。挖掘出 49 个恶意应用家族申请的敏感权限，为避免用户过度授权和为用户提供安全抉择做出了贡献。为进一步应用频繁模式挖掘算法进行其他行为特征恶意应用检测提供了启发性思考。

设计了多个实验评估本文提出方法的有效性和正确性，对比其他检测方法效果更优。

2　相关工作

恶意应用是对计算机安全的一个主要威胁，随着移动智能终端的广泛应用，智能手机上的恶意应用不断涌现，由于 Android 系统的开放性，使其成为恶意应用编写者的众矢之的。手机上的恶意应用检测技术包括基于特征码和基于行为的检测方法。基于特征码的检测方法有国外著名的检测工具 Androguard[6]，它的优点是误报率低，缺点是只能检测已知恶意应用程序库中已知类型的恶意应用，无法检测新的未知恶意应用。2012 年 11 月 13 日发布的最新版本 Android 4.2 中，Google 提出了一个防御恶意应用的服务“Application Verification Service”，使用 Google Cloud 对开启该服务手机上的应用程序进行恶意代码检测，以提醒用户该应用程序的危害。2013 年，Xuxian Jiang 等人[7]的测试结果显示，Application Verification Service 可能只是以应用程序的 SHA1 值和 package 名称作为特征码进行检测，导致检测率低下。

基于行为的恶意应用检测技术主要提取应用程序的各种行为特征来检测未知恶意应用。本文重点关注对于权限特征的相关研究。2012 年，Yajin Zhou 等人[8]提出了一个 Android 恶意代码检测方法 DroidRanger，其中有一部分是基于权限行为的过滤机制，手动查看总结出各个恶意应用家族使用的敏感权限，然后根据其总结出来的特征过滤出可疑恶意应用，并将此作为整个方法的第一步工作。Enck 等人[9]基于 Android 应用申请的权限开发了 Kirin，制定了 9

条权限相关安全规则对 310 个 Android 应用进行检测。Barrera 等人[10]测试了 1 100 个 Android 应用申请的权限，并使用 SOM（self organized learning）算法研究应用程序申请的相似权限。Felt 等人[11]研究了 Android API 和权限之间的关系，设计了 Stowaway 检测 940 个 Android 应用的权限滥用问题。Xuetao Wei 等人[12]深入研究了 Android 系统的权限申请和使用机制，提出了相关建议。

近年来，由于数据挖掘技术的广泛应用，结合数据挖掘算法进行恶意应用检测成为一个新的研究领域[13~15]。数据挖掘算法中的频繁模式挖掘算法最早是在文献[16]中提出的，随后被广泛应用到各个领域中。由于频繁模式挖掘算法可以挖掘频繁出现在数据集中的模式，文献[17]将频繁模式挖掘算法应用到 PC 平台的恶意代码检测。受其启发，本文构想将频繁模式挖掘算法应用到 Android 平台的恶意代码检测，自动挖掘出权限关联性，为用户提供参考，避免对恶意应用申请敏感权限的授权。

3 权限频繁模式挖掘算法 PApriori

频繁模式，简单地说就是指挖掘频繁出现在数据集中的模式[18,19]。频繁模式 Apriori 算法是 Agrawal R 等人[16]提出的为布尔关联规则挖掘频繁项集的原创性算法。本文基于频繁模式 Apriori 算法，设计了权限频繁模式挖掘算法 PApriori，挖掘同族应用申请的权限之间的关联，以此构建权限关系特征库，进行 Android 未知恶意应用检测。

算法设计思想为：Android 应用申请的各个权限在一定程度上反映了应用程序的行为模式，通过挖掘权限之间的关联性可以有效检测未知恶意应用。以往研究者对大量恶意应用申请的权限进行统计得出，恶意应用普遍都申请了的高危权限以及非恶意应用申请权限的统计特征。但是，通常一个恶意行为的产生需要多个权限的配合，例如，一个恶意应用要完成发送短信的恶意行为必须同时申请短信和网络相关权限。因此，单一权限的统计特性无法充分反映恶意应用行为信息，本算法的思想就是通过挖掘权限之间的关联性检测未知恶意应用。本算法分别对 49 个恶意应用家族的每族恶意应用分别挖掘其权限关联性，不是对所有恶意应用一起挖掘权限关联性，这样做提高了准确性。下面给出详细介绍。

3.1 概念定义

PApriori 算法建立在如下一些相关概念的基础上。

定义 1 权限项集。权限的集合 $I=\{I_1,I_2,\cdots,I_m\}$。包含 m 个权限的项集称为 m 权限项集。

定义 2 事务。设权限数据库 DB 是事务的集合，其中每个事务 T 是一个 Android 应用申请的权限集合。每一个事务有一个标识符，称作 TID。

定义 3 支持度 s。设 A,B 为权限项集，s 是权限数据库 DB 中事务包含 $A\cup B$ 的百分比，即概率 $P(A\cup B)$，$support(A\Rightarrow B)=P(A\cup B)$。

定义 4 支持度计数。权限项集的出现频率是包含权限项集的事务数，简称为权限项集的频率。

定义 5 最小支持度阈值。即 I 的支持度 s 满足对应的最小支持度计数阈值 min_sup，则 I 是频繁权限项集。频繁 k 权限项集的集合记作 L_k。

定义 6 极大频繁权限项集。权限项集 X 是 Z 中的极大频繁权限项集，如果 X 是频繁的，不存在超权限项集 Y 使得 $X\subset Y$，并且 Y 在 S 中是频繁的。

3.2 算法形式化描述

该算法可以形式化描述为，每一家族中所有恶意应用的权限模式为一个权限数据库 DB，其中的每一个应用程序的权限模式为一个事务 TID，每一个权限为一个项，权限的集合为权限项集，最小支持度阈值为 min_sup，从权限项集中发现满足支持度不小于 min_sup 的所有频繁权限项集，最终得到极大频繁权限项集。

通过构造一个实例来具体说明算法过程。假设有一族恶意应用中包含 4 个 Android 应用程序，以表 1 某族恶意应用权限数据库 DB 为例，给出了 4 个 Android 应用并列出了各自申请的权限信息，设最小支持度阈值 min_sup=50%。在每次迭代中，算法都产生一个大项集的候选集，然后计算每一个候选集的出现次数，在阈值的基础上选择频繁权限项集。

表 1 某族恶意应用权限数据库 DB

TID	权限项集
1.apk	ACCESS_NETWORK_STATE, READ_PHONE_STATE, ACCESS_FINE_LOCATION
2.apk	INTERNET, READ_PHONE_STATE, SEND_SMS
3.apk	ACCESS_NETWORK_STATE,INTERNET,READ_PHONE_STATE, SEND_SMS
4.apk	INTERNET, SEND_SMS

表 2 给出了所有这些步骤。第一次迭代有 5 个长度为 1 的候选权限项集，支持度不小于 50%的权限项集为频繁权限项集，选取了 4 个频繁权限项集。第二次迭代有 6 个长度为 2 的候选权限项集，支持度不小于 50%的权限项集为频繁权限项集，选取了 4 个频繁权限项集。第三次迭代只有 1 个长度为 3 的权限候选项集，支持度不小于 50%，选取了一个频繁权限项集。第四次迭代无法产生候选权限项集，则{INTERNET，READ_PHONE_STATE，SEND_SMS}为极大频繁权限项集。对每族恶意应用依次采用上述方法挖掘每族恶意应用的极大频繁权限项集，构建权限关系特征库。算法 1 给出了具体实现过程，其中，Apriori 算法参考文献[18]的描述。

3.3 算法分析

最小支持度计数阈值 *min_sup* 的含义是待挖掘的权限频繁模式在恶意应用的权限模式样本集中的最小出现概率。*min_sup* 的选取直接关系到检测方法的漏报率和误报率。该值过高会导致非恶意应用被判断为恶意应用，即误报率上升。该值过低会导致恶意应用被判断为非恶意应用，即检测率下降漏报率上升。通过反复实验对每族恶意应用挑选最优的 *min_sup* 取值，保持较高的检测率和较低的漏报率、误报率。

对每族恶意应用的权限频繁模式挖掘过程中会产生多个频繁模式，而本文的算法只选取了极大频繁模式来构建特征库。因为过短的权限模式容易被正常应用申请，无法反映恶意应用行为模式，造成较高的误报率。因此，本文使用每族恶意应用的极大频繁权限项集来匹配待检测应用程序，这样可以发现未知恶意应用，且不需要执行应用程序，避免了恶意应用造成的破坏。

本文提出的权限频繁模式挖掘算法 PAPriori 是基于 Apriori 算法实现的，Apriori 算法的执行效率较差，本文根据权限频繁模式的特点，对 49 个族的权限数据库进行了预处理。首先，对每族权限数据库中的所有应用都申请的权限进行提取，并删除相关数据。然后，对所有应用很少申请的权限数据进行删除。最后，将预处理后的数据库送入 Apriori 算法产生极大频繁权限项集，再将所有应用都申请的权限附加到极大频繁权限项集。这样进行预处理后大幅降低了数据的维度，提高了算法的执行效率。

算法 1　PApriori 构建权限关系特征库

输入：

D_i：　49 个家族恶意应用样本库，i=1, 2, k, …, 49

min_sup：最小支持度计数阈值

输出：　*Dataset* 最大频繁权限项集数据库

方法：

1) for (i=1; $i \leqslant 49$; i++) {

2) L_{append} = *delete_postfix_append* (D_{si}); //提取所有应用都申请的权限项集；

3) D_{mi} = *delete_postfix* (D_{si}); //删除所有应用都申请的权限项集数据；

4) D_i = *delete_prefix* (D_{mi}); //删除所有应用极少申请的权限项集数据；

5) L_k = *max_permissions_gen* (D_i, *min_sup*)；

表 2　　**针对某族恶意应用权限数据库 DB 的算法迭代过程**

迭代次数	候选权限项集	计数	S/%	选择
第一次	{ ACCESS_NETWORK_STATE }	2	50	取
	{ READ_PHONE_STATE }	3	75	取
	{ ACCESS_FINE_LOCATION }	1	25	舍
	{ INTERNET }	3	75	取
	{ SEND_SMS }	3	75	取
第二次	{ ACCESS_NETWORK_STATE，INTERNET }	1	25	舍
	{ ACCESS_NETWORK_STATE，READ_PHONE_STATE }	2	50	取
	{ ACCESS_NETWORK_STATE，SEND_SMS }	1	25	舍
	{ INTERNET，READ_PHONE_STATE }	2	50	取
	{ INTERNET，SEND_SMS }	3	75	取
	{ READ_PHONE_STATE，SEND_SMS }	2	50	取
第三次	{ INTERNET，READ_PHONE_STATE，SEND_SMS }	2	50	取

6) append L_{append} to L_k//将所有应用都申请的权限项集附加到极大频繁权限项集；

7) add L_k to *Dataset*；

8) }

9) return *Dataset*

procedure *max_permissions_gen* (*D*, *min_sup*)

10) $L_1 = find\_frequent\_1\text{-}itemsets(D)$;

11) for (k=2; $L_{k-1} \neq \varnothing$; k++) {

12) $C_k = apriori\_gen\ (L_{k-1})$;

13) for each permission sequence $t \in D$ {

14) $C_t = subset\ (C_t, t)$;

15) for each candidate $c \in C_t$

16) $c.count$ ++;

17) }

18) $L_k = \{c \in C_k \mid c.count \geqslant min\_sup\}$

19) }

20) return L_k;

procedure *apriori_gen* (L_{k-1})

21) for each $l_1 \in L_{k-1}$

22) for each $l_2 \in L_{k-1}$

23) if (l_1 [1] = l_2 [1]) ^ (l_1 [2] = l_2 [2]) ^ … ^ (l_1 [k−2] = l_2 [k−2]) ^ (l_1 [k−1] < l_2 [k−1]) then {

24) $c = l_1$ 连接 l_2; //连接步：产生候选

25) if *has_infrequent_subset* (c, L_{k-1}) then

26) delete c; //剪枝步：删除非频繁模式的候选；

27) else add c to C_k;

28) }

29) return C_k;

procedure *has_infrequent_subset* (c, L_{k-1}) //使用先验知识

30) for each (k−1)−subset s of c

31) if $s \notin L_{k-1}$ then

32) return TRUE

33) return FALSE

4 基于 PApriori 算法的 Android 恶意应用检测方法实现

基于 PApriori 算法的 Android 恶意应用检测方法框架如图 1 所示。首先，构建包含 49 个恶意样本族和待检测应用程序的样本库；然后，对每个 Android 应用文件使用静态分析方法提取各自申请的权限信息，构建权限特征集合，并进行格式化处理；其次，使用权限频繁模式挖掘算法依次对每族恶意应用挖掘出极大频繁权限项集，构建权限关系特征库；最后，对待检测应用程序匹配权限关系特征库检测未知恶意应用。

基于以上流程，下面给出基于 PApriori 算法的 Android 恶意应用检测方法的详细方案。

Step1 构建样本库。恶意应用样本使用文献[20]中提供的样本库，它也被研究 Android 恶意代码检测的其他研究者广泛使用。该样本库有 1 260 个恶意应用包含 49 个恶意应用家族。本文改进了 Akdeniz 的爬虫程序[21]用来从 Google Play 上

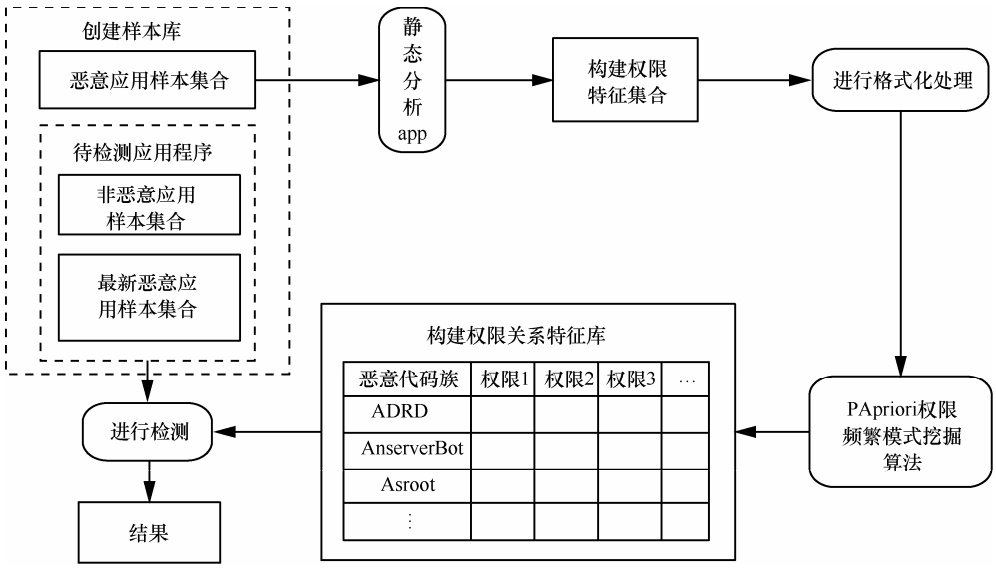


图 1 基于 PApriori 算法的 Android 恶意应用检测方法框架

批量下载大量应用程序以构建非恶意应用样本集合。为了保证样本库的准确性，使用各种杀毒软件、恶意应用检测工具和人工分析对大量 Google Play 上的应用程序进行过滤，确保 2 000 个非恶意应用的准确性。

Step2　采用基于权限的静态分析方法对每个 Android 应用进行自动化分析。首先，使用 Android 软件开发工具包（SDK, software development kit）中自带的工具 Android 资产打包工具（AAPT, Android asset packaging tool）对每个 Android 应用进行解压缩，提取出 AndroidManifest.xml 文件。每个应用都包含一个 AndroidManifest.xml 文件，位于根目录下，它定义了应用程序的内容和行为[22]。然后，对 AndroidManifest.xml 文件进行自动化分析，提取权限请求情况。如图 2 所示，该应用程序申请了“GET_ACCOUNTS”、“INTERNET”、“ACCESS_FINE_LOCATION”、“VIBRATE”和“WAKE_LOCK”权限。

```
N: android=http://schemas.android.com/apk/res/android
  E: manifest(line=2)
    A:  android:installLocation(0x010102b7)=(type 0x)0x0
    A:  package="com.example.android.service"(Raw:"com.example.android.service")
    E:  uses-permission(line=3)
      A: android:name(0x01010003)="android.permission.GET_ACCOUNTS"(Raw:
"android.permission.GET_ACCOUNTS"
    E: uses-permission(line=4)
      A: android:name(0x01010003)="android.permission.INTERNET"(Raw:
"android.permission.INTERNET")
    E: uses-permission(line=5)
      A: android:name(0x01010003)="android.permission.ACCESS FINE LOCATION"(Raw:
"android.permission.ACCESS_FINE LOCATION")
    E: uses-permission(line=6)
      A: android:name(0x01010003)="android.permission.VIBRATE"(Raw:
"android.permission.VIBRATE")
    E: uses-permission(line=7)
      A: android:name(0x01010003)="android.permission.WAKE LOCK"(Raw:
"android.permission.WAKE LOCK")
    E: uses-permission(line=8)
```

图 2　AndroidManifest.xml 文件部分内容

Step3　完成 Step2 的分析工作后将静态分析模块提取的每个应用程序申请的所有权限构建成权限特征集合。

Step4　使用 python 语言编写代码对 Step3 提取的权限特征进行处理，统一为 CSV（comma separated values）格式，即逗号分隔值格式，以便后面进行频繁模式挖掘。每行描述为一个 Android 应用申请的权限模式。

Step5　设计了权限频繁模式挖掘算法 PApriori，使用该算法挖掘出权限之间的相互依赖性，即极大频繁权限项集。

Step6　构建权限关系特征库用于存放恶意应用的极大频繁权限项集集合，匹配待检测应用程序。每行为一个族恶意应用的极大频繁权限项集。

Step7　使用 python 语言编写检测程序，对待检测应用程序提取权限信息匹配权限关系特征库来判断该应用是否为恶意应用。

Step8　输出日志信息，包含恶意应用名称和统计数量等信息。

5　实验与分析

本文设计了多个实验来验证基于权限频繁模式挖掘算法进行 Android 应用恶意代码检测的有效性和正确性，对比其他 Android 恶意应用检测方法，实验结果证明了本文提出的检测方法效果更优。

5.1　实验环境

实验在内存为 4 GB，处理器为 Intel(R) Core(TM)2 Quad 2.67 GHz 的机器上完成。本文将前面提出的检测方法使用 Java 和 Python 语言进行了实现。其中，静态分析、格式化处理主要由 Python 语言完成，算法实现和爬虫程序主要由 Java 语言完成。

5.2　实验样本

实验样本由 3 个部分组成。1) 1 260 个典型恶意应用测试样本，使用文献[20]中提供的样本库。该样本库有 1 260 个恶意应用，包含 49 个恶意应用家族。2) 2012 年和 2013 年发布的最新的 90 个恶意应用。3) 非恶意应用测试样本，从 Google Play 上批量下载大量应用程序，构建了 2 000 个非恶意应用的测试样本。

5.3　实验结果

对 1 260 个包含 49 族恶意应用的测试样本进行反复实验，权衡漏报率和误报率，适当选取 *min_sup* 值，得出了每族恶意应用的极大频繁权限项集。其中，Asroot 和 FakePlayer 恶意应用家族只申请了一个权限，无法表示权限关系，因此忽略此两族恶意应用；并且 DroidCoupon 、DroidDeluxe、DroidDream、DroidKungFuUpdate、Plankton 和 Tapsnake 6 个恶意应用家族申请的权限与正常应用程序申请的权限极为相似，因此也同样忽略了它们。最后，得出了 41 个恶意应用家族的极大频繁权限模式，限于篇幅，在此不再罗列，表 3 给出了部分结果。下面将使用同样的测试样本对本文提出的方法和其他 4 种 Android 恶意应用检测方法进行实验结果对比。

使用本文提出的方法对 1 260 个恶意应用检测的检测率为 87%，后面章节将给出本文实现的方法

对 1 260 个恶意应用的具体检测结果。对 2 000 个非恶意应用检测的误报率为 9.9%。对最新的 90 个恶意应用的检测率为 50%。误报率主要集中在被误判为 DroidKungFu2、DroidKungFu3 族的恶意应用，因为 DroidKungFu 族恶意应用的恶意行为代码隐藏在 native code 中，通过 UpdateCheck 服务加载 native code 病毒文件并执行，不需要申请过多的权限。DroidKungFu 族恶意应用的权限频繁模式无法正确反映该族恶意应用行为，导致了本文方法存在一定的误报率。

5.4 实验分析

使用相同的测试样本集合对其他检测方法进行评估，证明本文提出方法的有效性和正确性。下面进行详细介绍。

DroidRanger[8]。2012 年，Yajin Zhou 等人提出了一个 Android 恶意代码检测方法 DroidRanger[8]，其中有一部分是基于权限行为的过滤机制，手动查看并总结出 9 个恶意应用家族使用的敏感权限，根据总结出来的权限特征过滤出可疑恶意应用，将该过滤机制作为整个工作的第一步。而本文提出的方法对 49 族恶意应用的权限模式实现了自动的极大频繁权限模式项集挖掘，构建权限关系特征库，与待检测应用程序进行匹配。本文方法与 DroidRanger 提出的 9 个恶意应用家族申请的敏感权限对比如表 3 所示。由此可以看出，DroidRanger 提出的敏感权限比较有限，如果直接用于检测恶意应用会导致极高的误报率，因此，该文献也只是将其用于过滤出可疑恶意应用。而本文给出的方法可以直接用于检测恶意应用。

Kirin[9]。Enck 等人提出了一个基于权限的 Android 恶意应用检测方法 Kirin[9]，给出了 9 条权限安全规则如表 4 所示。使用 Kirin 对 1 260 个恶意应用进行检测只能检测出 36 个，检测率仅为 2.87%，具体结果如表 5 所示。对 2 000 个非恶意应用检测的误报率为 7.7%。对最新的 90 个恶意应用进行检测，只能检测出 1 个恶意应用。

表 4 Kirin 9 条安全规则

序号	权限规则
1	SET_DEBUG_APP
2	READ_PHONE_STATE, RECORD_AUDIO, INTERNET
3	PROCESS_OUTGOING_CALL, RECORD_AUDIO, INTERNET
4	ACCESS_FINE_LOCATION, INTERNET, RECEIVE_BOOT_COMPLETE
5	ACCESS_COARSE_LOCATION, INTERNET, RECEIVE_BOOT_COMPLETE
6	RECEIVE_SMS, WRITE_SMS
7	SEND_SMS, WRITE_SMS
8	INSTALL_SHORTCUT, UNINSTALL_SHORTCUT
9	SET_PREFERRED_APPLICATION

Androguard[6]。由于在网上可以很容易地获得

表 3 DroidRanger 和本文方法提取的权限对比

恶意代码族	DroidRanger 得出的敏感权限	本文方法得出的极大频繁权限项集
ADRD	INTERNET, ACCESS_NETWORK_STATE, RECEIVE_BOOT_COMPLETED	INTERNET, ACCESS_NETWORK_STATE, MODIFY_PHONE_STATE, READ_PHONE_STATE, RECEIVE_BOOT_COMPLETED, WRITE_APN_SETTINGS, WRITE_EXTERNAL_STORAGE
Bgserv	INTERNET, RECEIVE_SMS, SEND_SMS	INTERNET, RECEIVE_SMS, ACCESS_WIFI_STATE, WAKE_LOCK, ACCESS_NETWORK_STATE, SEND_SMS, RECEIVE_BOOT_COMPLETED, READ_PHONE_STATE, BROADCAST_SMS, CHANGE_NETWORK_STATE, ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, WRITE_EXTERNAL_STORAGE,
DroidDream	CHANGE_WIFI_STATE	INTERNET, CHANGE_WIFI_STATE, ACCESS_WIFI_STATE, READ_PHONE_STATE
DroidDreamLight	INTERNET, READ_PHONE_STATE	INTERNET, READ_CONTACTS, ACCESS_NETWORK_STATE, RECEIVE_BOOT_COMPLETED, READ_PHONE_STATE , GET_ACCOUNTS, READ_SMS
Geinimi	INTERNET, SEND_SMS	INTERNET, ACCESS_FINE_LOCATION, CALL_PHONE, MOUNT_UNMOUNT_FILESYSTEMS, READ_CONTACTS, READ_PHONE_STATE, SEND_SMS, SET_WALLPAPER, WRITE_CONTACTS, WRITE_EXTERNAL_STORAGE
jSMSHider	INSTALL_PACKAGES	INTERNET, ACCESS_NETWORK_STATE, DELETE_PACKAGES, READ_PHONE_STATE , ACCESS_COARSE_LOCATION, INSTALL_PACKAGES,
Pjapps	INTERNET, RECEIVE_SMS	INTERNET, RECEIVE_SMS, SEND_SMS, READ_PHONE_STATE, WRITE_ EXTERNAL_ STORAGE
Zsone	RECEIVE_SMS, SEND_SMS	INTERNET, RECEIVE_SMS, SEND_SMS, RESTART_PACKAGES, ACCESS_COARSE_LOCATION
zHash	CHANGE_WIFI_STATE	INTERNET, READ_CONTACTS, ADD_SYSTEM_SERVICE, CHANGE_WIFI_STATE, RECEIVE_SMS, MODIFY_PHONE_STATE, ACCESS_WIFI_STATE, ACCESS_NETWORK_STATE, MODIFY_AUDIO_SETTINGS, PROCESS_ OUTGOING_CALLS, SEND_SMS, WRITE_SMS, CALL_PHONE, READ_PHONE_STATE, CHANGE_NETWORK_STATE, READ_SMS

Androguard[6]检测工具，本文使用同样的测试样本库对 Androguard 进行了测试。最新版本 Androguard 对于 Android 应用的解析存在问题，并且最新版本的 Androguard 特征库并没有发生变化，只是加入了更多的程序分析功能，对本实验没有影响。因此本文还是对官方网站提供的 Ubuntu 系统镜像 ARE 默认安装的 Androguard 版本进行实验。实验结果显示，将 Androguard 用于检测 1 260 个恶意应用，能检测出 863 个恶意应用，检测率为 68.49%，具体结果如表 5 所示。对 2 000 个非恶意应用进行检测误报率为 0。对最新的 90 个恶意应用进行检测，检测率为 0。并且在检测过程中不能实现完全自动化，部分应用不能正确处理，进程会意外停止。

Google。Google 提出了一个防御恶意应用的服务“Application Verification Service”。Xuxin Jiang 等人对该服务进行了详细的检测[7]，同样使用本文使用的恶意应用库测试，对 1 260 个恶意应用进行检测，只检测出 193 个恶意应用，检测率只有 15.32%，如表 5 所示。通过测试结果显示，Application Verification Service 可能只是以应用程序的 SHA1 值和分组名作为特征码进行检测，导致检测率低下。

综上所述，DroidRanger 关于权限的过滤机制不能自动挖掘权限规则并且不能直接用于检测恶意应用。将本文的方法与 Kirin、Androguard 和“Application Verification Service”相比较，使用 1 260 个恶意应用样本和最新的 90 个恶意应用进行实验，

表 5　　本文方法与 Androguard、Google 和 Kirin 检测方法的实验结果对比

恶意代码族	样本个数	Androguard	Google	Kirin	本文方法	恶意代码族	样本个数	Androguard	Google	Kirin	本文方法
ADRD	22	19	8	1	19	AnserverBot	187	0	2	2	187
Asroot*	8	0	5	0	0	BaseBridge	122	100	7	3	82
BeanBot	8	0	0	0	7	Bgserv	9	0	0	0	8
CoinPirate	1	0	0	0	1	CruseWin	2	0	0	0	2
DogWars	1	0	1	0	1	DroidCoupon*	1	0	0	0	0
DroidDeluxe*	1	1	0	0	0	DroidDream*	16	16	6	0	0
DroidDreamLight	46	0	18	0	31	DroidKungFu1	34	33	8	1	22
DroidKungFu2	30	30	9	0	13	DroidKungFu3	309	285	21	14	228
DroidKungFu4	96	96	18	0	89	DroidKungFuSapp	3	0	0	3	3
DroidKungFuUpdate*	1	0	0	0	0	Endofday	1	0	0	0	1
FakeNetflix	1	0	0	0	1	FakePlayer*	6	0	5	0	0
GamblerSMS	1	0	0	1	1	Geinimi	69	67	11	3	53
GGTracker	1	0	1	0	1	GingerMaster	4	4	2	0	4
GoldDream	47	32	6	0	46	Gone60	9	0	0	0	9
GPSSMSSpy	6	0	1	0	6	HippoSMS	4	3	1	0	4
Jifake	1	0	0	0	1	jSMSHider	16	16	4	0	9
KMin	52	52	39	0	52	LoveTrap	1	1	0	0	1
NickyBot	1	0	0	1	1	NickySpy	2	2	0	2	2
Pjapps	58	40	8	3	55	Plankton*	11	11	2	1	0
RogueLemon	2	0	0	0	2	RogueSPPush	9	9	0	0	3
SMSReplicator	1	0	0	0	1	SndApps	10	10	0	0	10
Spitmo	1	0	0	0	1	Tapsnake*	2	1	1	0	0
Walkinwat	1	1	1	0	1	YZHC	22	22	3	1	22
zHash	11	0	1	0	11	Zitmo	1	0	0	0	1
Zsone	12	12	4	0	11						
总共	1 260	863	193	36	1 003						

注：标注*的是本文方法忽略的恶意应用家族，检测率为 0。

结果显示，本文的方法检测率最高。对 2 000 个非恶意应用的实验结果显示，对比其他检测方法，本文实现的方法在误报率上略高于 Kirin 和 Androguard 检测方法，而检测率高于这 2 种检测方法，而且 Androguard 只能检测已知恶意应用，所以误报率为 0，而本文方法重点研究对于未知恶意应用的检测。在检测未知恶意应用的研究中，为了保证较高的检测率就会牺牲一定的误报率，本文通过大量实验均衡了检测率和误报率的大小，得出了最优的检测方法。通过以上多个实验结果验证了本文提出的方法与其他 Android 恶意应用检测方法相比效果更优。

6 讨论

以上着重介绍了本文工作的创新点以及与其他相关工作对比的优点，但是该工作仍然存在不足，下面介绍本文提出方法的局限性和下一步工作的方向。

本文提出的算法 PApriori 是基于 Apriori 算法实现的，受到算法本身的限制，在运行时间，效率上具有一定的限制。本文提出了针对权限数据库的一个预处理方案，大大减少了执行时间。但是由于本文涉及的权限数据库较小，可以解决算法运行效率问题，后面当使用反馈机制加入更多新的恶意应用家族，数据库不断增大时，可以采用更合理的频繁模式挖掘算法完善系统，提高运行效率。

本文提出的方法分别对每族恶意应用分别挖掘其权限关联性，不是对所有恶意应用一起挖掘权限关联性，这样做提高了准确性。但是本文只实现了对权限特征的检测，正因为特征类型的单一性以及只使用了静态方法，造成了一定的误报率。后面将应用频繁模式挖掘算法研究恶意应用其他行为模式，如未经用户同意发送短信的行为模式、读取用户敏感数据的行为模式等，使用动静态结合的方法分析 Android 应用的多类行为特征，提高未知恶意应用的检测率，降低误报率。并且通过其他行为特征可以检测那些绕过权限申请实现的恶意应用。

本文设计的多个实验验证了 3 350 个应用程序的检测结果，后面将对多个第三方应用商店和 Google Play 上的大量应用程序进行检测，发现新类型家族的恶意应用，并采用反馈机制扩充权限关系特征库。

7 结束语

本文首次将频繁模式挖掘算法应用到智能手机的恶意应用检测领域。结合频繁模式挖掘算法和基于权限行为的静态分析方法，提出了一种基于权限频繁模式挖掘算法的 Android 系统恶意应用的检测方法，自动挖掘出权限关联性，为用户提供参考。首先，使用静态的分析方法提取 Android 应用的权限特征，设计了 PApriori 算法进行权限频繁模式发现，得到极大频繁权限项集，从而挖掘出相关规则，进行未知恶意应用检测。然后，实现了本文提出的检测方法，对 3 350 个应用程序进行检测。最后，通过多个实验与其他 Android 恶意应用检测方法进行对比。实验结果证明了本文提出方法的有效性和正确性。下一步将研究如何提高频繁模式算法的效率，进一步挖掘其他特征关联模式完善该检测方法。

参考文献：

[1] Gartner says worldwide sales of mobile phones declined 3 percent in third quarter of 2012; smartphone sales increased 47 percent[EB/OL]. http://www.gartner.com/newsroom/id/2237315, 2013.

[2] 中国互联网络信息中心[EB/OL]. http://www.cnnic.cn/, 2013.
China internet network information center[EB/OL]. http://www.cnnic.cn/, 2013.

[3] 下载 APP 安装信息被盗主因：盲目授权致信息泄露[EB/OL]. http://tech.gmw.cn/2013-04/01/content_7174916_2.htm, 2013.
Information stolen during download and installation APP, main reason: information disclosure due to blind authorization[EB/OL]. http://tech.gmw.cn/2013-04/01/content_7174916_2.htm, 2013.

[4] WITTEN I H. Data Mining: Practical Machine Learning Tools and Techniques[M]. Beijing: China Machine Press, 2012.

[5] 李海峰, 章宁, 朱建明等. 时间敏感数据流上的频繁项集挖掘算法[J]. 计算机学报, 2012, 35(11):2283-2293.
LI H F, ZHANG N, ZHU J M, *et al*. Frequent itemset mining over time-sensitive streams[J]. Chinese Journal of Computers, 2012, 35(11): 2283-2293.

[6] Androguard[EB/OL]. http://code.google.com/p/androguard/, 2013.

[7] JIANG X X. An evaluation of the application (″app″) verification service in Android 4.2[EB/OL]. http://www.cs.ncsu.edu/faculty/jiang/appverify/, 2013.

[8] ZHOU Y J, WANG Z, ZHOU W, *et al*. 2012 Hey, you, get off of my market: detecting malicious apps in official and alternative Android markets[A]. Proceedings of the 19th Annual Network & Distributed System Security Symposium[C]. 2012. 1-13.

[9] ENCK W, ONGTANG M, MCDANIEL P. On lightweight mobile phone application certification[A]. Proceedings of the 16th ACM Conference on Computer and Communications Security CCS '09[C]. Chicago, IL, USA, 2009. 235-245.

[10] BARRER D, KAYACIK H G, VAN OORSCHOT P C, *et al*. A methodology for empirical analysis of permission-based security models and its application to Android[A]. Proceedings of the 17th ACM Conference on Computer and Communications Security CCS '10[C]. Chicago, IL, USA, 2010. 73-84.

[11] FELT A P, CHIN E, HANNA S, *et al*. Android permissions demystified[A]. Proceedings of the 18th ACM Conference on Computer and Communications Security CCS '11[C]. Chicago, IL, USA, 2011. 627-638.

[12] WEI X T, GOMEZ L, NEAMTIU I, *et al*. Permission evolution in the Android ecosystem[A]. Proceedings of the 28th Annual Computer Security Applications Conference ACSAC '12[C]. Orlando, Florida, USA, 2012. 31-40.

[13] WU D J, MAO C H, WEI T E, *et al*. DroidMat: Android malware detection through manifest and API calls tracing[A]. Proceedings of the Seventh Asia Joint Conference on Information Security Asia JCIS 2012[C]. Tokyo, Japan, 2012. 62-69.

[14] SHABTAI A, KANONOY U, ELOVICI Y, *et al*. Andromaly: a behavioral malware detection framework for Android devices[J]. Journal of Intelligent Information Systems, 2012, 38(1):161-190.

[15] IKER B, URKO Z, SIMIN N T. Crowdroid: behavior-based malware detection system for Android[A]. Proceedings of the ACM CCS workshop on Security and Privacy in Smartphones and Mobile Devices SPSM'11[C]. Chicago, Illinois, USA, 2011. 15-26.

[16] AGRAWAL R, IMIELINSKI T, SWAMI A N. Ming association rules between sets of items in large databases[A]. Proceedings of the 1993 ACM SIGMOD the International Conference on Management of Data[C]. Washington DC, USA, 1993. 207-216.

[17] WANG L N, TAN X B, PAN J F. *et al*. Application of prefixspan* algorithm in malware detection expert system[A]. Proceedings of the First International Workshop on Education Technology and Computer Science[C]. 2009. 448-452.

[18] HAN J W, KAMBER M. Data Mining Concepts and Techniques[M]. Elsevier Inc San Francisco, 2007.

[19] KANTARDZIC M. Data mining: concepts, models, methods, and algorithms[A]. IEEE Computer Society, A John Wiley & Sons[C]. Hoboken, NJ, 2003. 143-151.

[20] ZHOU Y J, JIANG X X. Dissecting Android malware: characterization and evolution[A]. Proceedings of the 33rd IEEE Symposium on Security and Privacy[C]. Oakland, USA, 2012. 95-109.

[21] Google-play-crawler[EB/OL]. https://github.com/Akdeniz/google-play-crawler, 2012.

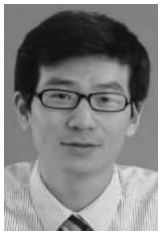
[22] The Android manifest.xml file[EB/OL]. http://developer.android.com/guide/topics/manifest/manifest-intro.html, 2013.

作者简介：

杨欢（1984-），女，天津人，西安电子科技大学博士生，主要研究方向为漏洞挖掘、恶意代码检测和数据挖掘。

张玉清（1966-），男，陕西宝鸡人，中国科学院大学博士生导师，主要研究方向为网络与信息系统安全。

胡予濮（1955-），男，河南濮阳人，西安电子科技大学博士生导师，主要研究方向为序列密码与分组密码、网络安全协议的设计与分析。



刘奇旭（1984-），男，江苏徐州人，中国科学院大学讲师、博士后，主要研究方向为漏洞挖掘、漏洞评估、漏洞管理、应急响应等。