文章编号:1671-4598(2016)01-0156-03 DOI:10.16526/j.cnki.11-4762/tp.2016.01.043 中图分类号:TP309.5 文献标识码:A

基于 Android 系统的手机恶意软件检测模型

马晋杨，徐 蕾

(沈阳航空航天大学 计算机学院，沈阳 110136)

摘要：为提高手机应用软件的安全性，提出一种基于 Android 系统的手机恶意软件检测模型；模型利用数据挖掘的方法对恶意软件中的敏感 API 调用进行数据挖掘，进而得到恶意软件检测规则；针对检测规则在检测非恶意软件时，产生较高误报率的问题，设计了加权 FP-growth 关联规则挖掘算法，算法在数据挖掘的两个步骤中，对敏感 API 调用加权，利用支持度阈值去除一些出现次数频繁而权重小的规则，降低了非恶意软件的误报率；实验结果表明，模型对恶意软件检测率达到 81.7%，非恶意软件的检错率降低到 11.3%。

关键词：Android 系统；恶意软件；数据挖掘；敏感 API；FP-growth 算法

Mobile Malware Detection Model Based on Android System

Ma Jinyang, Xu Lei

(School of Computer, Shenyang Aerospace University, Shenyang 110136, China)

Abstract: In order to improve the security of mobile application software based on Android system , a mobile malware detection model is proposed. The model manipulates sensitive API call via data mining to obtain detection rules. To reduce the false positive rate when the rule is used to detect the non malware. a weighted FP-growth association rule mining algorithm is proposed. Based on weighting the sensitive API call, we employ a support threshold to eliminate the rules which preserve small weight and occur with high frequency. The experiments show that the model achevied a detection rate of 81.7% for malwares, and reduced the false positive rate to 11.3%.

Keywords: Android system; malicious software; data mining ; sensitive API ; FP-growth algorithm

0 引言

随着手机应用软件的日益增多，人们对手机应用安全性的要求愈来愈高。鉴于多种型号的手机和平板 PC 使用 Android 操作系统，针对 Android 平台的恶意软件检测得到研究者的广泛关注。

基于 Android 系统的恶意软件检测方法主要有：1）利用软件获得的访问权限做检测。文献 [1] 运用统计学知识给出了权限风险等级计算方法，该方法可以计算出一组权限的风险值，并且在应用程序安装时将风险值呈现给用户，让用户决定是否安装。文献 [2] 通过对权限配置文件的挖掘分析，发现软件使用敏感权限之间的关联性，并将其作为检测规则进行恶意软件检测。这种方法能够发现一些敏感恶意行为，但对软件的权限申请要求严格，进而限制了具有特殊权限要求软件的使用。

2）利用软件中对敏感 API 的调用进行检测。恶意软件通常利用 API 调用获取或者操作用户关键数据和系统资源，包括短信、通讯录、通话记录等，然后向外部传送上述数据。文献 [3-4] 利用分类算法分别对 Symbian 和 Android 可执行文件中的函数调用进行分类，进而检测软件中是否包含恶意代码，但由于缺乏恶意软件样本，检测效果不易评价。文献 [5] 利用 K-means 算法区分基于同一应用软件改写的恶意代码和非恶意代码，该方法只能对同一应用软件不同变种进行检测。文献 [6] 设计了记录 Android 系统中敏感 API 调用的日志模块，实现在软件运行时记录敏感 API 调用及输入/输出的详细信息，文献没有涉及对记录的 API 进行分析的方法及结果。

本文通过对敏感 API 的研究发现，对敏感 API 的调用进行关联可以有效的检测出恶意软件，而且不会出现上述的一些问题，实验结果表明，本文提出的方法对恶意软件的检测率达到了 81.7%左右，具有很好的检测效果。

1 恶意软件检测模型

本文提出的方法属于静态检测方法，即对 Android 系统中运行软件的源文件进行分析，得到软件中敏感 API 的调用，利用这些 API 调用判断此软件是否为恶意软件。恶意软件检测模型首先根据 API 调用在软件恶意行为中重要程度的不同，运用 AHP 层次分析法对每类敏感 API 做加权处理，再利用改进后的 FP-growth 算法对恶意软件数据集做数据挖掘，进而获得基于敏感 API 调用的恶意软件检测关联规则，用于手机恶意软件检测，模型结构如图 1 所示。

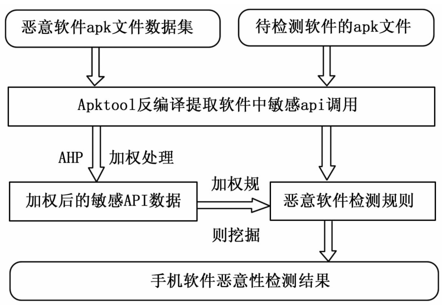


图 1 恶意手机软件检测模型

1.1 敏感 API 的定义与获取

出于安全考虑，Android 应用软件在利用系统框架提供的

收稿日期:2015-06-29； **修回日期**:2015-08-25。

作者简介：马晋杨(1991-),男,江苏南通人,硕士研究生,主要从事网络与信息安全方向的研究。

徐 蕾(1964-),女,上海人,教授,硕士研究生导师,主要从事网络与信息安全方向的研究。

关键性服务或系统调用时，需要申请权限，Android 系统中定义了 100 多种权限常量，应用程序只有获得了这些权限，才能在代码中调用其对应的 API，对于这类申请了权限才能进行调用的 API，本文称之为敏感 API。

Android 恶意软件主要表现为隐私窃取、恶意扣费、资费消耗、远程控制和系统破坏。恶意软件需要在程序中调用一些敏感 API 的组合实现这些恶意行为，本文通过对恶意软件数据集的分析，去除恶意软件中很少使用的敏感 API，最终确定 71 个常见的敏感 API 调用用于恶意软件检测，并根据其实现的功能分成六类，即交互类、系统类、网络类、隐私类、吸费类、数据读写类。表 1 列举几种常见的敏感 API 调用，为书写方便省略部分类名及参数。

表 1　敏感 API 分类及部分调用函数

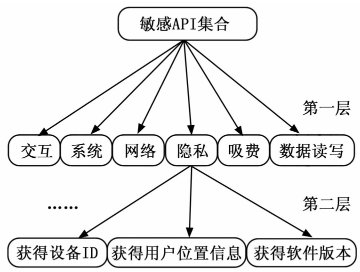
序号	敏感 API 分类	敏感 API 函数
1	交互类	插入短信：insert(Uri. parse("content://sms/"))； 插入联系人：insert(Uri. parse ("content://com. android. contacts/data"))
2	系统类	重打包：restartPackage()； 设置壁纸：setWallpaper()
3	网络类	获取网络状态：getNetworkInfo()； 更改 WiFi 状态：setWifiEnabled()
4	隐私类	获取设备 ID：getDeviceId()； 获取应用程序信息：getApplicationInfo()
5	吸费类	发送短信：sendTextMessage()； 获取账户：getAccounts()
6	数据读写类	对 SD 卡的读写操作需要获取 SD 卡路径 Environment. getExternalStorageDirectory(). getPath()

论文数据挖掘使用的是 Yajin Zhou 等人[7]收集的恶意软件数据集，该数据集中收集了 49 族 1260 款恶意软件。但由于 Android 系统的更新，一些敏感 API 的调用在不同的系统版本中其实现方法、参数等已经变更，本文根据 Android 系统的变化建立多版本的 API 链接关系，以适应不同时期软件 API 调用的检测。

1.2　基于 AHP 层次分析法求敏感 API 权值

实验发现，直接对恶意软件数据集做数据挖掘，将挖掘得到的规则集用于检测恶意软件，产生较高的误报率。分析后发现，不同的 API 的威胁程度是不同的，例如网络访问、读手机状态、写 SD 卡等敏感操作在正常软件中也频繁出现，如果将这些敏感 API 组合在一起作为检测规则会出现较大的误差。为避免上述情况发生，针对敏感 API 威胁程度的不同，采用 AHP 层次分析法对 71 种敏感 API 做加权，再对加权后的数据集进行挖掘，这样可以在规则集中过滤掉一些出现次数较多但权重很小的规则，从而提高预测的准确率。

AHP 层次分析法首先要对敏感 API 建立多级层次结构，本文构造两级层次结构，第一层为敏感 API 的分类，第二层为每类中的敏感 API，构成的层次树如图 2（a）。为获得权重值，在层次树双亲相同的同层节点中构造两两判断矩阵，矩阵中按照这一层各个节点之间重要程度进行两两比较，进而将节点划分为若干等级，论文采用了常用的 5 级定量法，对第一层构造的两两比较矩阵如图 2（b）所示。



(a)敏感API层次树

$$
\begin{array}{c|cccccc}
 & 交互 & 系统 & 网络 & 隐私 & 吸费 & 数据读写 \\
\hline
交互 & 1 & 7 & 5 & 4 & 5 & 9 \\
系统 & 1/7 & 1 & 1/4 & 1/3 & 1/2 & 7 \\
网络 & 1/5 & 4 & 1 & 1/2 & 1 & 3 \\
隐私 & 1/4 & 3 & 2 & 1 & 1/2 & 4 \\
吸费 & 1/5 & 4 & 1 & 1/2 & 1 & 3 \\
数据读写 & 1/9 & 1/7 & 1/3 & 1/4 & 1/3 & 1
\end{array}
$$

(b)第一层两两比较矩阵

图 2　敏感 API 分层结构图及第一层比较矩阵

计算第一层矩阵最大特征根 $\lambda_{max}=6.6037$ 以及其对应的归一化特征向量为：

$\boldsymbol{W}^{T}=$（0.485 5，0.079 8，0.128 0，0.145 3，0.128 0，0.033 4）

通过一致性检验，$\boldsymbol{W}$ 为归一化的权向量，其中的每一项 wi（$i=1，2，\cdots.\ 6$）即为每一类的权值。运用同样的方法计算每一类中敏感 API 相对于上层节点权重系数 $w_{i,api}$，最终每一个敏感 API 的权值为 $w_{api}=w_{i,api} * w_{i}$，其中 w_i 为此 API 所在类的权重值。

获得敏感 API 权重之后，将数据集中软件的 API 调用做加权处理，再利用 FP－growth 算法做数据挖掘。

1.3　加权 FP－growth 关联规则挖掘

FP－growth[8]算法因无需多次扫描事务数据库，在运行效率上优于 Apriori 算法，本文利用 FP－growth 算法进行加权数据挖掘。提取恶意软件训练数据集中每个软件调用的敏感 API 构成一项事务加入到事务数据库中，设置 API 频度最小筛选阈值 F 和最小支持度筛选阈值 S，加权 FP－growth 关联规则数据挖掘算法如下：

1）扫描事务数据库，计算事务数据库中个 API 的出现频度；计算 API 的加权频度＝API 频度 $* w_{api}$；删除 API 的加权频度小于 F 的 API；将 API 按照加权频度递减排序（这一序列称为 L）。

2）再次扫描事务数据库，对于每项事务表示的软件调用 API，按照 L 中的顺序重新排序，并将其插入到 FP－Tree 中。

3）从 FP－Tree 中找出频繁项集。

4）对第 3）步中找出的每项频繁项，计算频繁项的加权支持度。设某个频繁项 T 中出现的 API 调用集合为 SAPI；

T 的加权支持度＝max（w_{api} ｜ api∈SAPI）$*$ T 的支持度；其中 T 的支持度是指利用数据挖掘得到的频繁项支持度，若频繁项的加权支持度小于 S，则从频繁项集中删除该频繁项。

上述算法得到的频繁项集作为恶意软件的检测规则集。利用加权的方法克服了文献［2］中只选取极大频繁项的做法，对一些不是极大频繁项，但同样重要的规则也提取出来，加入到最终的检测集中，进而提高了检测率，降低了误报率。

2 实验及结果分析

为了验证系统的有效性，本文首先把收集的 1 260 款恶意软件分为两部分，其中 70%作为数据挖掘的样本数据，对这部分数据做挖掘得到恶意软件检测规则集；另外的 30%数据作为检测数据集，若恶意软件家族中只有一个恶意软件则不放入数据挖掘样本集，直接作为检测数据。实验中对每种恶意软件家族选择了不同的最小频度阈值和最小支持度阈值，以 ADRD 软件家族为例，选取最小频度阈值为 0.8，最小支持度阈值为 0.074。用数据挖掘样本做数据挖掘共得到 75 条规则，应用这些规则对检测数据集进行检测，检测结果如表 2 所示。

表 2 Android 系统恶意软件检测实验结果

恶意软件家族	样本软件数	用于检测软件数	识别软件数	恶意软件家族	样本软件数	用于检测软件数	识别软件数
ADRD	22	7	5	GingerMaster	4	2	0
AnserverBot	192	58	21	GoldDream	47	15	15
Asroot	8	3	0	Gone60	9	3	0
BaseBridge	122	37	34	GPSSMSSpy	6	2	0
BeanBot	8	3	3	HippoSMS	4	2	2
Bgserv	9	3	3	Jifake	1	1	1
CoinPirate	1	1	1	jSMSHider	16	5	5
CruseWin	2	1	1	KMin	52	16	16
DogWars	1	1	0	LoveTrap	1	1	1
DroidCoupon	1	1	0	NickyBot	1	1	1
DroidDeluxe	1	1	0	NickySpy	2	1	1
DroidDream	16	5	5	Pjapps	58	18	18
DroidDreamLight	46	14	11	Plankton	11	4	4
DroidKungFu1	34	11	10	RogueLemon	2	1	1
DroidKungFu2	30	9	7	RogueSPPush	9	3	3
DroidKungFu3	309	93	92	SMSReplicator	1	1	1
DroidKungFu4	96	29	29	SndApps	10	3	0
DroidKungFuSapp	3	1	0	Spitmo	1	1	1
DroidKungFuUpdate	1	1	0	Tapsnake	2	1	0
Endofday	1	1	1	Walkinwat	1	1	0
FakeNetflix	1	1	0	YZHC	22	7	6
FakePlayer	6	2	0	zHash	11	4	4
GamblerSMS	1	1	1	Zitmo	1	1	1
Geinimi	69	21	21	Zsone	12	4	4
GGTracker	1	1	1	总数	1260	404	330

利用数据挖掘得到的规则对 404 个恶意软件做检测，检测出 330 个恶意软件，检测率为 81.7%；分析实验结果发现，由于在小样本数据（某种恶意软件家族只有几个恶意软件样本）中，用于数据挖掘的样本较少，导致部分小样本恶意软件家族的检测率较低；再有就是部分恶意软件调用的敏感 API 权重过小，行为与正常软件行为接近，导致此类软件的检测率低。

本文的加权数据挖掘方法主要去除了一些出现次数频繁而权重小的规则，进而降低了非恶意软件的检错率。为验证提出方法在非恶意软件中的检错率，从百度官方 Android 市场下载各类 Android 应用软件 115 个，其中包括常用的软件集合，如社交软件、游戏软件、Android 读书软件、办公软件，最终检测结果是 13 个软件报出异常，误报率为 11.3%；分析发现，误报软件大多是社交类软件，这类软件调用的敏感 API 多，组合也极其敏感。若在数据挖掘过程中不做加权，利用得到的规则检测 115 个非恶意软件，报出 43 个软件异常，误判率达到了 37.4%。

3 结束语

为提高基于 Android 系统的手机安全性，本文提出一种检测 Android 系统手机恶意软件的方法；论文定义了敏感 API 调用及其分类，利用恶意软件样本数据集对其软件中的敏感 API 调用进行数据挖掘，进而得到恶意软件检测规则；针对挖掘规则检测非恶意软件时，产生较高误报率的问题，提出了加权 FP－growth 关联规则挖掘算法。实验结果表明，本方法的恶意软件检测率达到 81.7%，非恶意软件的检错率降低到 11.3%，获得了很好的检测效果。

方法获得的敏感 API 加权系数是根据主观判断不同类 API 的重要程度，给出的判定矩阵计算得到的，使得敏感 API 的权重系数是人为确定的，进一步的工作可利用数据集中的数据采用机器学习方法得出权重值。

参考文献：

[1] Sangho Lee，Da Young Ju. Assessment of malicious applications using permissions and enhanced user interfaces on Android [A]. (ISI'13)：Intelligence and Security Informatics，2013 IEEE International Conference on. [C]. IEEE，2013：270－S270.

[2] 杨 欢，张玉清，胡予濮，等. 基于权限频繁模式挖掘算法的 Android 恶意应用检测方法 [J]. 通信学报，2013，34 (Z1)：106－S115.

[3] Schmidt Aubrey－Derrick，Clausen Jan Hendrik，Cam－Tepe Ahmet，et al. Detection Symbian os malware through static function call analysis [A]. (Malware 2009)：Proceedings of the 4th IEEE International Conference on Malicious and Unwanted Software [C]. Montreal，Canada，2009：15－S22.

[4] Schmidt Aubrey－Derrick，Bye Rainer，Schmidt Hans－Gunther，et al. Static analysis of executables for collaborative malware detection on Android [A]. (ICC'09)：Proceedings of the 2009 IEEE International Conference on Communications [C]. Dresden，Germany，2009：631－S635.

[5] Iker Burguea，Urko Zurutuza，Simin Nadjmtehrani. Crowdroid：Behavior－based malware detection system for Android [A]. (SPSM－'11)：Proceedings of the ACM CSS workshop on Security and Privacy in Smartphones and Mobile Devices [C]. Chicago. USA. 2011：15－S26.

[6] 吕晓庆，邹仕洪. 基于 smali 的 Android 软件敏感 API 调用日志模块嵌入系统 [J]. 科技论文在线. 2012

[7] 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－S109.

[8] Han J W，P J，Y Y W. Mining frequent patterns without candidate generation：a frequent pattern tree approach [J]. Data Mining and Knowledge Discovery，2004，8 (1)：53－S87.