一种基于混合特征的移动终端恶意软件检测方法

姚 烨[1], 钱 亮[1], 朱怡安[1], 张黎翔[1], 贾 耀[1], 杜家伟[1], 牛军涛[1]

[1]西北工业大学 计算机学院 西安 中国 710064

摘要 随着移动终端恶意软件的种类和数量不断增大, 本文针对 Android 系统恶意软件单特征检测不全面、误报率高等技术问题, 提出一种基于动静混合特征的移动终端恶意软件检测方法, 以提高检测的覆盖率、准确率和效率。该方法首先采用基于改进的 CHI 方法和凝聚层次聚类算法优化的 K-Means 方法构建高危权限和敏感 API 库, 然后分别从静态分析和动态分析两个方面提取移动终端系统混合特征。在静态分析中, 首先反编译 APK 文件, 分析得到权限申请特征和敏感 API 调用特征; 在动态分析中, 通过实时监控 APP 运行期间的动态行为特征, 分别提取其在运行过程中的敏感 API 调用频次特征和系统状态等特征信息; 接着分别使用离差标准化、TF-IDF 权重分析法和优序图法对混合特征进行归一化和特征权重赋值处理。最后, 通过构建测评指标对本文所提基于混合特征恶意软件检测方法进行对比测试验证和评价分析。实验结果表明:本方法针对 Android 系统恶意软件的检测具有好的准确率和效率, 可有效提高移动终端恶意软件检测的精确度。

关键词 移动终端; 恶意软件检测; 混合特征检测; 机器学习; Android 系统

中图法分类号 TP391.9 DOI 号 10.19363/J.cnki.cn10-1380/tn.2022.03.08

A Malware Detection Method Based on Hybrid Feature for Mobile Terminals

Yao Ye[1] , Qian Liang[1], Zhu Yian[1], Zhang Lixiang[1], Jia Yao[1], Du Jiawei[1], Niu Juntao[1]

[1] School of computer science, Northwestern polytechnical University, Xi'an 710064, China

Abstract At present, with the large-scale use of the Android system, the types of malware based on the Android system are emerging in endlessly, and the types of viruses are increasing. Aiming at the problems of incomplete detection of single feature of the Android system malware, low accuracy rate, and high false alarm rate, this article proposed a mobile terminal malware detection analysis method based on mixed dynamic and static features to improve the coverage, accuracy and efficiency of malware detection for Android systems. By combining the feature values extracted by the two detection methods, such as the static analysis and dynamic analysis method, the efficiency and accuracy of malware detection are further improved. First, the paper built high-risk permissions and sensitive API libraries based on the improved CHI method and the K-Means method optimized by the agglomerated hierarchical clustering method, and then extracted the mixed characteristics of the mobile terminal system from static analysis and dynamic analysis. In the static analysis, the APK file was decompiled firstly, and the permission application characteristics and sensitive API call characteristics were analyzed. In the dynamic analysis, the dynamic behavior characteristics during the running of the APP were monitored in real time, and the frequency of sensitive API calls during the running process was extracted. Characteristics and system status characteristics. Then the paper used dispersion standardization, TF-IDF weight analysis method and optimal sequence graph method to normalize the mixed features and assign feature weights. Finally, the data sets downloaded from VirusShare and Drebin was de-duplicated and other related processing will be carried out. Then, the malware detection methods based on the mixed features proposed in this article was compared and evaluated, Experiments results showed that this method in this paper had good accuracy and efficiency for the detection of Android system malware, and effectively improves the detection accuracy of malware.

Key words mobile terminal; malware detection; hybrid feature detection; machine learning; Android system;

通讯作者: 朱怡安, 教授(博士生导师) , Email: zhuya@nwpu.edu.cn。

国家重点研发计划项目(No.2020YFB1712201); 国家工业互联网创新发展工程项目(No.TC190A3X8-16-1, No.TC200H038); 陕西省重点研发(重点产业链)项目(No.2019ZDLGY12-07); 太仓市大院大所创新项目(No.TC2019DYDS06); 东莞市科技装备动员项目(No.KZ2018-14)以及陕西省重点研发计划项目(No.2021ZDLGY05-05)等资助。

收稿日期: 2021-01-25; 修改日期: 2021-07-06; 定稿日期: 2022-01-11

1 引言

Android 系统作为移动智能终端市场的畅销系统之一, 近几年随着智能手机、平板电脑和智能可穿戴设备的普及, 其市场占有率越来越高。根据“中国互联网信息中心”2020 年第 46 次《中国互联网络发展状况统计报告》[1]分析, 2020 年上半年受疫情冲击, 我国就业压力显著加大, 互联网平台经济下的新业态、新模式增加了大量就业岗位, 正在成为支撑就业的重要力量。上半年, 我国个人互联网应用呈现平稳增长态势, 截至到 6 月份, 我国网民规模达 9.40 亿, 互联网普及率达 67.0%, 我国网民使用手机上网的比例达 99.2%。

同期, 针对 Android 系统的恶意软件也层出不穷。《2019 年度 Android 恶意软件专题报告》[2]分析指出: 2019 年全年, 360 安全大脑共截获移动端新增恶意程序样本约 180.9 万个, 平均每天截获新增手机恶意程序样本约 0.5 万个。新增恶意程序类型主要为资费消耗, 占比 46.8%; 其次为隐私窃取(41.9%)、远程控制(5.0%)、流氓行为(4.6%)、恶意扣费(1.5%)、欺诈软件(0.1%)。2019 年全年, 360 安全大脑累计为全国手机用户拦截恶意程序攻击约 9.5 亿次, 平均每天拦截手机恶意程序攻击约 259.2 万次。2019 全年从漏洞数量看, Android 系统以 414 个漏洞位居产品漏洞数量榜首。因此, 针对 Android 系统恶意软件的安全性检测一直是业界的一大难题。

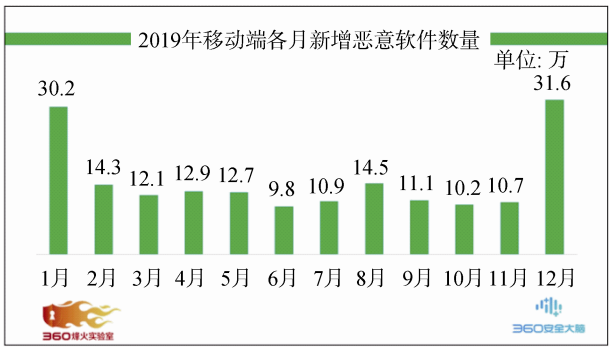

图 1 2019 年移动端各月新增恶意软件数量

Figure 1 Number of new malware on mobile terminals in each month in 2019

目前, 对于 Android 恶意软件的检测方式分为静态检测和动态检测两种[3]。静态检测主要分析 APK 文件, 通过反编译手段从配置文件及代码文件中分析特征; 动态检测主要通过动态监测软件运行过程中的数据和调用, 并描述为特征信息加以匹配处理实施检测。

如崔艳鹏等人[4]提出的一种基于抽象 API 调用序列的 Android 恶意软件检测方法。该方法采用 API 包名、混淆名和自定义名来抽象 API 调用序列, 使得抽象出来的序列不依赖 API 版本。在此基础上,使用随机森林分类算法进行恶意软件检测。这种方法只从静态分析的角度进行检测, 忽略了动态特征的作用性。同样, 宋鑫等人[5]基于随机森林算法提出的检测方法仅以 Android 权限为特征定义有效权限, 其检测结果的准确率只达到 92.84%。此外, 张玉玲等人[6]提出的挖掘数据隐含特征的检测方法和严喆等人[7]提出的基于改进的关联规则挖掘算法的检测方法都未对软件运行时的动态行为进行分析和检测。而吴帆等人[8]虽然分别基于动态分析和静态分析获得混合特征, 但是针对动态特征的分析和提取并不全面, 没有充分考虑到用户对于当前软件的操作状态等信息的采集和获取。

Feng 等人[9]提出一种基于语义特征码的检测方法, 并在此基础上做了改进, 通过分析软件中的控制流信息和数据流信息, 用以构建组件间调用图 ICCG(Inter-Component Call Graph), 再通过挖掘该图的最大公共子图, 进而通过匹配子图的方式来判断软件的恶意性, 但是该方法需要人工筛选用于构建特征码的样本, 费时费力。Alam 等人[10]提出了一种针对 native 恶意代码的检测方法; 该方法先将 native 代码解释成一种中间语言, 再在控制流图的基础上对每条状态语句用中间语言进行描述并构建带注释的控制流图, 最后构建特征码用于恶意软件检测; 但是该方法只针对 native 代码进行检测, 如果恶意代码并不存在于 native 代码中, 那么检测将毫无意义。Luca 等人[11]提出了一种面向进程和动态挖掘的检测方法。该方法利用过程挖掘技术来识别从应用软件收集的系统调用跟踪中的关系和重复执行模式, 以表征其行为。Manel 等人[12]提出了一种基于遗传算法的 AMD 方法, 该方法通过进化一组 API 调用序列, 按照这种进化规则来发现新的恶意行为。Milosevic 等人[13]提出了一种基于 LSTM 和编码-解码神经网络体系结构的深度学习技术, 通过检测 CPU、内存和电池使用情况来实现对恶意软件的检测分析。上述三组方法分别针对进程、API 调用序列和系统状态等进行分析, 但方式都很单一, 并没有对选择多特征进行检测分析, 导致检测准确率只能依靠各自的算法进行提升。

综上所述, 国内外研究者们针对移动终端恶意软件检测的研究方法互有优点。但是他们提取或分析的特征都仅限于某一个方面, 或仅将 Android 权限

作为有效特征, 或仅针对 native 代码进行分析, 并且在上述研究者们的内容中, 并没有充分说明特征的选择依据。

本文主要贡献体现在以下 4 点: (1)提出一种基于混合特征的移动终端恶意软件检测方法; (2)构建了一个高危权限和敏感 API 库; (3)通过分析和提取静态高危权限、敏感 API 调用、动态敏感 API 调用频次以及系统状态等混合特征, 作为恶意软件检测的数据依据; (4)通过对比实验对 Android 恶意软件进行分类检测和评价分析, 测试结果表明: 本文提出的方法在兼顾移动终端动态和静态行为特征的前提下, 提高了 Android 恶意软件的检测准确率。

2 构建高危权限和敏感 API 库

2.1 筛选高危权限

本文使用卡方(CHI, Chi-Square Test)值 χ^2 来衡量权限和恶意软件的相关程度, 其值越大则表示相关程度越高, 由此筛选出 Android 系统的高危权限集。

传统 CHI 统计方法存在特征项出现频率与类别负相关问题, 也没有考虑某一个特征项存在于某一个文本中的概率问题, 若一个特征项在其他类别的多数文本中集中存在, 而在该类别的少数文本中很少存在, 则会使得CHI的计算值偏高, 反之则会偏低。

针对上述问题, 马莹等人[14]提出一种改进的 CHI 统计方法。

首先, 针对特征项出现频率与类别负相关问题, 改进如公式(1)所示。

$$\chi^2(w,c)=\begin{cases}\dfrac{N(AD-BC)^2}{(A+B)(A+C)(B+D)(C+D)} & (AD-BC)>0\\ 0 & (AD-BC)<0\end{cases} \tag{1}$$

其中, $(AD-BC)>0$表示特征项出现频率与类别正相关, $(AD-BC)<0$表示特征项出现频率与类别负相关。

其次, 针对没有考虑某一个特征项存在于某一个文本中的概率问题, 将频度 α、集中度 β和分散度 γ引入 CHI 中, 改进如公式(2)所示。

$$\chi^2(w,c)=\begin{cases}\dfrac{N(AD-BC)^2}{(A+B)(A+C)(B+D)(C+D)}\times\alpha\times\beta\times\gamma & (AD-BC)>0\\ 0 & (AD-BC)<0\end{cases} \tag{2}$$

假设训练集类别分别为C_1(良性)和C_2(恶意), 设类别 C_i $(i=1,2)$ 对应的样本集为 $D_i=\{d_{i1},d_{i2},\ldots,d_{im}\}$, $f_{d_{ik}}$表示特征 w 在样本d_{ik} $(1\leqslant k\leqslant m)$中存在的频率, m 为样本集D_i中的样本数, t_{D_i}为样本集D_i中含有特征 w 的样本数。则 α、β 和 γ 分别表示如下。

频度 α: 指样本集D_i中出现特征 w 的次数占其特征总数的比重。

$$\alpha=\sqrt{\sum_{k=1}^{m}f_{d_{ik}}} \tag{3}$$

集中度 β: 指样本集D_i中含有特征 w 的样本数占所有样本集中含有特征 w 样数的比重。

$$\beta=\frac{(2\cdot t_{D_i}-\sum_{i=1}^{2}t_{D_i})^2}{2\cdot\sum_{i=1}^{2}t_{D_i}}=\frac{(t_{D_1}-t_{D_2})^2}{2(t_{D_1}+t_{D2})} \tag{4}$$

分散度 γ: 指样本集D_i中含有特征 w 的样本数占D_i样本总数的比重。

$$\gamma=\frac{t_{D_i}}{m} \tag{5}$$

本文首先从网站 VirusShare、Drebin、Android Malware Genome Project 和 Sharing Android Malware Dataset 上面收集公开数据集, 经过去重筛选后得到包含 8000 个恶意软件的恶意样本集; 其次从 Google 市场、国内 Android 市场(如: 华为应用市场、小米应用市场等)收集良性软件样本共 8000 个; 最后将这些样本合并作为筛选高危权限集的基础。由于上述数据都是公开的官方数据, 因此可以保证真实可靠。

设 APK 样本总数为 N, 其中恶意样本集为N_c, 良性样本集为 $N_{\overline{c}}$。Android 系统的权限集为 $P=\{p_1,p_2,\ldots,p_M\}$。对于权限 $p_i\in P$ $(i=1,2,\ldots,M)$, 其权限样本分组如表 1 所示。

表 1 权限p_i样本分组

Table 1 Permission p_i sample grouping

组别	含该权限	不含该权限	总计
恶意样本	A	B	A+B
正常样本	C	D	C+D
总计	A+C	B+D	A+B+C+D

其中, A 表示含有权限p_i的恶意样本数, B 表示不含权限p_i的恶意样本数, C 表示含有权限p_i的良性样本数, D 表示不含权限p_i的良性样本数。

基于公式(2)计算权限p_i的 CHI 值, 计算公式如公式(6)所示。

$$\chi^2(p_i,c)=\begin{cases}\dfrac{N(AD-BC)^2}{(A+B)(A+C)(B+D)(C+D)}\cdot\alpha\beta\gamma & (AD-BC)>0\\ 0 & (AD-BC)<0\end{cases} \tag{6}$$

其中, α 表示恶意样本集N_c中含有权限p_i的次数占该样本集中出现权限总数的比重; β 表示恶意样本集N_c中含有权限p_i的样本数占所有样本集中含有权限p_i样数的比重; γ 表示恶意样本集N_c中含有权限p_i的样本数占N_c样本总数的比重。

由于CHI值越高, 表示权限p_i与恶意性的相关程度越高。因此, 本文按CHI值从大到小排序, 选取前 30 个权限形成预筛选的高危权限集, 如表 2 所示。

表 2 预筛选高危权限集

Table 2 Prefilter high risk permission sets

权限	权限	权限
ACCESS_COARSE_LOCATION	SEND_SMS	PROCESS_OUTGOING_CALLS
ACCESS_FINE_LOCATION	RECEIVE_SMS	READ_CONTACTS
ACCESS_NETWORK_STATE	READ_SMS	WRITE_CALENDAR
INTERNET	RECEIVE_MMS	WRITE_CALL_LOG
READ_PHONE_STATE	WAKE_LOCK	WRITE_CONTACTS
RECEIVE_BOOT_COMPLETED	ACCESS_WIFI_STATE	CALL_PHONE
VIBRATE	CHANGE_WIFI_STATE	GET_ACCOUNTS
BLUETOOTH_ADMIN	WRITE_SETTINGS	READ_CALENDAR
RECORD_AUDIO	READ_EXTERNAL_STORAGE	READ_CALL_LOG
CAMERA	WRITE_EXTERNAL_STORAGE	BLUETOOTH

2.2 去除权限之间的相关性

Android系统的部分权限之间存在较强的相关性, 往往具有高相关性的一组权限会同时出现在同一恶意软件中。若同时选取具有较强相关性的权限对作为特征选取对象, 则会产生特征冗余, 使得特征向量数据长度过大, 从而对分类结果产生影响。因此, 本文需要对高危权限集中的权限进行聚类分组。

设在 2.1 节中预筛选的高危权限集为$P_{Fir} = \{p_1, p_2, \dots, p_{30}\}$, Android 恶意软件样本集为$S_{Vir} = \{s_1, s_2, \dots, s_N\}$, 对于高危权限集中的任一权限$p_i \in P_{Fir}\ (i = 1,2,\cdots,30)$, 对应的特征向量为$F_i = \{f_1, f_2, \cdots, f_N\}$, 其中, $f_j = \{0, 1\}\ (j = 1,2,\cdots,N)$, $f_j = 0$表示权限p_i在恶意软件样本s_j中没有出现。则需要进行聚类的数据集为:

$$D_{Per} = \{F_1, F_2, \cdots, F_{30}\}$$

本文中对需要进行聚类的数据集定义如下: 设在 2.1 节中预筛选的高危权限集为$P_{Fir} = \{p_1, p_2, \cdots, p_{30}\}$, Android 恶意软件样本集为$S_{Vir} = \{s_1, s_2, \cdots, s_N\}$, 对于高危权限集中的任一权限$p_i \in P_{Fir}\ (i = 1,2,\cdots,30)$, 对应的特征向量为$F_i = \{f_1, f_2, \cdots, f_N\}$, 其中, $f_j = \{0, 1\}\ (j = 1,2,\cdots,N)$, $f_j = 0$表示权限p_i在恶意软件样本s_j中没有出现, $f_j = 1$表示权限p_i在恶意软件样本s_j中有出现。因此, 这里需要进行聚类的数据集为$D_{Per} = \{F_1, F_2, \cdots, F_{30}\}$。

可以看出, 数据集D_{Per}的数据量不大, 但是数据的维数受样本集S_{Vir}的样本个数N控制, 本文中的N控制在 1500 左右, 因此, 数据维数比较高, 不适合使用传统 DBSCAN 算法。由于数据量不大, 使用凝聚层次聚类算法消耗的计算量和存储量在可接收范围之内, 但是也存在数据高维和噪声处理能力不足的问题。而以 K-Means 算法为代表的聚类算法除了受限于初始聚类中心和 K 值选取外, 非常契合本文数据集的聚类。

因此, 本文采用基于组平均凝聚层次聚类算法优化的 K-Means 算法去除高危权限之间的相关性。在组平均凝聚层次聚类算法中, 权限之间的相关性基于“皮尔逊相关系数”来衡量, 任意两个向量$F_i, F_j \in D_{Per}$的皮尔逊相关系数计算如公式(7)所示, 皮尔逊相关系数参考表 3。

$$Pearson\ (F_i, F_j) = \frac{cov(F_i, F_j)}{\sigma_{F_i}\sigma_{F_j}} \tag{7}$$

表 3 皮尔逊相关系数参考表

Table 3 Pearson correlation coefficient reference table

系数值范围	相关程度
0.8~1.0	极强相关
0.6~0.8	强相关
0.4~0.6	中等程度相关
0.2~0.4	弱相关
0.0~0.2	极弱相关或不相关

则使用组平均凝聚层次聚类算法对高危权限进行初始聚类的描述如算法 1 所示。

算法 1: 组平均凝聚层次聚类算法

输入: 高危权限数据集$D_{Per} = \{F_1, F_2, \ldots, F_{30}\}$, 皮尔逊相关系数阈值 0.95

输出: 高危权限初始聚类中心集 D_{AHC}

1 $P \leftarrow \varphi$, $D_{AHC} \leftarrow \varphi$, $D_{Cluster} \leftarrow D_{Per}$
2 DO $\boldsymbol{P.clear}()$AND $\boldsymbol{D_{AHC}.clear}()$
3 FOR EACH $D_m, D_n \subset D_{Cluster}\ (m \neq n)$
4 FOR EACH $F_i \in D_m$ AND $F_j \in D_n$
5 $\rho_{F_i,F_j} \leftarrow calPearson(F_i, F_j)$
6 $P.add(\rho_{F_i,F_j})$
7 IF $avg\_proximity(P) \geqslant 0.95$ THEN
8 $D_{AHC}.add(\{D_m, D_n\})$
9 $\boldsymbol{D_{Cluster} \leftarrow D_{AHC}}$
10 WHILE $D_{Cluster} \neq D_{AHC}$
11 RETURN D_{AHC}
12 END

算法 1 的具体解释如下:

(1) 第 1 行: 设定皮尔逊相关系数为 0.95, 初始聚类中心集D_{AHC}为空, 以及初始化聚类集$D_{Cluster}$;

(2) 第 2 ~ 10 行: 第一次聚类时, 将聚类集$D_{Cluster}$中的每一个特征向量$\boldsymbol{F_i} \in D_{Per}\ (i = 1, 2, \ldots, 30)$看作一簇, 计算两两之间的皮尔逊相关系数, 选取皮尔逊相关系数大于或等于 0.95 的两个簇合并成新簇; 之后的每次聚类, 是计算聚类集$D_{Cluster}$中任意两簇元素之间皮尔逊相关系数的平均值, 选择平均值大于或等于 0.95 的两个簇合并成新簇; 每次合并后更新聚类集$D_{Cluster}$直到该聚类集不发生变化;

(3) 第 11、12 行: 返回经过聚类后的聚类集D_{AHC}并结束。

使用 K-Means 算法对高危权限聚类集D_{AHC}进行二次聚类, 则使用组平均凝聚层次聚类算法优化后的 K-Means 算法描述如算法 2 所示。

算法 2: 使用组平均凝聚层次聚类算法优化后的 K-Means 算法

输入: 高危权限数据集 $D_{Per} = \{F_1, F_2, \ldots, F_{30}\}$, 初始聚类集 D_{AHC}

输出: 高危权限最终聚类集 $D_{K-Means}$

1 $D_{K-Means} \leftarrow \varphi$, $D_{Center} \leftarrow \varphi$, $\rho_{MAX} \leftarrow 0$
2 $D_{Center} \leftarrow getInitClusterCenter(D_{AHC})$
3 DO FOR EACH $F_i \in D_{Per}$
4 FOR EACH $F_j \in D_{Center}$ AND $F_i \neq F_j$
5 $\rho_{MAX}(i, j) \leftarrow calPearson\ (F_i, F_j)$
6 $D_{K-Means}.add(F_i, F_j)$
7 FOR EACH $D_k \subset D_{K-Means}$
8 FOR EACH $F_i \in D_k$
9 FOR EACH $F_j \in D_k$ AND $F_i \neq F_j$
10 $P.add(\rho_{avg}(i) \leftarrow calPearson\ (F_i, F_j))$
11 $D_{Center} \leftarrow getNewClusterCenter(D_{K-Means}, P)$
12 UNTIL FOR EACH $D_{Cluster} \subset D_{K-Means}$
13 $\rho_{avg} \leftarrow cal\_avg\_proximity(D_{Cluster})$ AND $\rho_{avg} \geqslant 0.8$
14 RETURN $D_{K-Means}$
15 END

算法 2 描述的具体解释如下:

(1) 第 1 行: 定义并初始化最终聚类集$D_{K-Means}$、每次迭代的聚类中心集D_{Center}和最大皮尔逊相关系数变量ρ_{MAX};

(2) 第 2 行: 从初始聚类集D_{AHC}的每个类簇中随机各选一个特征向量样本$\boldsymbol{F_i}$作为初始聚类中心;

(3) 第 3 ~ 13 行: 首先计算所有样本和每个聚类中心的皮尔逊相关系数, 按相关系数值最高的原则就近合并类簇; 然后, 对每一类簇中的样本, 计算类内该样本与其他样本之间的皮尔逊相关系数平均值, 选择平均值最大的样本作为该类簇在下次迭代过程中的新聚类中心; 最后, 对每一类簇的样本, 计算该类内所有样本之间的皮尔逊相关系数平均值ρ_{avg}。若对于每一类簇而言都满足$\rho_{avg} \geqslant 0.8$, 则停止迭代; 否则返回第 3 行继续迭代;

(4) 第 14、15 行: 返回最终聚类集$D_{K-Means}$并结束。

最后, 基于组平均凝聚层次聚类算法优化的 K-Means 算法的聚类结果, 将表 2 中的 30 个权限分为 14 组, 具体如表 4 所示。

2.3 基于高危权限的敏感 API 筛选

通常情况下, 恶意软件功能的实现都需要调用特定 API 来完成, 如: 调用录音的 API 和访问通讯录 API, 此类 API 被称为敏感 API。Android 系统的权限和 API 存在多对多的映射关系, 往往一个权限组下面对应多个 API。同时, 每一组高危权限对应的 API 函数集中包含多个重载的函数, 这些重载函数本身实现的功能基本一致, 所以只需要监控其中一个即可。

本文针对每一组高危权限对应的 API 函数集合进行筛选, 对于重载的函数或者功能几乎一致的多个 API 函数, 只保留其中一个。最后共筛选出 40 个敏感 API, 部分敏感权限与权限分组的映射关系如表 5 所示, 全部 API 的具体描述见附录 1: 权限列表。

表 4 去除权限相关性后的权限分组

Table 4 Permission grouping after removing permission correlation

权限组	权限	权限组	权限
INTERNET	INTERNET	SMS	RECEIVE_MMS
	ACCESS_NETWORK_STATE		RECEIVE_SMS
	ACCESS_WIFI_STATE		READ_SMS
	CHANGE_WIFI_STATE		SEND_SMS
CONTACTS	READ_CONTACTS	PHONE	READ_CALL_LOG
	WRITE_CONTACTS		WRITE_CALL_LOG
	GET_ACCOUNTS		READ_PHONE_STATE
STORAGE	READ_EXTERNAL_STORAGE		CALL_PHONE
	WRITE_EXTERNAL_STORAGE		PROCESS_OUTGOING_CALLS
CALENDAR	READ_CALENDAR	LOCATION	ACCESS_FINE_LOCATION
	WRITE_CALENDAR		ACCESS_COARSE_LOCATION
BLUETOOTH	BLUETOOTH	CAMERA	CAMERA
	BLUETOOTH_ADMIN	AUDIO	RECORD_AUDIO
VIBRATE	VIBRATE	BOOT	RECEIVE_BOOT_COMPLETED
WAKE_LOCK	WAKE_LOCK	SETTING	WRITE_SETTINGS

表 5 敏感权限与权限分组的映射关系

Table 5 Mapping relationship between sensitive permissions and permission groups

编号	权限组	权限	说明
1	CALENDAR	READ_CALENDAR	允许应用程序读取用户的日程信息
		WRITE_CALENDAR	允许程序写入日程, 但不可读取
2	CAMERA	CAMERA	允许程序访问摄像头进行拍照
3	CONTACTS	READ_CONTACTS	允许应用程序读取用户的联系人信息
		WRITE_CONTACTS	允许程序写入联系人, 但不可读取
		GET_ACCOUNTS	允许访问的帐户的 Gmail 列表服务
4	LOCATION	ACCESS_FINE_LOCATION	允许通过 GPS 接收卫星的定位信息
		ACCESS_COARSE_LOCATION	允许通过无线网络或移动基站的方式获取用户的经纬度信息
5	MICROPHONE	RECORD_AUDIO	允许程序录制声音通过手机或耳机的麦克
6	PHONE	READ_PHONE_STATE	允许程序访问电话状态
		CALL_PHONE	允许程序从非系统拨号器里拨打电话
		READ_CALL_LOG	允许应用程序读取用户的通话记录
		WRITE_CALL_LOG	允许程序写入联系人数据, 但不可读取
		ADD_VOICEMAIL	允许应用程序添加语音邮件系统
		USE_SIP	允许程序使用 SIP 视频服务
		PROCESS_OUTGOING_CALLS	允许程序监视, 修改或放弃播出电话
7	SENSORS	BODY_SENSORS	允许应用程序访问用户使用的传感器来测量
8	SMS	SEND_SMS	允许程序发送短信
		RECEIVE_SMS	允许程序接收短信
		READ_SMS	允许应用程序读取短信内容
		RECEIVE_WAP_PUSH	允许程序接收 WAP PUSH 信息
		RECEIVE_MMS	允许程序接收彩信
9	STORAGE	READ_EXTERNAL_STORAGE	允许应用程序读取设备外部存储空间的文件
		WRITE_EXTERNAL_STORAGE	允许程序写入外部存储

另外, 本文基于 Google 官方的 API 文档进行敏感 API 的筛选。因此, 筛选出来的敏感 API 只与 Google 官方 SDK(本文选用 SDK 24 作为参考标准)有关, 与系统应用无关。

3 动静特征提取方法

本文所提混合特征由两部分组成: 静态特征和动态特征。

3.1 基于 APK 文件的静态特征提取

静态特征提取主要包括两个方面: 一方面是面向高危权限的静态特征提取, 这一部分主要通过解析AndroidManifest.xml文件, 提取其中高危权限的申请情况来确定; 另一方面是面向敏感 API 调用的静态特征提取, 这一部分首先反编译classes.dex文件为对应的smali文件, 再按照smali语法格式匹配敏感 API 的调用情况来确定。

3.1.1 面向高危权限的静态特征提取

设在 2.1 节中提取的高危权限组集合为 $P=\{p_1, p_2, \ldots, p_{14}\}$, 面向高危权限的静态特征向量为 $F_{Static}(Per)=\{f_1, f_2, \cdots, f_{14}\}$, 且元素值满足 $f_i=\{0, 1\}\ (i=1,2,\cdots,14)$, 则本文提取 APK 文件中面向高危权限的静态特征过程如下:

(1) 使用Apk-Parser直接解析 APK 文件, 输出AndroidManifest.xml内容;

(2) 由于AndroidManifest.xml中对于权限的申请使用< uses-permission >来表示, 因此可以直接在其中找到该标识符对应的行, 提取出配置文件中所有的权限, 生成 APK 权限集合 $P_{APK}=\{p_1, p_2, \cdots, p_n\}$, 其中, n 表示从AndroidManifest.xml文件中提取的权限个数;

(3) 对于任一权限 $p_i \in P_{APK}$, 若满足 $p_i = p_j \in P\{i=1,2,\cdots,14\}$, 则有 $f_j=1$; 否则, $f_j=0\ (j=1,2,\cdots,14)$;

综上, 本文得到 APK 文件下面向高危权限的静态特征描述如下:

$$F_{Static}(Per)=\{f_1, f_2, \ldots, f_{14}\}$$

3.1.2 面向敏感 API 调用的静态特征提取

在 APK 文件中, 可以用于分析 API 调用情况的文件只有 classes.dex, 随着 APK 文件内容的越来越大, 有可能一个 APK 文件中出现多个 classes.dex 文件。因此, 本文基于 Dex 文件面向敏感 API 调用进行静态特征提取。

由于 Dex 文件是编译过后的字节码文件, 无法直接打开分析, 因此需要将Dex文件进行反编译, 通过对反编译后的文件进行分析, 才能得到敏感 API 调用的静态特征信息。当前反编译分析 Dex 文件的方式有三种: 一是直接将 Dex 文件 dump 到 txt 文件中, 然后分析 txt 文件, 这种方法的可阅读性非常差; 二是使用 dex2jar 将 Dex 文件解包成 jar, 再通过 XJad 进行 java 反编译得到源码, 这种方式可以直接阅读源码, 但是只能反编译出开发用的java文件, 但是lib包不能反编译出来; 三是只将 Dex 文件反编译成 Smali 格式文件, 这种格式的文件包含独有的一套语法, 可阅读性也很强。综上, 本文选择使用 bakSmali 将 Dex 文件反编译成Smali文件, 然后对Smali文件进行分析。

设在 2.1 节中得到的敏感 API 集合为 $APIs=\{api_1, api_2, \cdots, api_{40}\}$, 敏感API集对应的静态特征向量为$F_{Static}(APIs)=\{f_1, f_2, \cdots, f_{40}\}$, 且特征向量元素值满足$f_i=\{0, 1\}\ (i=1,2,\cdots,40)$。则本文面向敏感 API 的静态特征提取过程如下:

(1) 解压 APK 文件, 提取出其中所有 classes.dex 文件, 使用 bakSmali.jar 反编译成 Smali 文件;

(2) 遍历每一个 Smali 文件, 匹配所有.method开头的字符串, 截取所有 Android 系统 API, 得到 Smali 格式的 API 集合$APIs_{APK-Smali}=\{api_{s1}, api_{s2}, \cdots, api_{sn}\}$, 其中, n 为集合中元素个数, 即 APK 文件中 Smali 格式 API 个数;

对于 Smali 格式的每一个 API 元素$api_{sj} \in APIs_{APK-Smali}\ (j=1,2,\cdots,n)$, 参照敏感 API 集合Smali格式对照表 6, 将其转换为正常格式的 API, 设为api_j, 若满足$api_j = api_i \in APIs$, 则说明该 APK 文件中调用了敏感 API, 设对应的特征向量特征元素值为 1, 即$f_i=1$; 否则$f_i=0$。

表 6 部分敏感 API 对应的 Smali 格式对照表

Table 6 Comparison table of Smali format corresponding to some sensitive APIs

说明	敏感 API	Smali 格式 API
获取定位	android.location.LocationManager.getLastKnownLocation	Landroid/location/LocationManager;->getLastKnownLocation
获取电话状态	android.telephony.TelephonyManager.listen	Landroid/telephony/TelephonyManager;->listen
获取 WIFI 状态	android.net.wifi.WifiManager.getWifiState	Landroid/net/wifi/WifiManager;->getWifiState
无提示打开蓝牙	android.bluetooth.BluetoothAdapter.enable	Landroid/bluetooth/BluetoothAdapter;->enable
发送短信	android.telephony.SmsManager.sendTextMessage	Landroid/telephony/SmsManager;->sendTextMessage

续表

说明	敏感 API	Smali 格式 API
振动	android.os.SystemVibrator.vibrate	Landroid/os/SystemVibrator;->vibrate
发送非文本信息	android.telephony.gsm.SmsManager.sendDataMessage	Landroid/telephony/gsm/SmsManager;->sendDataMessage
接收彩信	android.telephony.SmsManager.downloadMultimediaMessage	Landroid/telephony/SmsManager;->downloadMultimediaMessage
始录音	android.media.MediaRecorder.start	Landroid/media/MediaRecorder;->start
结束录音	android.media.MediaRecorder.stop	Landroid/media/MediaRecorder;->stop
打开摄像头	android.hardware.camera2.CameraManager.openCamera	Landroid/hardware/camera2/CameraManager;->openCamera

本节静态特征提取主要描述了两个方面: 一方面是面向高危权限的静态特征提取; 另一方面是面向敏感 API 调用的静态特征提取。这两种静态特征提取已经在上述两个小节中进行了详细描述, 本节静态特征提取的描述如算法 3 所示。

算法 3: 静态特征提取算法

输入: 高危权限组集合$P=\{p_1,p_2,\ldots,p_{14}\}$,
apk 文件,
敏感 API 集合$APIs=\{api_1,api_2,\ldots,api_{40}\}$
输出: 静态特征向量$F_{\text{Static(Per)}}$、API 集合
1 分析 AndroidManifest.xml 文件内容
2 生成 APK 权限集$P_{APK}=\{p_1,p_2,\ldots,p_n\}$
3 IF $p_i \in P_{APK}$ AND$p_i=p_j \in P\ \{i=1,2,\ldots,14\}$ THEN
4 $f_j=1$
ELSE
5 $f_j=0\ (j=1,2,\ldots,14)$
6 RETURN $F_{Static}(Per)=\{f_1,f_2,\ldots,f_{14}\}$
7 WHILE $F_{Static}(APIs)=\{f_1,f_2,\ldots,f_{40}\}$ AND$f_i=\{0,1\}\ (i=1,2,\ldots,40)$ DO
8 APK → classes.dex 文件
9 classes.dex → bakSmali.jar → Smail 文件
10 .method → API
11 RETURN $APIs_{APK-Smali}=\{api_{s1},api_{s2},\ldots,api_{sn}\}$
12 IF $api_{sj} \in APIs_{APK-Smali}\ (j=1,2,\ldots,n)$ AND $api_j=api_i \in APIs$ THEN
13 $f_i=1$
ELSE
14 $f_i=0$
15 END

3.2 基于 App 的动态特征提取方法

传统基于 APK 文件提取静态特征进行检测的方法不足以完全描述恶意软件的全部特征, 且目前 APK 文件基本上都经过混淆处理, 单一依靠静态分析往往达不到理想效果。因此, 还需要提取恶意软件的动态特征来作为补充。

3.2.1 面向敏感 API 调用频次的动态特征提取

Android 恶意软件动态分析指在 App 运行的情况下, 对其动态行为及系统的相关响应进行的一系列监测以及状态行为分析技术[6,8,11]。

本文使用 Xposed 框架对敏感 API 调用频次特征进行提取。Xposed 在对 Java 方法进行 Hook 时, 先将 Dalvik 虚拟机中的 directMethod 改为 nativeMethod(通过修改标识字段实现); 然后将该方法的 nativeFunc 重定向指向由 Xposed 实现的一个 native 方法, 这样当 Java 方法在调用时, 就会调用到重定向后的 native 方法, Xposed 就接管了控制权; 在该 native 方法中, Xposed 直接对原 Java 方法进行调用, 并在调用前后插入“钩子”(自定义功能), 由此 Hook 到该 Java 方法。Hook 前后示意图如图 2 所示。

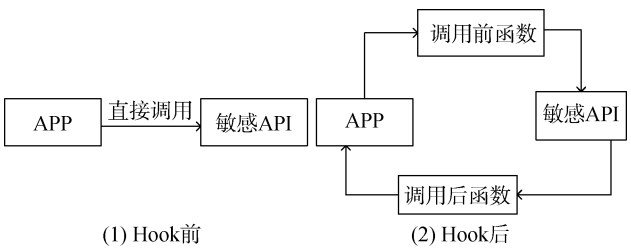

图 2 Hook 前后对比图
Figure 2 Comparison before and after Hook

设在 2.3 节中得到的敏感 API 集合为$APIs=\{api_1,api_2,\ldots,api_{40}\}$, 敏感 API 集对应的动态特征向量为$F_{Dynamic}=\{f_1,f_2,\ldots,f_{40}\}$, 且默认每个 App 对应的敏感 API 调用频次特征向量元素初始值为 $f_i=0\ (i=1,2,\ldots,40)$。则本文面向敏感 API 调用频次的动态特征提取过程如下:

(1) 获取 Android 移动端 root 权限, 安装 Xposed 框架, 部署自主开发的面向敏感 API 调用频次 Hook 的插件, 重启移动端, 运行 App 进行动态监控;

(2) 若有系统 API 被调用, 且满足$api_{App} = api_j \in APIs\ (j = 1,2,\ldots,40)$, 则记录本次调用的次数 k, 更新该 API 对应的特征元素值$f_j = f_j + k \cdot W_{SYS}$, 其中, W_{SYS}是根据 API 被调用时记录的系统状态特征向量值计算得到的特征权重。

(3) 不断执行步骤(2), 并将相关 App 信息和对应敏感 API 调用频次特征向量保存在后台数据库中。

则面向敏感 API 调用频次的动态特征信息描述如下:

$$F_{Dynamic} = \{f_1, f_2, \ldots, f_{40}\},\ f_i \in [0, +\infty)$$

3.2.2 面向系统状态的动态特征提取

基于 Xposed 框架对敏感 API 进行 Hook 以提取 App 运行时敏感 API 调用频次的动态特征。默认情况下, 所有敏感 API 调用时的场景都是一致的, 即不考虑恶意软件执行恶意行为时的系统状态情况。但事实上, 随着恶意软件攻击行为的发展, 本文发现恶意软件在不同系统状态下执行的恶意行为不一样, 同时执行恶意行为的频次也不一样。如在恶意软件 App 处于后台状态时, 其联网发送数据和非法下载的次数比在前台状态下的次数要高得多; 恶意软件 App 在用户未操作状态下主动发送短信次数比在用户操作状态下的次数要高得多。

本文实验中的移动端设备使用 Android7.0 版本的小米手机, 通过控制可变因素进行对比的方式来完成实验。首先检测在同一应用场景中, 良性软件与恶意软件调用 API 的种类与频率, 例如让恶意软件和良性软件均在手机上运行, 分别检测在后台使用时, 恶意软件与良性软件对 API 的调度情况, 通过测试多种不同恶意软件进行检测与分析。然后检测同一种软件在不同应用场景下, 对 API 调用的情况进行分析, 例如分别选择一个良性软件和一个恶意软件, 让这两个软件在前台运行时查看 API 的调用情况, 之后再分别查看这两个软件在亮屏、息屏、有用户操作、没有用户操作等不同场景下对 API 的调用情况。

本文对从 2.1 节收集到的恶意软件样本集中随机选取 5000 个样本作测试分析。在同一模拟器环境下, 分别测试恶意软件在前台/后台、亮屏/息屏和模拟用户操作/模拟用户未操作的活动情况(包括: 调用敏感 API 种类数、调用敏感 API 总频次), 测试时间为 10 min, 统计情况如图 3、图 4 所示。

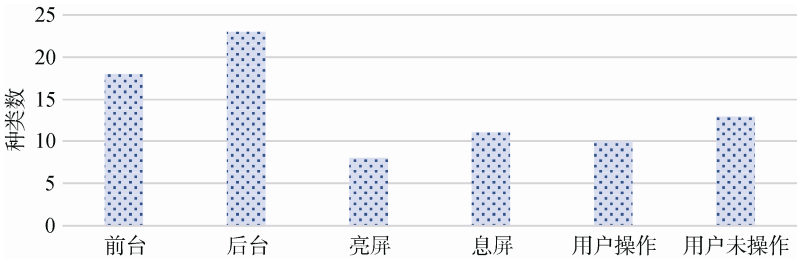

图 3 调用敏感 API 种类数

Figure 3 Number of sensitive API types called

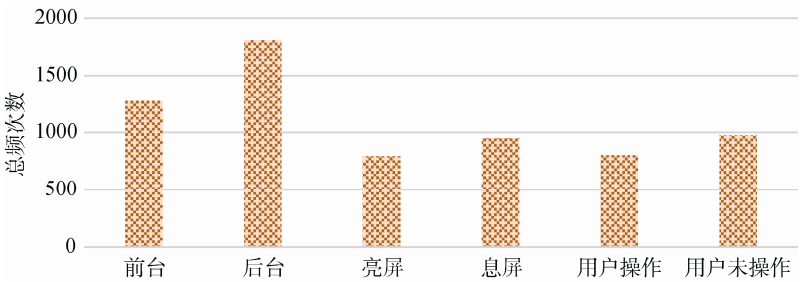

图 4 调用敏感 API 总频次数

Figure 4 Total number of calls to sensitive API

因此, 依据实验数据, 本文认为 Android 恶意软件在某些系统状态下执行恶意行为的可能性更大, 即在后台静默执行恶意行为的可能性比在前台执行的可能性要高; 在息屏状态下执行恶意行为的可能性比在亮屏状态下执行恶意行为的可能性更高; 在用户未操作状态下执行恶意行为的可能性比在用户操作状态下执行恶意行为的可能性更高。所以, 在 Hook 敏感 API 时监控系统的当前状态很有必要。

设系统状态特征向量为$\boldsymbol{F}_{SYS} = \{f_{active}, f_{user}, f_{bkg}\}$, 各特征向量值说明如表 7 所示。其中, 系统状态特征

向量值满足$f_i = \{0,1\}\ (i = 1,2,3)$。

表 7 系统状态特征向量值说明

Table 7 Description of system state eigenvector value

特征值	说明
f_{active}	是否处于息屏状态
f_{user}	是否被用户操作
f_{bkg}	是否处于后台状态

由于判断系统状态的方法也是可以通过 Xposed 框架 Hook 到的, 因此本文在 Hook 敏感 API 的同时, 也对系统状态的方法函数进行 Hook, 通过 Hook 方法的返回值来判断系统状态。

在得到面向系统状态的特征向量$\boldsymbol{F}_{SYS} = \{f_{active}, f_{user}, f_{bkg}\}$后, 由于本文对于该特征向量的定位是辅助优化面向敏感 API 调用频次的动态特征向量$\boldsymbol{F}_{Dynamic}(API)$的特征值。因此, 需要根据特征向量$\boldsymbol{F}_{SYS}$计算优化权重$W_{SYS}$。

首先, 本文认为f_{active}、f_{user}和f_{bkg}之间的重要程度是不一样的, 对于它们之间重要程度的判断时基于专家决策, 即按重要程度从大到小顺序如下:

$$f_{active} > f_{user} > f_{bkg}$$

因此, 本文采用“优序图法”来描述三者之间的权重关系, 如表 8 所示。

表 8 系统状态特征向量权重优序图

Table 8 Weight precedence graph of system state eigenvectors

	f_{active}	f_{user}	f_{bkg}
f_{active}	0.5	0.5	1
f_{user}	0	0.5	1
f_{bkg}	0	0	0.5

其次, 根据表 7 所示优序图, 计算动态敏感 API 调用时的特征值的辅助优化权重如下:

(1) 计算优序图中每一行数据的权重, 第$i\ (i = 1,2,3)$行对应的权重W_i如公式(8)所示:

$$W_i = \frac{\sum_{j=1} w_{i,j}}{\sum_{i=1}\sum_{j=1} w_{i,j}} \times 100\% \tag{8}$$

其中, $w_{i,j}$为优序图中第 i 行、第 j 列元素值;

(2) 计算动态敏感 API 调用时特征值的辅助优化权重, 如公式(9)所示。

$$W_{SYS} = \sum_{i=1}^{3} f_i \cdot W_i \tag{9}$$

其中, f_i为系统状态特征

$f_i \in \boldsymbol{F}_{SYS} = \{f_{active}, f_{user}, f_{bkg}\} = \{f_1, f_2, f_3\}$

W_i由公式(8)计算得出。

4 基于动静结合的特征混合方法

在 3.1 节中, 本文基于 APK 文件提取到面向高危权限和敏感 API 调用的静态特征向量:

$$\boldsymbol{F}_{Static} = \{f_1, f_2, \ldots, f_{14}, \ldots, f_{54}\}$$

在 3.2 节中, 本文基于 Xposed 框架获取到面向敏感 API 调用频次的动态特征向量$\boldsymbol{F}_{Dynamic} = \{f_1, f_2, \ldots, f_{40}\},\ f_i \in [0, +\infty)$和系统状态特征向量:

$$\boldsymbol{F}_{SYS} = \{f_{active}, f_{user}, f_{bkg}\}。$$

由此, 本文基于动静结合的混合特征组合信息描述如下:

$$F_{Hybrid} = \{\boldsymbol{F}_{Static}, \boldsymbol{F}_{Dynamic}(API)\}$$

系统状态特征向量$\boldsymbol{F}_{SYS}$用于$\boldsymbol{F}_{Dynamic}(API)$的特征值辅助调优。

4.1 混合特征归一化处理

由于本文需要通过计算特征向量之间的“距离”以进行分类检测, 若某一特征值范围特别大, 则距离计算就会偏向于该特征, 从而偏离实际分类效果。因此, 需要对特征向量进行归一化处理。

由 3.1 节可知, 静态特征向量$\boldsymbol{F}_{Static}$中的所有特征值的取值范围都在[0, 1]之间, 不需要对$\boldsymbol{F}_{Static}$进行归一化处理; 而动态特征向量$\boldsymbol{F}_{Dynamic}$中的所有特征值的取值范围在$[0, +\infty)$, 故需要将其归一化到[0, 1]区间内。

本文采用“离差标准化”方法进行归一化处理, 对于$f_i \in \boldsymbol{F}_{Dynamic}\ (i = 1,2,\ldots,40)$, 其归一化计算如公式(10)所知。

$$f_i' = \frac{f_i - \mathrm{f}\min}{f\min_{\max\ (i=1,2,\ldots,40)}} \tag{10}$$

其中, $f_{\max}$为$\boldsymbol{F}_{Dynamic}$中最大特征值, $f_{\min}$为$\boldsymbol{F}_{Dynamic}$中最小特征值;

因此, 经过归一化处理后的动态特征向量可描述为:

$$\boldsymbol{F}'_{Dynamic} = \{f_1', f_2', \ldots, f_{40}'\},\ f_i' \in [0,1]$$

4.2 混合特征权重赋值处理

由于本文在 3.2.2 节中提取了面向系统状态的特征向量$\boldsymbol{F}_{SYS}$, 并计算得到针对动态敏感 API 调用频次特征值的辅助优化权重W_{SYS}, 且将其运用到动态特征向量$\boldsymbol{F}_{Dynamic}$的提取过程中; 因此, 在 4.1 节中进行归一化处理之前的特征向量$\boldsymbol{F}_{Dynamic}$就已经经过权重赋值处理了; 所以, 本节只需要针对静态特征向量$\boldsymbol{F}_{Static}$进行权重赋值处理。

本文采用“TF-IDF”算法[15]对静态特征向量$\boldsymbol{F}_{Static}$进行权重赋值处理。选取 N 个恶意 APK 样本$D = \{d_1, d_2, \ldots, d_N\}$进行静态特征值权重赋值处理,

对于每一个特征值$f_i \in \boldsymbol{F}_{Static} = \{f_1, f_2, \ldots, f_{54}\}$ $(i = 1,2,\ldots,54)$, 其权重赋值处理过程如下:

(1) 计算f_i在 APK 样本文件d_j $(j = 1,2,\ldots,N)$中的 TF 值, 如公式(11)所示:

$$TF_{i,j} = \frac{t_{i,j}}{T_i} \tag{11}$$

其中, $t_{i,j}$为样本d_j中特征元素f_i对应的高危权限或敏感 API 出现的次数; T_i为$\boldsymbol{F}_{Static}$中所有特征元素对应的高危权限或敏感 API 在样本d_j中出现的次数。

(2) 计算f_i的 IDF 值, 如公式(12)所示:

$$IDF_i = \log\frac{|N|}{|N_i|+1} \tag{12}$$

其中, N为恶意 APK 样本总数; N_i为包含f_i对应的高危权限或敏感 API 的恶意 APK 样本数。

(3) 计算f_i的权重值, 如公式(13)所示:

$$w_i = TF_{i,j} \cdot IDF_i \tag{13}$$

因此, 静态特征向量$\boldsymbol{F}_{Static}$对应的权重向量为$\boldsymbol{W}_{Static} = \{w_1, w_2, \ldots, w_{54}\}$, 则对$\boldsymbol{F}_{Static}$进行权重赋值计算为

$$\boldsymbol{F}'_{Static} = \boldsymbol{F}_{Static} \times \boldsymbol{W}_{Static} = \{f'_1, f'_2, \ldots, f'_{54}\}$$

综上, 经过归一化处理和权重赋值处理后的混合特征向量为:

$$\boldsymbol{F}'_{Hybrid} = \{\boldsymbol{F}'_{Static}, \boldsymbol{F}'_{Dynamic}\} = \{f'_1, f'_2, \ldots, f'_{54}, f'_{55}, \ldots, f'_{94}\}$$

5 实验与结果分析

5.1 实验样本与配置环境

本文实验所用数据均为 APK 文件, 根据 APK 文件的是否为恶意的将它们分为良性 APK 数据集和恶意 APK 数据集。其中, 良性 APK 数据集是从国内外 Android 应用市场(豌豆荚和 Google Play)上下载的, 而恶意 APK 数据集是从著名恶意软件发布网站 VirusShare和Drebin上下载。

由于从网站上下载的 APK 文件参差不齐, 有部分文件是经过高度加固防护的, 部分文件缺少关键配置, 还有一些文件无法在模拟器或移动终端上运行。因此, 本文要对上述 APK 文件进行筛选去重。首先, 使用ApkTool工具反编译各个文件, 获取文件基本信息, 去除无法反编译和基本信息不全的文件; 其次, 根据文件版本信息和校验码对比信息, 去除重复的 APK 文件; 最后, 在模拟器中安装运行各个 APK 文件, 去除无法安装或安装失败以及无法运行的文件。经过筛选去重步骤后, 本文最终用于实验部分的数据共计 8000 个。其中, 恶意 APK 文件数量为 4000 个, 良性 APK 文件数量为 4000 个。通过静态特征提取、动态特征提取和特征数据预处理等步骤, 共计获得用于训练分类模型的训练集包含有 8000 个样本数据。

本文的实验环境分为两部分: 一部分为 Hadoop 云平台, 由三台集群服务器组成, 配置如表 9 所示; 另一部分为移动端测试设备, 配置如表 10 所示。

表 9 Hadoop 云平台设备配置

Table 9 Hadoop Device configuration of cloud platform

操作系统	内存	JDK 版本	Hadoop 版本	Spark 版本
CentOS 7	32GB	1.8.0_181	2.9.1	1.5.1

表 10 移动端测试设备配置

Table 10 Mobile terminal test equipment configuration

操作系统	移动端系统	内存	外存
MIUI 10.1	Android 7.0	4GB	32GB

5.2 分类模型评价指标

设 TP(True Positive)为恶意软件被预测为恶意的样本数, TN(True Negative)为良性软件被预测为良性的样本数, FP(False Positive)为良性软件被预测为恶意的样本数, FN(False Negative)为恶意软件被预测为良性的样本数。则本文针对移动端恶意软件检测模型的分类评价指标如表 11 所示。

表 11 本文检测模型分类评价指标

Table 11 The classification evaluation index of the detection model is given in this paper

指标	说明
TPR	$TPR = \frac{TP}{TP+FN}$
FPR	$FPR = \frac{FP}{TN+FP}$
ACC	$ACC = \frac{TP+TN}{TP+FP+TN+FN}$
Precision	$P = \frac{TP}{TP+FP}$
Recall	$R = \frac{TP}{TP+FN}$
F_1-Score	$F_1 = \frac{2P*R}{P+R}$

其中, TPR = Recall, 表示当前被预测为恶意的样本中, 真正为恶意的样本占所有恶意样本的比例。

5.3 对比试验和分析

本文使用统一分类模型(基于 CART 的随机森林集成分类模型[16])对所提的动静混合特征提取方法进行对比实验分析。具体的对比试验如下:

(1) 使用改进的 CHI 统计方法筛选高危权限集

使用传统 CHI 统计方法筛选高危权限集存在弊

端(低频权限被低估、与类别负相关权限被高估), 使得筛选出来的高危权限列表中包含了较多的正常权限, 与 Google 官方给出的危险权限相比, 重合率并不是特别高。本文使用改进的 CHI 统计方法筛选出高危权限集, 对比传统 CHI 方法筛选出来的高危权限集与Google官方危险权限重合率对比如表12所示。

表 12 CHI 方法改进前后高危权限集与官方的重合率

Table 12 The high-risk permission set and official coincidence rate before and after the improvement of CHI method (%)

CHI 方法	高危权限在官方危险权限中占比	官方危险权限在高危权限集中占比
改进前	62.5	50.0
改进后	83.3	66.7

可以看出, 采用改进后的 CHI 统计方法筛选出来的高危权限集与 Google 官方给出的危险权限有着更高的重合率, 证明本文筛选出来的高危权限更加贴近实际应用。

(2) 去除权限之间的相关性

1) 去除权限间相关性前后对于分类效果的影响: 由于 Android 权限之间的申请并不是完全独立的, 往往开发人员一次性申请一组权限, 同组内的权限之间相关性很强。因此, 本文对比去除权限之间相关性前后对于分类效果的影响, 如表 13 所示。

表 13 去除权限相关性前后对于分类效果的影响

Table 13 The influence of removing authority relevance on classification effect (%)

去除权限相关性	TPR	FPR	Precision	Recall	ACC	F1
去除前	90.6	9.0	91.0	90.6	90.8	91.0
去除后	96.0	3.5	96.4	96.0	96.3	96.2

可以看出, 本文在去除权限间的相关性后, 不仅使特征向量的维数降低了, 而且在最终分类效果上, 也有明显提高。

2) 使用凝聚层次聚类算法优化的 K-Means 算法去除权限间的相关性: 使用传统 K-Means 算法对高危权限之间的相关性进行去重处理, 其处理的效果很大程度上受到初始聚类中心的影响。因此, 本文使用凝聚层次聚类算法优化 K-Means 算法来对比传统 K-Means 算法对于分类效果的影响, 如表 14 所示。

可以看出, 本文在使用凝聚层次聚类算法优化过 K-Means 算法后对高危权限集进行聚类处理, 其最终体现在分类效果上也有明显提升。

表 14 优化 K-Means 算法前后对于分类效果的影响

Table 14 Influence of K-Means algorithm optimization on classification effect (%)

K-Means 算法	TPR	FPR	Precision	Recall	ACC	F1
优化前	93.0	8.0	92.1	93.0	92.5	92.5
优化后	96.0	3.5	96.4	96.0	96.3	96.2

(3) 使用系统状态特征辅助优化动态特征的提取

为了更好的反映敏感 API 在 App 运行中被调用时的动态特征, 本文在 Hook 敏感 API 的同时也记录下系统当前的状态特征, 用以实时调整动态特征数据。对比使用系统状态特征辅助优化动态特征提取前后对于最终分类效果的影响, 如表 15 所示。

表 15 使用系统状态特征优化动态特征提取前后对于分类效果的影响

Table 15 Influence of dynamic feature extraction optimized by system state feature on classification effect (%)

系统状态特征	TPR	FPR	Precision	Recall	ACC	F1
使用前	91.5	7.5	92.4	91.5	92.0	91.9
使用后	96.0	3.5	96.4	96.0	96.3	96.2

可以看出, 本文在使用系统状态特征优化动态特征提取的数据后, 在最终的分类效果上有着明显提升。

(4) 使用动静混合特征作为生成混合特征向量标准

针对单一特征无法全面描述恶意软件尽可能多特征的问题, 本文使用动静混合特征作为生成特征向量的标准。对比使用单一特征和混合特征生成的特征向量对于最终分类效果的影响, 如表 16 所示。

表 16 使用单一特征和混合特征对于分类效果的影响

Table 16 The influence of using single feature and mixed feature on classification effect (%)

特征选择	TPR	FPR	Precision	Recall	ACC	F1
静态特征	85.5	16.0	84.2	85.5	84.8	84.8
动态特征	90.5	11.0	89.1	90.5	89.8	89.8
本文混合特征	96.0	3.5	96.4	96.0	96.3	96.2

可以看出, 本文在使用动静混合特征作为特征提取的标准后, 在最终的分类效果上有着显著提升。

本文对上述对比实验采用 5 次、10 次、15 次和 20 次十折交叉验证法, 对检测精度 ACC 分别取平均

值, 最终分类效果的柱状对比如图 5 所示, 折线对比如图 6 所示。其中, Case1 代表未去除权限相关性的方法; Case2 代表使用未优化 K-Means 算法去除权限相关性的方法; Case3 代表未使用系统状态特征来优化动态特征提取的方法; Case4 代表只提取静态特征的方法; Case5 代表只提取动态特征的方法。

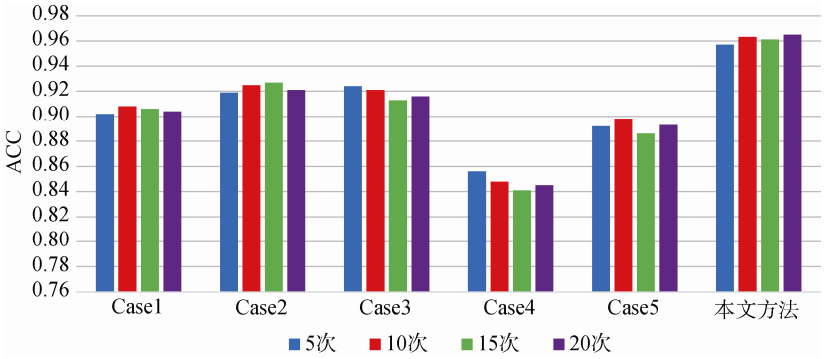

图 5　十折交叉验证柱状对比图

Figure 5　10 fold cross validation column chart

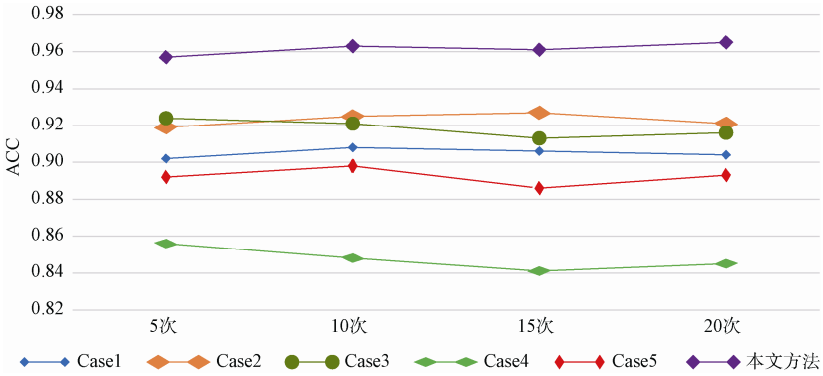

图 6　十折交叉验证折线对比图

Figure 6　10 fold cross validation line comparison chartt

最后, 本文分别将所提方案的实验结果同 Jin Li[17]、Andrea Saracino[18]和 ElMouatez Billah Karbab[19]等人的实验结果进行对比, 结合本文实验结果, 对比结果如表 17 所示。其中, Jin Li 等人提出一种基于权限使用分析的恶意软件检测系统, 通过挖掘识别最有效的权限以区分良性和恶意软件; Andrea Saracino 等人提出一种通过监视属于不同 Android 级别的功能来定义恶意行为的方法以检测恶意软件的方法; ElMouatez Billah Karbab 等人提出了一个自动 Android 恶意软件检测和家族归因框架, 该框架依赖于使用深度学习技术进行的序列分类。通过分析可以看出, Jin Li 的准确率(Precision)很高, 但是在召回率(Recall)上并没有本文所提方法高, 且仅挖掘了权限特征, 并没有更多地探讨其他特征对于恶意软件的表述; 而 Andrea Saracino 检测率在所提框架中很高, 但是选取的特征项并不足以完全描述恶意软件尽可能多的特征; ElMouatez Billah Karbab 等人在不同的样本集上进行了测试, 但是召回率(Recall)表现得并不稳定。本文所提方法无论在特征项的选取上, 还是在检测准确率和召回率上都表现全面和稳定。

表 17　与主流 Android 恶意软件检测情况对比

Table 17　Comparison with mainstream Android malware detection methods

	Dynamic/Static	Precision(%)	Recall(%)	Stable
本文	Both	96.40	96.00	Yes
SigPID[17]	Static	97.54	93.62	N.A.
MADAM[18]	Both	96.90	N.A.	N.A.
MalDozer[19]	Static	97.62	89.34	No

综上, 通过对比试验分析可知, 本文所提移动端恶意软件混合特征的提取方法相较于传统方法来说, 不仅更加全面地描述了恶意软件的特征, 而且在最终检测精度上也有着很大提升。

6　结束语

本文提出一种基于动静混合特征提取的 Android 恶意软件检测方法。首先, 分别基于改进 CHI

方法、组平均凝聚层次聚类算法优化的 K-Means 算法构建高危权限和敏感 API 库。然后分别从 AndroidManifest 配置文件中提取静态高危权限特征, 从反编译 Dex 文件后的 Smali 文件中提取静态敏感 API 调用特征; 通过 Xposed 框架实时监控 APP 运行时行为, 提取敏感 API 调用频次特征, 并基于实时的系统状态特征优化动态特征数据。最后, 对动态特征和静态特征进行归一化和权重赋值处理形成混合特征。在对比实验部分, 分别从改进 CHI、优化 K-Means 算法、使用系统状态特征优化动态特征和使用混合特征等 4 个方面对比分析本文所提方法与传统方法的优劣性。实验表明: 本文所提出的基于动态混合特征的 Android 恶意软件检测方法具有较高的检测精度, 各项评价指标达到了预期要求。

参考文献

[1] China Internet Information Center. The 46th “Statistical Report on Internet Development in China” [EB/OL].http://www.gov.cn/ xinwen/2020-09/29/content_5548176.htm, 2020-9-29.
(中国互联网信息中心.第 46 次《中国互联网络发展状况统计报告》 [EB/OL].http://www.gov.cn/xinwen/2020-09/29/content_5548176.htm,2020-9-29.)

[2] 360 Fiberhome Lab, 360 Security Brain. “2019 Android Malware Special Report” [EB/OL]. https://cert.360.cn/report/detail?id=d66c8ba239680d6674f2dba9f2be5f7, 2020-2-26.
(360 烽火实验室, 360 安全大脑.《2019 年度 Android 恶意软件专题报告》[EB/OL].https://cert.360.cn/report/detail?id= 0d66c8ba239680d6674f2dba9f2be5f7,2020-2-26.)

[3] Yang H, Zhang Y Q, Hu Y P, et al. Android Malware Detection Method Based on Permission Sequential Pattern Mining Algorithm[J]. *Journal on Communications*, 2013, 34(S1): 106-115.
(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(S1): 106-115.)

[4] Cui Y P, Yan B, Hu J W. Android Malware Detection Method Based on Abstract Api Call Sequences[J]. *Computer Applications and Software*, 2019, 36(9): 321-326.
(崔艳鹏, 颜波, 胡建伟. 基于抽象 API 调用序列的 Android 恶意软件检测方法[J]. *计算机应用与软件*, 2019, 36(9): 321-326.)

[5] Song X, Zhao K, Zhang L L, et al. Research on Android Malware Detection Based on Random Forest[J]. *Netinfo Security*, 2019(9): 1-5.
(宋鑫, 赵楷, 张琳琳, 等. 基于随机森林的 Android 恶意软件检测方法研究[J]. *信息网络安全*, 2019(9): 1-5.)

[6] Zhang Y L, Yin C H. Android Malware Outlier Detection Based on Feature Frequency[J]. *CAAI Transactions on Intelligent Systems*, 2018, 13(2): 168-173.
(张玉玲, 尹传环. 依特征频率的安卓恶意软件异常检测的研究[J]. *智能系统学报*, 2018, 13(2): 168-173.)

[7] Yan Z, Zhu B P. One of Android Malware Detection Method Based on Improved Permission Association Rules Mining Algorithm[J]. *Computer & Digital Engineering*, 2018, 46(6): 1167-1172.
(严喆, 朱保平. 一种基于改进的关联规则挖掘算法的 Android 恶意软件检测方法[J]. *计算机与数字工程*, 2018, 46(6): 1167-1172.)

[8] Wu F, Lu J X, Cao W J. Multi Feature Detection of Malicious Programs Based on Android Platform[J]. *Journal of Chinese Computer Systems*, 2018, 39(1): 151-155.
(吴帆, 陆济湘, 曹文静. Android 平台的恶意程序多特征检测[J]. *小型微型计算机系统*, 2018, 39(1): 151-155.)

[9] Feng Y, Bastani O, Martins R, et al. Automated Synthesis of Semantic Malware Signatures Using Maximum Satisfiability[C]. *2017 Network and Distributed System Security Symposium*, 2017.

[10] Alam S, Qu Z Y, Riley R, et al. DroidNative: Automating and Optimizing Detection of Android Native Code Malware Variants[J]. *Computers & Security*, 2017, 65: 230-246.

[11] Bernardi M L, Cimitile M, Distante D, et al. Dynamic Malware Detection and Phylogeny Analysis Using Process Mining[J]. *International Journal of Information Security*, 2019, 18(3): 257-284.

[12] Jerbi M, Dagdia Z C, Bechikh S, et al. On the Use of Artificial Malicious Patterns for Android Malware Detection[J]. *Computers & Security*, 2020, 92: 101743.

[13] Milosevic N, Huang J F. Deep Learning Guided Android Malware and Anomaly Detection[EB/OL]. 2019: arXiv:1910.10660.

[14] Ma Y, Zhao H, Li W L, et al. Optimization of TF-IDF Algorithm Combined with Improved CHI Statistical Method[J]. *Application Research of Computers*, 2019, 36(9): 2596-2598, 2603.
(马莹, 赵辉, 李万龙, 等. 结合改进的 CHI 统计方法的 TF-IDF 算法优化[J]. *计算机应用研究*, 2019, 36(9): 2596-2598, 2603.)

[15] Shi C Y, Xu C J, Yang X J. Study of TFIDF Algorithm[J]. *Journal of Computer Applications*, 2009, 29(S1): 167-170, 180.
(施聪莺, 徐朝军, 杨晓江. TFIDF 算法研究综述[J]. *计算机应用*, 2009, 29(S1): 167-170, 180.)

[16] Song X, Zhao K, Zhang L L, et al. Research on Android Malware Detection Based on Random Forest[J]. *Netinfo Security*, 2019(9): 1-5.
(宋鑫, 赵楷, 张琳琳, 等. 基于随机森林的 Android 恶意软件检测方法研究[J]. *信息网络安全*, 2019(9): 1-5.)

[17] Li J, Sun L C, Yan Q B, et al. Significant Permission Identification for Machine-Learning-Based Android Malware Detection[J]. *IEEE Transactions on Industrial Informatics*, 2018, 14(7): 3216-3225.

[18] Saracino A, Sgandurra D, Dini G, et al. MADAM: Effective and Efficient Behavior-Based Android Malware Detection and Prevention[J]. *IEEE Transactions on Dependable and Secure Computing*, 2018, 15(1): 83-97.

[19] Karbab E B, Debbabi M, Derhab A, et al. MalDozer: Automatic Framework for Android Malware Detection Using Deep Learning[J]. *Digital Investigation*, 2018, 24: S48-S59.

姚烨 于 2010 年在西北工业大学计算机科学与技术专业获得博士学位。现任西北工业大学计算机学院副教授。研究领域为网络信息安全。研究兴趣包括: 软件安全性测试、网络系统安全性测评、工业互联网及安全技术。Email: yaoye@nwpu.edu.cn

朱怡安 于 1999 年在西北工业大学计算机科学与技术专业获得博士学位。现任西北工业大学计算机学院教授(博导)。研究领域为嵌入式系统及安全。研究兴趣包括: 并行计算, 网络与信息安全, 复杂系统建模分析, 安全关键操作系统。Email: zhuya@nwpu.edu.cn

贾耀 于 2021 年在西北工业大学计算机技术专业获得工程硕士学位。现为国网山西省忻州市供电公司专业技术人员。研究领域为信息安全。研究兴趣包括: 深度学习、信息安全。Email: 834158371@qq.com

牛军涛 于 2019 年在西北工业大学环境科学专业获得理学学士学位。现在西北工业大学计算机技术专业攻读工学硕士学位。研究领域为区块链技术、网络安全。研究兴趣包括: 身份认证, 数据安全防护。Email: niujuntao@mail.nwpu.edu.cn

钱亮 于 2021 年在西北工业大学计算机技术专业获得工程硕士学位。现任南京莱斯信息股份有限公司单位助理研究员。研究领域为 JAVA 开发。研究兴趣包括: 计算机安全。Email: 18506170892@163.com

张黎翔 于 2018 年在西北工业大学计算机科学与技术专业获得硕士学位。现在西北工业大学计算机科学与技术专业攻读博士学位。研究领域为: 嵌入式系统及安全, 网络信息安全。研究兴趣包括: 复杂系统建模分析, 大数据分析及安全技术。Email: lilian@nwpu.edu.cn

杜家伟 于 2019 年在陕西科技大学计算机科学与技术获得学士学位。现在西北工业大学计算机技术专业攻读硕士学位。研究领域为网络安全。研究兴趣包括网络安全、深度学习。Email: dujiawei@mail.nwpu.edu.cn

附录 1　权限列表

名称	权限	描述	分类
访问登记属性	ACCESS_CHECKIN_PROPERTIES	允许读取或写入登记 check-in 数据库属性表的权限	普通权限
获取粗略位置	ACCESS_COARSE_LOCATION	允许通过无线网络或移动基站的方式获取用户的经纬度信息	危险权限
获取精确位置	ACCESS_FINE_LOCATION	允许通过 GPS 接收卫星的定位信息	危险权限
访问定位额外命令	ACCESS_LOCATION_EXTRA_COMMANDS	允许程序访问额外的定位提供者指令	普通权限
获取模拟定位信息	ACCESS_MOCK_LOCATION	允许获取模拟定位信息, 一般用于帮助开发者调试应用	普通权限
获取网络状态	ACCESS_NETWORK_STATE	允许获取网络信息状态	普通权限
访问通知策略	ACCESS_NOTIFICATION_POLICY	允许访问通知策略的应用程序的标记许可	普通权限
获取 WiFi 状态	ACCESS_WIFI_STATE	允许获取当前 WiFi 接入的状态以及 WLAN 热点的信息	普通权限
账户管理	ACCOUNT_MANAGER	允许程序通过账户验证方式访问账户管理 ACCOUNT_MANAGER 相关信息	普通权限
验证账户	AUTHENTICATE_ACCOUNTS	允许一个应用程序添加语音邮件系统	普通权限
允许接听来电	ANSWER_PHONE_CALLS	允许接听来电	普通权限
电量统计	BATTERY_STATS	允许获取电池电量统计信息	普通权限
只有系统可以绑定到 accessibilityservice	BIND_ACCESSIBILITY_SERVICE	请求 accessibilityservice 服务, 以确保只有系统可以绑定到它	普通权限
绑定小插件	BIND_APPWIDGET	允许应用程序告诉 AppWidget 服务应用程序可以访问 AppWidget 的数据	普通权限
绑定设备管理	BIND_DEVICE_ADMIN	请求系统管理员接收者 receiver, 只有系统才能使用	普通权限
绑定输入法	BIND_INPUT_METHOD	请求 InputMethodService 服务, 只有系统才能使用	普通权限
只有系统绑定到 HostApduService 或 OffHostApduService	BIND_NFC_SERVICE	由 HostApduService 或 OffHostApduService, 必须确保只有系统绑定到它	普通权限
只有系统绑定到 NotificationListenerService	BIND_NOTIFICATION_LISTENER_SERVICE	必须要求由 NotificationListenerService, 以确保只有系统可以绑定到它	普通权限
绑定 RemoteView	BIND_REMOTEVIEWS	必须通过 RemoteViewsService, 服务来请求, 只有系统才能用	普通权限
只有系统可以绑定到 TextService	BIND_TEXT_SERVICE	必须要求 TextService, 以确保只有系统可以绑定到它	普通权限
确保只有系统可以绑定到 VpnService	BIND_VPN_SERVICE	必须要求 VpnService, 以确保只有系统可以绑定到它	普通权限
绑定壁纸	BIND_WALLPAPER	必须要求 WallpaperService, 以确保只有系统可以绑定到它	普通权限
使用蓝牙	BLUETOOTH	允许应用程序连接到蓝牙设备配对	普通权限
蓝牙管理	BLUETOOTH_ADMIN	允许程序进行发现和配对新的蓝牙设备	普通权限
设置蓝牙设备不需要交互	BLUETOOTH_PRIVILEGED	允许应用程序对蓝牙设备, 不需要用户交互	普通权限
使用的传感器来测量	BODY_SENSORS	允许应用程序访问用户使用的传感器来测量	危险权限
应用删除时广播	BROADCAST_PACKAGE_REMOVED	允许应用程序广播通知应用程序包被移除后	普通权限
收到短信时广播	BROADCAST_SMS	允许程序当收到短信时触发一个广播	普通权限
连续广播	BROADCAST_STICKY	允许程序收到广播后快速收到下一个广播	普通权限
WAP PUSH 广播	BROADCAST_WAP_PUSH	允许应用程序播放一个 WAP 推收到通知	普通权限
拨打电话	CALL_PHONE	允许程序从非系统拨号器里拨打电话	危险权限
通话权限	CALL_PRIVILEGED	允许程序拨打电话, 替换系统的拨号器界面	普通权限
拍照权限	CAMERA	允许程序访问摄像头进行拍照	危险权限
捕获音频输出	CAPTURE_AUDIO_OUTPUT	允许应用程序捕获音频输出	普通权限
获取安全的视频输出	CAPTURE_SECURE_VIDEO_OUTPUT	允许应用程序获取安全的视频输出	普通权限
捕获视频输出	CAPTURE_VIDEO_OUTPUT	允许一个应用程序来捕获视频输出	普通权限
改变组件状态	CHANGE_COMPONENT_ENABLED_STATE	允许一个应用程序改变组件是否启用状态	普通权限
改变配置	CHANGE_CONFIGURATION	允许应用程序修改当前配置	普通权限

续表

名称	权限	描述	分类
改变网络状态	CHANGE_NETWORK_STATE	允许应用程序改变网络连接状态	普通权限
改变 WiFi 多播状态	CHANGE_WIFI_MULTICAST_STATE	允许应用程序进入无线多播模式	普通权限
改变 WiFi 状态	CHANGE_WIFI_STATE	允许应用程序改变 WiFi 连接状态	普通权限
清除应用缓存	CLEAR_APP_CACHE	允许应用程序清除所有已安装的 应用程序缓存	普通权限
清除用户数据	CLEAR_APP_USER_DATA	允许应用程序清除用户数据	普通权限
控制定位更新	CONTROL_LOCATION_UPDATES	允许程序获得移动网络定位信息改变	普通权限
删除缓存文件	DELETE_CACHE_FILES	允许应用程序删除缓存文件	普通权限
删除应用	DELETE_PACKAGES	允许应用程序删除应用	普通权限
电源管理	DEVICE_POWER	允许低级访问电源管理	普通权限
应用诊断	DIAGNOSTIC	允许应用程序 RW 诊断资源	普通权限
禁用键盘锁	DISABLE_KEYGUARD	允许应用程序禁用键盘锁	普通权限
转存系统信息	DUMP	允许程序获取系统 dump 信息从系统服务	普通权限
状态栏控制	EXPAND_STATUS_BAR	允许应用程序展开或折叠状态栏	普通权限
工厂测试模式	FACTORY_TEST	允许程序运行工厂测试模式	普通权限
使用闪光灯	FLASHLIGHT	允许访问的闪光灯	普通权限
强制后退	FORCE_BACK	允许应用程序强制操作任何事情	普通权限
访问账户 Gmail 列表	GET_ACCOUNTS	允许访问的帐户的 Gmail 列表服务	危险权限
获取应用大小	GET_PACKAGE_SIZE	允许一个程序获取任何 package 占用空间容量	普通权限
获取任务信息	GET_TASKS	允许应用程序获取当前的信息或最近运行的任务	普通权限
获取私人信息	GET_TOP_ACTIVITY_INFO	允许应用程序获取私人信息	普通权限
允许全局搜索	GLOBAL_SEARCH	允许程序允许全局搜索	普通权限
硬件测试	HARDWARE_TEST	允许访问硬件辅助设备	普通权限
注射事件	INJECT_EVENTS	允许访问本程序的底层事件, 获取按键、轨迹球的事件流	普通权限
安装定位提供	INSTALL_LOCATION_PROVIDER	允许程序安装定位提供	普通权限
安装应用程序	INSTALL_PACKAGES	允许程序安装应用	普通权限
快捷方式	INSTALL_SHORTCUT	允许应用程序快捷方式	普通权限
内部系统窗口	INTERNAL_SYSTEM_WINDOW	允许程序打开内部窗口, 不对第三方应用程序开放此权限	普通权限
访问网络	INTERNET	允许程序访问网络连接, 可能产生 GPRS 流量	普通权限
结束后台进程	KILL_BACKGROUND_PROCESSES	允许一个应用程序调用 killBackgroundProcesses()方法结束后台进程	普通权限
使用定位功能的硬件	LOCATION_HARDWARE	允许一个应用程序中使用定位功能的硬件, 不使用第三方应用	普通权限
管理账户	MANAGE_ACCOUNTS	允许程序管理 AccountManager 中的账户列表	普通权限
管理程序引用	MANAGE_APP_TOKENS	管理创建、摧毁、Z 轴顺序, 仅用于系统	普通权限
管理文件访问	MANAGE_DOCUMENTS	允许程序管理文件的访问	普通权限
软格式化	MASTER_CLEAR	允许程序执行软格式化, 删除系统配置信息	普通权限
知道什么是播放和控制其内容	MEDIA_CONTENT_CONTROL	允许一个应用程序知道什么是播放和控制其内容 ,不能被第三方应用使用	普通权限
修改声音设置	MODIFY_AUDIO_SETTINGS	允许程序修改声音设置信息	普通权限
修改电话状态	MODIFY_PHONE_STATE	允许程序修改电话状态	普通权限
格式化文件系统	MOUNT_FORMAT_FILESYSTEMS	允许程序格式化可移动文件系统	普通权限
挂载文件系统	MOUNT_UNMOUNT_FILESYSTEMS	允许程序挂载、反挂载外部文件系统	普通权限
允许 NFC 通讯	NFC	允许程序执行 NFC 近距离通讯操作	普通权限
永久 Activity	PERSISTENT_ACTIVITY	允许程序创建一个永久的 Activity, 该功能标记为将来将被移除	普通权限
处理拨出电话	PROCESS_OUTGOING_CALLS	允许程序监视, 修改或放弃播出电话	危险权限
读取日程提醒	READ_CALENDAR	允许应用程序读取用户的日程信息	危险权限

续表

名称	权限	描述	分类
读取通话记录	READ_CALL_LOG	允许应用程序读取用户的通话记录	危险权限
读取联系人	READ_CONTACTS	允许应用程序读取用户的联系人信息	危险权限
读取设备外部存储空间的文件	READ_EXTERNAL_STORAGE	允许应用程序读取设备外部存储空间的文件	危险权限
屏幕截图	READ_FRAME_BUFFER	允许程序读取帧缓存用于屏幕截图	普通权限
读取收藏夹和历史记录	READ_HISTORY_BOOKMARKS	允许应用程序读取浏览器收藏夹和历史记录	普通权限
读取输入状态	READ_INPUT_STATE	读取当前键的输入状态, 仅用于系统	普通权限
读取系统日志	READ_LOGS	允许应用程序读取系统底层日志	普通权限
读取电话状态	READ_PHONE_STATE	允许程序访问电话状态	危险权限
读取用户的个人资料信息	READ_PROFILE	允许应用程序读取用户的个人资料信息	普通权限
读取短信内容	READ_SMS	允许应用程序读取短信内容	危险权限
读取同步设置	READ_SYNC_SETTINGS	允许应用程序读取同步设置, 读取 Google 在线同步设置	普通权限
读取同步状态	READ_SYNC_STATS	允许程序读取同步状态, 获得 Google 在线同步状态	普通权限
读取用户字典	READ_USER_DICTIONARY	允许应用程序读取用户字典	普通权限
重启设备	REBOOT	允许能够重新启动设备	普通权限
开机自动允许	RECEIVE_BOOT_COMPLETED	允许程序开机自动运行	普通权限
接收彩信	RECEIVE_MMS	允许程序接收彩信	危险权限
接收短信	RECEIVE_SMS	允许程序接收短信	危险权限
接收 Wap Push	RECEIVE_WAP_PUSH	允许程序接收 WAP PUSH 信息	危险权限
录音	RECORD_AUDIO	允许程序录制声音通过手机或耳机的麦克	危险权限
排序系统任务	REORDER_TASKS	允许程序重新排序系统 Z 轴运行中的任务	普通权限
结束系统任务	RESTART_PACKAGES	允许程序结束任务通过 restartPackage(String)方法, 该方式将在外来放弃	普通权限
来电时候用进行即时短信息回复	SEND_RESPOND_VIA_MESSAGE	允许用户在来电的时候用你的应用进行即时的短信息回复	普通权限
发送短信	SEND_SMS	允许程序发送短信	危险权限
设置 Activity 观察其	SET_ACTIVITY_WATCHER	设置 Activity 观察器一般用于 monkey 测试	普通权限
设置闹铃提醒	SET_ALARM	允许程序设置闹铃提醒	普通权限
设置总是退出	SET_ALWAYS_FINISH	允许程序设置程序在后台是否总是退出	普通权限
设置动画缩放	SET_ANIMATION_SCALE	允许程序设置全局动画缩放	普通权限
设置调试程序	SET_DEBUG_APP	允许程序设置调试程序, 一般用于开发	普通权限
设置屏幕方向	SET_ORIENTATION	设置屏幕方向为横屏或标准方式显示, 不用于普通应用	普通权限
设置不能被第三方应用获取	SET_POINTER_SPEED	系统权限, 不能被第三方应用获取	普通权限
设置应用参数	SET_PREFERRED_APPLICATIONS	允许程序设置应用的参数	普通权限
设置进程限制	SET_PROCESS_LIMIT	允许程序设置最大的进程数量的限制	普通权限
设置系统时间	SET_TIME	允许程序设置系统时间	普通权限
设置系统时区	SET_TIME_ZONE	允许程序设置系统时区	普通权限
设置桌面壁纸	SET_WALLPAPER	允许程序设置桌面壁纸	普通权限
设置壁纸建议	SET_WALLPAPER_HINTS	允许程序设置壁纸建议	普通权限
发送一个永久的进程信号	SIGNAL_PERSISTENT_PROCESSES	允许程序发送一个永久的进程信号	普通权限
状态栏控制	STATUS_BAR	允许程序打开、关闭、禁用状态栏	普通权限
访问订阅内容	SUBSCRIBED_FEEDS_READ	允许应用程序访问已订阅的数据库	普通权限
写入订阅内容	SUBSCRIBED_FEEDS_WRITE	允许应用程序写入或修改订阅内容的数据库	普通权限
显示系统窗口	SYSTEM_ALERT_WINDOW	允许程序显示系统窗口	普通权限
使用设备的红外发射器	TRANSMIT_IR	允许使用设备的红外发射器, 如果可用	普通权限
删除快捷方式	UNINSTALL_SHORTCUT	删除快捷方式	普通权限
更新设备状态	UPDATE_DEVICE_STATS	允许程序更新设备状态	普通权限
使用证书	USE_CREDENTIALS	允许应用程序请求验证 AccountManager	普通权限

续表

名称	权限	描述	分类
使用 SIP 视频	USE_SIP	允许程序使用 SIP 视频服务	危险权限
使用振动	VIBRATE	允许程序振动	普通权限
唤醒锁定	WAKE_LOCK	允许程序在手机屏幕关闭后后台进程仍然运行	普通权限
写入 GPRS 接入点设置	WRITE_APN_SETTINGS	允许程序写入网络 GPRS 接入点设置	普通权限
写入日程提醒	WRITE_CALENDAR	允许程序写入日程, 但不可读取	危险权限
写入联系人数据	WRITE_CALL_LOG	允许程序写入联系人数据, 但不可读取	危险权限
写入联系人	WRITE_CONTACTS	允许程序写入联系人, 但不可读取	危险权限
写入外部存储	WRITE_EXTERNAL_STORAGE	允许程序写入外部存储	危险权限
写入 Google 地图数据	WRITE_GSERVICES	允许应用程序修改谷歌地图服务	普通权限
写入收藏夹和历史记录	WRITE_HISTORY_BOOKMARKS	允许应用程序写入浏览器历史记录或收藏夹, 但不可读取	普通权限
编写用户的个人资料信息	WRITE_PROFILE	允许应用程序编写用户的个人资料信息, 但不可读取	普通权限
读写系统敏感设置	WRITE_SECURE_SETTINGS	允许应用程序读取或写入安全系统设置	普通权限
读写系统设置	WRITE_SETTINGS	允许应用程序读取或写入系统设置	普通权限
编写短信	WRITE_SMS	允许应用程序编写短信	普通权限
写入在线同步设置	WRITE_SYNC_SETTINGS	允许应用程序编写 Google 在线同步设置	普通权限
编写用户字典	WRITE_USER_DICTIONARY	允许应用程序编写用户字典	普通权限