https://doi.org/10.1051/jnwpu/20203850965

基于 API 配对的 Android 恶意应用检测

管峻[1], 刘慧英[1], 毛保磊[1,2], 蒋煦[1]

(1.西北工业大学 自动化学院, 陕西 西安 710072; 2.郑州大学, 河南 郑州 450000)

摘 要:针对基于 Android 应用程序申请权限的检测过于粗粒度的问题,提出了基于敏感应用程序编程接口(application program interface,API)配对的恶意应用检测方法。通过反编译应用程序提取危险权限对应的敏感 API,将敏感 API 两两配对分别构建恶意应用无向图与良性应用无向图,再根据恶意应用和良性应用在敏感 API 调用上的差异分配相同边不同的权重,以此检测 Android 恶意应用。实验结果表明,提出的方法可以有效地检测出 Android 恶意应用程序,具有现实意义。

关 键 词:安卓系统;权限;应用程序编程接口;恶意应用

中图分类号:TP309 **文献标志码**:A **文章编号**:1000-2758(2020)05-0965-06

随着 5G 技术全面铺开,移动智能终端必将掀开崭新的一页。目前,移动智能终端操作系统主要是 Android 系统和 IOS 系统,其中 Android 系统因其开源,市场占有率一直处于领先的位置[1]。受到经济利益的驱使,针对 Android 终端用户的恶意攻击层出不穷,造成终端用户信息泄露,经济受损。据 2019 年发布的《中国网民权益保护调查报告》称[2],2018 年我国网民因为垃圾信息、诈骗信息等遭受的经济损失人均 124 元,总体经济损失约 805 亿元;78.2%的网民个人信息被泄露。

为了应对恶意应用的泛滥,检测技术不断升级。就检测方法而言,恶意应用检测技术主要分为静态检测分析技术、动态检测分析技术和基于机器学习的检测方法。静态检测分析技术是指应用程序不需要实际运行,只是将应用程序的源代码或者应用程序反编译后的代码作为研究对象,依据应用程序中存在的控制流、数据流和语义信息等检测应用程序是否存在恶意行为[3]。动态分析检测技术是指在虚拟机、沙盒等受控系统环境中运行待检测的应用程序,监控应用程序的行为[4]。2 种分析方法各有利弊,静态检测分析技术效率高,但无法应对代码混淆、加密、加壳等阻碍静态分析的手段。动态检测分析技术可以解决上述静态检测分析技术的短板,但代码执行的覆盖率低,恶意行为在沙箱运行过程中不一定会被触发执行。同时动态检测分析需要占用系统大量的资源,开销较大[5]。

随着应用程序数量指数级地增长,基于机器学习算法进行恶意应用检测的研究越来越多,通过静态、动态的方式提取机器学习算法的特征,输入分类器进行应用程序的分类。在现有的研究成果中,研究对象主要集中在权限和应用程序调用的 API。Arora[6]考虑到应用程序危险行为的发生需要获取危险权限,于是首次使用危险权限的配对去检测应用程序。但如果应用程序存在过度申请权限的问题,可能会导致检测的误判。文献[7]通过反编译应用程序提取应用程序实际使用的权限作为特征,采用分类算法进行恶意应用检测,与将 AndroidManifest.xml 中声明的权限作为特征进行检测相比,检测准确率提高。同时,应用程序的应用行为最终必然由 API 执行,所以将 API 作为研究对象相比于权限更加细粒度。文献[8]通过将 API 作为研究对象,总结不同恶意应用家族危险权限对应的 API,以此来实现应用程序的检测。文献[9]利用应用程序敏感 API 函数调用图的相似,对 1 326 个恶意应用进行检测,准确率达到 96.77%。文献[10]对恶意应用和良性应用在权限、API 的使用上进行了详细的分析与对比,利用随机森林算法选取危险权限对应的 50 个敏感 API 作为特征检测恶意应用。

收稿日期:2019-12-28 **基金项目**:河南省高等学校重点科研项目(21A520041)资助

作者简介:管峻(1983—),西北工业大学博士研究生,主要从事信息安全研究。

综上所述,本文为了避免粗粒度的权限特征造成误报,选取细粒度的敏感 API 作为研究对象,受文献[6]的启发,将恶意应用和良性应用调用的危险权限对应的敏感 API 两两配对,并分别根据它们在良性应用和恶意应用中的重要性分配不同的权重,建立检测模型。在检测应用程序时,提取危险权限对应的敏感 API 两两配对,再分别根据恶意应用无向图和良性应用无向图边的权重进行计算,通过对权重得分的比较,判断应用程序是否为恶意应用。

1 基础知识

Android 系统采用分层思想,架构清晰、层次分明、各层之间的依赖性小。Android 应用程序是 apk 格式的文件,使用 apktool 工具可以对应用程序进行反编译,其中 AndroidManifest.xml 是全局配置文件,包含申请的权限、组件信息,smali 代码等。

为了保障系统的安全和用户的信息安全,Android 系统主要设置了沙箱机制、权限管理机制、签名机制、系统安全机制,其中与本文研究内容相关的是权限管理机制。默认情况下,Android 应用程序只能访问有限的系统功能,如需要访问受保护的系统资源就必须在 AndroidManifest.xml 全局配置文件中声明需要使用的权限信息。Google 将权限分为正常权限和危险权限。如果应用程序在配置文件中申请危险权限,系统需要得到 Android 终端用户的确认才能授予;正常权限则不同,无须用户确认会被自动授予。

2 基于 API 配对的检测方法

本文实验方法如图 1 所示,首先从恶意应用家族应用程序中提取危险权限对应的敏感 API,将它们两两配对,并根据它们在恶意应用和良性应用中出现频率分配不同的权重,构建恶意应用敏感 API 无向图。

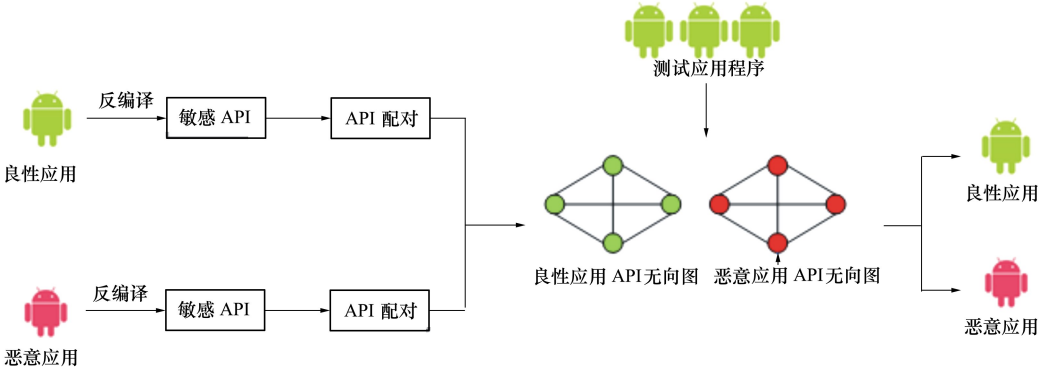

图 1 本文检测方法流程

从检测效率角度出发,在构建良性应用无向图时,以恶意应用提取的敏感 API 为基准,提取良性应用调用的敏感 API 两两配对并分配权重。当检测未知应用程序时,先反编译应用程序提取调用的敏感 API,根据恶意应用、良性应用无向图边的权重分别计算应用程序恶意应用和良性应用的权重分数。当恶意应用权重分数大于良性应用权重分数时,应用程序被判定为恶意应用。

2.1 无向图的构建

图表由顶点和边组成,表示为 $G(\boldsymbol{V},\boldsymbol{E})$,其中,$G$ 表示图,$\boldsymbol{V}$ 是 G 中顶点的集合,$\boldsymbol{E}$ 是图 G 中边的集合,分为无向图和有向图,无向图是 2 个顶点之间可以互相抵达,而有向图只能从一个顶点到另一个顶点。

应用程序调用的 API 数目数以万计,为了能够精准地反映出恶意应用和良性应用的差异,本文以敏感 API 作为无向图的顶点,依次反编译数据库训练集中的恶意应用,提取危险权限对应的敏感 API,两两配对构建无向图。如果一个应用程序调用的敏感 API 数目为 N,则创建图 G 的 N 个顶点,每一个顶点对应于一个敏感 API,顶点之间两两连接,并设置存在的边初始值为 1。分析第二个应用程序时,如果存在同样的边则增加 1;如果原图中不存在这条边时,则将此边插入图 G,设置边的数值为 1,直至数据库训练集中所有样本处理完毕。

2.2　权重的分配

给图 G 每条边 $\boldsymbol{E}$ 分配权重前,恶意应用无向图和良性应用无向图需要进行归一化的预处理,然后再分别计算每条边的权重。权重分配的原则就是要体现出相同 API 顶点组成的边在恶意应用和良性应用无向图中不同的重要性,因为恶意应用和良性应用在 API 调用上还是有区别的,尤其是敏感 API。对于在恶意应用无向图中出现较多而良性应用无向图中出现较少的边分配较高的恶意应用无向图边的权重;对于在良性应用无向图中出现较多而恶意应用无向图中出现较少的边分配较高的良性应用无向图边的权重。

以边 e_i 为例,在恶意应用无向图中 e_{Mi} 的权重定义为 $\omega(e_{Mi})$,如(1)式所示,即边 e_i 在恶意应用无向图中的数量 $Q(e_{Mi})$ 占 e_i 2 个无向图中数量的比例;在良性应用无向图中 e_i 的权重定义为 $\omega(e_{Bi})$,如(2)式所示。

$$\omega(e_{Mi}) = \frac{Q(e_{Mi})}{Q(e_{Mi}) + Q(e_{Bi})} \tag{1}$$

$$\omega(e_{Bi}) = \frac{Q(e_{Bi})}{Q(e_{Mi}) + Q(e_{Bi})} \tag{2}$$

2.3　权重得分的比较

当待检测的应用程序被反编译提取危险权限对应的 API 特征后,进行两两配对,所有边 e_i 分别参照恶意应用无向图和良性应用无向图的权重分配,恶意应用无向图权重得分记为 f_M,如(3)式所示;良性应用无向图权重得分记为 f_B,如(4)式所示。对于同一应用程序而言,虽然敏感 API 构成的边都是相同的,但由于恶意应用无向图和良性应用无向图对于同一条边分配的权重不同,所以需要比较两者的权重得分。当恶意应用权重分数大于良性应用权重分数时,判定测试应用程序为恶意应用,反之则判定为良性应用。

$$f_M = \omega(e_{M1})e_1 + \omega(e_{M2})e_2 + \cdots + (e_{Mi})e_i \tag{3}$$

$$f_B = \omega(e_{B1})e_1 + \omega(e_{B2})e_2 + \cdots + (e_{Bi})e_i \tag{4}$$

(3)式和(4)式中

$$e_i = \begin{cases} 1 & \text{边存在} \\ 0 & \text{否则} \end{cases}$$

3　实验与分析

本文的实验运行在 16G 内存的 Windows8 系统,所有应用程序通过 apktool 工具进行反编译,应用程序敏感 API 特征的提取、恶意应用无向图和良性应用无向图的构建,应用程序的测试程序均使用 Python 语言编写。

实验使用的恶意应用主要来自于 MalGenome[11]、AndroMalShare[12] 和 VirusShare[13],为了验证建立的检测模型是否具备检测未知恶意应用的能力,本文选取了部分恶意应用家族应用程序作为训练集样本,如表 1 所示,总计 460 个。良性应用训练样本均从正规应用市场下载,涵盖游戏类、即时通讯类、运动类等共计 400 个。

表 1　建模使用的恶意应用家族

YZHC、Adware、DroidKungFu、SmsReg、Plankton、Generic、BaseBridge、KminTrojan、PrivacyRisk、FakeInst

1) API 配对模型的分析与检测

建立的恶意应用无向图和良性应用无向图分别含有 3 922 条边,其中恶意应用无向图排名前 10 的边如下:

Landroid/telephony/TelephonyManager;->getDeviceId Ljava/net/URL;->openConnection;

Landroid/net/ConnectivityManager;->getActiveNetworkInfo Landroid/telephony/TelephonyManager;->getDeviceId;

Landroid/telephony/TelephonyManager;->getDeviceId Ljava/net/HttpURLConnection;->connect);

(Landroid/net/ConnectivityManager;->getActiveNetworkInfo, Ljava/net/URL;->openConnection);

(Landroid/net/ConnectivityManager;->getActiveNetworkInfo, Ljava/net/HttpURLConnection;->connect);

(Landroid/net/ConnectivityManager;->getActiveNetworkInfo, Lorg/apache/http/impl/client/DefaultHttpClient;->execute);

(Landroid/telephony/TelephonyManager;->getDeviceId, Lorg/apache/http/impl/client/DefaultHttpClient;->execute);

(Landroid/telephony/TelephonyManager;->getDeviceId, Landroid/telephony/TelephonyManager;->getSubscriberId);

(Landroid/net/ConnectivityManager;->getActiveNetworkInfo, ´Landroid/net/wifi/WifiManager;->getConnectionInfo);

(Landroid/net/wifi/WifiManager;->getConnectionInfo, ´Landroid/telephony/TelephonyManager;->getDeviceId)。

良性应用无向图排名前10的边如下:

(Landroid/app/NotificationManager;->notify, Landroid/net/ConnectivityManager;->getActiveNetworkInfo);

(Landroid/net/ConnectivityManager;->getActiveNetworkInfo, Landroid/net/wifi/WifiManager;->getConnectionInfo);

(Landroid/net/ConnectivityManager;->getActiveNetworkInfo, Landroid/telephony/TelephonyManager;->getDeviceId);

(Landroid/net/ConnectivityManager;->getActiveNetworkInfo, Ljava/net/HttpURLConnection;->connect);

(Landroid/net/ConnectivityManager;->getActiveNetworkInfo, Ljava/net/URL;->openConnection);

(Landroid/net/wifi/WifiManager;->getConnectionInfo, Ljava/net/HttpURLConnection;->connect);

(Landroid/net/wifi/WifiManager;->getConnectionInfo, Ljava/net/URL;->openConnection);

(Landroid/telephony/TelephonyManager;->getDeviceId, Ljava/net/HttpURLConnection;->connect);

(Landroid/telephony/TelephonyManager;->getDeviceId, Ljava/net/URL;->openConnection);

(Landroid/app/ActivityManager;->getRunningTasks′, Landroid/net/ConnectivityManager;->getActiveNetworkInfo)。

通过以上结果对比可以看出有5条边是相同的,说明无论良性应用还是恶意应用都频繁调用这10个API,而且这些API在同一应用程序中成对出现。这些API都属于Internet和READ_PHONE_STATE权限,与文献[6]对权限进行配对的结果相符。从表2、表3可以看出恶意应用和良性应用虽然在权限使用上差别并不是非常大,但存在行为差异,如恶意应用和良性应用都调用getDeviceId()获取IMEI,但恶意应用会频繁调用getSubscriberId()获取区分蜂窝网络中不同用户的国际移动用户识别码,该行为远高于良性应用。

使用构建好的无向图检测模型检测AndroMalShare和MalGenome中的1 032个恶意应用(包括没有参与建模的恶意应用家族)和正规应用市场下载的987个良性应用,所有测试应用程序均可以反编译并经过VirusTotal确认。经检测,本文提出的方法对恶意应用的检测准确率为95.44%,良性应用检测的准确率为92.19%。

2)检测未知恶意应用的能力

本文恶意应用无向图和良性应用无向图建立在表1恶意应用家族调用敏感API的基础上,为了测试模型检测未知恶意应用的能力,在测试集中加入了建模时未用的恶意应用家族,其中包含2013年收集的恶意应用家族和VirusShare在2016年收集的恶意应用家族。

检测结果如图2所示,可以看出除了敏感API较少的恶意应用家族(如FakePlayer、SndApps)无法检测外,对其余选择的恶意应用家族应用程序检测效果较好,即使它申请的危险权限较少。GoldDream中的恶意应用分为两部分:一部分是2013年前收集的,另一部分是近年收集的。而本文提出方法未检测出的该类应用程序属于后者,说明即使同类恶意应用家族的应用程序也在变异,且伪装性越来越强。Appcrack恶意应用家族是VirusShare在2016年收集的,分类的准确率相对低很多,主要还是因为它与良性应用的差异很小,需要通过扩大无向图的规模来提高检测准确率,但效率是必须面对的问题。

另外,为了检测提出方法的泛化能力,从2016年VirusShare收集的恶意应用中随机选取2 000个作为测试集。与第一部分的检测结果相比性能指标有所下滑,恶意应用检测的准确率为90.8%,误判的恶意应用主要集中在敏感API调用较少,通过动态加载技术执行程序的主要功能等,如SmsReg、Agent、DR.Gen等恶意应用家族。需要说明的是在误判的恶意应用中,存在应用程序上传VirusTotal后绝大多数杀毒工具均显示为良性应用的恶意应用,所以在实验中还需要对样本库的应用程序进一步地甄别,避免不必要的误判。另外,2016年收集的恶意应用相比于2013年左右收集的恶意应用伪装性更强,更接近于良性应用。

同时,在检测的过程中,发现根据权重得分可以对同类的恶意应用或者二次开发的恶意应用进行归类,比如DroidKungFu恶意应用家族中有多组权重得分相同的应用程序,通过进一步的分析,发现它们都是二次打包的应用程序。

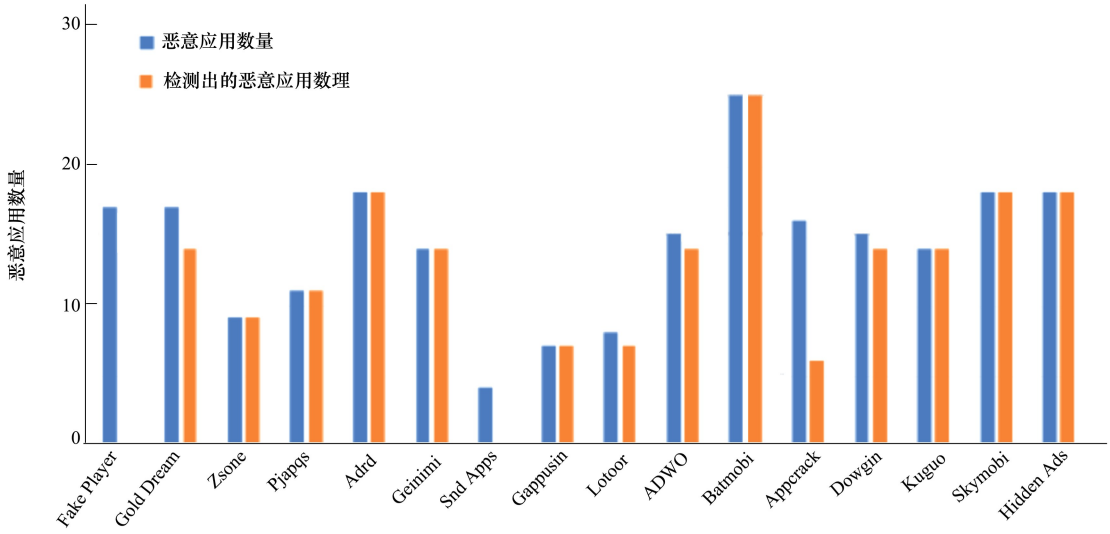

图 2　未知恶意应用家族检测结果

3）与同类检测工具的比较

本文提出的基于敏感 API 配对的检测方法较同思路基于权限的检测方法在特征的选择上更加细粒度,更能体现良性应用和恶意应用的差异,能够避免因为过度申请权限而导致的误判,但建模和分析应用程序的效率有待加强。与以 API 作为特征的分类算法相比,本文提出的方法主要是研究敏感 API 在恶意应用和良性应用中应用的差异,通过这种差异对待测应用程序的属性进行判断。

最后,从 2019 年的 VirusShare 收集的恶意应用库中随机选取 100 个应用程序输入 VirusTotal 工具中进行检测,与知名的杀毒软件对比,对比结果如图 3 所示。

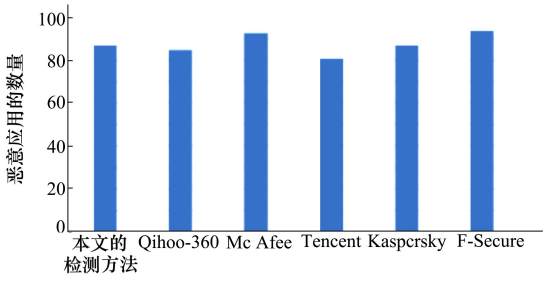

图 3　同类检测工具检测结果比较

本文检测结果略差于 McAfee、F-Secure,与 Qihoo-360 和 Kaspersky 接近,好于 Tencent。与 McAfee、F-Secure 的差距主要在本文检测方法误判的恶意应用检测上,这类恶意应用程序与良性应用程序差异性小,也是大多数杀毒工具(VirusTotal 提供)检测不出的。

4　结　论

本文通过提取危险权限对应的敏感 API 两两配对,建立恶意应用无向图和良性应用无向图,并根据它们在恶意应用和良性应用中的重要性分配不同的权重。测试集应用程序参考 2 个无向图边的权重分配,通过对比权重积分,判断是应用程序属性。

下一步研究的重点是考虑选取更多不同类型的特征,选择更多有差异性的恶意应用家族参与建模,构建更精准的、更完善的无向图。同时,需要对训练集和测试集的应用程序进行恶意属性的确认,以提升模型的检测能力。

参考文献:

[1] IDC. Smartphone Market Share[EB/OL]. (2019-01-24)[2019-11-12]. https://www.idc.com/promo/smartphone-market-share/os

[2] 中国互联网协会. 2019 年中国网民权益保护调查报告[EB/OL]. (2019-05-29)[2019-11-02]. https://max.book118.com/html/2019/0630/813513211700203-2.shtm

[3] KABAKUS A T. What Static Analysis Can Utmost Offer for Android Malware Detection[J]. Information Technology and Control, 2019, 48(2): 235-249

[4] HE Y, YANG X, HU B, et al. Dynamic Privacy Leakage Analysis of Android Third-Party Libraries[J]. Journal of Information Security and Applications, 2019, 46: 259-270

[5] FARUKI P, BHARMAL A, LAXMI V, et al. Android Security: A Survey of Issues, Malware Penetration, and Defenses[J]. IEEE Communications Surveys & Tutorials, 2017, 17(2): 998-1022

[6] ARORA A, PEDDOJU S K, CONTI M. PermPair: Android Malware Detection using Permission Pairs[J]. IEEE Trans on Information Forensics and Security, 2020, 15: 1968-1982

[7] LIANG S, DU X. Permission-Combination-Based Scheme for Android Mobile Malware Detection[C]// Proceedings of the 2014 International Conference on Communications, Sydney, 2014: 2301-2306

[8] MIRZAEI O, SUAREZ-TANGIL G, DE FUENTES J M, et al. Andrensemble: Leveraging Api Ensembles to Characterize Android Malware Families[C]// Proceedings of the 14th ACM Asia Conference on Computer and Communications Security, Aucland, 2019: 307-314

[9] ZHOU H, ZHANG W, WEI F, et al. Analysis of Android Malware Family Characteristic Based on Isomorphism of Sensitive API Call Graph[C]// Proceedings of the 2nd International Conference on Data Science in Cyberspace(DSC), Shenzhen, 2017: 319-327

[10] TAO G, ZHENG Z, GUO Z, et al. MalPat: Mining Patterns of Malicious and Benign Android Apps via Permission-Related APIs[J]. IEEE Trans on Reliability, 2018, 67(99): 355-369

Android Malware Detection Based on API Pairing

GUAN Jun[1], LIU Huiying[1], MAO Baolei[1,2], JIANG Xu[1]

(1.School of Automation, Northwestern Polytechnical University, Xi'an 710072, China;
2.Zhengzhou University, Zhengzhou 450000, China)

Abstract: Aiming at the problem that the permission-based detection is too coarse-grained, a malware detection method based on sensitive application program interface(API) pairing is proposed. The method decompiles the application to extract the sensitive APIs corresponding to the dangerous permissions, and uses the pairing of the sensitive APIs to construct the undirected graph of malicious applications and undirected graph of benign applications. According to the importance of sensitive APIs in malware and benign applications, different weights on the same edge in the different graphs are assigned to detect Android malicious applications. Experimental results show that the proposed method can effectively detect Android malicious applications and has practical significance.

Keywords: Android; permission; application program interface (API); malware detection