基于深度神经网络的 Android 恶意软件检测方法

超凡[1]，杨智[1]，杜学绘[1]，孙彦[2]
（1. 信息工程大学密码工程学院，河南 郑州 450001；
2. 中国电子技术标准化研究院，北京 100007）

摘　要：Android 系统正日益面临着恶意软件的攻击威胁。针对支持向量机等传统机器学习方法难以有效进行大样本多分类的恶意软件检测，提出一种基于深度神经网络的 Android 恶意软件检测与家族分类方法。该方法在全面提取应用组件、Intent Filter、权限、数据流等特征基础上，进行有效的特征选择以降低维度，基于深度神经网络进行面向恶意软件的大样本多分类检测。实验结果表明，该方法能够进行有效检测和分类，良性、恶意二分类精度为 97.73%，家族多分类精度可达到 93.54%，比其他机器学习算法有更好的分类效果。

关键词：安卓；恶意软件检测；静态分析；特征选择；深度神经网络
中图分类号：TP309.5
文献标识码：A
doi: 10.11959/j.issn.2096−109x.2020060

Android malware detection method based on deep neural network

CHAO Fan[1], YANG Zhi[1], DU Xuehui[1], SUN Yan[2]
1. College of Cryptogram Engineering, Information Engineering University, Zhengzhou 450001, China
2. China Electronics Standardization Institute, Beijing 100007, China

Abstract: Android is increasingly facing the threat of malware attacks. It is difficult to effectively detect large-sample and multi-class malware for traditional machine learning methods such as support vector machine, method for Android malware detection and family classification based on deep neural network was proposed. Based on the comprehensive extraction of application components, Intent Filter, permissions, and data flow, the method performed an effective feature selection to reduce dimensions, and conducted a large-sample detection and multi-class classification for malware based on deep neural network. The experimental results show that the method can conduct an effective detection and classification. The accuracy of binary classification between benign and malicious Apps is 97.73%, and the accuracy of family multi-class classification can reach 93.54%, which is higher

收稿日期：2019−11−26；**修回日期**：2020−02−03
通信作者：杨智，zynoah@163.com
基金项目：国家重点研发计划（2018YFB0803603）；国家自然科学基金（61972040, 61802436）
Foundation Items: The National Key R&D Program of China (2018YFB0803603), The National Natural Science Foundation of China (61972040, 61802436)

论文引用格式：超凡，杨智，杜学绘，等. 基于深度神经网络的 Android 恶意软件检测方法[J]. 网络与信息安全学报, 2020, 6(5): 67-79.
CHAO F, YANG Z, DU X H, et al. Android malware detection method based on deep neural network[J]. Chinese Journal of Network and Information Security, 2020, 6(5): 67-79.

than other machine learning algorithms.

Key words: Android, malware detection, static analysis, feature selection, deep neural network

1 引言

目前，Android 操作系统已经牢牢占据半数以上的市场，是最受欢迎的智能移动平台之一。然而，这样的成功背后潜藏着大量的安全隐患。Android 开放的生态环境为应用程序的编写提供了便利，但也使恶意软件可利用的漏洞增多。为了牟取金钱利益，恶意软件开发人员热衷于针对 Android 应用程序展开恶意活动，窃取用户隐私、重新打包良性应用等行为层出不穷。

恶意软件数量的快速增长促进了研究人员对安全工具的开发。虽然 Android 框架本身拥有沙箱隔离、权限机制等安全设计，但这不足以抵抗恶意软件越来越隐蔽多变的技术手段。在 Android 安全领域，已经有许多工作致力于检测恶意软件，主要分为动态分析和静态分析两类。动态分析在 Android 应用程序运行时跟踪其行为，而静态分析对应用的 APK 文件进行分析。随着人工智能技术的发展，基于机器学习的恶意软件检测方法逐渐得到探索，支持向量机（SVM，support vector machine）是其中的代表模型，代表工作有文献[1-3]。与传统机器学习算法相比，深度学习能够更充分地挖掘特征之间的深层联系，是近年来非常具有发展潜力的一个研究方向。

深度学习使用的特征可以通过动态或静态方式提取。文献[4-6]是使用动态特征的深度学习检测系统。Dahl 等[4]首次将神经网络应用于恶意软件检测，从文件的动态分析中提取低级特征，通过随机映射降维，并使用深度神经网络（DNN，deep neural network）进行分类，他们的工作能够预测未知文件是恶意还是良性，也可以识别恶意软件所属类别。MtNet[5]是一个与之类似的多任务深度学习恶意软件分类架构。Tobiyama 等[6]提出基于进程行为的恶意软件进程检测，将一组 API 调用序列输入长短期记忆网络进行进一步的特征精炼，然后通过卷积神经网络（CNN，convolutional neural network）进行分类。

使用静态特征的系统从 APK 文件中挖掘不同的分析关键。例如，文献[7-10]分别提取函数调用图、API 调用序列、原始操作码序列、原始操作码序列与指令功能序列作为特征，使用的分类器均为 CNN。文献[11]通过自动提取请求的权限和敏感 API，训练基于深度置信网络（DBN，deep belief network）的学习模型。文献[12]在 DBN 的基础上添加自动编码器网络，能够有效地对特征数据集进行降维。文献[13]提取 API 调用、服务及敏感权限作为特征，提出了一种基于深度森林的恶意软件行为检测机制，采用 ReliefF 算法进行降维，并将样本分为良性应用及 3 类不同行为的恶意软件。

还有一些工作从静态与动态两方面提取混合特征。Droid-Sec[14]是一个基于半监督 DBN 的 Android 恶意软件检测系统，特征包括权限、敏感 API 及动态行为。同样采用 DBN 的混合分析方法还有 DroidDetector[15]和 DeepDroid[16]。HADM[17]为每个特征向量集训练 DNN，将 DNN 学习的特征与原始特征相结合，然后使用多核学习进行分类。SDADLDroid[18]收集 APK 文件的混合特征，经主成分分析数据降维后输入多层降噪自动编码机进行学习。

本文提出一种基于深度神经网络的 Android 恶意软件检测方法，可以判断应用程序是良性还是恶意，并能识别恶意软件所在的家族。针对以往工作中提取的特征不够全面，本文对应用程序的 APK 文件进行静态分析，从配置文件和反编译代码中收集应用组件、Intent Filter、权限、数据流等多方面特征。由数据集样本的所有特征组成特征库，并为每个样本生成各自的特征向量。经观察，这样的特征向量中存在大量为零的维度，且有很多对于分类来说无意义的特征，应当对特征空间予以简化。因此，本文基于遗传算法[19]设计了一个特征选择方案。

本文预期实现恶意软件检测与家族分类的双重目标，不仅能够将良性应用与恶意软件区分开来，而且可以准确地将恶意软件定位到其所属家族。家族是指具有共同起源特征的一类恶意软件，

家族内的恶意软件之间有着相似的行为方式。因此，本文设计的分类实验并非是传统恶意软件检测的二分类，而是将良性应用单独作为一个类别、恶意软件直接以家族为类别的多分类。要达到这样精细的分类，需要一个强有力的深度学习模型。实验表明，深度神经网络可以较好地完成这一任务。

2　特征空间的组成模式

本节探讨了哪些特征可以在 Android 恶意软件检测中发挥作用。通过观察良性应用与恶意软件的不同之处并分析差异产生的原因，选出了一系列有代表性的特征，它们均可以通过应用程序的配置文件和 DEX 代码整理获得。

2.1　应用组件

（1）四大组件

Android 应用程序由 4 种类型的组件构成，分别是 Activity、Service、Broadcast Receiver 和 Content Provider。Activity 提供用户操作的可视化界面，完成特定的用户交互功能。Service 进行背景任务，通常用于在后台处理耗时的操作或监控其他组件的运行状态。Broadcast Receiver 是一个全局监听器，接收特定广播消息并作出响应。Content Provider 的主要作用是跨程序共享数据，提供标准接口以支持多个应用存储和读取结构化数据。应用程序组件都需经过注册方能被系统识别。

（2）组件比重

事实上，恶意软件与良性应用的组件构成比重有所不同。普遍来说，恶意软件会比良性应用注册更高比例的 Broadcast Receiver，而 Activity、Service 和 Content Provider 等组件的比重相应降低。

由于隐蔽的恶意活动需要选择合适的时机，恶意软件希望能够在系统环境发生改变时得到相关提示，如网络连通性的改变、电池电量不足等。Broadcast Receiver 接收广播的功能决定了该类型组件在监视系统事件方面有着广泛的用途，在恶意软件中相当流行。

与此同时，作为一款应用程序，恶意软件仍然需要假意向终端用户提供一些实用功能，但这些功能非常有限，完全无法与良性应用所能提供的相提并论。这就意味着恶意软件注册的 Activity、Service 和 Content Provider 组件数目占比较小。

（3）组件名称

Android 开发鼓励代码的重复利用。这种情况不仅在良性应用中会出现，而且在恶意软件中相当常见，这也是区分两类应用程序的一个重要线索。

在良性应用中，同一开发团队设计的应用之间固然会有一定的相似性，但也会利用其他知名应用开放的各种组件，在自家产品中申请其他应用的自定义权限，这些权限通常来自华为、腾讯、阿里巴巴、谷歌、Facebook 等企业。

然而，代码的再利用问题是一把双刃剑。同一家族内的恶意软件大多具有类似的恶意行为，可能拥有相同的核心组件，共享实现攻击的代码。例如，在恶意软件家族 DroidKungFu1 中，尽管每个恶意软件具有不同的包名，但大多包含组件 com.google.ssearch.SearchService。而诸如 com. google. ads.AdActivity 等组件，甚至遍及数个恶意软件家族。另外，有些重复利用的组件在保留名称的基础上改变前缀。例如，在 RogueSPPush 家族中，组件 MoreExcting Activity 非常流行，某一样本中含有名为 com. talkweb.comm.MoreExctingActivity 的组件，而在另一样本中则变成了 com.heroit.tzuwei. lite. More Excting Activity。

恶意软件热衷于把自己掩饰成良性应用，以便更好地在 Android 市场上隐蔽。通常，应用程序的功能通过编写组件来实现，因此组件的命名需要清晰地表达用途。一般来说，组件名称采用几个单词的组合，如 ModifyPasswordActivity 是一个提供修改密码功能的 Activity 组件。笔者观察到，良性应用的组件名多为 3 个及以上单词的组合，恶意软件多为两个单词的组合，由此可见这两类应用程序在组件名的长度和复杂性方面存在一定的差异。而有时，恶意软件未必遵循这一编程惯例，故意将组件冠以具有一定迷惑性的名称，如 com.google.update.Update。普通用户可能会被这些名字欺骗，误将恶意软件当作知名公司提供的产品。

（4）特征提取

本文在考虑与应用程序组件相关的特征时，分别记录每个应用中各类型组件数占总数的比例及其无前缀组件名称。

2.2 Intent Filter

（1）ICC 机制

Android 操作系统拥有一个独特的通信模型，称为组件间通信（ICC，inter-component communication）机制，它提供特定的方法以便同一应用内甚至不同应用间的组件进行数据交换。然而，恶意软件很可能滥用该机制，以达到窃取用户隐私等目的，因此，可在特征库中加入可以反映 ICC 活动的特征。

ICC 机制使用被称为 Intent 的特殊异步消息，可在 Activity、Service、Broadcast Receiver 这 3 种组件间传递信息。Intent 通过属性对期望执行的动作进行抽象，主要分为显式和隐式两种。显式 Intent 直接指明目标组件名称，多用于在应用程序内部传递消息。隐式 Intent 的目标组件不是通过指定名称，而是通过设置 Intent 的其他属性进行筛选，因而在跨应用消息传递方面有着广泛应用。想要接收隐式 Intent 的组件，必须在 AndroidManifest.xml 文件中声明相应的 Intent Filter。然后，系统进行 Intent 解析，匹配出合适的组件。

（2）Intent Filter 的用途

Intent Filter 在 Android 应用程序中应用广泛。无论是在良性应用还是恶意软件中，有很多组件注册 Intent Filter，但这两种应用程序对其的使用方式大不相同。良性应用主要用于正常的组件间信息传递和协调工作，而恶意软件则可以劫持 Intent 趁机发动攻击。通过精心设计 Intent Filter 来截获未受到权限保护的隐式 Intent，恶意软件可以通过 Activity 劫持或 Service 劫持获取用户数据信息、故意发送错误结果对原程序造成破坏，也可以窃听、篡改、丢弃广播内容，或者根据被广播的系统范围事件进行其他恶意活动。

（3）Intent Filter 的属性

Intent Filter 具有多个属性，最重要的当属 action 和 category，因为它们与 Intent 要执行的动作息息相关，可以确切反映出它的意图。属性 action 是表现 Intent 要执行的动作名称的字符串，Android 定义了一系列的标准动作，应用可以根据需要自定义动作。为了防止应用程序之间互相影响，一般自定义 action 的命名方式是在动作名称前加上包名作为前缀，如 com.intsig.camscanner.Launcher。属性 category 是包含 Intent 额外信息的字符串，用来表现动作的类别。这两种属性通常一起使用。同样地，category 也有系统定义和自定义两种。

（4）特征提取

manifest 文件中声明了 Intent Filter 的组件，本文将其组件名称与对应 Intent Filter 的两大属性信息作为一个整体记录在特征库中。单纯的组件命名可能有偶然性，但加上 Intent Filter 能在很大程度上反映出该组件的用途及其关注点。

这一组合加强了组件名称这一类特征的针对性，但本文同样关注 Intent Filter 本身所包含的意义，通过记录 Intent Filter 中声明的 action 和 category 属性字符串，本文可以得知组件感兴趣的事件，从而推断出该应用程序是否具有恶意。

2.3 权限

（1）权限机制

权限机制是 Android 平台中一种很重要的安全机制。权限主要通过限制应用程序对系统资源的访问以及组件之间的数据共享，达到保护用户隐私和平台安全性的目的。它是对进程沙箱隔离机制的一种有力补充，增强了单独进程空间中应用获得系统信息和与其他应用通信的能力。

（2）权限范围

默认情况下，Android 应用不具有任何权限，必须在 AndroidManifest.xml 文件中进行申请。Android 框架提供了一套系统权限，开发人员也可以根据需要自定义权限。根据 protectionLevel 属性值的不同，Android 权限可划分为普通、危险、签名、系统 4 个保护级别。随着 Android 版本不断提升，可选择的系统权限也会有所变化。在本文的数据集中，应用程序使用的 API 版本不同，加上大量自定义权限的存在，导致需要考虑的权限范围很广泛。

（3）Android 版本对权限机制的影响

从 Android 6.0 版本开始，Google 对权限模型做了一次大调整。在以往的权限模型中，应用在配置文件中申请所需的权限，并在安装时向用户呈现，用户只能选择全部接受或放弃安装。权限一旦授予就无法更改，用户无法掌握权限的具体用途。新权限模型采用动态授权，应用程序申请危险权限后，在运行时向用户请求授权，且用户可以随时在设置中修改这一授权。一旦某个权限被撤销，应用在使用时必须重新请求用户的同意。此外，新增了权限组的概念，用户对危险权限的授予以组为单位进行。

然而，程序在运行时具体遵循何种权限机制，不仅与应用的 SDK 版本有关，也与用户设备安装的 Android 系统版本有关，其中存在二者的兼容问题。此外，在新权限模型中，用户在实际应用时具体会触发哪些权限是未知的，而这些权限是否会得到用户授权也是一个问题。考虑到这些情况，本文采用一种较为保守的方式，忽略数据集中各种应用程序的不同 SDK 版本，统一以静态授权的方式看待所有权限，即不考虑同组权限无须明确申请的可能，也不考虑部分权限可能无法得到用户授权的可能。

（4）权限使用

如果应用程序实际使用了没有申请的权限，在调用权限的相应功能时，会抛出 Security Exception 异常。然而，应用程序在 manifest 文件中申请的权限与代码中以 API 形式体现出的权限使用点并非完全一致。有许多原因可能会造成这种情况。

① 并非所有权限的目标都是保护系统 API 接口，也可能用于数据库操作和消息传递等其他方面。Content Provider 存储用户数据，有时需要权限来限制其他应用对其进行操作。例如，为了对存放联系人数据的 Content Provider 执行读取查询，应用必须拥有 READ_CONTACTS 权限。权限也可以限制系统 Intent 消息的接收，只有拥有适当权限的应用才可以得到相关通知。

应用程序可能会申请其他应用定义的权限。Android 系统允许应用程序自定义权限，一般用于保证跨应用组件通信与共享数据的安全性。除了同一企业内不同应用通过自定义权限共享信息的需要，也可以申请其他开发者定义的权限。应用程序可以公开某些组件，便于其他应用调用，从而实现应用之间的跳转。

② 本文选择的实验工具可能不够完善，无法检测到所有需要权限的 API 函数。

③ 对于目标 SDK 版本大于或等于 23 的应用程序来说，新权限模型导致同组权限无须全部申请。

④ 程序开发人员经验不足造成的错误。Android 文档可以提供的权限信息内容比较有限，甚至存在错误和遗漏之处，使开发人员对于需要申请哪些权限把握不当。

⑤ 部分权限无法发挥作用。例如，应用程序申请签名级别的权限，却不具有相应的私钥签名，Android 系统会自动判定该权限无效。

（5）特征提取

权限能够在较大程度上反映出应用程序所需的资源，是一种很重要的行为指示。Android 应用的权限使用情况复杂，本文在提取特征时简化了对其的考虑。根据应用程序在配置文件中申请的权限，以及工具在代码中找到的受权限保护的 API 使用点，将权限特征分成 3 个部分：申请且以 API 形式使用的权限及其 API 使用点、申请且未以 API 形式使用的权限、未申请且以 API 形式使用的权限及其 API 使用点。除了权限以外，本文还记录受权限保护的 API 使用点，虽然要达到同一目的有很多 API 可供选择，但同一家族恶意软件针对某些权限的常用 API 是有偏向性的，且良性应用与恶意软件对 API 的选择上有一定的差异，因此记录使用权限的 API 及其出现次数很有必要。

2.4　数据流

（1）污点分析

隐私泄露是 Android 应用程序面临的主要安全问题之一，是指在用户不知情的情况下将个人信息传出设备的行为。在现有的 Android 机制下，智能设备中存储了大量的用户隐私数据，而用户对应用具体如何使用这些数据缺乏清晰的认识，这使恶意软件有了可乘之机。

污点分析是检测 Android 隐私泄露问题的常

用手段，通过跟踪隐私数据从源头到流出点的流动过程，指出应用对隐私数据的使用方式，从而判断是否存在隐私泄露。数据源即数据进入程序的入口，被称为 source，是读取资源的 API 调用接口；数据流出点即数据离开程序的出口，被称为 sink，是写入资源的 API 调用接口。数据流就是从 source 到 sink 的 API 路径。

（2）分析工具

本文使用 FlowDroid[20]来识别 Android 应用程序中的数据流。FlowDroid 提供了一个高度精确的静态污点分析，追踪隐私数据在控制流图中的传递，具有完全的对象敏感、流敏感、上下文敏感性。FlowDroid 默认配置的是 SUSI[21]项目提供的 source/sink API 列表。

（3）特征提取

虽然隐私泄露的情况可以反映应用程序的恶意性，但并非所有恶意软件都会窃取隐私，而且在良性应用中存在着使用敏感数据的合理诉求。即 FlowDroid 显示的数据流结果未必是隐私泄露的表现，但在某种程度上体现了应用对数据的获取与使用方式。因此，本文不是通过确定特定数据流是否具有恶意，而是从整体数据流情况上衡量良性应用与恶意软件之间的行为差异。

本文认为 FlowDroid 输出的 source-sink 对代表了敏感数据流，因此将应用程序输入 FlowDroid 进行分析。从相同源获取的数据可以有不同的流出点，而不同源的数据也可以流向同一个出口，但只要处于不同的代码位置上就会被忠实地记录，因此得到的 source-sink 对的数量相当可观。

虽然 SUSI 从语义上提供了 source、sink 的进一步分类，但本文没有采用这种形式，而是以完整的方法签名来表达数据流。前者固然能够更好地说明敏感数据的使用方式，但通常存在多种接口函数可以达到同样的效果。一般来说，良性应用会选择较为普及的 Android API 方法来访问和使用敏感信息，而恶意软件开发者出于躲避安全分析工具的心理会选择一些知名度较低的 API 函数。因此，良性应用和恶意软件在 FlowDroid 中输出的 source、sink 以及 source-sink 对结果都会有所不同。

2.5 特征空间

为了使本文的分析通用可扩展，根据前文对良性应用和恶意软件的观察，笔者将特征空间进行相应划分，并将所有特征表示为字符串的形式，而应用的特征向量在各自维度上表现为对应特征的数量。从本文数据集样本中提取的所有特征组成了最终的特征库。综上所述，本文的特征空间组成如表 1 所示。

表 1 特征空间组成

Table 1 Composition of feature space

特征类别	特征格式
应用组件	Activity 组件数占应用中总组件数的比重
	Activity 组件名（无前缀、含 Intent Filter 信息）
	Service 组件数占应用中总组件数的比重
	Service 组件名（无前缀、含 Intent Filter 信息）
	Broadcast Receiver 组件数占应用中总组件数的比重
	Broadcast Receiver 组件名（无前缀、含 Intent Filter 信息）
	Content Provider 组件数占应用中总组件数的比重
	Content Provider 组件名（无前缀）
Intent Filter	Activity 组件中声明的 action 属性（系统定义、自定义）
	Activity 组件中声明的 category 属性（系统定义、自定义）
	Service 组件中声明的 action 属性（系统定义、自定义）
	Service 组件中声明的 category 属性（系统定义、自定义）
	Broadcast Receiver 组件中声明的 action 属性（系统定义、自定义）
	Broadcast Receiver 组件中声明的 category 属性（系统定义、自定义）
权限	申请且以 API 形式使用的权限及其 API 使用点
	申请且未以 API 形式使用的权限（系统定义、自定义）
	未申请且以 API 形式使用的权限及其 API 使用点
数据流	FlowDroid 输出的 source-sink 对（方法签名形式）

3 系统设计

为了进行面向 Android 平台的恶意软件检测，需要将应用程序的大量活动从不同角度抽象成特征，并逐步处理成学习模型可接受的表达，

本文的整个系统设计如图 1 所示，主要分成 5 个步骤。

1) 特征提取：对数据集中的 Android 应用程序进行静态分析，提取相应的特征，所有样本特征组成特征库。

2) 特征向量生成：将特征库转化为一个特征空间，并分别将每个应用的特征集在其中进行映射，生成应用的特征向量。

3) 特征选择：初步提取的特征总数较大，而且其中有许多无用特征，在此情况下进行机器学习的效率较低，因此采用遗传算法对特征空间进行压缩。

4) 深度神经网络学习：在得到最终的特征向量后，可以采用有效的深度学习技术进行分类。本文选择深度神经网络模型。

5) 分类：经过深度神经网络学习，本文预期的分类结果既可以将恶意软件与良性应用区分开来，又可以得到有关恶意软件所属家族的信息。因此，本文采用了一种特殊的多分类方法，其中包含二分类信息。

3.1　特征提取

为了更好地从 Android 应用程序中获得所需信息，本文借助工具 Androguard[22]和 FlowDroid[20]对应用的 APK 文件进行静态分析，提取相关特征。

Androguard 基于 Python 开发，是一款常用的 Android 应用程序静态分析工具。Androguard 在其源码主目录下集成了 10 多种子工具，分析人员可以以命令行的方式按需获取应用程序的各项基本信息，并且生成一些相关的调用图。

FlowDroid 是一款用 Java 实现的、针对 Android 的开源静态污点跟踪工具。FlowDroid 通过准确地建模 Android 应用程序的生命周期及其与操作系统之间的交互，从源头开始追踪敏感数

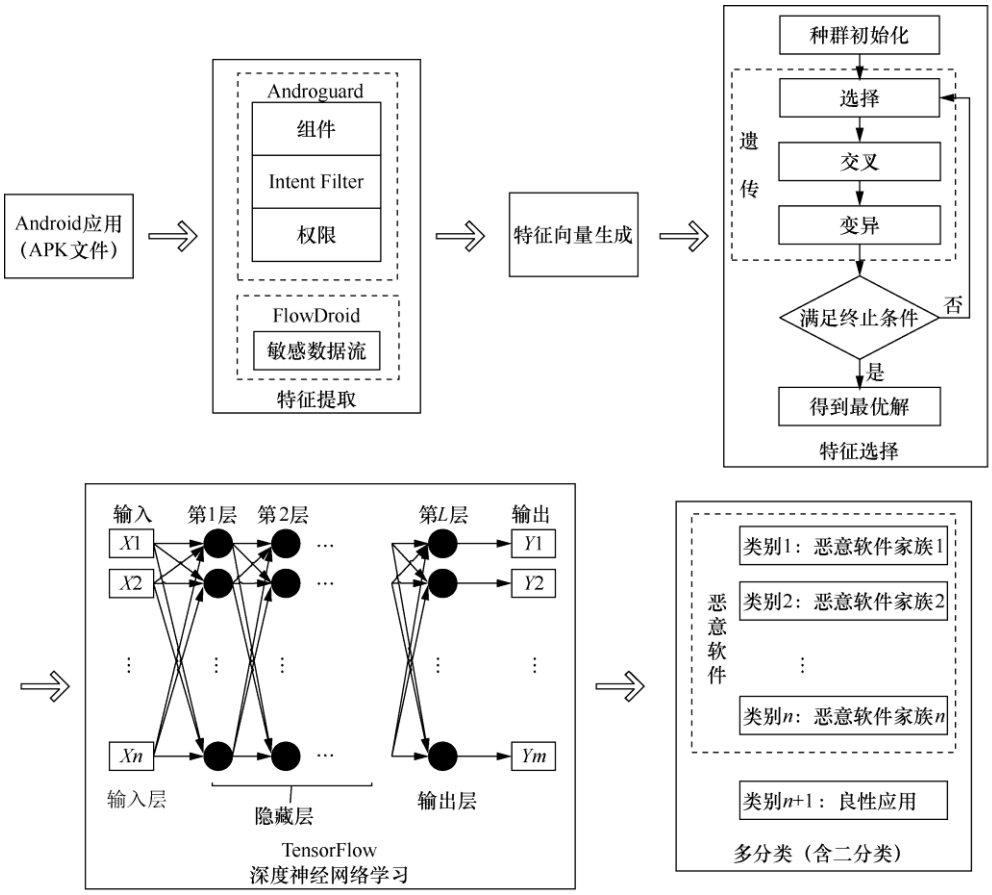

图 1　系统设计
Figure 1　System design

据在组件中流动的过程。

图 2 概述了本文如何利用这两大静态分析工具提取所需特征。

3.2 特征向量生成

在应用程序特征提取完毕后，本文将数据集中所有样本的特征汇总成一个特征库，其中的每个特征都以字符串的形式来表达。大多数的机器学习方法适用于数值型向量，因此需要将提取的特征字符串嵌入向量空间，只记录特征的数值。对于每个应用，特征向量的维度都与特征库的规模相同，是一一对应的关系。根据特征库，每个应用将自身拥有的特征映射成特征向量相应位置上的数字，不具备的特征对应 0。

3.3 特征选择

根据前面所述的特征提取方式，本文从 1 340 个良性应用和 1 260 个恶意软件中初步提取了 24 162 个特征。平均而言，每个应用程序拥有的特征不过数十个，特征向量中存在大量值为 0 的维度。在原始特征数量下进行机器学习花费的时间较长，且容易产生过拟合现象，导致模型泛化性能较差。因此，本文拟采用遗传算法[19]对特征空间进行压缩，从中过滤掉无用特征，筛选出表达能力更强的 200 个特征。

遗传算法是一种以遗传学理论为基础、模拟生物进化过程的搜索优化计算模型。标准的遗传算法遵循“生成+测试”的迭代过程，从初始解开始进行择优繁殖，逐步筛选出一定条件下的最优解。下面是基于遗传算法对特征选择问题的解决方案。

1) 问题建模：根据遗传算法的模型，对问题进行结构化表示。在遗传算法中，问题的可行解被称为染色体，它是基因的集合。具体来说，基因指的是特征库中的特征，染色体就是特征的组成方案。因此，每个特征选择方案对应的染色体都带有 24 162 个基因。本文采用二进制编码，用 0、1 来表达解中是否存在某个特征。

2) 种群初始化：种群是现有染色体组成的集合。在算法运行之初，通过随机生成一组可行解，形成初始种群。此处，对可行解的定义是在 24 162 个基因中只有 200 位被标为 1 的染色体。种群规模越大，越有可能找到全局最优解，但所需的运行时间相对较长。本文设定的种群规模为 30。

3) 个体评价：适应度函数是个体对环境适应程度的评价标准，体现了在当前背景下个体的优

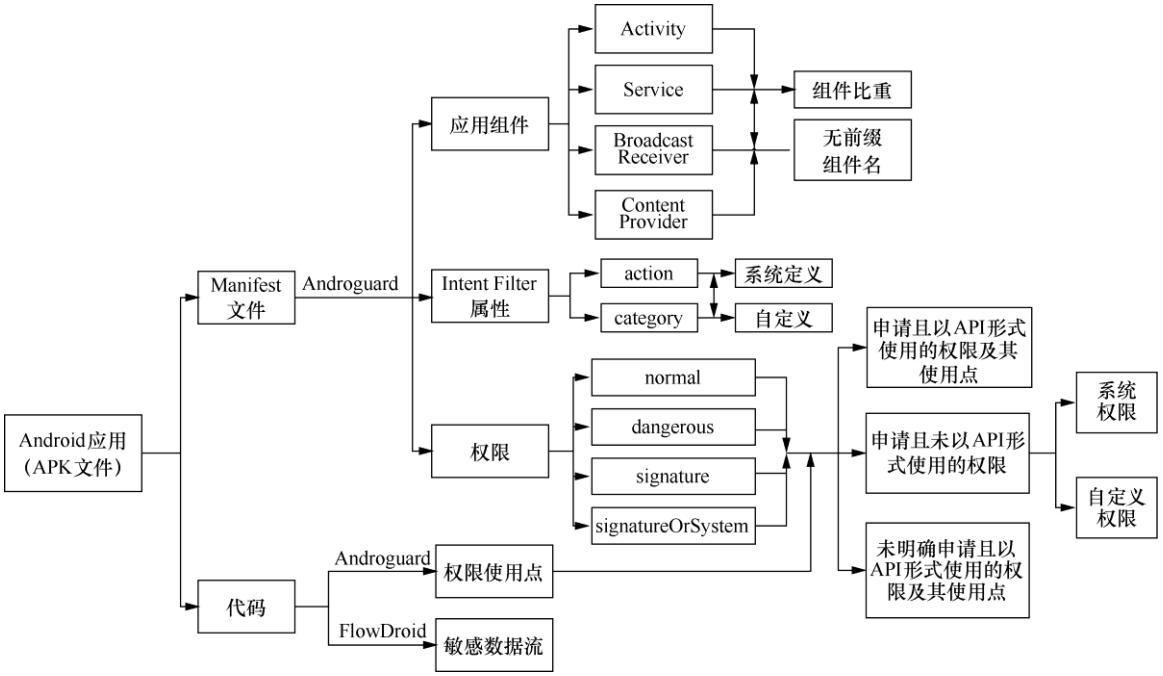

图 2 特征提取

Figure 2 Feature extraction

劣情况。适应度的取值范围为[0,1]，得分越高，个体的适应程度越好。在特征选择问题中，本文将适应度定义为当前特征选择方案下的机器学习精度。

4) 遗传操作：遗传操作模拟基因进化，是实现优胜劣汰的有力手段。在初始种群形成后，通过将优秀个体的基因进行结合重组，以期产生具有更强适应性的后代。遗传操作包括选择、交叉和变异 3 个基本的遗传算子。

① 选择运算：选择操作确定每次迭代时种群中哪些个体可以成为父代。根据生物进化规律，适应度更高的个体理应得到更多的机会繁殖后代。当然，这不意味着适应度低的个体完全不可能遗传基因。因此，父代选择应当是一种基于适应度但具有一定概率性的过程。

轮盘赌选择是一种常用的选择算子。它将当前种群中所有个体适应度的总和对应到一个轮盘的圆周，每个个体按其适应度在适应度总和中的比例占据轮盘的相应扇区。每次选择父代时，只需转动轮盘，即可得到一个候选者。每轮进化都将种群进行一次彻底更新，因此进行 30 次选择。父代之间随机配对，经过繁殖后得到相同数量的子代。

② 交叉运算：交叉是将父母的染色体按照一定的方式进行结合，从而生成新的个体。由于本文问题的可行解需要满足特征数为 200，因此交叉算法在均匀交叉的基础上进行一点改变。将父母染色体按其适应度之比等比例分配 200 个值为 1 的基因位，根据分配到的个数，从自身 200 个非零基因中进行等概率选取，遗传给子代。本文设定的交叉概率为 0.6。

③ 变异运算：变异以很小的概率随机改变染色体上某些基因值，能够向已有染色体中引入新的遗传结构。本文采用均匀变异算子，每个基因的值都以相同的概率发生随机变化，但同样要保证染色体中非零基因只有 200 个。通常，变异率是一个比较小的值，以保证优秀的解不会发生过多的变形。本文设定的变异率为 0.001。

5) 替换策略：经过一轮遗传操作，所有新生成的子代会组成当代种群，替代原有个体。进化的过程不是一帆风顺的，相邻子代之间未必会得到更好的种群，但是从全局来看，种群必定会向着更加适应环境的方向发展。

6) 终止条件：在理想情况下，进化次数应该是无穷的，但实际应用时往往需要人为限制一个终止条件，以平衡算法效果和执行效率之间的关系。本文设置算法在进化 100 代后终止，这足以得到一个最优可行解。

3.4　深度神经网络学习

由特征选择的最优解得到最终特征空间后，还需要对数据集中样本的原始特征向量进行相应的调整，然后进入深度学习的过程。

（1）监督学习

监督学习是一种常用的机器学习范式，利用已标记的数据集，通过某种学习方法建立一个模型，实现对未标记新数据的分类。监督学习主要经历训练和测试两个阶段，分别输入训练样本和测试样本。训练样本需要给出类别标签来指示模型的生成，而测试样本对模型来说类别未知，以便测试训练效果。在训练阶段，根据学习算法对训练样本的特征向量进行训练，据此生成一个模型。然后进入测试阶段，输入测试样本的特征向量，模型会对其进行分类，可预测未知样本的类别，也可以据此与样本的真实类别进行比较，评估模型的学习效果。

（2）深度神经网络

本文选择的算法模型是 DNN。DNN 是深度学习中经典的全连接神经网络架构，适用于处理复杂数据情况下的建模问题。它的模型结构层次较深，追求学习效果，忽视结果的可解释性。以 SVM 为代表的传统机器学习算法结构较为简单，通常只能达到二分类的效果，无法应对大样本情况下的多分类问题。在基于深度学习的恶意软件检测系统中，CNN、DBN 是两种应用较多的网络。CNN 的卷积网络使其适于处理具有空间结构或连续性的数据，如文献[7,10]为 CNN 提取了函数调用图、指令序列等连续特征，但对于多个离散特征集来说并不合适。与 DNN 纯粹的监督训练相比，DBN 引入一个无监督的预训练过程来进行权重偏置的初始化，该过程对于无标签的小样本数据来说较有意义，对当前数据集来说收益不大。

相比之下，本文方法通过对恶意软件和良性应用的丰富观察完成较为全面的特征工程，且在进入学习前已经通过特征选择进行降维，深度学习只需完成分类任务即可，在全连接网络中的计算速度也不慢。因此，DNN 更适合本文的整个系统架构。

（3）算法实现

本文利用流行的 TensorFlow 框架[23]实现深度神经网络学习。TensorFlow 由 Google 人工智能团队 Google Brain 开发维护，是一个基于数据流编程的符号数学系统，可进行各类机器学习和深度学习算法的编程实现，被誉为人工智能的开源神器。TensorFlow 框架提供了很多实用的开发接口，支持 GPU 计算，便于高效搭建出网络架构。

3.5 分类

本文预期达到两个分类效果，区分恶意软件和良性应用，并了解恶意软件所在的家族。因此，对数据集样本性质的标注并非采用传统的良性、恶意两个类别，除了所有良性应用作为单独的一个类别，恶意软件并非作为一个整体，而是按照其所属家族进行归类。以这种方式，本文把良性应用与恶意软件家族放在一个类别层面上进行考虑。这样的多分类方法将二分类包含在内，既能够比纯粹的二分类得到更多有关恶意软件的信息，又省去了“先二分类，再多分类”的多次学习方式的不便。

本文也曾尝试为良性应用采用一种分类标准。然而，目前各大 Android 应用市场的分类尚未统一、具有较强的主观性，且同一类别下的应用程序功能分散，相似点不够明显，远不如恶意软件家族内的联系紧密。显然，市场分类不足以从技术层面上区分良性应用，不具备借鉴意义。因此，本文没有对良性应用进行分类，而是将其作为一个整体来考虑，希望可以在将来的工作中通过聚类等手段为其构建新的分类。

4 评估

本节介绍实验实施的细节，包括数据集的来源、特征选择前后特征数量的分布、二分类与多分类实验的评估结果，以及全过程的运行时间开销。

4.1 数据集

本文的实验数据集分为良性样本和恶意样本两部分。恶意软件来自 Android Malware Genome Project[24]，能够评估 Android 恶意软件检测方法的有效性。同时，在 2019 年 8—12 月期间，笔者在华为应用市场上下载了各个类别中排名靠前的 Android 应用程序。为确保良性数据集的真实性和可靠性，本文将其发送到 VirusTotal[25]服务进行检查。VirusTotal 是一个免费提供可疑文件分析服务的网站，使用 70 多种反病毒扫描引擎对用户上传的文件进行检测，给出各个工具的安全分析结果。在未超时的情况下，当且仅当所有的反病毒扫描程序均显示为未发现警告时，才将应用程序标注为良性。在使用 Androguard 和 FlowDroid 进行分析的过程中，部分应用程序由于文件损坏或无法被完整地提取特征而排除在考虑范围之外。最终，本文得到来自 49 个家族的 1 260 个恶意软件，以及来自华为市场 17 个类别的 1 340 个良性应用，共计 2 600 个实验样本。

4.2 特征分布

在提取所有样本特征后，组成了原始的特征空间。这一特征空间过大，导致分类效率较低，因此，基于遗传算法对已有特征进行选择，构成新的特征空间。表 2 给出了特征选择前后的特征分布统计情况。

表 2 特征分布

Table 2 Feature distribution

特征类别	原始特征数量	新特征数量
应用组件	17 879	72
Intent Filter	2 146	21
权限	2 845	30
数据流	1 292	77
总和	24 162	200

本文最终得到的 200 个特征涉及各个特征类别，由此可见，本文的特征提取方案是有效的，但其中有许多特征对于分类来说无足轻重，或特征之间存在关联性导致其包含冗余信息。此外，由表 2 可以看出，遗传算法筛选后的特征分布与原始特征分布并不一致。

与应用组件相关的特征分为组件名称和组件占比这两部分。在原始特征集中，由于良性应用

的组件数量较多，导致组件名特征占据了很大一部分空间。经过特征选择后，保留的多为恶意软件家族中出现频率较高的组件名称，且通常带有 Intent Filter。良性应用组件名特征的缩减幅度最大，因为其中有大量组件只出现过一次，对于分类来说毫无意义。虽然组件比重在原始特征集中只有 4 个，但仍有 2 个保留了下来，分别是 Activity 和 Broadcast Receiver 的占比，充分表明记录该特征的必要性。

Intent Filter、权限类别的特征分布与原分布有一定的相似性。虽然 Intent Filter 和权限特征在新特征空间中的数量低于样本类别数，但这两类特征与组件名特征的不同之处在于，重复的组件名称主要限定在家族之内，而常用的 Intent Filter 和权限大致有一定的范围且多为系统定义，家族之间的区别在于特征组合方式不同。相同的是，这些类型的原始特征有很大一部分是由良性应用引入的，且自定义特征占据了相当的比重。

信息流特征主要涉及恶意软件的信息泄露行为，良性应用中虽有对敏感数据的使用，但目的不同导致两种应用程序偏好的敏感 API 及数据流路径不同。遗传算法保留的信息流特征兼顾了良性应用与恶意软件的不同情况。

由此可见，特征选择过滤掉的绝大部分特征与良性应用相关，因为良性应用普遍体量较大、特征较多，且大部分特征在其他样本中的出现次数极为有限，对于分类来说缺乏指示性。这充分说明了恶意软件与良性应用之间的一个巨大差别，即恶意软件以家族为单位呈现出较强的联系性，而良性应用的特征则较为分散。

4.3　实验评估

本文采用 10 折交叉验证[26]对实验结果进行评估，基于同一数据集将 DNN 与 SVM、CNN、DBN 方法的表现相比较。其中，SVM 是传统机器学习算法的代表，而 CNN 和 DBN 是近期相关工作中出现频率较高的深度学习网络。在保证数据分布较为一致的前提下，本文把良性和恶意数据集各自随机分成 10 个互不相交的子集。在每一轮实验中，将一个尚未挑选过的良性子集和一个恶意子集作为测试数据集，剩余子集作为训练数据集。如此依次进行 10 轮实验，每轮实验互不干扰，不仅可以保证每次实验使用的训练与测试样本不会重复，而且数据集中的每个应用程序都能得到一次测试分类的机会。

（1）实验配置

本文在一台搭载 6 核 1.10 GHz 的 Intel Core i7-10710U 处理器和 16 GB 内存的计算机上进行实验。基于当前数据集，深度学习的最佳网络结构需通过实验来决定，下面给出检测效果最好的参数配置。

① DNN 采用全连接网络，具有 2 个隐藏层，每个隐藏层有 120 个节点。

② DBN 由 2 层受限玻尔兹曼机和 1 层反向传播网络构成，其中隐藏层节点数均为 150。

③ CNN 由 1 个卷积层、1 个最大池化层、1 个全连接层和 Softmax 分类器组成，采用 128 个长度为 3 的卷积核。

此外，激活函数均为 ReLU，丢弃率为 0.5。

（2）实验结果

本文的多分类实验涉及 50 个类别，其中 49 个是恶意软件家族，1 个是良性应用类别。这一设计既能得到 Android 应用程序是否具有恶意的二分类结果，又可以进一步区分恶意软件家族之间的差异。3 个深度学习网络均可直接进行多分类，而 SVM 是二值分类器，无法一次性得出多分类结果，因此采用一对多法，每次训练将其中一个类别与其他类别相区分。

表 3 比较了不同机器学习算法对于应用程序性质判断的二分类结果，这些数据基于多分类实验结果，并规定恶意软件为正样本，良性应用为负样本。对于二分类任务，本文使用的度量标准是查准率、查全率、F1 值和精度。查准率是所有分类结果为正的样本中正确预测的比例，查全率是所有正样本中正确预测的比例，精度是所有样本中正确预测的比例。查准率和查全率是一对矛盾的度量，F1 值是二者的调和平均，给出这两个度量标准的一个综合性能。二分类结果表明，DNN 可以将数据集中 97.73%的应用程序正确分类为恶意软件或良性应用，分类效果优于其他 3 种机器学习算法。

表 3　二分类结果比较
Table 3　Comparison of binary classification results between different networks

算法	查准率	查全率	F1 值	精度
DNN	97.40%	97.94%	97.67%	97.73%
SVM	92.49%	93.89%	93.18%	93.35%
CNN	94.63%	95.16%	94.89%	95.04%
DBN	96.55%	97.86%	97.20%	97.27%

表 4 记录了不同机器学习算法的多分类比较结果。对于多分类任务，对应的度量标准分别是宏查准率、宏查全率、宏 F1 值和精度。宏查准率和宏查全率分别是各类别查准率和查全率的平均值，宏 F1 值是二者的调和平均。此外，表 4 给出了各算法的计算时间，其中 SVM 的耗时为 50 次二分类的平均用时。

表 4　多分类实验结果比较
Table 4　Comparison of multi-class classification results between different networks

算法	宏查准率	宏查全率	宏 F1 值	精度	耗时/s
DNN	62.27%	62.36%	62.31%	93.54%	22.05
SVM	46.76%	49.58%	48.13%	88.62%	27.81
CNN	58.91%	59.73%	59.32%	91.96%	23.46
DBN	57.65%	58.53%	58.09%	92.12%	21.6

在表 4 中，由于数据集中各恶意软件家族样本数目分布不均，部分小样本家族的查准率、查全率对整体的宏查准率、宏查全率产生了较大的影响，从而导致相关度量值普遍较低。由表 3 与表 4 比较可得，在这些被错误分类的样本中，发生应用程序性质的判断错误属于少数情况，大部分问题出现在恶意软件内部家族的分类失误。实验结果表明，DNN 的多分类精度可达 93.54%，考虑到恶意软件家族数量之多，这已经颇有成效。DBN 与 CNN 的分类精度仅次于 DNN，而 SVM 与 3 种深度学习网络之间有一定的距离。在这些比较对象中，DBN 与 DNN 的分类性能最为接近，且耗时略小于 DNN，但从整体表现来说，DNN 仍是最佳选择。

5　结束语

识别恶意软件是 Android 安全领域中一个重要的问题。为了应对这一问题，本文提出一种基于深度神经网络的 Android 恶意软件检测方法，既可以判断应用程序是良性应用还是恶意软件，又能识别恶意软件所属的家族。本文方法采用静态分析与机器学习相结合的概念，具有较强的可扩展性。对 Android 应用的静态分析能够从多个角度较为全面地提取相关特征，较好地捕获了良性应用与恶意软件之间的差异。基于遗传算法的特征选择，对原始特征空间进行有效的压缩，保留了对于分类来说真正有价值的特征。将良性应用与恶意软件家族一同分类的特殊多分类方法，使深度神经网络能够更大限度地发挥功用。上述系统设计保证了本文方法既能有效地辨别良性应用与恶意软件，又能准确地区分恶意软件家族之间的差异。实验结果表明，深度神经网络表现出比其他机器学习方法更好的分类性能。然而，静态分析方式缺少应用程序的实际执行信息，还面临着代码混淆、动态代码加载、Java 反射和本地代码等带来的挑战。这些问题暂时无法被静态分析工具高效地解决，可能会成为恶意软件可利用的漏洞。这将是未来工作努力提高的方向。

参考文献：

[1] ARP D, SPREITZENBARTH M, HUBNER M, et al. Drebin: efficient and explainable detection of Android malware in your pocket[C]//The 19th Annual Network and Distributed System Security Symposium (NDSS). 2014: 1-12.

[2] AVDIIENKO V, KUZNETSOV K, GORLA A, et al. Mining APPs for abnormal usage of sensitive data[C]//The 37th International Conference on Software Engineering (ICSE). 2015: 426-436.

[3] XU K, LI Y, DENG R H. ICCDetector: ICC-based malware detection on Android[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(6): 1252-1264.

[4] DAHL G E, STOKES J W, DENG L, et al. Large-scale malware classification using random projections and neural networks[C]//The 38th International Conference on Acoustics, Speech and Signal Processing. 2013: 3422-3426.

[5] HUANG W Y, STOKES J W. MtNet: a multi-task neural network for dynamic malware classification[C]//The 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. 2016: 399-418.

[6] TOBIYAMA S, YAMAGUCHI Y, SHIMADA H, et al. Malware detection with deep neural network using process behavior[C]//IEEE 40th Annual Conference on Computer Software and Applications. 2016: 577-582.

[7] 李璐. 基于函数调用图的 Android 恶意软件检测[J]. 现代计算机(专业版), 2018(12): 28-33.
LI L. Android malware detection based on function call graph[J].

Modern Computer, 2018(12): 28-33.
[8] NIX R, ZHANG J. Classification of Android APPs and malware using deep neural networks[C]//The 17th International Joint Conference on Neural Networks. 2017: 1871-1878.
[9] MCLAUGHLIN N, RINCON J M, KANG B J, et al. Deep Android malware detection[C]//The 7th ACM on Conference on Data and Application Security and Privacy (CODASPY). 2017: 301-308.
[10] 杨宏宇, 那玉琢. 一种 Android 恶意软件检测模型[J]. 西安电子科技大学学报, 2019, 46(3): 45-51.
YANG H Y, NA Y Z. Android malware detection model[J]. Journal of Xidian University, 2019, 46(3): 45-51.
[11] 欧阳立, 芦天亮. 基于深度置信网络的 Android 恶意软件检测[J]. 信息技术与网络安全, 2019, 38(5): 22-27.
OUYANG L, LU T L. Android malware detection using deep belief network[J]. Journal of Information Technology and Network Security, 2019, 38(5): 22-27.
[12] 吴招娣, 徐洋, 谢晓尧. 基于 AE-DBN 的 Android 恶意软件检测[J]. 贵州师范大学学报(自然科学版), 2019, 37(3): 96-101.
WU Z D, XU Y, XIE X Y. Android malware detection based on AE-DBN[J]. Journal of Guizhou Normal University (Natural Sciences), 2019, 37(3): 96-101.
[13] 石兴华, 曹金璇, 芦天亮. 基于深度森林的安卓恶意软件行为分析与检测[J]. 软件, 2019, 40(10): 1-5, 72.
SHI X H, CAO J X, LU T L. Analysis and detection of Android malware based on Gcforest[J]. Computer Engineering & Software, 2019, 40(10): 1-5, 72.
[14] YUAN Z L, LU Y Q, WANG Z G, et al. Droid-Sec: deep learning in Android malware detection[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(4): 371-372.
[15] YUAN Z L, LU Y Q, XUE Y B. DroidDetector: Android malware characterization and detection using deep learning[J]. Tsinghua Science and Technology, 2016, 21(1): 114-123.
[16] 苏志达, 祝跃飞, 刘龙. 基于深度学习的安卓恶意应用检测[J]. 计算机应用, 2017, 37(6): 1650-1656.
SU Z D, ZHU Y F, LIU L. Android malware application detection using deep learning[J]. Journal of Computer Applications, 2017, 37(6): 1650-1656.
[17] XU L F, ZHANG D P, JAYASENA N, et al. HADM: hybrid analysis for detection of malware[C]//The 3rd SAI Intelligent Systems Conference. 2016: 702-724.
[18] 王涛, 李剑. 基于深度学习的 Android 恶意软件检测系统的设计和实现[J]. 信息安全研究, 2018, 4(2): 140-144.
WANG T, LI J. Design and implementation of Android malware detection system based on deep learning[J]. Journal of Information Security Research, 2018, 4(2): 140-144.
[19] 周明, 孙树栋. 遗传算法原理及应用[M]. 北京: 国防工业出版社, 1999.
ZHOU M, SUN S D. Genetic algorithm theory and applications[M]. Beijing: National Defense Industry Press, 1999.
[20] ARZT S, RASTHOFER S, FRITZ C, et al. FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android APPs[C]//The 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI). 2014: 259-269.
[21] RASTHOFER S, ARZT S, BODDEN E. A machine-learning approach for classifying and categorizing Android sources and sinks[C]//The 19th Annual Network and Distributed System Security Symposium (NDSS). 2014(42): 1-15.
[22] DESNOS A. Androguard documentation, release 3.3.5[R]. 2019.
[23] 黄文坚, 唐源. TensorFlow 实战[M]. 北京：电子工业出版社, 2017.
HUANG W J, TANG Y. TensorFlow practice[M]. Beijing: Electronic Industry Press, 2017.
[24] ZHOU Y J, JIANG X X. Dissecting Android malware: characterization and evolution[C]//The 33rd IEEE Symposium on Security and Privacy (Oakland). 2012: 95-109.
[25] GoogleGroups. VirusTotal[EB].
[26] 周志华. 机器学习[M]. 北京: 清华大学出版社, 2016.
ZHOU Z H. Machine learning[M]. Beijing: Tsinghua University Press, 2016.

[作者简介]

超凡（1995− ），女，江苏启东人，信息工程大学硕士生，主要研究方向为信息安全、代码信息流分析。

杨智（1975− ），男，河南开封人，博士，信息工程大学副教授，主要研究方向为操作系统安全、云计算安全。

杜学绘（1968− ），女，河南新乡人，博士，信息工程大学教授、博士生导师，主要研究方向为空间信息网络、云计算安全。

孙彦（1986− ），男，江苏南京人，博士，中国电子技术标准化研究院工程师，主要研究方向为数据安全、信息安全国际标准化、网络产品和服务安全。