Journal of Computer Applications
计算机应用,2017,37(6):1650 - 1656
ISSN 1001-9081
CODEN JYIIDU
2017-06-10
http://www.joca.cn

文章编号:1001-9081(2017)06-1650-07
DOI:10.11772/j.issn.1001-9081.2017.06.1650

基于深度学习的安卓恶意应用检测

苏志达*,祝跃飞,刘　龙
(数学工程与先进计算国家重点实验室,郑州 450001)
(*通信作者电子邮箱 dhszd1992@163.com)

摘　要:针对传统安卓恶意程序检测技术检测准确率低,对采用了重打包和代码混淆等技术的安卓恶意程序无法成功识别等问题,设计并实现了 DeepDroid 算法。首先,提取安卓应用程序的静态特征和动态特征,结合静态特征和动态特征生成应用程序的特征向量;然后,使用深度学习算法中的深度置信网络(DBN)对收集到的训练集进行训练,生成深度学习网络;最后,利用生成的深度学习网络对待测安卓应用程序进行检测。实验结果表明,在使用相同测试集的情况下,DeepDroid 算法的正确率比支持向量机(SVM)算法高出 3.96 个百分点,比朴素贝叶斯(Naive Bayes)算法高出 12.16 个百分点,比 *K* 最邻近(*K*NN)算法高出 13.62 个百分点。DeepDroid 算法结合了安卓应用程序的静态特征和动态特征,采用了动态检测和静态检测相结合的检测方法,弥补了静态检测代码覆盖率不足和动态检测误报率高的缺点,在特征识别的部分采用 DBN 算法使得网络训练速度得到保证的同时还有很高的检测正确率。

关键词:安卓;恶意软件检测;恶意代码分析;深度学习
中图分类号:TP309.5　　**文献标志码**:A

Android malware application detection using deep learning

SU Zhida*, ZHU Yuefei, LIU Long
(*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou Henan* 450001, *China*)

Abstract: The traditional Android malware detection algorithms have low detection accuracy, which can not successfully identify the Android malware by using the technologies of repacking and code obfuscation. In order to solve the problems, the DeepDroid algorithm was proposed. Firstly, the static and dynamic features of Android application were extracted and the Android application features were created by combining static features and dynamic features. Secondly, the Deep Belief Network (DBN) of deep learning algorithm was used to train the collected training set for generating deep learning network. Finally, untrusted Android application was detected by the generated deep learning network. The experimental results show that, when using the same test set, the correct rate of DeepDroid algorithm is 3.96 percentage points higher than that of Support Vector Machine (SVM) algorithm, 12.16 percentage points higher than that of Naive Bayes algorithm, 13.62 percentage points higher than that of *K*-Nearest Neighbor (*K*NN) algorithm. The proposed DeepDroid algorithm has combined the static features and dynamic features of Android application. The DeepDroid algorithm has made up for the disadvantages that code coverage of static detection is not enough and the false positive rate of dynamic detection is high by using the detection method combined dynamic detection and static detection. By using the DBN algorithm in feature recognition, the proposed DeepDroid algorithm has guaranteed high network training speed and high detection accuracy at the same time.

Key words: Android; malware detection; malicious code analysis; deep learning

0　引言

Android 作为当前最为流行的移动智能操作系统,设备和用户数量庞大,应用程序丰富,其安全性受到了广泛关注。而且移动设备与用户的真实身份密切相关,随着移动设备的智能化,移动设备中包含的用户位置信息、社会关系、隐私数据等敏感信息越来越多,使其安全问题更显突出。来自 TrustGo 公司的分析报告显示,GooglePlay 上有 3.15% 的应用可能泄露用户隐私或者存在恶意行为,而国内知名的 91 应用市场这一比例则为 19.7%。因我国 Android 用户无法直接从 GooglePlay 下载和安装应用,导致了大量管理混乱的第三方应用市场存在,对于 Android 设备的安全造成了严重威胁。

随着移动智能终端的高速发展,手机恶意软件的不断涌现已经成为亟待解决的安全问题。2009 年 11 月,Android 平台上发现了首个间谍程序 Mobile Spy,该程序可以记录用户的键盘输入信息并发送至攻击者邮箱,还可以视频监控用户操作。2010 年 8 月,Android 平台出现了首个木马程序 FakePlayer,攻击者可以利用该程序控制用户设备不断发送短信开通高额收费服务。2011 年 12 月,出现了首个恶意程序 Carrier IQ,该程序能够实时监控用户设备状态,记录用户位置信息,静默收集敏感信息,并获取设备管理员权限[1]。之后,随着 Android 设备的广泛普及,Android 平台的恶意程序也出现了爆发式增长。来自 360 的安全报告显示,2011 年至 2015 年 Android 恶意程序样本数量和受感染人次的发展迅速[2],

收稿日期:2016-11-17;**修回日期**:2017-02-20。　**基金项目**:国家自然科学基金资助项目(61271252)。
作者简介:苏志达(1992—),男,内蒙古赤峰人,硕士研究生,主要研究方向:安卓逆向、安卓安全检测;　祝跃飞(1962—),男,浙江杭州人,教授,博士,主要研究方向:计算数论、密码学、信息安全;　刘龙(1983—),男,河南郑州人,讲师,硕士,主要研究方向:漏洞挖掘、移动安全。

2011 年时恶意程序样本数量为 0.5 万个,受感染人次为 489 万,而 2014 年全年监控到的恶意程序样本已经达 326.0 万个,共有 3.19 亿人次被感染,2015 年这一数据增长 5.7 倍,全年截获样本数达到 1 874.0 万个,累计 3.7 亿人次被感染,恶意应用程序已经成为 Android 智能终端的最主要安全威胁。

随着恶意代码的进化,基于特征和规则检测方法的局限性日益突出。在静态分析和动态分析的方法中,有研究成果使用了机器学习技术。2009 年,Schimidt 等[3]使用静态分析方法获取函数调用特征,对 Symbian 和 Android 系统的应用程序进行基于分类算法的恶意代码检测;Shabtai 等[4]提出基于主机的恶意 Android 应用检测框架 Andromaly,通过从移动端获取多种特征和行为记录来训练分类器,以完成对应用程序的分类。Burguera 等[5]使用动态方法提取系统调用特征,应用 *K*-means 聚类算法区分基于同一应用程序改写的恶意代码和非恶意代码。

深度学习(DeepLearning, DL)是近年来出现的机器学习领域方法[6]。深度学习通过学习深层非线性网络结构,实现复杂函数逼近,表征输入数据分布式表示,展现从样本集中学习数据集本质特征的强大能力。深度学习与经典神经网络的区别在于:1) 强调模型结构的深度,通常有 3 层、5 层,甚至 10 多层的隐层节点;2) 突出特征学习的重要性,逐层训练特征,将样本在原空间的特征表示逐层变换到新特征空间,使分类或预测更加准确。

DeepLearning 的主要思想是把学习结构(learning hierarchy)看作是一个网络(network),无监督学习用于每一层网络的 pre-train;每次用无监督学习只训练一层,并将其训练结果作为其更高(更抽象)一层的输入;用有监督学习去调整所有层。最终,构建一个深度有监督学习的分类器,如神经网络分类器,或构建一个深度生成模型,如深度玻尔兹曼机(Deep Boltzmann Machine, DBM)[7]和深度置信网络(Deep Belief Network, DBN)。

深度学习的优势在于它是具有多层非线性映射的深层结构,可以完成复杂的函数逼近。此外深度学习理论上可以获取分布式表示,即可通过逐层学习算法获取输入数据的主要驱动变量[8]。通过深度学习的优势主要表现在非监督预训练算法中,通过生成性训练避免因网络函数表达能力过强而出现的过拟合情况。由于单层计算能力有限,深度学习通常采用多层映射单元提取主要的结构信息[9]。

基于上述原因,本文结合动态特征和静态特征,采用深度学习的算法进行安卓代码特征分析:一方面结合静态特征和动态特征来获取完整的应用程序信息;另一方面利用深度学习的深层学习机制自动挖掘深层特征,通过这种动静结合的深度学习系统对安卓应用进行检测。

1　特征提取

Android 应用程序的行为信息分为静态信息和动态行为。任何针对应用程序的分析手段都依赖于对应用程序行为信息的获取。本章从静态信息和动态行为两个方面,分别设计了自动化程序行为信息提取方法。首先针对静态信息,综合优化现有提取手段,给出了标准化提取流程和方法;然后重点阐述了对动态行为的提取,提出了一种基于虚拟化技术的动态行为提取技术,实现了对内核指令、系统调用、Dalvik 指令等多个层次的程序行为特征提取。

1.1　静态特征提取

Android 应用程序静态信息的获取是指在不执行应用程序的情况下,使用逆向工程手段,提取待分析程序的静态特征。本文中主要是对其 AndroidManifest.xml 文件的获取和解析。AndroidManifest.xml 文件定义了多种标签来声明应用程序的基本信息、组件、权限等,本文获取的静态信息如表 1 所示。

表 1　静态信息列表

Tab. 1　Static information list

获取目标信息	描述
Android 安装程序文件 MD5 值	以安装程序文件的 MD5 值作为程序样本的识别信息
程序包名	应用程序安装运行后的进程标识
启动 Activity 名称	应用程序的主启动界面标识
权限信息	应用程序所申请的权限信息

现有静态信息提取方法主要为使用 APKTool 工具或 aapt(android asset packaging tool)工具。aapt 工具是 Android SDK 提供的工具之一,不解包 APK(AndroidPackage,Android 安装包)文件,可以对资源文件和 APK 文件的基本信息进行管理,提取 AndroidManifest.xml 文件中的信息。aapt 工具的执行效率远高于 apktool,故本文以 aapt 工具为基础进行静态信息的提取。

静态信息的提取流程如图 1 所示。在样本程序静态信息的提取过程中,对其所申请的权限信息进行了预处理,主要是:

1) 去除自定义的权限信息。Android 允许程序自定义权限,以对其他应用能否访问该应用的组件进行管理。自定义权限与 Android 应用程序编程接口(Application Programming Interface, API)以及系统资源的保护无关,故本文只关注应用程序申请的 Android 系统所定义的权限。

2) 为所有权限赋予唯一标识。AndroidManifest.xml 文件中的权限信息为字符串,系统在运行程序时才会对其进行解析并识别为对应 ID。为了之后分析工作的便利,在提取过程中,为每一种新提取到的权限信息赋予其全局唯一的整数型 ID。通过此操作,简化了数据处理和运算过程。

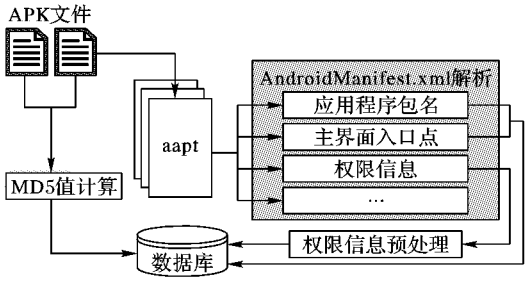

图 1　静态信息提取流程

Fig. 1　Flow chart of static information extraction

1.2　动态特征提取

由于 Android 的多层次体系架构,程序语义在执行中也会有多层次的展现。对于恶意程序包含的恶意代码,也会在多个层面上表现出恶意行为。越处于底层的恶意行为,在系统中隐蔽性越高,反之则越明显。多层次的恶意代码展现也为恶意行为的监控提供了多个层次选择,部署于不同层次的

监控方法，所能够获取的语义层次也不同。图 2 给出了 Android 系统各个层次所对应的程序行为，及其可能存在的攻击行为。理论上，下层的监控方法可以捕获或还原上层的恶意行为，而上层的监控方法对下层的语义执行信息无法获知。

图 2 动态行为监控层次

Fig. 2 Dynamic behavior monitoring levels

快速模拟器（Quick Emulator，QEMU）是采用了动态二进制翻译技术的高效 CPU 模拟器，本文使用 QEMU 模拟器分别获取了内核指令、系统调用、Dalvik 指令的信息，从这些信息中提取应用程序动态行为特征。整体结构如图 3 所示。

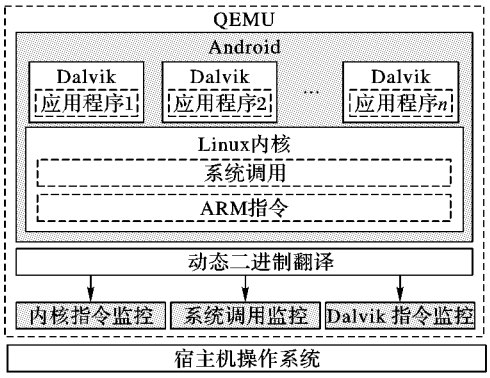

图 3 动态监控系统架构

Fig. 3 Dynamic monitoring system architecture

1）内核指令。

要对客户机内的 Android 系统进行监控，首先要获取到每个翻译好的宿主机基本指令块（X86 指令）以及其对应的原始客户机指令块（进阶指令机（Advanced Risc Machine，ARM）指令）。在微代码引擎（Tiny Code Generator，TCG）翻译阶段插装代码，读取原始代码块以及翻译后代码块的起始地址和大小，在每个代码段执行之前，再根据所记录的位置和大小，将翻译前和翻译后的指令记录到本地文件，将 X86 指令还原到 ARM 指令。

2）系统调用。

在 X86 架构中，从指令层面来看系统调用的流程为：1）int0x80 指令来进入特权模式；2）通过 eax 寄存器传递要执行的系统调用编号；3）内核执行相应系统调用。ARM 架构中使用 swi #0 来执行系统调用，系统调用编号存储于 R7 寄存器之中。在获取原始指令块的模块中加入一个监控模块，当发现模拟器动态翻译的指令块中有 swi #0 指令时，在 swi #0 指令后插装额外的 TCG 指令，执行一个回调函数来收集内存和寄存器中关于此次调用的信息，首先取得 R7 寄存器中的此次系统调用编号，再根据内存中的系统调用表查找到该编号对应的系统调用名称及地址，从而获取到进程信息、系统调用的参数以及返回值等。经过插装的代码被翻译执行时，我们插入的回调函数也将被调用，从而获得系统调用信息。

3）Dalvik 指令。

Dalvik 字节码通过 Dalvik 虚拟机翻译为相应的可执行的本地指令并执行，采用解释执行（Interpretation）和实时编译（Just-In-Time compilation，JIT）两种方式来完成这一过程。

Dalvik 的解释器模块名为 mterp，采用偏移寻址方式将 Dalvik 操作码（opcode，指令编号）映射到机器码。每条操作码都有 64 B 的内存用于存储其相应模拟解释的本地代码，对于模拟解释代码不足 64 B 的操作码，会进行填充，使其存储满 64 B。mterp 计算偏移地址公式：

$$NativeAddress = BaseAddress + opcode * 64 \quad (1)$$

其中：*NativeAddress* 为该操作码对应的本地代码；*BaseAddress* 为整个存储操作码对应的本地代码的内存块的起始地址；Dalvik 虚拟机可以通过此式计算得到每个操作码对应的本地代码。根据 Dalvik 指令的完整还原过程，本文对 mterp 和 JIT 分别进行处理。

对于 mterp 解释执行的指令，Dalvik 虚拟机中程序计数器（Program Counter，PC）的值存储于 *R*15 寄存器中，通过监控 *R*15，可以获知 mterp 的行为，由式（1），其执行的 *opcode* 为：

$$opcode = (R15 - BaseAddress)/64 \quad (2)$$

通过式（2）获得 mterp 执行的 *opcode*，由于每个 *opcode* 对应唯一的一个指令，从而得到当前执行的指令。

JIT 编译时一个代码段中包含多个基本的代码块，难以和指令一一对应，因此禁用 JIT，仅使用 mterp 解释执行。

1.3 特征向量

根据 1.1 节和 1.2 节提取的静态信息和动态信息可以构筑特征向量。静态信息部分，本文对在 Google Play Store 得到的 12 170 个非恶意样本和在 VirusTotal 上得到的 2678 个恶意应用进行了静态信息提取，从所有权限中采用了 41 个使用频率比较高的权限作为静态特征。若应用程序被检测到上述权限就会在应用程序的特征向量中被标记为 1，否则为 0。

动态信息部分，本文从 Android 系统中选择了 129 个关键动态特征来对 Android 应用程序的行为进行刻画，覆盖了进程控制、文件操作、文件系统操作、系统控制、网络管理、socket 控制、用户控制和进程间通信 8 种类别。应用程序的动态分析中若检测到相应的动态特征则在特征向量中标记为 1，否则为 0。

根据提取的 41 个静态特征和 129 个动态特征，本文构建一个包含 170 个特征的特征向量，其中前 41 个是代表静态信息的静态特征，后 129 个是代表动态信息的动态特征，每一个特征在向量中的取值只有 0 和 1，当应用程序中检测到相应特征时，特征值为 1，没有检测到时特征值为 0。

2 深度学习算法

目前基于 DL 理论的应用系统中，深度置信网络（DBN）是应用比较广泛的一类学习结构，它由多层受限波尔兹曼机（Restricted Boltzmann Machine，RBM）单元和一层有监督网络层组成。陈宇等[10]采用 DBN 算法对中文实体关系进行了分析，取得了很好的结果，本文参照其中的 DBN 网络结构设计了 DBN 分类网络，利用 DBN 对提取的应用程序特征向量进行分类。与其他深度学习网络（深度玻尔兹曼机、卷积神经网络等）相比，DBN 算法对一维特征向量的分类能力更强且训练速度更快，因此本文采用 DBN 算法对恶意应用进行分类。本文所采用的 DBN 是由若干层自底向上的 RBM 和一层有监督的反向传播（Back Propagation，BP）网络组成的深层神经网络。

2.1　深度置信网络

深度置信网络(DBN)是由若干层自底向上的 RBM 和一层有监督的 BP 网络组成的深层神经网络[6],其结构如图 4 所示。图 4 中,$\boldsymbol{V}$和$\boldsymbol{H}$分别表示可视层和隐含层内的节点值,$\boldsymbol{W}$表示可视层与隐含层之间的权值。底层的神经网络接收原始的特征向量,在自底向上的传递过程中,从具体的特征向量逐渐转化为抽象的特征向量,在顶层的神经网络形成更易于分类的组合特征向量。为确保每一层的联合概率分布 $p(\boldsymbol{V},\boldsymbol{H})$ 最大,必须使当前层 RBM 网络调整自身层内的权值以对该层特征向量映射达到最优;但仅依靠 RBM 层并不能对整个 DBN 的特征向量映射达到最优。因此,BP 网络还担负着微调功能:BP 网络将错误信息自顶向下传播至每一层 RBM,可以达到微调整个 DBN 的作用。

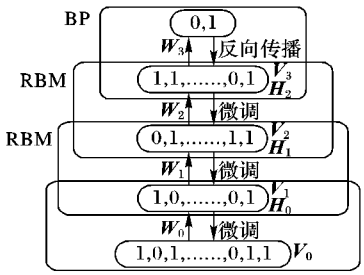

图 4　DBN 网络结构

Fig. 4　DBN network architecture

在训练模型的过程中,DBN 主要分为两步:1)分别单独无监督地训练每一层 RBM 网络,确保特征向量映射到不同特征空间时,都尽可能多地保留特征信息;2)在 DBN 的最后一层设置 BP 网络,接收 RBM 的输出特征向量作为它的输入特征向量,有监督地训练实体关系分类器。将特征向量 $\boldsymbol{x}$ 作为第一层 RBM 的输入 $\boldsymbol{V}_0$,计算第一层的输出 $\boldsymbol{H}_0$ 并修改权值 $\boldsymbol{W}_0$,在计算 $\boldsymbol{H}_0$ 时采用 Sigmoid 函数进行标准化,Sigmoid 函数的结果作为特征节点开启(结果为 1)的概率值 p,将 p 与一个从(0,1)均匀分布中抽取的随机值 u 比较,若 $p > u$ 则该节点开启,其值取 1,否则值取 0,因此 $\boldsymbol{H}_0$ 的每个特征取值均为 0 或 1。当第一层训练完成后,将第一层的输出 $\boldsymbol{H}_0$ 作为第二层的输入 $\boldsymbol{V}_1$,同样计算 $\boldsymbol{H}_1$ 并修改权值 $\boldsymbol{W}_1$,以此类推充分训练每层 RBM。当所有 RBM 层都单独充分训练完成后,将最后一层 RBM 的输出 $\boldsymbol{H}_{n-1}$(n 为 RBM 隐藏层层数)作为 BP 层的输入 $\boldsymbol{V}_n$,BP 网络算法有监督地训练分类器,并自顶向下反向微调整个 DBN。DBN 训练过程实现步骤如算法 1 所示。

算法 1　DBN 训练算法。

对于层数为 N 的 DBN 算法,$n(0 < n \leqslant N)$ 为当前 RBM 层数。

1)　充分训练第一层 RBM,$n = 1$。

2)　固定当前 RBM 的权重和偏移量,将当前 RBM 的输出作为下一个 RBM 的输入向量。

3)　充分训练下一个 RBM 后,堆叠在当前 RBM 的上方,$n \leftarrow n + 1$。

4)　若 $n = N$,转步骤 5);否则,转步骤 2)。

5)　使用 BP 网络有监督地训练分类器,并自顶向下反向微调整个 DBN 网络。

6)　训练结束。

2.2　限制玻尔兹曼机

受限玻尔兹曼机(RBM)由 Hinton 等[11]提出,该网络由可见单元(visible unit,对应可见变量,亦即数据样本)和隐藏单元(hidden unit,对应隐藏变量)构成,可见变量和隐藏变量都是二元变量,其状态取{0, 1}。整个网络是一个二部图,可见单元和隐藏单元之间连接,可见单元之间以及隐藏单元之间都不连接。

RBM 的训练过程,实际上是求出一个最能产生训练样本的概率分布。也就是说,要求一个分布,在这个分布里,训练样本的概率最大。由于这个分布的决定性因素在于权值 $\boldsymbol{W}$,所以本文训练 RBM 的目标就是寻找最佳的权值。

对于包含 N 个二值可见单元和 M 个二值隐单元的 RBM,设定 v_i 表示第 i 个可见单元的状态,h_j 表示第 j 个隐单元状态.那么,给定状态$(\boldsymbol{V},\boldsymbol{H})$ 所具备的能量定义如式(3):

$$E(\boldsymbol{V},\boldsymbol{H}) = -\sum_{i=1}^{N} v_i b_i - \sum_{j=1}^{M} h_j c_j - \sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij} v_i h_j \tag{3}$$

式中:w_{ij} 表示可视层与隐藏层之间的权值;b_i 表示可视单元的偏置;c_j 表示隐单元的偏置。RBM 处于状态$(\boldsymbol{V},\boldsymbol{H})$ 的概率如式(4)所示:

$$P(\boldsymbol{V},\boldsymbol{H}) = \frac{\exp(-E(\boldsymbol{V},\boldsymbol{H}))}{\sum_{\boldsymbol{V},\boldsymbol{H}} \exp(-E(\boldsymbol{V},\boldsymbol{H}))} \tag{4}$$

由于层间单元是无连接的,由可视层的节点值得到隐含层的节点值计算式[12]如式(5):

$$P(h_j = 1 \mid \boldsymbol{V}) = \sigma\left(\sum_{i=1}^{N} W_{ij} v_i + c_j\right) \tag{5}$$

RBM 是对称网络,同理,利用式(6)可以由已知的隐含层的节点值得到可视层的节点值:

$$P(v_i = 1 \mid \boldsymbol{H}) = \sigma\left(\sum_{j=1}^{M} W_{ij} h_j + b_i\right) \tag{6}$$

式(5) ~ (6) 中,$\sigma(x) = 1/(1 + \exp(-x))$。

通常应该采用马尔可夫链的方法计算权值 $\boldsymbol{W}$,但是马尔可夫链的计算复杂且收敛速度难以保证,因此本文采用对比散度(Contrastive Divergence, CD)的学习算法[13], CD 准则计算速度快且保持精度,本文利用这个算法计算权值 $\boldsymbol{W}$。CD 准则利用 Kullback-Leibler 距离衡量两个概率分布的"差异性",表示为 $KL(P \parallel P')$,如式(7)所示:

$$CD_n = KL(p_0 \parallel p_\infty) - KL(p_n \parallel p_\infty) \tag{7}$$

其中:p_0 为 RBM 网络初始状态的联合概率分布;p_n 为经过 n 步马尔可夫链之后的 RBM 网络的联合概率分布;p_∞ 为马尔可夫链末端的 RBM 网络的联合概率分布。所以,CD_n 可以看作是 p_n 衡量介于 p_0 和 p_∞ 之间的位置。不断地将 p_n 赋值给 p_0,得到新的 p_0 和 p_n。由于 n 取 1 时就可以得到很好的结果[14],因此本文中 $n = 1$。RBM 的训练过程实现步骤如算法 2 所示。

算法 2　基于 CD 准则的 RBM 网络自训练过程。

对于容量为 N 的训练集中的一个特征向量 $\boldsymbol{x}_n(0 \leqslant n < N)$

1)　$n = 0$。

2)　将 $\boldsymbol{x}_n$ 附给显层 $\boldsymbol{V}_0$,根据式(5)计算隐藏层 $\boldsymbol{H}_0$:

$$P(h_{0j} = 1 \mid \boldsymbol{V}_0) = \sigma(\boldsymbol{W}_j \boldsymbol{V}_0)$$

3)　根据式(6)重构显层得到 $\boldsymbol{V}_1$:

$$P(v_{1i} = 1 \mid \boldsymbol{H}_0) = \sigma(\boldsymbol{W}_i^{\mathrm{T}} \boldsymbol{H}_0)$$

4)　再次根据式(5)计算隐藏层 H_1

$$P(h_{1j} = 1 \mid \boldsymbol{V}_1) = \sigma(\boldsymbol{W}_j \boldsymbol{V}_1)$$

5)　对于所有的节点 j,按下式更新权重:

$$\boldsymbol{W}_j \leftarrow \boldsymbol{W}_j + \lambda(P(h_{0j} = 1 \mid \boldsymbol{V}_0)\boldsymbol{V}_0^{\mathrm{T}} - P(h_{1j} = 1 \mid \boldsymbol{V}_1)\boldsymbol{V}_1^{\mathrm{T}})$$

若 $n = N - 1$,结束;否则 $n = n + 1$,转步骤 2)。

2.3　BP 网络

BP 网络是有监督分类器,在 DBN 的最后一层对前端提

取的特征向量进行分类，并与正确结果比对，进而微调整个 DBN。本文采用 BP 网络训练方法[14]，利用 Sigmod 函数作为 BP 网络节点的求值函数。其训练过程如算法 3 所示。

算法 3　BP 网络的训练过程。

1）　随机初始化顶层反向传播网络的参数，设定训练步长为 N。

2）　进行前向计算，对第 l 层的 j 单元节点，值为 $y_j^l(n) = \sum W_{ij}(n) y_i^{l-1}(n)$，若神经元 j 属于输出层（$l = L$），则令 $y_j^L(n) = o_j(n)$，误差 $e_j(n) = d_j(n) - o_j(n)$，d_j 为正确结果。

3）　计算 δ，将 δ 反向传递自顶向下修正权值，对于输出单元：

$$\delta_j^l(n) = e_j(n) o_j(n)[1 - o_j(n)]$$

对于隐藏单元：

$$\delta_j^l(n) = y_j^l(n)[1 - y_j^l(n)] \sum \delta_k^{l+1}(n) W_{kj}^{l+1}(n)$$

4）　修改权值：

$$W_{ji}^l(n+1) = W_{ji}^l(n) + \eta \delta_j^l y_i^{l-1}(n)$$

η 为学习速率。

5）　如果 $n = N$，训练结束；否则 $n = n + 1$，转步骤 2）。

3　基于深度学习的安卓应用检测

本文设计了 DeepDroid 算法，提取并结合了安卓应用的动态特征和静态特征，使用深度学习算法中的 DBN 算法对安卓应用进行分析。静态特征和动态特征的结合使得特征向量对安卓应用的描述更加全面，同时采用 DBN 算法可以学习特征的深层结构，使安卓应用的检测更加精确。DeepDroid 算法的结构如图 5 所示。

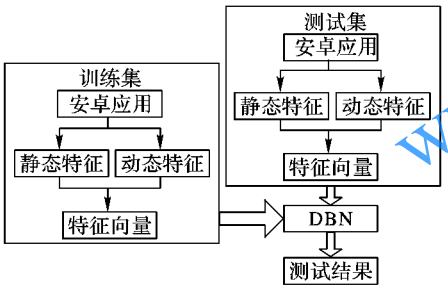

图 5　DeepDroid 算法结构

Fig. 5　Structure diagram of DeepDroid algorithm

3.1　提取特征向量

特征向量由静态特征和动态特征共 170 种特征组成，其中前 41 种特征为静态特征，后 129 种特征为动态特征，针对每个安卓应用分别检测它是否含有相应的静态特征和动态特征行为，并根据检测结果得到该安卓应用的特征向量。对于特征向量中的每一个特征值，若检测到相应的特征则标记为 1，否则标记为 0。

3.2　深度学习网络训练

DBN 由多层 RBM 组成，最后一层采用 BP 网络。DBN 的训练分为两步：1）分别单独无监督地训练每一层 RBM 网络，确保特征向量映射到不同特征空间时，都尽可能多地保留特征信息；2）在 DBN 的最后一层设置 BP 网络，接收 RBM 的输出特征向量作为它的输入特征向量，有监督地训练实体关系分类器。每一层 RBM 网络只能确保自身层内的权值对该层特征向量映射达到最优，并不是对整个 DBN 的特征向量映射达到最优，所以反向传播网络还将错误信息自顶向下传播至每一层 RBM，微调整个 DBN。与传统的神经网络所有层同时训练不同，深度学习算法每层隐层都单独充分训练后再训练下一层。将 3.1 节中得到的特征向量集 $\boldsymbol{X}$ 作为第一层 RBM 的输入 $\boldsymbol{V}_0$，通过算法 2 计算修改权值 $\boldsymbol{W}_0$。当第一层训练完成后，将第一层的输出 $\boldsymbol{H}_0$ 作为第二层的输入 $\boldsymbol{V}_1$，同样通过算法 2 计算修改权值 $\boldsymbol{W}_1$，以此类推充分训练每层 RBM。当所有 RBM 层都单独充分训练完成后，将最后一层 RBM 的输出 $\boldsymbol{H}_{n-1}$（n 为 RBM 隐藏层层数）作为 BP 层的输入 $\boldsymbol{V}_n$，通过 2.3 节中的 BP 网络算法有监督地训练分类器，并自顶向下反向微调整个 DBN。

3.3　恶意应用检测

通过 3.2 节训练得到的深度置信网络，就可以对安卓应用进行安全检测，将安卓应用的特征向量输入到网络中就可以对其进行检测。实验部分将会使用测试集对 DeepDroid 算法的准确率进行验证。

4　实验结果与分析

本文在 Google Play Store 共下载得到近 13 000 个应用程序（https://github.com/Akdeniz/google-play-crawler），去除掉不可解包的应用后共得到 12 170 个非恶意样本。恶意样本来自 VirusTotal（https://www.virustotal.com/），共计 2 678 个恶意应用。训练集和测试集均由 1 300 个非恶意程序以及 1 300 个恶意程序组成。

为了验证本文所提方法的有效性，将本文方法与其他传统方法进行比较，设计了 3 组实验。第 1 组实验采用不同 RBM 层数和隐藏层节点数，确定检测准确率最高的深层网络结构；第 2 组实验采用不同的特征向量，验证动静态特征结合对恶意程序检测的有效性；第 3 组实验采用不同的机器学习算法，验证了本文采用的 DBN 算法比传统的机器学习算法更加有效。在实验中，采用准确率（Precision）、召回率（Recall）、F 系数（F-Measure）和正确率（accuracy）来评价对恶意行为检测的结果。深度学习网络的隐藏层数越多，隐藏层节点数越多其学习能力越强，但是由于样本集数量的关系，过多的层数不但导致效率低下，还可能只是干扰层反而影响学习结果。由于目前关于深度网络结构的选取还没有完善的理论依据，本文通过实验的方法来确定最优的网络结构。

本文测试了多种网络结构，首先采用单隐藏层，DBN 网络结构为[170,140,2]，输入层节点数为 170，隐藏层节点数为 140，输出层节点数为 2，学习速率设置为 0.05，采用单隐藏层的结构时收敛速度较快，检测的正确率为 95.96%。采用双层隐藏层时，DBN 网络结构为[170,140,140,2]，输入层节点数为 170，隐藏层节点数为 140，输出层节点数为 2，学习速率设置为 0.05，检测的准确率为 97.08%，收敛速度变慢。采用三层隐藏层时，DBN 网络结构为[170,140,140,140,2]，输入层节点数为 170，隐藏层节点数为 140，输出层节点数为 2，学习速率设置为 0.05，检测的准确率为 96.73%，虽然隐藏层层数增加了，但是正确率却略微下降。采用四层隐藏层时，DBN 网络结构为[170,140,140,140,140,2]，输入层节点数为 170，隐藏层节点数为 140，输出层节点数为 2，学习速率设置为 0.05，检测的准确率为 94.92%，收敛速度非常慢。除了隐藏层层数，隐藏层节点数也会对网络的效率有一定的影响，修改前面实验中四个网络的隐藏层节点数为 160，得到四个新的网络，分别为单隐藏层网络[170,160,2]、双隐藏层网络[170,160,160,2]、三隐藏

层网络[170,160,160,160,2]、四隐藏层网络[170,160,160,160,160,2],其正确率分别为96.54%、96.58%、94.54%、94.54%。同样将隐藏层数为二、三、四的三个网络隐藏层节点数修改为 120,得到三个新的网络,分别为双隐藏层网络[170,120,120,2]、三隐藏层网络[170,120,120,120,2]、四隐藏层网络[170,120,120,120,120,2],其正确率分别为96.35%、96.08%、94.35%。当采用两层隐藏层,隐藏层节点数为 140,网络结构为[170,140,140,2]时检测的正确率最高为97.08%。实验 1 的结果如表 2 所示。

为了验证结合静态特征和动态特征作为特征向量比单独采用某种特征向量有更好的性能,在实验 2 中,分别使用态静态特征、动态特征、动静态结合特征作为输入特征向量进行恶意行为检测,结果如表 3 所示。从表 3 的结果中可以看出:采用静态特征可以更加准确地检测恶意应用,而非恶意应用的准确率低说明有很多误报;采用动态特征时,恶意应用的准确率较低,非恶意应用的准确率与采用静态特征的结果相比更高;当结合了动态和静态特征时,恶意应用和非恶意应用的准确率都比较高,与单独使用一种特征相比性能有很大提升。

在实验 3 中,比较与传统的机器学习模型和深度学习模型,结果如表 4 所示。对于文中的其他机器学习算法(SVM、Naive Bayes、*KNN*),测试了 linear kernel、polynomial kernel、sigmoid kernel 等多种核函数,并选取性能最好的作为实验结果。从表 4 中可以清楚地看到,在使用相同测试集的情况下,DBN 算法的正确率比 SVM 算法高出 3.96 个百分点,比 Naive Bayes 算法高出 12.16 个百分点,比 *KNN* 算法高出13.62个百分点,深度学习的模型明显优于其他恶意软件检测模型。

表 2　不同深度学习网络结构检测结果　%

Tab. 2　Detection results of different deep learning network structures　%

层数	隐藏层节点数	恶意应用			非恶意应用			正确率
		准确率	召回率	F 系数	准确率	召回率	F 系数	
4	[160,160,160,160]	95.37	93.61	94.48	93.73	95.46	94.59	94.54
4	[140,140,140,140]	95.70	94.07	94.88	94.18	95.76	94.96	94.92
4	[120,120,120,120]	95.14	93.46	94.29	93.57	95.23	94.39	94.35
3	[160,160,160]	92.07	97.46	94.69	97.30	91.62	94.37	94.54
3	[140,140,140]	95.85	97.69	96.76	97.65	95.77	96.70	96.73
3	[120,120,120]	95.11	97.15	96.12	97.09	95.00	96.03	96.08
2	[160,160]	96.26	96.92	96.59	96.90	96.23	96.56	96.58
2	[140,140]	96.29	97.92	97.10	97.89	96.23	97.05	97.08
2	[120,120]	95.13	97.69	96.39	97.63	95.00	96.30	96.35
1	[160]	95.83	97.31	96.56	97.27	95.77	96.51	96.54
1	[140]	95.09	96.92	96.00	96.86	95.00	95.92	95.96

表 3　不同特征向量检测结果　%

Tab. 3　Detection results of different feature vectors　%

特征	恶意应用			非恶意应用			正确率
	准确率	召回率	F 系数	准确率	召回率	F 系数	
静态	97.12	97.69	97.40	81.41	77.69	79.51	87.69
动态	86.18	75.31	80.38	78.07	87.92	82.70	81.62
静态 & 动态	96.29	97.92	97.10	97.89	96.23	97.05	97.08

表 4　不同机器学习算法检测结果　%

Tab. 4　Detection results of different machine learning algorithms　%

算法	恶意应用			非恶意应用			正确率
	准确率	召回率	F 系数	准确率	召回率	F 系数	
SVM	92.49	93.85	93.17	93.75	92.38	93.06	93.12
Naive Bayes	80.63	91.92	85.91	90.61	77.92	83.79	84.92
KNN	81.75	86.15	83.89	85.37	80.77	83.01	83.46
DBN	96.29	97.92	97.10	97.89	96.23	97.05	97.08

实验结果表明,深度学习算法与其他传统机器学习算法相比有更好的性能。当深度学习网络结构在当前数据集下达到最高性能时,深度学习算法有很高的准确率。从表 2 中可以看到,即使深度学习网络没有采用性能最佳的 2 隐层结构,其最低正确率也达到 94.35%。

5　结语

本文通过静态特征和动态特征结合的检测方法,使用深度置信网络(DBN)对提取的安卓应用程序特征进行分析识别,实现了 DeepDroid 安卓恶意软件检测算法。本文实验中在 Google Play Store 得到 12 170 个非恶意样本,在 VirusTotal 上得到2678 个恶意应用,并通过这些样本构建实验集。实验中,首先对应用程序进行了特征提取,并将提取到的静态特征和动态特征整合到同一个特征向量中;然后,从数据集中选取部分非恶意应用和部分恶意应用,使用这些应用程序的特征向量训练 DBN;最终,通过训练好的 DBN 对安卓恶意应用程序进行识别。深度学习是机器学习研究的一个新领域,与传统机器学习方法相比,深度学习可以学习到数据中更深层的特征,从而提高检测性能。本文中共提取了 170 个特征,其中包含 41 个常见恶意静态特征和 129 个高危动态特征,动态特征包含内核指令、系统调用以及 Dalvik 指令。根据提取到的 170 个特征将数据集通过深度学习算法对特征向量进行分类。从实验结果可以看出,虽然训练集不同,深度学习网络的性能可能会略有不同,但是与传统机器学习算法相比,深度学习算法依然有更高的准确率。

本文研究依然存在一些不足和可以改进的地方:由于恶意应用收集比较困难,本文收集了 2 678 个恶意应用,因此数据集相对较小,如果采用更大的数据集则可以得到更准确的结果。此外特征向量的值可以采用离散值来增加其信息量,采用二进制(0,1)只能表示检测到了某种恶意行为,而采用离散值可以增加其含义。比如相同的恶意行为出现了两次,可以标记为 2,有些高危特征可以加权来突出其危险性。

参考文献(References)

[1] 杨欢. 协议漏洞挖掘及 Android 平台恶意应用检测技术研究[D]. 西安: 西安电子科技大学, 2014: 2 - 3. (YANG H. Research on protocols vulnerability discovery and Android malware detection [D]. Xi'an: Xidian University, 2014: 2 - 3.)

[2] 360 互联网安全中心. 2015 年中国手机安全状况报告[R/OL]. [2016- 09- 15]. http://zt. 360. cn/1101061855. php? dtid = 1101061451&did = 1101593997. (360 Internet Security Center. China mobile security status report 2015 [R/OL]. [2016-09-15]. http://zt. 360. cn/1101061855. php? dtid = 1101061451&did = 1101593997.)

[3] SCHMIDT A D, BYE R, SCHMIDT H G, et al. Static analysis of executables for collaborative malware detection on Android [C]// Proceedings of the 2009 IEEE International Conference on Communications. Piscataway, NJ: IEEE, 2009: 631 - 635.

[4] SHABTAI A, KANONOV 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.

[5] BURGUERA I, ZURUTUZA U, NADJM-TEHRANI S. Crowdroid: behavior-based malware detection system for Android [C]// SPSM'11: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. New York: ACM, 2011: 15 - 26.

[6] HINTON G E, OSINDERO S, TEH Y W. A fast learning algorithm for deep belief nets [J]. Neural Computation, 2006, 18(7): 1527 - 1554

[7] SALAKHUTDINOV R, HINTON G E. Deep Boltzmann machines [C]// Proceedings of the 12th International Conference on Artificial Intelligence and Statistics. [S. l.]: AISTATS, 2009: 448 - 455.

[8] 孙志军, 薛磊, 许阳明, 等. 深度学习研究综述[J]. 计算机应用研究, 2012, 29(8): 2806 - 2810. (SUN Z J, XUE L, XU Y M, et al. Overview of deep learning [J]. Application Research of Computers, 2012, 29(8): 2806 - 2810.)

[9] 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.

[10] 陈宇, 郑德权, 赵铁军. 基于 Deep Belief Nets 的中文名实体关系抽取[J]. 软件学报, 2012, 23(10): 2572 - 2585. (CHEN Y, ZHENG D Q, ZHAO T J. Chinese relation extraction based on deep belief nets [J]. Journal of Software, 2012, 23(10): 2572 - 2585.)

[11] HINTON G E, SEJNOWSKI T J. Learning and relearning in Boltzmann machines [M] // Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Cambridge, MA: MIT Press, 1986: 282 - 317.

[12] HINTON G E. A practical guide to training restricted Boltzmann machines [R]. Toronto: University of Toronto, Machine Learning Group, 2010.

[13] HINTON G E. Training products of experts by minimizing contrastive divergence [J]. Neural Computation, 2002, 14(8): 1771 - 1800.

[14] CARREIRA-PERPINAN M A, HINTON G E. On contrastive divergence learning [C]// Proceedings of the 2005 Tenth International Conference on Artificial Intelligence and Statistics. [S. l.]: AISTATS, 2005: 33 - 40.

This work is partially supported by the National Natural Science Foundation of China (61271252).

SU Zhida, born in 1992, M. S. candidate. His research interests include Android reverse, Android security detection.

ZHU Yuefei, born in 1962, Ph. D., professor. His research interests include number theory for computing, cryptography, information security.

LIU long, born in 1983, M. S., lecturer. His research interests include vulnerability mining, mobile security.

(上接第 1643 页)

[22] DHANABAL L, SHANTHARAJAH S P. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms [J]. International Journal of Advanced Research in Computer and Communication Engineering, 2015, 4(6): 446 - 452.

[23] HINTON G E. A practical guide to training restricted Boltzmann machines [M]// Neural Networks: Tricks of the Trade, LNCS 7700. Berlin: Springer, 2012: 599 - 619.

[24] 张春霞, 姬楠楠, 王冠伟. 受限波尔兹曼机[J]. 工程数学学报, 2015, 32(2): 159 - 173. (ZHANG C X, JI N N, WANG G W, et al. Restricted Boltzmann machine [J]. Chinese Journal of Engineering Mathematics, 2015, 32(2): 159 - 173.)

[25] 邱龙金, 贺昌政. 神经网络稳定性的交叉验证模型[J]. 计算机工程与应用, 2010, 46(34): 43 - 45. (QIU J L, HE C Z. Cross validation model for stability of neural networks [J]. Computer Engineering and Applications, 2010, 46(34): 43 - 45.)

[26] 范永东. 模型选择中的交叉验证方法综述[D]. 太原: 山西大学, 2013: 19 - 41. (FAN Y D. A summary of cross-validation in model selection [D]. Taiyuan: Shanxi University, 2013: 19 - 41.)

[27] CHANG C C, LIN C J. LIBSVM : a library for support vector machines [J]. ACM Transactions on Intelligent Systems and Technology, 2011, 2(3): 389 - 396.

[28] 贺其备. 基于支持向量机的入侵检测研究[D]. 长春: 东北师范大学, 2013: 29 - 44. (HE Q B. Research on intrusion detection based on support vector machine [D]. Changchun: Northeast Normal University, 2013: 29 - 44.)

This work is partially supported by the Science and Technology Research Project of Liaoning Education Department (LJYL052).

CHEN Hong, born in 1967, M. S., associate professor. Her research interests include information security.

WAN Guangxue, born in 1992, M. S. candidate. Her research interests include information security, deep learning.

XIAO Zhenjiu, born in 1968, M. S., associate professor. His research interests include information security.