doi:10.11835/j.issn.1000-582X.2020.09.006

一种基于系统行为序列特征的 Android 恶意代码检测方法

杨吉云[a],陈　钢[a],鄢　然[a],吕建斌[b]
(重庆大学 a.计算机学院;b. 期刊社, 重庆 400044)

摘要:基于行为特征建立机器学习模型是目前 Android 恶意代码检测的主要方法,但这类方法的特征集中各行为特征相互独立,而行为特征间的顺序关系是反映恶意行为的重要因素。为了进一步提高检测准确率,提出了一种基于系统行为序列特征的 Android 恶意代码检测方法。该方法提取了程序运行发生的敏感 API 调用、文件访问、数据传输等系统活动的行为序列,基于马尔科夫链模型将系统行为序列转换为状态转移序列并生成了状态转移概率矩阵,将状态转移概率矩阵和状态发生频率作为特征集对 SAEs 模型进行了学习和训练,最后利用训练后的 SAEs 实现了对 Android 恶意代码的检测。实验结果证明,提出的方法在准确率、精度、召回率等指标上优于典型的恶意代码检测方法。

关键词:Android 应用;恶意代码检测;动态分析;深度学习
中图分类号:TP309　　**文献标志码**:A　　**文章编号**:1000-582X(2020)09-054-10

An android malware detection method based on system behavior sequences

YANG Jiyun[a], *CHEN Gang*[a], *YAN Ran*[a], *LYU Jianbin*[b]
(a. College of Computer Science; b. Journals Department, Chongqing University, Chongqing 400044, P. R. China)

Abstract: At present, behavior features of machine learning based Android malicious code detecting approaches are independent from each other, whereas the sequential relationships between behavior features could indicate malicious behavior. In order to furtherly improve the detection accuracy, an Android malicious code detection method based on the features of system behavior sequence was proposed. Firstly, the sequences of system activities including sensitive API calls, file access, data transmission, etc. were extracted. Next, based on Markov chain model the system behavior sequences were transformed into state transition sequence, and state transition probability matrix were created. Then, the state transition probability matrix and the state occurrence frequency were used as feature sets to train the SAEs model. Finally, we examined the performance of the trained SAEs model on a dataset. The experimental results show that the proposed method performed better than the typical malicious code detection method on accuracy, precision and recall.

Keywords: Android applications; malicious code detection; dynamic analysis; deep learning

收稿日期:2020-00-00
基金项目:重庆市技术创新与应用发展专项(CSTC2019jscx-msxm0341)。
Supported by Technological Innovation and Application Projects of Chongqing(CSTC2019jscx-msxm0341).
作者简介:杨吉云(1975—),男,副教授,博士,主要从事密码分析、恶意代码检测、大数据分析与处理等方面的研究工作,(E-mail) yangjy@cqu.edu.cn。

随着智能通信技术的不断发展，移动智能终端普及率大大增长，智能手机已经成为人类社会重要的工具之一。根据 Gartner 的最新数据，在智能手机操作系统市场，谷歌的 Android 系统在 2018 年市场份额增至 84.1%[1]。而随着 Android 操作系统的市场份额持续增高，且平台具有开放性等特点，使得越来越多的 Android 智能终端成为攻击目标。

Android 应用程序获取的途径包括 Google Play 和其他第三方 Android 市场，攻击者通常会在一些热门的 Android 应用中插入恶意代码，或者将含有恶意代码的软件伪装成正常的软件，在安全管理较差的应用市场进行发布和传播[2]，因此需要高准确率，低误报率的恶意代码检测方法对 Android 应用进行检测。目前恶意代码检测的方法主要有基于特征签名匹配的检测和基于行为特征的检测[3]。基于特征签名匹配的检测是指将待测应用的签名与已知的恶意代码的签名进行匹配，这种方法可以检测出签名库中存在的恶意代码，但缺点是不能有效检测出未知恶意代码[4]。而基于行为的检测方法则可细分为静态分析方法和动态分析方法，静态分析方法首先对 APK 文件进行反编译，然后通过分析反编译代码的特征来检测恶意代码，而动态分析则通过分析代码在执行过程中的行为特征来检测恶意代码。

在基于行为特征的检测方法中，典型特征主要有权限 、API 调用以及程序运行时发生的系统调用等信息，如文献[5-15]根据这些特征建立了恶意代码检测模型，能有效的检测出恶意代码。但目前这类检测方法在建立检测模型时各行为特征相互独立，没有将各行为特征间的顺序关系作为建模参数，而此顺序关系是反映代码行为的重要因素。为了进一步提高恶意代码检测的准确性，提出了一种状态转移和深度学习相结合的 Android 恶意代码检测算法，该算法基于马尔可夫模型建立了行为及其相互关系的特征向量，使用栈式自动编码器进行训练，实现了对恶意代码有效检测。

1　相关工作

目前基于行为特征的 Android 恶意代码分析主要有静态分析和动态分析 2 类[4]，静态分析通过对反编译代码的特征进行分析并建立检测模型，如 Sahs J 等人在文献[5]中提出了使用 Androguard 提取 APK 文件的权限和控制流图，使用 SVM 进行训练并建立检测模型。Wu S 等人在文献[6]使用与敏感数据流相关的 API 调用的频率作为特征，通过优化计算相邻节点距离的算法，改进了 KNN 分类模型，进一步提高分类器的准确率。Wang W 等人在文献[7]中提取了 APK 文件的权限、敏感 API 调用、代码相关信息以及硬件信息等特征，使用集成学习的方法对 Android 应用进行恶意代码检测。Zhu H J 等人在文献[8]中则使用了权限、监听系统事件、敏感 API 以及权限率作为特征，使用旋转森林来进行恶意代码检测。Nix R 等人在文献[9]中使用 Androguard 获取了 APK 文件的 smali 代码，从中提取出所有可能的程序执行路径，并生成 API 调用序列，使用 CNN 进行训练和检测。Li W 等人在文献[10]中提取了应用的权限、敏感 API 函数调用、以及 Kirin[11]中的 9 种权限组合作为特征，使用 DBN 进行训练，实现了恶意软件的检测。

静态分析方法不需要运行恶意代码，具有速度快、对系统资源要求低等优点。但如果恶意代码运用了代码混淆、加壳等反静态分析技术，反编译代码的特征难以准确提取，由此将严重降低检测的准确率。

为了有效的克服静态分析的这个缺陷，研究人员提出通过分析代码在执行过程中的行为特征来检测恶意代码，如 Wu W C 等人在文献[12]使用 APIMonitor 和 DroidBox 来获取了 Android 应用运行时发生的系统事件和敏感的 API 调用，并使用 SVM 对其进行训练和检测。在文献[13]中，Canfora G 等人收集了 Android 程序运行时的系统调用序列，计算固定长度的系统调用序列的互信息，并选择互信息最大的前 k 个系统调用序列作为特征，利用 SVM 来训练分类器实现了恶意代码的检测。文献[14]中，Yeh C W 等人则是在文献[12]的基础上进行改进，将收集到的系统事件和敏感 API 调用按发生的时间先后顺序排列，组成矩阵模型，然后使用 CNN 进行特征学习和分类。孙承庭等人在文献[15]中提出了一个使用 API 调用和系统调用以及权限调用作为特征，并使用机器学习算法进行恶意代码检测的动态检测系统。Xiao X 等人在文献[16]中使用 monkey 和 strace 获取了 Android 应用的系统调用序列，并使用 LSTM 作为分类器进行了恶意代码的检测。

由以上分析可知目前基于行为分析的方法主要选取权限、API 调用、系统调用等特征，在此基础上建立了有效的恶意代码检测模型。但在构建特征模型时各行为特征相互孤立，比如将恶意代码中是否使用某个

权限、API调用、系统调用等作为特征，或者是使用发生的API调用和系统调用的频率作为特征，而忽略了这些特征之间的相互关系，尤其是这些特征发生的先后顺序。由于行为特征序列反映了代码的行为，因此很可能成为决定应用是否是恶意软件的重要因素，比如，对于一个API调用的集合，其中的每个API函数都在一段程序中进行了调用，但是当且仅当这些API按照一定的顺序进行调用时才表现出恶意行为。因此，结合系统行为序列的特征模型更能准确的反映恶意代码行为，提出了一种基于系统行为序列特征的恶意代码检测算法。

2 基于行为序列的检测方法

2.1 特征选择

目前在基于行为特征的Android恶意代码的检测方法中，研究人员认为某些敏感的函数被调用或者某些权限被使用时，恶意软件就可能发生了恶意行为，因此使用权限 、API调用以及系统调用等是否发生作为分类特征可以有效检测出恶意代码[17]。但有些行为特征在恶意软件和非恶意软件中都存在，只有在执行某些恶意行为时可能会产生特定的系统行为序列，因此特征间的顺序关系也是区分恶意软件和正常软件的重要信息[18]。例如有的恶意扣费的Android应用和一些正常Android应用都有发送SP扣费短信的行为，但它们发送SP扣费短信的前后系统行为序列不同。如图1所示。

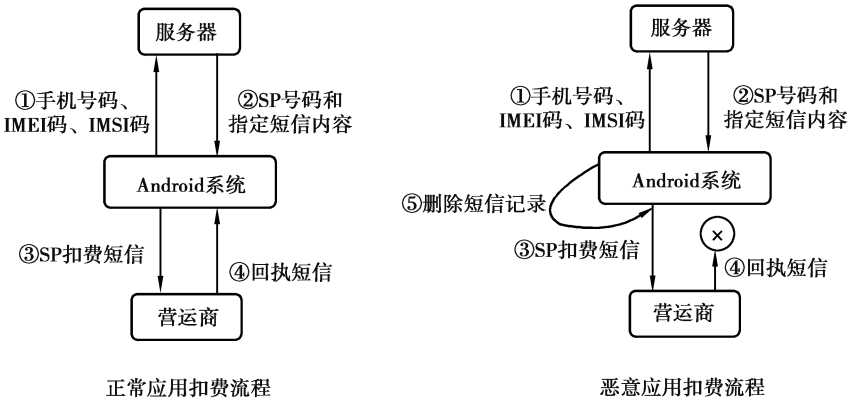

图1 良性应用与恶意应用扣费流程对比

Fig. 1 Deduction Process Comparing of Benign App and Malicious App

由图1可以看出，恶意软件和正常软件都会上传用户手机号码、IMEI码、IMSI码等固件信息到服务器，请求获取SP号码和指定短信内容，然后发送SP扣费短信，订购某收费业务。但是在订购收费业务并接收到回执短信后，恶意程序会立刻删除发送的SP短信运营商的回执短信，而正常的Android应用却不会。对比两个应用的扣费流程会发现恶意应用只比正常应用多了一个删除短信记录的步骤，但仅仅根据是否发生删除短信记录这一行为来区分恶意应用和正常应用是不够准确的。因为正常的应用也可能发生删除短信的行为，只是出现的时机不同。因此对于发生了扣费行为的应用，判断其是否是恶意软件不能仅仅根据该应用是否发生了发送SP扣费短信、删除短信等行为，还要根据收到回执短信之后是否立刻产生了删除短信的行为。

由此可见，仅仅根据某些敏感API调用或者些敏感行为是否发生来判断是否是恶意行为是不够准确的，因此希望能够利用Android应用的系统调用序列或API调用序列等系统行为序列来检测Android恶意代码[19]。

选择Android应用程序运行时的系统行为序列作为特征，系统行为序列中包括敏感API调用和各种系统活动。对于敏感API的选取，使用的方法为收集样本中的所有API调用作为特征，用SVM算法对样本进行训练，进行恶意软件和正常软件的分类，通过训练将得到每个API调用的权重ω，ω的绝对值越大表示对应的API越重要，将ω按照绝对值大小排序，选择前k个API调用作为需要监控的敏感API。系统活动包括传输网络数据、读写文件、泄露敏感数据、发送短信、通话、启动服务以及动态加载等等。

2.2　基于马尔可夫链的特征模型

马尔可夫链表示的是状态空间中从一个状态到另一个状态转换的随机过程，过程中下一状态的概率分布只由当前状态决定。假设状态序列为$\cdots x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2}\cdots$，则$x_{t+1}$时刻的状态只与$x_t$有关，如式(1)所示

$$P(x_{t+1} \mid \cdots x_{t-2}, x_{t-1}, x_t) = P(x_{t+1} \mid x_t)。\tag{1}$$

在马尔可夫链的每一步中系统根据概率分布，可以从一个状态转移到另一个状态，也可以保持当前状态，在状态之间发生改变的概率称为转移概率，由转移概率组成的矩阵称为状态转移概率矩阵。

由于 Android 应用在 $t+1$ 发生的行为只与 t 时刻发生的行为相关，而 t 时刻以前发生的行为可以看做是通过影响 $t-1$ 时刻的行为来间接影响 t 时刻的行为，与 t 时刻发生的行为不直接相关。因此可以将系统行为变化的随机过程近似地看作一个马尔可夫链(markov chain)，进而可用马尔可夫链来对 Android 程序的行为进行描述。

将 Android 程序的系统行为按照表 1 的方法一一映射，得到 Android 应用的状态概率转移序列。

表 1　系统行为与状态映射方法

Table 1　Mapping Method of System Behavior and State

类别	描述	映射方法
资源操作类	包括对文件的访问，外部加载程序，启动服务等	activityname[base-path-x]，activityname 表示系统活动的名字，如文件访问为 Fdaccess，x 表示被裁剪到的深度，当 $x=2$ 时，对路径/dev/input/event0 的文件访问将被映射到状态 fdaccess /dev/
数据传输类	包括敏感数据的泄露，网络数据的传输等	activityname [tags]，如数据泄露中，tags 表示泄漏数据的信息类型。例如，存储电话 IMEI 的网络数据的 tags 为 Network TAINT_IMEI。
敏感 API 调用	表示我们进行标记的敏感 API 的调用情况	直接表示为调用 API 的名字
其他系统活动	包括打开网络连接，发生短信等行为	直接表示为发生的系统活动的名字，如打开网络连接表示为 opennet

以从 drebin 数据库中获得的包名为 breakingdawn.countdown.iconosys.eng 的恶意应用为例，通过修改后的 DroidBox 工具从该应用中获得如图 2 所示的 json 数据，其中的数字如"1.5877950191497803"对应发生事件的时间戳。由此可得到如下的状态序列：sendsms ✍ opennet ✍ fdaccess /dev/ ✍ opennet ✍ sendsms ✍ opennet。

```
"sendsms": {
    "1.5877950191497803": {...},
    "9.8877950191497803": {...}
},
"fdaccess": {
    "4.230021953582764": {"path": "/dev/",...},
},
"opennet": {
    "3.340162992477417": {...},
    "9.754840850830078": {...},
    "10.697690963745117": {...}
},
```

图 2　状态日志示例图

Fig. 2　Example of State Log

根据以上序列生成的马尔可夫链模型如图3所示。

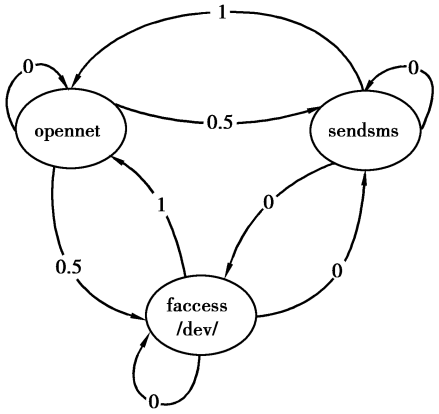

图3 马尔可夫链模型

Fig. 3 Markov chain model

则它的状态转移概率矩阵为

$$\boldsymbol{P}=\begin{bmatrix}0 & 1 & 0\\ 0.5 & 0 & 0.5\\ 0 & 1 & 0\end{bmatrix}。\tag{2}$$

由此获得的状态转移概率矩阵就可以用来表示行为序列的统计特性,从而进一步的应用到机器学习模型中。

2.3 特征向量的生成

使用的特征矩阵由Android应用运行过程中各个状态发生的频率以及其状态转移概率矩阵构成。

通过在模拟器中运行程序可以得到Android应用运行时的系统行为序列,按照表1将序列中的动作逐一映射到对应的状态中,即可获得状态转移序列。

假设所有的行为序列可以映射到N个状态中,则状态集合为$\{S1,S2,\cdots Sk,\cdots Sn\}$,第$i$个样本的各个状态发生的频率可以表示为$fi_{S1},fi_{S2},\cdots fi_{Sk},\cdots fi_{Sn}$,频率$fi_{Sk}$的计算如式3所示,其中$T_k$表示状态$k$发生的次数

$$fi_{Sk}=\frac{T_k}{\sum_{i=1}^{n}T_i}。\tag{3}$$

根据得到的状态转移序列,能够从中统计状态发生频率和生成该状态序列对应的状态转移概率矩阵。生成状态转移概率矩阵的算法如图4所示,输入为状态转移序列SSeq和状态字典SDict,其中SDict记录了映射得到的所有状态以及状态对应的唯一编号。

通过图5的算法,输出了SSeq对应的状态概率转移矩阵**SPM**,第k个样本的**SPM**如图3所示:

由此得到的状态转移概率矩阵的大小为$n\times n$,如图6所示,将状态转移概率矩阵转换为一维向量,与各个状态的频率组合成特征,即最后的特征向量的长度为n^2+n。

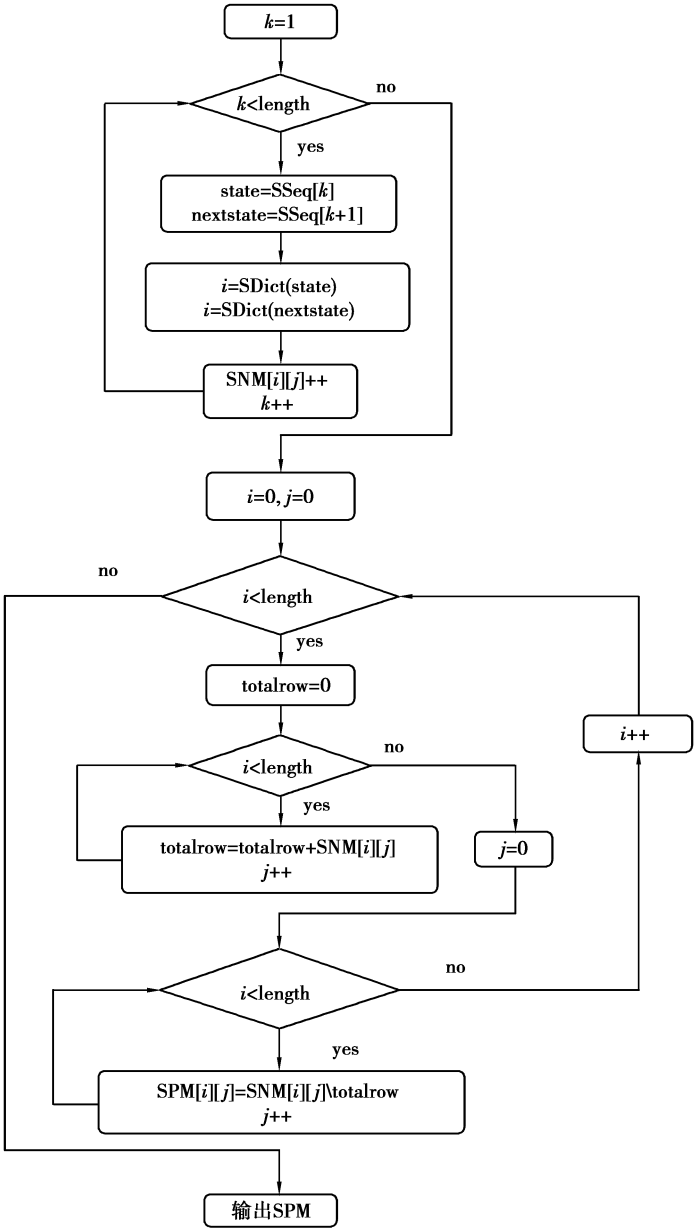

图 4　获取状态概率矩阵流程图

Fig. 4　Flow Chart of State Probability Matrix Acquisition

$$\begin{pmatrix} p^k_{11} & \cdots & p^k_{1j} & \cdots & p^k_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ p^k_{i1} & \cdots & p^k_{ij} & \cdots & p^k_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ p^k_{n1} & \cdots & p^k_{nj} & \cdots & p^k_{nm} \end{pmatrix}$$

图 5　状态转移概率矩阵

Fig. 5　State Transition Probability Matrix

2.4　训练和检测

目前用于恶意代码检测的机器学习算法主要是基于浅层学习结构的分类方法，如支持向量机、人工神经网络、朴素贝叶斯和决策树等。而深层学习具有层次特征学习模型的结构，可以从大量的特征中学习到更深层的表达，从而实现比传统机器学习算法更好的性能。因此，将使用深度学习算法 Stacked AutoEncoder

(SAE)建立训练和检测模型。

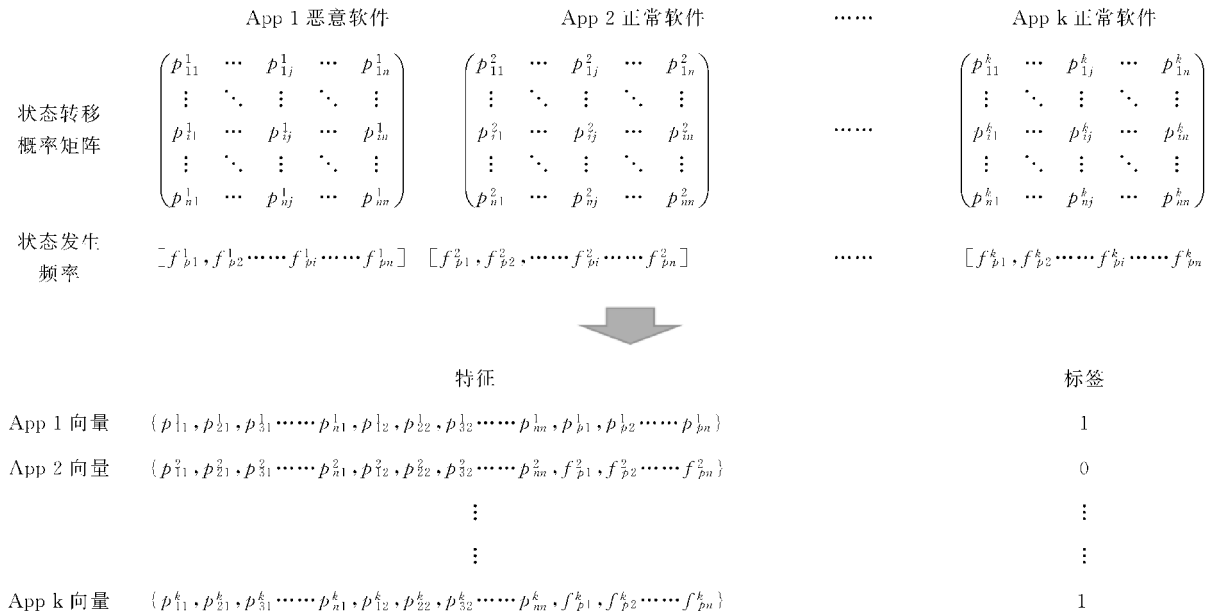

图 6 特征向量

Fig. 6 Feature Vector

SAE 模型由多层自动编码器构建而成,其中自动编码器是一种用于学习高效编码的人工神经网络,具有输入层,输出层,以及一个或多个隐层。图 7 表示了一个只有一层隐层的自动编码器,当数据通过这样的网络时,首先会被编码成一个较小的向量,然后再将这个向量进行解码。编码器训练的任务就是尽可能的减少输入层和输出层的误差,如式 4 所示,即为输入数据寻找一个最优的紧凑表示,如式 5 所示。

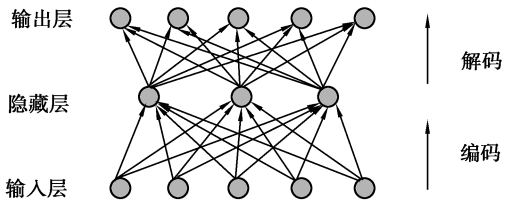

图 7 自动编码器

Fig. 7 Automatic Encoder

$$E(\boldsymbol{x},\boldsymbol{z})=\frac{1}{2}\sum_{i=1}^{n}\parallel \boldsymbol{x}_i-\boldsymbol{z}_i\parallel^2, \tag{4}$$

其中 $\boldsymbol{x}$ 是输入向量,$\boldsymbol{z}$ 是输入空间中重构的多维向量,n 是训练样本数。

$$\theta=\{w,b\}=\underset{\theta}{arg\,\min}E(\boldsymbol{x},\boldsymbol{z}) \tag{5}$$

其中 $\boldsymbol{w}$ 是连接权重矩阵,$\boldsymbol{b}$ 是偏移向量,θ 是映射参数集 $\theta=\{\boldsymbol{w},\boldsymbol{b}\}$。

通过堆叠多个自动编码器构建深度学习网络,即形成栈式自动编码器,其中当前层的一个自动编码器的输出将作为下一个自动编码器的输入。图 8 表示一个基于栈式自动编码器的检测模型,将训练集的特征向量作为第一层输入,将其训练成一个自动编码器 A1,然后将训练好的自动编码器的输出作为下一个自动编码器 A2 的输入,再对 A2 进行训练,重复这一过程直到隐层数达到 h,即获得了一个具有 h 层深度的 SAEs 模型。在 SAEs 模型上再添加 softmax 层,计算 softmax 层的输出与实际标签的误差,使用反向传播的方法进行整体调整,最终组成一个具有深层体系结构的检测模型。

在检测阶段,将待测应用使用研究方法提取特征,将特征向量作为训练好的 SAEs 检测模型的输入,SAEs 的输出即为该 Android 应用的分类结果。

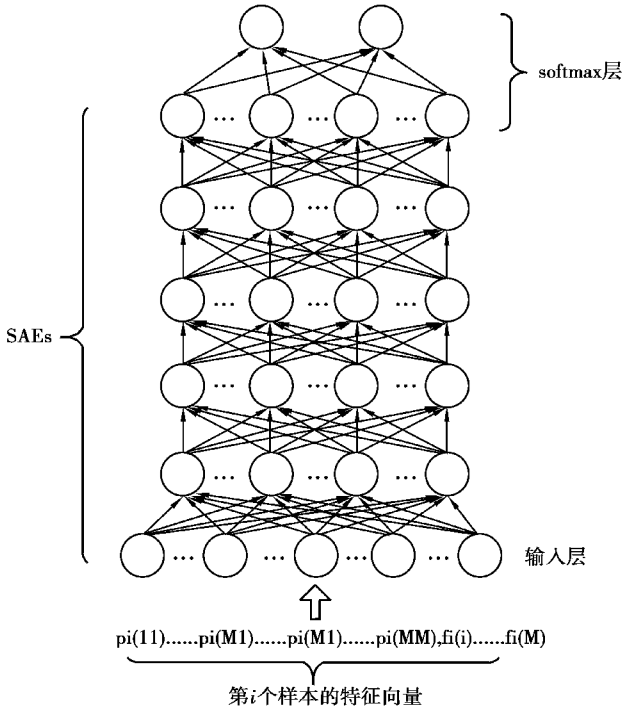

图 8　SAEs 检测模型

Fig. 8　Detection Model Constructed with SAEs

3　实　验

为了验证提出检测方法的有效性，将通过实验评估方法的性能，以及与其他典型的 Android 恶意代码检测方法的对比分析。实验环境为处理器 Intel(R) Core(TM)3 i5-3470 @ 3.20GHz，内存 8GB，使用 Python 编程实现 APK 的捕获和动态行为特征的提取，在 Matlab 实现分类器的训练及测试。笔者选取了 1 000 个恶意软件和 1 000 个非恶意软件作为数据集，恶意软件来源于使用 Drebin 数据库中，非恶意软件来源于安卓应用市场。对每一个样本进行了 5 次模拟实验，每次运行时间为 5 min，总共收集到 1 315 种不同的状态，经过对稀疏状态的剔除，最后得到 238 种状态，其中 API 调用为 124 个，系统活动为 114 个。

3.1　评估指标

使用混淆矩阵来分析实验结果，将恶意应用定义为正元组，非恶意应用定义为负元组。混淆矩阵如表 2 所示，其中，TP 表示恶意应用被正确识别为恶意应用的数量，FN 表示恶意应用被错误识别为正常应用的数量，FP 表示正常应用被错误识别为恶意应用的数量，TN 表示正常应用被正确识别为正常应用的数量。

表 2　混淆矩阵

Table 2　Confusion Matrix

参量	预测为恶意	预测为正常
实际为恶意	TP	FN
实际为正常	FP	TN

根据混淆矩阵可以计算分类器的评估参数，选用的评估参数如下

$$\text{Accuracy}=\frac{TP+TN}{TP+TN+FP+FN}, \tag{6}$$

$$\text{Precision}=\frac{TP}{TP+FP}, \tag{7}$$

$$\text{Recall}=\frac{TP}{TP+FN}, \tag{8}$$

$$F-Measure=\frac{2\times Precision\times Recall}{Precision+Recall}。\tag{9}$$

3.2 性能分析

在这组实验中，利用2.3中描述的特征集，评价了SAEs在不同参数下的性能。每次使用800个恶意样本和800个正常样本作为训练集，剩下的200个恶意样本和200个正常样本作为测试集，重复进行10次实验并取实验结果的平均值作为最终的检测结果。在构建深度学习模型时，有2个关键参数：隐层的数量和每个隐层中的神经元数量。表3显示了提出的检测模型在不同网络结构下的性能变化。从表3可以看出，在隐层层数为4，每层神经元个数为[5 000,5 000,5 000,1 000]时性能达到了最佳，准确率为93.22%，精度为94.397%，召回率为91.94%，F1度量也达到了93.135%。

表3 不同网络结构的分类结果

Table 3 Classification Results with Different Model Constructions %

网络结构	Accuracy	Precision	Recall	F-Measure
[1 000,1 000,1 000]	91.61	91.824	91.420	91.585
[5 000,5 000,1 000]	91.82	91.239	92.60	91.882
[5 000,5 000,500]	92.16	93.187	91.00	92.070
[5 000,5 000,5 000,1 000]	93.22	94.397	91.94	93.135
[5 000,2 500,2 500,500]	92.80	92.274	93.56	92.862
[5 000,5 000,5 000,5 000,1 000]	92.52	93.021	92.06	92.487

3.3 检测效果对比

为了验证提出方法的检测效果，将数据集进行划分，使用其中的800个恶意样本和800个正常样本作为训练集，剩下的200个恶意样本和200个正常样本作为测试集进行实验。使用典型的Android恶意代码检测工具Androguard对测试集进行检测，并将结果与使用研究方法进行实验的结果进行对比，对比结果如图9所示。

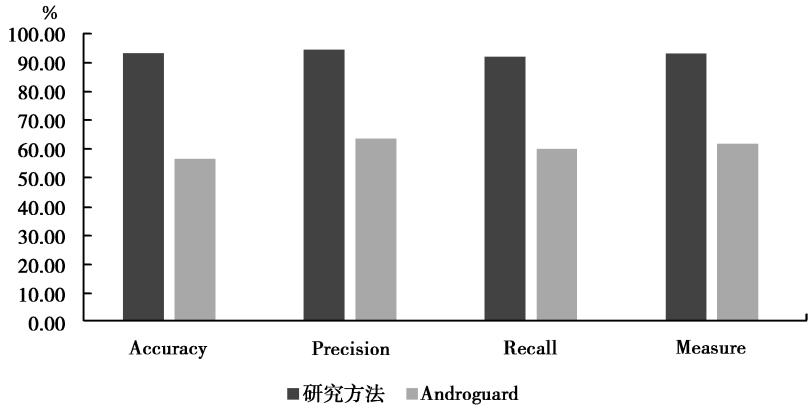

图9 检测结果对比

Fig. 9 Compare Detection Results

实验结果表明提出的方法在准确率、精度、召回率和F1度量等性能指标上均优于Androguard。

4 总 结

提出了一种基于行为序列的Android恶意代码检测方法，首先从Android应用中提取了程序运行时的系统行为序列，包括Android应用运行时发生的敏感API调用、文件访问、数据传输等系统活动。基于马尔科夫链模型将系统行为序列转换为状态转移序列，然后生成了状态转移概率矩阵。将状态转移概率矩阵和状态发生频率作为特征集对SAEs模型进行了学习和训练，最后利用训练后的SAEs实现了对Android恶意代码的检测。实验结果表明，提出方法在准确率上优于典型的恶意代码检测方法。在下一步的工作中，将更加深入地研究Android恶意样本特征，进一步提高检测算法的准确率。

参考文献:

[1] Gartner. Gartner says worldwide sales of smartphones grew 9 percent in first quarter of 2017[EB/OL]. https://www.gartner.com/newsroom/id/3725117.

[2] 张玉清，王凯，杨欢，等. Android 安全综述[J]. 计算机研究与发展，2014，51(7)：1385-1396.
ZHANG Yuqing，WANG Kai，YANG Huan，et al. Survey of android OS security[J]. Journal of Computer Research and Development，2014，51(7):1385-1396.(in Chinese)

[3] 陈铁明，杨益敏，陈波. Maldetect：基于 Dalvik 指令抽象的 Android 恶意代码检测系统[J]. 计算机研究与发展，2016，53(10)：2299-2306.
CHEN Tieming，YANG Yimin，CHEN Bo，et al. Maldetect：an android malware detection system based on abstraction of dalvik instructions[J]. Journal of Computer Research and Development，2016，53(10)：2299-2306.(in Chinese)

[4] Yan P，Yan Z. A survey on dynamic mobile malware detection[J]. Software Quality Journal，2018，26(3)：891-919.

[5] Sahs J，Khan L. A machine learning approach to android malware detection[C]// 2012 European Intelligence and Security Informatics Conference. Piscataway，NJ：IEEE，2012：141-147.

[6] Wu S Y，Wang P，Li X，et al. Effective detection of android malware based on the usage of data flow APIs and machine learning[J]. Information and Software Technology，2016，75：17-25.

[7] Wang W，Li Y Y，Wang X，et al. Detecting android malicious apps and categorizing benign apps with ensemble of classifiers[J]. Future Generation Computer Systems，2018，78：987-994.

[8] Zhu H J，You Z H，Zhu Z X，et al. DroidDet：effective and robust detection of android malware using static analysis along with rotation forest model[J]. Neurocomputing，2018，272：638-646.

[9] Nix R，Zhang J. Classification of android apps and malware using deep neural networks[C]// 2017 International Joint Conference on Neural Networks (IJCNN). Piscataway，NJ：IEEE，2017：1871-1878.

[10] Li W J，Wang Z，Cai J C，et al. An android malware detection approach using weight-adjusted deep learning[C]// 2018 International Conference on Computing，Networking and Communications (ICNC). Piscataway，NJ：IEEE，2018：437-441.

[11] Enck W，Ongtang M，McDaniel P. On lightweight mobile phone application certification[C]// Proceedings of the 16th ACM Conference on Computer and Communications Security - CCS '09. New York，USA：ACM Press，2009：235-245.

[12] Wu W C，Hung S H. Droid Dolphin：a dynamic android malware detection framework using big data and machine learning [C]// Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems. New York，USA：ACM Press，2014：247-252.

[13] Canfora G，Medvet E，Mercaldo F，et al. Detecting android malware using sequences of system calls[C]// Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile-DeMobile 2015. New York，USA：ACM Press，2015：13-20.

[14] Yeh C W，Yeh W T，Hung S H，et al. Flattened data in convolutional neural networks[C]// Proceedings of the International Conference on Research in Adaptive and Convergent Systems - RACS '16. New York，USA：ACM Press，2016：130-135.

[15] 孙承庭，吴凯娇，马文海. 基于特征分析 Android 恶意应用检测方法的研究[J]. 南京邮电大学学报(自然科学版)，2016，36(4)：113-118.
SUN Chenting，WU Kaiqiao，MA Wenhai. Android malware detection method based on characteristic analysis[J]. Journal of Nanjing University of Posts and Telecommunications(Natural Science)，2016，36(4)：113-118.(in Chinese)

[16] Xiao X，Zhang S F，Mercaldo F，et al. Android malware detection based on system call sequences and LSTM[J]. Multimedia Tools and Applications，2019，78(4)：3979-3999.

[17] Peiravian N，Zhu X Q. Machine learning for android malware detection using permission and API calls[C/OL]. 2013 IEEE 25th International Conference on Tools with Artificial Intelligence. Piscataway，NJ：IEEE，2013(2014-04-10)[2020-05-25]. https://doi.org/10.1109/ICTAI.2013.53.

[18] Mariconti E，Onwuzurike L，Andriotis P，et al. MaMaDroid：detecting android malware by building markov chains of behavioral models[C/OL]. Proceedings 2017 Network and Distributed System Security Symposium. Reston，VA：Internet Society，2017(2017-04-27)[2020-05-25]. https://arxiv.org/abs/1711.07477.

[19] 赵艳丽，孙志挥. 基于静态马尔可夫链模型的实时异常检测[J]. 计算机应用，2004，24(11)：30-32.
ZHAO Yanli，SUN Zhihui. Real-time anomaly detection based on the static makov chain model[J]. Journal of Computer Applications，2004，24(11)：30-32.(in Chinese)

(编辑　侯　湘)