

基于 HTM 算法的恶意 Android 应用检测

仇慎健^a, 张仕斌^a, 刘苹光^b

(成都信息工程大学 a. 信息安全工程学院;b. 通信工程学院, 成都 610225)

摘要:随着互联网用户从传统 PC 端到移动端的转换,移动安全受到越来越多的关注。为了提高对未知恶意移动应用的检测效率,针对传统检测对引入多态和变形技术的恶意应用检测能力较差的问题,提出了一种基于 HTM 算法的恶意 Android 移动应用检测方法。该应用检测包含针对 Android 应用 Dalvik 指令特点的特征提取、采用信息增益的方式进行特征选择与融合,并利用 HTM 算法进行序列模式训练和推导,然后将测试样本特征提取与融合后的结果输入到完成训练的 HTM 网络中,达到检测恶意应用的目的。实验仿真表明,所设计的恶意应用检测方法的检测率接近 100%,检测效率高,误报率 0.08%。相较于其他算法,提出的恶意检测方法的检测率、误报率、分类准确率均更优,并能应用于不同类型的恶意应用,但训练和测试时间较长。

关键词:移动安全;HTM 算法;Dalvik 指令;信息增益;恶意应用;检测

中图分类号:TP393

文献标志码:A

引言

以前,移动端恶意应用主要窃取的是文本信息,如短信、联系人信息、地理位置等。随着攻击者攻击技术的逐渐提升,恶意应用进行环境录音、通话录音、读取 sdcard 内容的攻击方式也逐渐出现。随着对反病毒引擎的免杀能力逐渐增强,恶意应用通过逆向对抗、加壳、Rootkit 等技术逃避杀毒软件的查杀。传统基于规则和特征的检测技术难以适应引入多态和变形技术的恶意代码检测,于是出现了基于深度学习算法 HTM (Hierarchical Temporal Memory) 的恶意移动应用检测算法。深度学习算法 HTM 是一种新型的机器学习算法,其核心思想由 Jeff Hawkins 在其 2004 年出版的著作《人工智能的未来》中提出,它以模仿新大脑皮层的处理信息特性为核心,有着高度的学习、识别和预测能力。2005 年 George 与 Jeff 介绍了 HTM 的基本模

型^[1],利用 Bayesian Belief Propagation 原理解释分类和重建的过程,同时把算法部分原理通过生物学现象进行解释。2009 年 Jeff 和 George 对 HTM 算法^[2]进行了深入的研究,阐述了 HTM 算法序列记忆的约束条件以及序列模型在生物学上的意义和具体的算法实现过程。2010 年 George 提出了皮质学习的主题条件^[3],并且提出了 HTM 的改进算法——Recursive Cortical Network (RCN) 算法^[4],该算法是在原 HTM 的各层的相邻节点 nodes 之间加入了连接 (connections)。2014 年,Rehn 等人^[5]提出了针对 HTM 算法的 Incremental Learning 方法。在国内,针对 HTM 算法的研究也只是停留在理论层面。例如,2011 年李元诚、樊庆君提出将 HTM 算法应用于未知恶意代码检测^[6],该算法利用字节级 n 元文法提取训练集中文件的特征向量,构建 HTM 网络进行序列模式学习训练和分类推导,然后将测试集中提取的特征向量输入到完成训练的 HTM 网

收稿日期:2015-06-08

基金项目:四川省科技支撑计划项目(2013GZX0137;2014GZ0002);成都市科技攻关项目(2014-HM01-00108-SF);四川省科技创新研发专项(2014GZ0006)

作者简介:仇慎健(1990-),男,江苏徐州人,硕士生,主要从事计算机取证、网络与信息安全方面的研究,(E-mail)1074271150@qq.com;
张仕斌(1971-),男,重庆人,教授,博士,主要从事计算机取证、信息安全理论及应用方面的研究(E-mail)498251651@qq.com

络进行序列识别。本文在分析 Android 应用的基础上,结合 HTM 算法优势,提出使用基于 Dalvik 指令进行 Android 恶意应用的特征提取,生成二进制稀疏表征,并通过 HTM 网络进行深度学习训练和分类推导,基于此方式进行恶意应用识别,目的是提高对未知恶意移动应用的检出比。

1 移动应用检测系统组成

1.1 特征提取

Dalvik 是由 Google 开发的应用在 Android 平台上的虚拟机。Dalvik 虚拟机运行的是 Dalvik 字节码,所有的 Dalvik 字节码由 java 字节码转换而来,并被打包到一个 DEX(Dalvik Executable)中。可以根据 Dalvik 指令集的特点来提取恶意应用程序的特征,从指令中精简出反应程序语义的指令,精简后的指令仅包括函数调用、赋值、跳转等^[7]。在 Android 恶意应用中,通常使用反射机制调用有关程序,在语义提取过程中忽视调用的方法名,进而可产生一种新型式的描述语言:

```
Procedure: = StatementList;
M: = MOVE|MOVE_WIDE_FROM16;
R: = RETURN|RETURN_OBJECT;
V: = INVOKE_INTERFACE_RANGE.
```

获取到病毒等恶意应用进行反编译后得到 .smali 文件。可以使用 jeb 将 samli 语言转换为 java 语言,或者将恶意应用的核心代码使用 NDK 原生程序去编写,使用 IDA 进行反编译。so 文件获取关键代码,然后找到对应的关键方法,如短信截取、静默安装等恶意行为实现反编译之后的源代码,最后使用新型式描述语言进行定义,如 [IRVM]@[MVRIR]等,可简化存储并且可以摒弃冗余的指令。

1.2 特征选择与融合

因为随机样本进行提取的特征种类较多,但不是所有的特征都具有很强的分类性能,因此需要选择出具有较好分类的特征,本文应用信息增益实施衡量^[8]。假设 Y 代表分类结果的参数, X_i 为与特征 i 对应的参数,那么特征 i 的信息增益可表示为

$$I(Y, X_i) = H(Y) - H(Y | X_i) \quad (1)$$

其中, $H(Y)$ 是参数 Y 的熵, $H(Y | X_i)$ 为在参数 X_i 限定下参数 Y 的条件熵。信息增益 $I(Y, X_i)$ 结果值越大,说明参数 X_i 所对应的特征 i 的分类性能越优。在 HTM 算法中,输入序列为离散表征二进制数,因此可以构建特征集合 S ,每个特征元素即为一段二进制序列,选取信息增益值最大的前 500 个特征构成该集合。集合 S 可以作

为特征融合的原始输入,特征融合的输出即为 HTM 算法序列输入。

下面对特征融合进行说明。假设 U 代表先知分类的代码集合,对 $\forall u \in U, u$ 符合 $u = \langle x_1, x_2, \dots, x_n, y \rangle$, 其中, $x_i, y \in \{0, 1\}, i = 1, 2, \dots, n, x_i = 0$ 代表 u 没有特征 $x_i, y = 0$ 代表 u 为正常程序, $y = 1$ 代表 u 是恶意的程序。令 M 代表 U 中的恶意程序集合, N 代表 U 中的正常程序集合,则有

$$M = \{m | m \in U \wedge y = 1\}$$

$$N = \{n | n \in U \wedge y = 0\}$$

假设 V 代表将要检测的未知恶意程序集合体,对 $\forall v \in V, v$ 符合 $v = \langle x_1, x_2, \dots, x_n, y \rangle$, 其中 $y = 2$ 表示 v 未分类, v 表示抗原,特征 x_1, x_2, \dots, x_n 将融合成危险信号和安全信号。令危险特征集合为 DS , 安全特征集合为 SS , 普通特征集为 PS , 则有

$$DS = \{x_i | \forall u \in U \wedge (x_i = 1 \rightarrow u \in M)\} \quad (2)$$

$$SS = \{x_i | \forall u \in U \wedge (x_i = 1 \rightarrow u \in N)\} \quad (3)$$

$$PS = S - DS - SS \quad (4)$$

1.3 HTM 算法实现过程

1.3.1 HTM 深度学习算法组成及优势

HTM 层次实时记忆算法是一种以捕捉新大脑皮层结构与算法特性为目标的机器学习方法^[9]。新的大脑表层对于高等思维有着举足轻重的位置。生物学研究指出,并不是每一种认知功能都对应一个神经算法,由于新大脑皮层的回路具有很高的统一性,新大脑皮层用一套公用的算法去实现不同功能。该算法是基于流数据的,而不是静态数据库。它可以通过最新的输入来学习、识别和预测。它是一个基于记忆的系统,HTM 网络被大量具有时间特性的数据训练而成,并依赖于大量储存的模式序列。HTM 的核心原理是它的层级组织结构、区域的构建、信息基于时间以及数据的存储方式要以离散稀疏表征存储等。一个 HTM 网络是以按层级排列的区域构成。该区域由一存储和预测单元组成,区域表示该层级中的下级多个子元素会被聚合后形成的存储单元。由于反馈连接的存在,信息也会随着等级的下降不断分流。稀疏离散表征是将自然界语言(如图像,文本,音频等)转换为二进制序列,而且是稀疏的,这是 HTM 的重要基础,所以 HTM 区域所做的第一件事就是将输入转化为稀疏离散表征。

1.3.2 HTM 算法训练及检测

将特征融合后得到的危险特征集以稀疏离散特征的方式存储,并作为输入序列进入 HTM 网络底层节点进行学习,直到底层所有节点的空间沉积池都完成了空

间模式的学习,时间沉积池完成时间分组的学习。HTM的整体层次框架如图1所示。

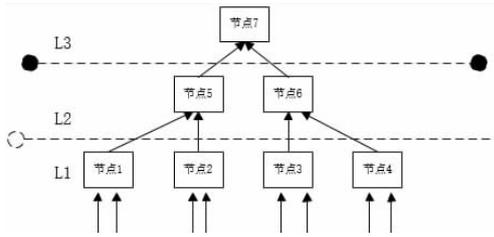


图1 HTM层次框架

HTM具有树状的层级网络结构,节点是HTM中记忆与预测的基本单元。在节点中主要存储三种数据^[10-12],C(coincidences的集合)、G(temporal groups的集合),每个group实际上是coincidences的集合和transition probability matrix(每个group中各个coincidence之间的转移概率组成的矩阵)。HTM对空间模式的提取依赖于父节点对子节点子模式的pooling,而时间模式与序列记忆的实现依赖于节点中不同的temporal groups以及其各coincidences组成的Markov chains。通过记忆不同order的Markov chains,可以由一个coincidences往前或者后推知另外coincidences发生的可能性,从而实现序列记忆。而每个group也是这种coincidences之间转移概率最大化分类,以获得时间相近、模式相似的结果。coincidence模式和Markov链时间分组利用belief propagation完成节点的记忆结构。例如,Markov链时间组的子节点由coincidence模式下的 C_i 定义成向量 $[ri^{m1}, \dots, ri^{mM}]$ 的形式,当 $M=2$ 时,则group2和group5来自于子节点 $m1$ 与 $m2$,则 C_4 的coincidences模式可等价于 $[2,5]$ 。

空间沉积池(spatial pooler)主要功能是将一个区域的输入转换为稀疏模式,因为在学习序列和预测的机制中第一步就要求从稀疏离散表征开始。时间沉积池(temporal pooler)是将空间沉积池中的输出作为输入,然后根据序列模式的时间邻接特性进行离散性存储,它是基于海量学习数据之下的,微量或单一数据的学习起不到效果。单一节点学习如图2所示。

空间沉积的实现包括初始化、覆盖、抑制、学习等几个部分。根据当前的输入计算柱状区域的覆盖情况,在抑制作用完成后计算最终活跃的柱状区域,最后更新突触的联通值和内部变量。完成覆盖抑制过程的伪代码为:

- 1) for k
- 2) $overlap(k) = 0$

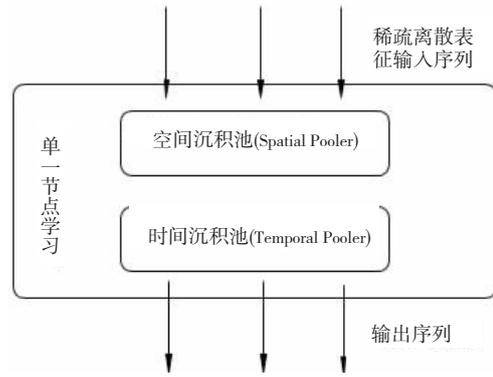


图2 单一节点学习

- 3) for s in connectedSynapses(k)
- 4) $overlap(k) = overlap(k) + input(t, s, sourceinput)$
- 5) if $overlap(k) < minOverlap$ then
- 6) $overlap(k) = 0$
- 7) else
- 8) $overlap(k) = overlap(k) * boost(k)$
- 9) for k in columns
- 10) $minLocalActivity = KthScore(neighbors(k), desiredLocalActivity)$
- 11) if $overlap(k) > 0$ and $overlap(k) > = minLocalActivity$ then
- 12) $activeColumns(t).append(k)$

给定一个输入向量并根据这个向量计算每个柱状区域的覆盖情况。柱状区域的覆盖情况可以简单的理解为与当前激励输入相连的突触数量,然后乘以促进系数。如果值小于 $minOverlap$,将覆盖值($overlap(c)$)置为0,然后从在柱状区域经过抑制作用后作为胜利者留下来, $desirLocalActivity$ 是用来控制最终活跃的柱状区域数量的参数。

根据需要更新突触的连通值以及柱状区域的促进系数和抑制半径完成学习操作,其过程的伪代码描述为:

- 1) for g in ActiveColumns(t)
- 2) for m in PotentialSynapses(g)
- 3) if active(m) then
- 4) $m.permanence + = permanenceInc$
- 5) $m.permanence = min(1.0, m.permanence)$
- 6) else
- 7) $m.permanence - = permanenceDec$
- 8) $m.permanence = max(0.0, m.permanence)$
- 9) for c in columns:

- 10) $\text{minDutyCycle}(g) = 0.01 * \text{maxDutyCycle}(\text{neighbors}(g))$
- 11) $\text{activeDutyCycle}(g) = \text{updateActiveDutyCycle}(g)$
- 12) $\text{boost}(g) = \text{boostFunction}(\text{activeDutyCycle}(g), \text{minDutyCycle}(g))$
- 13) $\text{overlapDutyCycle}(g) = \text{updateOverlapDutyCycle}(g)$
- 14) if $\text{overlapDutyCycle}(g) < \text{minDutyCycle}(g)$
- 15) then
- 16) $\text{increasePermanences}(g, 0.1 * \text{connectedPerm})$
- 17) $\text{inhibitionRadius} = \text{averageReceptiveFieldSize}()$

部分参数说明如下:

columns:所有柱状区域的列表。

overlap(c):柱状区域 c 对于特定输入的空间沉积池的覆盖值。

activeColumns(t):因自底向上输入而活跃的柱状区域列表。

desiredLocalActivity:一个控制经过抑制后仍活跃的柱状区域数量的参数。

inhibitionRadius:被连接到柱状区域的平均感受域大小。

neighbors(c):在柱状区域 c 抑制半径内所有柱状区域的列表。

minOverlap:会进入抑制过程的柱状区域覆盖值的最小值。

boost(c):在学习时被用与柱状区域计算的 c 的促进系数,用于增加非活跃柱状区域的覆盖值。

节点学习记忆过程如图 3 所示。

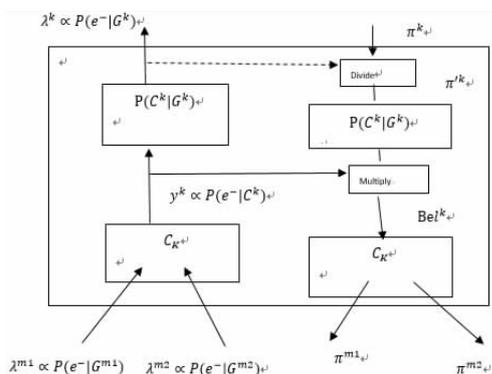


图 3 节点学习记忆过程

根据 1.1 节中特征提取的方法将测试集中恶意应用的特征值提取并转化为 HTM 网络可识别的稀疏离散变量的形式,输入到完成训练的 HTM 网络中进行序列识别以确定测试集中是否为恶意应用。

在算法的实际实现中,因为 numenta 公司已经把开

源项目 nupic 放到 github 上,使用 python 和 c++ 实现。官网上也给出了开源的 3 个应用的源代码 GROC FOR IT ANALYTICS^[9]、ROGUE BEHAVIOR DETECTION^[10]和 GEOSPATIAL TRACKING^[11],它们是从 3 个不同的角度实现 HTM 算法的思想。在实际实现中参考它们算法实现的思想,包含简单的层级、时间性、稀疏离散表征等基本实现满足实验的基本要求。

1.4 系统流程图

图 4 为该系统的流程图,反映了系统的各个组成部分。

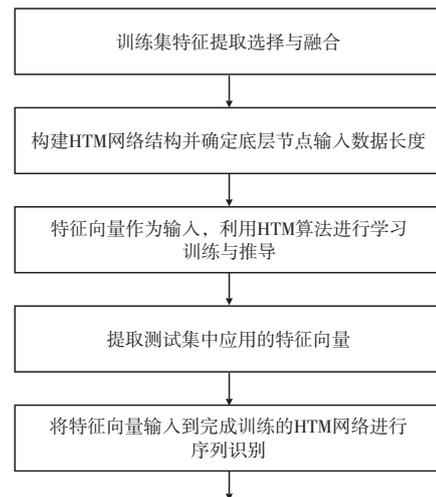


图 4 系统流程图

2 实验结果及分析

2.1 实验条件及指标

实验平台选取 Win 7 64 位操作系统,4 G 内存、酷睿 i5 处理器。采集的样本由正常程序代码和恶意正常程序代码组成,其中正常的应用随机采集于 Google Play 商店或者一些大型互联网厂商的官方 App,恶意应用来自于北卡罗来纳州立大学建立的恶意样本数据库和一些自己平时收集已经披露的大型安全事件的恶意样本。例如,通过疯狂截图来获取聊天信息、短信内容,窃取用户隐私信息;通过手机僵尸网络“挖矿”的 CoinKrypt 家族木马^[13];将移动设备加密锁屏后,对用户进行勒索的 simplelock 家族木马;伪造关机,实际上在后台窃听的 shutdownhack 家族木马以及短信蠕虫等。根据恶意应用的行为种类选取合适的学习样本,以达到更好的训练效果。

本文准确率 (TPR) 的定义为正确检测出恶意样本的个数和恶意样本总数的比例,定义 TP 表示正确的检测出的恶意样本的数目, FN 表示被错误的检测为正常

样本的数目, TPR 的测算可表示为:

$$TPR = \frac{TP}{TP + FN} \tag{5}$$

本文误报率 (FPR) 的定义为正常样本被误报为恶意样本数目和被分类正常样本总数之比, 定义 FP 为被误报为恶意样本的正常样本的数目, TN 表示正确的被分类的正常样本数目, FPR 的测算^[14]表示为:

$$FPR = \frac{FP}{FP + TN} \tag{6}$$

分类准确率 (CA) 为把正常样本和错误样本分开的正确率, 其中, TOT 为正常样本和病毒样本的总数, FN 代表被错误的检测为正常样本的数目, 定义 FP 为被误报为恶意样本的正常样本的数目:

$$CA = \frac{TOT - FN - FP}{TOT} \tag{7}$$

2.2 实验测试

在该实验中选取 300 个正常的应用和 126 个恶意应用作为测试样本, 选取传播范围比较广、影响比较大的 $ADRD$ 和 $DroidKungFu$ 恶意应用为实验样本, 并规定每类恶意应用的有效特征数为 3 个, 检测到的结果大于等于 2 时即认为该样本为恶意应用, 并利用该算法和第三方杀毒软件 (金山手机毒霸、360 手机卫士、卡巴斯基 KAV 手机杀毒) 作对比, 该测试样本经过本算法的训练和测试得出的结果和第三方杀毒应用对样本库的扫描结果如图 5 所示。

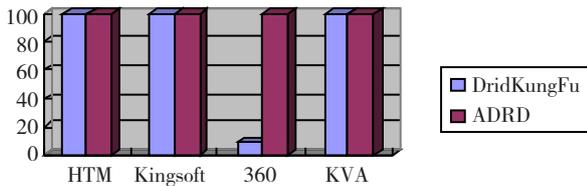


图5 TPR 对比图

从图 5 可以看出, 对于有少量样本的 $ADRD$ 类检测率均为 100%, 而对于量为 350 的 $DroidKungFu$ 类, 本文的检测算法与金山、 KAV 结果接近, 其正确率均接近于 100%, 但是 360 杀毒软件的准确率在 10% 左右, 这是因为其特征库中很少这类病毒的特征, 故准确率很低。由此可知, 第三方应用对软件特征库很依赖, 如果恶意代码软件变化范围较大时就不太容易检测出其正确结果。图 6 为误报率的对比图。

从图 6 可以看出, 本文的检测算法与第三方杀毒软件的误报率都比较低, 误报率均低于 0.2%。所以本文的检测算法是行之有效的。

本文检测算法、金山手机毒霸、360 手机卫士、卡巴

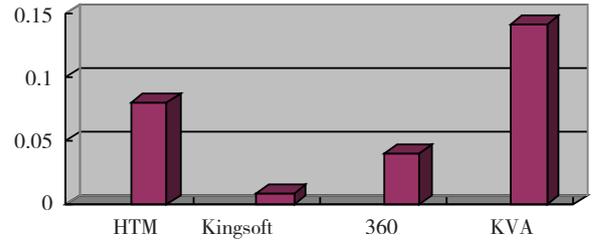


图6 FPR 对比图

斯基 KVA 手机杀毒检测时间随样本数目的变化关系如图 7 所示。从图 7 可以看出, 检测时间随样本数目的变化而变化, 基本呈线性关系, 本文算法的检测效果为 12 分钟 400 个样本, 即检测每个样本需要 1.8 s, 其检测效率没有 360、 KVA 效率高, 略高于金山毒霸, 360 检测效率较高但是其准确率较低。

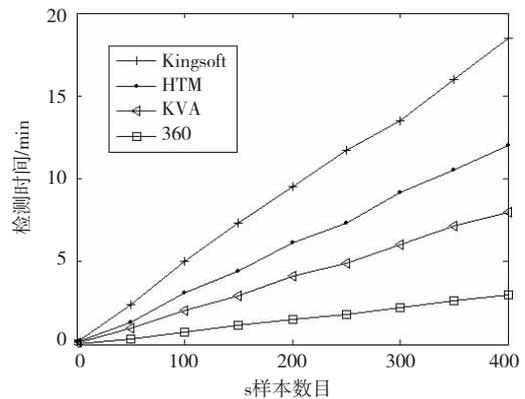


图7 检测时间图

2.3 不同检测算法性能

实验在 WEKA 平台上比较本算法和其他的数据挖掘算法的性能, 其他算法由决策树算法 ($J48$)、朴素贝叶斯算法 ($Naive Bayes$)、支持向量机算法 (SVM) 和本算法作比较。

收集的样本分为正常应用和恶意应用, 其中正常 App 数量为 126 个, 恶意应用一共 106 个, 来源于杀软测试中的样本进行抽取。其中木马 35 个、病毒 35 个、蠕虫 36 个, 表 1 给出了相关数据的统计信息。

表1 数据集信息

程序类型	文件个数	平均大小/KB	最小/KB	最大/KB
正常应用	126	1176	6	132436
木马	35	312	4	7649
病毒	35	285	5	5837
蠕虫	36	139	3	1362

在该平台验证下, 检测的结果见表 2。从表 2 的数据可以看出 HTM 算法在 3 种测试集中的准确率、误报率、分类准确率均优于其他 3 种数据挖掘算法。

表 2 不同算法检测结果

测试集	木马测试集 - 正常应用				病毒测试集 - 正常应用				蠕虫测试集 - 正常应用			
	NB	J48	SVM	HTM	NB	J48	SVM	HTM	NB	J48	SVM	HTM
TPR	0.921	0.921	0.971	0.973	0.968	0.962	0.983	0.984	0.961	0.986	0.953	0.992
FPR	0.085	0.114	0.069	0.05	0.058	0.096	0.046	0.047	0.049	0.079	0.062	0.038
CA	0.925	0.921	0.952	0.986	0.959	0.941	0.984	0.978	0.961	0.967	0.957	0.994

在实验分析过程中,可以看出在同一测试集情况下,正常应用 - 木马测试集中,HTM 算法准确率、误报率、分类准确率均优于其他 3 种算法。同样其他两个测试集正常应用 - 病毒测试集、正常应用 - 蠕虫测试集也是这种状况,从表 2 中还可以看出 HTM 算法对含有木马测试集、病毒测试集、蠕虫测试集的正常应用都能正常的检测出其恶意应用。

从表 3 可以看出本算法所用的训练时间较长,测试时间和其他算法相比处于中间水平,其原因主要在于 HTM 算法的数据反馈时间^[13]和 HTM 网络训练完成的训练时间较长。

表 3 不同算法的效率

算法	训练的时间/s	测试的时间/s
Naïve Bayes	0.12	0.04
J48	0.31	0.09
SVM	0.46	0.14
HTM	0.51	0.10

3 结束语

本文提出了一种新的基于深度学习 HTM 算法的恶意移动应用检测系统,该移动应用检测系统包含适合 Android 应用特点的特征提取、特征选择与融合、HTM 算法的学习训练,通过对该系统分析得出本算法和杀毒软件对比其准确率、误报率均达到了要求,与不同检测算法比较可知,本文算法的检测率、误报率、分类准确率均优于其他算法,但本算法和其他算法相比训练和测试时间较长,在下一步的算法模型改进中尝试建立一种算法融合的实践,把其他检测算法和 HTM 算法融合,做出一种最优的判决规则,使检测率、误报率、分类准确率达到较好的状况下,训练时间和测试时间也不会耗费较长时间。

参考文献:

[1] George D,Hawkins J.A hierarchical Bayesian Model of invariant pattern recognition in the visual cortex[C]//Proceedings of 2005 IEEE International Joint Conference on Neural Networks(IJCNN),Montreal,Canada,July 31-

August 4,2005:1821-1817.
 [2] Hawkins J,George D,Niemasik J.Sequence memory for prediction,inference and behavior[J].Philosophical Transactions on the Royal Society B,2009,364:1203-1211.
 [3] vicarious[EB/OL].[2015-05-10].http://vicarious.com/
 [4] Vicarious announces MYM15 million funding for AI software based on the brain [EB/OL]. (2012-08-24) [2015-05-10]. http://www.kurzweilai.net/vicarious-announces-15-million-funding-for-ai-software-based-on-the-brain
 [5] Rehn E M,Maltoni D.Incremental learning by message passing in Hierarchical Temporal Memory [J]. Neural Computation,2014,26(8):1763-1809.
 [6] 李元诚,樊庆君.未知恶意代码的深度学习检测算法:中国,201110373558[P].2012-04-11.
 [7] 黄聪会,陈靖.一种基于危险理论的恶意代码检测方法[J].中南大学学报:自然科学版,2014,45(9):3057-3058.
 [8] Numenta Platform for Intelligent Computing[EB/OL]. [2015-05-12].http://numenta.org/htm-white-paper.html
 [9] Breakthrough science for it anomaly detection[EB/OL]. [2015-05-12]. http://numenta.com/assets/pdf/grok/resources/1.6/Grok-1.6-DataSheet.pdf
 [10] Numenta.The science of anomaly detection[EB/OL]. [2015-05-12]. http://numenta.com/assets/pdf/whitepapers/Numenta%20White%20Paper%20-%20Science%20of%20Anomaly%20Detection.pdf
 [11] Numenta.Geospatial tracking[EB/OL]. [2015-05-12]. http://numenta.com/assets/pdf/whitepapers/Geospatial%20Tracking%20White%20Paper.pdf
 [12] 李挺,董航.基于 Dalvik 指令的 Android 恶意代码特征描述及验证[J].计算机研究与发展,2014,51(7):1458-1466.
 [13] Weka 3:Data Mining Software in Java[EB/OL].[2015-

- 05-15].<http://www.cs.waikato.ac.nz/ml/weka/> and temporal model for learning and recognition[D].
[14] George D. How the brain might work: A hierarchical Stanford: Stanford University, 2008.

Detection of Malicious Android Application Based on HTM Algorithm

QIU Shenjian^a, ZHANG Shibin^a, LIU Pingguang^b

(a. College of Information Security Engineering; b. College of Communication Engineering, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: With the Internet users' conversion from traditional PC to mobile terminal, the mobile security has been more and more concerned. In order to improve the detection efficiency of unknown malicious mobile application, aiming at the poor detection ability problem of traditional detection in detecting the malicious applications that introduces polymorphic and deformation techniques, a method to detect malicious Android mobile applications based on HTM algorithm is proposed, the application detection contains the feature extraction that aims at Android application Dalvik instructions characteristic and the feature selection and integration by using the information gain method, and the sequence mode is trained and deduced by HTM algorithm, then the feature extraction and fusion result of test sample is input into the HTM network that completes training, therefore, the purpose of detecting malicious applications is achieved. The experiment simulation show: the detection rate of designed malicious application detection method is nearly 100%, and has high detection efficiency; the false positive rate is 0.08%. Compared to other algorithms, the detection rate, false positive rate and classification accuracy rate of proposed malicious detection method are better, and it can be applied to different types of malicious applications, but the training and testing time is longer.

Key words: mobile security; HTM algorithm; Dalvik instruction; information gain; malicious application; detection