软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2012,23(12):3149−3160 [doi: 10.3724/SP.J.1001.2012.04221]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 基于证据推理的程序恶意性判定方法*

张一弛, 庞建民, 赵荣彩

(解放军信息工程大学 信息工程学院,河南 郑州 450002)

## Evidential Reasoning Method for Decision of Program Maliciousness

ZHANG Yi-Chi, PANG Jian-Min, ZHAO Rong-Cai

(Institute of Information Engineering, PLA Information Engineering University, Zhengzhou 450002, China)

+ Corresponding author: E-mail: each5595@163.com

**Zhang YC, Pang JM, Zhao RC. Evidential reasoning method for decision of program maliciousness. *Journal of Software*, 2012,23(12):3149−3160 (in Chinese).** http://www.jos.org.cn/1000-9825/4221.htm

**Abstract**: Considering the fact that the determination of the executable file maliciousness is hard to achieve, an approach based on the evidence theory is presented in this paper. First, a model for determining the maliciousness is established. Then characters compromising security are extracted to construct the set of program behaviors through decompiling the program. The model is trained using the BP neural network to gain the basic probability assignment functions (BPAF) of each behavior, and the weighted sum method is applied to combine the program behaviors, determining the executable file maliciousness. Experimental results demonstrate the validity of the approach which uses the evidence theory to determine the maliciousness of program.

**Key words**: malware detection; evidence reason; neural network; program behavior; similarity

**摘 要**: 针对可执行程序恶意性难以判定的情况,提出一种基于证据推理的程序恶意性判定方法.首先,建立程序恶意性判定模型;然后,通过对程序进行反编译,抽取影响程序安全性的特征,建立程序行为集合;使用BP神经网络对模型进行训练得到各个行为的概率分配函数BPAF(basic probability assignment functions),并使用加权和形式的合成法则对程序行为进行合成;最后,实现对程序恶意性的判定.实验结果表明了该方法的有效性.

**关键词**: 恶意代码检测;证据推理;神经网络;程序行为;相似度

**中图法分类号**: TP311 **文献标识码**: A

程序恶意性分析是众多应用领域中的核心技术,包括病毒检测与清除、入侵检测、恶意代码行为阻断等领域.判定一个程序是否具有恶意性是一项困难的任务,Cohen 进一步证明了不存在一种检测方法能够检测出所有的恶意程序,也不存在一种方法可以检测出某一个特定恶意程序的所有变体[1].但是,各大反病毒研究机构都在努力研究程序恶意性的近似判定方法,以应对爆炸式增长的恶意代码.

进行程序恶意性分析,传统的方法主要有基于特征码方法、基于完整性方法、基于行为方法以及程序语义方法.近年来又出现了许多新型的检测方法,如基于数据挖掘的方法、基于生物免疫的方法以及基于人工智能的方法等[2].不管使用什么方法检测,最终还是依赖恶意性判定规则和判定算法,检测能力大小取决于规则的定

* 基金项目: 国家高技术研究发展计划(863)(2006AA01Z408, 2009AA01Z434); 河南省重大科技攻关项目(092101210501)
收稿时间: 2011-10-18; 定稿时间: 2012-04-01

义,检测的效率和精度取决于恶意性判定算法.误报率取决于规则的可靠性,漏报率取决于规则的完备性.以上方法中,特征码检测、完整性检测、行为检测、语义检测方法是基础,是恶意性判定中规则的具体体现;数据挖掘、生物免疫、人工智能的方法是对恶意性判定规则的扩展.因此,只有实现系统的自动扩展,才能实现恶意代码检测的智能化、自动化,实现检测系统的相对完备性和可靠性.

由于正常程序与恶意程序有着一些明显的不同行为模式,本文从这些行为特点入手,确定程序恶意性判定模型,选取程序特征行为,利用 BP 神经网络完成对模型的训练,并使用加权和形式的合成法则进行证据合成,以达到程序恶意性判定的目的.由于在程序恶意性判定的过程中使用了多个行为特征,这些特征都具有一定的不确定性,因此,本文将证据理论应用于程序恶意性判别中,提出一种基于证据推理的程序恶意性判定方法.

## 1 相关工作

传统的程序恶意性判定方法是通过分析程序中是否具有事先定义的某种特征来确定,但随着混淆、变形技术的不断发展,原有的恶意特征不再明显.为了应对这种局面,出现了基于人工智能的恶意代码检测方法,由于该方法能够将程序中多种不精确的、模糊的特征进行融合与推理,综合判断程序中是否具有恶意性,提高了判断的准确性,因此成为近年来研究的热点.

2004 年,Abou-Assaleh 等人[3]提出基于 N-Gram 特征码的新型恶意代码检测手段.N-Gram 分析是近年来比较成功的一种文本分类方法,N-Gram 算法能够有效抽取频率最高的 N-Gram,从而利用它作为检测特征,依据它匹配最多的情况来区分该程序是良性的还是恶意的.

2007 年,田新广[4]提出了一种基于马尔可夫模型的程序行为异常检测方法,该方法所采用的模型将马尔可夫链的状态同特权程序运行时的系统调用联系在一起.基于状态序列的出现概率对特权程序当前行为的异常程度进行分析,从而得出恶意性判定结果.

2009 年,El-Bakry[5]提出使用快速神经网络检测检测恶意代码的方法,该方法利用神经网络的强大分类能力识别恶意代码,并能够与现有的病毒检测方法结合使用.

2010 年,Golovko[6]提出了基于神经网络和人工免疫系统(AIS)的方法检测未知恶意代码和入侵检测,神经网络技术和人工免疫系统已经成功应用于异常活动检测与识别领域中的很多问题,将两种智能识别方法结合起来能够成功识别多种经过变形的恶意代码和入侵.

上述方法都应用了人工智能领域的相关方法,但本文提出的方法与上述方法存在不同:首先,本文使用反编译逆向分析技术得到程序中具有的多种特征;其次,本文使用模糊识别技术计算程序行为的相似度;再次,本文使用 BP 神经网络确定推理所需的概率分配函数;最后,本文基于证据推理完成对程序恶意性的判定.

证据推理应用在程序恶意性判定上的优势主要体现在 3 个方面:首先是证据理论具有比 Bayes 概率理论更弱的条件,满足了程序恶意性不具有概率可加性的要求;其次,证据理论中利用概率分配函数进行证据的合成,当无法判定程序的恶意性时,能够通过概率分配函数保留相关信息,分析是“不确定”还是“不知道”程序的恶意性;最后,证据理论不但允许人们将信度赋予假设空间的单个元素,而且还能够赋予它的子集,这样就方便安全人员在不同层次上收集程序恶意性的相关证据.

## 2 证据推理

### 2.1 Mass函数

为了描述和处理知识的不确定性,证据理论使用了概率分配函数、信任函数和似然函数[7].具体定义如下:

假设 $D$ 为取值空间,$A$ 为 $D$ 的任意子集.

**定义 1(概率分配函数)**. 设函数 $Mass:2^D\to[0,1]$,并且满足:

1) $Mass(\varnothing)=0$;

2) $\sum_{A\subseteq D} Mass(A)=1$.

则称 Mass 是 $2^D$ 上的概率分配函数,$Mass(A)$为 $A$ 的基本概率数.

**定义 2(信任函数(belief function))**. 设函数 $Bel:2^D\to[0,1]$,并且满足:

$$Bel(A)=\sum_{B\subseteq A}Mass(B).$$

信任函数又称为下限函数,$Bel(A)$表示当前环境下,对假设集 $A$ 的信任程度,其值为 $A$ 的所有子集的基本概率之和,表示对 $A$ 的总的信任度.

**定义 3(似然函数(plausibility function))**. 设函数 $Pl:2^D\to[0,1]$,并且满足:

$$Pl(A)=1-Bel(\sim A),$$

其中,$\sim A=D-A$.似然函数又称为上界函数,$Pl(A)$表示对 $A$ 为非假的信任程度.

信任函数 $Bel(A)$和似然函数 $Pl(A)$分别表示命题 $A$ 信任度的下界和上界.同样,也可以用它来表述程序恶意性的下界和上界.本文借助概率分配函数、信任函数和似然函数等概念及相关理论,完成程序恶意性的判定.

### 2.2 合成规则

Dempster 合成规则(dempster's combinational rule)也称证据合成公式,其定义如下:

对于$\forall A\subseteq D$,$D$ 上的两个概率分配函数 $m_1,m_2$ 的 Dempster 合成规则为

$$m_1\oplus m_2(A)=\frac{1}{K}\sum_{B\cap C=A}m_1(B)\cdot m_2(C),$$

其中,$K$ 为归一化常数: $K=\sum_{B\cap C\neq\varnothing}m_1(B)\cdot m_2(C)=1-\sum_{B\cap C=\varnothing}m_1(B)\cdot m_2(C)$.

同理,$n$ 个概率分配函数的 Dempster 合成规则如下:

对于$\forall A\subseteq D$,识别框架 $D$ 上的有限个概率分配函数 $m_1,m_2,\ldots,m_n$ 的 Dempster 合成规则为

$$(m_1\oplus m_2\oplus\ldots\oplus m_n)(A)=\frac{1}{K}\sum_{A_1\cap A_2\cap\ldots\cap A_n=A}m_1(A_1)\cdot m_2(A_2)\cdot\ldots\cdot m_n(A_n)\tag{1}$$

其中, $K=\sum_{A_1\cap A_2\cap\ldots\cap A_n\neq\varnothing}m_1(A_1)\cdot m_2(A_2)\cdot\ldots\cdot m_n(A_n)=1-\sum_{A_1\cap A_2\cap\ldots\cap A_n=\varnothing}m_1(A_1)\cdot m_2(A_2)\cdot\ldots\cdot m_n(A_n)$.

## 3 基于证据推理的程序恶意性分析

### 3.1 模型建立

定义程序恶意性分析的相关论域如下:

1) 论域 $A$ 表示可执行程序的行为特征集合.设程序的行为共有 $m$ 种,则 $A=\{a_1,a_2,\ldots,a_m\}$;
2) 本文将所有待检测程序作为识别对象,定义识别框架:$\Theta_s=\{N,M\}$.其中,$N$ 表示正常程序,$M$ 表示恶意程序.此外, $2^{\Theta_s}$ 的非空子集还包括$\{N,M\}$表示可疑程序;
3) 论域 $V$ 表示程序恶意性分类集合.设程序恶意性分类集合中含有 $n$ 种类型,则 $V=\{v_1,v_2,\ldots,v_n\}$.

给定程序全集论域 $P$,待检测程序 $x\in P$.程序行为的识别程度用集合 $B=\{b_1,b_2,\ldots,b_m\}$表示,其中,$b_i$ 表示对行为 $a_i$ 的识别程度,$b_i\in[0,1](1\leqslant i\leqslant m)$.

本文将程序的恶意性分析结果用集合 $T=\{t_1,t_2,\ldots,t_n\}$表示,其中,$t_j\in[0,1](1\leqslant j\leqslant n)$表示程序隶属于程序恶意性分类集 $v_j$ 的程度.

根据以上描述,我们建立了 ERMD(evidence-reasoning based malicious decision)模型,该模型的输入为表示程序 $x$ 恶意性的特征向量 $B_x$;输出为程序恶意性分析结果 $T_x$ 的最终判定类 $t_x(1\leqslant x\leqslant n)$.按照输入/输出的形式对程序中的行为进行合成,计算出程序的恶意性.

Nombril(note producer of malicious behavior for binary file)是作者参与研制的基于证据推理的程序恶意性判定系统,图 1 给出了 Nombril 系统框架.系统始于对 PE 格式的可执行程序进行逆向分析,得到文件格式、机器指令、库函数调用、高级语言以及它们之间的关联信息、模糊识别程序中具有的行为.最后,通过证据推理融合程序中具有的多个行为特征,判定程序具有的恶意程度.

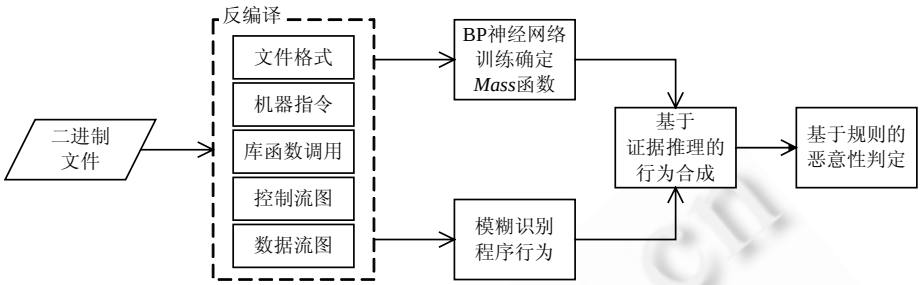


Fig.1 Framework of Nombril

图 1 Nombril 系统框架

程序恶意性判定过程主要由以下模块实现:

1) 行为获取:该模块使用反编译逆向分析技术,从二进制文件 $x$ 中获取程序特征行为 $A_x=\{a_1,a_2,\ldots,a_m\}$;
2) 行为相似度计算:由于程序的特征行为具有模糊性和相似性,该模块计算特征行为与已定义的标准行为之间的相似度 *wSim*.结合模块 1)中的特征行为 $A$,得到特征行为识别程度 $B_x=\{b_1,b_2,\ldots,b_m\}$;
3) *Mass* 函数确定:该模块将训练集中的程序行为作为输入,经过神经网络的学习,得到程序特征行为对恶意性分类集合的概率分配函数 $Mass=\{Mass_{\alpha\beta}|\alpha\in A,\beta\in T\}$;
4) 证据合成:该模块将程序 $x$ 的特征行为识别程度 $B_x$ 与对应行为的 *Mass* 函数做乘积,得到用于恶意性判定的概率分配函数,使用改进的合成公式完成程序 $x$ 特征行为的合成,得到恶意性的分析结果 $T_x=\{t_1,t_2,\ldots,t_n\}$;
5) 恶意性判定:该模块计算恶意性分析结果类的信任度 $Bel(T)$和不确定度 $Unc(T)$,采用基于规则的方法对程序的恶意性进行判别,得到程序 $x$ 最终的判定结果类 $t_x$.

### 3.2 行为获取

Nombril 系统前端使用反编译逆向分析技术获取程序的行为,对软件反编译逆向分析包括文件装载、指令解码、语义映射、流图构造、过程分析、类型分析、结果输出这 7 个阶段.

1) 文件装载:此阶段主要完成读入二进制文件,并进行一些初步分析,包括文件格式解析、文件信息搜集、文件性质判定等;
2) 指令解码:主要是根据体系结构的指令编码规则,对文件中使用的指令进行解释、识别和翻译的过程,得到程序对应的机器指令;
3) 语义映射:是将机器指令通过语义描述的方法表示出来,屏蔽不同体系结构对程序分析带来的影响;
4) 流图构造:在程序的分析过程中,需要借助编译原理中的相关概念完成,因此需要构建相关流图,主要有:控制流图 CFG(control flow graph)、调用图 CG(call graph)和依赖图 DG(dependence graph);
5) 过程分析:此阶段是将目标文件中的过程信息恢复出来,包括过程边界分析、过程名、参数列表和返回值信息;
6) 类型分析:此阶段是分析程序中各个存储单元所携带的类型信息;
7) 结果输出:指是逆向分析的最终阶段,输出的内容包括汇编代码、中间表示、控制流和数据流图以及高级语言代码等程序信息.

Nombril 系统中选定 7 类行为来描述程序的恶意性,具体定义见表 1.

**Table 1** Summary of the program behaviors

**表 1** 行为定义及表述

| 行为编号 | 行为表述 |
|---|---|
| $b1$ | 文件结构异常,文件头大小异常 |
| $b2$ | 文件结构异常,起始节指向非代码节 |
| $b3$ | 指令序列特征,对 Kernel32 基地址进行搜索等 |
| $b4$~$b7$ | API 函数调用有关,每类特征分别对应若干恶意特征子类,每个恶意特征子类又对应若干 API 函数序列 |

每一个特征分量的特征值用特征的识别程度来表示,即从待检测程序中识别出某一恶意特征类的程度.以上行为的提取工作在对程序进行反编译的过程中完成.程序装载阶段完成对程序文件格式的分析与加载工作,并根据分析结果对 $b1$ 和 $b2$ 赋值;反汇编阶段完成指令的解析工作,并检测是否存在恶意指令序列,同时对 $b3$ 进行赋值;在流图构造和过程分析阶段,重点分析程序的 API 调用行为,并对 $b4$~$b7$ 进行赋值.

### 3.3 行为相似度计算

在分析 API 调用行为时,对 $b4$~$b7$ 赋值存在困难:一方面,由于恶意代码编写者常采用混淆或其他变形方法来隐藏其想要实现的恶意行为;另一方面,API 函数都经过封装处理,因此存在等价 API 函数情况,也就是不同的 API 函数完成相同的功能.

当从可执行程序中提取的 API 函数调用序列与特征库中存储的恶意特征序列不能完全匹配时,需要采用模糊模式匹配的方法计算二者之间的相似程度,并以相似度作为此程序具备该恶意行为的程度.除此以外,考虑到每个 API 函数序列中都存在着一个或多个关键函数完成该恶意行为类的主要或核心功能,因此在计算 API 序列相似度时,还需要考虑不同 API 函数对序列相似度的影响因子.

假设特征库中某一恶意特征行为类对应的某个 API 函数序列为$(f_1,f_2,\ldots,f_n)$,根据各个 API 函数对序列功能的影响力不同,给每个 API 函数分配一个权值 $\tau_{f_k}$ $(1\leqslant k\leqslant n)$,并使得该序列中所有库函数的权值之和为 1,即

$$\sum_{k=1}^{n}\tau_{f_k}=1.$$

本文选用 LD(levenshtein distance)算法计算两序列之间的相似度 $Sim$,并利用点乘运算来计算加权相似度 $wSim$.设反编译分析得到程序的 API 函数序列为 $(f_1', f_2',..., f_m')$,则该序列与特征库中序列$(f_1,f_2,\ldots,f_n)$的加权相似度 $wSim=Sim\cdot\sum_{k=1}^{m}\tau_{f_k'}$,其中,$Sim$ 是采用 LD 算法计算得到的两序列之间的相似度,权值的求和遵循以下约定:

- 设 $f_i'\in(f_1', f_2',..., f_m')$,$f_j\in(f_1,f_2,\ldots,f_n)$,其中,$1\leqslant i\leqslant m$ 且 $1\leqslant j\leqslant n$.若 $f_i'=f_j$,则 $\tau_{f_i'}=\tau_{f_j}$;
- 否则,$\tau_{f_i'}=0$.

按照以上描述,求解两序列间加权相似度的算法流程如图 2 所示.

**算法 1**. 计算两函数调用序列之间的加权相似度.
输入:已识别函数调用序列 $A$ 和函数调用特征序列 $B$,$A$ 和 $B$ 的长度分别为 $m$ 和 $n$;特征序列 $B$ 中函数的权值序列 $C$;
输出:序列 $A$ 和序列 $B$ 的加权相似度 $wSim$.

```
1  初始(n+1)×(m+1)阶距离矩阵 D,并令 D[1][j]=j−1,D[i][1]=i−1,其中,1≤i≤m+1,1≤j≤n+1;
2  初始化距离增量 temp=0;
3  初始化权值变量 weight=0;
4  遍历两字符串,如果 A[i]=B[j],则令 temp=0,weight=weight+C[j];否则,令 temp=1,weight 值不变;
5  矩阵元素 D[i][j]取以下值中的最小值:D[i−1][j]+1,D[i][j−1]+1 和 D[i−1][j−1]+temp;
6  扫描完后,令 d=D[m+1][n+1],d 即为序列 A 和 B 之间的距离值
7  按照下式计算 A 和 B 的相似度 Sim=1−d/max{m,n};
8  计算序列 A 和 B 的加权相似度,wSim=weight×Sim;
9  算法结束,返回 wSim 的值
```

Fig.2 Similarity algorithm of function call sequence

图 2 函数调用序列的加权相似度算法

### 3.4 Mass函数确定

应用证据理论的困难在于如何将训练集中程序的行为特征转换为可以进行推理的概率分配函数,这是应用证据理论进行程序恶意性判定的前提条件.目前,确定概率分配函数的方法大都是人工设定的,这种方法存在计算复杂、自适应差、主观性强等缺点.而神经网络具有较强的泛化能力,在知识学习中具有归纳全部数据的自学习能力[8].因此,可以利用神经网络对大量样本进行学习,就能起到领域专家的作用,解决概率分配函数赋值难以获取的问题.

BP(反向传播)神经网络适用于分类问题的处理,因此可以用来构建概率分配函数.BP 网络是一种反向传递并能修正误差的多层映射网络,它由输入层、隐层以及输出层神经元构成,其中,隐层可以是多层的.输入信息向前传播到隐层神经元,经过神经元作用函数的处理,最后将隐层的输出信号传播到输出神经元,得到输出结果.图 3 给出了 BP 网络的结构图.

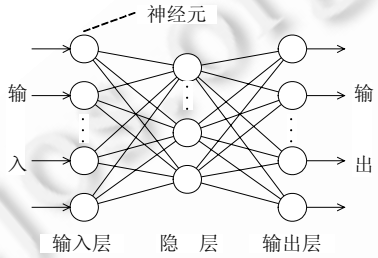


Fig.3 Structure of BP neural network

图 3 BP 神经网络结构

在 BP 网络中,神经元作用函数选用 *S* 形函数.一般形式为

$$f(x)=a+\frac{b}{1+\mathrm{e}^{-dx}} \tag{2}$$

其中,*a*,*b*,*d* 是常数.

BP 网络的学习过程由正向传播和反向传播两部分组成[8]:

(1) 正向传播过程:输入信息经过每层神经元的处理,最后得到输出信号;

(2) 若实际输出与样本不一致,则将误差信号反向逐层传播,从而修正各连接权值.

重复上述两个过程,直到输出信号的误差小于预先设定值.*i* 为非输出层中的一个神经元,下一层中存在神经元 *j*,则神经元 *j* 的输入为 $net_j=\sum_i W_{ij}O_i$,其中,$O_i$ 为来自前一层神经元 *i* 的输入;$W_{ij}$ 为神经元 *i* 与神经元 *j* 之间的连接权重,若神经元 *i* 和神经元 *j* 处于不相邻的两层,则 $W_{ij}$=0.因此,神经元 *j* 的输出为 $O_j=f(net_j)$,其中,*f* 如公式(2)所示.

BP 训练算法是一种迭代算法,用网络的实际输出与期望输出之间的最小方差调整连接权值.BP 网络的均方差函数为 $E=\frac{1}{2}\sum_j(O_j-D_j)^2$,其中,$D_j$ 为神经元 *j* 的期望输出.

为了使训练过程尽快收敛,调整连接权值:$\Delta W_{ij}(t+1)=\eta\delta_j O_i+\alpha\Delta W_{ij}(t)$,其中,$\alpha$是动量因子($0<\alpha<1$),反映前次学习效果的常数:$\Delta W_{ij}\propto\frac{\partial E}{\partial W_{ij}}=\frac{\partial E}{\partial net_j}\times\frac{\partial net_j}{\partial W_{ij}}=\delta_j\times O_i$,$\eta$为学习速率($0<\eta$).

把训练集中的程序每个行为作为输入,经过神经网络的学习,得到行为对恶意性分类集合的概率分配函数.本文采取将 *m* 个特征向量作为神经网络的输入,*n* 个输出分别对应程序恶意性集合,*C* 层神经元.在完成训练的同时,连接权值 *W* 就随之确定.我们定义 $Mass_{\alpha\beta}$函数,表示程序行为$\alpha$对输出类别$\beta$的基本置信函数:

$$Mass_{\alpha\beta}=(f')^C\prod_{0\leqslant i<C}^{j=i+1}\left(\sum_i W_{ij}\right),$$

其中,$f'$表示 $f$ 的一阶导数.

### 3.5 证据合成

经典的 Dempster 合成规则,在对具有冲突的证据进行合成时会产生悖论,使得结果有悖常理.证据冲突与证据相关是实际应用证据理论时常常遇到的问题,在程序恶意性判定中也存在这种问题.自 Dempster 和 Shafer 提出组合框架后已有不少改进,目前,处理证据冲突的经典方法有 Yager 改进方法、Toshiyuki 方法、平均法以及 Smets 合成规则等等.本文采用加权和形式的合成方法[9].

设 $m_1,m_2,\ldots,m_n$ 所对应的证据集为$\{F_1,F_2,\ldots,F_n\}$,并设证据 $i$ 和 $j$ 之间的冲突大小为 $K_{ij}$,则

$$K_{ij}=\sum_{\substack{A_i\cap A_j=\varnothing\\ A_i\in F_i,\ldots,A_j\in F_j}} m_i(A_i)m_j(A_j).$$

定义$\varepsilon$为证据的可信度,$\varepsilon=e^{-k}$,其中,$k=\dfrac{1}{n(n-1)/2}\sum_{i<j}K_{ij}$ .其中,$i,j\leqslant n,n$ 为证据的个数.$k$ 与经典的 Dempster 合成规则公式(1)中的冲突程度 $K$ 不同,$k$ 反映了证据集中,两个证据之间的冲突程度.$\varepsilon$是 $k$ 的减函数,反映了证据的可信度.

新的合成公式定义如下:

$$\begin{cases} m(\varnothing)=0\\ m(A)=p(A)+k\times\varepsilon\times q(A),A\neq\varnothing,X\\ m(X)=p(A)+k\times\varepsilon\times q(X)+k(1-\varepsilon) \end{cases} \tag{3}$$

其中,$p(A)=\sum_{\substack{A_i\in F_i\\ \cap_{i=1}^{n}A_i=A}} m_1(A_1)m_2(A_2)\ldots m_n(A_n)$,$q(A)=\dfrac{1}{n}\sum_{i=1}^{n}m_i(A)$ .公式(3)又可以写成如下形式:

$$m(A)=(1-k)\frac{p(A)}{1-k}+k\times\varepsilon\times q(A) \tag{4}$$

通过对比公式(1)和公式(4)可以发现,公式(4)第 1 项中的 $p(A)/(1–k)$就是经典的 Dempster 合成规则.因此,可以认为本文所选用的合成规则实际上是经典 Dempster 合成规则的加权和形式,加权系数为$(1–k)*k$.当证据的冲突较小时,$k$ 值较小,公式(4)的第 1 项起主要作用,证据合成结果近似于经典 Dempster 合成规则结果;当证据的冲突较大时,公式(4)的第 2 项起主要作用,由$\varepsilon*q(A)$决定,由于$\varepsilon$表示证据可信程度,$q(A)$为证据对 $A$ 的平均支持度,因此,证据合成结果将主要由证据可信度$\varepsilon$和证据平均支持度 $q(A)$的乘积决定.该合成公式在证据存在冲突时,也能利用证据中的部分可用信息,可用程度取决于$\varepsilon$.

### 3.6 恶意性判定

通过神经网络训练得到每个程序行为的概率分配函数,使用加权和形式的合成规则对证据形成合成,得到一个表示程序中所有行为证据共同作用的概率分配函数.目前,对概率分配函数的分析与决策没有统一的一般性方法,必须根据具体的问题选择不同的方法.本文采用基于规则的方法对程序的恶意性进行判别,主要有以下 4 条规则:

规则 1. 判别结果类应具有最大的信任度,且大于阈值$\rho_1$;

规则 2. 判别结果类的信任度与其他类别信任度的差值必须大于阈值$\rho_2$;

规则 3. 判别结果类的不确定度必须小于阈值$\rho_3$;

规则 4. 判别结果类的信任度必须大于不确定度.

规则 1 表示具有最大信任度的类是判别结果类;规则 2 表示每一证据对不同类的支持程度应保持足够大的差异;规则 3 表示判别结果类的不确定度不能太大;规则 4 表示当对结果类的支持度不够时,不能对其进行分类.按照上述对合成算法的描述,可以得到基于规则的判决算法,如图 4 所示.

**算法 2**. 基于规则的程序恶意性判定算法.

输入:经过证据推理得到的恶意性分析结果 $T$,结果类 $T_i$ 的信任度为 $Bel(T_i)$,不确定度 $Unc(T_i)=Pl(T_i)-Bel(T_i)$;

输出:程序属于的恶意性分析结果类 $t_x$.

1 初始化 $Bel(T_{max})=0$,$T_{max}$ 用于存放信任度最大的类别;
2 初始化循环变量 $i=1$;
3 若 $i>n$,转步骤 6;
4 若 $Bel(T_i)>Bel(T_{max})$,则 $T_{max}=T_i$;
5 $i$++,转步骤 3;
6 断言($Bel(T_{max})>\rho_1$); //规则 1:判别结果类应具有最大的信任度,且大于阈值$\rho_1$
7 若不满足断言,转步骤 18;
8 初始化循环变量 $i=1$;
9 若 $i>n$,转步骤 13;
10 断言($|Bel(T_{max})-Bel(T_i)|>\rho_2$); //规则 2:判别结果类的信任度与其他类别的信任度的差值必须大于阈值$\rho_2$
11 若不满足断言,转步骤 18;
12 $i$++,转步骤 9;
13 断言(($Unc(T_{max})$)<$\rho_3$); //规则 3:判别结果类的不确定度必须小于阈值$\rho_3$
14 若不满足断言,转步骤 18;
15 断言($Bel(T_{max})>Unc(T_{max})$); //规则 4:判别结果类的信任度必须大于不确定度
16 若不满足断言,转步骤 18;
17 算法结束,返回 $t_x=T_{max}$; //判决结束,正常返回;
18 算法结束,返回 $t_x=NULL$; //不能分类,返回错误.

Fig.4 Rule-Based decision algorithm

图 4 基于规则的判决算法

## 4 实验评估

本节通过对 Nombril 系统进行测试来验证我们提出的方法.测试分为 3 个部分:首先,利用样本对系统进行训练,使用经典的分类方法作为对比,测试系统的性能;之后,使用经变形、混淆的病毒程序,与知名的杀毒软件进行对比,测试系统的抗混淆能力;最后,对系统进行综合测试,测试系统对于恶意代码检测的漏报率、误报率以及整体精度.

### 4.1 判定性能测试

样本空间中的样本程序主要来自于以下 3 种途径:

(1) Windows XP 安装后,系统分区下 Windows 目录中的全部 PE 可执行程序;

(2) 正版应用软件中随机采集的各种 PE 程序;

(3) 网络上的经多种途径收集到的病毒程序.

其中,前两种途径收集到的程序共 3 583 个,途径(3)收集到的程序共 3 572 个,因此样本空间由 7 155 个 PE 程序构成.

在对模型进行训练时,使用了 500 个正常程序和 500 个恶意程序用来给神经网络进行训练构造概率分配函数,之后随机抽取 1 000 个正常程序和 1 000 个恶意程序进行测试.在对程序恶意性进行判定时,规则 1 的阈值$\rho_1$为 0.741,规则 2 的阈值$\rho_2$为 0.253,规则 3 的阈值$\rho_3$为 0.098.BP 网络使用的参数见表 2.

为了验证提出的方法,我们使用 MatlAB7.0 实现如下检测方法,包括:

(1) 文献[10]提出的基于朴素贝叶斯的检测方法(简称 natïve Bayes);

(2) 文献[5]提出的基于神经网络的检测方法(简称 neural network);

(3) 文献[11]提出的基于支持向量机的检测方法(简称 SVM).

我们使用模式识别领域的 ROC(receiver operating characteristics)曲线作为评价指标,ROC 曲线用于比较不同分类方法的性能,曲线下面的面积(AUC)越大,算法的分类性能就越好、越稳定.

**Table 2** Parameters of BP network

**表 2** BP 网络选用的参数

| 参数名称 | 参数选取 |
|---|---|
| 层数 | 3 层 |
| 输入节点 | 7 个 |
| 输出节点 | 3 个 |
| 隐层节点 | 56 个 |
| 正常程序输出 | [100] |
| 恶意程序输出 | [001] |
| 阈值 | 0.3 |
| 中间层阈值初值 | 随机数 |
| 输出层阈值初值 | 随机数 |
| 输入层到中间层的权值初值 | 随机数 |
| 中间层到输出层的权值初值 | 随机数 |
| 误差平方和 | 0.05 |

为了更加方便直观地展示实验结果,绘制出了各种方法的 ROC 曲线图,实验结果如图 5 所示.

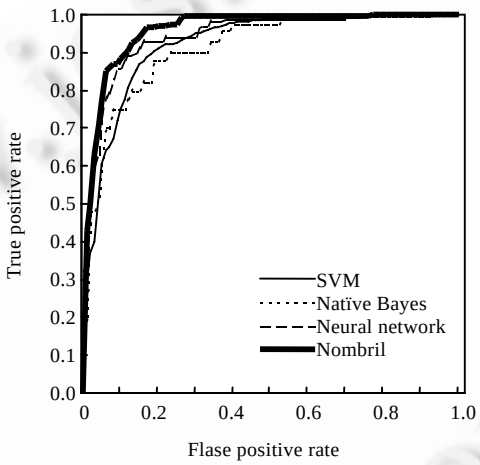


Fig.5 ROC curves for detecting malicious executables

图 5 检测恶意程序的 ROC 曲线图

由图 5 可以看出,使用基于证据推理的 ROC 曲线下方的面积大于其他分类方法,使用该方法的性能较优.这是由于证据推理能够更好地融合程序中的不同行为特征,使其具有较好的判定结果.

### 4.2 抗混淆测试

为验证 Nombril 系统对变形、混淆程序的检测效果,我们选择了 9 种知名的 PE 格式病毒进行测试.对这些病毒程序进行了混淆,得到了不同的变形版本.由于某些病毒的特殊性,在使用相应的混淆方法之后,无法生成可执行文件或者生成文件没有病毒功能,最后共成功得到了 65 个程序.使用的混淆、变形方法及生成程序的数目如下:

- original 版:原始版本,未经过变形、混淆(9 个);
- v1 版:插入垃圾代码(9 个);
- v2 版:修改数据段(9 个);
- v3 版:修改控制流(9 个);
- v4 版:相同语义指令替换(7 个);
- v5 版:不透明谓词混淆(9 个);

- v6 版:条件分支混淆(8 个);
- v7 版:隐式函数调用(5 个).

我们使用病毒的原始版本进行对 BP 神经网络进行训练.为了更好地说明判定效果,我们还选择了不同特点的国内外知名杀毒软件对病毒的原始版本及变形版本进行对比测试:

- 卡巴斯基 KAV2012,版本号:12.0.0.374;
- BitDefender 反病毒 2011,版本号:14.0.28.44;
- ClamAV,版本号:0.97.2;
- Rising2011,版本号:23.00.45.97;
- KV-2011,版本号:SP7.2.061628.

实验前,各款杀毒软件的病毒库全部在线更新至最新版本(2011 年 10 月 18 日上午 10 点).对各类病毒程序的检测结果如图 6 所示.

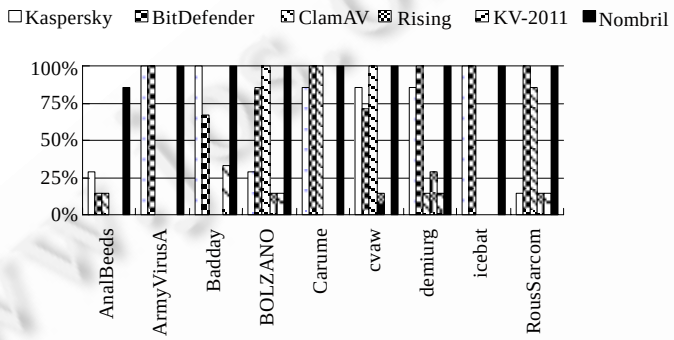


Fig.6 Detection rate of individual viruses

图 6 对每种病毒的检测率

从图 6 中能够发现,各款反病毒软件对于变形、混淆之后的恶意代码的检测率不高,而 Nombril 系统能够有效地对抗各种混淆手段.为了进一步分析变形、混淆对恶意性判定的影响,下面将按照变形、混淆的类型进行统计,如图 7 所示.

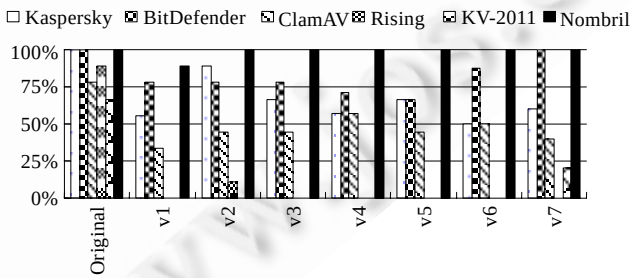


Fig.7 Detection rate of individual confusing methods

图 7 对每种混淆类型的检测率

从图 7 可以进一步看出,常用的反病毒软件对病毒的原始版本检测正确率较高,而对于经过变形、混淆的病毒程序检测率较低.本文提出的检测方法对病毒原始版本和变形版本均有效,且检测准确率最高.

### 4.3 综合测试

测试中随机抽取了 2 000 个正常程序和 2 000 个恶意程序进行测试.为便于对实验结果进行分析,本实验中由以下指标评测 Nombril 系统对程序恶意性的判定能力.

- 恶意程序被判定为恶意程序的情况,称为 TP(true positive);
- 正常程序被判定为正常程序的情况,称为 TN(true negative);
- 正常程序被判定为恶意程序的情况,称为 FP(false positive);
- 恶意程序被判定为正常程序的情况,称为 FN(false negative).

测试指标一般包括检测率、误报率以及综合检测率和错报率的整体检测精度,其计算方式分别为:

- 检测率=$\frac{TP}{TP+FN}$;
- 误报率=$\frac{FP}{TN+FP}$;
- 检测精度=$\frac{TP+TN}{TP+TN+FP+FN}$.

使用 Nombril 系统对测试样本集合中的 4 000 个程序进行检测,程序恶意性分析结果见表 3.

**Table 3** Result of determine malicious program

**表 3** 程序恶意性判定结果

| AV 软件 | *TP* | *TN* | *FP* | *FN* | 检测率(%) | 误报率(%) | 检测精度(%) |
|---|---|---|---|---|---|---|---|
| Kaspersky | 1 925 | 1 996 | 4 | 75 | 96.25 | 0.26 | 98.03 |
| Rising | 1 912 | 1 992 | 18 | 88 | 95.60 | 0.90 | 97.60 |
| ClamAV | 1 653 | 1 955 | 45 | 347 | 82.65 | 2.25 | 90.20 |
| Nombril | 1 977 | 1 979 | 21 | 23 | 98.85 | 1.05 | 98.90 |

在检测率方面,Nombril 的性能最优,比排名第二的 Kaspersky 高出 2.6 个百分点,比排名最后的 ClamAV 高出 16.2 个百分点.这一结果表明,本节提出的基于证据推理的程序恶意性判定是有效的,并且在一定程度上要优于当前主流的安全防护软件.不过,Nombril 在误报率方面的排名却不理想,仅仅是稍强于 ClamAV.这一结果表明,Nombril 在平衡检测的检测率与误报率方面还存在一定的欠缺.这是由于 Nombril 没有针对误报率的问题进行特殊处理,而主流的反病毒软件为低误报率都配备了专用的模块,该模块主要采用白名单方式来减少误报情况的产生.相信 Nombril 在结合白名单等方法之后,误报率高的问题会得到改善.

尽管误报率排名结果靠后,但从检测精度上来看,Nombril 的整体检测性能要优于其他 3 种安全防护软件.

## 5 结束语

由于证据推理模型在证据融合方面具有优势,并且能够区分“不确定”和“不知道”,本文提出了一种基于证据推理的程序恶意性判定模型,并在此基础上设计并实现了可执行程序恶意性分析系统 Nombril.实验结果表明,该系统不但能够有效地对可执行程序的恶意性进行判定,而且对病毒变种判定的准确率也较高.

由于证据推理要求证据之间具有不相关性,一方面,证据的相关性影响判定结果;另一方面,对相关性的处理非常耗时.因此,本文的下一步工作目标是设计并实现更加快速、有效的能够消除证据相关性的程序恶意性推理算法,进一步提高系统的判定能力和效率.

**References**:

[1] Cohen F. Computer viruses: Theory and experiments. Computers & Security, 1987,6(1):22−35. [doi: 10.1016/0167-4048(87)90122-2]

[2] Moskovitch R, Feher C, Elovici Y. A chronological evaluation of unknown malcode detection. In: Chen H, *et al*., eds. Proc. of the Pacific Asia Workshop on Intelligence and Security Informatics (PAISI 2009). LNCS 5477, Heidelberg: Springer-Verlag, 2009. 112−117. [doi: 10.1007/978-3-642-01393-5_12]

[3] Abou-Assaleh T, Cercone N, Keselj V, Sweidan R. N-Gram-Based detection of new malicious code. In: Yau SS, *et al*., eds. Proc of the 28th Annual Int’l Computer Software and Application Conf. New York: ACM Press, 2004. 41−42.

[4] Tian XG, Gao LZ, Sun CL, Zhang EY. Anomaly detection of program behaviors based on system calls and homogeneous Markov chain models. Journal of Computer Research and Development, 2007,44(9):1538−1544 (in Chinese with English abstract). [doi: 10.1360/crad20070912]

[5] Hazem M, El-Bakry. Fast virus detection by using high speed time delay neural networks. Journal of Comput Virology, 2009. [doi: 10.1007/s11416-009-0120-x]

[6] Golovko V, Bezobrazov S, Kachurka P, Vaitsekhovich L. Neural network and artificial immune systems for malware and network intrusion detection. In: Advances in Machine Learning II. SCI 263, 2010. 485−513. [doi: 10.1007/978-3-642-05179-1_23]

[7] Yang FB, Wang XX. Combination Method of Conflictive Evidences in D-S Evidence Theory. Beijing: National Defense Industry Press, 2010. 15−28 (in Chinese).

[8] HayKin S. Neural Network and Learning Machines. 3rd ed., New Jersey: Pearson Education, 2009. 129−140.

[9] Sun Q, Ye XQ, Gu WK. A new combination rules of evidence theory. ACTA ELECTRONICA SINICA, 2008,28(8):117−119 (in Chinese with English abstract).

[10] Wang C, Pang JM, Zhao RC, Liu XX. Using API sequence and Bayes algorithm to detect suspicious behavior. In: Chai S, *et al*., eds. Proc. of the Int'l Conf. on Communication Software and Networks. Washington: IEEE Computer Society, 2009. 544−548. [doi: 10.1109/ICCSN.2009.60]

[11] Zhang BY, Yin JP, Hao JB. Using RS and SVM to detect new malicious executable codes. In: Wang G, *et al*., eds. Proc. of the RSKT. LNAI 4062, Heidelberg: Springer-Verlag, 2006. 574−579. [doi: 10.1007/11795131_83]

**附中文参考文献:**

[4] 田新广,高立志,孙春来,张尔扬.基于系统调用和齐次 Markov 链模型的程序行为异常检测.计算机研究与发展,2007,44(9):1538−1544.

[7] 杨风暴,王肖霞.D-S 证据理论的冲突证据合成方法.北京:国防工业出版社,2010.15−28.

[9] 孙权,叶秀清,顾伟康.一种新的基于证据理论的合成公式.电子学报,2008,28(8):117−119.

**张一弛**(1983－),男,湖北武汉人,博士生,CCF 学生会员,主要研究领域为逆向工程,信息安全.

**赵荣彩**(1957－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为信息安全,软件逆向工程.

**庞建民**(1964－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为逻辑与推理,信息安全.