(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 115587361 A
(43)申请公布日 2023.01.10

(21)申请号 202211264125.6

(22)申请日 2022.10.14

(71)申请人 中南大学
地址 410083 湖南省长沙市岳麓区麓山南路932号

(72)发明人 周芳芳 米佳朋 袁键 房钰深 韩非江 郑天雨 吕胜蓝 赵颖

(74)专利代理机构 成都初阳知识产权代理事务所(特殊普通合伙) 51305
专利代理师 杨继栋

(51)Int.Cl.
G06F 21/56(2013.01)
G06F 21/55(2013.01)

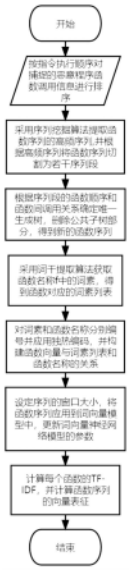
权利要求书2页 说明书7页 附图3页

(54)发明名称

基于词素词向量模型的恶意程序行为表征方法

(57)摘要

本发明公开了一种基于词素词向量模型的恶意程序行为表征方法,包括对捕捉的恶意程序函数调用信息进行排序并抽象,提取高频序列,设置分割点;进行去冗余和去混淆,得到新的函数序列S′;获取函数名称f中的词素,得到函数对应的词素列表M,为非最大长度的词素列表填充标记符Mask,对词素和函数名称分别编号,并对函数名称、词素和占位符的编号应用独热编码,训练词向量模型;计算函数的特征向量,并计算每个函数的TF-IDF。有效解决了动态函数调用中存在的加密、混淆问题,可以做到快速感知理解恶意程序的行为。



CN 115587361 A

1.基于词素词向量模型的恶意程序行为表征方法，其特征在于，包括以下步骤：

步骤1、按指令执行顺序对捕捉的恶意程序函数调用信息进行排序，并抽象为函数序列S；

步骤2、采用GSP序列挖掘算法提取步骤1中函数序列S的高频序列，高频序列对应恶意程序中的热点代码段，在高频子序列之间设置分割点，将函数序列S切割为若干序列段s_i；

步骤3、根据相邻序列段的重合程度，对函数序列S进行去冗余和去混淆，得到新的函数序列S′；

步骤4、对步骤3中函数序列S′中的函数，采用词干提取算法获取函数名称f中的词素，得到函数对应的词素列表M，为非最大长度的词素列表填充标记符Mask，Mask用于填充长度小于最大长度的词素列表，直至每个词素列表长度一致；

步骤5、对词素和函数名称分别编号，并对函数名称、词素和占位符的编号应用独热编码，在模型的编码层中，函数的最终向量w_{vec}与词素列表M和函数名称f有如下关系：

$$w_{vec} = \lambda W_{\cdot}\mathrm{encode}(f) + (1-\lambda)\frac{\sum_{1,m_i\neq \mathrm{Mask}}^{n} W_{\cdot}encode(m_i)}{n}, m_i \in M \quad (1)$$

其中encode(•)为编号和独热编码处理函数，m_i为词素列表M中第i个词素，n为词素列表的长度，W为词向量模型中可训练权重矩阵，λ为超参数；得到的w_{vec}是结合了函数名称的形态信息和函数调用中窗口上下文信息的函数向量；

步骤6、训练词向量模型，词向量模型采用CBOW和Bert模型，在Bert模型中设置上下句的训练任务，两种模型对编码层中函数向量化的逻辑进行调整，即函数的向量编码由步骤5)中的w_{vec}得到；

步骤7、利用训练好的词素词向量模型按照公式(1)计算函数的特征向量，并计算每个函数的TF-IDF，函数序列的向量表征s_{vec}如下：

$$s_{vec} = \frac{\sum_{1}^{len} tf\_idf(f) * w_{vec}(f, M)}{len} \quad (2)$$

s_{vec}即最终的恶意程序行为向量化表征，tf_idf(f)表示函数序列中函数f的TF-IDF值，w_{vec}(f,M)表示按照公式(1)计算得到的函数的特征向量，len表示当前函数序列的函数数量。

2.根据权利要求1所述的基于词素词向量模型的恶意程序行为表征方法，其特征在于，所述步骤1中，函数调用信息包含多条函数调用数据，每一条函数调用数据包含一个调用函数和对应的被调函数、以及执行相对时间，按执行相对时间序列化调用函数和被调函数，其中同次函数调用中，调用函数排序在被调函数之前，函数序列是按指令执行顺序序列化后的函数调用数据，标记为S＝(a_1,b_1)，(a_2,b_2)，…(a_n,b_n)，其中(a_i,b_i)表示第i次函数调用数据中的调用函数和被调函数，i＝1,2…n。

3.根据权利要求1所述的基于词素词向量模型的恶意程序行为表征方法，其特征在于，所述步骤2中，基于序列挖掘算法识别步骤1中的函数调用序列中的频繁序列，并设置分割点，遍历S时在以下任一种情况中设置分割点：

情况1)：在a_1前设置分割点；即在序列起点处设置一个分割点；

情况2)：若当前函数调用组(a_i,b_i)在上一个分割点到前一个函数组中出现了，则在a_i前设置分割点；

情况3)：若当前函数调用组(a_i,b_i)与上一个分割点到前一个函数组在图结构层面不处于同一个连通分量中，则在a_i前设置分割点；

情况4)：若上一个分割点到当前函数调用组(a_i,b_i)是序列挖掘算法识别到的一个频繁序列，则在b_i后设置分割点；

最后根据分割点的位置将函数序列分割为若干序列段s_i，每一个序列段s_i对应恶意程序中的功能块。

4.根据权利要求1或3所述的基于词素词向量模型的恶意程序行为表征方法，其特征在于，所述步骤3包括：

步骤3.1、将每一函数调用组中a_i,b_i用边相连，所有调用组中a_i,b_i用边相连后便确立了唯一生成树，即得到每一序列段对应的调用树t_i，调用树t_i包含功能块中的行为和结构信息；

步骤3.2、对于相邻调用树(t_i,t_{i+1})，$i\in[0,p-1]$，p是序列段的数量，将t_i,t_{i+1}应用于树同构算法，识别相邻调用树t_i,t_{i+1}的公共子树，并删除t_i,t_{i+1}的公共子树部分，前序遍历删除公共子树后的t'_i，得到遍历序列作为更新的序列段s'_i，新的序列段s'_i消除了邻居序列段中的冗余和混淆信息；

步骤3.3、按S中的序列段的顺序对步骤3.2中更新后的序列段a'_i重组，得到去混淆的新的函数序列S′。

5.根据权利要求1或3所述的基于词素词向量模型的恶意程序行为表征方法，其特征在于，所述步骤6包括：

步骤6.1，设定序列的滑动窗口大小，中心词设置为滑动窗口中正中间的函数，窗口中的其余函数作为上下文，然后输入：窗口内每一个函数的词素和函数名称编号的独热编码作为模型输入，即$encode(m_i)$和$encode(f)$；

将所有函数序列拼接为一个长序列，窗口每次向前滑动1步得到一个新的样本，直到滑动窗口移动至长序列尾部结束，得到样本集合，并且不区分训练集和测试集；

步骤6.2，将样本集合作为词素词向量模型的输入，然后编码：$encode(m_i)$和$encode(f)$与编码层中的权重矩阵W相乘后，得到对应的固定长度的向量，将词素列表中每个非Mask的词素对应向量求和并平均，最后与λ加权的函数名称向量相加；

步骤6.3，计算损失：窗口内中心词编码与上下文的差异关系作为模型损失，优化网络参数。

基于词素词向量模型的恶意程序行为表征方法

技术领域

[0001] 本发明属于网络安全和表征学习技术领域,特别是涉及一种基于词素词向量模型的恶意程序行为表征方法。

背景技术

[0002] 恶意程序是指带有攻击性的一类程序,攻击者利用恶意程序完成一些特定的功能,如病毒传播、远程控制、数据窃取和破坏等。在代码编写的过程中,攻击者通过加密、混淆等手段将攻击意图的代码段进行隐藏,这样会增加基于静态代码的对恶意代码的检测难度。

[0003] 应用自然语言处理方法对恶意程序动态执行的函数调用进行向量化表征是一种有效的恶意程序行为表征方法。但是,传统方法在面对带加密、混淆的恶意程序段时,通常不能快速感知和理解样本的恶意行为和典型特征,难以做到有效信息的高效挖掘、关联分析和推理判断,无法为精细化的恶意文件检测、分析、处置、响应和预防提供辅助决策参考。

发明内容

[0004] 本发明实施例的目的在于提供一种基于词素词向量模型的恶意程序行为表征方法,有效解决了动态函数调用中存在的加密、混淆问题,可以做到快速感知理解恶意程序的行为。

[0005] 为解决上述技术问题,本发明所采用的技术方案是,基于词素词向量模型的恶意程序行为表征方法,包括以下步骤:

[0006] 步骤1、按指令执行顺序对捕捉的恶意程序函数调用信息进行排序,并抽象为函数序列S;

[0007] 步骤2、采用GSP序列挖掘算法提取步骤1中函数序列S的高频序列,高频序列对应恶意程序中的热点代码段,在高频子序列之间设置分割点,将函数序列S切割为若干序列段s_i;

[0008] 步骤3、根据相邻序列段的重合程度,对函数序列S进行去冗余和去混淆,得到新的函数序列S′;

[0009] 步骤4、对步骤3中函数序列S′中的函数,采用词干提取算法获取函数名称f中的词素,得到函数对应的词素列表M,为非最大长度的词素列表填充标记符Mask,Mask用于填充长度小于最大长度的词素列表,直至每个词素列表长度一致;

[0010] 步骤5、对词素和函数名称分别编号,并对函数名称、词素和占位符的编号应用独热编码,在模型的编码层中,函数的最终向量w_{vec}与词素列表M和函数名称f有如下关系:

$$w_{vec} = \lambda W_{\cdot}\mathrm{encode}(f) + (1-\lambda)\frac{\sum_{1,m_i\neq \mathrm{Mask}}^{n} W_{\cdot}encode(m_i)}{n}, m_i \in M \quad (1)$$

[0011] 其中encode(•)为编号和独热编码处理函数,m_i为词素列表M中第i个词素,n为词素列表的长度,W为词向量模型中可训练权重矩阵,λ为超参数;得到的w_{vec}是结合了函数名

称的形态信息和函数调用中窗口上下文信息的函数向量；

[0012] 步骤6、训练词向量模型，词向量模型采用CBOW和Bert模型，在Bert模型中设置上下句的训练任务，两种模型对编码层中函数向量化的逻辑进行调整，即函数的向量编码由步骤5)中的w_{vec}得到；

[0013] 步骤7、利用训练好的词素词向量模型按照公式(1)计算函数的特征向量，并计算每个函数的TF-IDF，函数序列的向量表征s_{vec}如下：

[0014] $$s_{vec}=\frac{\sum_{1}^{len} tf\_idf(f)*w_{vec}(f,M)}{len} \quad (2)$$

[0015] s_{vec}即最终的恶意程序行为向量化表征，tf_idf(f)表示函数序列中函数f的TF-IDF值，w_{vec}(f,M)表示按照公式(1)计算得到的函数的特征向量，len表示当前函数序列的函数数量。

[0016] 本发明的有益效果是：首先按执行顺序对恶意程序的函数调用信息进行排序，对函数信息序列化；然后通过序列挖掘算法将函数序列切割为若干序列段，树化每一个序列段，识别并删除相邻树化序列段的公共子树部分；并且采用词干提取方法识别和提取函数名中的词素，采用词素词向量模型计算每个函数的特征向量；最后计算每个函数的TF-IDF，根据TF-IDF对特征向量进行加权，得到恶意程序行为的向量化表征。本发明提供为描述恶意程序的动态行为提供了一种有效的向量化表征方法，实现了将带词素的词向量模型应用于恶意程序表征中，有效解决了动态函数调用中存在的加密、混淆问题，为下游恶意程序的分析、检测和分类打下坚实基础。

附图说明

[0017] 为了更清楚地说明本发明实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本发明的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

[0018] 图1为本发明基于词素词向量模型的恶意程序行为表征方法的流程示意图；

[0019] 图2为词素词向量网络结构；

[0020] 图3为相邻子树公共子树示意。

具体实施方式

[0021] 下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅仅是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0022] 基于词素词向量模型的恶意程序行为表征方法，流程如图1所示，包括以下步骤：

[0023] 步骤1、按指令执行顺序对捕捉的恶意程序函数调用信息进行排序，并抽象为函数序列S；

[0024] 其中，函数调用信息包含多条函数调用数据，每一条函数调用数据包含一个调用函数和对应的被调函数、以及执行相对时间，执行相对时间可以确定每次函数调用发生的前后次序，按执行相对时间序列化调用函数和被调函数，其中同次函数调用中，调用函数排序在被调函数之前，函数序列是按指令执行顺序序列化后的函数调用数据，标记为$S=(a_1,b_1),(a_2,b_2),\cdots(a_n,b_n)$，其中$(a_i,b_i)$表示第i次函数调用数据中的调用函数和被调函数，$i=1,2\cdots n$。

[0025] 步骤2、采用GSP序列挖掘算法提取步骤1中函数序列S的高频序列，高频序列对应恶意程序中的热点代码段，在高频子序列之间设置分割点，将函数序列S切割为若干序列段s_i；

[0026] 其中，基于序列挖掘算法识别步骤1中的函数调用序列中的频繁序列，并设置分割点，具体地，遍历S时在以下任一种情况中设置分割点：

[0027] 情况1)：在a_1前设置分割点；即在序列起点处设置一个分割点，以保证序列S至少存在一个分割点。

[0028] 情况2)：若当前函数调用组(a_i,b_i)在上一个分割点到前一个函数组中出现了，则在a_i前设置分割点；出现了重复的函数调用说明相邻两个序列段可能存在冗余。

[0029] 情况3)：若当前函数调用组(a_i,b_i)与上一个分割点到前一个函数组在图结构层面不处于同一个连通分量中，则在a_i前设置分割点；从而保证步骤3在序列的树化过程中能够生成正常的层次调用树，需要保证序列段可以树化为一个连通分量。

[0030] 情况4)：若上一个分割点到当前函数调用组(a_i,b_i)是序列挖掘算法识别到的一个频繁序列，则在b_i后设置分割点。

[0031] 高级编程语言编写的代码中一般存在多个循环和条件控制语块，而循环的次数对于恶意程序的行为表征的影响较大，在动态函数调用序列中的高频序列是静态代码的循环块的表现，故需要在高频序列之间设置分割点。

[0032] 最后根据分割点的位置将函数序列分割为若干序列段s_i，每一个序列段s_i对应恶意程序中的功能块。

[0033] 步骤3、根据相邻序列段的重合程度，对函数序列S进行去冗余和去混淆，得到新的函数序列S′；

[0034] 具体而言，如图3所示，根据序列段s_i的函数顺序和函数间调用关系确定唯一生成树，并为每一个序列段建树，标记为t_i，根据树同构算法识别相邻t_i的公共子树，并删除公共子树部分，前序遍历删除公共子树的t_i，得到新的序列段s_i，按原顺序重组序列段，得到新的函数序列S′；

[0035] 更具体的说，包括以下步骤：

[0036] 步骤3.1、将每一函数调用组中a_i,b_i用边相连，步骤2中情况3)设置分割点可以保证所有的节点属于同一连通分量，函数调用特性可以保证连通分量是树结构，所有调用组中a_i,b_i用边相连后便确立了唯一生成树，即得到每一序列段对应的调用树t_i，调用树t_i包含功能块中的行为和结构信息；

[0037] 步骤3.2、对于相邻调用树(t_i,t_{i+1})，$i\in[0,p-1]$，p是序列段的数量，将t_i,t_{i+1}应用于树同构算法，识别相邻调用树t_i,t_{i+1}的公共子树，并删除t_i,t_{i+1}的公共子树部分，前序遍历删除公共子树后的t'_i，得到遍历序列作为更新的序列段s'_i，新的序列段s'_i消除了邻居

序列段中的冗余和混淆信息；

[0038] 步骤3.3、按S中的序列段的顺序对步骤3.2中更新后的序列段s'_i重组，得到去混淆的新的函数序列S′。

[0039] 步骤4、对步骤3中函数序列S′中的函数，采用词干提取算法morfessor获取函数名称f中的词素，得到函数对应的词素列表M，为非最大长度的词素列表填充标记符Mask，Mask用于填充长度小于最大长度的词素列表，直至每个词素列表长度一致；

[0040] 其中，采用词干提取算法获取函数名称中的词素，词素是词最小的语法单位，一个词一般由若干个词素组成，由于函数名称可能包含不止一个词，采用词干提取算法前，先采用切词算法把函数名称分割为若干个词，再对每一个词应用于词干提取算法切割为若干个词素，标记为词素列表M。为非最大长度的词素列表填充标记符Mask，直至每个词素列表长度一致。特别地，对于无意义的函数名称不做分词和词干提取处理。

[0041] 步骤5、对词素和函数名称分别编号，并对函数名称、词素和占位符的编号应用独热编码，在模型的编码层中，函数的最终向量w_{vec}与词素列表M和函数名称f有如下关系：

[0042]
$$w_{vec} = \lambda W_{\cdot}\text{encode}(f) + (1-\lambda)\frac{\sum_{1,m_i \neq \text{Mask}}^{n} W_{\cdot}encode(m_i)}{n}, m_i \in M \quad (1)$$

[0043] 其中encode(•)为编号和独热编码处理函数，m_i为词素列表M中第i个词素，n为词素列表的长度，W为词向量模型中可训练权重矩阵，λ为超参数，计算得到的w_{vec}是结合了函数名称的形态信息和函数调用中窗口上下文信息的函数向量。在恶意文件检测中，有相同的词素内置函数和用户自定义函数一般存在相似的行为语义，因此加入函数名称的形态信息有助于恶意文件的检测。

[0044] 步骤5中，词素和函数名称的编号是相互独立，以避免词素和函数名称编号的冲突，保证不冲突的原则下，词素和函数名称的编号随机的，最终向量w_{vec}与词素列表M和函数名称f的关系中，λ超参数用于调整词素编码和函数名称编码之间的权重。

[0045] 步骤6、训练词向量模型，本实施例中词向量模型采用CBOW(continuous bag of words)和Bert(bidirectional Encoder Representation from Transformers)这2种针对词嵌入的模型，并不改变模型中编码层的网络结构，在Bert模型中设置上下句的训练任务，两种模型对编码层中函数向量化的逻辑进行调整，即函数的向量编码由步骤5)中的w_{vec}得到。

[0046] 具体而言包括：

[0047] 设定序列的窗口大小，将S′应用到词向量模型中，并将步骤5中公式(1)应用于词向量模型的编码层中，更新词向量神经网络模型的参数，直至满足训练结束条件；

[0048] 更具体的包括：

[0049] 步骤6.1，设定序列的滑动窗口大小，中心词设置为滑动窗口中正中间的函数，窗口中的其余函数作为上下文，然后输入：窗口内每一个函数的词素和函数名称编号的独热编码作为模型输入，即encode(m_i)和encode(f)；

[0050] 将所有函数序列拼接为一个长序列，窗口每次向前滑动1步得到一个新的样本，直到滑动窗口移动至长序列尾部结束，得到样本集合，并且不区分训练集和测试集。

[0051] 步骤6.2,如图2所示,将样本集合作为词素词向量模型的输入,然后编码:encode(m_i)和encode(f)与编码层中的权重矩阵W相乘后,得到对应的固定长度的向量,将词素列表中每个非Mask的词素对应向量求和并平均,最后与λ加权的函数名称向量相加。

[0052] 步骤6.3,计算损失:窗口内中心词编码与上下文的差异关系作为模型损失。

[0053] 根据步骤6.3计算损失函数值,优化网络参数。

[0054] 步骤7、利用训练好的词素词向量模型按照公式(1)计算函数的特征向量,并计算每个函数的TF-IDF(term frequency-inverse document frequency),函数序列的向量表征s_{vec}如下:

[0055] $$s_{vec}=\frac{\sum_{1}^{len} tf\_idf(f)*w_{vec}(f,M)}{len} \quad (2)$$

[0056] s_{vec}即最终的恶意程序行为向量化表征,tf_idf(f)表示函数序列中函数f的TF-IDF值,w_{vec}(f,M)表示按照公式(1)计算得到的函数的特征向量,len表示当前函数序列的函数数量。

[0057] 其中,TF-IDF是一种评估字词的重要程度的统计方法,计算序列中函数的TF-IDF值可以评估函数对函数序列的重要性,依此给每个函数分配合理的权重,对函数序列进行向量化。

[0058] 实施例1:

[0059] 步骤1、对4份数据集DS1、DS2、DS3和DS4中的后台恶意程序函数调用信息分别排序,4份数据集大小分别为:183、264、518和1061。每个后台恶意程序的函数调用信息包含多条执行记录,每条执行记录的条目有调用函数名、被调用函数名、执行相对时间,抽象为函数序列S。

[0060] 步骤2、采用GSP序列挖掘算法识别S中的高频序列,其中高频序列的最短长度和最低出现频率分别设置为3,4,并在遍历S时在以下任一种情况中设置分割点:(1)在a_1前设置分割点;(2)若当前函数调用组(a_i,b_i)在上一个分割点到前一个函数组中出现了,则在a_i前设置分割点。(3)若当前函数调用组(a_i,b_i)与上一个分割点到前一个函数组在图结构层面不处于同一个连通分量中,则在a_i前设置分割点;(4)若上一个分割点到当前函数调用组(a_i,b_i)是序列挖掘算法识别到的一个频繁序列,则在b_i后设置分割点。最后根据分割点的位置将序列分割为若干个序列段s_i。

[0061] 步骤3、根据相邻序列段的重合程度,对函数S进行去冗余和去混淆,得到新的函数序列S′,具体包含以下步骤:

[0062] 步骤3.1、将每一函数调用组中调用函数与被调用函数用边相连,得到每一序列段对应的调用树t_i。由于在步骤2中设置分割点时已保证连通分量是树结构,所以调用组中调用函数与被调用函数用边相连后便确立了唯一生成树,

[0063] 步骤3.2、对于相邻调用树(t_i,t_{i+1}),i∈[0,p-1],p是序列段的数量,将t_i,t_{i+1}应用于树同构算法,识别相邻调用树t_i,t_{i+1}的公共子树,并删除t_i,t_{i+1}的公共子树部分,前序遍历删除公共子树后的t'_i,得到遍历序列作为更新的序列段s'_i;

[0064] 步骤3.3、按S中的序列段的顺序对步骤3.2)中更新后的序列段s'_i重组,得到去混

淆的函数序列S′。

[0065] 步骤4、对步骤3中S′序列中的函数，采用词干提取算法morfessor获取函数名称f中的词素，得到函数对应的词素列表M，为非最大长度的词素列表填充标记符Mask，直至每个词素列表长度一致。

[0066] 步骤5、对序列S′中的函数名称从1开始编号，对词素从最大函数编号+1起开始编号，Mask占位符编号为0。并对函数名称、词素和占位符的编号应用独热编码，在模型的编码层中，函数的向量w_{vec}与词素列表M和函数名称f存在以下关系：$w_{vec}=\lambda W.encode(f)+(1-\lambda)\frac{\sum_{1,m_i\neq Mask}^{n}W.encode(m_i)}{n},m_i\in M$，λ为超参数，在此实施例中λ＝0.5。

[0067] 步骤6、训练词向量模型，本实施例中词向量模型采用CBOW和Bert 2种针对词嵌入的模型，并不改变模型中编码层的网络结构，两种模型对编码层中函数向量化的逻辑进行调整，即函数的向量编码由步骤5)中的w_{vec}得到。具体包括以下的步骤：

[0068] 步骤6.1)设定序列的滑动窗口大小为5，中心词设置为滑动窗口中第3个函数，窗口中的其余函数作为上下文，在CBOW和Bert模型中仅设置预测中心词的训练任务；

[0069] 步骤6.2)每个窗口内中心词和对应词素列表，以及上下文和对应词素列表编号的独热编码作为一个样本，即encode(m_i)和encode(f)。将所有函数序列拼接为一个长序列，窗口每次向前滑动1步得到一个新的样本，直到滑动窗口移动至长序列尾部结束，得到样本集合，并且不区分训练集和测试集；

[0070] 步骤6.3)将步骤6.2)中的样本集合作为词素词向量模型的输入，模型损失由窗口的中心词与上下文的均方误差计算，学习速率取0.001，训练迭代次数设置为1000。在编码层中，encode(m_i)和encode(f)与编码层中的权重矩阵W相乘后，得到对应的固定长度的向量，将词素列表中每个非Mask的词素对应向量求和并平均，最后与λ加权的函数名称向量相加，优化网络参数。

[0071] 步骤7、利用训练好的词素词向量模型按照公式(1)计算函数的特征向量，并计算每个函数的TF-IDF，函数序列的向量表征s_{vec}如下：$s_{vec}=\frac{\sum_{1}^{len}tf\_idf(f)*w_{vec}(f,M)}{len}$，$s_{vec}$即最终的恶意程序行为向量化表征。

[0072] 为证明本发明的有效性，由步骤7得到的s_{vec}将作为Webshell分类的特征向量，作为Kmeans聚类方法的输入，设置Kmeans的参数k与真实数据的类别数一致，并采用步骤1)中从真实世界得到的4份Webshell函数调用信息作为实验数据。

[0073] 实验例1效果如下：

[0074]

数据集 / 词向量模型	DS1	DS2	DS3	DS4
CBOW	0.823	0.898	0.877	0.840
Bert	0.891	0.893	0.850	0.738

[0075] 表中的数字表示聚类的准确率，由匈牙利最大匹配算法计算得到。

[0076] 为进一步证明本发明的有效性，设置了以下两个辅助实施例进行消融对比实验。

[0077] 实施例2：

[0078] 本实施例包含实施例1中步骤1)4)5)6)7)，不包含步骤2)3)，并且步骤1)4)5)6)7)中参数保持与实施例1一致。该实施例为实施例1简化去混淆版本，实验例2效果如下：

[0079]

数	DS1	DS2	DS3	DS4

[0080]

据集 词向量模型				
CBOW	0.732	0.803	0.734	0.728
Bert	0.781	0.787	0.790	0.724

[0081] 实施例3：

[0082] 本实施例包含实施例1中步骤1)2)3)6)7)，不包含步骤4)5)，并且步骤1)4)5)6)7)中参数保持与实施例1一致，在步骤6中词素词向量模型弱化为词向量模型。该实施例为实施例1的不带词素的词向量版本，实验例3效果如下：

[0083]

数 据集 词向量模型	DS1	DS2	DS3	DS4
CBOW	0.792	0.825	0.794	0.757
Bert	0.743	0.858	0.769	0.701

[0084] 实施例1相比实施例2、3具有较高的聚类准确率，证明本发明的有效性。

[0085] 本说明书中的各个实施例均采用相关的方式描述，各个实施例之间相同相似的部分互相参见即可，每个实施例重点说明的都是与其他实施例的不同之处。尤其，对于系统实施例而言，由于其基本相似于方法实施例，所以描述的比较简单，相关之处参见方法实施例的部分说明即可。

[0086] 以上所述仅为本发明的较佳实施例而已，并非用于限定本发明的保护范围。凡在本发明的精神和原则之内所作的任何修改、等同替换、改进等，均包含在本发明的保护范围内。

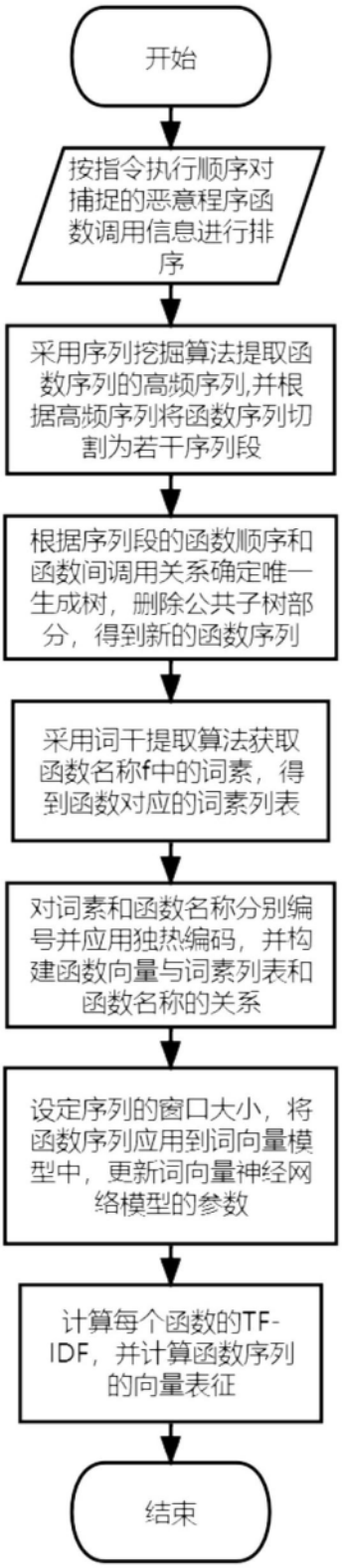

开始
按指令执行顺序对捕捉的恶意程序函数调用信息进行排序
采用序列挖掘算法提取函数序列的高频序列,并根据高频序列将函数序列切割为若干序列段
根据序列段的函数顺序和函数间调用关系确定唯一生成树，删除公共子树部分，得到新的函数序列
采用词干提取算法获取函数名称f中的词素，得到函数对应的词素列表
对词素和函数名称分别编号并应用独热编码，并构建函数向量与词素列表和函数名称的关系
设定序列的窗口大小，将函数序列应用到词向量模型中，更新词向量神经网络模型的参数
计算每个函数的TF-IDF，并计算函数序列的向量表征
结束

图1

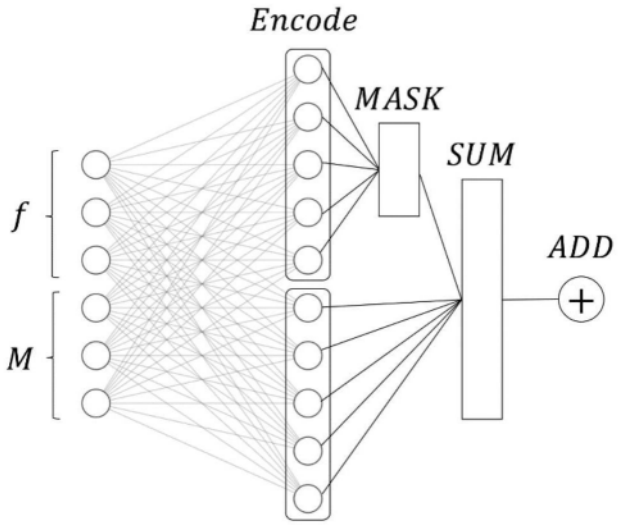


图2

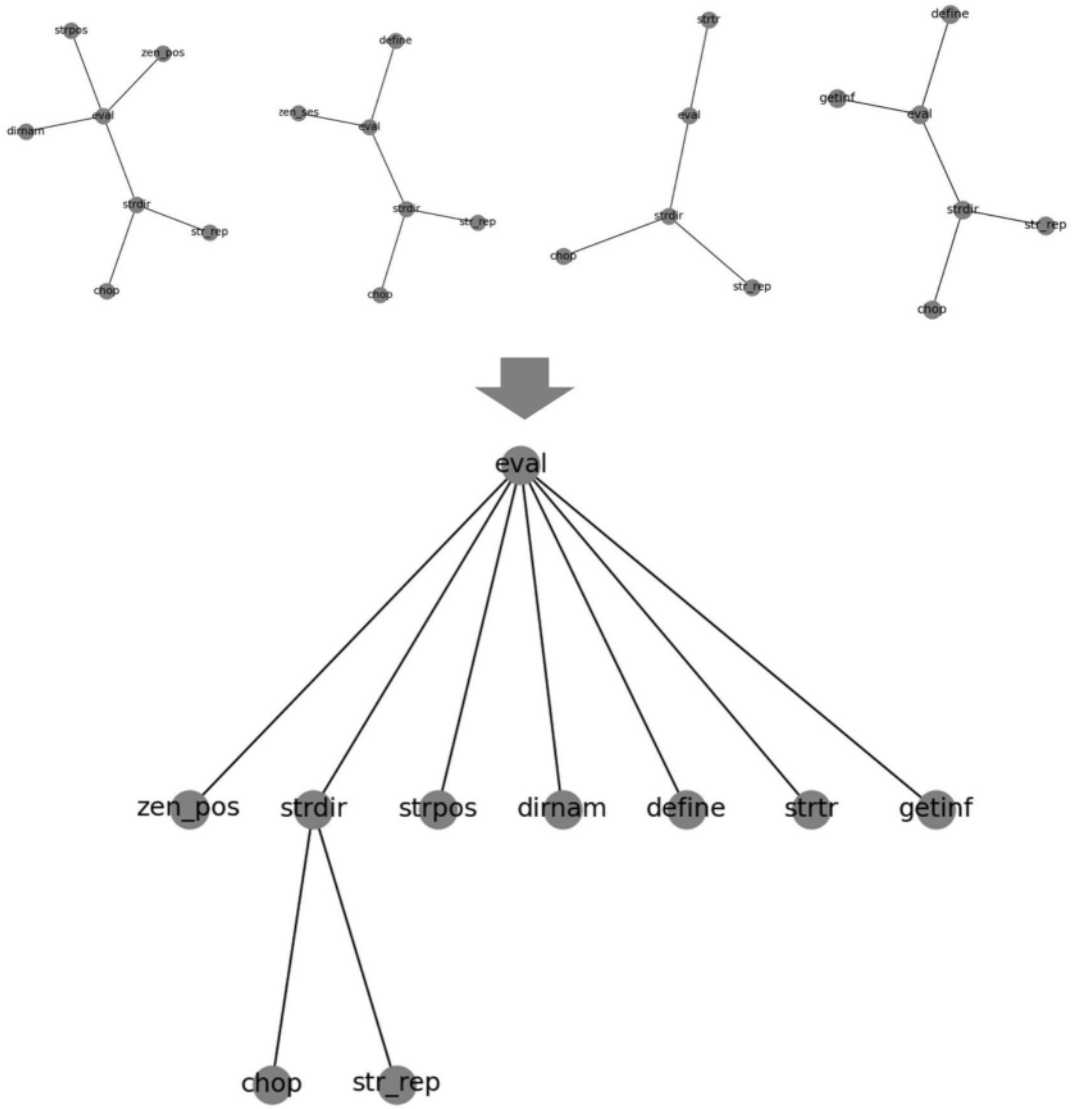

strpos
zen_pos
eval
dirnam
strdir
str_rep
chop
define
zen_pos
eval
strdir
str_rep
chop
strtr
eval
strdir
chop
str_rep
define
getinf
eval
strdir
str_rep
chop
eval
zen_pos
strdir
strpos
dirnam
define
strtr
getinf
chop
str_rep

图3