基于汇编指令分布的恶意代码检测算法研究

王 冰 方 勇
(四川大学电子信息学院 成都 610064)
(349031587@qq.com)

Study of Malware Detection Based on Distribution of Assembly Instruction

Wang Bing and Fang Yong
(*Sichuan College of Electronic and Information, Sichuan University, Chengdu* 610064)

Abstract Nowadays, there are more and more various malicious codes which cause a huge threat to the computer. In this paper, a variety of assembly instructions are classified into three categories, and a novel algorithm is presented. In consideration of the relationship between instructions, this paper turns this relationship into probability distributions as the signature of the program. Then, this paper uses a large amount of samples to train. The experiment results show that the method proposed in this paper has a significant effect and high accuracy on detecting unknown and metamorphic programs.

Key words malware detection, machine learning, information security, assembly instruction, probability distribution

摘 要 如今的恶意代码种类越来越多,形式越来越复杂,对计算机造成了巨大的威胁.将可执行程序的汇编指令进行了分类,只将其中3类指令用于检测,并提出一种创新的算法,考虑到指令前后逻辑关系,将这种关系转换为概率分布,用此分布来代表程序的特征,再对大量样本的这种分布进行聚类,以减少样本的复杂度并且提高效率.这种机器学习的算法经过实验证明,对检测未知的恶意代码以及变形的恶意代码都具有显著的效果,并具有很高的准确率.

关键词 恶意代码检测、机器学习、信息安全、机器指令、概率分布

中图法分类号 TP309.5

随着互联网的发展,恶意代码也逐步盛行,各种蠕虫、病毒、木马在网络中广泛传播,并且种类繁多,恶意代码的作者使用各种手段对它们进行变形、混淆.因此,传统的基于特征匹配的检测方法对于各种变形的恶意代码已经显得有点力不从心.随着机器学习和数据挖掘技术进入到大家的视野之中,已经有许多研究证明将其应用到恶意代码检测中的效果显著,也有许多研究者纷纷提出具有创新性的算法达到了良好的检测效果.比如基于函数调用图表的检测[1-2]、基于行为的检测[3]等等,这些方法通常都需要进行复杂的建模过程,本文在程序的汇编代码中提取特征,然后通过将汇编指令简化达到了简化模型的效果,文献[4-5]均提出了一些将汇编指令翻译成中间语言的方法,的确能够很好地归纳指令的特征,但精炼度还是不够,对翻译后的指令进行分析依然是一项繁琐的工作.因此,本文为了实现更有效

收稿日期:2015-11-30

率的检测，将汇编指令分为5类，然后仅取其中的3类用于检测.

本文首先将汇编代码翻译为简化的中间代码，然后提取汇编代码的特征进行统计分析，得到特征数组. 由于样本数量多，为了简化工作量，本文对样本进行聚类，分别将恶意样本和正常样本聚合为一个数量较少的新的样本簇.

1 汇编指令简化

将汇编指令简化后再进行分析是很多研究常用的方法，它在牺牲了少量精确性的同时，提供了显著的效率提升. 本文根据汇编指令的种类，将其划分为五大类：

1) 数据传输指令，比如：MOV，PUSH，IN等等；

2) 运算指令，其中包含算数运算指令（ADD、SBB等）和逻辑运算指令（OR，XOR，TEST等等）；

3) 程序转移指令，这是指影响程序正常流程的指令，比如：JMP，CALL，RET，INT等；

4) 处理机控制指令，比如：NOP，HLT，WAIT等等；

5) 伪指令，例如：SEGMENT，ASSUME，END等等.

使用IDE获取到程序的汇编指令后，将其转换成以上几类指令，然而，其中的处理机控制指令和伪指令对于分析恶意代码并没有很大影响，在程序中出现的次数非常少，因此本文将其忽略，只分析前3个类别代码，因为在对程序的汇编指令进行分析时发现，出现频率最高的基本都是这3类. 在此分类基础上，本文也对少许指令的分类进行了改动，比如将WAIT指令加入到程序转移指令中等等，表1示出这3种指令的表示方法：

表1 汇编指令的表示

种类	表示方法
数据传输指令	ASSIGN
运算指令	TEST
程序转移指令	JMP

图1是截取样本中的一段汇编指令，将它翻译后指令顺序的表示如图2所示.

```
loc_9645:                    ; CODE XREF: seg000:0000963C j
        push 3F9h
        mov ecx,[ebp-14h]
        call sub_11E8
        mov ecx,eax
        call sub_E91
        test eax,eax
        jz short lov_9664
        mov dword ptr[ebp-74h],1
```

图1 指令样本

Assign→Assign→Jmp→Assign→Jmp→Test→Jmp→Assign

图2 转换后的指令顺序

2 判定算法

文献[6]提出了一种给样本打分的方法，依靠CFG(control flow graph)中指令间指向的频率来对整个CFG进行表示，进而得到特征来区分恶意代码，这种方法取得不错的效率但仍然比较复杂，不利于实际应用. 文献[7]使用汇编代码序列中每个代码的出现频率作为权重来对程序进行检测，该方法虽然简单，但没有考虑整个程序上下的逻辑关系，导致结果准确率并不高，本文提出的方法与该方法有一些共同之处，就是针对程序指令的前后关系来构建一个概率分布，也就是让一个节点指令与其下一条指令组成一个元素，该元素在整个情况下所占的比例将会构成此分布，汇编指令在经过第1节所描述的翻译以及去除无用的指令后，剩下的就是ASSIGN，JMP以及TEST三类指令，因此这个概率分布的元素一共有9种情况分别是：ASSIGN接ASSIGN；ASSIGN接JMP；ASSIGN接TEST；JMP接JMP；JMP接ASSIGN；JMP接TEST；TEST接TEST；TEST接ASSIGN；TEST接JMP. 统计在整个程序中以上9种情况分别出现的次数，例如表2示出了图1这段代码的统计结果. 然后根据次数得到9种情况的概率分布，此概率分布则为代表该程序的特征，本文将其称为模式概率分布，那么表2的模式概率分布则为(1/7,2/7,0,2/7,0,1/7,0,1/7,0).

表2 样本的转换指令连接分布 次

	ASSIGN	JMP	TEST
ASSIGN	1	2	0
JMP	2	0	1
TEST	0	1	0

本文使用 K-最邻近算法来对所得到的概率分布进行距离计算，进而对待检测程序进行判定，具体步骤为：

首先，由于样本数量过多，如果分别将待检测样本与每个训练样本进行计算，效率将会非常低下，并且许多训练样本具有很高的相似性，因此本文会对样本进行聚类，将聚类后的样本（聚类方法在第 3 节中）的模式概率分布与待检测的程序的模式概率分布进行距离计算，文献[8]中的恶意代码检测算法使用了海宁格距离，因为海宁格距离可以很好地度量 2 个概率分布的相似度. 这里也使用海宁格距离公式来进行距离计算：

$$\frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{9}(\sqrt{p_i}-\sqrt{q_i})^2}, \tag{1}$$

其中，p_i 和 q_i 代表的均为模式概率分布内的概率.

然后，将样本与待检测程序的距离从小到大构成一个数组$\{dis_1, dis_2, \cdots, dis_k\}$，$k$ 为整个样本的数量（恶意的和正常的样本聚类后的数量）. 取前 n 个样本进行比较，看其中大多数是属于哪个分类的，如果前 n 个大多数是属于恶意代码样本的，那么就判定待检测样本为恶意代码，反之，则判定其为正常程序.

关于 n 的取值，每个待检测样本得到的值都不一样，遍历距离数组$\{dis_1, dis_2, \cdots, dis_k\}$，当 $dis_i > 2dis_1$ 时，dis_i 与检测样本的相似性就已经非常低，后面更大的距离就都可以忽略，于是取 $n=i$.

3 样本聚类

本文选取的训练样本包含 200 个恶意代码样本、80 个正常样本. 如果将每个样本的模式概率分布和待检测程序进行比较，检测的效率将会非常低，因此，本文会先对恶意样本和正常样本分别进行压缩，以得到一个新的精简的样本簇，用该样本簇进行检测. 将恶意样本和正常样本分别进行下面的操作，这里采用了与 K-最邻近算法类似的聚类算法，聚类的过程如下：

1) 在样本中随机选择一个样本作为第 1 个比较值，假设它的模式概率分布为$(p_1, p_2, \cdots, p_9)$，在所有的样本中找到与它距离最近的样本，假设概率为$(b_1, b_2, \cdots, b_9)$，距离的计算公式仍然为(a). 如果这 2 个模式概率分布之间的海宁格距离小于 K，就算出它们概率分布的平均值作为新的一个样本点. 如果距离大于 K 则不将其合并.

2) 重复步骤 1)，直到每一个样本之间的海宁格距离都大于一个值 K，K 的取值由自己决定，K 越小则聚合的程度越低，得到的检测结果越准确，但是检测过程越复杂效率越低，K 的值越大则聚合度越高，检测结果准确度降低，但检测效率越高. 样本中所有距离小于 K 的样本都会被合并，距离大于 K 则保留为单独的样本.

如果对 K 不进行限制，最后就会分别得到 2 个恶意样本和正常样本模式概率分布的平均值. 也可以用其对检测样本进行检测，此时的检测效率最高. 但是本文为了保证更高的检测准确率，取 $K=0.070\,000$，这个值来自于本文中所有正常样本与恶意样本之间模式概率分布距离的最小值（准确值为 0.072 873），意味着若 2 个模式概率分布的距离大于 0.07 时，就有可能不属于一类，因此不将其进行合并.

所有过程完成后，最后剩下的结果就是用于分类检测的样本.

4 实验数据与结果

本文在实验过程中选取了 200 个恶意代码样本、80 个正常代码样本，都是可执行程序. 其中取 80 个恶意代码样本作为检测样本、120 个恶意代码样本为训练样本. 取 30 个正常代码样本为检测样本、50 个正常样本为训练样本. 实验的平台为 Windows 32 位旗舰版，经过实验统计，本文提出的检测方法得到总的检测准确率为 96.3%，恶意代码检测准确率为 96%，正常程序检测准确率为 96.7%.

由于本文与文献[7]所用的检测方法都涉及到汇编代码的出现频率，因此与文献[7]的检测结果进行对比，其选取了 40 个正常样本和 200 个恶意样本，检测准确率为 90%，因此本文的检测准确率更高，并且由于本文对样本的聚类而使检测效率获得极大提高.

在第 3 节提到本文在对样本进行聚类时，取 $K=0.070\,000$，即所有距离大于 0.070 000（所有距离均保留为小数点后 6 位）的样本都不再进行聚合，聚合

后样本总个数为 27 个，恶意样本 15 个，正常样本 12 个. 图 3 表示了所有待检测样本和聚合后的这 27 个训练样本间的海宁格距离的平均值，虚线代表正常检测样本，实线代表恶意检测样本. 图 3 中的 b1～b15 代表的是恶意训练样本，g1～g12 代表正常训练样本，可以看出，前段 b1～b15 离正常检测样本更远，离恶意检测样本更近，后段 g1～g12 恰好相反，由此可见，此距离算法判定效果非常明显.

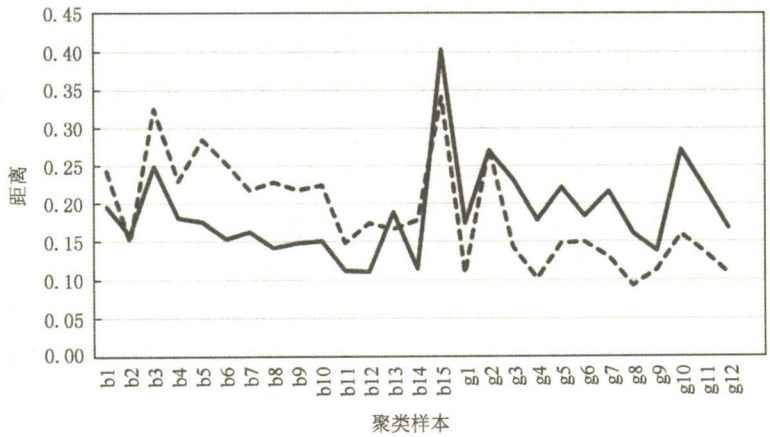

图 3　测试样本和训练样本间的距离均值

下面通过图例来展示整个判定算法的流程，图 4 为实验过程中的一个恶意代码检测样本分别与各聚类后的样本的海宁格距离；图 5 根据检测规则将各个距离从小到大排列.

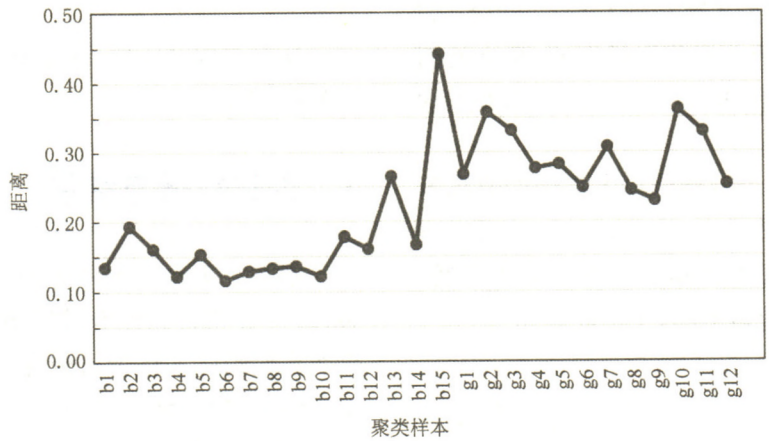

图 4　检测样本的距离

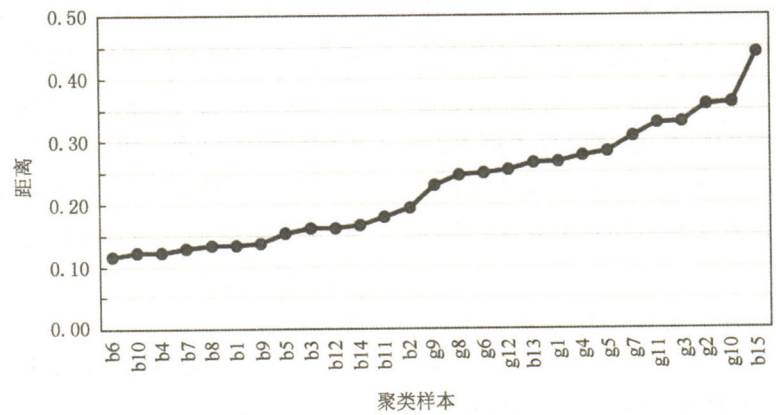

图 5　距离升序排列

图5中，$dis_1=0.116\,815$，其中g8与检测样本的距离为$dis_{15}=0.245\,605$，大于dis_1的2倍，因此选择前15个样本进行比对，其中有13个恶意样本，2个正常样本，随即判定该待检测程序为恶意程序，成功地得到正确的结论. 可以看出，其中一个恶意代码样本点b15是距离待检测程序最远的点，像这种特殊的点就称其为孤立点，在有些聚类算法中会将此类点删去.

5 总　　结

本文提出的检测方法大大简化了对于程序的分析和建模过程，对样本进行有伸缩性的聚类也提供了在精确性和检测效率间的平衡选择，并且经过实验证明，本方法尤其是对于现在流行的变种恶意程序都有着非常好的检测效果，适用于对未知或者变形的恶意代码进行检测.

参 考 文 献

[1] Kinable J, Kostakis O. Malware classification based on call graph clustering [J]. Journal in Computer Virology, 2010, 7(4): 233-245

[2] Hu X, Chiueh T C, Kang G S. Large-scale malware indexing using function-call graphs [C] //Proc of the 16th ACM Conf on Computer and Communications Security. New York: ACM, 2009: 611-620

[3] Bayer U, Comparetti P M, Hlauschek C, et al. Scalable, behavior-based malware clustering [OL]. 2009 [2015-11-10]. http://www.researchgate.net/publication/221655390_Scalable_Behavior-Based_Malware_Clustering

[4] Alam S, Horspool R N, Traore I. MAIL: Malware analysis intermediate language—A step towards automating and optimizing malware detection [C] //Proc of the 6th Int Conf on Security of Information and Networks. New York: ACM, 2013: 233-240

[5] Kranz J, Sepp A, Simon A. GDSL: A universal toolkit for giving semantics to machine language [G] //Programming Languages and Systems. Berlin: Spinger, 2003: 209-216

[6] Runwal N, Low R M, Stamp M. Opcode graph similarity and metamorphic detection [J]. Journal in Computer Virology, 2012, 8(1/2): 37-52

[7] Santos I, Brezo F, Ugarte-Pedrero X, et al. Opcode sequences as representation of executables for data-mining-based unknown malware detection [J]. Information Sciences, 2013, 231(9): 64-82

[8] Wagener G, State R, Dulaunoy A. Malware behaviour analysis [J]. Journal in Computer Virology, 2008, 4(4): 279-287

王　冰

硕士，主要研究方向为恶意代码检测.

349031587@qq.com

方　勇

博士，教授，主要研究方向为信息安全、网络信息对抗.

yfang@scu.edu

大数据相关词条（续）

- MongoDB：是一个基于分布式文件存储的数据库.
- Nginx：开源的高性能HTTP服务器.
- Outlier：见异常点词条.
- 判别分析 (discriminant analysis)：是在分类确定的条件下，根据某一研究对象的各种特征值判别其类型归属问题的一种多变量统计分析方法.
- PU学习：正例和无标记样本学习 (learning from positive and unlabeled examples)，一般称为LPU或PU学习，是一种半监督学习方法.
- Pig：在HDFS和MapReduce上处理大规模数据集的脚本语言，它提供更高层次的抽象并转化为优化处理的MapReduce运算.
- 频繁集 (frequent itemset)：是大于最小支持度的项目集.