概率后缀树在入侵检测中的应用研究

郑 琪[1,2],蒋盛益[2],汤 庸[1]
ZHENG Qi[1,2],JIANG Sheng-yi[2],TANG Yong[1]

1.中山大学 中山医学院,广州 510080
2.广东外语外贸大学 信息科学与技术学院,广州 510006
1.Sun Yat-sen College of Medical Science,Sun Yat-sen University,Guangzhou 510080,China
2.School of Informatics,Guangdong University of Foreign Studies,Guangzhou 510006,China
E-mail:zq@mail.gdufs.edu.cn

ZHENG Qi,JIANG Sheng-yi,TANG Yong.Applying probabilistic suffix tree to intrusion detection.Computer Engineering and Applications,2010,46(23):79-81.

Abstract: System call trace is one of the behavior characters of system process.Each system call of the trace depends on a few previous system calls.Thus,probabilistic suffix tree is used to model the system call trace,and capture the probabilistic characteristic of the system call.Two definitions of abnormal metric are given.When detecting the abnormal trace,train the PST with normal system call traces,and then calculate the abnormal metric of each trace,which is used to compare with a given limit.Experiment shows that this measurement can well distinct normal process from abnormal process.
Key words: intrusion detect;system call trace;probabilistic suffix tree

摘 要: 系统调用序列能够反映系统进程的行为特征。而系统调用序列中每个调用的出现都与它之前出现的若干个调用相关。因此可以利用概率后缀树(PST)对系统调用序列建模,反映系统调用基于上下文的概率特性。提出了系统调用序列异常度的定义。在进行序列的异常检测时,先利用正常系统调用序列训练PST模型,然后通过该模型,利用计算未知系统调用序列的异常度,根据给定的阈值判断该序列是否异常。实验表明这一度量对于正常进程与异常进程有着良好的区分效果。
关键词: 入侵检测;系统调用序列;概率后缀树
DOI:10.3778/j.issn.1002-8331.2010.23.022 **文章编号:** 1002-8331(2010)23-0079-03 **文献标识码:** A **中图分类号:** TP393.08

1 引言

基于网络的计算机系统在当今得到了广泛的应用。这些系统在提供服务的同时,也面临着各种各样的攻击和入侵的危险。因此我们有必要去找到可以检测入侵的方法,便于采取进一步的措施保护我们的系统。很多学者提出和实现了不同的检测方案,这些方案分别定义了不同的区别正常和非正常进程的特征。1996年,Forrest等[1]提出了通过监测活动特权进程的系统调用检测入侵的方法。他们认为一个正常特权进程的系统调用序列与异常进程的调用序列有着不同的模式,并且可以用系统调用的短序列描述。根据这一思想,学者们尝试采用不同的方法来描述正常进程的这种模式。Ye[3]和张响亮[4]分别利用马尔可夫链和隐马尔可夫模型通过概率特征来反映这一模式。低阶的马尔可夫模型精度也不是很高。但是高阶马尔可夫模型随着阶次的增加,时间复杂度将以指数方式增加。隐马尔可夫模型也存在着学习时间代价较高的问题[4-5]。因此,利用一种可变阶的马尔可夫模型-概率后缀树[5](Probabilistic Suffix Tree,PST)来解决上述存在的问题。

2 概率后缀树

概率后缀树PST用于描述一个序列集合的概率特征。在一个符号集上的PST是一个非空的树。树上每个节点的出度在零(叶子节点)和符号集大小之间。树上的每一条边用符号集中的一个符号标记。从一个节点出来的两条边不能有相同的标记。因此每一个节点的出度最多等于符号集的大小。树的节点用一个字符串标记。节点的标记通过从这个节点回溯到根节点所经过的边的标记生成。每个节点都有一个概率分布向量。根节点概率向量是符号集中每个符号的无条件概率。其他节点的概率向量是该节点的标记符号序列的下一个符号出现的条件概率向量。

符号序列下一个符号的概率是通过统计学习样例中符号

基金项目: 国家自然科学基金(the National Natural Science Foundation of China under Grant No.60673191);广东省高等学校自然科学研究重点项目(No.06Z012)。
作者简介: 郑琪(1973-),男,讲师,主要研究领域为数据挖掘,网络安全,生物信息学;蒋盛益(1963-),男,博士,教授,主要研究领域为数据挖掘;汤庸(1964-),男,博士,教授,主要研究领域为时态数据库,协同。
收稿日期: 2009-05-12 **修回日期:** 2009-07-29

出现相对频率$P(\sigma|s)$得到的。其中s代表符号序列,σ代表某个符号。$P(\sigma|s)=|s\sigma|/|s^*|$。$|s\sigma|$表示样例中序列$s\sigma$出现的次数。$|s^*|$表示序列$s$与任意一个符号构成的序列出现的次数。例如$P(b|a)=|ab|/|a^*|=2/4=1/2$。图1表示一个通过样例“abracadabra”学习得到的二阶的PST。

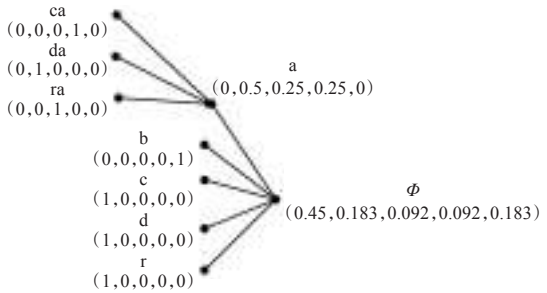

图1 “abracadabra”训练得到的PST

PST的构造算法Build-PST($P_{\min},\gamma,r,L$)如下:

(1)初始化PST只包含一个根节点,根结点的概率向量值为每个符号在所有符号序列中出现的相对频率。将所有相对频率超过阈值$P_{\min}$的符号作为候选子节点。

(2)递归扩充每一个候选节点

①计算该候选节点的后续符号概率向量。

②设候选节点的标记字符串为s。如果存在一个$\sigma\in\Sigma$,$P(\sigma|s)>\gamma$,并且$P(\sigma|s)/P(\sigma|\text{suff}(s))>r$或者$P(\sigma|s)/P(\sigma|\text{suff}(s))<1/r$,则添加该子节点到树中。其中suff($s$)代表候选节点父节点的标记。

③如果该节点的深度小于PST的最大深度L,则对于每个$\sigma\in\Sigma$,如果σs的相对频率超过$P_{\min}$,则标记为σs的节点作为该节点的候选子节点。

PST构造算法的时间复杂度为$O(Ln^2)$,空间复杂度为$O(Ln)$。其中n代表训练序列的总长度。PST构造的时间复杂度可以优化为$O(n)$[6]。PST构造好后,计算$P(\sigma|s)$的时间复杂度为$O(L)$。

3 异常检测

将正常进程训练集中的系统调用序列通过滑动窗口方法转换为长度为L的调用子序列的集合,利用子序列集合构造L阶PST,用以记录正常进程调用序列的概率特征。一个进程可以划分为完成多个独立子功能的不同长度的调用子序列,也就意味着与一个系统调用有关联的前驱系统调用个数是不同的。一个系统调用出现所依赖的前驱系统调用的个数是与上下文相关的。PST本质上是一个变阶的马尔可夫模型。利用正常系统调用子序列集合训练构建PST的过程,在某种程度上可以反映系统调用序列中的这一特点。可以利用构建好的PST计算调用序列中每个调用基于若干前驱调用出现的概率$P(\sigma_i|\sigma_{i-k}...\sigma_{i-2}\sigma_{i-1},k\leq L)$。其中$L$为PST的最大深度。

在进行系统调用序列的异常检测时,利用构建好的PST模型计算调用序列的异常度,如果超过给定的阈值,则认为该调用序列异常。系统调用序列$\sigma_1\sigma_2...\sigma_N$的异常度的定义可以采用两种方法。

定义1 对于给定系统调用序列$\sigma_1\sigma_2...\sigma_N$,异常度为异常子序列所占所有子序列的比例,即异常度:

$$\lambda=\frac{\sum_{i=1}^{N-L+1}MM(\sigma_i\sigma_{i+1}\cdots\sigma_{i+L-1})}{N-L+1}$$

其中,$MM(\sigma_i\sigma_{i+1}\cdots\sigma_{i+L-1})=\begin{cases}1, \text{若}P(\sigma_i\sigma_{i+1}\cdots\sigma_{i+L-1})<\mu_{\min}\\0, \text{否则}\end{cases}$,$\mu_{\min}$为一给定的阈值,$P(\sigma_i\sigma_{i+1}\cdots\sigma_{i+L-1})=P(\sigma_i)P(\sigma_{i+1}|\sigma_i)\cdots P(\sigma_{i+L-1}|\sigma_i\sigma_{i+1}\cdots\sigma_{i+L-2})$。

文献[1-3]采用了定义1的方法。这里给出调用序列异常度的另外一种定义方法。

定义2 对于给定系统调用序列$\sigma_1\sigma_2...\sigma_N$,异常度为异常的系统调用个数与系统调用总数的比值,即异常度:

$$\lambda=\frac{\sum_{i=1}^{N}MM(\sigma_i)}{N}$$

其中$MM(\sigma_i)=\begin{cases}1, \text{若}P(\sigma_i|\sigma_{i-k}\cdots\sigma_{i-2}\sigma_{i-1})<\mu_{\min}\\0, \text{否则}\end{cases}$,$\mu_{\min}$为一给定的阈值,$k\leq L$($L$为PST的最大深度),且$k$的大小与系统调用的上下文相关,由PST决定。

计算概率$P(\sigma_i|\sigma_{i-k}\cdots\sigma_{i-2}\sigma_{i-1})$的方法如下,其中$\sigma_i$为序列中某一调用,它的前缀为$\sigma_{i-k}\cdots\sigma_{i-2}\sigma_{i-1}$。从根节点出发,沿依次与$\sigma_{i-1}\sigma_{i-2}\cdots\sigma_{i-k}$匹配的边访问PST中节点,到达叶子节点或者到达标记为$\sigma_{i-k}\cdots\sigma_{i-2}\sigma_{i-1}$的节点时停止。根据到达节点的后续符号概率向量,得到调用σ_i的概率。例如,利用正常调用序列构造了如图1的PST。给定调用序列“acd”,利用d的前缀“ac”访问PST,到达叶子节点“c”,所以$P(d|ac)=0$。如果给定序列“cad”,可以到达节点“ca”,所以$P(d|ca)=1$。

检测一个未知的调用序列时,根据定义1或者定义2计算它的异常度,如果该调用序列的异常度λ高于某个阈值$\lambda_{\min}$,则认为该调用序列是异常的。其中计算值中设定的阈值$\mu_{\min}$和$\lambda_{\min}$可以根据实际检测的要求进行估计。未知调用序列检测的时间复杂度为$O(Ln)$。

例如设定$\mu=0.5$,$\lambda=0.5$,利用图1的PST和定义2的异常度计算方法对未知序列进行检测。对于序列“abrabr”,每个调用的概率依次为{5/11,1/2,1,1,0,1},它的异常度是2/6=0.22,小于0.5,认为正常。对于序列“abcabc”,每个调用的概率依次为{5/11,1/2,0,1,0,0},它的异常度为4/6=0.67,大于0.5,认为异常。

4 实验结果

实验采用墨西哥大学(UNM)采集的sendmail进程在正常运行时产生的系统调用数据和入侵进程产生的数据[7]进行测试。数据集描述见表1。

表1 UNM sendmail数据特征表述

	数据集名称	调用次数	进程个数
正常数据	sendmail	19 526	147
异常数据	Local 1	1 516	6
	Local 2	1 574	6
	Remote 1	1 861	7
	Remote 2	1 553	4
	Sm565a	275	3
	Sm5x	1 537	8

首先将正常进程系统调用序列的1/2用于训练,正常进程系统调用序列的另外1/2和异常进程系统调用序列用于测试。利用训练集构造PST的参数为$P_{\min}=0.001$,$\gamma=0$,$r=1.05$。PST的最大深度L分别取3~20。为了检验这种方法对正常序列和异常序列的总体区分度,利用构造好的PST模型计算不

同测试集中每个调用基于上下文的出现概率的平均值。具体的实验结果如图2:

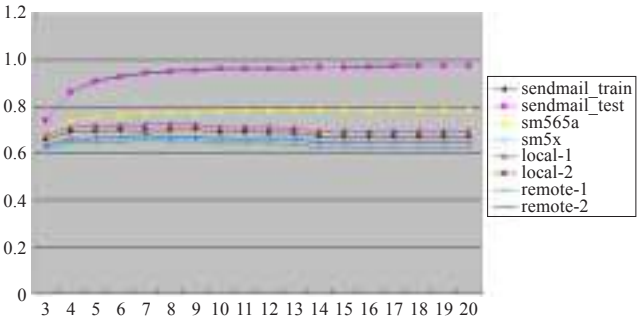

图2　利用不同阶次的PST测试各个数据集的系统调用概率平均值

通过图2可以看出当PST的深度L>10时正常测试集的平均概率不再显著地提高,当L>14时异常测试集的平均概率不再显著地降低。正常测试集的平均概率明显高于异常测试集的平均概率。

取PST深度L=10、11、14、15,正常概率阈值μ_{min}=0.001,分别计算正常测试集和异常测试集的系统调用的总体异常度,并与文献[4]的结果进行了对比。结果见表2。

表2　测试集的总体异常度及与文献[4]方法的结果比较

	文献[4]		本文方法					
			定义1			定义2		
	k=10	k=11	L=10	L=11	L=14	L=10	L=11	L=14
正常测试数据	0.35	0	4.24	4.67	5.99	0.40	0.40	0.42
local_1	16.70	6.31	45.88	46.97	55.03	15.59	15.59	20.60
local_2	19.20	6.39	47.95	49.01	56.78	17.04	17.04	21.88
remote_1	25.60	14.32	54.49	55.44	62.45	21.11	21.13	25.69
remote_2	21.70	11.73	55.19	56.06	62.19	19.96	19.95	23.78
sm565a	27.70	4.91	33.88	33.88	34.76	12.24	12.40	13.73
sm5x	28.10	2.75	67.19	68.78	73.12	20.41	20.43	20.82

由表2可知,PST模型可以很好地区分正常和异常的系统调用序列。在检测异常的过程中,异常度定义2的区分效果要好于定义1。PST模型的区分效果与文献[4]中基于HMM模型的效果基本相同。

PST模型构建算法的时间复杂度经优化后为$O(n)$。其中n代表训练序列集中的系统调用总数,本实验中n=19 526。HMM模型通常采用Baum-Welch方法学习[8],时间复杂度为$O(mk^2n)$。其中m为算法求得局部最优解需迭代的次数;k代表HMM中不同的状态数,如果k值过小会影响HMM方法的准确率,文献[4]中k=53。在检测未知序列时,PST方法为$O(Ln)$。HMM模型通常利用前向算法,时间复杂度为$O(k^2n)$。由此可见PST模型构建和检测未知序列所需的时间复杂度远小于HMM模型,更适合对未知序列进行动态实时监测。

5　结论和进一步的工作

基于正常进程的系统调用序列与异常进程的系统调用序列有着不同的特征,提出利用PST,一个变阶的马尔可夫模型,描述正常进程系统调用的上下文概率特征。给出了序列异常度的两种定义,在检测系统调用序列时,根据待检测序列的异常度是否超过给定阈值来判断该序列是否异常。实验证明这种方法可以很好地区分正常和异常的系统调用序列,检测效果与HMM模型相当,而PST模型训练的时间复杂度要远远优于HMM模型,便于及时更新模型以反映正常系统调用序列有可能随时间变化的概率特征。今后的研究工作中,考虑如何根据正常系统调用序列的变化增量更新PST,实时在线反映系统进程的正常变化,以更好地与异常系统进程区分。同时希望将其他可变阶的马尔可夫模型,如CTW、PPM[5],应用于异常系统调用序列检测,并且与PST方法进行区分度与效率的比较。

参考文献:

[1] Forrest S,Hofmeyr S A.A sense of self for Unix processes[C]//Proceedings of the 1996 IEEE Symposium on Security and Privacy,Oakland,USA.IEEE Computer Society Press,1996:120-128.

[2] Ye N.A Markov chain model of temporal behavior for anomaly detection[C]//Proceedings of the 2000 IEEE Systems,Man,and Cybernetics Information Assurance and Security Workshop,United States Military Academy,West Point,NY.IEEE Computer Society Press,2000:171-174.

[3] 张响亮,王伟,管晓宏.基于隐马尔可夫模型的程序行为异常检测[J].西安交通大学学报,2005,39(10):1056-1059.

[4] Bejerano G,Yona G.Variations on probabilistic suffix trees:statistical modeling and prediction of protein families[J].Bioinformatics,2001,17(1):23-43.

[5] Begleiter R,El-Yaniv R,Yona G.On prediction using variable order Markov models[J].Journal of Artificial Intelligence Research,2004,22:385-421.

[6] Apostolico A,Bejerano G.Optimal amnesic probabilistic automata or how to learn and classify proteins in linear time and space[C]//Proceedings of the Fourth Annual International Conference on Computational Molecular Biology,Tokyo,Japan.ACM,2000:25-32.

[7] Computer immune systems data sets[EB/OL].http://www.cs.unm.edu/~immsec/data/synth-sm.html.

[8] Rabiner L R.A tutorial on hidden Markov models and selected applications in speech recognition[J].Proceedings of the IEEE,1989,77(2):257-289.