基于时间戳马尔可夫模型的入侵检测技术研究

谷胜伟, 宋如顺

(南京师范大学 数学与计算机科学学院, 江苏 南京 210097)

[摘要] 入侵检测是保障网络安全的重要技术. 在改进 LPMC 算法的基础上提出了 LPMCST (Linear Prediction and Markov Chain With Time Stamp)算法. LPMCST算法采用时间戳标识, 对特权进程的系统调用序列进行分段训练和检测, 特别是在系统调用序列波动较大的情况下, 使得模型更能反映系统实时状态, 从而在保持原算法优点的基础上进一步降低了误报率和漏报率, 提高了检测的准确度.

[关键词] 入侵检测系统, 马尔可夫模型, 系统调用, 时间戳, 异常检测

[中图分类号] TP393 08 [文献标识码] A [文章编号] 1672-1292(2008)01-0080-04

Research on Technology of Intrusion Detection Based on Markov Model With Time Stamp

Gu Shengwei, Song Rushun

(School of Mathematics and Computer Science, Nanjing Normal University, Nanjing 210097, China)

Abstract Intrusion detection is an important approach for protecting network security. In this paper, we propose a new algorithm LPMCTS (Linear Prediction and Markov Chain with Time Stamp) which is based on LPMC. LPMCTS employs time stamps to mark the system call sequences of the privileged processes during training and detection. It reflects system real time state better than LPMC, especially on fluctuate situation, so that we get lower false negative rate and false positive rate and promote the success probability of detection while keeping the advantages of the original algorithm.

Key words IDS; Markov model; system call; time stamp; anomaly detection

入侵检测技术可分为误用检测和异常检测两类. 误用特征检测是根据入侵或攻击的知识来检测入侵, 即当前的行为模式与特征库中的入侵特征是否匹配来检测是否发生了入侵. 异常检测是根据系统在正常状态下建立的模型, 用比较系统的当前行为与正常状态的偏移来判断是否发生了入侵行为. 误用特征检测需要大量的入侵知识或特征建库, 对已知的入侵检测比较可靠, 但是对未知的入侵则无能为力. 而异常检测能够检测到未知的入侵, 但是通常误报率和漏报率都很高.

1996年, Forrest提出的基于特权进程系统调用的正常模式进行入侵检测, 并伴随着近几年来机器学习的发展, 提出了许多基于统计的算法及模型, 其中马尔可夫模型在监视特权进程的系统调上要比其它模型要好些. 马尔可夫模型应用于入侵检测的基本思想是, 根据系统正常状态的转移概率与系统实时状态的转移概率作比较来判断是否发生了入侵行为.

1 相关研究

1.1 马尔可夫模型(MM)概述

马尔可夫模型是一个随机模型, 它假定系统将来的变化只与系统现在的情况有关, 而与过去的无关. 即: $P\{X(t_{n+1}) = i_{n+1} \mid X(t_1) = i_1, X(t_2) = i_2, \cdots, X(t_n) = i_n\} = P\{X(t_{n+1}) = i_{n+1} \mid X(t_n) = i_n\}$. 式中,

收稿日期: 2007-05-20
基金项目: 国家 十五 211工程 建设基金(181070H901)资助项目.
作者简介: 谷胜伟(1982-), 硕士研究生, 研究方向: 信息网络安全保密技术. E-mail: gushengwei0328@163.com
通讯联系人: 宋如顺(1953-), 教授, 研究方向: 信息网络安全保密技术. E-mail: rssong@njnu.edu.cn

$\{X(t_n), t_n \quad 0\}$ 是随机变量序列, $I = \{i_n, n \quad \mathbf{N}\}$ 是状态空间.

马尔可夫模型可以表示成一个三元组 $= (I, P, \quad)$, 其中 I 是状态空间, P 表示状态间的转移概率, 表示模型的初始概率[1].

1.2 基于 MM 的入侵检测技术的研究及不足

利用马尔可夫模型进行入侵检测的研究一般分为以下几个步骤:

(1)建立一个马尔可夫模型来描述一个系统的正常行为, 该模型的各个状态对应于系统的各个状态.

(2)利用系统的观测 (或经过预处理过的)数据, 来训练模型并计算模型的参数.

(3)利用训练好的模型, 应用系统状态转移所出现的转移概率来判决系统是否符合正常系统的状态转移概率.

由上可知, 第一步是获取训练数据. 数据的获取一般有两个途径, 一是系统的日志或审计数据. 这种数据获取一般比较简单. 但是数据量大、冗余度大, 所以必须进行预处理; 其次日志或审计数据只是记录了系统运行的表面现象, 并未揭示系统的本质特征, 不利于特征的提取. 二是特权进程的系统调用序列. 这是大多数马尔可夫模型采用的数据源. 因为特权进程的系统调用与用户进程相比是特定的、有限的、稳定的, 其运行过程中功能转换和衔接也是有限和稳定的[2]. 模型的训练算法包括: LPMC(Linear Prediction and Markov Chain)算法[2]、Baum-Welch算法[3]等.

LMPC算法在很大的程度上改进了 Forrest提出的算法. 但由于在建立及训练模型时, 总是基于 Markov 状态转移的时齐性或平稳性, 即 Markov模型的状态转移是与时间无关的. 而在实际系统中, 这个条件是不满足的. 因为不同的时间, 特权进程的系统调用序列是有差异的.

针对上述问题, 本文对 LMPC方法进行了改进, 提出了带有时间戳的 LMPCTS算法.

2 LPMCTS算法

2.1 LPMCTS算法的原理

LPMCTS算法建立模型时对不同时间段的训练数据进行了划分, 得到一个关于不同时间段的 Markov 模型, 即: $= (\ _1, \ _2, \ , \ _n)$, 对于不同时间段调用不同的参数 $\ _i$, 能更准确地描述系统的实时状态. 相对于 LPMC算法, LPMCTS算法修改了各个时间段参数的初值选取及入侵的判定标准. 关于 LPMC算法的算法描述参见[4]. LPMCTS算法原理如下.

(1) 对时间段 T 进行划分. T 是一个单位时间段, 为了一般性和简单性起见, 文中 T 代表一天. 令 $T = \{T_1, T_2, \ , T_m\}$, 即把时间段 T 划分为 m 段.

(2) 对特权进程发起的系统调用, 按照(1) 给出的时间划分标准进行初次分类, 将在同一时间段的系统调用分为一类. 令 $O = \{O_1, O_2, \ , O_s\}$ 为正常系统调用集序列构成的训练集合, 初次分类后的集合为 $O_{T_i} = \{O_{T_i1}, O_{T_i2} \ , O_{T_ik_i}\}$, O_{T_i} 是对应于时间段 T_i 的系统调用序列.

(3) 通常在一个较短的时间中, 时间对系统调用序列的转换差异产生很小的影响, 因而对每个时间段, 文中假设是时齐的(或稳定的), 对集合 O_{T_i} 进行如下训练, 获得 $\ _i$:

(i) 设 O_{T_i}, w 为滑动窗口(window s)的大小, 用 w 分割 , 步长 $= 1$, 则 被划分成 $l = | \ | - w + 1$个子序列, 即 $= (\ _1, \ _2, \ , \ _l)$. 利用 Levinson-Durbin算法[5] 对序列 进行维数压缩, 求得每个子序列对应的 LP系数, 若 $\ _k$ 对应的 LP系数为 $\ _k = (\ _{k1}, \ _{k2}, \ , \ _{kd})$, $1 < d < w$. 则 对应的 LP系数为 $(\ _1, \ _2, \ , \ _l)$.

(ii) 把 的 LP系数作为相应的特征向量建库(剔除重复记录), 则对应于 O_{T_i} 的正常状态空间 $I = \{\ _1, \ _2, \ , \ _n\}$, 库的大小由特征向量数决定.

(iii) 由训练集合计算状态间的转移概率矩阵 $\boldsymbol{P}$, 算法如下:

Step 1: 设 $\boldsymbol{P} = (p_{ij})_{n \ n} = 0$, 对训练集合的每个系统调用序列特征向量 $\ _i$ 执行 Step 2.

Step 2: 用计数器记录每个状态出现的次数 N_i, 状态转换次序及次数 N_{ij}.

Step 3: 用状态的转移频率近似状态间的转移概率, $p_{ij} = \dfrac{N_{ij}}{N_i}$.

因为训练集合不可能包括所有的正常系统调用, 为了系统扩展的方便, 文中引入了一个附加状态 $\ _0$,

并令 $_{i0} = {}_{0i} = $. 因为附加状态在训练过程中没有出现过, 所以 的值很小, 令 $= \min(p_{ij})/10^{[3]}$.

(iv) 模型的初始状态概率 $= ({}_1, {}_2, \ , {}_n)$, 因为初始概率是系统的初始时刻位于各个状态的概率, 用初始频率来近似初始概率, 即 $_i = \dfrac{N_i}{N_j}$.

(4) 检测方法. 检测的基本思想是建立在系统调用正常状态的转移概率上的. 在检测中, 与建立马尔可夫模型过程一样, 求得需要检测序列的连续 L 个状态, 并求得 L 个连续转换序列的转换概率: $p({}_{s-L+1}, {}_{s-L}, \ , {}_s) = {}_{s-L} \prod_{i=s-L+1}^{L} p_{i-1\ i}$ $(0 < L < s)$ 作为判断值, 与在此时间段的门限值 $_i$ 作比较, 得到比较结果.

2 2 *LPMCTS* 训练算法

输入训练集合, 输出各个时段的模型参数:

(1) T $\{T_1, T_2, \ , T_m\}$.

(2) 主迭代:

For each O_{T_i} $i = 1, 2, \ , m$

$P = (p_{ij})_{n\ n}$, $_i = 0$, count[n] = 0, num[n][n] = 0, windows = w;

For each O_{T_i}

$= ({}_1, {}_2, \ , {}_{|\ |-w+1})$

$= ({}_1, {}_2, \ , {}_{|\ |-w+1}) = LP({}_1, {}_2, \ , {}_{|\ |-w+1})$

$({}_1, {}_2, \ , {}_n) = ({}_1, {}_2, \ , {}_l)$;

For each 系统序列的 LP 变换 :

if($_i$) count[i] = count[i] + 1

if($_i$ $_j$) num[i][j] = num[i][j] + 1

$p_{ij} = \dfrac{\text{num}[i][j]}{\sum_j \text{num}[i][j]}$, $_i = \dfrac{\text{count}[i]}{\text{count}[i]}$.

(3) 结束, 输出结果.

2 3 *LPMCTS* 检测算法

对某特权进程的系统调用序列 , 都划分为连续 L 个状态序列, 求其转换概率:

(1) For $i = 1, 2, \ , m$

if(T_i) = $_i$

(2) 对 进行如上 LPMCTS训练算法处理, 并

$$p_{ij} = \begin{cases} p_{ij} & \text{若从状态 } i\text{到状态 } j\text{的转换包含在训练集合中,} \\ & \text{若从状态 } i\text{到状态 } j\text{的转换不含在训练集合中;} \end{cases}$$

(3) 对 的如下的每 L 个连续状态转换概率计算:

$_0 = p({}_{s-L+1}, {}_{s-L}, \ , {}_s) = {}_{s-L} \prod_{i=s-L+1}^{L} p_{i-1\ i}$;

if($_0 > {}_i$) 为正常调用

else 为非法调用.

(4) 结束, 输出检测结果.

3 LPMCTS算法参数设定及性能分析

3 1 参数 *T* 确定

参数 T 的划分对提高 LPMCTS的检测准确性很重要. 在实际中, 在一个程序运行过程中的不同时间段上, 系统调用序列的分布情况往往是不同的, 而且序列之间的相关性也与时间有关, 特别是在序列相互相交的情况下[6].

参数 T 的确定没有精确的原则, 以下是一些通用的原则:

(1)T的划分应该能够使得 L 个连续状态的转换概率在各个时段有明显的区别.

(2)T的划分应该充分考虑服务器的访问策略、客户机的访问时段及请求服务类型的差异.

(3)T的划分应考虑各个程序在系统调用序列上的相互依赖关系.

基于以上原则和系统实际运行服务的统计数据可以对 T 进行划分, 如某一部门的服务期可初步分为正常工作时间、系统备份时间及非工作时间. 其次参考制定的策略, 并可根据系统调用的统计数据进一步细分, 方法如下: 把特权进程的调用分为进程控制、文件系统控制、系统控制、内存管理、网络管理、socket控制、用户管理及进程间通信等几大类, 统计出它们的调用频率以对 T 进一步细分.

3.2 参数 w, L, d的确定

试验结果[4,6,7]表明, w 的选取对检测性能的影响较大, 而 L 的选取对性能的影响较小. w 在区间 [6, 12] 中, 其中在 w = 10时是个转折点, 故可取 w = 10.

由条件熵准则[8], 马尔可夫状态序列 L 越长, 条件熵越小, 但是在给定的阈值的情况下, 条件熵的这种变化会越来越小, 根据性能的考虑, 选择一个合理的 L.

根据 Akaike的 FPE(Fina Prediction Error) 准则来衡量, 当 FPE最小时 d 的取值最佳, 但考虑到计算量问题, 在 w 与 d 之间通常有个约束条件[9], 即: $0 < d \quad 0$ 1w. 故这里可以取 d = 2.

3.3 性能分析

LPMCTS算法与 LPMC算法相比, 添加了时间戳, 并在每个状态的 $_i$ 计算中改变了部分计算方法, 但与 LPMC算法相比并没有增加算法的时间复杂度, 即每个状态序列的时间复杂度都是 $O(N^2L)$, 其中 N 为状态的数目, L为状态序列长度. 但 T的划分提高了模型的检测精度, 特别在特权进程总体波动比较大的情况下, 能很好地降低漏报率和误报率. 考虑到时间的连续性, 对于 $_0$的计算可以取相邻 k个时间段的均值, 以提高算法的灵活性.

4 结论

本文改进了 LPMC算法, 提出了 LPMCTS算法. 由理论分析可知, LPMCTS算法在没有增加算法复杂度的情况下, 使得马尔可夫检测模型更能适应系统调用序列波动较大的环境, 降低了漏报率和误报率, 提高了算法的准确度.

[参考文献] (References)

[1] 刘次华. 随机过程[M]. 2版. 武汉: 华中科技大学出版社, 2001: 42-43.
Liu Cihua. Stochastic Processes[M]. 2nd ed. Wuhan: HuaZhong University of Science and Technology Press, 2001: 42-43 (in Chinese)

[2] Warrender C, Forrest S, Pearlmutter B. Detecting intrusions using system calls: alternative data models[C]// Proc the 1999 IEEE Symposium on Security and Privacy. Berkely, California, USA: IEEE Computer Society, 1999: 133-145.

[3] Lane T. Machine Learning techniques for the computer security domain of anomaly detection[D]. West Lafayette: Purdue University, 2000.

[4] 尹清波, 张汝波, 李雪耀, 等. 基于线性预测与马尔可夫模型的入侵检测技术研究[J]. 计算机学报, 2005, 28(5): 900-907.
Yin Qingbo, Zhang Rubo, Li Xueyao, et al. Research on technology of intrusion detection based on linear prediction and markov model[J]. Chinese Journal of Computers, 2005, 28(5): 900-907. (in Chinese)

[5] Rabiner L, Juang B. Fundamentals of Speech Recognition[M]. New Jersey: Prentice-Hall International Inc, 1993.

[6] 孙宏伟, 田新广, 邹涛, 等. 基于隐马尔可夫模型的 IDS程序行为异常检测[J]. 国防科技大学学报, 2003, 25(5): 63-67.
Sun Hongwei, Tian Xinguang, Zou Tao, et al. Anomaly detection of the program behaviors for IDS based on hidden Markov models[J]. Journal of National University of Defense Technology, 2003, 25(5): 63-67. (in Chinese)

[7] Forrest S, Hofmeyr S A, Somayaji A, et al. A sense of self for UNIX processes[C]// Proceedings of the 1996 IEEE Symposium on Security and Privacy. Oakland, California, 1996: 120-128.

[8] Simon Haykin. Neural Networks: A Comprehensive Foundation[M]. 2nd ed. New Jersey: Prentice-Hall International Inc, 1999.

[9] Thottan M, Ji C. Adaptive thresholding for proactive network problem detection[C]// Proceedings of the Third IEEE International Workshop on Systems Management. Newport, Rhode Island, 1998: 108-116.

[责任编辑: 严海琳]