

# 基于动态行为和特征模式的异常检测模型

林果园<sup>1),2)</sup> 郭山清<sup>1)</sup> 黄 皓<sup>1)</sup> 曹天杰<sup>2),3)</sup>

<sup>1)</sup>(南京大学计算机科学与技术系软件新技术国家重点实验室 南京 210093)

<sup>2)</sup>(中国矿业大学计算机学院 徐州 221008)

<sup>3)</sup>(中国科学院软件研究所信息安全国家重点实验室 北京 100080)

**摘 要** 该文针对现有的异常检测方法大多只关注系统调用出现的频率或者局部变化的情况,提出了一种将动态行为和全局特征结合起来的检测模型(DBCPIDS)。文章针对满足支持度要求的系统调用短序列,给出了特征模式的概念,并以此为基础提出了基于改进的隐马尔科夫方法(IHMM)。当利用该模型进行检测时,首先用程序轨迹匹配特征模式,如果不匹配再用 IHMM 进行检测,从而使得该检测模型充分利用了程序正常运行的全局特征和程序运行期间的局部变化。通过实验表明,利用该模型进行异常检测,具有很高的检测率和较低的误报率。

**关键词** 特征模式;子序列;系统调用;异常检测

中图法分类号 TP309

## An Anomaly Detection Model Based on Dynamic Behavior and Character Patterns

LIN Guo-Yuan<sup>1),2)</sup> GUO Shan-Qing<sup>1)</sup> HUANG Hao<sup>1)</sup> CAO Tian-Jie<sup>2),3)</sup>

<sup>1)</sup>(State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

<sup>2)</sup>(School of Computer, China University of Mining & Technology, Xuzhou 221008)

<sup>3)</sup>(State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100080)

**Abstract** Differing from existed anomaly detection methods which only dealt with the frequencies of system calls or local variation, the paper puts forward a model named DBCPIDS. It took in both dynamic behavior and character patterns of programs. In this model, the authors defined the short sequence of system calls as a character pattern if this sequence satisfied the certain support degree, and propose an improved HMM(IHMM) on this basis. When detecting intrusions, firstly, we would judge whether the program trace is matched character patterns. If not, then the authors would use IHMM to detect. The model can not only reflect the global character of the program normal traces, but also pay much attention to the local warp in the execution. The experiments results show that the authors can get higher detection rate and lower false positive rate with DBCPIDS.

**Keywords** character pattern; sub sequence; system call; anomaly detection

### 1 引 言

很多入侵行为(不包括某些拒绝服务攻击),无

论是本地的还是通过 Internet 的远程攻击,都是利用了存在漏洞的关键程序。针对这些关键程序进行攻击是非法用户入侵系统的主要手段,监视这些程序的运行状况是否异常就可以检测出大部分系统入

收稿日期:2006-04-03;修改稿收到日期:2006-06-06。本课题得到国家“八六三”高技术研究发展计划项目基金(2003AA142010)、江苏省自然科学基金(BK2002073)、江苏省高技术计划项目基金(BG2004030)和中国矿业大学青年基金(OD4546)资助。林果园,男,1975年生,博士研究生,讲师,主要研究方向为计算机安全。E-mail: lgy8864@163.com。郭山清,男,1977年生,博士研究生,主要研究方向为计算机安全。黄 皓,男,1957年生,教授,博士生导师,主要研究领域为计算机网络安全。曹天杰,男,1967年生,副教授,主要研究方向为网络安全。

侵. 针对关键程序的入侵检测逐渐成为主机入侵检测的研究重点之一.

Forrest 等人<sup>[1]</sup>通过对入侵行为及入侵方法的分析研究,发现这些入侵最终都体现在一系列非法的或者说是异常的系统调用上. 因此围绕着系统调用来发现入侵行为成为了研究的热点,也提出了多种异常检测方法.

本文在前人工作的基础上,提出了一个基于程序动态行为和特征模式的异常入侵检测模型,兼顾了时序和频率,既从全局角度把握程序的行为特性,又能注意局部出现的行为异常,力图在提高入侵检测率的同时,减少漏报和误报.

本文第 2 节指出了与本文相关的工作;第 3 节详细介绍了我们提出的异常检测模型;第 4 节针对该模型进行了实验,并与其它模型进行了实验结果的比较和分析;第 5 节给出结论及以后的工作.

## 2 相关工作

Forrest 等人<sup>[1]</sup>提出的时延嵌入序列(TIDE)的方法中,考虑到了短时间序列在全局序列中出现频率的特性. 检测时,来自检测轨迹的序列与正常数据库轮廓中的序列相比较,找不到一样的序列叫做不匹配. 任何一次不匹配都说明该序列是没包括在正常训练数据库轮廓的序列,就可能是异常行为. 通过计数不匹配次数并计算其占数据库中总序列的百分比,再将这个百分比与预先给定的阈值相比较,就可以判断程序此次执行是正常还是异常. 此后,出现了多种基于频率的检测方法. Lee<sup>[2]</sup>等人用 ripper 基于频率统计的方法从系统调用序列中挖掘正常和异常的模式. KNN<sup>[3]</sup>分析的方法仅仅是观察所有关键程序中的一部分重要的系统调用,而并非全部系统调用,它没有观察系统调用的局部顺序,而是观察描述系统行为的系统调用的频率. Lee<sup>[4]</sup>提出了信息论测度的方法,计算相对条件熵和信息增益及信息损耗等用于分析入侵检测数据的性质,并对 Unix 系统的系统调用数据进行了实际检测.

在 Forrest 改进的 STIDE<sup>[5]</sup>方法中,把出现在局部区域的不匹配数目作为一种异常行为的度量. 假设局部区域的大小选为  $L$ ,则类似文献<sup>[1]</sup>仍将长度为  $k$  的窗口通过检测轨迹,一次滑动一个系统调用,将得到的序列与数据库轮廓中的序列相比较,记录前  $L$  个序列中有多少不匹配,将不匹配数  $LFC$  与阈值  $Y(1 \leq Y \leq L)$  比较,若  $LFC$  大于  $Y$ ,则认为有

异常行为发生. 该文论证了局部区域的不匹配数目有时也能够比较好地表征异常行为. 随后基于局部序列的检测方法不断涌现.

在基于局部序列的检测方法中,普渡大学的 Lane<sup>[6]</sup>使用隐马尔科夫模型(HMM)对用户命令序列进行了分析. 他通过对正常用户的命令序列建立正常行为简档,然后将当前的数据序列与正常行为简档进行比较,给出一个相似度,用以区分合法用户与入侵者. Raman<sup>[7]</sup>意识到了 STIDE 和 HMM 的优势与缺点,提出了综合两者优点的模型 Hybrid-HMM(HHMM),但是其系统调用短序列依然基于出现次数作为其特征,且每个短序列长度都限定为 2,并且依然没有改进 HMM 计算量大的缺陷,人为地限定短序列的长度不能很好地反映程序的行为特征,对检测性能带来了不利的影响. 另外,由于其检测方法中要用到系统调用出现的概率,所以这个方法不适于在线检测. 文献<sup>[8]</sup>基于局部序列的思想,提出只考虑具有“写”性质的系统调用,容易造成漏报,因为“读”动作也可以是入侵行为.

从以上的分析中可以看出,针对系统调用的异常检测技术主要包括两类:一类是基于频率的异常检测方法,另一类是基于序列的检测方法. 在基于频率的检测方法中,由于过分关注全局的环境,容易忽略局部的异常变化,容易造成漏报,同时又无法完成在线检测,不能及时发现入侵行为. 在基于局部序列的方法中,又过分关注局部的微小变化,而没有从整体去考虑程序正常行为的特点,故而容易造成误报.

根据 Forrest<sup>[5]</sup>的研究,正常系统调用迹中存在大量的有规律的、重复出现的系统调用序列,把这些重复出现的序列看成一个独立的程序特征,可以更精确地刻画程序的正常行为模式. 受到上述思想的启发,在我们的检测模型 DBCPIDS(Dynamic Behavior and Character Patterns based Intrusion Detection System)中,将从正常系统调用序列中提取程序的特征模式库,从而保证在全局角度把握程序的行为特征. 这个特征模式不同于文献<sup>[7]</sup>中提到的行为特征,它是不定长的且每个特征模式都满足支持度要求. 在检测时,首先考察要测试的系统调用序列是否符合程序的全局行为的特征模式,如果不符合,则继续考察是否在局部发生明显的异常行为,若是,则判其为入侵. 这样既可以充分利用程序的全局行为特征又能充分利用程序局部行为变化特点,从而达到提高检测率,降低误报和漏报的目的.

### 3 DBCPIDS 模型与相关算法

如图 1 所示,该模型中,首先从程序正常行为的系统调用序列中提取其特征模式,构造特征模式库。根据特征模式库,对训练集轨迹进行压缩,在此基础上训练改进的隐马尔可夫模型(Improved-HMM,以下简称 IHMM)得到程序正常运行的模型。

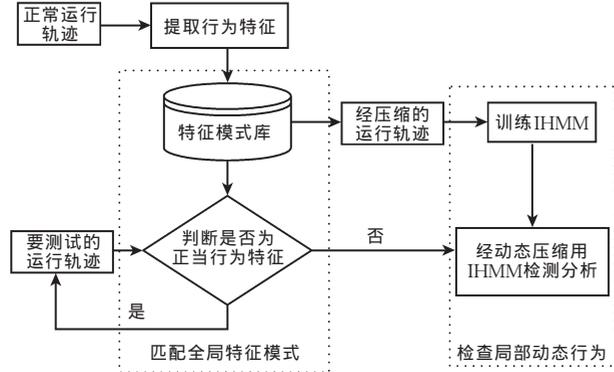


图 1 DBCPIDS 模型结构图

检测时,首先考察系统调用序列是否满足特征模式库,如果满足,则认为正常;如果不满足,则根据 IHMM 模型进行相似性计算,低于给定的阈值即为入侵。

基于系统调用短序列库<sup>[1]</sup>和有限状态机<sup>[9,10]</sup>是目前最常用的程序异常检测方法,DBCPIIDS 模型综合利用了这两种方法。以系统调用短序列库的形式存储特征模式,实现简单,不仅保留了同一特征模式内部系统调用间的时序关系,也反映了该模式在程序正常运行中的支持程度。基于特征模式库,利用 IHMM 这一概率有限状态机不仅能完整保留程序正常执行轨迹上系统调用间的长距离时序信息,也使原本复杂的处理过程有了很大的简化。

#### 3.1 相关定义

为了说明模型中的相关算法,下面给出了相关符号的定义与表示,其中表 1 给出了文中用到的符号表示。

表 1 各种符号表示

符号	含义
$\Sigma$	操作系统调用集合
$\Sigma^c$	特征模式库中出现的系统调用集合
$sc_j$	编号为 $j$ 的系统调用
$supp_{\min}$	最小支持度
$c^j$	编号为 $j$ 的特征模式
$ C $	特征模式库的模

定义 1. 系统调用短序列。它是指连续出现的

系统调用构成的序列,表示为  $(sc_1, \dots, sc_i, \dots)$ ,其中  $sc_i$  表示序列中第  $i$  个系统调用。

定义 2.  $T$  为程序运行轨迹。它表现为程序第一个系统调用到最后系统调用构成的序列  $T: (sc_1, sc_2, \dots, sc_i, \dots)$ ,在提取特征模式和训练阶段使用。

定义 3. 特征模式。它用来表征程序行为特征的、满足一定支持度要求的系统调用短序列。 $c^j$  表示编号为  $j$  的特征模式。

对特征模式采用类似系统调用编号的方式。如果特征模式中只有一个系统调用,那么它的编号就是原系统调用的编号;否则,该特征模式在行为轨迹中表现为大于 300 的编号。因为一般的操作系统中,系统调用的个数不会超过 300 个,所以这种编号方式不会引起混淆。另外,因为 IHMM 是适合处理离散数据序列的模型,所以编号大小不会影响隐马尔可夫链的性能和准确度。

定义 4.  $T'$  为  $T$  经过压缩处理后的程序轨迹,该轨迹是由系统调用与特征模式混合构成的序列,表示成  $T': (s_1, s_2, \dots, s_i, \dots)$ ,  $s_i$  表示第  $i$  个系统调用,可能是  $sc^k$  也可能是  $c^j$ 。例如  $T': (53, 67, 23, 307, 14, 312, \dots)$ ,其中,307 与 312 是两个特征模式。

定义 5.  $len()$  为长度函数。它是指一个序列中含有多少个项。

在没有提取特征模式前,一个程序行为轨迹表现为系统调用构成的序列,即  $T: (sc_1, sc_2, \dots, sc_i, \dots, sc_n)$ ,其长度为  $n$ 。在进行特征模式提取之后,轨迹序列就表现为系统调用序列与特征模式混合构成的序列,即  $T': (s_1, s_2, s_3, \dots, s_i, \dots, s_m)$ ,其长度为  $m$ 。显然,  $n \geq m$  成立。

定义 6.  $link(l_i, l_j)$  为连接函数。将  $l_i$  与  $l_j$  连接成一个新的系统调用短序列  $(l_i, l_j)$ 。当  $l_i$  与  $l_j$  在轨迹  $T$  中连续出现才可以进行连接运算。

定义 7.  $\chi_{T,l}$  表示系统调用短序列  $l$  在  $T$  中出现的次数。

定义 8. 支持度。对系统调用短序列  $l$  的支持度用  $supp(l)$  表示,  $supp(l) = \frac{len(l) \times \chi_{T,l}}{len(T)}$ 。支持度用来表征短序列作为一个整体在全局轨迹中出现的概率,满足最小支持度要求的系统调用短序列才能被看作特征模式。

定义 9. 特征模式库  $C$  是一个个特征模式构成的集合,实际上是一系列系统调用序列构成的。为了检索快捷,可以考虑采用树形结构存储。每个特征

模式的支持度都大于等于预先设定的最小支持度阈值  $supp_{\min}$ .

引理 1. 当  $l' \subseteq l$  且  $l \in C, l' \in C$  时, 那么从  $C$  中删除  $l'$ , 只保留  $l$ . 此时, 我们称  $l$  为特征模式,  $l'$  为子特征模式.

此引理体现在提取特征模式时贯彻了长度优先原则, 减小了特征库的大小. 显然, 每个子特征模式  $l'$  的支持度都不小于其母特征模式  $l$  的支持度.

定理 1.  $\forall l \in C$ , 当  $len(l) = n$  时, 则  $l$  中实际包含  $n \times (n+1)/2$  个子特征模式.

证明(采用数学归纳法).

当  $n=1$  时, 该特征模式只含有一个系统调用, 就是它自身, 显然它是一个特征模式, 且  $1 \times (1+1)/2 = 1$ , 因此  $n=1$  时命题成立.

假设  $n=k$  时命题成立, 即  $len(l) = k$  时,  $l$  中包含  $k \times (k+1)/2$  个子特征模式.

那么, 当  $n=k+1$  时, 即有  $len(l) = k+1$ . 设  $l = (sc_1, sc_2, sc_3, \dots, sc_{k-1}, sc_k, sc_{k+1})$ , 由刚才的假设知, 子序列  $(sc_1, sc_2, sc_3, \dots, sc_{k-1}, sc_k)$  含有  $k \times (k+1)/2$  个子特征模式, 当在此子序列后面增加了一个系统调用  $sc_{k+1}$  后, 长度为 1 的子序列增加了  $sc_{k+1}$ , 长度为 2 的子序列增加了  $(sc_k, sc_{k+1})$ , 长度为 3 的子序列增加了  $(sc_{k-1}, sc_k, sc_{k+1})$ , 以此类推, 长度为  $i$  的子序列增加了  $(sc_{k-i+2}, \dots, sc_{k-1}, sc_k, sc_{k+1})$ , 长度为  $k$  的子序列增加了  $(sc_{k-k+2}, \dots, sc_{k-1}, sc_k, sc_{k+1})$ , 即  $(sc_2, \dots, sc_{k-1}, sc_k, sc_{k+1})$ , 还有一个长度为  $k+1$  的子序列即  $l$  自身  $(sc_1, sc_2, sc_3, \dots, sc_{k-1}, sc_k, sc_{k+1})$ , 它也是新增加的.

从上面的推理中, 不难看出, 长度为  $1, 2, \dots, k, k+1$  的子序列各增加了一个, 因此共增加了  $k+1$  个子序列, 所以当  $l$  含有  $k+1$  个系统调用时, 其子特征模式个数为  $(k+1) + k \times (k+1)/2$ , 也就是  $(k+1) \times (k+2)/2$ . 证毕.

### 3.2 特征模式提取

在很多文献<sup>[1~3,5]</sup>中, 对系统调用序列的长度都做了明确的限制, 我们在提取特征模式时, 系统调用序列的长度不加限制, 只要满足最低支持度的要求, 都被看作是其特征模式. 首先寻找长度为 1 的特征模式, 即满足最低支持度的单个系统调用, 然后在此基础上, 再将相邻的、满足支持度要求的单个系统调用合并成长度为 2 的特征模式. 在此基础上, 再将相邻的满足支持度要求的长度为 2 和 1 的特征模式进行连接, 以求得长度为 3 的特征模式. 依此类推, 直到没有新的更长的特征模式出现. 在构建特征模式

库时, 采取了长度优先原则, 即若存在较长的特征模式, 那么就不存在单独列举其包含的较短的特征模式. 这样就明显减少了系统特征库的大小, 同时有效缩短了程序轨迹的长度, 降低了 IHMM 模型的训练时间. 算法 1 示意了特征模式库的构造方法.

算法 1. 特征模式提取.

输入: 系统调用序列  $T$ , 最小支持度  $supp_{\min}$

输出: 程序特征模式库  $C$ , 新的轨迹  $T'$

// 计算所有支持度大于  $supp_{\min}$  的单个系统调用, 将其

// 加入到特征库.

$J = len(T)$

While ( $i \leq j$ )

{

If ( $\chi_{T, sc_i} / j \geq supp_{\min}$ ) and  $sc_i \notin C$  then  $addin(C, sc_i)$ ;

$i++$ ;

}

// 根据引理 1, 将满足支持度要求相邻的系统调用进行

// 合并, 直到轨迹序列不再缩短为止.

do

$fro = 1; tail = 2; T' = T''; T'' = T;$

while ( $tail \leq len(T)$ )

{ if  $supp(link(s_{fro}, s_{tail})) \geq supp_{\min}$  then

$\{ s_{new} = link(s_{fro}, s_{tail});$

$addin(C, s_{new});$

$del(C, s_{fro});$

$del(C, s_{tail});$

$T' = T' + s_{new};$

$fro = fro + 2;$

$tail = tail + 2;$

else

$\{ fro = tail;$

$tail = tail + 1;$

$T' = T' + s_{fro};$

endif

$T = T';$

}

Until  $T' = T''$

### 3.3 IHMM 及其训练

考虑到隐马尔可夫模型能够根据观察到的序列揭示出隐含的状态序列, 所以我们在 HMM 模型基础上加以改进, 使它能够在再检测时能够适应系统调用短序列长度变化的情况. 原始  $T$  经过特征提取后变为  $T': (s_1, s_2, \dots, s_m)$ ,  $T'$  实质上是特征模式或系统调用构成, 用于 IHMM 模型的建立和训练. 对于 IHMM:  $\lambda = (Q, O, A, B, \pi)$  模型, 有

(1) 观察值集合  $O$  为  $C \cup (\Sigma - \Sigma')$ ,  $T'$  (设  $T'$  长度为  $m$ ) 中对应的观察值序列流为  $(Seq_1, Seq_2, \dots,$

$Seq_{m-k+1}$ ), 其中序列  $Seq_i = (s_i, s_{i+1}, \dots, s_{i-k+1})$ ,  $k$  为序列长度. 每一个观察值  $s_j (1 \leq j \leq m)$  或者体现为特征模式, 或者为单个的系统调用.

(2) 状态值集合为  $Q = \{1, 2, \dots, n, n+1\}$ ,  $n$  为特征库中特征模式的个数, 每个特征对应一个状态, 当实际的训练轨迹中的系统调用不是特征库中的元素时, 就落在第  $n+1$  状态. 这样就实际上构成了观察值和状态之间的映射关系, 但不是一一映射的关系, 因为特征库中的特征模式是遵从长度优先原则的, 根据引理 1 与定理 1, 可知其中的每个子序列均为子特征模式, 长度为  $j$  的特征模式实际包含  $j \times (j+1)/2$  个子特征模式, 所有这些子特征模式都映射到了一个状态, 第  $n+1$  状态实际上映射的是所有没有落入特征库的而又出现在轨迹中的系统调用. 状态序列由长度为  $k$  的滑动窗口构成, 即  $Q_1 = (q_1, q_2, q_3, \dots, q_k)$ ,  $Q_2 = (q_2, q_3, q_4, \dots, q_{k+1})$ ,  $\dots$ ,  $Q_i = (q_i, q_{i+1}, q_{i+2}, \dots, q_{i+k-1})$ .

(3)  $\pi = \{\pi_i\}$ : 初始状态分布设为提取特征模式时计算出来的支持度, 对于没有进入到特征模式库的系统调用落在的状态  $n+1$ , 其初始值为  $1 - \sum_{j=1}^n \pi_j$ .

(4)  $A = \{a_{ij}\}$  表示状态转移矩阵即  $a_{ij} = P(q_i = j / q_{i-1} = i)$ . 在计算状态转移概率时, 要首先识别出观察值序列流所在的状态, 进而统计状态之间的转移. 如果某状态  $i$  所对应的特征模式在当前观察值序列中的所在权重最大, 则认为当前观察值序列对应的滑动窗口处于状态  $i$ .

$$q_i = \arg \max_{i \leq j \leq i+k-1} \{supp(s_j) \times len(s_j)\} \quad (1)$$

如前面所述, 因为我们知道了观察值跟状态之间有一定的映射关系, 所以适合采用最大似然估计的方法:

$$a_{ij} = \frac{\sum_{t=1}^{m-k+1} \delta(seq_t, q_i) \times \delta(seq_{t+1}, q_j)}{\sum_{t=1}^{m-k+1} \delta(seq_t, q_i)} \quad (2)$$

其中,  $\delta(x, y) = \begin{cases} 1, & \text{如果 } x \rightarrow y \\ 0, & \text{如果 } x \nrightarrow y \end{cases}$ .

(5)  $B = \{b_i(j)\}$  表示输出的概率矩阵. 根据最大似然估计方法, 其计算公式如下

$$b_{ij} = \frac{\sum_{t=1}^{m-k+1} \delta(seq_t, q_i) \times \delta(s_t, q_j)}{\sum_{t=1}^{m-k+1} \delta(seq_t, q_i)} \quad (3)$$

对 IHMM 的训练算法如下.

## 算法 2. 训练算法.

$T' = (s_1, s_2, s_3, \dots)$ ;

初始化  $A, B, \pi$ ;

$i = 1; j = len(T')$

// 计算状态序列

while  $i < j$  do

{

if  $i \leq j - k + 1$  then

{ 针对  $(s_i, s_{i+1}, \dots, s_{i+k-1})$ , 根据式(1)确定其状态  $q_i$ ;  
 $i++$ ;

}

// 计算相关矩阵

for  $1 \leq i, j \leq n+1$  do

{

根据式(2)计算状态转移矩阵;

根据式(3)计算观测值转移概率矩阵;

}

## 3.4 在线检测算法

在线检测时, 程序运行实际表现为原始系统调用序列, 因此在检测时要改造成用特征模式表示的轨迹. 再比对当前的系统调用是否为特征模式, 如果是, 继续分析后面的系统调用. 如果遇到不是特征模式的序列, 则用训练好的 IHMM 模型计算该系统调用出现的概率. 计算时, 采用前向算法, 即

$$p(s_j | \lambda) = p(q_j | \lambda) \prod_{v=j}^{j+k-2} p(q_{v+1} | q_v, \lambda) p(o_{v+1} | q_{v+1}, \lambda) \quad (4)$$

如果低于给定的阈值, 则判断为入侵; 否则窗口向前滑动一次, 继续检测. 详细的在线检测算法见算法 3.

## 算法 3. 在线检测算法.

初始化异常标志  $anomaly\_flag = false$

$T = (sc_1, sc_2, sc_3, \dots)$

设异常检查窗口大小为  $k$ ;

$P_{threshold}$  为 IHMM 异常检测阈值;

$i = 1; j = 1; s_j = sc_i$

do

{  $i++$

if  $s_j + sc_i \in C$  then

{// 试图合并成最长的特征

$s_j = s_j + sc_i$ ;

else // 若不能合并, 则开始一个新的  $s$

{  $j++$ ;

$s_j = sc_i$ ;

// 不是特征模式, 要用 IHMM 检测

if  $not(s_j \in C)$  then

根据 IHMM,  $\lambda$ , 利用式(4)计算  $P(s_j / \lambda)$ ;

if  $P < P_{threshold}$  then {  $anomaly\_flag = true$ ; Alarm }

// 检测完毕后, 若不是入侵则开始分析下一个

```

//系统调用或者特征模式
endif
}
endif
}
until process stop

```

## 4 实验与评价

为了将该模型与其它算法进行比较分析,实验针对 lpr, ftpd, sendmail 和 ps 四种程序进行,所采用的系统调用轨迹主要来自 University of New Mexico<sup>①</sup> 和 MIT 的 Lincon 实验室<sup>②</sup>. 实验结果是在 P4 2.4GHz、内存为 1G 的 Windows 2000 Professional 平台上生成. 实验分别对最小支持度  $supp_{min}$ 、滑动窗口大小  $k$ 、判定阈值  $p_{threshold}$  的选择进行了分析,其中对前两个参数的实验只使用到了程序的正常轨迹,最后以这些参数为基础比较了 Hybrid-HMM(HHMM)模型<sup>[7]</sup>与本文模型的检测性能.

### 4.1 最小支持度 $supp_{min}$ 的选择

图 2 表示出了在不同最小支持度的情况下,特征模式数目的变化情况. 其中, lpr 程序的特征模式在 0.18 时达到了最大值,在  $[0.26, 0.34]$  间变化最小,之后又逐渐减少,而 ps, ftpd, senmail 三个程序分别在  $supp_{min}$  为 0.18, 0.14, 0.18 时达到了最大值,特征模式数目变化幅度较小的区间为  $[0.22, 0.34]$ ,  $[0.18, 0.30]$ ,  $[0.14, 0.30]$ . 可以看出随着  $supp_{min}$  值的增加,四个程序的特征模式数目一般都表现为先由少到多,维持一定恒定区间后又减少的趋势. 这是因为提取特征模式时,采取了长度优先的策略,当  $supp_{min}$  较小时,多个满足最小支持度要求的子特征模式被包含在一个比较长的特征模式中. 随着  $supp_{min}$  的增大,这些较长的特征模式中的一部分子序列不再满足要求,因而被分割成多个较小的特征模式. 如果  $supp_{min}$  继续增大,满足要求的特征模式将越来越少,所以后面出现了数目逐渐减少的趋势.

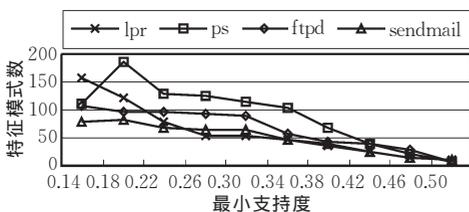


图 2 最小支持度对特征模式数目的影响

根据 Forrest<sup>[13]</sup> 的实验结论,在实际构造特征

库时,若特征库变化的速度小于某给定阈值时,就可认为已经收集了能够表征该程序的特征模式. 从图 2 可以看出,随着  $supp_{min}$  的增大,四种程序特征模式数量变化不大的区间都包含  $[0.22, 0.30]$ ,因此取其中的某个值作为后续实验的参数,都能够保持特征模式库相对稳定,可以用提取来的特征模式表征应用程序的行为特征. 为了方便,我们在后续实验中对四种程序的  $supp_{min}$  都取这个区间的中间值,即 0.26.

### 4.2 $k$ 值选择

对于滑动窗口长度  $k$  值的选择,根据文献<sup>[4]</sup>的研究 6~15 是最佳合适的选择. 但是从条件熵和计算成本的综合角度考虑, Lee 将窗口选择为 6, 并且 Forrest<sup>[1]</sup> 的实验窗口也选为 6. 为了与其它方法比较,同时考虑到计算速度的因素,我们应该相应地选择  $k$  值,但是  $k$  值可以选择比 6 小的某个数值. 因为本文提到的滑动窗口中的每一个元素不一定是单个的系统调用,很有可能是长度大于 1 的特征模式,所以滑动窗口长度的选择跟窗口内含有多少个较长的特征模式有关.

从图 3 可以看出, ps 的模式长度集中在了 2~6, 其中长度为 3~5 的居多,而 lpr, sendmail 重要集中在了 2~5 长度范围内,只有 ftpd 的特征模式长度集中在 1~4 之间. 根据加权平均的计算方法  $\frac{\sum_{i=1}^{|C|} c_i \times len(c_i)}{\sum_{i=1}^{|C|} c_i}$ , 得到四种程序的特征模式的平均长度为 3.68, 4.22, 2.69 和 3.8, 这说明大部分的特征模式的长度是大于等于 3 的, 而文献<sup>[7]</sup>中采用长度为 2 的短序列作为考察对象, 显然有些偏颇. 在 DBCPIDS 模型中当取滑动窗口大小为 2 时, 只有 ftpd 程序中实际考察的系统调用短序列平均长度小于 6, 为 5.38. 因此, 在对 IHMM 训练和在线检测时, 取  $k$  值为 3, 实际对应的系统调用短序列长度的平均值分别达到了 11.04, 12.66, 8.07 和 11.4, 符合 Lee 的相关理论与实验结果<sup>[4]</sup>.

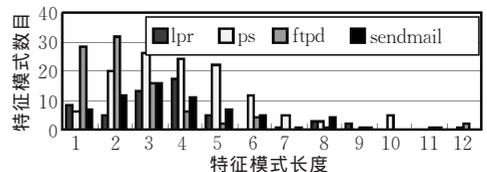


图 3  $supp_{min}$  取 0.26 时特征模式长度的分布情况

① <http://www.cs.unm.edu/~immsec/data> [OL]  
 ② <http://www.ll.mit.edu/IST/ideval/index.html> [OL]

### 4.3 检测性能分析与比较

图 4 揭示了阈值变化给入侵漏报带来的影响,可以看出在阈值小于 0.2 时,对四种程序检测的漏报率都很高,均在 50% 以上.当阈值大于 0.2 时,除了 lpr 外,其它三个程序的漏报率都急剧下降,在阈值为 0.37 时降低到了 20% 以下,而对 lpr 检测的漏报率在 20% 的情况则发生在阈值为 0.43 的时候,说明 lpr 程序的正常轨迹和入侵轨迹对较小的阈值反映不敏感.当阈值分别设为 0.48, 0.53 时,四种程序的漏报率都在 10% 和 3% 以下,而当阈值达到 0.57 时,基本上杜绝了漏报.

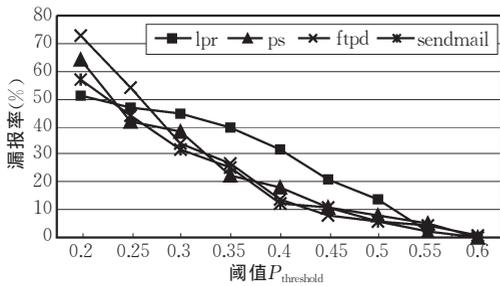


图 4 阈值  $P_{threshold}$  对漏报率的影响

为了与文献[7]使用了 HHMM 模型进行比较,将本文提到的模型对四种程序在不同阈值情况下的误报率(FPR)和检测率(DR)与 HHMM 模型的实验结果进行了比较,相关数据见表 2.可以看出,HHMM 模型在将各种程序混合起来检测的情况下

取得了较好的检测结果,特别是在检测率近 70% 的情况下,误报率低于 0.1%,而 DBCPIDS 模型在同样检测率的水平下的误报率在 0.17~0.38,显示出 HHMM 模型在检测率较低情况下的准确性优于 DBCPIDS.但是随着检测率的提高,HHMM 的检测准确性有了大幅度的下降,特别在检测率从 98%~100% 的时候,误报率从 0.17% 飙升到 0.68%. DBCPIDS 模型在设定的阈值不断提高的情况下,检测率也不断提高,误报率虽然也相应有所变大,但相比 HHMM 模型变化幅度不大.在检测率接近 100% 时,四种程序中只有对 ps 的误报率略高于 HHMM 模型的 0.68%,其余的误报率都在 0.65% 以下,这说明 DBCPIDS 模型总体上是优于 HHMM 的.

另外,从表 1 中可以看出 DBCPIDS 比 HHMM 模型无论在检测率还是误报率方面都具有比较平滑的变化趋势,其中的原因是 DBCPIDS 模型在检测时不仅考虑到了特征模式也充分注意到了动态行为的变化,对系统调用短序列的长度没有限制,而 HHMM 只是简单地将系统调用序列划分成长度为 2 的单元进行处理.正是由于 DBCPIDS 模型的这个特点,才使得在实际检测时检测率和误报率的变化对环境不敏感,对四种程序的检测都能保持良好的变化趋势就说明了这一点,而 HHMM 却出现了一些陡然的变化,因此说,DBCPIIDS 模型具有更强的鲁棒性.

表 2 IHMM 与 DBCPIDS 检测性能的比较

HHMM		DBCPIIDS							
		lpr		ps		ftpd		sendmail	
FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)	FPR(%)	DR(%)
		0	49	0.04	36	0.01	27	0.02	43
		0.02	53	0.15	58	0.18	46	0.08	56
0	46	0.02	55	0.28	62	0.27	66	0.17	68
0.09	63	0.35	60	0.32	78	0.32	73	0.25	75
0.09	67	0.38	68	0.38	82	0.37	86	0.32	88
0.13	85	0.39	79	0.38	89	0.49	92	0.51	89
0.13	94	0.43	86	0.47	92	0.53	94	0.52	94
0.17	98	0.57	98	0.49	95	0.62	96	0.59	98
0.68	100	0.6	100	0.72	100	0.65	99	0.63	100

## 5 结 论

本文利用长度不限的满足一定支持度的系统调用短序列作为程序正常环境下的特征模式,在此基础上改进了 HMM,提出了 DBCPIIDS 检测模型.该

模型将全局的程序特征和局部的动态行为进行了有机结合,适合用于在线检测,且实验证明实时性强、检测率高、误报率低,有较好的适应环境变化的能力.但检测前的准备工作除了进行训练 IHMM 外,还要发掘特征模式.因此,我们下一步的工作方向将放在改进特征模式挖掘和 IHMM 训练上.

## 参 考 文 献

- 1 Forrest S. *et al.* A sense of self for unix processes. In: John McHugh IEEE Symposium on Security and Privacy Proceedings. Oakland CA: IEEE Computer Society Press, 1996, 120~128
- 2 Lee W. , Stolfo S. J. . Data mining approaches for intrusion detection. In: Proceedings of the 7th USENIX Security Symposium. Berkeley: USENIX, 1998, 79~94
- 3 Liao Yihua, Vemuri V. R. Use of  $k$ -nearest neighbor classifier for intrusion detection. *Networks and Security*, 200, 21(5): 438~448
- 4 Lee Wenke, Xiang Dong. Information-theoretic measures for anomaly detection. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy, Oakland, California, USA, 2001, 130~143
- 5 Hofmeyr S. A. , Forrest S. , Somayaji A. . Intrusion detection using sequence of system calls. *Journal of Computer Security*, 1998, 6(3): 151~180
- 6 Lane T. , Brodley C. E. . Temporal sequence learning and data reduction for anomaly detection. In: Proceedings of the 5th ACM Conference on Computer & Communication Security, San Francisco, California, USA, 1998, 295~331
- 7 Raman C. V. , Atul Negi. A hybrid method to intrusion detection systems using HMM. In: ICDCIT 2005, Lecture Notes in Computer Science 3816, 2005, 389~396
- 8 Zhang Xiang-Feng, Sun Yu-Fang, Zhao Qing-Song. Intrusion detection based on sub set of system calls. *Acta Electronica Sinica*, 2004, 32(8): 1338~1341(in Chinese)  
(张相锋,孙玉芳,赵庆松.基于系统调用子集入侵检测. *电子学报*, 2004, 32(8): 1338~1341)
- 9 Kosoresow A. P. , Hofmeyr S. A. . Intrusion detection via system call traces. *IEEE Software*, 1997, 14(5): 35~42
- 10 Bin Y. , Qiao Y. , Xin X. W. , Ge S. . Anomaly intrusion detection method based on HMM. In: *IEEE Electronic Letters Online No: 20020467*, 2002, 38(13): 663~664
- 11 Hu J. , Hoang X. D. , Bertok P. . A multi layer model for anomaly intrusion detection using program sequences of system calls. In: Proceedings of the IEEE International Conference on Networks, Sydney, Australia, 2003, 531~536
- 12 Radha Krishna, Raju P. , Bapi S. , Laha Arijit, Kumar Pradeep, Rao M. Venkateswara. Intrusion detection system using sequence and set preserving metric. In: Proceedings of the IEEE International Conference on Intelligence and Security Informatics, ISI, Atlanta, USA, 2005, 498~504
- 13 Christina Warrender, Stephanie Forrest, Barak Pearlmutter. Detecting intrusion using system calls: Alternative data models. In: Proceedings of the 1999 IEEE Symposium on Security and Privacy, Oakland, USA, 1999, 133~145

**LIN Guo-Yuan**, born in 1975, Ph.D. candidate, lecturer. His research interests focus on computer security.



**GUO Shan-Qing**, born in 1977, Ph. D. candidate. His research interests focus on computer security.

**HUANG Hao**, born in 1957, professor, Ph. D. supervisor. His research interests focus on computer network security.

**CAO Tian-Jie**, born in 1967, associate professor. His research interests focus on network security.

## Background

This research is supported by the National High Technology Research and Development Program(863 Program) of China under grant No.2003AA142010 named "Distributed Network Monitor and Pre-Alert System", the High-Tech Research and Development Plan of Jiangsu Province under grant No. BG2004030 named "Research on Distributed Active defense, Monitor and Pre-Alert System of Computer Network", and the Young Technology Foundation of China University of Mining and Technology(No. OD4546).

The objective of the group is to study how to get higher detection rate and lower false positive rate of host intrusion detection system, which is an important part of "Distributed Network Monitor and Pre-Alert System". In the past year, the groups have done a lot of work including attack descrip-

tion, program profile construction based on system calls, and also have developed a prototype to test all the techniques proposed by the authors.

In the past, some authors, like Forrest S. , Lee W. , *et al.* , have proposed some anomaly detection methods on system calls, they either only paid attention to the sequence of system calls or only discussed system calls frequency. This paper puts forward a novel method, which took in both dynamic behavior and character patterns of programs. The model can not only reflect the global character of the program normal traces, but also pay much attention to the local warp in the execution. Experiments results show this model exceed other existed anomaly detection methods concerning system calls.