基于进程行为的异常检测模型

苏璞睿,冯登国
(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: 利用系统漏洞实施攻击是目前计算机安全面临的主要威胁.本文提出了一种基于进程行为的异常检测模型.该模型引入了基于向量空间的相似度计算算法和反向进程频率等概念,区分了不同系统调用对定义正常行为的不同作用,提高了正常行为定义的准确性;该模型的检测算法针对入侵造成异常的局部性特点,采用了局部分析算法,降低了误报率.

关键词: 入侵检测; 异常检测; 非层次聚类
中图分类号: TN918 **文献标识码**: A **文章编号**: 0372-2112 (2006) 10-1809-03

An Anomaly Intrusion Detection Model Based on Nonhierarchical Clustering

SU Pu-rui, FENG Deng-guo
(State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China)

Abstract: More and more intruders exploit the vulnerabilities of system and applications to intrude the system. This paper introduced an anomaly intrusion detection model analyzing processes' behaviors. It introduces the similarity calculation method based on Vector-space. And it introduces an argument to value the capabilities of system calls to differentiate the process behaviors. Thinking of the characters of the abnormalities caused by intrusions, the detection algorithm adopts the method of locally analyzing.

Key words: intrusion detection; anomaly detection; nonhierarchical clustering

1 引言

利用系统漏洞实施攻击是目前计算机安全面临的主要威胁.监控特权进程的系统调用数据是防范该类入侵的有效方式之一.系统调用是进程访问系统资源的接口,进程执行的系统调用序列粗略的描绘出进程的行为.基于系统调用的检测方法最早由 S. Forrest 等人提出,目前已取得了一些进展,包括 S. Forrest 等人最初提出的 N-gram 模型[2],在 N-gram 基础上发展的 Var-gram[13]模型, Vt-path 模型[17]等等. N-gram 模型直接将训练数据截为定长片段定义正常行为,正常行为定义规模大,影响检测效率. Var-gram 模型主要是针对 N-gram 模型中序列长度不易确定的问题而提出的,该模型采用不同长度的序列来定义正常行为,但 Var-gram 模型不仅没有有效解决长度问题,还引入了复杂的匹配算法,严重影响了检测效率.

本文提出了一种无监督训练算法,该算法直接利用系统实时运行数据作为训练数据,降低了对训练数据的要求.本文引入了反向进程频率和系统调用出现频率等参数,通过权值区分不同系统调用在定义正常行为中的不同作用.在正常行为定义生成和实时检测过程中,序列匹配采用了基于向量空间模型的相似度计算算法,更准确地衡量序列的匹配程度.在检测过程中,针对正常行为定义的不完整性等原因造成的干扰,引入滤噪函数,提高了检测准确性.

本文第 2 节详细介绍模型基本思想和其中的关键算法;第 3 节介绍实验结果,对模型的检测效率和准确性进行评估;最后总结全文并讨论下一步工作.

2 检测模型

操作系统采用编号来区分不同的系统调用,编号范围一般在 0～255 之间.本文直接采用系统调用号 i 表示一个系统调用类型.一个系统调用序列(System Call Sequence)是多个系统调用有序的排列,即系统调用序列 $S = \langle s_0, s_1, \cdots\cdots, s_{l-1} \rangle$,其中,$0 \le s_i < n$, $0 \le i < l$, n 是系统调用总数, $0 \le n < 256$.

定义 1 对于训练集合 P, P 中包含了各种不同应用程序在不同环境下运行进程的系统调用序列,系统调用 s 在集合 P 中的反向进程频率定义如下:

$ipf_i = \frac{N}{N_i}$,其中 N 是 P 中进程总数, N_i 是出现系统调用 i 的进程数.

对于系统调用集合中每一个系统调用,计算其反向进程频率得到: $IPF = (ipf_0, ipf_1, \cdots\cdots, ipf_{n-1})$. 其中 ipf_i 是系统调用 i 的反响进程频率.

对于应用程序 APP 的训练数据,计算各个系统调用在

收稿日期:2005-12-20;修回日期:2006-04-10
基金项目:国家 973 重点基础研究发展规划(No. G1999035802);国家杰出青年基金(No. 60025205);国家自然科学基金(No. 60273027)

训练数据中出现的频率得到:

$$SF_{APP} = (sf_0, sf_1, \cdots\cdots, sf_{n-1})$$

其中,sf_i 是系统调用 i 在应用程序 APP 训练数据集和出现的频率.

根据系统调用频率和反向进程频率得到各个系统调用在定义正常行为中的权值:

$$W_{APP} = (w_0, w_1, \cdots\cdots, w_{n-1}), \text{其中 } w_i = ipf_i \cdot sf_i$$

对于一个确定的系统调用序列,它在定长为 l 的系统调用序列中可能在 l 个不同位置出现,对于 n 种不同的系统调用.将长度为 l 的系统调用序列用 $n \cdot l$ 的向量空间表示,定义 $v_{i,j} = \langle s_{0,0}, s_{0,1}, \cdots\cdots, s_{0,l-1}, s_{1,0}, s_{1,1}, \cdots\cdots, s_{1,l-1}, \cdots\cdots s_{i,j}, \cdots\cdots, s_{n-1,l-1}\rangle$

其中 $s_{i,j} = 1$,其他均为 0.

对于应用程序 APP,如果其所有系统调用的权值为 $W = \langle w_0, w_1, \cdots\cdots, w_{n-1}\rangle$,则系统调用 $S = \langle s_0, s_1, \cdots\cdots, s_{l-1}\rangle$ 可表示为向量

$$S = w_{s_0} gv_{s_0,1} + w_{s_1} gv_{s_1,1} + \cdots\cdots + w_{s_{l-1}} gv_{s_{l-1},l-1}$$

对于应用程序 APP 的系统调用序列集 S 中,各个系统调用序列对应的向量的集和称之为 S 的向量集,记为 S.如果 S 为 l-系统调用序列集,则 S 称之为 l-系统调用向量集.

向量 X 和向量 Y 之间的相似性定义为:

$$Sim(X, Y) = \cos(X, Y) = \frac{X \cdot Y}{|X| \cdot |Y|}$$

向量 X 和向量 Y 之间的距离定义为:

$$Distance(X, Y) = 1 - Sim(X, Y) = 1 - \cos(X, Y)$$

定义 2 如果两个向量 X 和 Y 之间的距离 $Distance(X, Y) \leq \varepsilon$,则称 X 为 Y 的 ε 邻居(ε Neighborhood)(或 Y 为 X 为的 ε 邻居)[17].

定义 3 在 l-系统调用向量集 S 中,X 的 ε 邻居集(ε Neighborhood Set)为:

$$Neighbor(X, \varepsilon) = \{Y \mid Distance(X, Y) \leq \varepsilon, Y \in S\}$$

定义 4 在 l-系统调用向量集 S 中,X($X \in S$)在半径为 ε 时的密度(Density)记为 $\rho_{X,\varepsilon} = |Neighbor(X, \varepsilon)|$[17].

训练生成正常行为定义首先需要分析各个系统调用的反向进程频率,该方法已在前面介绍.另外对搜集的训练数据统计各系统调用的使用频率,最后得到各个系统调用的权值 W_{APP}.之后对训练数据进行预处理.利用定长 l 的滑动窗口,以步长为 1 的方式将各个进程的系统调用序列截为定长 l 的序列,得到训练数据集合 l-系统调用序列集 S,并转换得到 l-向量集 S,按如下算法训练生成正常行为定义:

正常行为定义生成算法

输入:S 训练数据集合,S 应用程序 APP 的 l-系统调用向量集,ε 密度计算半径,ε 类间距离,α 异常点控制比例

输出:行为定义 $Profile_{APP}$

Begin:

(1) 初始化,Profile: = NULL,j: = 0,Covered: = 0,Sum: = |S|;

(2) 对训练数据集合 S 中,计算每一个向量 X_i 半径为 ε 的密度 $\rho_{X_i,\varepsilon}$: = $|Neighbor(X_i, \varepsilon)|$;

(3) 取 S 中密度最大的向量 X_i,

if Profile 为空 then

S: = S - Neighbor(X_i, ε)

Covered: = Covered + |Neighbor(X_i, ε)

Profile: = Profile U{X_i}

Else

If Profile 中存在 Y,Distance(Y, X_i) $\leq \varepsilon'$,then

S: S - {X_i}

Covered: = Covered + 1;

Else

S: = S - Neighbor(X_i, ε)

Covered: = Covered + |Neighbor(X_i, ε)|

Endif

Endif

Endif

(4) If Covered $\geq$ Sum * (1 - α) then 返回集合 Profile,即正常行为定义

Else 执行 3

Endif

End

实时监控过程中,将为每一个被监控的进程建立一个队列,并利用一个长度为 l(与正常行为定义中序列长度一致),步长为 1 的滑动窗口将实时数据截为定长片断 S_j. $S_j = \langle s_0, s_1, \cdots\cdots, s_{l-1}\rangle$.

定义 5 向量

$$X = w_{x_0} gv_{x_0,0} + w_{x_1} gv_{x_1,1} + \cdots\cdots + w_{x_{l-1}} gv_{x_{l-1},l-1}$$

和正常行为定义 Profile 中凝聚点的最短距离称之为向量 X 与 Profile 的距离,记为

$$Distance_{X,Profile} = \min\{Distance(X, Y) \mid Y \in Profile\}$$

在检测过程中,我们借鉴了 Terran D. Lane 的滤噪思想,引入了滤噪函数 $F(S_i)$,保留入侵行为造成的明显异常的同时,去除了一些噪声的影响,降低误报率.滤噪函数 $F(S_i)$ 如下所示:

$$F(S_i) = \frac{1}{R} \sum_{j=i-R+1}^{i} Distance_{S_i,Profile}$$

其中,R 是滤噪序列长度.检测过程中,设置检测阈值 ξ,当 $F(S_i) > \xi$ 时,认为进程受到入侵.

本模型中的正常行为定义生成算法可直接改造为正常行为定义更新算法,主要作以下修改:

(1) 增加了一个输入参数——现有正常行为定义(OldProfile);

(2) 初始化时,将 Profile 初始化为 OldProfile,即 Profile: = OldProfile;

3 实验结果

本实验通过两方面的数据完成:(1) 来自于美国新墨西哥大学的数据包[4],包含了 ftp、lpr、xlock 等多种应用的实验数

据,数据量丰富;(2)搜集 wu-ftpd、traceroute 两个应用程序的相关数据,在 Redhat Linux 平台上构造了 LBNL Traceroute 堆溢出攻击[5]和 Wu-ftpd/ bin SITE EXEC 攻击[6].所有实验数据分析处理在 RedHat 9.0,P4 1.7G系统上完成.

实验中 xlock 包含了 25 个正常进程和 2 个异常进程的数据;Traceroute 包含了 10 个正常进程和 1 个异常进程数据.表 1 中的最佳值平衡了行为定义规模和行为定义的条件熵两个主要指标.当采用上述设置时,xlock 行为定义的大小为 194(是 Hofmeyr 方法[7]的 37.2 %),Traceroute 行为定义大小为 107(是 Hofmeyr 方法[8]的 32.9 %).行为定义的大小直接影响到实时检测时的效率,通过降低行为定义规模,本模型可有效提高检测效率.

我们对比分析了一个正常进程和一个异常进程系统调用序列和正常行为定义的距离之间的差异。从实验结果我们发现,正常进程中,$Distance_{S_i,Profile}$主要集中在 0.3 附近,而入侵进程的 $Distance_{S_i,Profile}$分布则比较分散,从 0 到 0.9 均大量存在.异常进程中也存在大量序列的 $Distance_{S_i,Profile}$值较小,这主要是因为在入侵发生前,异常进程执行了大量的正常操作.如果采取类似于文献[8]中的全局统计分析方法,可能会因为大量的正常行为掩盖入侵造成的异常,因此,本模型采用的局部分析方法则更有利于区分异常.

另外,从进程中系统调用的距离变化情况来看,正常进程系统调用的距离基本保持在一个相对低的水平,而入侵进程中,存在大量连续的系统调用序列,其距离保持在一个高水平,但也存在大量的系统调用序列的距离保持在较低水平(比如在进程的开始部分).从实验结果看出:

(1)正常进程和入侵进程系统调用序列在与正常行为定义的距离上具有明显差异,$Distance_{S_i,Profile}$具有较强的区分正常行为和异常行为的能力;

(2)滤噪函数可以去除正常进程中出现的偶然性异常,但保留了异常进程中的连续异常特征,降低了误报率.

为了更全面的评估检测算法的准确性,我们以新墨西哥大学提供的数据作为测试数据完成了准确性评估.评价入侵检测系统的准确性主要包括两个参数,误报率和漏报率[8].

实验数据主要包括新墨西哥大学提供的 Inetd、log、named、ftp、lpr、sendmail、xlock 等应用程序数据和采集的 wu-ftpd 和 traceroute 数据.实验中各个应用程序行为定义参数选择参考 3.2 进行,R 统一设置为 $R = 2l$.通过对所有应用程序大批量数据的实验,检测漏报率低于 5 %(详细数据请参见附录).对于误报率,选择其中的 xlock、traceroute 等应用程序相关数据,误报率低于 0.3 %.这一实验结果要优于 N-gram 等模型[12].

4　总结

利用系统漏洞实施攻击是目前计算机安全面临的主要威胁.通过分析进程行为差异是检测该类入侵的主要方法之一.本文提出了基于非层次聚类的异常检测模型.该模型具有以下特点:

(1)在模型中引入了反向进程频率和系统调用使用频率等参数,更准确地区分序列的相似程度,提高了检测准确性;

(2)利用基于非层次聚类的无监督算法训练生成正常行为定义,提高了正常行为定义质量,降低了正常行为定义规模;

(3)提出了正常行为定义更新算法,系统能在运行过程中不断完善正常行为定义,有利于确保正常行为定义与动态变化的运行环境保持一致.

参考文献:

[1] CERT Coordination Center, CERT/CC Overview Incident and Vulnerability Trends [OL], http://www.cert.org/present/cert-overview-trends/

[2] S Forrest, S A Hofmeyr, A Somayaji and T A Longstaff. A sense of self for unix processes [A]. In Proceedings of 1996 IEEE Symposium on Computer Security and Privacy[C]. 1996.

[3] 马振华. 现代应用数学手册 ——概率统计与随机过程卷[M]. 北京:清华大学出版社

[4] Wenke Lee, Dong Xiang. Information-Theoretic Measures for Anomaly Detection[A]. In Proceedings of the 2001 IEEE Symposium on Security and Privacy[C]. 2001.

[5] Steven A Hofmeyr. Stephanie Forrest, Anil Somayaji. Intrusion detection using sequences of system calls[J]. Journal of Computer Security, 1998, 6:

[6] Chowalit Tinnagonsutibout, Pirawat Watanapongse. A novel approach to process-based intrusion detection system using read-sequence finite state automata with inbound byte proler[A]. ICEP2003[C]. 2003.

[7] N Nuansri. A process state-transition analysis and its application to intrusion detection[A]. 15th Annual Computer Security Applications Conference[C]. 1999.

[8] A Wepsi, M Dacier, H Debar. Intrusion detection using variable-length audit trail patterns[A]. 3rd International Workshop on the Recent Advances in Intrusion Detection, LNCS 1907 [C]. 2000. 110 - 129.

[9] R Sekar, M Bendre, D Dhurjati, P Bollineni. A fast automaton-based method for detecting anomalous program behaviors[A]. IEEE Symposium on Security and Privacy[C]. 2001.

[10] H Feng, O Kolesnikov, P Fogla, W Lee, W Gong. Anomaly detection using call stack information[A]. IEEE Security and Privacy[C], 2003.

作者简介:

苏璞睿　男,湖北宜昌人,毕业于中国科学院软件研究所,获博士学位,主要从事网络安全和入侵检测相关研究,发表论文 10 余篇,参与并组织了多个重大项目技术攻关.

冯登国　男,现为中国科学院软件所研究员、博士生导师,信息安全国家重点实验室主任,国家计算机网络入侵防范中心主任,国家信息化专家咨询委员会委员,目前主要从事信息与网络安全方面的研究与开发工作.