基于 Windows Native API 序列的系统行为入侵检测

朱莺嘤[1],叶 茂[1],刘乃琦[1],李 筝[2],郑凯元[1]
ZHU Ying-ying[1],YE Mao[1],LIU Nai-qi[1],LI Zheng[2],ZHENG Kai-yuan[1]

1.电子科技大学 计算机学院,成都 610054
2.电子科技大学 通信学院,成都 610054
1.College of Computer,University of Electronic Science and Technology of China,Chengdu 610054,China
2.College of Communication and Inf. Eng.,University of Electronic Science and Technology of China,Chengdu 610054,China
E-mail:zy.2@163.com

ZHU Ying-ying,YE Mao,LIU Nai-qi,et al.Host intrusion detection based on sequence of Windows Native API.Computer Engineering and Applications,2008,44(18):109-112.

Abstract: Considering the shortcomings of Windows system intrusion detection and the advantages of the Linux system intrusion detection based on the sequence of the system call,a kernel-level host intrusion detection program based on the BP neural network algorithm to study and classify the sequence of Windows Native API is proposed in this paper.Experiment results prove that the sequence of Native API can be used for intrusion detection.Windows Native API means the kernel model API,which is similar to the Linux system call.The neural network is trained to learn the normal and abnormal sequence of Native API.In the intrusion detection,use the trained neural network to classify the emerging Native API sequence,and find whether the intrusion happens.
Key words: intrusion detection;Windows Native API;BP neural network

摘 要:针对 Windows 系统入侵检测的不足,研究并借鉴 Linux 下基于系统调用序列进行入侵检测的方法,提出一种采用 BP 神经网络算法对 Windows Native API 序列学习和分类的内核级主机入侵检测方案。通过实验,验证了采用 Windows Native API 序列进行系统入侵的可行性。Native API 是 Windows 系统内核模式下的 API,可以类比于 Linux 下的系统调用。通过训练神经网络学习 Native API 序列,建立一个对正常和异常 Native API 序列进行分类的 BP 神经网络。在入侵检测时,利用训练后的神经网络对不断出现的 Windows Native API 序列进行分类,判断系统是否出现异常入侵。
关键词:入侵检测;Windows Native API;BP 神经网络
DOI:10.3778/j.issn.1002-8331.2008.18.034 **文章编号:**1002-8331(2008)18-0109-04 **文献标识码:**A **中图分类号:**TP391

目前,大量计算机个人用户都在使用 Windows 操作系统,而 Windows 操作系统却受到越来越多的严重攻击,例如蠕虫、特洛伊木马以及一些变形加密病毒。同时 Windows 平台上的入侵检测现在主要采取检测网络数据包的方式,只基于特征库的匹配,并没有依据入侵行为的本质特征,深入到 Windows 系统的内核,因而不能从根本上判断系统是否发生异常入侵。

而基于 Linux 系统的许多这方面研究都已深入到系统的内核。通过分析特权进程的系统调用序列,1996 年 Forrest 等提出了用系统调用短序列描述程序行为,并建立正常行为自体来进行入侵检测的方法以后[4],许多人又在此基础上,做了很多研究工作,包括基于有限自动机[8,11],基于隐马尔可夫链模型[2,5],采用数据挖掘方法[1,7]以及基于神经网络[6,10]等多种方法在内。这些研究都在系统入侵检测上取得了很好的效果,从而在 Linux 上验证了采取基于系统调用序列的方式对操作系统进行入侵检测的可行性。

事实上,在 Windows 系统中,也有一种 Native API 类似 Linux 下的系统调用函数[9]。这类 API 是在 Windows NT 系统中一种内核级的 API 函数,它提供了应用程序和操作系统内核之间的接口,功能类似于 Linux 下的系统调用。当一个 Win32 应用程序调用一个 Win32 API 函数时,就会通过客户端的 Win32 动态链接库调用内核级的 API 函数 Native API。这与 Linux 下应用程序执行时调用系统调用函数的过程类似,并且 Native API 的拦截通过一定方法是可以实现的[3]。

同时,BP 神经网络在许多实际应用中有很好的效果[13],其算法寻优精确,分类准确,并且神经网络是分布式的存储信息和大规模的并行处理,其运算速度很快[14]。因此,BP 神经网络很好的满足实际入侵检测准确和高效的要求,能够用以对截获的 Native API 序列进行分类处理。

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60702071);国家教育部新世纪人才支持计划(the New Century Excellent Talent Foundation from MOE of China under Grant No.NCET-06-0811);四川省科技厅应用基础研究基金(No.2006J13-065)。
作者简介:朱莺嘤(1985-),女,硕士生,从事信息安全的研究;叶茂,男,副教授,从事智能信息处理、信息安全的研究。
收稿日期:2007-09-19 **修回日期:**2007-11-30

本文针对 Windows 平台的异常入侵检测存在的不足,提出一种新的使用 BP 神经网络对 Windows Native API 序列进行学习和分类的 Windows 内核级异常检测的方案。通过对关键进程的 Native API 调用进行跟踪,采用类似 Linux 的系统中基于调用序列的滑动窗口处理方式得到 Native API 短序列并归一化处理,再由 BP 神经网络对预处理过的短序列进行分类,以判断系统是否发生入侵。

本文首先对 Native API 序列的截获及其预处理的进行介绍;之后将着重进行神经网络检测模型的分析和建立;然后通过建立起来的神经网络对系统处于不同状态时截获的 Native API 短序列进行具体实验以及对实验结果进行分析比较;最后是对已有工作的总结和未来工作的展望。

1 Windows Native API 截获和序列模式的提取

1.1 Windows Native API

Windows 中存在两种模式:用户模式和内核模式。

用户应用程序在用户模式下运行,而系统程序在内核模式下运行。两种模式的重要区别是在处理文件、调用内存和使用 CPU 的优先级上不同,内核模式比用户模式拥有更高的优先级。即使用户应用程序出现了严重的错误,也不会对整个系统造成太大影响,保证了操作系统的正常运行。

API 是 Windows 操作系统在动态连接库中给用户提供系统服务的接口函数,运行在用户模式下或内核模式下。其中在内核模式下运行的 API 就是 Native API,是动态连接库中的内核级系统服务的接口函数,其与用户模式下的 API 有很大的区别[9]。Native API 调用序列能够在内核级层次上反映应用程序的特征,因此能够用来作为异常检测的数据源。

在 Win32 系统中,以下 4 个动态连接库都提供 API:User32.dll(用户接口 API)、Gdi32.dll(图形接口 API)、Kernel32.dll(系统管理接口 API)、Adapi32.dll(高级系统管理接口 API)。其中 Kernel32.dll 提供内核模式的 API,即 Native API。在 Windows XP 系统中,大约有 949 个 Native API,这里认为其类似于 Linux 的系统调用函数,可以基于其序列模式进行异常检测。

1.2 截获 Native API

Windows XP 系统中共有 949 个 Native API,其中在 NTdll.dll 中的 284 个 Native API 是系统最为关键的 Native API。Win32 API 中的所有调用最终都转向了 NTdll.dll,内核模式的驱动大部分时间调用这个模块。如果它们请求系统服务,NTdll.dll 的主要作用就是让内核函数的特定子集可以被用户模式下运行的程序调用。因此在实验中,主要跟踪截获这 284 个关键的 Native API。实验中截获的数据片段如表 1 所示。

表 1 截获的数据片段

时间戳	进程 ID	Native API	Native API ID
25	796	NtAllocateVirtualMemory(分配虚拟内存)	7
26	796	NtQueryVirtualMemory(获取虚拟内存的信息)	78
27	796	NtFreeVirtualMemory(释放虚拟内存)	3
28	796	NtSetEvent(事件对象设置)	19
29	796	NtCreateEvent(事件对象创建)	5
30	796	NtCancelTimer(撤消当前时间设置)	3
31	796	NtSetTimer(设置时间)	44
32	796	NtDelayExecution(延迟运行)	9
33	796	NtClearEvent(事件对象清除)	4
34	796	NtOpenThreadToken(打开线程标志)	29
35	796	NtAllocateVirtualMemory(分配虚拟内存)	17

拦截 Native API 是通过设计内核虚拟设备截获系统服务分配表来实现的,从而实时地获取 Native API 信息。在实验中,分别搜集正常情况下和一些系统漏洞攻击下的关键进程的 Native API 序列。

1.3 序列的提取和数据的归一化

Native API 序列的提取采用的是序列时延嵌入法,即 STIDE,选择长度为 K 的窗口通过每个程序行为轨迹,一次滑动一个 Native API,得到一个新的 Native API 短序列。在实验中,本文分别采用了长度为 15、10、6 的短序列输入神经网络训练,最后得到长度为 6 的序列分类效果最佳,因此采用序列的长度为 6。滑动窗口得到的短序列如表 2 所示。

表 2 Native API 短序列

35	17	17	137	155	137	53	155	200	206	18	24
…											
270	259	187	24	129	224	224	183	224	183	224	183
…											

为了更加准确地分类 Native API 序列,本文将短序列进行归一化,每个 Native API 用一个数字 X 唯一标志,$X\in[0,283]$,Y 表示将 X 归一化以后的对应数值。归一化采用方法:$Y=(X-142)/283$。

2 BP 神经网络入侵检测模型

2.1 算法选择

神经网络算法的选择由该入侵检测所需的检测功能设计决定。本文提出的入侵检测,实际上是通过对收集到的训练数据和测试数据的学习和分类来判定进程行为是否异常,从而判断是否系统出现异常入侵。同时,入侵检测的实时性要求也很高。而 BP 神经网络运行机制是并行的,运算速度很快,分类准确。因此,考虑到 BP 神经网络处理精确并且运算速度较快,本文采用 BP 神经网络来解决本方案中学习和分类问题。

2.2 BP 网络学习算法

BP 网络由正向传播和反向传播组成。其中正向传播是指输入信号从输入层经隐含层传向输出层,若输出层得到了期望的输出,则学习算法结束,否则转向反向传播。反向传播是指将误差信号(样本的期望输出与网络的实际输出之差),按原连接通路进行反向计算,由梯度下降算法调整各层神经元的权值,使误差最小。

BP 网络模型进行分类是通过一种在学习的过程中采用反向传播误差的前向网络实现的。该模型把一组样本的 I/O 问题变为一个非线性优化问题,使用了优化中最普通的梯度下降法。训练网络的最终目的是确定网络中各个结点之间的连接权值。BP 算法通过不断修改这些权值,来调整整个网络,使其能够达到最好的分类效果。

从数学角度看,它把一组样本的输入输出问题转化为非线形优化问题,再经过迭代运算,求解权值,使误差信号达到要求的范围。通过隐含层的调整使优化问题的可调节参数增加,求得更精确的解。所以 BP 网络能够处理对一些线形不可分问题的分类[12]。

本文用来检测的神经网络采用三层结构,三层的神经网络已经被证明足够处理很多复杂分类问题。

该模型的输入层分别采用了 6、15、20 个神经元;隐含层分别采用了 15、20、30、40 个神经元,输出层为一个神经元。经过

大量实验表明：输入层为 6，隐含层为 20 个神经元的效果最佳。因此,本文最后采取输入层神经元为 6,隐含层神经元为 20,输出层神经元为 1 的神经网络进行检测。其具体的神经网络结构图如图 1 所示。

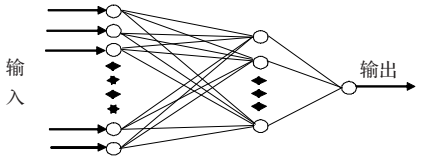

图 1　神经网络结构图

神经网络建立以后，采用 BP 算法训练神经网络,BP 算法涉及到的一系列参数描述如表 3 所示。

表 3　参数描述

参数	描述
$p=1,2,3\cdots$	表示第 p 组样本
$u_p=(u_{1p},u_{2p},\cdots,u_{np})$	输入向量(n 为维数)
$d_p=(d_{1p},d_{2p},\cdots,d_{np})$	输出向量(n 为维数)
I_{jp}	第 p 组样本输入后,每层神经网络的第 i 个结点的输入
$f(*)$	可微分的 Sigmoid 型激活函数式,是输入-输出之间的传递函数
$E_p(t)$	第 p 组样本输入时,经 t 次权值调整后,网络的目标函数
$y_{kp}(t)$	第 p 组样本输入时,经 t 次权值调整后,网络的输出层的第 k 个神经元的输出
$J(t)$	经 t 次权值调整后,网络的总目标
ε	预先确定的一个较小值的正常数
$w_{ij}(0)$	初始权值
$w_{ij}(t)$	可得到神经元 j 到神经元 i 的联接权值 t 次调整后的值
η	步长,也称学习算子或学习速率

算法的实现步骤如下：

(1)设置初始权值 $w_{ij}(0)$为较小的随机非零值。

(2)输入归一化处理得到的正常情况下的短序列,每个输入神经元与短序列中的每个值一一对应,网络的期望输出设置为 1(以 1 表示序列为正常模式),计算网络的输出：

设第 P 组样本输入、期望输出分别为：

$u_p=(u_{1p},u_{2p},\cdots,u_{np})$

$d_p=(d_{1p},d_{2p},\cdots,d_{np})$

$p=1,2,3,\cdots$

第 P 组样本输入以后,神经网络每层第 i 个神经元的输入为 I_{jp},则神经元 i 的输出 y_{ip} 为:$y_{ip}=f[\sum_j w_{ij}(t)I_{jp}]$。神经网络第一层中,$I_{jp}=u_{jp}$,第二层和第三层中 I_{jp} 为上一层网络中与该层网络神经元 i 对应的输出值,其中,$f(x)=\dfrac{1}{1+e^{-\beta x}}$。

(3)计算网络的总目标函数 $J(t)$

$$E_p(t)=\frac{1}{2}\sum_k [d_{kp}-y_{kp}(t)]^2$$

网络的总目标函数为

$$J(t)=\sum_p E_p(t)$$

作为对网络学习状况的评价。

若本次迭代总误差 $J(t)>J(t-1)$,则这次迭代无效,并恢复以前的步长,减小步长重新迭代,此时 $\eta(t+1)=\eta(t-1)-\eta(t-1)/t$;反之,本次迭代有效,增大学习步长进行下一次迭代,此时 $\eta(t+1)=\eta(t)+\eta(t)/t$。

(4)判别

若 $J(t)\leqslant\varepsilon$,ε 是预先确定的很小的常数,$\varepsilon>0$。则算法结束,否则,至步骤(5)。

(5)反向传播计算

由输出层,根据 $J(t)$,按照梯度下降法反向计算,依据下式反向调整权值：

$$w_{ij}(t+1)=w_{ij}(t)-\eta(t+1)\sum_p \frac{\partial E_p(t)}{\partial w_{ij}(t)}+\alpha G$$

$G=[w(t)-w(t-1)]$

$0<\alpha<1$

其中 η,α,β 的数值对于收敛速度至关重要，其选择因样本不同而有所区别。目前,还没有关于它们选择的统一方法。本文在实验中分别取不同的值进行训练,最后取效果最好的值。

在修改这三个参数值时一般遵循以下原则：

①η 增大,学习步长增大,但权重系数可能发生波动。

②α 增大,学习速度减慢,但可阻止权重系数发生波动。

③在非饱和区(线性区域)内,η 增大,学习速度提高,但权重系数可能发生波动。

最后实验得到,$\eta=3$,$\alpha=0.6$,$\beta=1.5$ 时,分类效果最好。神经网络收敛以后,再将用于测试的数据输入网络,测试数据包括经过归一处理后的正常序列和异常序列，神经网络的输出值在 0~1 之间连续变化。输入序列的异常程度越高,输出的值就越接近于 0;反之,正常序列的输出值应该接近 1。算法流程如图 2 所示。

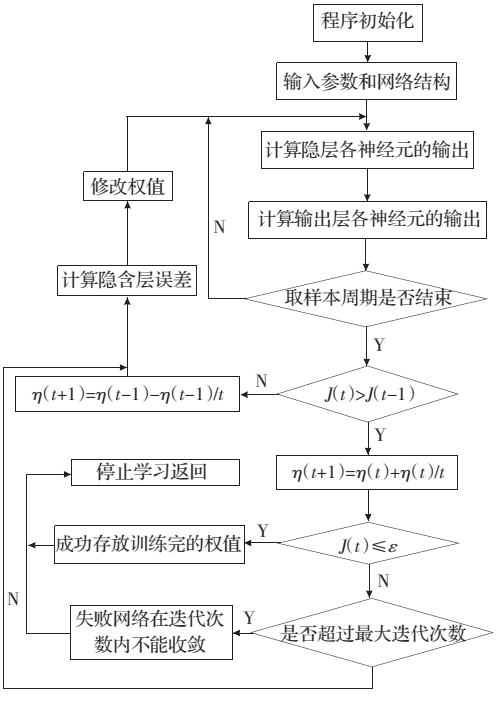

图 2　BP 算法流程图

3　实验结果

在实验中,采用的是 Windows XP SP1 操作系统,使用 BP 神经网络和 Forrest 的方法[4]分别对同一组漏洞和进程进行了检测，搜集其系统的关键进程 svchost、Lsass、csrss 以及一些常

用的应用程序的 Native API 短序列 921 600 个,其中包括正常短序列和 8 种不同漏洞攻击情况下的短序列。

Native API 短序列时序图如图 3 所示。可以看到，异常 Native API 短序列反映出操作系统对同一 Native API 在一定时间内反复调用，而正常 Native API 短序列却是随机和无序的。这样的特征差别从病毒的作用机理上来看,也是符合其运行原理的。因此,可以通过对 Native API 短序列的调用情况来判定操作系统是否处于异常入侵之中。

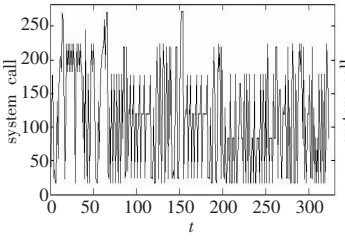

图 3(a)正常 Native API 短序列时序图

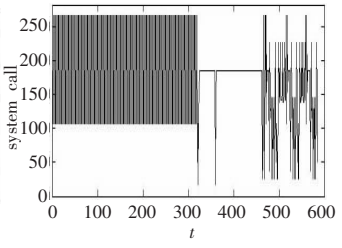

图 3(b)异常 Native API 短序列时序图

实验结果如表 4 所示。其中 BP 神经网络的输出为 0~1 之间的数值,以用来表示是否异常;而 Forrest 的方法则依据输出不匹配比例来进行判定。

表 4 实验的检测结果

漏洞	攻击类型	检测进程	输出平均值	不匹配比例/%	Forrest 时间/s	BP 时间/s
MS03-026	Buffer Overrun	svhost.exe	0.000 300	94.589 6	2.45	0.58
MS04-011	Buffer Overrun	lsass.exe	0.001 229	94.412 1	2.43	1.31
MS03-039	Buffer Overrun	svhost.exe	0.000 036	91.688 6	2.45	1.18
MS05-018	Buffer Overrun	csrss.exe	0.365 221	89.087 8	2.19	1.58
MS04-007	DOS	lsass.exe	0.119 656	93.245 8	2.45	1.29
MS06-021	Buffer Overrun	iexplore.exe	0.074 011	79.163 1	2.39	1.43
MS06-024	Buffer Overrun	wmplayer.exe	0.147 819	91.979 2	2.45	2.12
MS06-028	Buffer Overrun	powerpnt.exe	0.100 743	93.034 1	2.41	1.39

通过对两者输出结果的分析,可以看到其对异常入侵的检测都取得了相当准确的结果。但是,在 Forrest 的方法中,需要分析处理大量的数据来得到一个正常的序列库,包括尽量多的短序列,才能更好地描述系统的正常 Native API 调用情况。因此,检测时候对正常的序列库的遍历查找需要较长的时间。而 BP 算法学习时,收敛速度较快,训练需要的数据较少,网络收敛以后,检测速度较快并且检测精度也较高。所以从实验中看,对于同样多的序列,BP 算法的检测时间更短一些，适时性更好,更适合于在真实的情况下检测系统入侵。

由此,Native API 序列也反映了程序的执行情况，它跟在 Linux 中使用系统调用序列来描述程序的运行是一致的。正常程序产生系统的调用序列和异常程序产生的调用序列具有不同的特性,可以通过分析调用序列来分析程序是否正常。

另外,虽然目前在 Linux 下已经有很多基于系统调用序列的检测方法,但是由于 Linux 和 Windows 系统本身有很大的差异性,因此这里不能将本文的实验与 Linux 下基于系统调用序列的实验做比较。并且由于现在基于 Native API 序列异常检测的研究还很少,所以本文也没能找到其他的检测方法进行比较。

4 结束语

本文提出了一种在 Windows 平台上采用 BP 神经网络算法对 Native API 序列进行分类的异常入侵检测的方法，并通过实验证明 Native API 序列对于 Windows 异常入侵的可行性,在针对 Windows 的异常入侵检测的工作中具有很高的实际应用价值。其采用的 API 截获及相应的基于 Native API 序列的入侵检测方法，在 Windows 系统安全检测方面值得深入探讨。在今后的工作中,将对 Native API 序列进一步的研究,以期提出针对 Native API 序列的入侵检测更好的算法,提高检测的准确率和时实性。

参考文献:

[1] Lee W,Stolfo S J.Data mining approaches for intrusion detection[C]//Proc of the 7th USENIX Security Symp.San Antonio:USENIX,1998:6-9.

[2] Ye N.A Markov chains model of temporal behavior for anomaly detection[C]//Proc of the 2000 IEEE Workshop on Information Assurance and Security.United States Military Academy,West Point:IEEE Press,2000:171-174.

[3] Battistoni R,Gabrielli E,Mancini L V.A host intrusion prevention system for Windows operating systems[C]//9th European Symposium on Research in Computer Security,Sophia Antipolice,France,2004:134-142.

[4] 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,6-8 May 1996:120-128.

[5] Lane T.Hidden Markov models for human/computer interface modeling[C]//Proc of the International AI Society.Proc of the IJCAI-99 Workshop on Learning about Users.Stockholm:International AI Society,1999:35-44.

[6] Han S J,Cho S B.Evolutionary neural networks for anomaly detection based on the behavior of a program[J].IEEE Transactions on Systems,Man,and Cybernetics—Part B:Cybernetics,2006,36(3):559-570.

[7] Michael C,Ghosh A.Simple,state-based approaches to program-based anomaly detection[J].ACM Transactions on Information and System Security,2002,5(3):203-237.

[8] Sekar R,Bendre M,Dhurjati D,et al.A fast automaton-based method for detecting anomalous program behaviors[C]//IEEE Computer Society.Oakland:IEEE,2000:144-155.

[9] Nebbett G.Windows NT/2000 Native API Reference[M].[S.l.]:Macmillan Technical Publishing(MTP),2000-02-15.

[10] Zhang C L,Jiang J,Mohamed K.Intrusion detection using hierarchical neural networks[J].Pattern Recognition Letters,2005,26(6):779-791.

[11] Yu Zhen-wei,Tsai J J P,Weigert T.An automatically tuning intrusion detection system[J].IEEE Transactions on Systems,Man,and Cybernetics—Part B:Cybernetics,2007,37(2):373-384.

[12] 胡伍生.神经网络理论及其工程应用[M].北京:测绘出版社,2006-01.

[13] 宋歌,喻建平.神经网络在异常检测中的应用[J].计算机工程与应用,2002,38(18):146-148.

[14] 李晓峰.人工神经网络 BP 算法的改进及其应用[J].四川大学学报:工程科学版,2000,32(2):105-109.