文章编号: 1001- 9081(2005)09- 2050- 04

基于模糊模式识别的未知病毒检测

张波云[1,2], 殷建平[1], 唐文胜[1], 蒿敬波[1]
(1. 国防科技大学 计算机学院, 湖南 长沙 410073;
2 湖南公安高等专科学校 计算机系, 湖南 长沙 410138)
(hnjxzby@yahoo.com.cn)

摘 要: 提出了一种基于模糊模式识别的检测方法来实现对计算机病毒的近似判别。该方法可以克服病毒特征代码扫描法不能识别未知病毒的缺点。在该检测方法的基础上, 文中设计了一个病毒检测网络模型, 此模型既可以实现对已知病毒的查杀, 又可以对可疑程序行为进行分析评判, 最终实现对未知病毒的识别。

关键词: 计算机病毒; 模糊模式识别; 模糊集; 病毒检测
中图分类号: TP309.5 文献标识码: A

Unknown computer virus detection based on fuzzy pattern recognition

ZHANG Bo2yun[1,2], YIN Jian2ping[1], TANG Wen2sheng[1], HAO Jing2bo[1]
(1. School of Computer Science, National University of Defense Technology, Changsha Hunan 410073, China;
2 Department Computer Science, Hunan Public Security College, Changsha Hunan 410138, China)

Abstract A fuzzy recognition algorithm to detect computer virus approximately was present. It could overcome the shortage of normal virus scanner- which could not detect unknown virus. Based on this method, a virus detect network model was designed. This model can detect virus in the on2line system, and it can also detect known and unknown computer virus by analyzing the program behavior.

Key words computer virus; fuzzy pattern recognition; fuzzy set; virus detection

0 引言

Fred Cohen博士在 20世纪 80年代指出了/计算机病毒检测的不可判定性0[1], 即/精确检测计算机病毒是不可判定的0。此后 Diomidis Spinellis证明了/有界长度病毒的可靠检测是一个 NP完全问题0[2]。因此长期以来, 各反病毒研究机构都在努力探讨病毒检测的近似算法, 在 IBM 病毒研究中心, 曾经成功地将神经网络用于判断引导型病毒[3], 他们声称在不久的将来把人工免疫方法应用于病毒检测产品中去。基于上述思想的启发, 本文采用一种基于模糊模式识别(Fuzzy Pattern Recognition, F2PR)[4]的模型来实现对计算机病毒的检测。

当前的计算机病毒检测技术主要基于特征检测法, 其基本方法是提取已知病毒样本的特征, 并将此特征数据添加到病毒特征库中, 在病毒检测时通过搜索病毒特征库查找是否存在相匹配的病毒特征来发现病毒。这种检测办法只能用于检测已知的病毒, 对于新出现的病毒的检测无能为力。为了解决这一问题, 本文提出一种基于模糊模式识别分类算法的检测方法来实现对计算机病毒的近似判别。采用模糊模式识别算法对提取的可疑文件行为特征进行分析, 并利用病毒程序与正常程序的行为特征的差异性进行分类, 从而达到检测未知病毒的目的。

M. Schultz等[5]曾提出用数据挖掘的算法如朴素贝叶斯算法等检测未知恶意代码, 其算法建立在对程序静态分析的基础上, 他们直接选用程序的机器码 (用 hex dump获取)、程序中的 ASCII字符串以及对 PE 程序的静态分析而取得的 API引用序列作为样本程序的特征向量, 各算法以此为基础进行学习与分类。显然, 该法对目前日渐盛行的变形病毒的检测无能为力, 因为变形病毒的机器代码序列变化十分迅速[8]。本文的分类算法主要针对程序的执行过程中使用的系统 API函数的调用, 它监视程序的行为并进行分析, 能有效地检测未知病毒和各种多态与变形病毒。

在该检测方法的基础上, 本文设计了一个恶意代码检测网络模型, 从模型的设计和技术实现的角度进行分析, 介绍了它的工作原理。该模型既可以实现对已知病毒的查杀, 又可以对可疑文件进行分析评判, 最终实现对未知病毒的识别。最后我们对模型的功能进行了实验测试, 结果达到了很好的实际应用效果.

1 模型结构分析

系统结构框图如图 1所示。系统由病毒防火墙、应用服务器和病毒检测服务器组成, 进入系统的可疑文件首先经过系统的第一道防线病毒防火墙的检测, 若没有检测出病毒, 则将其复制两份: 一份拷贝直接存放于应用服务器中; 一份拷贝则进入基于 F2PR的检测服务器中, 对其行为特征进行检测, 如果发现其可疑行为, 判断为感染了未知病毒, 检测服务器则将信息反馈给应用服务器, 然后在应用服务器中对该文件进

收稿日期: 2005- 03- 14; 修订日期: 2005- 05- 22
基金项目: 国家自然科学基金资助项目(60373023); 湖南省自然科学基金资助项目(04JJ6032)
作者简介: 张波云(1972-), 男, 湖南永州人, 讲师, 博士研究生, 主要研究方向: 计算机病毒、网络安全; 殷建平(1964-), 湖南益阳人, 教授, 博士生导师, 主要研究方向: 算法设计与分析、人工智能、模式识别、信息安全; 唐文胜(1970-), 男, 湖南永州人, 副教授, 博士研究生, 主要研究方向: 移动计算、无线网络安全; 蒿敬波(1976-), 男, 河北魏县人, 博士研究生, 主要研究方向: 网络安全.

行隔离、监管或删除。由于基于程序行为的杀毒系统可能引发对系统不可预料的破坏, 故在该系统中配置了专门进行未知病毒检测的检测服务器, 它对正常服务器的性能影响极微, 十分适用于实时在线系统中的未知病毒检测。而对于检测服务器自身的安全防护, 可以采用系统还原保护法等。

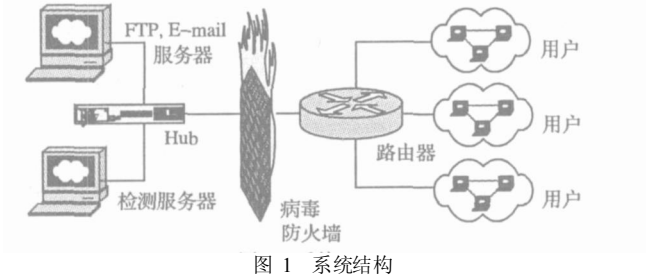

图 1　系统结构

检测服务器中的病毒检测引擎采用 F2PR 算法实现。第一阶段, 系统首先进行自我训练, 采用特征提取工具对样本空间中的训练样本程序进行特征提取, 并保存于训练样本特征库中以备将来学习和检测病毒之用。第二阶段, 当外来文件进入系统时, 系统响应过程如图 2所示。首先经过病毒防火墙的第一次测试, 若发现其已染有病毒, 则在应用服务器中对其隔离, 保存入病毒隔离库中; 若没发现病毒, 则在检测服务器中继续调用 F2PR检测引擎测试。将可疑程序特征提取出来以获取被检测文件的特征矢量, 然后用 F2PR 算法对可疑程序进行病毒判定。若判断为病毒, 则将信息反馈给应用服务器, 在应用服务器中对病毒文件进行隔离、监管或删除, 也可以将其发送给相关专家进行进一步精确分析。与此同时把新病毒样本添加到训练样本空间, 重复上述过程进行新一轮的学习。

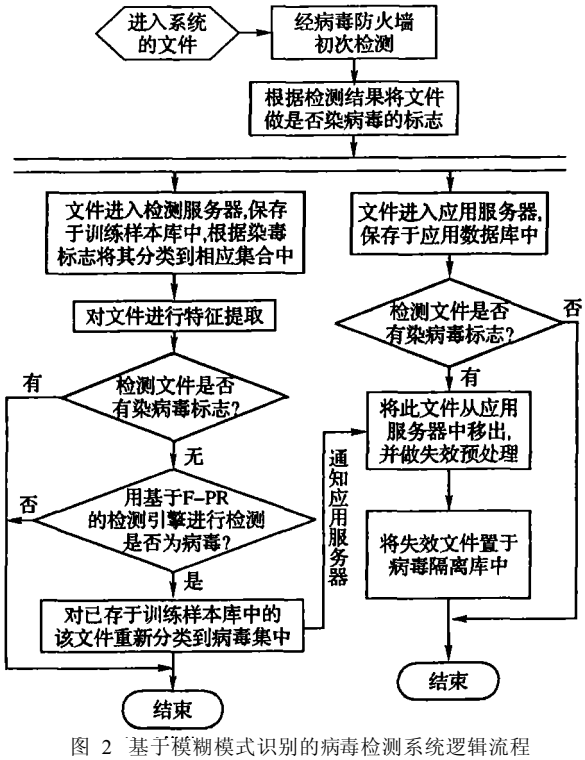

图 2　基于模糊模式识别的病毒检测系统逻辑流程

2　模糊模式识别数学模型

设有 n个样本组成样本集合:

$$X = \{x_1, x_2, \ldots, x_n\} \quad (1)$$

每个样本用 m个指标特征向量表示:

$$x_j = (x_{1j}, x_{2j}, \ldots, x_{mj})^T \quad (2)$$

则样本集可用 m @n阶指标特征值矩阵表示:

$$x_{m@n} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} = (x_{ij}) \quad (3)$$

x_{ij} 为样本 j指标 i的特征值, i = 1, 2, , , m; , j = 1, 2, , , n。

由于 m 个指标特征值物理量的量纲不同, 在进行识别时要先消除指标特征值量纲的影响, 使指标特征值规格化, 即将指标特征值矩阵变换为指标特征值的相对隶属度矩阵:

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \cdots & \cdots & & \cdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix} = (r_{ij}) \quad (4)$$

其中, r_{ij} 为指标特征值规格化数或相对隶属度, 0 [r_{ij} [1。

设将 n个样本依据样本的 m个指标特征按 c个级别 (或类别) 加以识别, 其模糊识别矩阵为:

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ \cdots & \cdots & & \cdots \\ u_{c1} & u_{c2} & \cdots & u_{cn} \end{bmatrix} = (u_{hj}) \quad (5)$$

其中, u_{hj} 为样本 j从属于级别 h的相对隶属度, h = 1, 2, , , c。满足条件:

$$\begin{cases} \sum_{h=1}^{c} u_{hj} = 1, & P\ j\ \ 0 [u_{hj} [1 \\ \sum_{j=1}^{n} u_{hj} > 0, & P\ h \end{cases} \quad (6)$$

设每个级别 h有 m 个指标特征值 (称为标准指标特征值), 则 c个级别的指标特征可用 m @c阶标准指标特征值矩阵表示:

$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1c} \\ y_{21} & y_{22} & \cdots & y_{2c} \\ \cdots & \cdots & & \cdots \\ y_{m1} & y_{m2} & \cdots & y_{mc} \end{bmatrix} = (y_{ih}) \quad (7)$$

式中 y_{ih} 为级别 h指标 i的标准指标特征值。

对于一个要分类样本, 获得其 m 个特征的隶属度 $L_1, L_2, , , L_m$, 而这 m 个特征在分类中的作用是不同的, 故对这些隶属度分别加上适当的权数: $a_1, a_2, , , a_m$, 然后求得:

$$F = \sum_{i=1}^{m} a_i L_i$$

若给定阈值 (限度)H, 当满足 F \ H时即将所论对象划分到某一类, 并且根据 H的不同而分类, 从而达到分类的目的。同时也可计算相关贴近度, 用择近原则分类。学习中的离散目标分类函数为: f X_j^m y C, 其中 C是有限集合 $\{C_1, C_2, , , C_n\}$, 即各不同分类集。

3　未知病毒检测

3 1　病毒特征提取

病毒与一般程序的区别在于是否执行了一些特殊的动作来破坏系统。从外在表现来看, 如木马程序常进行如下的操作: 对 win. ini文件夹特定项的修改; 对 system. ini文件的特定项的修改; 对注册表特定键值的修改, 文件关联; 端口异常与此端口上的数据流量; 硬盘数据共享; 远程文件操作; 远程抓屏操作; 计算机关闭与启动; 鼠标键盘的操纵; 远程执行可执行程序; 消息发送; 进程管理; 更改服务; 文件加壳; 修改注册表; 文件打开; 文件复制; 文件修改等。

不管是二进制可执行病毒、脚本病毒还是宏病毒, 它们都

是一种程序, 它需要调用操作系统提供的各种功能函数才能达到传播自身和破坏系统的目的。因此可以将下列信息作为待检测程序的行为特征: (a)程序使用的动态链接库 (DLL)文件; (b)程序在所使用的 DLL中调用的 API函数[6]; (c)程序的机器代码序列或源代码等。

在试验过程中, 主要检测程序用到 API函数。对样本库中的程序进行 API调用跟踪处理后, 可以得到大量的系统调用序列, 这些不同的 API对于病毒识别所起的作用是不一样的。可以想象, 当一个 API调用在病毒文件中出现的频率非常高, 而在正常程序文件中出现频率较低时, 该 API对识别病毒所作的贡献就比较大。因此尽量提取此类 API构成/模糊特征集0。在此, 采用概率统计学中的均方差法来进行实验。类间频率均方差能较好地体现各不同的 API调用的/贡献程度0, 一个被调用 API的分类作用与它的类间频率均方差成正比。均方差的计算过程如下: (1)对所有训练库的样本进行系统调用跟踪, 获得调用的 API序列 A = {$A_1, A_2, \ldots, A_i$}, (1[i[p), 统计各 API(即 A_i)在每一病毒程序 V_j中出现的频率 A_{ij}^V及在每一正常程序 N_j中出现的频率 A_{ij}^N; (2)计算每个被调用的 API在病毒程序与正常程序文件中出现的频率均值 $E(A_i^V) = \frac{1}{s}\sum_{j=1}^{s} A_{ij}^V$, s为病毒样本数量, $E(A_i^N) = \frac{1}{n}\sum_{j=1}^{n} A_{ij}^V$, n为正常程序数量; (3)计算每个 API的总出现频率均值 $E(A_i) = \frac{E(A_i^V) + E(A_i^N)}{2}$; (4)计算每个 API的类间频率均方差:

$$D(A_i) = \sqrt{[E(A_i) - E(A_i^V)]^2 + [E(A_i) - E(A_i^N)]^2}$$

然后, 将所有的 API按均方差大小排序, 选出前 t个组成/模糊特征集0。我们通过实验后选取了 88个主要 API, 如表1所示。并对不同的程序的行为特征划分为三个危险级别: 即一般、较严重、严重, 在表中分别用m、m m、m m m表示。

然后, 我们编写了相应的特征提取工具对待检测程序的上述特征进行获取。

表 1 特征列表

序号	程序行为	相关 API 调用	危险度	动态链接库
1	堆操作	RtlFreeHeap; RtlAllocateHeap	☆☆	NTDLL.DLL
2	动态库加载与释放	LoadLibraryA; FreeLibrary; GetModuleHandleA; GetModuleFileNameA	☆	KERNEL32.dll
3	API 地址获取	GetProcAddress	☆	
4	进程操作	OpenProcess; CloseHandle; CreateRemoteThread; ExitThread; WaitForSingleObject; CreateThread	☆☆	
5	内存读写	VirtualAlloc; GetProcessHeap; VirtualAllocEx; WriteProcessMemory; VirtualFreeEx; OpenMutexA; CreateMutexA; VirtualProtect; VirtualFree; HeapFree	☆☆	
6	读注册表	RegCloseKey; RegEnumKeyExA; RegOpenKeyExA; RegQueryValueExA; RegNotifyChangeKeyValue	☆	
7	写注册表	RegSetValueExA; RegCreateKeyExA ; RegDeleteKeyA	☆☆	
8	程序执行	WinExec; CreateProcess	☆☆	
9	文件读&创建	CreateFileA; ReadFile; OpenFile	☆	
10	文件写	WriteFile; WriteFileEx	☆☆	
11	文件删除	DeleteFileA	☆☆☆	
12	文件移动	MoveFileA; MoveFileExA;	☆☆	
13	变更属性权限	SetFileAttributesA; SetFileTime	☆☆	
14	文件搜索	FindClose; FindFirstFileA; FindNextFileA; FindResourceA	☆☆☆	
15	目录搜索	GetWindowsDirectoryA; GetSystemDirectoryA; SetCurrentDirectoryA; CreateDirectoryA	☆☆☆	
16	目录删除	RemoveDirectoryA	☆☆☆	
17	磁盘操作	GetDiskFreeSpaceA; GetDriveTypeA	☆	
18	时间操作	GetTickCount; GetSystemTime; GetTickCount; GetLocalTime	☆	
19	系统重启	Exitwindows; ExitwindowsEx; AbortSystemShutdown; InitialteSystemShutdown	☆☆	
20	加密与解密	CryptAcquireContextA; CryptGenKey; CryptDestroyKey; CryptImportKey; CryptExportKey; CryptEncrypt; CryptDecrypt; CryptReleaseContext	☆☆☆	ADVAPI32 .dll
21	远程通讯	WSAStartu; Socket; Connect; Recv; Send; Bind; Listen; Accept; Gethostname; Gethostbyname; Closesocket; WSACleanup	☆☆	WSOCK32.dll
22	进程搜索	EnumProcesses; EnumProcessModules; GetModuleBaseNameA	☆☆	PSAPI.dll
23	注册表操作二	SHDeleteValueA; SHGetValueA; SHSetValueA	☆☆	SHLWAPI.DLL

3.2 病毒检测引擎

通过搜集大量的同类程序可执行代码可以构成样本空间, 利用现有的病毒检测工具将每一个程序准确的归为两种类型: 病毒程序和普通正常程序。将样本空间划分为训练集 $U_{training}$和测试集 U_{test}两个集合 (训练集和测试集为样本空间中两个不相交的子集 $U_{training}$ H U_{test} = Á , $U_{training}$ G U_{test} = U)。另将训练集划分为病毒程序集 V和普通程序集 N, 且 VH N = Á , VG N = $U_{training}$。同样, 测试集也划分为病毒程序集 Vc和普通程序集 Nc, 且 VcH Nc = Á , VcG Nc = U_{test}。然后根据病毒类型构造相应的特征提取工具对程序进行特征提取, 并应用 F2PR算法对训练集进行分类。同时, 当新的病毒样本添加到样本空间后。可以重复上述过程进行新一轮的学习。

计算机病毒检测是一个二值分类问题, 即病毒与非病毒两类。对于样本空间中的每一个样本程序 x我们均可以从中提取出感兴趣的一组特征集 A, 定义 C为分类集, 即 {正常, 病毒 }, 令 C_1表示正常, C_2表示病毒, 它是一个随机变量。我们的目标是: 在获得给定样本程序文件中的特征集 F后, 判别出该样本是正常程序或是病毒程序。

因为一个程序具有相应的特征, 故可以对某对象或对象类通过它所具有的特征来描述, 进而可用一个定义在特征类上的模糊集来描述它们。对于程序文件而言, 设 Q = {$q_1, q_2, \ldots, q_n$} 为由 n个特征量组成的论域, 则一个可执行程序或程序类可用定义在论域 Q上的一个模糊集来描述, 即 M = {$L_1/q_1, L_2/q_2, \ldots, L_n/q_n$}。它表示文件具有特征 q_i的隶属度是 L_i, 其中 L_i是 [0, 1] 间的一个实数。同理, 病毒程序类与正常程序类也可用 Q上的模糊集来描述。

设某 API函数在病毒文件中出现的频率均值为 $E(A_i^V)$, 在正常文件中出现的频率均值为 $E(A_i^N)$, 则用 F 分布中的正态偏大型模糊分布来构造病毒程序集 V的隶属函数:

$$L_V(E(A_i^V)) = \begin{cases} 0, & E(A_i^V) < 0 \\ 1 - e^{-(E(A_i^V))^2/R^2}, & E(A_i^V) \geq 0 \end{cases} \quad (8)$$

式中 $R = \max\{E(A_1^V), E(A_2^V), \cdots, E(A_t^V)\}/3$, t为特征量的个数。

同样构造正常程序集 N的隶属函数为:

$$L_N(E(A_i^N)) = \begin{cases} 0, & E(A_i^N) < 0 \\ 1 - e^{-(E(A_i^N))^2/R^2}, & E(A_i^N) \geq 0 \end{cases} \quad (9)$$

对于待检测程序文件, 其隶属函数为:

$$L_M(A_i) = \begin{cases} 0, & A_i < 0 \\ 1 - e^{-(A_i)^2/R^2}, & A_i \geq 0 \end{cases} \quad (10)$$

在检测引擎自学习阶段, 对表 1中所列特征量在病毒程序测试集中出现的频率进行统计, 并据隶属函数 $L_V(E(A_i^V))$ 计算得到模糊集:

$$V = \{L_1^V/A_1, L_2^V/A_2, \cdots, L_t^V/A_t\} \quad (11)$$

同样, 对它们在正常程序测试集中做同样处理, 可以得到模糊集:

$$N = \{L_1^N/A_1, L_2^N/A_2, \cdots, L_t^N/A_t\} \quad (12)$$

对于进入系统的待检测文件M, 首先对它进行 API调用跟踪, 获得其调用序列。并对表 1中所列特征量在其中出现的频率进行统计, 得到: $\{A_1, A_2, \cdots, A_t\}$, 式中 t= 88, 由隶属函数 $L_M(A_i)$ 计算可得到该文件的模糊集:

$$M = \{L_1/A_1, L_2/A_2, \cdots, L_t/A_t\} \quad (13)$$

然后, 计算 M与 V之间的贴近度 W(M, V) 以及 M与 N之间的贴近度 W(M, N)。在此采用的是 Euclid贴近度, 其计算公式为:

$$W(A, B) = 1 - \frac{1}{\sqrt{t}}\left[\sum_{i=1}^{t}(L_i^A - L_i^B)^2\right]^{\frac{1}{2}} \quad (14)$$

然后根据/ 择近原则0 将待检测文件归于病毒类或正常类。该原则的分类思路即: 对于模糊集 $A_i, B(i = 1, 2, \cdots, n)$, 若存在 i使得 $W(A_i, B) = \max_{1 \leq j \leq n} W(A_j, B)$, 则认为 A_i 与 B 最贴近, 将 A_i 与 B 归于一类。

实验中使用的基于模式识别的病毒检测算法如下所示:

1　训练算法

t　输入: 训练库中的全部样本

t　输出: 模糊集 V及 N

t　过程:

1) 获取每个样本的 API调用序列;

2) 计算每个 API函数在病毒程序集中出现的频率以及在正常程序集中出现的频率, 然后求得类间频率均方差 $D(A_i)$;

3) 根据类间频率均方差值降序排列, 选取前 t个 API, 构成模糊特征集;

4) 根据特征集中各 API函数在病毒集与正常集中出现的频率以及相应的隶属函数求得模糊集:

$V = \{L_1^V/A_1, L_2^V/A_2, \cdots, L_t^V/A_t\}$

$N = \{L_1^N/A_1, L_2^N/A_2, \cdots, L_t^N/A_t\}$

5) 输出 V及N, 算法结束。

2　分类算法:

t　输入: 待检测文件

t　输出: 是否病毒

t　过程:

1) 获取待测程序的 API调用序列;

2) 计算各特征量出现的频率, 结合隶属函数, 求得模糊集:

$M = \{L_1/A_1, L_2/A_2, \cdots, L_t/A_t\}$;

3) 依据训练过程得到的 V与 N, 计算 Euclid贴近度 W(M, N) 及 W(M, N);

4) 根据/ 择近原则0 对检测文件分类;

5) 输出类别, 算法结束。

4　实验分析

实验的样本数据如表 2所示。样本空间中样本总数为 632, 分为正常程序与染毒程序。正常程序从操作系统平台中选取, 选用 Windows 2000 Server首次安装后, 机器中的全部 PE文件共 423个。通过各种途径收集的病毒程序 209个。为获得样本空间中程序的特征向量, 编写了 API调用跟踪器, 可以实现 Windows 2000 Server环境下的全部 API函数调用的拦截。

表 2　用于实验的样本数据

	样本空间	训练集	测试集
正常程序	423	373	50
染毒程序	209	159	50
合计	632	532	100

实验目标主要是对 F2PR查病毒引擎的错误率进行测试。我们将错误类型分为两种, 即: (1)将正常程序判断为病毒, 称为 False Negative; (2)将病毒程序判断为正常程序, 称为 False Positive。

用测试集 (共 532个文件) 作为训练数据先对分类器进行训练, 然后对测试集中的文件 (共 100个) 进行分类测试, 并进行自学习。然后用全体数据集作为训练数据, 测试集从该样本空间中随机选出不等数目的文件而获得, 实验结果见图 3。

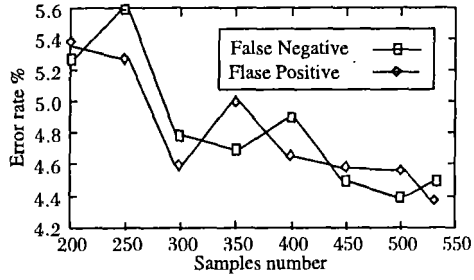

图 3　基于模糊模式识别的检测系统性能

从实验结果知分类器用较少训练集中的样本测试与用较多训练样本进行测试获得的精度几乎相等, 这对于计算机病毒样本较难获得的情况下检测未知病毒非常有用。

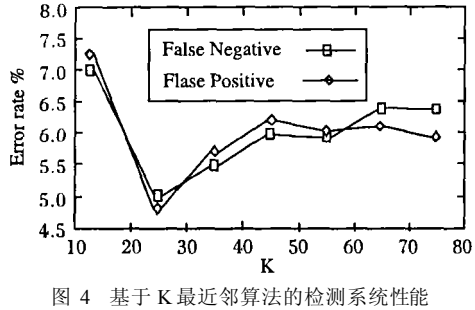

图 4　基于 K最近邻算法的检测系统性能

在文献 [7]中利用基于实例学习的 K2最近邻算法对本组样本空间中的数据进行过测试, 结果如图 4所示, 基于 KNN算法的检测系统在 K = 25时的 False Negative error rate最小值约为 4.8%。通过对这两种不同的算法结果的比较, 我们发现使用模糊模式识别分类算法得到的 False Negative error rate最小值约为 4.45%, 效果稍好。但 KNN算法开销很小, 而 F2PR算法的开销则大得多。在实际应用中这二方面都要兼顾。

(下转第 2062页)

成员数目一定的情况下, 如果扩展到 m2LKH, 则可以相应减少组成员存储的密钥数。当 m扩展到组成员数目 n相等时, 则成为 GKMP方案。

复杂性分析: 如表 1所示及上文描述, 当 n是组成员的数目且m是 LKH的维度时, 则此 LKH树的深度是 $h = \log_m n + 1$。相应地, 成员加入和离开阶段的通信代价是 h + 2和 (m - 1)(h - 1)。因为每个成员的密钥存储量随着 LKH树的深度而变化, 因此每个组成员的密钥存储量为 h + 1。

表 1 m2LKH与组成员数目的关系 ($h = \log_m n + 1$)

	2维	m维	n维
成员加入时的密钥通信量	h + 2	h + 2	4
成员离开时的密钥通信量	h - 1	(m - 1)(h - 1)	n - 1
组成员的密钥存储量	h + 1	h + 1	3

有效性分析: 基于 LKH树结构方案的主要性能瓶颈是密钥存储量和密钥更新代价。如表 2所示, 使用二维树结构来和其他三种方案在 6个方面的进行分析和比较。本方案在成员加入或离开群组时的密钥通信量以及转发时是否需要解开组播信息方面和 WGL方案、批更新方案相当, 但性能略低于 IOLUS方案; 在转发时延方面优于 IOLUS方案和批更新方案; 但在 GC的密钥存储量方面明显优于其他三种方案, 只需要 O(1), 是一个常量。而同谋攻击的概率也明显低于 IOLUS方案, 在组规模较大时性能更佳。

表 2 在 22LKH情况下和其他方案的比较 ($K = \log_2 n$)

	本方案	WGL	IOLUS	批更新方案
成员加入时的密钥通信量	O(K)	O(K)	O(1)	O(K)
成员离开时的密钥通信量	O(K)	O(K)	O(1)	O(K)
GC的密钥存储量	O(1)	O(n)	O(m)	2m + 1
同谋攻击概率	低	较低	较高	)))
转发延时	短	短	较长	长
转发时需要解开组播信息	不需要	不需要	需要	不需要

5 结语

安全组播的密钥管理是保证组播技术能够广泛应用的核心问题之一。本方案采用了门限单向函数, 充分考虑了系统对减少 GC密钥存储量和降低同谋破解概率的要求, 可以较好的解决组播成员动态变化带来的安全隐患。成员加入后, 通过公开的单向函数计算该成员的 ID来避免非法成员的仿冒攻击, 然后和组内的其他成员同时自行计算组播密钥, 极大的减少了 GC的计算负载; 成员退出后, 由于门限函数的特性, 如果同时退出的成员数目 k< t, 则 k个成员无法完成同谋破解。当组播内的成员越多, 同时退出的成员要满足拉格朗日条件的概率会更低, 因此本方案非常适合可扩展的大规模组播应用。

参考文献:

[1] HARNEY H, MUCKENHIRN C. Group key management protocol (GKMP) specification[S]. RFC 2093, July 1997.

[2] HARNEY H, MUCHENHIRN C. Group key management protocol (GKMP) architecture[S]. RFC 2094, July 1997.

[3] WONG CK, GOUDA M, LAM SS. Secure group communication u2sing key graphs[J]. IEEE/ACM Transactions on Networking, 2000, 8(1): 16-30.

[4] MITTRA S. Iolus: A framework for scalable secure multicasting[J]. ACM SIGCOMM Computer Review, 1997, 27(4): 277-288.

[5] TSENG YM. A scalable key management scheme with minimizing key storage for secure group communication[J]. International Journ2al of Network Management, 2003, 13(6): 419-425.

[6] CHANG L, ENGLER, KANDLUR D. Key Management for Secure Internet Multicast Using Boolean Function Minimization Techniques [A]. IEEE Infocomm99[C], 1999. 689-698.

[7] LI XS, YANG YR, GOUDA MG, et al. Batch Rekeying for secure group communications[A]. Proceedings of the 10th international conference on World Wide Web[C]. ACM Press, 2001. 525-534.

[8] 徐明伟, 董晓虎, 徐恪. 组播密钥管理的研究进展[J]. 软件学报, 2004, (15): 141-150.

(上接第 2053页)

5 结语

由于计算机病毒检测的不可判定性, 本文尝试用基于模糊模式识别的算法来实现对计算机病毒的检测, 在此基础上, 设计了一个恶意代码检测网络模型, 较适用于实时在线系统中的未知病毒检测。

本检测模型结合了特征码检测技术与模糊智能学习技术, 检测成功率较高。因为我们选择的程序特征向量是 API调用序列, 故它能有效地对付变形与多态病毒。系统开销主要集中于程序特征的提取和隶属度的获得, 为简便起见我们采用的是统计方法来计算后者。

本系统是一个病毒检测器, 只能隔离病毒而不能实现对病毒的清除, 与其他反病毒工具的集成是一个可行的解决方案。因为基于病毒行为的检测会对系统造成不可预料的破坏性, 故如何寻找高效与安全的特征提取工具是我们以后研究的要点。

在我们的测试床中, 将计划进行其他的基于机器学习算法的恶意代码检测实验, 以期对其他的恶意代码如木马、蠕虫、间谍程序等进行测试。并对各种方法的性能与开销进行对比选择和优化以期用于实际的产品中。

参考文献:

[1] COHEN F. Computer Viruses: Theory and Experiments[J]. Com2puters& Security, 1987, 6(1): 22-35.

[2] SPINELLIS D. Reliable Identification of Bounded2Length Viruses Is NP2Complete[J]. IEEE Trans2actions on information Theory, 2003, 49(1): 280-284.

[3] TESAURO GJ, KEPHART JO, SORKIN GB. Neural networks for computer virus recognition[J]. IEEE Expert, 1996, (8): 5-6.

[4] BARGIELA A, PEDRYCZW, HIROTA K. Granular prototyping in fuzzy clustering[J]. IEEE Trans2actions on Fuzzy Systems, 2004, 12(5): 697-709.

[5] SCHULTZ M, ESKIN E, ZADOK E, et al. Data mining methods for detection of new malicious executables[A]. Proceedings of the IEEE Symposium on Security and Privacy[C]. Los Alamitos, CA, 2001. 38-49.

[6] KRUGLINSKI D, WINGO S, SHEPHERD G. Programming Visual C++ [M]. Washington: Microsoft Press, 1998.

[7] 张波云, 殷建平, 张鼎兴, 等. 基于 K2最近邻算法的未知病毒检测[J]. 计算机工程与应用, 2005, 41(6). 7-10.

[8] 祝恩, 殷建平, 蔡志平, 等. 计算机病毒自动变形机理的分析[J]. 计算机工程与科学, 2002, 24(6): 7-10.