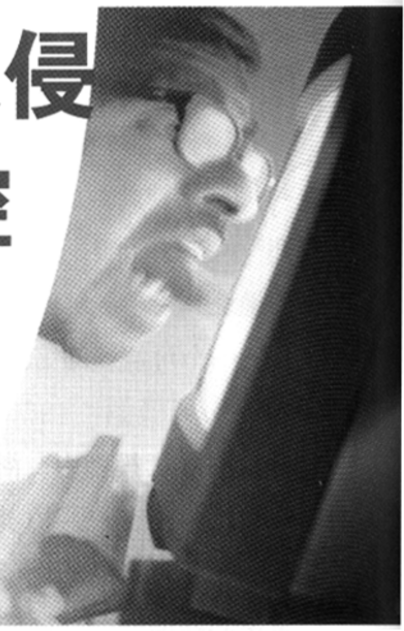
基于windows NT主机入侵检测系统的文件和进程监控



冯德旺 兰建容 谢纯珀 （福建农林大学计算机与信息学院 353001）

摘要：主机入侵检测系统可以实时监测主机的各种状态，辨别可能发生的入侵行为或非法操作，在入侵行为发生的时候自动阻断非法操作，保护主机系统不受入侵。本文主要介绍了其中文件监控和进程监控这两个模块。

关键词：主机监控 文件系统 进程 驱动程序

1 引言

主机入侵检测系统是与网络防火墙、网络入侵监测代理、系统监控台结合在一起组成的一套针对网上黑客非法入侵的防范软件。

网络防火墙是数据进出内部网络的大门，通过它来屏蔽内部网络，使外部网络的非法用户无法访问内部网络；自身可以实时判断和过滤来自外部网络的不良数据包和黑客的攻击企图；还可以实时响应来自内部网络其他子系统的阻断请求，对来自外部网络的攻击行为进行阻断。

网络入侵监测代理可以24小时不间断地对受保护网段上的数据包进行侦听和分析，判断是否发生入侵行为。在入侵行为发生时，实时做出响应，记录所发生的攻击事件，对危险级别高的事件请求网络防火墙和主机入侵监测代理对攻击行为进行阻断。

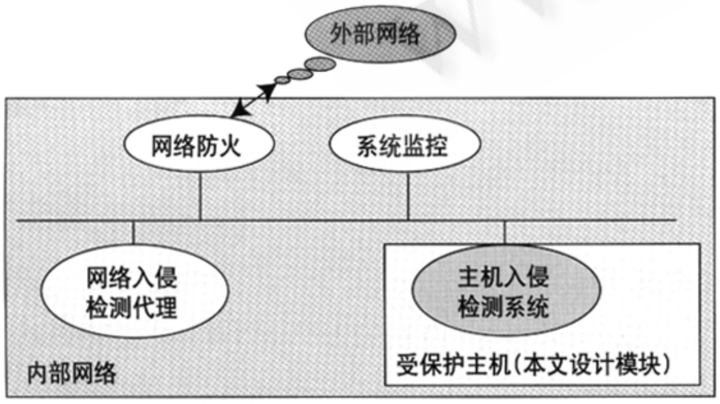


图1 结构框图

系统监控台为系统管理员提供一个友好的用户界面，负责收集其他子系统发送来的消息，并以适当形式报告管理员发生在受保护系统中的入侵行为。

主机监控是装在受保护的主机系统中，通过实时地检测和分析系统的各种状态，不论是来自内部网络还是外部网络的非法请求或非法操作，都应给出通知（以适当的形式报告给管理员），且应对非法操作给予阻断，并记录在日志文件中，供管理员分析。

如图1所示，这四个子系统是相互协同工作，共同实现对网络和主机的保护。

2 Windows NT 主机入侵检测系统

在受监控机上安装主机入侵检测系统，安装程序将把主机监控系统及一些设备驱动程序添加到服务控制管理器SCM（Service Control Manager）中，并把启动类型设为自动，让系统启动时自动运行该服务。主机监控服务启动后，可实时检测系统中的信息，并可根据配置文件对一些非法操作直接进行阻断，并保存在日志文件中，同时把日志文件的监控信息及一些响应操作送到管理端，供管理员分析。

Windows NT 主机入侵检测系统主要分为：文件监控模块、进程监控模块、网络连接监控模块、注册表监控模块、性能监控模块和安全日志事件监控模块，其中网络连接监控模块可包括TCP/IP连接监控及端口监控的一些方面（因为每一个网络连接监控都要给出其连接的远程和本地的IP地址及所连接的端口号）。本文主要介绍其中的文件监控和进程监控模块。

3 文件监控模块

3.1 实现功能

能够实时监控在被监控机上文件系统的运行情况，

指出当前运行的文件正在使用哪些进程、调用哪些DLL文件，追踪系统中和应用程序配置中的错误，监控任何用户应用程序对文件操作时传给文件设备驱动程序的IOCTL控制码，指出操作的结果。对某个用户文件或系统文件进行读、写、删除、关闭等操作时，能够映射出当前操作的时间，记录文件的完整路径及该文件调用了哪些进程，操作的结果记录在日志文件中。当用户应用程序想对文件进行操作时，可以利用中间层文件设备驱动程序（处在应用程序和文件设备驱动程序之间）来获取控制码，检查用户程序的权限，进行必要的阻断,实现文件监控和保护，并把日志文件送到管理端。

3.2 实现原理

3.2.1 Windows NT 文件系统

文件系统可以对怎样命名文件、怎样保护文件、怎样在发生系统错误后恢复文件以及将文件存放在什么介质上等进行控制。Windows NT支持三种文件系统:FAT、HPFS和NTFS，其中只有NTFS支持较多的安全特色，并能很好地从磁盘错误中恢复，在CPU失败、系统崩溃或I/O出错后，它可以通过参考其文件改变日志而快速地恢复磁盘信息。当一个严重的错误使不完整的交易被遗留在缓冲区后，磁盘缓冲系统就可能破坏文件，NTFS系统可以通过将每次文件交易保存在日志文件中以避免这种错误。

3.2.2 Windows NT 设备驱动程序

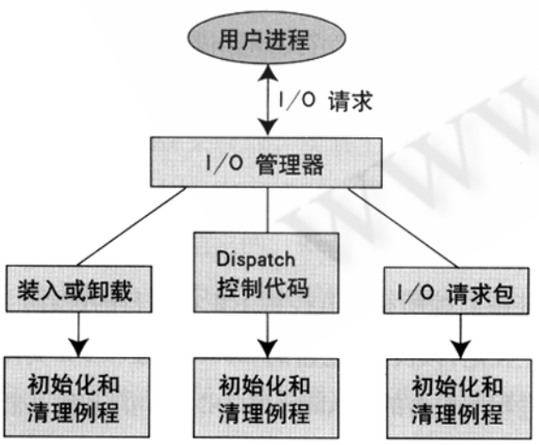


图2 数据控制流程

设备驱动程序是管理某个设备（包括虚拟设备）的一段代码，它负责进程的数据缓冲区与外部设备缓冲区的数据交换。Windows NT以两种模式来运行，即内核模式与用户模式，驱动程序以内核模式来运行，普通应用程序以用户模式运行。用户模式中要对硬件平台进行访问，必须经过内核模式，设备驱动程序就是沟通二者的桥梁。为了使用户模式能有效地对内核模式进行访问和控制，I/O管理器定义了一些类似IRP_MJ_XXX的操作代码（根据需要，用户也可自定义这些代码），每个操作代码都有相对应的派发（Dispatch）例程。当用户进程调用诸如DeviceIoControl等I/O控制函数时，这些操作代码通过I/O管理器将IRP包（即I/O请求包）发送给驱动程序，I/O管理器根据操作代码调用相应的派发例程来执行，执行完毕后将数据返回供用户进程使用，其数据控制流程如图2所示。

3.2.3 利用 windows NT 设备驱动程序对文件进行保护

当黑客入侵系统后，往往能通过各种攻击手法取得管理员的权限，因此就可以对文件进行随意的修改、破坏。下面就介绍一下如何利用windows NT中间层设备驱动程序实现对文件的保护。

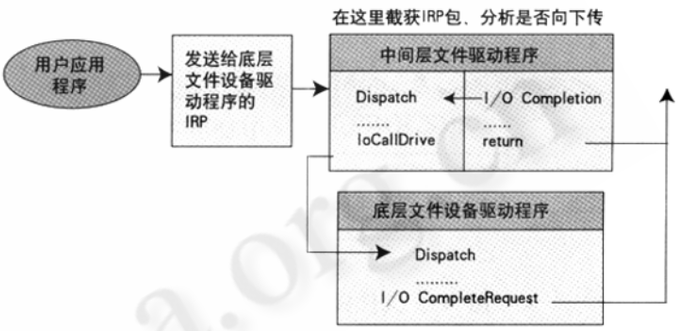


图3 中间层文件设备驱动程序工作原理

当任何的用户应用程序想对文件进行操作时，必须通过将IOCTL控制码传给文件设备驱动程序，文件设备驱动程序就是根据用户应用程序传给它的IOCTL控制码实现对文件的操作。例如，用户应用程序想对文件进行读写操作时就必须将IOCTL控制码IRP_MJ_READ、IRP_MJ_WRITE传给文件设备驱动程序。

中间层文件设备驱动程序是处在应用程序和文件设备驱动程序之间的一个我们编写的驱动程序，图3所示的是其工作原理。该驱动程序截获由应用程序发给文件设备驱动程序的IRP写请求（也就是IOCTL控制码IRP_MJ_WRITE），若文件需要保护，则立即返回，不往下传给文件设备驱动程序处理，因此文件就得到了保护，图4是该模块的流程图。

Windows NT下驱动程序的编程是用DDK(Device

Driver Kit)编程技术。Windows NT 系统磁盘下的每个分区都对应一个设备对象名，设备对象名是为一个设备提供的本地 Windows NT 名称，驱动程序查找每个分区就是查找它的设备对象名，如：分区盘 C：对应 Windows NT 的设备对象名为＼Device＼HardDisk0＼Partition1,分区盘 D：的设备对象名为＼Device＼HardDisk＼Partitin2,......以此类推。在驱动程序入口要为每个分区创建一个设备对象，通过函数 IoCreateDevice()或者 AddDevice()实现，这样中间层驱动程序就拥有文件系统驱动程序的大多数 IRP_MJ_XXX 例程（通过设备对象获得），中间层驱动程序接到 I/O 请求时，在确定自身 IRP 当前堆栈单元参数有效的前提下，设置好 IRP 中下一个驱动程序（文件系统驱动程序）的堆栈单元，然后再调用 IoCallDriver 将请求传递给下层驱动程序处理。

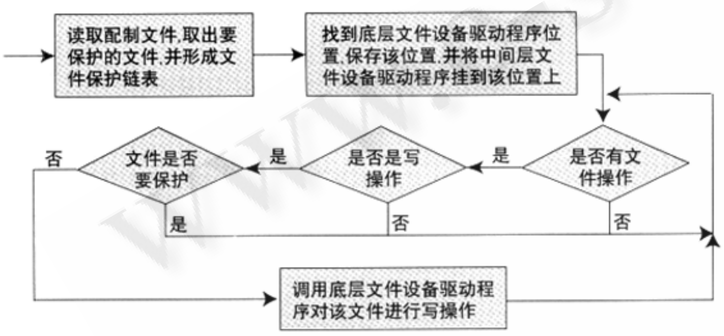


图 4　文件保护模块流程图

3.3 主要代码

```
if(!LoadDeviceDriver( SYS_NAME, driverPath,
&syshandle, &error ) ) {....... };
//SYS_NAME=FILEMON.SYS,driverPath=WinNT＼system32＼driver,syshandle= 可返回找 // 开 FILEMON.SYS 驱动程序的一个句柄，error= 返回操作的错误代码；
BOOL LoadDeviceDriver( const TCHAR * Name, const TCHAR * Path, HANDLE *
lphDevice, PDWORD Error )  // 装入驱动程序（Name 参数）放在 VCM 中，并启动服务
{    SC_HANDLE    schSCManager;
     chSCManager = OpenSCManager( NULL, NULL,
SC_MANAGER_ALL_ACCESS );
     RemoveDriver( schSCManager, Name );// 从 VCM 移去 filesys.sys
InstallDriver( schSCManager, Name, Path );// 装入 filesys.sys 到 VCM 中，CreateService()
StartDriver( schSCManager, Name ); // 启动服务，用函数 StartService( )实现
      okay = OpenDevice( Name, lphDevice );  //open filemon.sys and get handle
    // 通过函数 CreateFile( )找开驱动程序，并得到句柄
      *Error = GetLastError();
      CloseServiceHandle( schSCManager );
}
DeviceIoControl(syshandle,
IOCTL_FILEMON_ZEROSTATS,NULL, 0, NULL, 0,
&nb, NULL )DeviceIoControl(syshandle,
IOCTL_FILEMON_SETFILTER,&FilterDefinition,sizeof
(FILTER),NULL,0, &nb, NULL )
DeviceIoControl(syshandle,
IOCTL_FILEMON_STARTFILTER,NULL, 0, NULL, 0,
&nb, NULL )
DeviceIoControl(syshandle,
IOCTL_FILEMON_SETDRIVES,&CurDriveSet, sizeof
CurDriveSet,&CurDriveSet, sizeof CurDriveSet,&nb, NULL )
DeviceIoControl(syshandle,
IOCTL_FILEMON_GETSTATS,NULL, 0, &Stats, sizeof
Stats,&StatsLen, NULL ) // 每隔 0.5 秒取得它们的值
// 这五个函数往驱动程序 filemon.sys 传送控制码，实现相应功能，分别为清空各项状态、设置过滤文件、开始过滤功能、设置监控盘符、得到监控值
// 另外是一些把监控得到的值显示到列表中的代码。
```

4 进程监控模块

4.1 实现功能

能够监控所有被监控机上正在运行的进程，以及每个正在运行进程所包含的 modules 数和 Dll 文件数，列出进程所包含的 modules 和 Dll 文件的名称，能够指出该进程的运行时间、进程使用内存情况、进程所在的完整路径及进程、线程的句柄，父进程、线程的句柄（即子线程是由哪个父线程创建的），可以终止某个选中的进程或线程，可以控制某个进程是否要在前台运行，通过改变进程的优先级别实现。该模块可以配合性能监控模块执行，当 CPU 利用率达到某个阈值或内存占用率太大，影响整个系统性能时，可以终止占用率大的进程，优化整个系统。

4.2 实现原理

在 Windows NT 中，进程是应用程序的一个运行实例。每个进程由若干线程组成。线程是进程中的一个执行单元。同一个进程中的各个线程对应于一组 CPU 指令、一组 CPU 寄存器以及一个堆栈。进程并不执行代码，它只是代码存放的地址空间。进程地址空间中所存放的代码由线程来执行，这些独立的线程都是由 Window NT Kernel(核心)来调度 CPU 时间。

当 Windows NT 生成进程时，该进程的第一个线程称为主线程（Primary thread),由系统自动产生。然后可以由这个主线程创建其他的线程。在 Windows NT 中，如果进程的任何一个线程还在执行，则这个进程就不会被终止；只有当进程中所有的线程都结束时，它才可以终止。

Windows NT 的进程信息可以通过以下两种方法来获得：性能数据辅助器（Performance Data Helper,简称 PDH）和进程状态辅助器(Process Status Helper)。

性能数据辅助器包括两种方法：直接调用 PDH 接口函数和查询注册表。

PDH 接口函数以查询（queries）和计数器（counters）来运行。一个查询是分组的计数器的集合，这样你就能一次性地读取它们的数据。一个计数器是存储在 Windows NT 注册表的性能数据项，以层次结构的路径来表示，其路径表达式是：\\机器名\性能对象\(父实例/对象实例#实例索引)\计数器。

所有系统信息都存储在注册表的 HKEY_PEFORMANCE_DATA 键下，可以通过调用函数 RegQueryValueEx 来获取。实质上 PDH 接口函数就是通过查询注册表的 HKEY_PEFORMANCE_DATA 键下的信息来获得系统信息的。RegQueryValueEx 返回一系列性能对象集，其中包括进程性能对象，它包括进程运行时间、进程使用内存等信息。

由于通过进程状态辅助器获得进程信息较为困难，至少没有直接通 API 函数获得来得方便，因此 Windows NT 为获得有关进程方面的信息设计了专门的进程状态辅助器函数。利用进程状态辅助器函数可以很容易地取得进程信息，这些函数存放在 PSAPI.DLL 动态链接库中，本文用 PDH 接口函数来实现。

4.3 主要代码

```
h=CreateToolhelp32Snapshot
(TH32CS_SNAPPROCESS,0);// 得到进程快照句柄
  typedef struct _tagPROCS{
    PROCESSENTRY32 ProcessEntry32; // 进程信息结构
    MODULES Modules;   // 进程调用的 DLL
    _tagPROCS *pNextProc;  // 指向下一个进程结构的指针
}PROCS,*PPROCS;
typedef struct _tagMODULES{
    MODULEENTRY32 ModuleEntry32;// DLL 信息结构
    _tagMODULES *pNextModule; // 指向下一个 DLL 结构的指针
}MODULES,*PMODULES;
//PROCS  p,q;
while(Process32Next(h,&ProcessEntry32)) // 枚举所有进程，放在链表变量 p 中
    {
        p->pNextProc=new PROCS;
        p=p->pNextProc;
        if (!p)    return 0;
        ZeroMemory(p,sizeof(PROCS));
        p->ProcessEntry32=ProcessEntry32;
        GetModuleInfo(p);
    }
HANDLE hProcess=OpenProcess
(PROCESS_TERMINATE,FALSE,
current_select_process);
 if(hProcess!=NULL)
{
    TerminateProcess(hProcess,0); // 终止选定的进程
    CloseHandle(hProcess);
    m_listctrl2.DeleteItem (select_row);
}
```

5 结束语

由于 Windows NT 下的驱动程序编程与 Windows 98 不一样（两者驱动程序的核心模式不一样），Windows NT 是采用分层驱动，Windows 98 没有这种结构，其驱动程序主要采用 vxd 编程，而 Windows NT 下主要采用 WDM(Win32 Driver Model)编程，所以本文所述的主机入侵检测系统只能在 NT/W2000 下运行。■