基于 NDIS-HOOK 与 SPI 的个人防火墙研究与设计

高泽胜，陶宏才

(西南交通大学 计算机与通信工程学院，四川 成都 610031)

摘　要: 介绍了 NDIS 系统结构以及在 Windows 2000/NT 下 NDIS-HOOK 的原理，并给出了一个基于 NDIS-HOOK 技术的个人防火墙驱动程序——Packet.sys。利用该驱动程序与 SPI 技术相结合，可方便地实现基于 Windows 的个人防火墙。

关键词: NDIS; HOOK; TDI; SPI; 中间层驱动; 协议; 防火墙

中图法分类号: TP393.08　　**文献标识码**: A　　**文章编号**: 1001-3695(2004)11-0279-03

Research and Implementation of Personal Firewall Based on NDIS-HOOK

GAO Ze-sheng, TAO Hong-cai

(School of Computer & Communication Engineering, Southwest Jiaotong University, Chengdu Sichuan 610031, China)

Abstract: Introduces the system structure of NDIS and principle of NDIS-HOOK in Windows 2000/NT, gives a drivers-Packet.sys, which based on NDIS-HOOK. We can combine this drivers and SPI to implement personal firewall easily in Windows system.

Key words: NDIS; HOOK; TDI; SPI; Intermediate Drivers; Protocol; Firewall

从应用角度看，防火墙基本上可以分为两种: 网络级的防火墙和个人防火墙。由于 Windows 操作系统是目前使用最为广泛的 PC 操作系统，因此在 Windows 操作系统下开发的个人防火墙产品数不胜数。然而，所有基于 Windows 操作系统的个人防火墙，其核心技术一般是基于 Windows 操作系统下网络数据包的拦截技术。但是这些技术或多或少地存在着某些缺陷，如基于 SPI(Service Provider Interface)技术的只能捕获应用层的数据，基于 TDI(Transport Drivers Interface)技术的不能接收更改底层的数据包(如 ICMP)，基于中间层驱动程序技术的防火墙安装却又十分麻烦，而且很容易造成网络瘫痪。而本文讨论的 NDIS-HOOK 技术则很好地避免了这些问题，将其与 SPI 技术相结合可以方便地实现基于核心层与应用层的个人防火墙。

1　NDIS 系统结构

NDIS(Network Driver Interface Specification)是 Microsoft 和 3Com 公司开发的网络驱动程序接口规范。它为 Windows 下网络驱序程序的开发带来许多方便，编写符合 NDIS 规范的驱动程序时，只要调用 NDIS 函数，而不用考虑操作系统的内核以及与其他驱动程序的接口问题，为操作系统对不同网络的支持提供了方便。

Windows 使用 NDIS 函数库实现 NDIS 接口，所有的网络通信最终必须通过 NDIS 完成。NDIS 负责上下层驱动程序间服务原语与驱动程序入口之间的转换，分派消息通知，保证符合 NDIS 的驱动程序无须知道其他驱动程序的入口就可以与之通信。NDIS 横跨传输层、网络层和数据链路层。NDIS 的结构如图 1 所示。

收稿日期: 2003-08-10; 修返日期: 2003-12-03

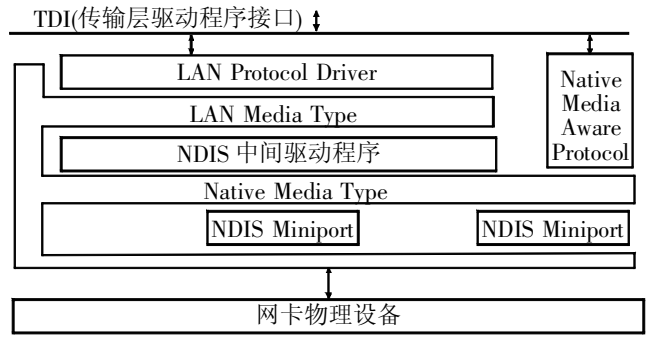

图 1　NDIS 拓扑结构图

由图 1 可看出，微软提供了以下四种标准接口编程方式:

(1) 小端口驱动程序(Miniport Drivers)。它可通过 NDIS 接口来完成对网卡的操作，同时开放 Miniport 接口供上层驱动程序调用。可用 Miniport 和 Protocol 驱动程序来实现网卡驱动。

(2) 中间驱动程序(Intermediate Drivers)。它位于 Miniport 驱动程序与 Protocol 驱动程序之间，中间驱动程序同时具有 Miniport 和 Protocol 两种驱动程序接口。中间驱动程序在自己的上、下两端分别开放出一个 Miniport 接口和一个 Protocol 接口，上面的 Miniport 接口同上层驱动程序的 Protocol 接口进行对接; 下面的 Protocol 接口同底层驱动程序的 Miniport 接口进行对接，这样中间驱动程序就在两个驱动之间插入一层中间层。

利用 NDIS 中间驱动程序，可以在网卡驱动程序与传输驱动程序之间插入一层自己的自定义驱动程序(类似 TcpIp.sys)，从而可以截获网络封包，并重新进行封包、加密、网络地址转换、过滤、认证等操作。由于 NDIS 中间驱动程序位于网卡与传输驱动程序之间，所以它可截获较为底层的封包，从而可以完成更为低级的操作。但是也正因为如此，它有一个弱点，就是编程接口复杂，而且编写出来的软件自动化安装太困难，很容易造成整个网络瘫痪，所以目前个人防火墙的产品还很少用到这种技术。NDIS 中间驱动程序安装前、后的对比结

构示意图如图 2 所示。

(3) 协议驱动程序(Protocol Drivers) 。开放 Protocol 接口供底层驱动程序调用,实现 Protocol 接口与 Miniport 接口的对接。协议驱动程序执行具体的网络协议,如 IPX/SPX, TCP/IP 等,协议驱动程序为传输层提供 Miniport 接口,接收来自网卡或中间驱动程序的信息。

(4) TDI 传输驱动程序。通常用于封包过滤,也是防火墙和 VPN(Virtual Private Network) 软件常用的技术。但是它有一个缺陷: TDI Filter Driver 属于 Upper Driver,位于 TcpIp. sys 之上,这就意味着由 TcpIp. sys 接收并直接处理的数据包,不会传送到上层,从而无法过滤某些接收的数据包,如 ICMP 包。ICMP 的应答包,直接由 TcpIp. sys 生成并回应,而上面的过滤驱动程序则全然不知。传输层过滤驱动程序 TDI Filter,常见的 TCP Filter Driver 即属此类。

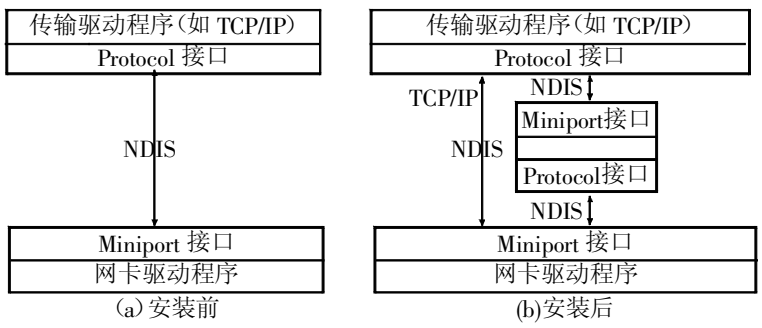

图 2 NDIS 中间驱动程序安装前、后的对比结构示意图

2 NDIS-HOOK

2.1 NDIS-HOOK 原理

NDIS-HOOK 克服了上面各种驱动程序的缺陷。NDIS-HOOK 的工作原理是直接替换 NDIS 函数库中的函数地址,这样,向 NDIS 的请求就会先经过自定义函数处理,处理完后再转发给系统函数(图 3) 。

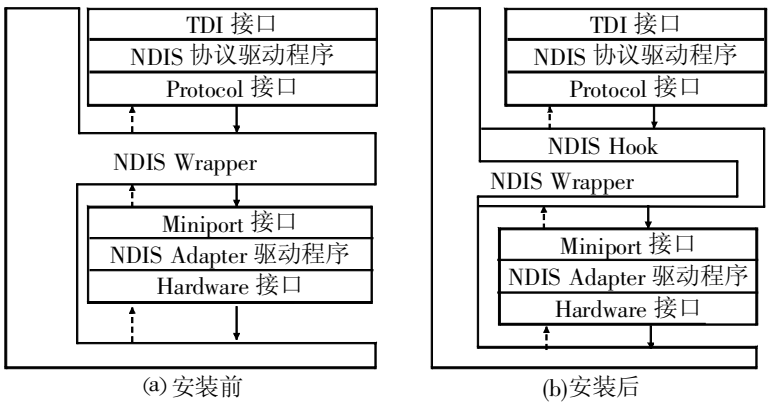

图 3 NDIS-HOOK 安装前、后结构示意图

NDIS-HOOK 在 Windows 2000/ NT 下的实现思路大致如下: 通过修改系统目录下的 NDIS. SYS 的 Export Table,在 Windows 2000/ NT 下,可执行文件(包括 DLL, SYS) 都是遵从 PE(Portable Executable) 格式的。所有向其他操作系统组件提供接口的驱动程序,都有 Export Table,因此只要修改 NDIS. SYS 的 Export Table,就可以实现对关键 NDIS API 的挂接。由于协议驱动程序在系统启动时,会调用 NdisRegisterProtocol,来向系统进行协议注册,因此这种方法关键在于修改 NDIS. SYS 所提供的 NdisRegisterProtocol/ NdisDeRegisterProtocol/ NdisOpenAdapter/ NdisCloseAdapter/ NdisSend 函数的起始地址。在用户态模式下,要修改 PE 文件格式可以用一些 API 来实现,而 NDIS. SYS 位于系统的核心内存区,因此要修改 NDIS. SYS 就不得不通过写驱动程序来实现(即我们自定义的驱动函数 Packet. sys) 。使用这种方法,还要注意驱动程序的加载次序, Hook Driver 必须在 NDIS. SYS 被加载之后,而在协议驱动程序,如 TcpIp. sys, 被加载之前。

2.2 NDIS-HOOK 在个人防火墙中的实现过程

该实现过程采用 Windows DDK( ) 开发,所有的工具包有 VC ++ 6. 0, MSDN Library January 2002 以及 Windows 2000 DDK。下面着重介绍自定义 NDIS-HOOK 驱动函数 Packet. sys 的实现过程(图 4) 。

图 4 Packet. sys 的加载过程

各函数的部分源代码如下:

```
/* 驱动程序入口函数,这里用来完成缓冲区初始化、HOOK 和设备的创建* /
NTSTATUS DriverEntry( IN PDRIVER_OBJECT DriverObject,
IN PUNICODE_STRING RegistryPath)
{
//申请、初始化封包缓冲区
if(! InitBuffer( ) )
return NDIS_STATUS_FAILURE;
//Hook NDIS 函数
/* GetNdisModuleAddress( ) 得到 Ndis. sys 在内存中的基地址,存放到变量 m_NdisBaseAddress 中* /
if( GetNdisModuleAddress( ) && m_NdisBaseAddress ! = NULL)
{
/* 根据 m_NdisBaseAddress 变量 HOOK NDIS. SYS 中的 NdisSend 函数* /
if( HookFunction( m_NdisBaseAddress," NdisSend" , O_NdisSend,
( ULONG* ) &m_pNdisSend) = = NULL
)
dprintf( ( " Hook NdisSend Failure \n" ) ) ;
else
dprintf( ( " Hook NdisSend Success \n" ) ) ;
/* 根据 m_NdisBaseAddress 变量 HOOK NDIS. SYS 中的 NdisRegisterProtocol 函数* /
if( HookFunction ( m_NdisBaseAddress, " NdisRegisterProtocol" , O_NdisRegisterProtocol, ( ULONG* ) &m_pNdisRegisterProtocol) = = NULL
)
dprintf( ( " Hook NdisRegisterProtocol Failure \n" ) ) ;
else
dprintf( ( " Hook NdisRegisterProtocol Success \n" ) ) ;
/* 根据 m_NdisBaseAddress 变量 HOOK NDIS. SYS 中的 NdisDeregisterProtocol 函数* /
if( HookFunction( m_NdisBaseAddress, " NdisDeregisterProtocol" , O_NdisDeregisterProtocol, ( ULONG* ) &m_pNdisDeregisterProtocol) = = NULL)
```

```
dprintf( ( "Hook NdisDeregisterProtocol Failure \n" ) ) ;
else
dprintf( ( "Hook NdisDeregisterProtocol Success \n" ) ) ;
/* 根据 m_NdisBaseAddress 变量 HOOK NDIS. SYS 中的 NdisOpenAdapter 函数 * /
if( HookFunction( m_NdisBaseAddress, "NdisOpenAdapter", O_NdisOpenAdapter, ( ULONG* ) &m_pNdisOpenAdapter)  = = NULL
)
dprintf( ( "Hook NdisOpenAdapter Failure \n" ) ) ;
else
dprintf( ( "Hook NdisOpenAdapter Success \n" ) ) ;
}
…
return( 0) ;
}
```

当 Packet. sys 加载完后, 发送网络数据包, 就先调用自定义的函数 O_NdisSend。O_NdisSend 先对要发送的网络数据包进行检查, 然后才调用系统 NdisSend 函数完成转发。接收网络数据包, 先调用自定义的函数 O_Receive 进行处理, 然后才调用系统 ProtocolReceive 函数完成转发。由此, 可以完成对 TCP, UDP, ICMP, IPX 和 FTP 等协议的认证了。

2. 3　基于 NDIS- HOOK 技术的 Windows 个人防火墙的特点

利用 NDIS- HOOK 技术实现的 Windows 个人防火墙具有以下特点: ①编程方便、接口简单、思路明确、性能稳定; ②更灵活, 可以仅仅截获自己所需要的信息, 不需要冗余的代码; ③功能强大, 可以截获所有 NDIS 和 TDI 函数完成的功能。比标准方式功能强大许多; ④安全性高, 这样截获封包较为底层, 不容易被穿透; ⑤安装简单。

3　SPI 技术

Winsock 2 SPI 技术其实是一个非常常见的技术, 它工作在 API 之下 Driver 之上, 属于应用层的范畴。利用这项技术可以截获所有的基于 Socket 的网络通信。比如: IE, OUTLOOK 等常见的应用程序都是使用 Socket 进行通信。它以 DLL 的形式存在, 因为封包还没有按照底层协议进行切片, 所以比较完整, 很容易作内容过滤, 因此, 在应用层用它来实现过滤是非常理想的。

Windows Socket SPI 提供三种协议: 分层协议, 基础协议和协议链。分层协议是在基础协议的上层, 依靠底层协议实现更高级的通信服务; 基础协议是能够独立, 安全地与远程端点实现数据通信的协议, 它是相对于分层协议而言的。协议链是将一系列的基础协议和分层协议按特点的顺序连接在一起的链状结构。SPI 结构示意图如图 5 所示。

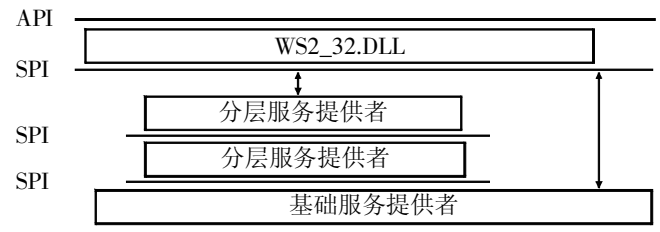

图 5　SPI 结构示意图

WS2_32. DLL 数据传输部分的主要功能是在服务提供者和应用程序之间提供流量管理的功能。每个应用程序通过 WS2_32. DLL 和相应的服务提供者进行严格的数据交换。WS2_32. DLL 根据应用程序在创建套接字时所提供的参数来选择特定的服务提供者, 然后把相应程序的实现过程转发由所选创建套接字的服务提供者来管理。也就是说, WS2_32. DLL 只是一个中间过程, 而应用程序只是一个接口, 数据通信的实现却是由服务提供者来写成的。

4　结束语

本文讨论了在 Windows 2000/NT 下如何利用 NDIS- HOOK 技术与 SPI 技术实现个人防火墙, 深入讨论了它的原理和实现过程。在实际的实现过程中, NDIS- HOOK 用于核心层而 SPI 用于应用层, 这样就可以相互利用两者的长处, 互补两者的短处, 使个人防火墙的设计更高效、合理。

参考文献:

[1]　朱雁辉. Windows 防火墙与网络封包截获技术[M] . 北京: 电子工业出版社, 2002.

[2]　icrosoft Corporation. Microsoft Developer Network MSDN Library for Windows DDK[Z] . Microsoft Corporation, 2000.

[3]　陈弘, 刘彦. NDIS 网络驱动程序设计研究和 IP 隧道驱动程序实现[J] . 电子技术, 2000, (2) : 7-11.

[4]　美] Terry Willian Ogletree. 防火墙原理与实施[M] . 李之棠, 李伟明, 陈琳, 等. 北京: 电子工业出版社, 2001.

作者简介:

高泽胜(1976-) , 男, 安徽人, 硕士研究生, 研究方向为网络安全; 陶宏才(1964-) , 男, 湖北人, 副教授, 硕士生导师, 研究方向为计算机网络与信息系统、数据库、数据仓库与数据挖掘。

(上接第 272 页) 企业的电子商务网站, 主要实现下列功能: 公司最新动态、公司介绍、产品展示、客户和代理商管理、分支机构和职员管理、异地库管理、办公系统等; 根据权限的不同, 提供各种查询功能, 如库存查询、价格查询、供应商查询、计划查询、合同查询等; 基于权限的供应商部分基本信息维护功能; 动态在 Web 上发布招标信息, 供应商可在网上日填写投标信息; 利用基于权限的电子看板, 向定点供应商传达采购计划、修正订货计划和协商合同明细等, 实现供应链管理的同步化运作和 JIT 采购。

6　总结

本文结合企业信息管理的实际需要, 对供应链管理系统进行了探讨, 并对在供应链管理环境下的采购管理中的供应商评价进行了研究, 在此基础上给出了一个结合中国国情, 融合供应链和现代 ERP 管理思想的采购管理系统的原型系统的总体设计方案。

参考文献:

[1]　马士华, 林勇, 等. 供应链管理[M] . 北京: 机械工业出版社, 2000.

[2]　柴跃进, 等. 敏捷供需链管理[M] . 北京: 清华大学出版社, 2000.

[3]　谢澍, 张洪伟, 魏筱毛, 等. ERP 与供应链结合的采购管理研究[J] . 计算机应用研究, 2002, 19(9) : 91-93.

作者简介:

李璟(1971-) , 女, 讲师, 在读硕士研究生, 主要研究方向为企业资源计划以及供应链的理论研究等; 李玉忱(1946-) , 女, 教授, 主要研究方向为数据挖掘、数据库系统应用、电子商务; 任磊(1979-) , 男, 在读硕士研究生, 主要研究方向为数据库系统应用、电子商务技术及应用。