目 录

第 1 章 引 言 - 3 -
第 2 章 防 火 墙 - 4 -
2.1 防火墙理论 - 4 -
2.1.1 防火墙原理简介 - 4 -
2.1.2 防火墙的分类 - 4 -
2.2 Windows 个人防火墙 - 6 -
第 3 章 Windows 网络驱动 - 7 -
3.1 OSI 网络参考模型 - 7 -
3.1.1 OSI 七层网络体系结构简介 - 7 -
3.1.2 TCP/IP 协议的架构 - 8 -
3.2 Windows 网络模型 - 9 -
3.2.1 网络七层协议在 Windows 中的实现 - 9 -
3.2.2 TCP/IP 协议在 Windows 中的实现 - 10 -
3.3 Windows 网络驱动程序 - 11 -
3.3.1 Windows 2000 驱动程序 - 11 -
3.3.2 Windows 网络驱动程序 - 12 -
3.4 个人防火墙实现技术 - 13 -
3.4.1 应用层包拦截技术 - 14 -
3.4.2 核心层包拦截技术 - 14 -
3.5 TDI 过滤驱动程序开发 - 15 -
3.5.1 驱动程序开发环境 - 15 -
3.5.2 TDI 过滤驱动程序的特性 - 16 -
3.5.3 TDI 过滤驱动程序结构 - 17 -
3.5.4 自旋锁和同步事件 - 19 -
第 4 章 PFW 个人防火墙的实现 - 22 -
4.1 系统设计 - 22 -
4.1.1 功能分析 - 22 -
4.1.2 系统结构 - 22 -
4.1.3 安全规则结构 - 23 -
4.2 过滤驱动程序 - 25 -
4.2.1 结构和功能 - 25 -
4.2.2 IRP 处理和回调事件 - 26 -
4.2.3 对象表管理 - 27 -
4.2.4 数据包过滤 - 28 -
4.2.5 规则管理 - 30 -
4.2.6 通知管理 - 30 -

Evaluation Copy
PDFCreator Plus 4.0

4.2.7 pfw_Control 设备与驱动控制 - 32 -
4.3 驱动控制接口 - 32 -
4.3.1 结构和功能 - 32 -
4.3.2 过滤规则维护接口 - 33 -
4.3.3 应用程序访问规则维护接口 - 33 -
4.3.4 日志与通知管理 - 34 -
4.3.5 驱动管理接口 - 35 -
4.4 用户界面 - 35 -
4.4.1 结构和功能 - 35 -
4.4.2 过滤规则设置 - 36 -
4.4.3 程序访问规则设置 - 36 -
4.4.4 数据包监视界面 - 37 -
4.4.5 系统设置 - 37 -
4.5 小结 - 37 -
总 结 - 39 -
附录一：驱动程序（pfw.sys）源代码 - 42 -
附录二：接口程序（pfw.dll）源代码 - 96 -
附录三：主界面程序（pfw.exe）源代码 - 113 -
附录四：过滤规则 - 155 -

Evaluation Copy
PDF Creator Plus 4.0

第 1 章 引 言

随着网络的逐步普及，网络安全的问题已经日益突出地摆在各类用户的面前。同其它任何社会一样，Internet 也受到某些无聊之人的困扰，这些人喜爱在网上做这类的事,像在现实中向其他人的墙上喷染涂鸦、将他人的邮箱推倒或者坐在大街上按汽车喇叭一样。网络安全已成为 Internet 上事实上的焦点，它关系着 Internet 的进一步发展和普及，甚至关系着 Internet 的生存。近年来，无论在发达国家，还是在发展中国家（包括我国），黑客活动越来越猖狂，他们无孔不入，对社会造成了严重的危害。目前在互联网上大约有将近 20%以上的用户曾经遭受过黑客的困扰。与此同时，更让人不安的是，互联网上病毒和黑客的联姻、不断增加的黑客网站，使学习黑客技术、获得黑客攻击工具变的轻而易举。这样，使原本就十分脆弱的互联网越发显得不安全。

于是我们经常为黑客的危害而惊叹不已，然后我们应该提醒自己，惶恐是无济于事的，想方设法迎接挑战才是唯一的出路。俗话说：人吃五谷杂粮，怎能不生病？医学的发展为我们创造出了许多防病治病的好办法，计算机技术的发展同样也为个人电脑预防黑客提供了许多有效的措施。所以，我们可以使用相适应的防黑工具。

防火墙就是其中一种简洁有效的工具。它是一个或一组系统，它在网络之间执行访问控制策略，实现它的实防方式各不相同，但是在原则上，防火墙可以被认为是这样一种机制：拦阻不安全的传输流通行，允许安全的传输流通过。

它可以监控进出网络的通信信息，仅让安全的、核准了的信息进入；它可以限制他人进入内部网络，过滤掉不安全服务和非法用户；它可以封锁特洛伊木马，防止机密数据的外泄；它可以限定用户访问特殊站点，禁止用户对某些内容不健康站点的访问；它还可以为监视 Internet 安全提供方便。

总之，防火墙是防御网络入侵者的最有效机制之一。

Evaluation Copy
PDFCreator Plus 4.0

第 2 章 防 火 墙

2.1 防火墙理论

2.1.1 防火墙原理简介

防火墙就是一个位于计算机和它所连接的网络之间的软件。该计算机流入流出的所有网络通信均要经过此防火墙。

所有的 Internet 通信都是通过独立数据包的交换来完成的，而每个数据包都是由源主机向目标主机传输的，所以数据包是 Internet 上信息传输的基本单位。虽然我们常说电脑之间的“连接”，但这“连接”实际上是由被“连接”的两台电脑之间传送的独立数据包组成的。为了到达目的地，不论两台电脑是隔着几米远还是在不同的大洲上，每个数据包都必须包含一个目标地址及端口号和源地址及端口号，以便接收者知道是谁发出了这个包。也就是说，每一个在 Internet 上传送的数据包，都必须含有源地址和目标地址。那么基于源主机 IP 地址及端口号和目标主机 IP 地址及端口号的一些组合，防火墙就可以决定该拦截这个数据包还是将其放行。

防火墙时刻监视着进出主机的数据包，这样能够过滤掉一些攻击，以免其在目标计算机上被执行。防火墙还可以关闭不使用的端口。而且它还能禁止特定端口的流出通信，封锁特洛伊木马。最后，它可以禁止来自特殊站点的访问，从而防止来自不明入侵者的所有通信。

防火墙具有很好的保护作用。入侵者必须首先穿越防火墙的安全防线，才能接触目标计算机。

Evaluation Copy
PDF Creator Plus 4.0

2.1.2 防火墙的分类

防火墙采用的技术和标准可谓五花八门。这些防火墙的形式多种多样：有的取代系统上已经装备的 TCP/IP 协议栈；有的在已有的协议栈上建立自己的软件模块；有的干脆就是独立的一套操作系统。还有一些应用型的防火墙只对特定类

型的网络连接提供保护（比如 SMTP 或者 HTTP 协议等）。还有一些是基于硬件的防火墙产品，其实它应该归入安全路由器一类。

防火墙形式多样，划分标准也比较杂。主要分类如下：

1. 从软、硬件形式上分为：软件防火墙和硬件防火墙以及芯片级防火墙。

软件防火墙运行于特定的计算机上，它需要客户预先安装好的计算机操作系统的支持，一般来说这台计算机就是整个网络的网关。硬件防火墙基于 PC 架构，并在这些 PC 架构计算机上运行一些经过裁剪和简化的操作系统实现防火墙功能。芯片级防火墙基于专门的硬件平台，没有操作系统。速度更快，处理能力更强，性能更高，漏洞少，不过价格相对比较高昂。

2. 从防火墙使用的技术分为：“包过滤型”和“应用代理型”两大类。

包过滤型防火墙工作在 OSI 网络参考模型的网络层和传输层，它根据数据包头源地址，目的地址、端口号和协议类型等标志确定是否允许通过。只有满足过滤条件的数据包才被转发到相应的目的地，其余数据包则被从数据流中丢弃。应用代理型防火墙是工作在 OSI 的最高层，即应用层。其特点是完全“阻隔”了网络通信流，通过对每种应用服务编制专门的代理程序，实现监视和控制应用层通信流的作用。

3. 从防火墙结构分为：单一主机防火墙、路由器集成式防火墙和分布式防火墙三种。

单一主机防火墙是最为传统的防火墙，这种防火墙其实与一台计算机结构差不多，独立于其它网络设备，它位于网络边界，其中的硬盘就是用来存储防火墙所用的基本程序。路由器集成式防火墙是一些中、高档路由器集成防火墙功能的产物，类似单一主机防火墙。分布式防火墙再也不是只位于网络边界，而是渗透于网络的每一台主机，对整个内部网络的主机实施保护。

4. 从防火墙的应用部署位置分为：边界防火墙、个人防火墙和混合防火墙三大类。

边界防火墙是最为传统的那种，它们于内、外部网络的边界，所起的作用是对内、外部网络实施隔离，保护边界内部网络。个人防火墙安装于单台主机中，防护的也只是单台主机，通常为软件防火墙。价格最便宜，非常适用于广大的个人用户。混合式防火墙可以说就是“分布式防火墙”或者“嵌入式防火墙”，它

Evaluation Copy PDFCreator Plus 4.0

是一整套防火墙系统，由若干个软、硬件组件组成，分布于内、外部网络边界和内部各主机之间，既对内、外部网络之间通信进行过滤，又对网络内部各主机间的通信进行过滤。它属于最新的防火墙技术之一，性能最好，价格也最贵。

5. 从防火墙性能分为：百兆级防火墙和千兆级防火墙两类。

这主要是指防火墙的通道带宽，或者说是吞吐率。当然通道带宽越宽，性能越高，这样的防火墙因包过滤或应用代理所产生的延时也越小，对整个网络通信性能的影响也就越小。

2.2 Windows 个人防火墙

如前所述，网络安全形势日趋严峻，而网络用户中的大部分是个人用户。他们中的大多数是不具备专业的网络安全知识的普通用户，他们是最易受到黑客的攻击群体，选择一款合适的防火墙产品，对他们甚为重要。最适合他们的防火墙显然是包过滤软件型个人防火墙。

包过滤软件型个人防火墙不需要额外的硬件投资、价格便宜、价格快、功能多、性能较好、安装方便、操作简单、实用性强，完全能满足个人用户对性能、价格的要求。

考虑到 Windows 操作系统作为使用最为广泛的 PC 操作系统，所以基于 Windows 操作系统的个人防火墙是这些用户的首选。所有基于 Windows 操作系统的个人防火墙核心技术点在于 Windows 操作系统下网络数据包过滤技术。

我们知道，进行 Internet 网络通讯的基本单位是数据包，每个数据包都有数据包的源地址、端口和目的地址、端口，防火墙软件根据地址和端口信息拦截或放行此数据包，实现过滤功能。要过滤 Windows 下的网络数据包可以在两个层面进行：用户态（user-mode）和内核态（kernel-mode）。在用户态下进行数据包拦截最致命的缺点就是只能在 Winsock 层次上进行，而对于网络协议族中底层协议的数据包无法进行处理。对于一些木马和病毒来说很容易避开这个层次的防火墙。所以大多数的基于 Windows 操作系统的个人防火墙都是利用网络驱动程序实现传输层或网络层的数据包拦截。

后续章节将探讨利用 Windows 2000 网络驱动程序来实现基于 Windows 2000 操作系统的个人防火墙。

Evaluation Copy
PDFCreator Plus 4.0

第 3 章 Windows 网络驱动

3.1 OSI 网络参考模型

3.1.1 OSI 七层网络体系结构简介

从 1977 年到 1983 年，网络专家制定了一个叫做开放式系统互连（Open Systems Interconnection）参考模型的网络设计模型，简称 OSI 参考模型。

OSI 模型代表一个理想化的网络，它用“层”将一个网络分成定义好的功能化模块，网络设计人员按照此模型关于各层的描述建立实际的网络。如图：

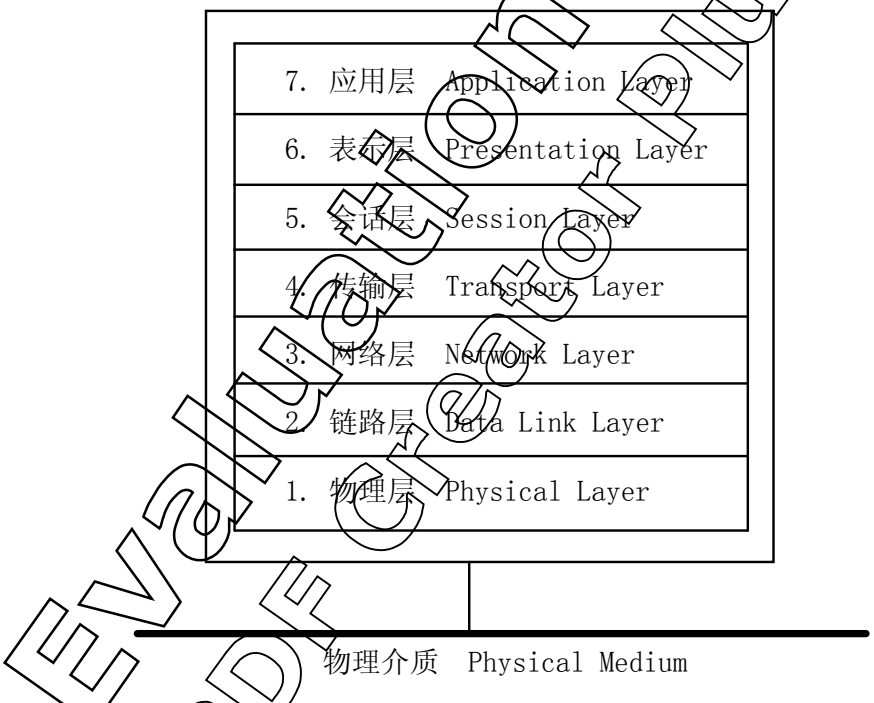


图 3.1 OSI 七层模型

在分层化网络中，每层为其相邻层提供特殊的服务，实现专门的功能。另外每层对其上面各层隐藏了低层的实现细节，各层只关心与网络中下一层的接口。

OSI 参考模型为网络开发提供了一个标准，设计人员也可以很灵活地根据网络的实际需要改变层的数目、各层的名字以及层的功能等，因此，一个按照 OSI 模型建立的网络可能和按照同样模型建立的其他网络有很大不同。

Evaluation Copy
PDFCreator Plus

3.1.2 TCP/IP 协议的架构

TCP/IP 产生于 Internet 的研究和应用的实践中，它可以用 OSI 参考模型描述，但在细节上它于 OSI 参考模型存在着一些差异。

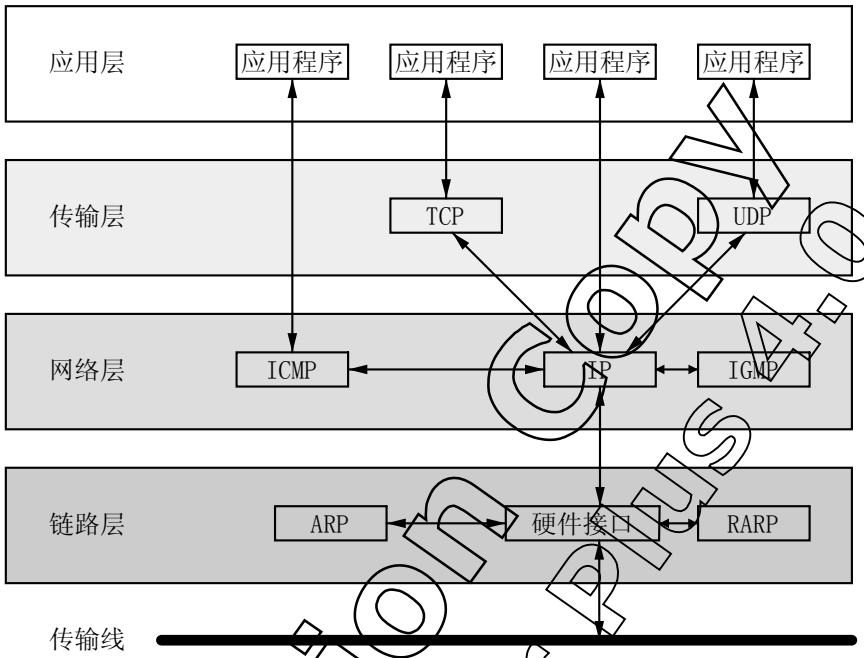


图 3.2 TCP/IP 协议架构

如图所示，TCP/IP 协议族模型由应用层、传输层、网络层和链路层四个层次所组成。

应用层负责向用户提供一组常用的应用协议，例如 FTP、SMTP、POP3、TELNET、HTTP 和 TFTP 等。用户可以使用这些应用协议，也可以直接在传输层之上建立自己专用的应用程序。

传输层为应用程序间端到端的通信提供可能。TCP（Transmission Control Protocol，传输控制协议）是面向连接的，提供可靠的数据传输，对传输的数据进行正确性检查。UDP（User Datagram Protocol, 用户数据报协议）提供主机间可靠的地址路由，但是对网络上数据包的传送是无连接的，不保证数据要以安全地到达目的地。

网络层是基于 TCP/IP 协议族的任意网络的核心。网络层包括 IP（Internet Protocol, 网际协议）、ICMP（Internet Control Messages Protocol, 网际控制报文协议）、IGMP（Internet Group Manager Protocol，网际组管理协议）

链路层是 TCP/IP 协议族分层模型的最底层，它负责接收网络层数据报并通

Evaluation Copy PDFCreator Plus4.0

过传输线发送，或者从传输线上接收数据帧，从中取出数据报，交给网络层。

3.2 Windows 网络模型

3.2.1 网络七层协议在 Windows 中的实现

Windows 操作系统没有严格地划分出这七层协议，它们之间还存在一些交叉。我们就不能将 OSI 模型的每一层一一对应到 Windows 操作系统内，不过，可以用一个不严谨的例子将 OSI 模型映射到 Windows 操作系统。如图：

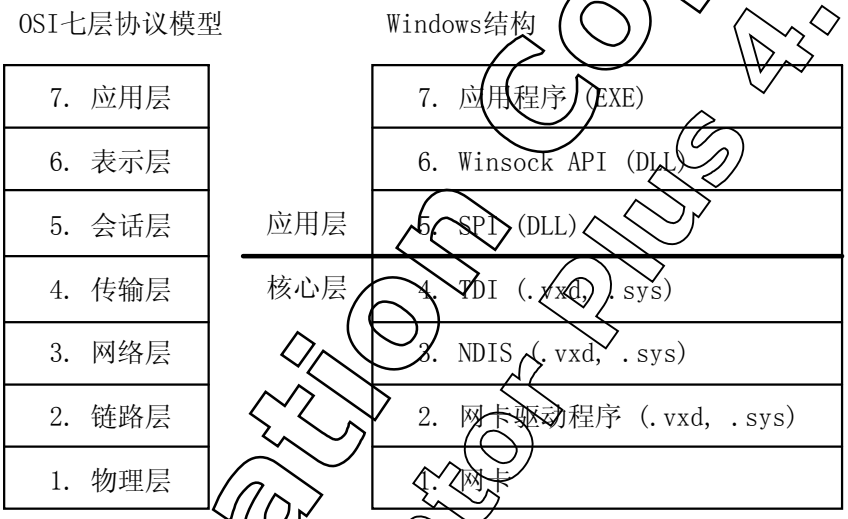


图 3.3 OSI 模型与 Windows 网络模型的映射

物理层很好理解，看得见，摸得着，它就是网卡。数据链路层是网卡驱动程序。

网络层是 NDIS，NDIS 是 Windows 的网络驱动接口规范，从网卡驱动程序到协议驱动程序都要利用 NDIS 这个规范来进行操作。NDIS 为整个网络驱动提供接口，实际上它横跨数据链路层、网络层和传输层这 3 层。

传输层是 TDI，TDI 是传输驱动接口，它不仅是一个简单的传送带，它还要对信息进行检索、分类并重新组织。TCP 协议的数据包处理就是在这一层进行的。

会话层是 SPI，SPI 是服务提供者接口，上面介绍的各层都处于核心层，其程序都是驱动程序，扩展名为.vxd 或.sys。SPI 属于应用层范畴，它的程序为动态链接库(DLL)形式。SPI 负责连接核心层驱动程序和高层应用程序，SPI 的上级为 API。

表示层为 API，它为应用程序提供接口。API 负责 SPI 与应用程序之间的数

Evaluation Copy PDFCreator Plus 4.0

据传输。

应用层就是应用程序，最常见的是 EXE 文件。它是人机界面，负责将数据传输结果显示给用户，并将用户下达的命令传送到下一层。

3.2.2 TCP/IP 协议在 Windows 中的实现

如前述，TCP/IP 和 OSI 有一映射关系，所以可以将 TCP/IP 通过 OSI 映射到 Windows 操作系统中去，同样这些层映射到应用程序并不那么界限分明，下图又是一个不严谨的映射：

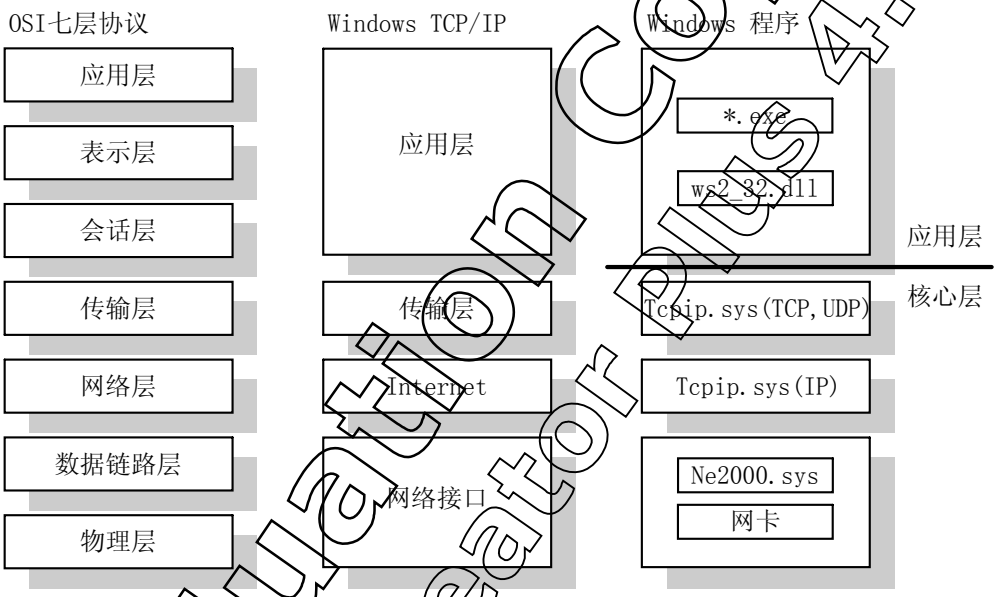


图 3.4 Windows TCP/IP 协议

Windows 程序中使用到的 Tcpip.sys 是 Windows TCP/IP 协议的核心。Tcpip.sys 至少提供了 3 种服务，一是 TCP 协议的实现；二是 UDP 协议的实现；三是 IP 协议的实现。通过对 TCP/IP 协议结构的了解，可以知道 TCP、UDP 工作在传输层，而 IP 协议工作在 Internet 层，对应到 OSI 就是网络层。所以 Tcpip.sys 不仅仅工作在传输层，它也工作在网络层。

数据链路层由网卡（NIC）和网卡驱动程序实现。上图中是以 NE2000 网卡为例，网卡驱动程序 ne2000.sys 和 NE2000 网卡组成网络接口，映射数据链路层和物理层。

应用层是 Windows 系统平台中运行的程序或一些动态链接库（DLL）。Windows 平台下的绝大部分应用程序使用 winsock API（Windows 套接字）编程，

Evaluation Copy
PDFCreator Plus 4.0

它们直接或间接使用 ws2_32.dll 提供的接口。ws2_32.dll 大致对应七层网络模型的表示层。

3.3 Windows 网络驱动程序

前面我们看到 TCP/IP 协议族的实现程序 Tcpip.sys 位于 Windows 操作系统的核心层，它是驱动程序。网卡驱动 Ne2000.sys 也是驱动程序。它们都是 Windows 2000 网络驱动程序。

3.3.1 Windows 2000 驱动程序

为实现 Windows 2000 的稳定性、可靠性和可移植性的设计目标，Windows 2000 的设计者选择了软件的分层结构。用户程序在操作系统的用户层中运行，操作系统内核程序在核心层运行。如图：

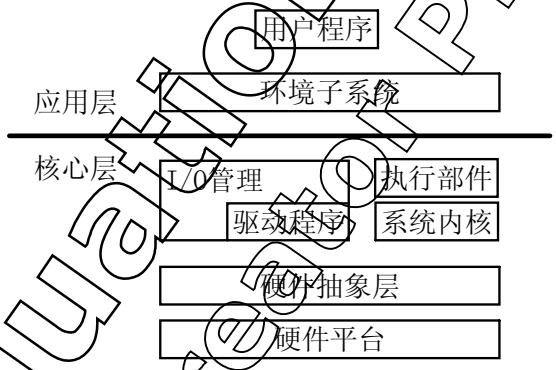


图 3.3 Windows 2000 分层结构

应用层中，代码被严格约束在对系统没有损害的范围内。例如：通过虚拟内存映像，一个程序不能访问其它程序的内存区（两个程序共同定义的公用内存区除外）。硬件 I/O 指令不能被执行。所有的 CPU 指令（如 CPU 中断）不能被执行（在特定的特权级下）。所有的这些被阻止的操作如果想运行，它们必须通过陷阱门来请求操作系统内核。

操作系统内核程序运行在核心层，它可以拥有 CPU 的完全控制权，执行所有有效的 CPU 指令；可执行硬件 I/O 操作，可以完成非常底层的操作；它可以并可以突破 Windows 的应用程序保护机制；可访问任何程序的内存区。

驱动程序就是一种操作系统内核程序，它对代码质量要求很高，错误的代码会让整个系统崩溃， CIH 病毒就是驱动程序（基于 Windows 9x 平台）的一个例

Evaluation Copy PDFmyPlus 4.0

子，它可执行特权指令，破坏硬件设备。

驱动程序有硬件驱动程序和软件驱动程序之分。硬件驱动程序，是连接硬件和操作系统之间的纽带。它一方面对硬件直接操作，另一方面提供标准接口供操作系统调用。软件驱动程序位于硬件驱动程序之上，没有直接操作硬件的动作，它们要在上层驱动程序和下层驱动程序之间做一个中间操作。这种软件驱动程序有着很重要的作用，它们可以提供分层式的服务。Tcpip.sys 就是软件驱动程序。

3.3.2 Windows 网络驱动程序

网络驱动程序是 Windows 2000 设备驱动程序的一种。它使用 OSI 标准，它的顶层是应用软件层，底层是硬件连接和网络的拓扑结构，网卡给大多数的平台提供网络的硬件接口，网卡驱动程序提供符合 NDIS（网络驱动程序接口规范）的服务，NDIS 又为传输层提供标准的网络接口。

Windows 网络驱动程序结构可用下图表示，下图也说明了 NDIS 在 Windows 中所处的位置：

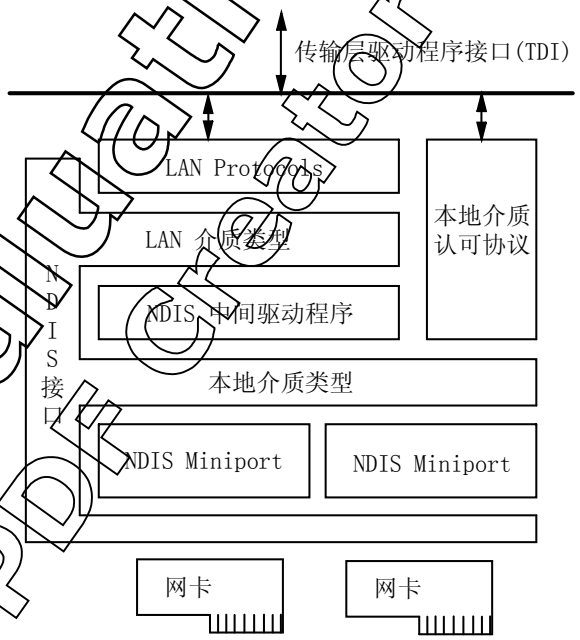


图 3.6 Windows 网络驱动程序结构

NDIS 是 Network Driver Interface Specification 的缩写，网卡厂商根据 NDIS 提供管理硬件特殊细节的 Miniport 驱动程序，同时 NDIS 为传输层提供标准的网络接口，所有的传输层驱动程序都需要调用 NDIS 接口来访问网络。

NDIS 支持 3 种类型的驱动程序：

Evaluation Copy
PDFCreator Plus 4.0

1．Miniport（微端口）驱动程序。微端口驱动程序可以通过 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。

2．中间驱动程序。中间驱动程序位于 Miniport 和 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。

3．Protocol（协议）驱动程序。协议驱动程序开放 Protocol 接口供底层驱动程序调用，实现 Protocol 接口与 Miniport 接口的对接。

TDI（传输驱动程序接口）提供了仅可被 TDI 客户方所使用的 TDI 接口，协议驱动程序实现了 TCP/IP 和 IPX/SPX 等协议。Tcpip.sys 是 TCP/IP 协议的实现，它就是一个协议驱动程序，它为 TDI 客户提供 TDI 接口。

TDI 接口包括以下内容：

1．一组 Dispatch 例程，客户通过调用支撑例程向传输驱动程序提交 I/O 请求（IRP），如 ZwXxxFile 例程和 IoCallDriver；

2．一组由 TDI 客户导出的 ClientEventXxx 回调例程，通过向低层传输驱动程序注册这些例程来接收特定的网络事件通知；

3．一组由 TDI 客户导出的 ClientPnPXxx 回调例程，通过向 TDI 注册这些例程来接收有关动态绑定、网络地址和电源状态改变等来自支持 PnP 的设备通知。

4．与 TDI 传输、ClientEventXxx 例程和 ClientPnPXxx 例程相关的参数、结构、IOCTL 和程序规则；

5．一组系统提供的 TdiXxx 函数，通过调用这些函数，传输驱动和客户进行相互通信；

6．一组系统提供的 TdiBuildXxx 宏和函数，客户用它们生成提交给低层传输驱动的 I/O 请求。

3.4 个人防火墙实现技术

前面介绍了 Windows 2000 操作系统的网络体系结构和网络驱动程序，特别是详细地说明了 TDI 接口内容，这些都是开发 PFW 个人防火墙的基础。接下来将论述为什么要用驱动程序实现防火墙。

前面我们谈到，要实现包过滤防火墙，首要解决的问题是网络数据包拦截。

Evaluation Copy PDF Creator Plus 4.0

要拦截 Windows 下的网络数据包可以在操作系统的两个层面实现：应用层和核心层。我们还谈到，只有在核心层拦截数据包才能让我们的防火墙更有效、更强大、更能安全地保护我们的数据。

3.4.1 应用层包拦截技术

1．Winsock Layered Service Provider （LSP）。这种方法在 MSDN 里有很详细的文档，并且给出了一个例子（SPI.CPP）。这种方法的好处是可以获得调用了 winsock 的进程详细信息。这就可以用来实现防火墙、数据流加密等目的。但是，如果应用程序直接通过 TDI 使用 TCP/IP 来发送数据包，这种方法就无能为力了。对于一些木马和病毒来说要实现通过 TDI 直接使用 TCP/IP 是一件很容易的事情。因此，大多数的个人防火墙都不使用这种方法。国内也有使用该方法实现的个人防火墙，例如 Xfilter（见参考书目《Windows 防火墙与网络封包截获技术》）。

2．Windows 2000 包过滤接口。Windows 2000 IPHLP API 提供了安装包过滤器的功能。但是，包过滤的规则有很多限制，对于个人防火墙来说是远远不够的。

3．替换系统自带的 winsock 动态连接库（ws2_32.dll）。这种方法同样存在着第一种方法存在的问题。

3.4.2 核心层包拦截技术

应用层包拦截技术不能有效的实现对所有网络数据包的拦截，所以我们应该利用核心层包过滤技术实现防火墙，主要方法有：

1．TDI 过滤驱动程序（TDI Filter Driver）

当应用程序要发送或接收网络数据包的时候，都是通过与协议驱动所提供的接口来进行的。协议驱动提供 TDI 接口标准。在 Windows 2000/NT 下，ip、tcp、udp 是在一个驱动程序里实现的，它就是 Tcpip.sys，这个驱动程序创建的设备有：DeviceRawIp、DeviceUdp、DeviceTcp、DeviceIp 和 DeviceMULTICAST。应用程序所有的网络数据操作都是通过这几个设备进行的。因此，我们只需要开发一个过滤驱动来替换这些交互的接口，就可以实现网络数据包的拦截。TDI 层的网络数据拦截还可以得到操作网络数据包的进程详细信息，这也是个人防火墙的一

Evaluation Copy
BPCreator Plus 4.0

个重要功能。

2．NDIS 中间层驱动程序（NDIS Intermediate Driver）

中间层驱动介于协议驱动和微端口驱动之间，对上开放出一个 Miniport 接口，对下开放出 Protocol 接口，它能够截获所有的网络数据包。NDIS 中间层驱动的应用很广泛，不仅仅是个人防火墙，还可以用来实现 VPN，NAT，PPPOverEthernet 以及 VLan。Windows DDK 提供了两个著名的中间层驱动例子：Passthru 以及 Mux。开发人员可以在 Passthru 的基础上进行开发，Mux 则实现了 VLan 功能。目前个人防火墙的产品还很少用到这种技术，主要的原因在于中间层驱动的安装过于复杂，尤其是在 Windows NT 下。Windows 2000 下可以通过程序实现自动安装，但是如果驱动没有经过数字签名的话，系统会提示用户是否继续安装。中间层驱动功能强大，应该是今后个人防火墙技术的趋势所在，特别是一些附加功能的实现。

3．Win2k Filter-Hook Driver

这是从 Windows2000 开始系统所提供的一种驱动程序，该驱动程序主要是利用 ipfiltdrv.sys 所提供的功能来拦截网络数据包。Filter-Hook Driver 在结构非常简单，易于实现。但是正因为其结构过于简单，并且依赖于 ipfiltdrv.sys，Microsfot 并不推荐使用 Filter-Hook Driver。

4．NDIS Hook Driver。

Hook 是一个很流行的概念。实现它的方式也有很多，一种是修改 NDIS.SYS 的 Export Table，一种是向系统注册假协议（Fake Protocol）。这种方法对平台的依赖性比较大，需要在程序中判断不同的操作系统版本而使用不同的结构定义。

总之，TDI 过滤驱动程序是一种简便、高效、实用的防火墙实现方法。

3.5 TDI 过滤驱动程序开发

本文探讨的 PFW 个人防火墙是基于 Windows 的包过滤型个人防火墙，它的核心是 TDI 过滤驱动程序。我们再来探讨核心层的 TDI 过滤驱动程序开发技术。

3.5.1 驱动程序开发环境

我的 Windows 2000 驱动程序开发环境是：Visual C++ .NET 2003 + Win2000

Evaluation Copy PDFCreator-Plus4.0

DDK + DebugView 4.21。

开发 Windows 2000 驱动程序必须备有 Windows 2000 DDK，DDK（Device Driver Kit，设备驱动程序资源包）是用来开发驱动程序的工具包。它包含开发驱动程序所需的库文件、头文件、文档、示例和一些必备工具。

Windows 2000 驱动程序的开发，要使用 C/C++语言，所以还要选择 C/C++语言的编译器，我选择的是 Visual C++ .NET 2003。

驱动程序需要更高级的调试技术，最常用的是双机、双监视器，但我的条件有限，只好用 DebugView 4.21 作为调试器，监视驱动程序的运行状态。

3.5.2 TDI 过滤驱动程序的特性

过滤驱动程序通过创建一个或多个设备对象（DeviceObject）直接挂接到一个现有的驱动程序设备对象之上。当有应用程序或其他驱动程序调用这个设备对象时，会首先映射到过滤驱动程序之上，然后由过滤驱动程序再传递给原来的设备对象。一个设备驱动程序允许挂接多个过滤驱动程序，它们按安装的先后顺序从下到上依次排列。不过有一些非标准的过滤驱动程序编写方法，仅仅使最后安装的过滤驱动程序有效，而忽略了以前安装的中间驱动程序。下图说明了一个过滤驱动程序 TcpipDog1.sys 挂接到 Tcpip.sys 协议驱动程序之上，然后又在 Tcpip.sys 上挂接了一个 TcpipDog2.sys，可能产生的两种情况。挂接前后的工作模式对比如图：

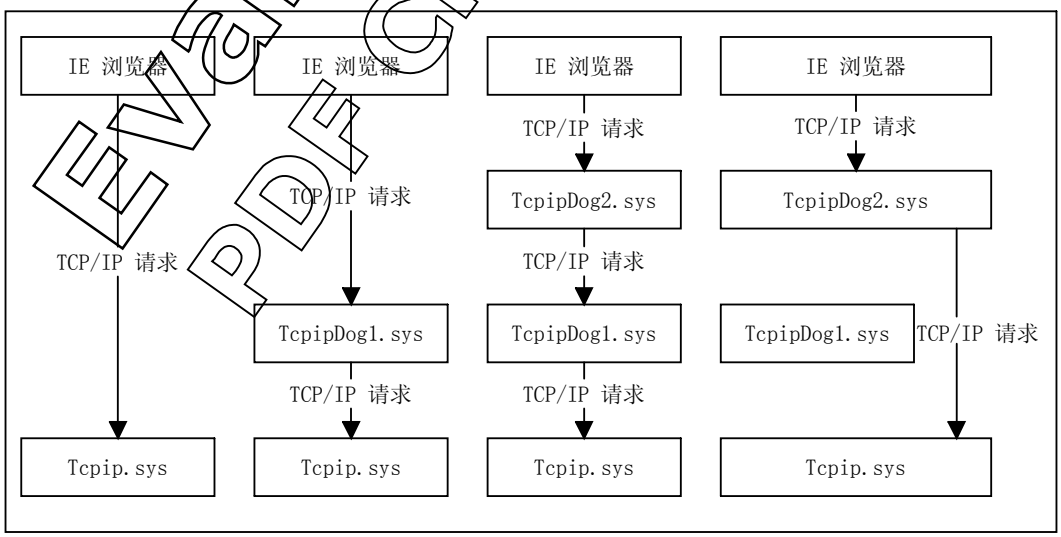


图 3.7 挂接过滤驱动程序前后对比

TDI 过滤驱动程序挂接在协议驱动程序之上，Tcpip.sys 是 TCP/IP 协议族驱

Evaluation Copy
PDF Creator Plug 4.0

动程序，它创建的设备有：DeviceRawIp、DeviceUdp、DeviceTcp、DeviceIp 和 DeviceMULTICAST。实现了 TCP、UDP、IP 协议。TDI 过滤驱动程序应该创建这些设备对象，并挂接到 Tcpip.sys 驱动程序之上。过滤驱动程序对用户程序是透明的，上层的应用程序或驱动程序认为自己接触的仍然是 Tcpip.sys，而实际上，请求在发往底层时已经全部经过了过滤驱动程序的过滤。

3.5.3 TDI 过滤驱动程序结构

过滤驱动程序必须有入口函数 DriverEntry，任何驱动程序的调用都必须首先执行 DriverEntry 函数。在 DriverEntry 函数中最重要的工作就是设置开放给其他程序调用的函数指针。这样，其他程序就可以直接调用这个驱动程序的功能函数来完成任务。

```
NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject,
                     IN PUNICODE_STRING RegistryPath)
{
    status = create_attach_device( DriverObject,
                                   L"\\Device\\Tcp",
                                   L"\\Device\\pfw_Tcp",
                                   &g_TcpDevice );

    DriverObject->DriverUnload = OnUnload;

    for (int i=0; i<=IRP_MJ_MAXIMUM_FUNCTION; i++)
        DriverObject->MajorFunction[i] = TdiDispatch;

    return 0;
}
```

create_attach_device()函数创建了 pfw_tcp 设备，并将它挂接在 DeviceTcp 设备之上，接收用户的 TCP 请求。

DriverObject->DriverUnload 是卸载驱动程序的函数指针，在卸载驱动程序时会直接调用 OnUnload 函数。

DriverObject->MajorFunction[] 就是一组开放给其它程序调用的函数指针。高层程序向设备发送 IRP（I/O Request Packet，I/O 请求包），TdiDispatch 分析处理这些 IRP。

```
NTSTATUS TdiDispatch(IN PDEVICE_OBJECT DeviceObject,
                     IN PIRP Irp)
```

Evaluation Copy PDF Creator Plus 4.0

```
{
    NTSTATUS status = STATUS_SUCCESS;
    PIO_STACK_LOCATION    irps = IoGetCurrentIrpStackLocation(Irp);

    switch(irps->MajorFunction)
    {
    case IRP_MJ_CREATE:
        DBGPRINT("TdiDispatch(IRP_MJ_CREATE)...");
        break;
    case IRP_MJ_CLOSE:
        DBGPRINT("TdiDispatch (IRP_MJ_CLOSE)...");
        break;
    case IRP_MJ_CLEANUP:
        DBGPRINT("TdiDispatch (IRP_MJ_CLEANUP)...");
        break;
    case IRP_MJ_DEVICE_CONTROL:
        DBGPRINT("TdiDispatch (IRP_MJ_DEVICE_CONTROL)...");
        break;
    case IRP_MJ_INTERNAL_DEVICE_CONTROL:
        DBGPRINT("TdiDispatch (IRP_MJ_INTERNAL_DEVICE_CONTROL)...");
        switch (irps->MinorFunction)
        {
        case TDI_ACCEPT:
            break;
        case TDI_ACTION:
            break;
        case TDI_ASSOCIATE_ADDRESS:
            break;
        case TDI_DISASSOCIATE_ADDRESS:
            break;
        case TDI_CONNECT:
            break;
        case TDI_DISCONNECT:
            break;
        case TDI_LISTEN:
            break;
        case TDI_QUERY_INFORMATION:
            break;
        case TDI_RECEIVE:
            break;
        case TDI_RECEIVE_DATAGRAM:
            break;
        case TDI_SEND:
            break;
```

Evaluation Copy PDF Creator Plus 4.0

```
            case TDI_SEND_DATAGRAM:
                break;
            case TDI_SET_EVENT_HANDLER:
                break;
            case TDI_SET_INFORMATION:
                break;
            default:
                break;
            }
            break;
        default:
            DBGPRINT("TdiDispatch (OTHER_MAJOR_FUNCTION)...");
            break;
        }
        return DispatchToLowerDevice (DeviceObject,Irp);
}
```

TdiDispatch 可以处理这些用户的 IRP ，也可由 DispatchToLowerDevice()函数将 IRP 请求转发给 Tcpip.sys。

3.5.4 自旋锁和同步事件

1．临界区与自旋锁

Windows 2000 操作系统为每个进程分配 4GB 的虚拟内存空间，它分为两部分：高端 2GB 内存空间（0xC0000000－0xFFFFFFFF），低端 2GB 内存空间（0x00000000－0xBFFFFFFF）。高端 2GB 的内存空间是系统保留空间，每个进程的高端内存都映射到同一物理空间，操作系统装载所有驱动程序到这块内存区域中。所以同一个驱动程序的同一变量可能被不同进程并发访问。当多个进程同时使用同一个公共变量时，需要对变量的访问进行同步，使用临界区技术可以很好的解决公共变量的同步访问。

在驱动程序中可用自旋锁实现临界区。自旋锁是由内核定义的内核模式仅有的一种同步机制，它对外提供一种不透明类型 KSPIN_LOCK。在同步公共资源访问之前，驱动程序必须为它所使用的自旋锁提供存储空间，并调用 KeInitializeSpinLock 来初始化自旋锁。当进入临界区和退出临界区时使用 KeAcquireSpinLock 和 KeRelaeseSpinLock。

持有自旋锁的进程一定要避免做下列事情：引起硬件异常或软件异常；试图访问可分页内存；做可能引起死锁或自旋锁持有时间超过 25 毫秒的递归调用；

Evaluation Copy PDFCreator Plus 4.0

试图获得另一个自旋锁（这样做可能会导致死锁）；调用一个违反了上述任一条规则的外部程序。

2．核心层线程与应用层线程间的通信

在 PFW 防火墙的开发中，有时驱动程序有一事件产生且需要及时通知用户应用程序。驱动程序运行在核心层，而用户程序运行在应用层，它们之间的通信不能使用 Windows 消息机制，需要使用同步事件实现通信。

一种简便有效的方法是驱动程序和应用程序共用一个同步事件。驱动程序调用 IoCreateSynchronizationEvent 创建一个有名事件，应用程序调用 CreateEvent/OpenEvent 打开此有名事件即可。

同一个有名事件，驱动程序中的名字在“\BaseNamedObjects\”下，例如应用程序用“xxxEvent”,那么驱动中就是“\BaseNamedObjects\xxxEvent”。还要注意不要在驱动初始化的时候创建事件，此时大多不能成功创建。

驱动程序中创建同步事件代码片段：

```
void CTdiNotify::Initialize()
{
    //...
    UNICODE_STRING event_name;
    RtlInitUnicodeString( &event_name, L"\\BaseNamedObjects\\PFW_EVENT" );
    m_event = IoCreateSynchronizationEvent( &event_name, &m_event_handle );
    DBGPRINTVAR(m_event);
    if ( NULL == m_event )
    {
        DBGPRINT("m_NotifyEvent can not create. ");
        m_event_handle = NULL;
    }
    else
    {
        KeClearEvent(m_event);
    }
    //...
}
```

应用层程序创建事件代码片段：

```
HANDLE StartWaitNotifyThread()
{
    // 创建接受设备通知的同步事件
    g_hEvent = OpenEvent( SYNCHRONIZE, FALSE, _T("PFW_EVENT"));
    if ( NULL == g_hEvent )
```

Evaluation Copy PDF Creator Plus 4.0

```
    {
        MessageBox(NULL,_T("Can not create event!"), NULL, MB_ICONERROR);
        return NULL;
    }

    g_hThread = CreateThread( NULL, 0,
                        (LPTHREAD_START_ROUTINE)WaitThread,
                        0, 0, 0);
    return g_hThread;
}
```

驱动程序中发出通知代码片段：

```
NTSTATUS CTdiNotify::Notify( ULONG type, PVOID pContext, USHORT lSize )
{
    NTSTATUS status;

    //...

    KeSetEvent( m_event, 0, FALSE );
    status = STATUS_SUCCESS;

    //...

    return status;
}
```

应用层程序等待通知代码片段：

```
long WINAPI WaitThread( LPARAM )
{
    while (1)
    {
        WaitForSingleObject( g_hEvent, INFINITE );
        // 创建线程,分发通知
        CreateThread( NULL, 0,
                (LPTHREAD_START_ROUTINE)DispatchThread,
                0, 0, 0);
    }
    return 0;
}
```

本章探讨了用 TDI 过滤驱动程序实现个人防火墙的基本原理和方法，并解决了驱动程序编程的几个问题，下一章的内容是实现一个基于 TDI 过滤驱动技术的个人防火墙——PFW 个人防火墙的方法。

Evaluation Copy
PDF Creator Plus 4.0

第 4 章 PFW 个人防火墙的实现

4.1 系统设计

4.1.1 功能分析

PFW 个人防火墙的核心技术是包过滤，它工作在传输层，它具备一般防火墙应具备的功能：

1．根据安全规则对进出网络的数据包进行过滤；

2．根据应用程序访问规则对应用程序连网动作进行过滤；

3．对应用程序访问规则具有自学习功能；

4．可实时监控、监视网络活动；

5．日志记录网络访问动作的详细信息；

6．受到攻击时给用户报警提示。

4.1.2 系统结构

PFW 个人防火墙工作在传输层，使用传输层过滤驱动程序方式，创建并挂接三个 I/O 设备。PFW 软件的工作原理如图：

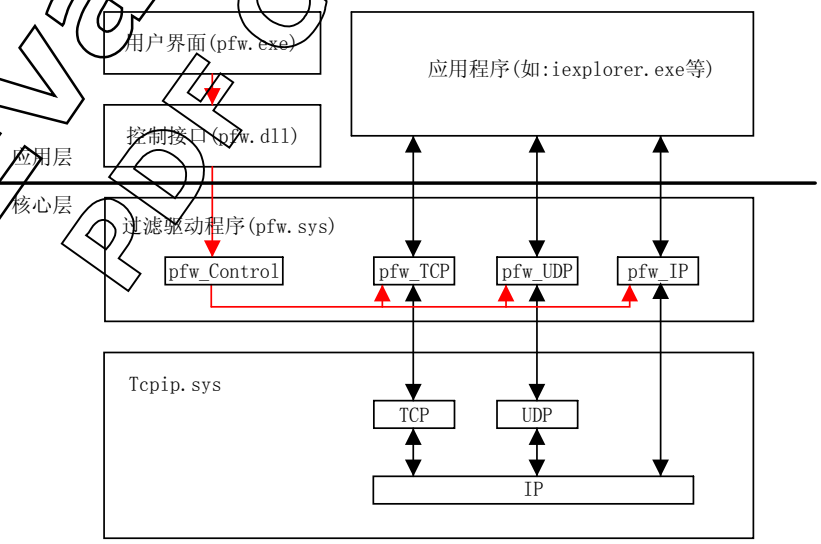


图 4.1 PFW 个人防火墙系统结构

Evaluation Copy
PDFCreator Plus 4.0

PFW 防火墙由三个部件组成：用户界面、驱动控制接口和过滤驱动程序。

1．用户界面是防火墙与用户的接口，它是整个系统的控制中心。提供读写安全规则，修改安全规则的功能，并向用户显示网络数据包的状况，提供日志记录等功能。

2．驱动控制接口提供给用户界面对防火墙驱动程序的控制功能。它使用户界面部件对 PFW 个人防火墙的操作变得简单，还能用它开发出更完善的界面。它是一个动态链接库，不仅增强了代码的复用性，还使整个 PFW 防火墙的可维护性大大提高。

3.过滤驱动程序进行网络数据包的过滤，创建三个设备（pfw_TCP、pfw_UDP 和 pfw_IP）并挂接到 Tcpip.sys 对应的三个设备上，实现数据包的监视和过滤。同时创建了 pfw_Control 设备，用来维护过滤规则，控制网络动作。

4.1.3 安全规则结构

PFW 防火墙的核心是数据包过滤，数据包过滤必定要有一定的规则，称为安全规则。根据防火墙的功能要求，PFW 防火墙有两类安全规则：过滤规则和应用程序访问规则。

1．过滤规则

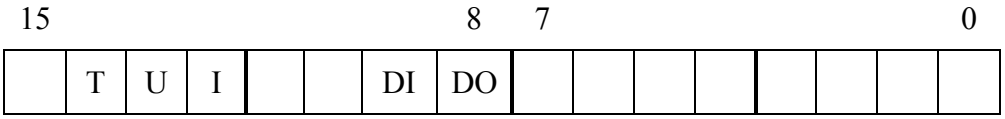
过滤规则是 PFW 防火墙过滤数据包的主要依据，它针对所有需要网络操作的应用程序。当数据包符合过滤规则的条件时，防火墙执行指定的动作 action。

```
typedef struct _FILTER_RULE
{
    LPTSTR name;              //规则名称;   在驱动程序模块中不使用,无意义
    LPTSTR description;       //规则说明;   在驱动程序模块中不使用,无意义
    USHORT type;              //规则类型字
    USHORT action;            //过滤动作字
    ULONG  ip;                //对方 IP 地址
    ULONG  mask;              //地址掩码
    USHORT port1;             //起始端口;   当数据包方向为 in, 表示本地端口
    USHORT port2;             //结束端口;   当数据包方向为 out, 表示对方端口
} FILTER_RULE, *PFILTER_RULE;
```

Evaluation Copy
PDF Creator Plus 4.0

（1）规则类型字（type）

指出这条规则对应的数据包类型。各位意义如下：



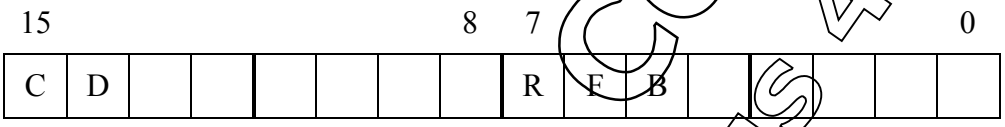
协议标志位：T（TCP），U（UDP），I（IP）。

方向标志位：DI（DIRECT IN），DO（DIRECT OUT）。

其它位不使用，置为零。

（2）过滤动作字（action）

指明要采取的网络动作。各位意义如下：



C(Continue)：为 1 表示继续检查下一条规则。过滤规则以链式组织，Continue 表示当数据包符合此条规则时，不停止在规则链的检查，仍继续下一规则的检查。

D(Deny)：0 表示允许数据包通行，1 表示禁止通行。

R(Record)：1 表示记录日志。

F(Flash)：1 表示闪烁图标。

B(Beep)：1 表示发声报警。

2．应用程序访问规则

它针对指定的应用程序，这个应用程序只允许使用规则中允许的网络服务。

```
//应用程序安全规则结构
//    如果 port_num 等于 0,表示可访问端口从 ports[0]到 ports[1],当 ports[0]
//    和 ports[1]都为 0,表示无端口可访问
//    如果 port_num 大于 0,ports[port_num]是所有访问的端口列表 typedef struct _APP_RULE
{
    ULONG  size;          //结构大小
    LPTSTR app_name;      //应用程序文件名；    在应用层使用
    HANDLE     pid;       //进程 ID；(在核心层使用,在 DLL 和 exe 中不使用)
    USHORT allow;         //允许的操作(协议及方向,意义见下)
    USHORT action;        //过滤动作字; 不符合条件时的动作
    USHORT pt_tcp;        //pt(port list type): 端口列表类型
```

Evaluation Copy PDF Creator Plus 4.0

```
    USHORT pt_udp;              //PT_LIST          PT_RANGE
    USHORT pn_tcp;              //pn(port number): 端口数组大小
    USHORT pn_udp;              //
    USHORT ports[2];        //可访问端口列表(设成 4 是为了对齐)
} APP_RULE, *PAPP_RULE;

/*pt_xxx 的意义*/
#define PT_NULL             0       //无端口可访问
#define PT_LIST             1       //端口数组是一系列端口号
#define PT_RANGE            2       //端口数组指定了一个范围

/*allow 的意义*/
#define   ALLOW_TCP_IN      0x0001   //允许接收 TCP 数据
#define   ALLOW_TCP_OUT         0x0002   //允许发送 TCP 数据
#define   ALLOW_UDP_IN      0x0010   //允许接收 UDP 数据
#define   ALLOW_UDP_OUT         0x0020   //允许发送 UDP 数据
```

action 的意义同过滤规则。

4.2 过滤驱动程序

4.2.1 结构和功能

过滤驱动程序创建四个设备：pfw_TCP、pfw_UDP 和 pfw_IP 分别挂接到 tcpip.sys 的 TCP、UDP 和 IP 设备上，处理用户的 IRP（I/O 操作包）。pfw_Control 设备实现了 PFW 防火墙接口对过滤驱动程序的控制和管理。PFW 过滤驱动程序模块结构图见下页。

IRP 处理模块处理用户的 IRP，监视网络操作，记录网络状态，根据过滤规则和应用程序规则检查用户网络请求的合法性，实现数据包的过滤，并向 PFW 驱动控制接口发出通知，通知 PFW 向用户显示网络数据包状况。

pfw_Control 设备是 PFW 控制接口实现对过滤驱动程序控制的媒介，使用它实现过滤规则的维护、应用程序访问规则（简称为应用规则）的维护、通知的管理及其它的驱动控制（如：启用/停用防火墙、对象表维护等）。

Evaluation Copy
PDFCreator Plus 4.0

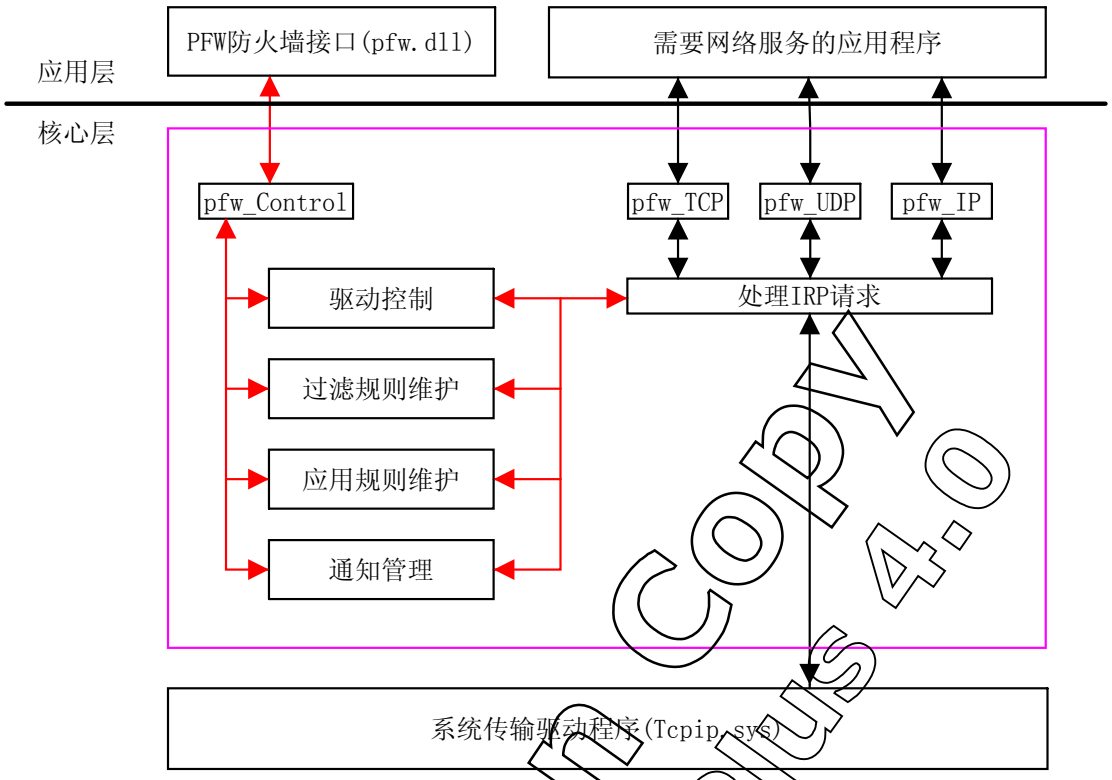


图 4.2 驱动程序结构

4.2.2 IRP 处理和回调事件

上层程序通过 IRP 请求低层设备，低层设备分析 IRP 并作出处理。同样，pfw.sys 过滤驱动程序也要分析处理上层程序发出的 IRP。

上层程序建立网络连接、发送接收数据等网络操作时，向驱动程序发出 TDI_MJ_DEVICE_CONTROL 和 TDI_MJ_INTERNAL_DEVICE_CONTROL 请求。

上层程序使用 TCP 协议发送数据包时首先要和远程结点建立联接，向驱动程序发出 TDI_CONNECT 请求，上层程序使用 UDP 协议发送数据报时，发出 TDI_SEND_DATAGRAM 请求。

情况也有例外，上层程序接受远程主机的 CONNECT 请求时，并不向驱动程序发出 TDI_CONNECT 请求，还有，上层程序使用 UDP 协议接收数据包时，也不发出 TDI_RECIVE——DATAGRAM 请求。

上层程序怎么处理呢？答案是设置回调事件。首先上层程序发出 TDI_SET_EVENT_HANDLER 请求，设置 TDI_EVENT_CONNECT 、TDI_EVENT_RECEIVE_DATAGRAM 事件。当驱动程序接受请求或接收数据包

Evaluation Copy PDFCreator Plus 4.0

时直接调用已设置的事件处理函数。

PFW 驱动程序需要处理的用户 IRP 有：IRP_MJ_CREATE、IRP_MJ_CLEANUP、IRP_MJ_DEVICE_CONTROL 和 IRP_MJ_INTERNAL_DEVICE_CONTROL，当用户的请求 IRP 是 IPR_MJ_INTERNAL_DEVICE_CONTROL 时，只处理如下 TDI 请求：TDI_ASSOCIATE_ADDRESS、TDI_DISASSOCIATE_ADDRESS、TDI_CONNECT、TDI_SEND_DATAGRAM 和 TDI_SET_EVENT_HANDLER。

4.2.3 对象表管理

驱动程序在处理 IRP 请求时，需要知道是什么进程在请求网络服务，以及这个进程的 IP 地址和端口是什么。所以在驱动程序中需要有相应的机制来记录和进程有关的信息，PFW 防火墙使用对象表管理来解决这个问题。

文件对象表包含三类信息：对象、进程 ID、IP 地址和事件句柄。

```
typedef struct _OBJECT_TABLE_ENTRY
{
    PFILE_OBJECT    fileobj;
    HANDLE          pid;
    ULONG           ip;
    USHORT          port;
    USHORT          reserved; //双字对齐
    PFW_EVENT_CONTEXT   *handlers;
    _OBJECT_TABLE_ENTRY *next;
} OBJECT_TABLE_ENTRY, *POBJECT_TABLE_ENTRY;
```

当应用层程序或其它高层驱动程序请求网络服务时，首先创建文件对象（FileObject），然后应用层程序或其它高层驱动程序发出 IRP 请求，把文件对象包含在 IRP 中。驱动程序根据 IRP 中的文件对象来识别请求访问的进程。我们可以在文件对象创建时，使用文件对象表记录进程的 IP 地址和端口号以及其它信息。

文件对象的种类多种多样，不同种类的 IRP 使用不同种类的文件对象。

使用驱动程序时，首先要向驱动程序发出 IRP_MJ_CREATE 请求，同时创

Evaluation Copy
PDF Creator Plus 4.0

建一个文件对象，这个文件对象可能是 TransportAddress，也可能是 ConnectionContext，驱动程序可以根据此文件对象来确定驱动所在的进程，所以 IRP 请求处理模块要向文件对象表中添加进程的属性、IP 地址端口等信息。结束网络服务时，上层程序向进程发出 IRP_MJ_CLEANUP，驱动程序删除文件对象表中相应的条目。

上层程序建立网络连接、发送接收数据时，向驱动程序发出 TDI_MJ_DEVICE_CONTROL 和 TDI_MJ_INTERNAL_DEVICE_CONTROL 请求，TDI_ASSOCIATE_ADDRESS 和 TDI_DISASSOCIATE_ADDRESS 请求是把地址（TransportAddress）对象和连接上下文（ConnectionContext）对象关联时发出的，ConnectionContext 代表了一个本地节点和远程节点的连接，所以也要向对象表中添加一个条目。

文件对象表记录了对象和进程的相应属性，为数据包过滤和设置回调事件函数提供支持。它的实现源代码见附录一（obj.h、obj.cpp）。

4.2.4 数据包过滤

PFW 驱动程序的数据包过滤由 IRP 请求处理模块完成，该模块的详细源代码见附录一（dispatch.cpp、disp_crt.cpp、disp_flt.cpp、disp_ev.cpp、filter.cpp）。

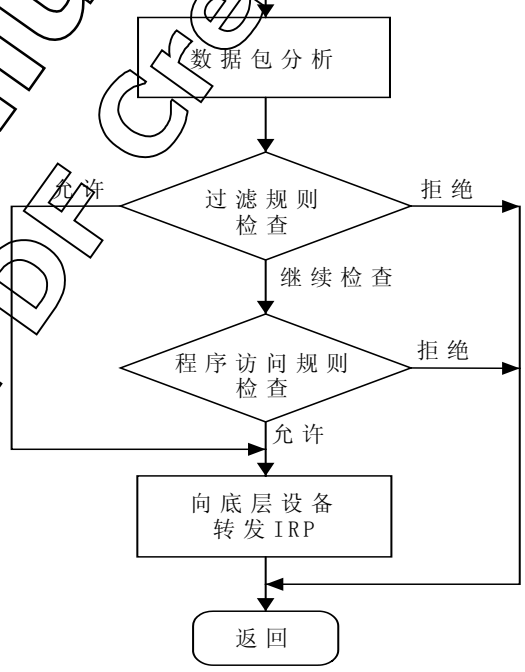


图 4.3 IRP 请求处理流程

Evaluation Copy
PDF Creator Plus 4.0

当上层程序接收或发送数据包时，会向驱动程序发出 TDI_CONNECT、TDI_SEND_DATAGRAM 请求，或是驱动程序调用上层程序设置的回调事件。所以数据包的过滤应在 IRP 处理函数和回调函数中实现。

首先分析 IRP，根据 IRP 中的文件对象，在对象表中查出数据包的本地地址、远程地址和传送方向等数据表基本信息，再检查它是否符合防火墙规则要求，不符合时返回错误，不再向底层转发请求，起到防火墙过滤作用。

包过滤流程如上页图。下面是处理 TDI_SEND_DATAGRAM 请求的过滤源代码：

```
NTSTATUS TdiDispatchSendDatagram( IN PDEVICE_OBJECT  DeviceObject, IN PIRP  Irp )
{
    NTSTATUS status;
    PIO_STACK_LOCATION  irps = IoGetCurrentIrpStackLocation(Irp);
    PTDI_REQUEST_KERNEL_SENDDG                    param                    =
(TDI_REQUEST_KERNEL_SENDDG*)(&irps->Parameters);
    TA_ADDRESS          *remote_addr  =          ((TRANSPORT_ADDRESS
*)(param->SendDatagramInformation->RemoteAddress))->Address;
    sockaddr_in *remote_saddr = (sockaddr_in*)(&remote_addr->AddressType);

    //协议类型
    USHORT proto;
    if ( DeviceObject == g_TcpDevice )
        proto = RT_TCP;
    else if ( DeviceObject == g_UdpDevice )
        proto = RT_UDP;
    else
        proto = RT_IP;

    HANDLE pid = PsGetCurrentProcessId();
    ULONG ip = 0;
    USHORT port = 0;
    status = g_Objects.GetInfo( irps->FileObject, &pid, &ip, &port );

    //数据包信息
    PACKET_INFO pi;
    pi.pid = pid;          //进程 ID
    pi.type = proto | RT_DIRECTOUT;      //数据包类型,意义见规则类型字
    pi.ip = ntohl( remote_saddr->sin_addr.s_addr );              //对方 IP 地址
    pi.lport = port;            //本地端口(local port)
    pi.rport = ntohs(remote_saddr->sin_port);          //对方端口(remote port)
```

Evaluation Copy. PDF Creator Plus 4.0

```
    //过滤
    USHORT action = Filter(&pi);
    //记录日志
    status = Log( action, &pi );

    if ( IS_ACTION_PASS(action) )
        return DispatchToLowerDevice( DeviceObject, Irp );

    status = Irp->IoStatus.Status = STATUS_ACCESS_VIOLATION;
    IoCompleteRequest (Irp, IO_NO_INCREMENT);

    return status;
}
```

4.2.5 规则管理

数据包过滤核心就是规则检查，PFW 防火墙有两种规则：过滤规则、应用程序访问规则。规则管理模块负责这两种规则的维护：添加、删除和修改。上层的程序通过 pfw_Control 设备实现对这两种规则的维护。

一系列过滤规则按照顺序组成过滤规则列表，程序中使用链式存储结构，利用自旋锁（Spin Lock）保证多个线程共享过滤规则表，过滤规则表用类 CFilterRules 实现，源代码参见附录一（frule.h、frule.cpp）。程序访问规则以集合的形式组成程序访问规则表，表中各条目不分先后，但都各不相同。程序访问规则表用类 CAppRules 实现，源代码参见附录一（arule.h、arule.cpp）。

规则管理模块还提供了按规则对数据包进行检查。（流程图分别见下页）

4.2.6 通知管理

当驱动程序需要记录网络日志时，应及时通知主界面程序，主界面程序还要知道日志的基本信息，这里就要用到驱动程序与应用程序间的通信技术。

通知管理模块使用类 CTdiNotify 实现了驱动程序（pfw.sys）与驱动控制接口程序（pfw.dll）的通信，源代码参见附录一（log.h、notify.h、notify.cpp）。

PFW 防火墙驱动程序使用事件实现与驱动控制接口程序的通信。当有日志需要显示给用户，调用 CTdiNotify:: Notify()函数； CTdiNotify:: Notify()设置事件，使驱动控制接口程序中等待它的线程得以继续执行，再通过操作 pfw_Control

Evaluation Copy PDF Creator Plus 4.0

设备取得日志的详细信息。

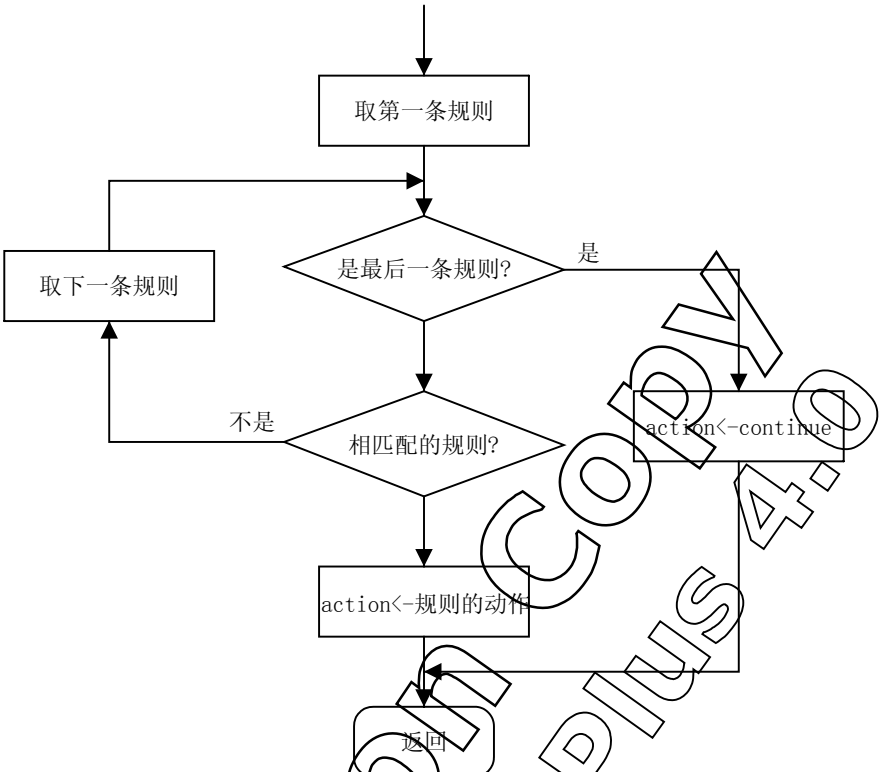


图 4.4 过滤规则检查流程图

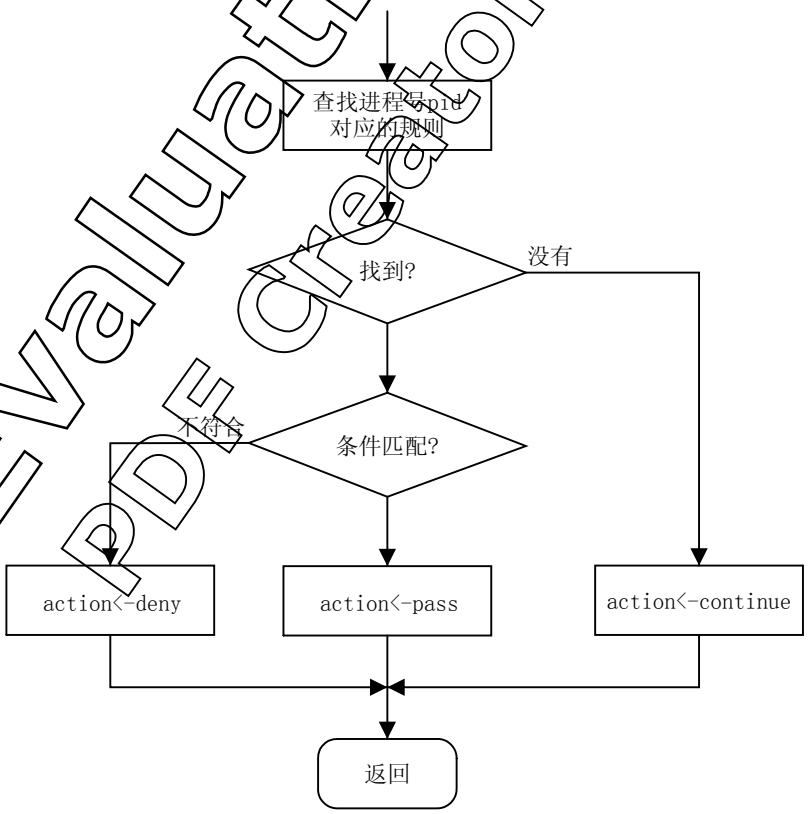


图 4.5 程序访问规则检查流程图

Evaluation copy
PDF Creator Plus 4.0

4.2.7 pfw_Control 设备与驱动控制

pfw_Control 设备的作用正如其名，是用来实现对 PFW 防火墙驱动程序的控制。利用它可以实现规则的维护，可以实现通知的管理，可以实现过滤功能的启动与停止，它是用户命令的传递者。源代码参见附录一（ioctl.h、disp_ctl.cpp）。

pfw_Control 是 PFW 过滤驱动程序创建的自定义设备（参见附录一 drvmain.cpp），它不同于 pfw_TCP、pfw_UDP 和 pfw_IP 设备，不是挂接在 Tcpip.sys 的任何设备之上，它直接处理用户的 IRP_MJ_DEVICE_CONTROL 请求，执行下列 I/O 控制命令：

IOCTL_CMD_FRULE_INSERT：插入过滤规则；

IOCTL_CMD_FRULE_DELETE：删除过滤规则；

IOCTL_CMD_FRULE_MODIFY：修改过滤规则；

IOCTL_CMD_FRULE_MOVEDOWN：下移过滤规则；

IOCTL_CMD_ARULE_INSERT：插入应用规则；

IOCTL_CMD_ARULE_DELETE：删除应用规则；

IOCTL_CMD_ARULE_MODIFY：修改应用规则；

IOCTL_CMD_PFW_ENABLE：允许数据包过滤；

IOCTL_CMD_NOTIFY_GETINFO：取得通知信息；

IOCTL_CMD_NOTIFY_START：启动通知功能；

IOCTL_CMD_NOTIFY_STOP：停止通知功能；

IOCTL_CMD_CLEAR_ALL_RULES：清除所有规则（包括过滤规则和应用规则）。

4.3 驱动控制接口

4.3.1 结构和功能

驱动控制接口程序（pfw.dll）是一个动态链接库，介于主界面程序和过滤驱动程序之间。它为主界面提供了 PFW 防火墙的规则维护、日志管理和防火墙控制接口。实际上，主界面程序（pfw.exe）可直接使用过滤驱动程序（pfw.sys）提供的功能，pfw.dll 的存在提高了软件的可复用性、可维护性，也降低了软件开

Evaluation Copy
PDF Creator Plus 4.0

发的复杂性。

本层的组成模块及在系统的位置如下图所示：

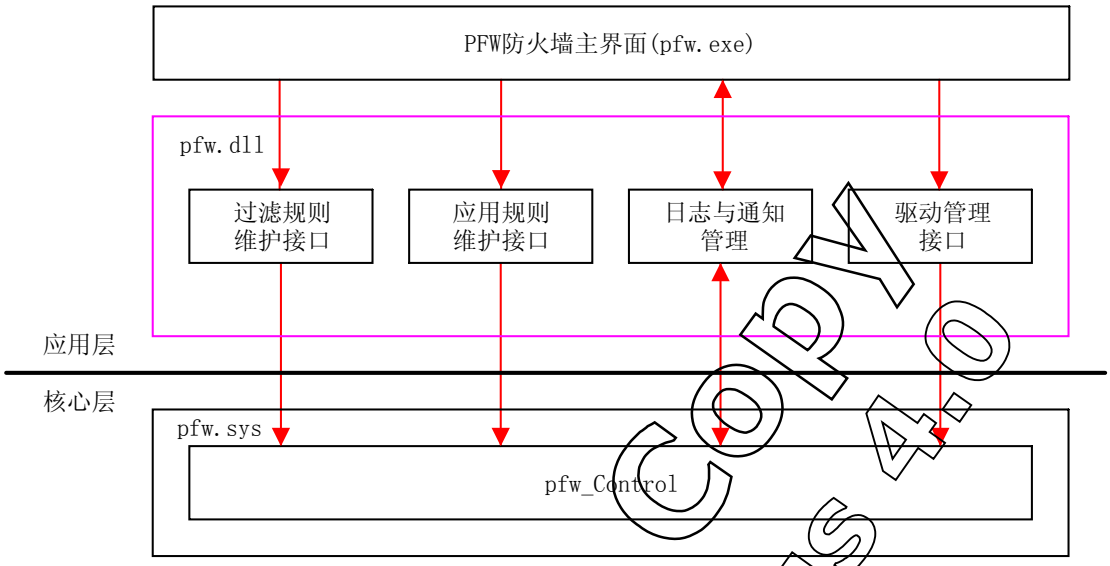


图 4.6 控制接口结构

本层为上层的界面程序提供接口，并调用下层的驱动程序提供的功能。下层的驱动程序创建了 pfw_Control 设备，利用它实现对防火墙驱动程序的控制。

要使用防火墙驱动控制接口，首先调用 PfwInitialize()函数，该函数调用 CreateFile()创建一个 pfw_Control 设备，然后就可以利用 DeviceIoControl()向 pfw_Control 设备发出 IRP，实现防火墙的控制。详细源代码参见附录二（pfwdll.h、dllmain.cpp）。

4.3.2 过滤规则维护接口

过滤规则维护接口提供了防火墙过滤规则的维护接口：PfwFilterRuleInsert()、PfwFilterRuleDelete()、PfwFilterRuleModify()、PfwFilterRuleMoveUp()、PfwFilterRuleMoveDown()。

从它们的名称可以知道它们的功能。它们都是直接调用 DeviceIoControl()向驱动程序发出相应的 IRP 请求。源代码参见附录二（d_filter.cpp）

4.3.3 应用程序访问规则维护接口

应用规则维护接口的实现源代码参见附录二（d_arule.h、d_arule.cpp、d_pidset.h、d_pidset.cpp、d_app.cpp）。它为上层和主界面程序提供应用程序访问

Evaluation Copy
PDF Creator Plus 4.0

规则维护接口：PfwAppRuleInsert()、PfwAppRuleDelete()、PfwAppRuleModify()，即插入、删除和修改应用程序访问规则。

在应用层（pfw.exe、pfw.dll），应用程序访问规则的关键字是程序的全路径文件名。而在核心层（pfw.sys）由于很难取得当前正运行进程的文件名，应用程序访问规则不能再使用文件名，而是使用进程 ID 作关键字。所以，应用规则维护接口对外提供以文件名为关键字的维护操作，对内调用低层驱动提供的以进程 ID 为关键字的维护操作。

这样，接口程序就要实现以进程文件名为关键字的规则到以进程 ID 为关键字的规则间的对应。类 CAppRules 以文件名为关键字存储应用程序访问规则，类 CPidSet 记录当前正在运行的进程 ID 和进程文件名。当需要插入、删除或修改规则时，先从类 CPidSet 中查找进程 ID 的文件名，再操作驱动插入、删除或修改类 CAppRules 中相应的规则。 同时，当有新进程创建时，还需要查找出该进程的文件名所对应的规则，并将此规则插入到 PFW 防火墙驱动程序中。

4.3.4 日志与通知管理

有时驱动程序发生一事件（如：有数据包被拦截）时，需要及时通知用户，使用户知道发生了什么事件。PFW 系统分为驱动、接口和界面三层，驱动和接口之间的通信使用事件机制，而接口和界面之间的通信使用 Windows 消息机制。PFW 驱动控制接口等待驱动的事件，当事件发生时向主界面窗口发送 WM_PFW_NOTIFY 消息。详细源代码参见附录二（d_notify.cpp）。

控制接口程序处理驱动发来的四种通知：

1．日志记录通知（NT_LOG）

有数据包拦截日志产生，接口程序向主窗口发送 WM_PFW_NOTIFY（NT_LOG）消息，让主界面记录日志。

2．进程请求网络操作通知（NT_TDI_REQUEST）

当新的进程请求网络操作，但没有找到相应的应用程序访问规则，也就是说 PFW 防火墙驱动程序不知道怎样处理网络数据包时，发出此通知。接口程序可以根据用户的要求插入新规则到驱动中，让驱动程序根据新规则拦截或放行数据包。

Evaluation Copy PDFCreator Plus 4.0

3．进程创建通知（NT_PROCESS_CREATE）

当有进程创建时，发出此通知。

4．进程终止通知（NT_PROCESS_TERMINATE）

当有进程终止时，发出此通知。

4.3.5 驱动管理接口

该模块的详细代码参见附录二（d_manager.cpp）。提供的功能有：防火墙过滤功能的启动和停止、设置和取得接收 WM_PFW_NOTIFY 消息的窗口句柄。

4.4 用户界面

4.4.1 结构和功能

图 4.7 过滤规则设置界面

PFW 防火墙主界面采用类似属性页的界面风格。它分成四个部分：数据包监视界面、过滤规则设置界面、程序访问规则设置界面和系统设置界面。数据包监视界面记录拦截或放行的数据包信息；过滤规则设置界面和程序访问规则设置

Evaluation Copy
Protector Plus 4.0

界面让用户管理维护各种规则；系统设置界面提供一些其它简单设置。

主界面程序的模块结构图如下：

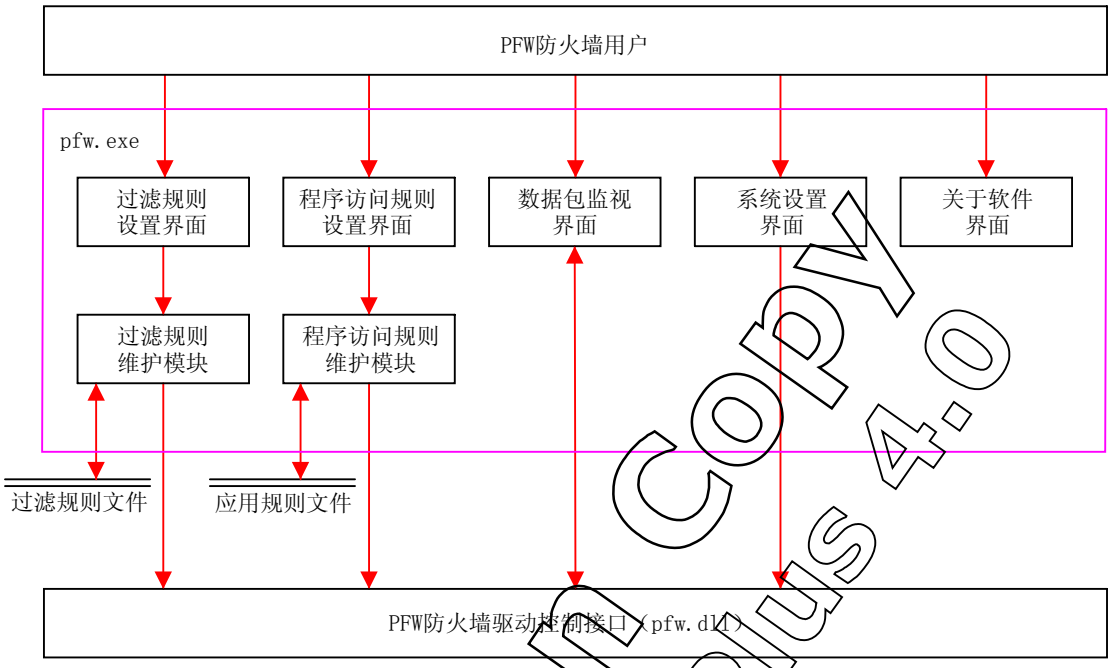


图 4.8 主界面结构

4.4.2 过滤规则设置

过滤规则设置界面为用户提供过滤规则的维护，再由过滤规则维护模块调用驱动控制接口的相应函数。过滤规则维护模块还负责读写过滤规则文件。

程序使用的类有：CFilterDlg（过滤规则设置界面类）、CFRuleInfoDlg（规则设置类）、CFilterRules（规则维护类）。详细代码参见附录三（FilterDlg.cpp、FRuleInfo.cpp、fltrule.cpp）。

4.4.3 程序访问规则设置

应用程序访问规则设置界面为用户提供程序规则的维护，再由程序规则维护模块调用驱动控制接口的相应函数。程序规则维护模块也负责读写程序访问规则文件。

程序使用的类有：CAppDlg（应用程序访问规则设置界面类）、CARuleInfoDlg（规则设置类）、CAppRules（规则维护类）。详细代码参见附录三（AppDlg.cpp、ARuleInfo.cpp、apprule.cpp）。

Evaluation Copy PDF Creator Plus 4.0

4.4.4 数据包监视界面

它的作用是为用户显示和保存防火墙日志记录。它也是属性页的子对话框窗口，类名是 CLogDlg。详细源代码参见附录三（LogDlg.cpp）。

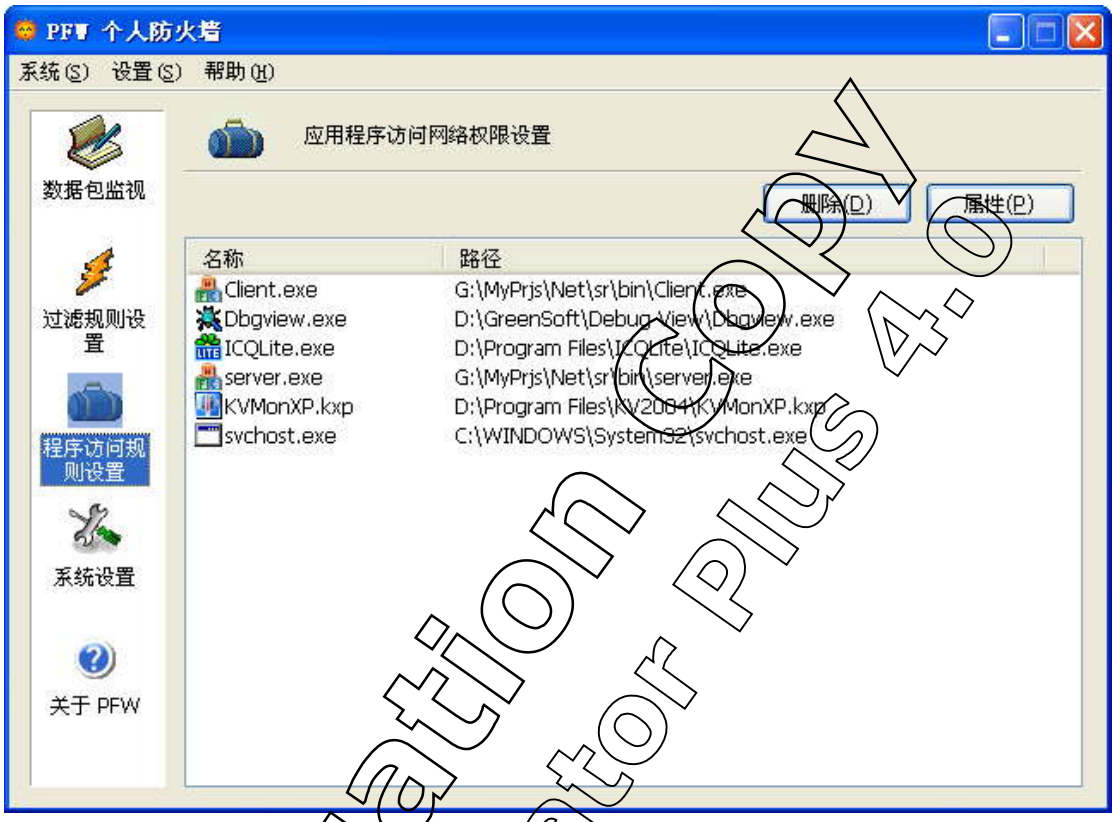


图 4.9 程序访问规则设置界面

4.4.5 系统设置

只提供了一些简单的设置功能：启动 Windows 时打开防火墙、运行时隐藏主窗口等辅助功能。

4.5 小结

PFW 个人防火墙是基于数据包过滤的个人防火墙，它具有的功能与目前常见的个人防火墙相差无几；通过优化设定安全规则，它能有效的减少黑客木马的攻击；它简单易用，具有良好的人机界面。附录四收录了 PFW 个人防火墙的过滤规则。很遗憾，由于诸多原因，本系统未能实现安全规则网上自动更新功能，只能由用户手工添加。

Evaluation Copy
PDF Creator Plus 4.0

本章未能把 PFW 个人防火墙实现的每一部分都写的很详细，这里只是将该软件系统的主要部分和突出该系统特点的内容呈现出来。附录一至三收录了 PFW 防火墙系统的源代码。

Evaluation Copy
PDF Creator Plus 4.0

总 结

毕业设计基本上完成了，现在回头看看，感触颇深。从一开始的盲目，到现在的清醒，我经历了认识上根本的变化。在做毕业设计之前，我虽有较好的 VC 基础，但只会使用防火墙，不了解它的原理，更不知道怎么实现它，心里也没底。我只有从头开始看书，逐渐地知道了它的实现技术，学会了网络驱动程序的开发技术，更是体会到了一种高效的学习方法。

1．学以致用，边学边做。只有在用的过程中才能遇到问题，在解决问题的过程中对所学知识有更深刻的理解。所以我现在学东西的时候会去查找很多材料，先把它们大致看一遍，知道什么地方讲什么内容并掌握最基本的结构或方法，然后就开始动手做，遇到问题了再去查资料，明白了再回来做，如此反复。这样会学得也快，做的也快，学做结合，相得益彰。

2．多问。一般来说，向高手请教是学习的捷径，互联网缩小了世界的距离，csdn.net 开发论坛、驱动开发网（driverdevelop.com）论坛为我提供了向高手学习、讨论的机会。当然不只是跟高手学习，同学之间相互讨论也是必不可少的，很多问题在争论之下，突然间豁然开朗。从与别人的交流中受益非浅。所以，珍惜每一个问和讨论的机会就显得很重要。

PFW 个人防火墙的开发也使我真正的了解了软件开发的基本流程，同时深刻地理解了软件工程的概念和软件系统的多层设计思想。

导师的指引、互联网丰富的资料、理论与实践相结合的学习方式、同学的帮助是我设计得以成功实现的保证。

Evaluation Copy
PDFCreator Plus 1.0

参考文献

[1] Microsoft,《Windows 2000 DDK Document》, Built on Wednesday, June 28, 2000。

[2] 朱雁辉，《Windows 防火墙与网络封包截获技术》，电子工业出版社，2002。

[3] 《csdn 开发高手》2004 年第 05 期，文章:《学习是快乐着——“N-Byte 网络守望者”的开发手记》，撰文/N-Byte 团队；文章:《.NET 框架中的驱动程序应用技术》，N-Byte 团队项目经理周翔。

[4] 王罡、林立志,《基于 Windows 的 TCP/IP 编程》,清华大学出版社,2000。

[5] Mikey Williams（美），《Windows 2000 编程技术内幕》，前导工作室译，机械工业出版社，1999。

[6] Al Williams（美），《MFC 技术内幕》（MFC Black Book），龚波、赵军锁、陈胜等译，机械工业出版社，1999。

[7] Scott Meyers,《Effective C++ Second Edition》, 1999。

Evaluation Copy
PDF Creator Plus 4.0

致 谢

在此谨向郭骏老师表示衷心感谢，感谢郭老师给我的指引和关怀，使对防火墙开发一无所知的我得以顺利地实现了本系统。

同时感谢曾给我帮助和支持的同学和网友。

Evaluation Copy
PDF Creator Plus 4.0