基于 Windows 2000 的防火墙设计*)

Windows 2000 Based Firewall Design

杨 琛 杨寿保

(中国科技大学计算机科学技术系 合肥 230026)

Abstract In this paper, we will firstly analyze the design and implementation of Windows Based Firewall. To improve the rule matching performance, we introduce a new, high efficient rule matching algorithm and present the Windows 2000 based firewall prototype.

Keywords Firewall, Packet filtering, Rule matching

1. 引言

随着网络技术的迅速发展和网络应用的急剧膨胀,网络安全日益成为人们关注的问题。防火墙是置于内部可信网络和外部不可信网络之间、作为一个阻塞点来监视和抛弃应用层的流量以及传输层和网络层的数据包的系统或系统组[1],用以达到对网络连接的安全性管理。

在本文中,我们将讨论一种基于 Windows 2000 新结构的防火墙设计,首先我们将介绍一般的防火墙结构和原理,针对这种结构的不足,引出我们设计的结构,并给出具体的设计实现原型和性能分析,最后给出结论。

2. 防火墙的一般原理

当用户连上 Internet,就可在用户和 Internet 之间插入一个或几个中介系统的控制关联,防止通过网络进行的攻击,并提供单一的安全和审计的安装控制点,这些中间系统就是防火墙。防火墙是在两个网络之间加强访问控制的一个装置,具体地说,防火墙是放在两个网络之间,能完成如下功能的一个系统或系统组:

- 所有从网内到网外(或者相反方向)的信息流,都必须经过它;
- 只有由本地安全策略所允许的信息流,才能穿过它;
- 系统本身不受信息穿越的影响。

从结构上分,我们可以将防火墙分成两类:①*包过滤型防火墙* 包过滤防火墙是根据定义好的过滤规则审查每个数据包并确定数据包是否与过滤规则匹配,从而决定数据包能否通过,它的特点是开销小,速度快;②*代理型防火墙* 这种防火墙主要使用代理技术来阻断内部网络和外部网络之间的直接通信,达到隐蔽内部网络的特性。

包过滤防火墙在结构上可以分为两层,即报文采集层和策略实施层,如图 1 所示。

报文采集层负责采集到达协议层前的所有数据包,而策略实施层根据先前定义的规则对由报文采集层搜集的数据报文进行处理,符合安全策略的数据报文通过;与安全策略违背的数据报文丢弃,以达到保护主机的目的。

目前的类 UNIX 操作系统对包过滤防火墙都有很好的支持,如 Free BSD 的 ipfw[2,3]、ipfilter[2],以及 linux 2.2 开始从 ipfw 派生的 ipchains[4],到现在的 linux 2.4 的 netfilter iptables[5],它们都提供了类似的功能部件,能够在内核中提供专门的程序接口,配合用户层的应用来定义报过滤规则,以实施一定的安全策略。

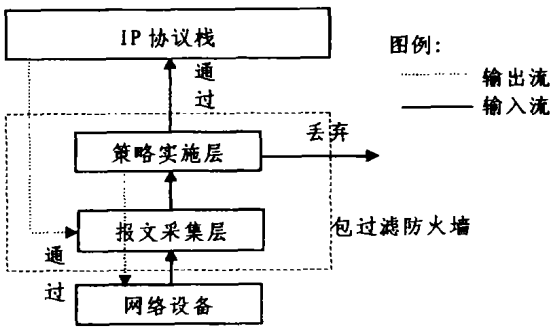

图 1 包过滤防火墙的分层

3. Windows 2000 的网络驱动结构

在 Windows 上一直欠缺类似的包过滤支持,Windows NT 系列虽然支持一定的安全策略,允许用户对连接到本机的 TCP/IP 网络流量类型进行控制:TCP 端口、UDP 端口、IP 协议[6]。但这种控制比较原始,远不能达到类 unix 系统中提供的对各种防火墙的支持:缺少对源地址、端口、目的地址的控制;缺少对系统中的网络接口的单个控制。从 Windows 2000 开始,系统提供了专门的包过滤应用程序接口(Packet Filtering API)[7],允许用户对每个 IP 适配器定义一个或多个过滤规则。Windows 2000 Server 系列提供了专门的控制台[8]来对各种规则进行管理,另外也提供了专门的针对 Windows 2000 Server 的 Internet Security and Acceleration 2000[9]。同时,Windows 2000 系列也提供了内核态的包过滤驱动接口[10],以补充在包过滤应用程序上的不足。我们的设计是基于内核态包过滤驱动接口的。

我们先简单介绍一下 Windows 2000 的网络驱动结构。Windows 2000 下的网络符合网络驱动接口规范 NDIS(Network Driver Interface Specification)[10],NDIS 在分层的驱动中定义了一些标准的接口,同时也提供了一些基本的例程,以简化网络驱动方面开发的负担。NDIS 支持以下三种类型的驱动:

1)小端口驱动:处于最底层,它负责直接和网络设备的交互。

2)中间层驱动:处于小端口和协议层之间,可以灵活地对

*)本文得到国家自然科学基金重大研究计划 90104030 项目的资助。杨 琛 硕士生,主要研究方向为计算机网络及应用。杨寿保 教授,博士生导师,主要研究方向为计算机网络及应用、信息安全与密码学。

来自上层协议层、下层来自小端口驱动的信息进行处理、控制，一般可以用来实现不同网络传输介质之间的数据报文的翻译、信息的加密传输、负载均衡等。

3）协议层驱动：一般的协议栈的实现，如通常的 TCP/IP、IPX 等协议。

它们的结构如图 2 所示。

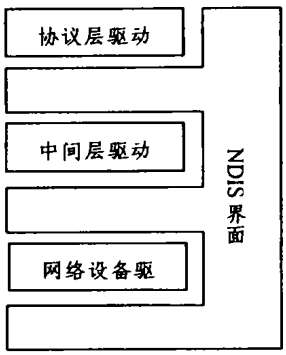

图 2　NDIS 驱动结构

在提出我们的设计前，我们先看一下目前基于 Windows 的防火墙的实现，主要的结构有以下几种：

1）*协议层驱动*　这种类型的实现在国内出现的时间比较早，大多数的实现都是基于 Windows9x 下面的协议层驱动 vpacket. vxd 的，或者是 packet. sys 的实现。就目前的个人防火墙的实现方面，瑞星个人防火墙[11]、sygate 的个人防火墙[13]的设计是基于协议层驱动实现的。

2）*中间层驱动*　这种类型的防火墙实现目前是绝大多数的，中间层驱动源于 Windows NT 中分层设计，允许系统在协议层驱动和小端口驱动之间存在人以多个中间层驱动，以完成所需要完成的工作。目前的产品中，我们可以看到蓝盾个人防火墙[12]就是基于中间层驱动实现的。

3）*垫片型驱动*　这种类型的防火墙实现是基于 Windows 9x 的，由于 Windows9x 的驱动程序的分层性不好，不能使用通常的类似于 Windows NT 系列中使用的中间层驱动。这一类的实现通常也被称为伪中间层驱动(Pseudo-Intermediate)。就目前的产品中，各种 Windows 9x 版本的个人防火墙都是基于这种驱动实现的。

我们比较一下这几种实现的特点：在三种实现中，协议层驱动出现得最早，流传也比较广泛。这种实现在系统现有协议的基础上增加了新的协议，可以实现对数据的采集和处理，但不能从根本上过滤系统现有协议栈的输入输出，需要使用额外的技术对 TCP/IP 协议部分的输入输出进行处理；中间层驱动由于能够对所有的输入输出进行处理，因此一般能够获得更大的灵活性；但中间层驱动有个很大的弊端，只是针对通常的网络设备(以太网络设备等)，对于拨号网络的输入/输出不能管理，并且中间层的引入，会在所有协议驱动的处理中引入中间层处理的开销；垫片型驱动的实现通常是挂接自己的 NDIS 实现环境中，如果需要额外处理的协议，在登记注册回调例程时加入自己的处理链，这种处理比较依赖系统，一般都需要在系统启动的时候确定顺序装入特定的驱动；如果错过了特定的装入顺序，系统将不能按照预定设计工作；同时，由于垫片型驱动挂接了 NDIS 原先的实现，也会在系统中的其他协议处理中引入开销。

基于上述的分析，我们可以看到，目前的实现各有利弊，因此我们提出一种新的设计。

4. 基于 Windows2000 的防火墙设计与实现

从通常的包过滤防火墙的结构我们可以看到，最清晰的设计是对 IP 协议栈的输入/输出做规则匹配处理。由于 Windows 的设计开放的材料较少，相关的开发材料较少，因此现有设计都是基于所开放的材料及其应用。

从 Windows 2000 开始，网络模块的设计中加入了对 filter-hook[10]的支持。Filter-hook 允许用户针对 IP 协议栈的输入输出做灵活的规则，主要是通过向系统的 IP 协议栈的输入输出中加入对相应数据报文的规则判断处理。处理的原型如下：

```
typedef PF_FORWARD_ACTION
(*PacketFilterExtensionPtr)(
  IN unsigned char*PacketHeader,
  IN unsigned char*Packet,
  IN unsigned int PacketLength,
  IN unsigned int RecvInterfaceIndex,
  IN unsigned int SendInterfaceIndex,
  IN IPAddr RecvLinkNextHop,
  IN IPAddr SendLinkNextHop
  );
```

其中，PF_FORWARD_ACTION 包括：PF_DROP、PF_PASS、PF_FORWARD，分别为丢弃、通过和转发。PF_PASS 的功能在于让报文经过该过滤驱动后能够配合 packet filter API 一起工作，比如可以与 Internet Security and Acceleration 中采取的处理互相配合使用；其他的各个参数分别可以用作对 IP 报文的 IP 头信息，TCP/UDP/ICMP/IGMP 头信息中的内容作判断、处理使用。一般的，不涉及对内容的过滤的话，这部分信息已经足够。

在设计中，我们参考了 Hollistech 公司的 IP Hook 的设计[15]。原型设计中，我们的工作主要是实现自己的规则处理例程，这种实现非常类似于 linux 2.2.x 系列中的防火墙支持部分[16]的工作。

如果单独地只让 Filter-Hook 在 PacketFilterExtensionPrt 类型的驱动中实现通常的规则，这样实现的难度小，但实用性差，必须在每一个需要定制规则的地方定制自己的驱动。为此，我们提出基于 Filter-Hook，实现一个灵活规则定制的通用型过滤防火墙。

规则实施层是整个防火墙系统的核心，影响防火墙系统的效率；而规则是整个防火墙中非常重要的部分，系统安全与否完全由系统制定的规则决定。规则的实施层是 PacketFilterExtensionPtr 型的例程，而规则的管理是由我们设计的部分完成。让这两个部分良好地协调工作，是我们主要的设计目标。

一般的防火墙驱动的效率取决于实施层的效率和规则库的规模，随着规则库规则的增多，一般的防火墙的性能会出现较大的变化。作为比较，我们可以参考一下 netfilter 中对 firewall 部分支持[17]的实现。Linux2.4 的 kernel 中对规则的匹配处理中，整个匹配是以链式结构组织和匹配的，因此对确定的某个报文的规则匹配如果失效，系统需要遍历整个规则链。在规则链很小时，这部分并不是很大的开销；但如果规则链规模很大，这部分的开销就不容忽视了。在 linux 系统现有的实现中，时间开销如下：

时间开销＝规则库规模＊匹配一条规则需要处理的时间

我们的实现能够在最差的情况下达到 ln(*规则库规模*)＊匹配规则需要的单位时间。

4.1　基本原理

现有的规则匹配中，基本的工作都是对数据报文中的源地址、目的地址与规则库中的源地址、目的地址的比较。在比较关系时只使用了一种关系：等于关系。等于的比较直接导致了规则匹配时间复杂度的线性增长，我们可以对这部分的比较进行改进。使用由地址本身携带的更多的性质来提高匹配的使用效率。

我们知道，在IPv4地址的构成中，IP地址是使用网络序的32位的整型，这种地址本身在构成上就有通常的数字性质，只是我们在通常的比较中忽略了这种基于大小的比较。我们的基本方法是，将规则链按照某种关系进行排序，然后规则匹配的时候使用这种关系进行二分查找，相对于线性查找，在规则库规模较大时，可以较大地提高匹配效率。

4.2 算法

从前面的描述我们可以看到，对规则链按照某种关系进行排序，然后在比较时，不是单纯地检查是相等就可以达到我们的要求。简单来说，我们可以按照源地址的大小对规则链进行排序。如图3所示。

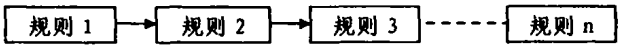

图3 规则链的组织

与通常的组织不同的是，在规则链中的元素除了通常的指向下一个规则元素的指针外，还提供了指向中间规则元素的指针，这个指针域用于在规则匹配中的下一条规则的选取。

```
struct rule_chain{
struct ip_fw_rule rule;
struct rule_chain * next;
struct rule_chain * mid;
……
}
```

具体的匹配算法如下：

```
match_routine (struct iphdr * ip,struct rule_chain * head,struct rule_chain * tail)
{
    struct rule_chain * phead=head, * ptail=tail;
    int result;
    while(phead! =ptail){
    result=(ip->saddr&p->rule. net_mask)-p->rule. saddr;
    if(! result) return MATCH;
    if (result>0) {
              result = ip-> saddr&ptail-> rule. netmask )-ptail->
              rule. saddr;
              if(!result) return MATCH;
              if(result<0) ptail=phead->mid; else return phead=
              phead->mid;
        }
    else
        return NOT_MATCH;
        }
}
```

4.3 分析

在4.2提到的算法中，我们使用了IP地址中的数字特性，以达到较高的效率。如果规则链的长度为n，并且组织中源地址的分布比较均衡的话，则由二分查找算法的时间复杂度分析，我们可以得到，查找算法的最坏复杂度为log(n)。对于单个源地址/目的地址的多规则情况，在上述的算法中没有处理，按照与源地址组织类似的方法，很容易将数字特性应用到规则的其他字段而对这部分的性能进行改善。

4.4 具体实现

我们提出的结构，得益于Windows 2000的Filter-Hook支持，我们主要实现策略实施层。另外，在具体实现中，还需要一个设备对象，以便通过专门的应用程序界面对规则进行设置和组合。

Filter-Hook的实现主要负责对规则的匹配以及对报文采取必要的动作；而设备对象主要是用来和用户层交互，提供的功能类似于netfilter的iptables。Windows系列提供了友好的用户界面，可以为用户提供更好的交互性。具体的结构如图4所示。

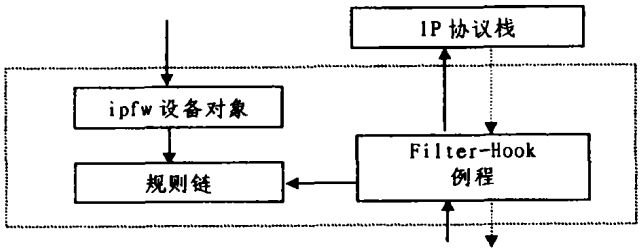

图4 基于Filter-Hook的防火墙结构

其中，ipfw设备对象负责管理规则链，在相应的添加删除规则后对规则链进行排序，以提供给Filter-Hook例程高效使用。而Filter-Hook例程一方面接收从网络硬件设备接收的待送给协议层处理的报文，另一方面接收从协议层发送的报文，进行安全策略的匹配后，采取相应的动作。

5. 性能分析

基于Filter-Hook的处理中，我们可以看到，程序的结构大大简化，极大地减轻了开发的负担，减少了代码出错的可能。另外，与传统的基于NDIS驱动的设计相比较，由于在整个系统的处理中，只是针对IP协议栈的，并不影响系统中的其他协议部分，这种设计引起的系统开销小。

由于采用良好结构的规则匹配链，我们的设计可以在较大规模的规则链时达到更好的性能。

结束语 在Windows 2000本身提供的扩展结构基础上，结合我们的高效的规则匹配引擎，我们可以设计出高性能的防火墙，这不仅在研究上具有重要的意义，对于防火墙在实际应用方面也有着很强的经济意义和现实意义。

参 考 文 献

1 (美)Hare C, Siyan K. Internet 防火墙与网络安全. 北京：机械工业出版社，1998.5
2 Lavigne D, BSD Firewalls. http://www.onlamp.com/pub/a/bsd/2001/04/25/FreeBSD_Basics.html
3 Palmer G, Nas A. FreeBSD Handbook. http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls.html
4 Russell R. Linux IP Firewalling Chains. http://netfilter.samba.org/ipchains/
5 Netfilter Core Team. The Netfilter Project: Packet Mangling for Linux 2.4. http://netfilter.samba.org/
6 网络属性. Microsoft Windows 2000 Professional
7 Microsoft, Microsoft Develop Network, 2001,07
8 Microsoft, Microsoft Windows 2000 Server Documentation. http://www.microsoft.com/windows2000/en/server/help/default.asp?url=/windows2000/en/server/help/sag_RRAS-Ch2_10.htm
9 Microsoft, ISA Server SDK Documentation, 2001, 05. http://www.microsoft.com/isaserver/techinfo/productdoc/2000/SDKdownload.asp
10 Microsoft, Microsoft Device Driver Kit Document, 2000,08
11 瑞星公司主页. http://www.rising.com.cn/
12 蓝盾公司主页. http://www.bluedon.com/
13 Sygate Personal Firewall. http://www.sygate.ca/free/ssdesktop.htm
14 NDIS Pseudo-Intermediate (PIM) Driver for Windows 9X and Windows Millennium, Page of PCAUSA. http://www.pcausa.com/ndispim9x/Default.htm
15 IP Hook Driver Sample. http://www.hollistech.com/Resources/IpHook/formresult3.htm
16 Bonn D. firewall.c of Linux Source Code 2.2.18 ,http://www.kernel.org/
17 Linux Source Code V2.4.12. http://www.kernel.org/