文 章 编 号: 1000-1220(2001)12-1516-03

基于Linux 内核防火墙 Netfilter 的安全应用的设计方法[1]

王宏健[1]　邵佩英[1]　张　籍[2]

[1](中国科技大学 研究生院 计算机学部 北京 100039)
[2](北京工业大学 计算机学院 北京 100080)

摘　要: 本文在分析 netfilter 处理数据包过程的基础上, 给出了在系统空间和在用户空间开发基于Linux 内核防火墙 netfilter 安全应用的设计方法

关 键 词: 防火墙; 安全应用; 钩子函数
中图分类号: TP393　　**文献标识码**: B

1　前　言

Linux2.2 内核防火墙 ipchains 提供了基本的防火墙功能, 如包过滤, 地址伪装, 透明代理等, 目前在安全应用的设计中得到了广泛的应用. 但 ipchains 也存在一些问题, 主要有:

1. Ipchains 以内核级的 C 和 C++ 代码编写, 目前从用户空间调用 ipchains 函数还没有好的接口. 这样用户要开发适合自己需要的安全系统时, 不能使用常用的编程语言如 Java 或 Perl 等.

2. Ipchains 和 NAT (网络地址转换) 是作为两个单独的 Linux 项目开发的, 两者配合使用比较复杂.

3. Ipchains 没有采用开放的结构, 用户不能直接扩展 ipchains 的功能. 如果要实现一些高级的功能如: 为每个用户产生过滤规则和基于 MAC 地址的过滤是不可能的.

4. 随着过滤规则的繁杂和数目增多, 效率问题也逐渐明显.

为了解决这些问题, 为用户提供一个开放的、扩展性好的内核防火墙, 在未来的 Linux 2.4 内核中, 包含了被称为 netfilter 的内核防火墙. 该防火墙项目由 Rusty Russell 提出, 采用更好的框架结构 (framework) 重新构造, 并可实现许多新功能, 如完整的动态 NAT, 基于用户及 MAC 地址的过滤, 真正的基于状态的过滤, 包速率限制等[1]. 为了更好地利用 netfilter 提供的强大功能, 本文在分析 netfilter 处理数据包过程的基础上, 给出了两种基于 netfilter 安全应用的设计方法.

2　内核防火墙 netfilter 处理数据包的过程

Linux 的网络子系统基于 BSD 套接字接口, 在利用 Linux 系统调用建立新的套接字时, 需要传递套接字的地址标识符、套接字类型及协议. 内核将为该套接字分配新的 socket 数据结构, socket 数据结构实际是虚拟文件系统 (VFS) 索引结点 (inode) 数据结构的一部分, 分配新的 socket 数据结构实际就是分配新的 VFS 索引结点, 因而可以象操作普通文件一样操作套接字. 当两台主机建立好套接字, 发送方用一条 C 语句: write (socket, data, length) 将一长度为 length 的数据 data 写入套接字 socket. 接收方通过一条 C 语句: read (socket, data, length) 从套接字中读数据. 数据包在 Linux 网络子系统协议栈传输过程如图 1 所示:

Netfilter 作为中间件在协议栈中提供了一些钩子函数 (Hooks), 用户可以利用钩子函数插入自己的程序, 扩展所需的功能. 目前, 基于 IPv4、IPv6 和 IPX 的 netfilter 钩子函数都已开发完成. 我们在这里仅以 IPv4 为例加以说明, 其它的与此类似. 图 2 说明了 IPv4 的五个钩子函数的放置位置, 函数定义在内核头文件 linux/netfilter_ipv4.h 中. 当一数据包进入 netfilter 框架, 首先进入 Hook1 (NF_IP_PRE_ROUTING), 进行目的地址转换; 然后经路由, 将此数据包发往本机进程或转发至其它主机; 在发往本机进程之前, 数据包将进入 Hook2 (NF_IP_LOCAL_IN), 进行 INPUT 包过滤; 如果转发至其它主机, 数据包将先进入 Hook3 (NF_IP_FORWARD), 进行 FORWARD 包过滤, 然后进入 Hook4 (NF_IP_POST_ROUTING), 进行源地址转换, 最后发出本机; 如果数据包由本地进程产生, 将先进入 Hook5 (NF_IP_LOCAL_OUT), 进行 OUTPUT 包过滤, 然后经过路由进入 Hook4. 这五个钩子函数运行结束, 将返回以下 4 个动作之一:

1. NF_DROP: 丢弃此数据包;
2. NF_ACCEPT: 允许此数据包继续传递;
3. NF_STOLEN: 钩子函数接收此数据包, 就好象此数据包已到达目的地;
4. NF_QUEUE: 将此数据包放入一特定队列, 可以在用户空间进行处理.

3　基于 netfilter 安全应用的设计方法

3.1　在系统空间编写内核级的 netfilter 扩展模块

编译 Linux 内核时, CONFIG_NETFILTER 选项应设为 Y. 我们看到在 IP 层代码中, 有一些带有 NF_HOOK 宏的语句, 如在 net/ipv4/ip_input.c 中, 有 NF_HOOK (PF_INET, NF_IP_LOCAL_IN, skb, skb_dev, NULL, ip_local_deliver_

收稿日期: 2001-04-11　基金项目: 中国科学院知识创新工程项目"黑客入侵防范体系"项目的支持　作者简介: 王宏健, 硕士研究生, 研究方向为网络安全. 邵佩英, 教授. 张　籍, 硕士研究生.

finish). NF_HOOK 对给定的协议(PF_INET)调用注册的钩子函数(NF_IP_LOCAL_IN)处理数据包(skb), 可以对进入本机和本机发出的数据包分别处理, 这里进入设备为 skb_dev, 发出设备为NULL, 表示仅处理进入的数据包. 如果钩子函数返回NF_ACCEPT, 最后一个参数 ip_local_deliver_finish 所指的函数将被调用, 继续传递数据包; 如果返回的是NF_DROP, 数据包将被丢弃.

如果只使用缺省的Hook函数, 并不能发挥框架结构的完全功能, 我们还要知道如何注册和编写钩子函数来侦听、修改数据包. 同时要给每一个钩子函数赋予一个优先级, 优先级低的将先被执行. 下面先看一段例子程序my.c, 这是一个可动态插入的内核模块, 其功能是丢弃所有由本机发出的长度为 100 的数据包

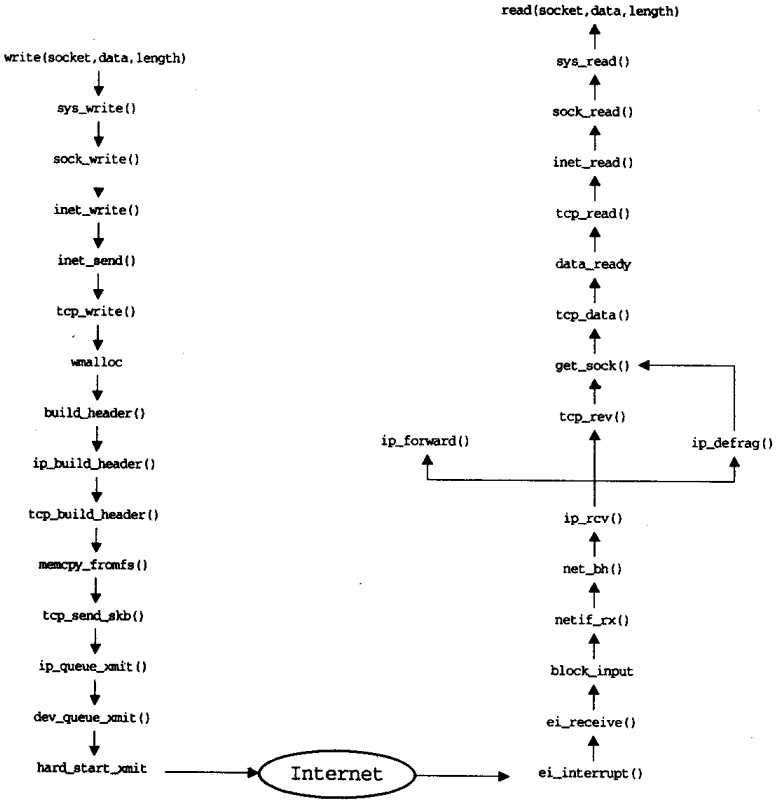

图 1　数据包在Linux 系统协议栈的传输过程

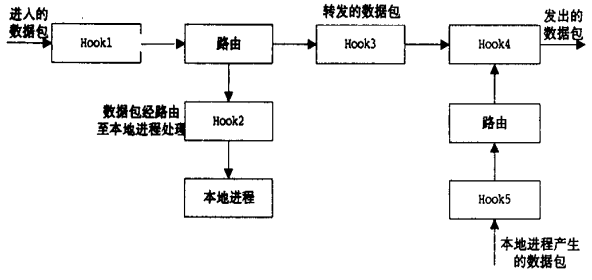

图 2　IPv4 中 netfilter 的五个钩子函数的放置位置

```
# include < linux/config. h>
# include < linux/module. h>
# include < linux/ip. h>
# include < linux/netfilter_ipv4. h>
static unsigned int my_hook(unsigned int hook, struct sk_buff
                                                    * * skb,
const struct net_device * in, const struct net_device * out,
int (* okfn) (struct sk_buff * ))
{
    unsigned char * data= (void * ) (* skb)- > nh. iph + (* skb)
- > nh. iph- > ihl* 4;
    (* skb)- > nfcache |= NFC_UNKNOWN;
    if ((* skb)- > len= = 100)
    {
      printk("my_hook: dropping packet! \n"); //Log the event
      return NF_DROP;
    }
    return NF_ACCEPT;
    }
    static struct nf_hook_ops my_ops
    = {{NULL, NULL}, my_hook, PF_INET, NF_IP_LOCAL_
                                    OUT, NF_IP_PRI_FILTER- 1};
    int init_module(void)
    { return nf_register_hook(&my_ops); }
    void cleanup_module(void)
    { nf_unregister_hook(&my_ops); }
```

我们看到, 注册一个钩子函数要说明一个 nf_hook_ops 结构 my_ops, my_ops 第一项{NULL, NULL}是一内部使用的双向链表; 第二项是执行我们要完成的功能的 my_hook 函数; 然后是 PF_INET 协议; NF_IP_LOCAL_OUT 是钩子函数的放置位置, 这里我们仅处理由本机发出的数据包; 最后是优先级, 表明 my_hook 在包过滤之前执行. my_hook 函数共有 5 个参数, 第一个是 Hook, 在这里为 NF_IP_LOCAL_OUT; 第二个是 sk_buff 指针的指针, 指示数据包, 这样可以

在需要的时候对整个数据包进行替换; 第三个参数指向输入设备的指针, 本例中为NULL; 第四个参数指向输出设备的指针, 本例中为NF_IP_LOCAL_OUT 钩子函数发出数据包的接口; 最后一个参数也为一指针, 如果钩子函数成功调用, 将执行该指针指向的函数, 除非特别需要, 一般不直接调用该函数. 最后用 nf_register_hook 注册钩子函数.

3.2 在用户空间编程使用 netfilter

为了不影响网络速度, 对于流量大的网关我们不推荐在用户空间处理数据包. 但对流量较小的点(特别是较小规模的网络或单机), 在用户空间进行数据包控制是非常有用的.

先用以下Linux 命令生成一个虚拟设备(文件):

```
mknod /dev/netfilter_ipv4 c 120 2
```

该设备的缺省动作是在系统中注册 netfilter 钩子函数, 用户可以通过该设备读取所有通过 netfilter 框架的数据包, 先取得文件操作符(这里我们仍使用 c 语言, 当然你也可以使用 Java 等):

```
int fd= open("/dev/netfilter_ipv4",O_RDONLY);
```

然后就象对文件读取一样, 对文件操作符进行读取操作.

如果要对通过指定钩子函数的数据包进行处理, 可以通过 ioctl() 系统调用来完成. 通过说明一个 nfdev_condition 结构, 用户可以增加或删除一个条件, 通过说明一个 nfdev_verdict 结构, 用户可以对数据包进行接收(NF_ACCEPT)、丢弃(NF_DROP)或放入队列(NF_QUEUE)操作. 这两个结构在头文件 netfilter_device.h 中定义, 限于篇幅这里就不详细说明了.

3.3 测试步骤

步骤 1. 运行以下命令发一长度为 100 的数据包, -c 表示数据包的个数, -s72 表示数据包总长度为 100(72 字节加上 IP 头 20 字节, 再加上 ICMP 头 8 字节).

```
# ping -c1 -s72 1.2.3.4
PING 1.2.3.4 (1.2.3.4) from 5.6.7.8: 72(100) bytes of
                                                    data.
80 bytes from 666 (1.2.3.4): icmp_seq= 0 ttl= 255 time= 1.5
                                                    ms


--- 1.2.3.4 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.5/1.5/1.5 ms
```

我们看到正确收到响应.

步骤 2. 编译例子程序 my.c 为 my.o, 无误后, 在 Linux 下以超级用户的身份运行以下命令, 插入 my.o 模块.

```
# insmod ./my.o
```

重复步骤 1:

```
# ping -c1 -s72 1.2.3.4
PING 1.2.3.4 (1.2.3.4) from 5.6.7.8: 72(100) bytes of
                                                    data.
ping: sendto: Operation not permitted
ping: wrote 1.2.3.4 80 chars, ret= -1


--- 1.2.3.4 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
```

数据包不能发出, 从系统日志文件 dmesg 中我们看到:

```
# dmesg -c
linuxmag: dropping packet
```

我们在网关机器上运行 tcpdump 侦听程序, 也截获不到数据包.

步骤 3. 发长度为 84 的数据包:

```
# ping -c1 1.2.3.4
PING 1.2.3.4 (1.2.3.4) from 5.6.7.8: 56(84) bytes of data.
64 bytes from 666 (1.2.3.4): icmp_seq= 0 ttl= 255 time= 1.5
                                                    ms


--- 1.2.3.4 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.5/1.5/1.5 ms
```

正确收到响应, 说明 my.o 正确完成功能.

4 小 结

Netfilter 新一代内核防火墙, 功能强大且提供源码. 通过对其分析, 对于编写我们特殊的基于 netfilter 安全应用具有重要意义.

参 考 文 献

1 Rusty Russell. Linux 2.4 packet filtering HOWTO [EB/OL]. mailing list netfilter@lists.samba.org. v1.0.1 Mon May 1 18:09:31 CST 2000

2 Rusty Russell. Linux netfilter hacking HOWTO [EB/OL]. mailing list netfilter@lists.samba.org. v1.0.1 Sat Jul 1 18:24:41 EST 2000

3 Rusty Russell. Writing a module for netfilter [EB/OL]. Linux Magazine June 2000 http://www.linux-mag.com/2000-06/gear_01-03.html

4 Dave Wreski. Linux kernel 2.4 firewalling matures: netfilter [EB/OL]. Feb. 2001

DESIGN OF SECURITY APPLICATION BASED ON LINUX KERNEL FIREWALL NETFILTER

WANG Hong-jian SHAO Pei-ying ZHANG Ji

[1](*Department of Computer Science of Graduate School of University of Science and Technology of China, Beijing* 100039, *China*)
[2](*Computer Institute of Beijing Polytechnic University, Beijing* 100080, *China*)

Abstract This paper analyzed how Netfilter deals with data packet in the Linux networking subsystem. At the end, we give the method to design security application based on Linux kernel firewall netfilter both in system-space and user-space.

Key words Firewall; Security application; Hook mechanism