基于 Netfilter 的数据包捕获技术研究*)

李惠娟[1] 王汝传[1,2] 任勋益[1]
(南京邮电大学计算机科学与技术系 南京 210003)[1]
(南京大学计算机软件新技术国家重点实验室 南京 210093)[2]

摘 要 在 Linux 下通常的数据包捕获系统,通过 Libpcab 函数框架实现,而在该体系下实现的包捕获存在着一些缺陷。在 Linux2.4 版本后,Linux 使用了 Netfilter 框架,便于用户构建自己的防火墙。在该框架下,通过注册钩子函数,可以轻松实现数据包的捕获。本文通过研究 Linux2.4 后版本内核中的网络框架 Netfilter,给出了在该框架下对数据包进行捕获的设计方案,并集中解决了数据信息提取和使用 Netlink 实现内核与用户态通信的关键问题。

关键词 Linux,Netfilter,数据包捕获,Netlink

Design of Package Capture Based on Netfilter in Linux

LI Hui-Juan[1] WANG Ru-Chuan[1,2] REN Xun-Yi[1]
(School of Computer Technology, Nanjing Post and Communication University, Nanjing 210003)[1]
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)

Abstract In Linux, we often use Libpcap function to trap package, but Libpcab has proved to has low effect when the net has a heavy traffic whit small size packets. In this paper, we design the packet capture based on Netfilter frame in Linux, which is the new net frame of Linux 2.4 and 2.6. By registering hook fuction to the Linux kernel, we can easily achieve packet capture. We propose the design and achieve method of trapping package based on Netfilter, and resolve the kernel problem: get information of package and the communication between kernel space running process and user space running process by Netlink socket.

Keywords Linux, Netfilter, Package capture, Netlink

1 引言

在 Linux 下的数据包捕获常用的实现方法是通过调用 Libpcab 库函数,通过对协议栈的数据包进行拷贝,包捕获程序运行在用户态下。在 Linux 内核 2.4 和内核 2.6 版本中,提供了 Netfilter/Iptable 框架,在这一框架上,可以通过注册回调函数实现多种功能,包括数据包过滤,网络地址转变(NAT),实现路由器优化等。而且,通过 Netfilter 框架注册钩子函数可以轻松实现数据包捕获,同时由于是挂载在协议栈处理程序中可以增强对数据包的控制能力,如控制数据包的下一步流向等。本文对基于 Netfilter 框架下的数据包捕获进行研究,给出了设计方案和实现,并解决了数据包信息提取和用户态与内核态进程间通信的关键问题。

2 Netfilter 框架

Netfilter 是在内核内的一系列钩子函数的集合,它提供了接口,用户或者开发人员将函数作为接口参数中的回调函数,将设计的模块挂接在钩子函数上,从而使函数作为内核函数运行在 Linux 系统中[1]。本文中对于数据包的捕获是基于 Netifilter 的钩子函数 1 实现的。

在 Linux 中嵌入程序都是以模块的方式,通过内核挂接命令将模块嵌入内核,Netifilter 框架如图 1 所示:

```
1 init()初始化模块
完成定义钩子函数和注册模块的功能

2 unsigned int rec_hook(unsigned int hooknum, struct sk_buff **skb, const struct net_device *in, const struct net_device *out, int (*okfn)(struct sk_buff *)
钩子函数原型定义

3 void cleanup_module(void)
注销函数,完成清除钩子函数和注销模块的功能
```

图 1 Netfilter 函数调用框架

函数说明:

(1)init( )这是模块的初始化函数,完成对 hooker 函数的参数指定

(2)unsigned int rec_hook(unsigned int hooknum, struct sk_buff * * skb, const struct net_device * in, const struct net_device * out, int (* okfn)(struct sk_buff *)钩子函数定义,可以在 Linux 内核代码中:/include/linux/netfilter.h 中看到

*)本课题得到国家自然科学基金(60573141 和 70271050)、江苏省自然科学基金(BK2005146)、江苏省高技术研究计划(BG2004004、BG2005037、BG2006001)、国家高科技 863 项目(2005AA775050)、南京市高科技项目(2006 软资 105)、现代通信国家重点实验室基金(9140C1101010603)、江苏省计算机信息处理技术重点实验室基金(kjs050001、kjs06)资助和江苏省高校自然科学研究计划(05KJB520092)资助。**李惠娟** 硕士研究生,主要研究方向为计算机网络、网格计算、信息安全和虚拟现实技术;**王汝传** 教授,博士生导师,主要研究方向是计算机软件、计算机网络和网格、信息安全、无线传感器网络、移动代理和虚拟现实技术等;**任勋益** 博士研究生,主要研究方向为计算机网络和网格计算、移动代理和信息安全技术等。

函数的声明以及相关变量定义。

(3) void cleanup module(void),模块注销函数,完成对hooker函数的注销。

通过这三个函数,就搭建起了数据接收的框架,通过TCP/IP协议栈的数据流将降调用钩子函数,这样通过钩子函数可以对数据流进行捕获当然也可以过滤。

3 基于Netfilter的数据包捕获设计

通过Netfilter框架的理解,设计准备在钩子一处挂在钩子函数,对IP层进行包捕获,这样到达协议栈的数据包在到达IP处理进程前,会先经过钩子函数。通过钩子函数,对数据包信息进行提取,并可以规定数据包下一步流向,在我们设计框架下,主要目的是为了获取数据包信息,并不实现过滤或者拦截功能,为了不破坏正常的通信,提取信息后的数据包后都执行NF_ACCEPT宏操作,继续正常传输数据报。

下面是通过Netfilter框架对数据包捕获的设计结构,主要分为两个模块:数据包信息提取模块和内核用户态通信接口模块。

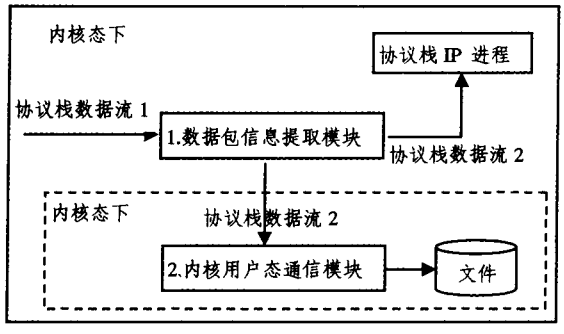

图2 数据包捕获设计

捕获包的流程:在控制台挂载数据包信息提取模块,模块首先完成对模块注册和钩子函数的注册,然后处于等待状态。在启动内核用户态通信接口模块后,首先建立内核与用户态通信接口,启动数据包信息提取模块的包捕获功能,到达节点的数据流经过协议栈到达IP_input处理进程之前会先调用数据包信息提取模块中的钩子函数,钩子函数根据规定的规则对每一个到达的数据包提取包的大小和IP包头,并将这一个内容作为单播消息向netlink套接字发送一条信息,并对每个数据包的下一步流向进行规定。内核用户态通信接口收到内核态传出的消息后,创建一个该数据结构类型的对象,并添加写入文件。数据包捕获设计中关键是数据包信息的提取和内核态与用户态进程通信。

3.1 数据包特征的提取

数据包的提取中,我们关心的往往是数据包的某些特征,比如数据包的大小,数据包中上层协议类型,在Netfilter框架中,它提供了一个统一的特征数据结构:sk_buff,在Linux内核中/include/linux/目录下的skbuff.h文件可以看到skbuff的数据结构定义[5]:

```
struct sk_buff{
struct sk_buff *next;
struct sk_buff *pre;
struct sk_buff_head *list;
struct sock *sk;
struct timecal stamp;
struct net_device #dev;
struct net_device *real_dev;
……
}
```

以上的数据结构说明这是一个代表头的链表数据结构,通过建立套接字可以对数据结构内的数据信息进行访问。省略的部分包括一些联合体数据,对每一层的数据信息进行接口封装。包括:链路层,IP层,传输层。另外sk_buff提供了对优先权的一些控制信息,可以保存自定义的数据,当然数据的类型是规定了的。并需要通过skb_clone( )函数进行链接控制。

在本设计中提取的是两个数据信息:数据包大小和包头的大小,并定义了数据结构packet_info这个结构很简单,就封装了一个IP头数据结构:

```
struct packet_info
{
struct iphdr *iph;
};
```

通过这一数据结构,我们可以在IP层获取数据包的所有的可能需要的信息,并可以对数据包进行优先权设置。将提取到的所关心的数据包信息,保存在自定义的数据结构中。通过向Netlink套接字发送单播信息,就可以将数据包信息传入用户空间程序了。

3.2 内核态与用户态进程通信

进程间通信的方式有很多:管道,共享内存区,信号量等等。但是在内核态和用户态之间,这些方式不能使用。内核态与用户态进程通信方式有三种[2]:

(1)通过Linux提供的copy_from_user()和copy_to_user()这一对函数实现数据通信,但这两个函数会引起阻塞,不能用在软硬中断中,这样效率较低,还需要编写系统调用函数。

(2)通过软硬中断中的自旋锁作为中断的同步机制,通过自旋锁来实现中断环境与中断环境,中断环境与内核线程的同步,这样便可以在内核线程中使用套接字或消息队列来取得用户空间的数据,然后再将数据通过临界区传递给中断过程。

(3)使用Linux的netlink套接字。这是在2.4以后的版本中,主要使用的内核与用户空间的通信方式。

在本系统中的通信方式使用的就是Netilink套接字[2,3]。在Linux内核目录:linux/net/netlink/netlink.c下可以找到有关函数的定义,这也是作为一个模块挂接在内核程序中。在Linux系统文件目录:proc/net/netlink目录系统下存储有关的文件信息。

在Netfilter中,在用户态进程和内核态进程都需要建立一个netilik套接字,通过pid号发送消息。过程如下图所示:

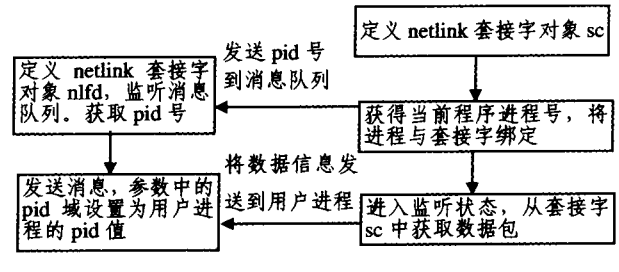

图3 netlink通信方式

图3是在本设计中用户态下和内核太下运行程序通过netlink套接字进通信过程。用户态下的程序建立套接字sc,并与程序进程绑定。像套接字中发送进程号pid。挂载在内核中的模块运行后,首先创建netlink对象,进入监听,检查消息队列直到收到进程号,赋值给参数dst_pid,在发送函数中需要以此作为参数将消息传给用户态下的进程。这样就实现

了通信的建立。

4 基于 Netfilter 的数据包捕获实现

按照图的设计，将主要由两个程序模块实现。自定义的数据结构保存在头文件中。

数据包捕获 packcap 模块，这一模块是挂载在内核下的模块。完成的功能包括：模块的维护(挂载和清除)，注册钩子函数，定义 Netlink 套接字对象，提取并发送数据包信息。下面是模块主要核心代码。主要实现截获报文，提取报文头信息，并通过 netlink 套接字将 IP 报文头信息发送到用户进程。

```
unsigned int rec_hook(unsigned int hooknum, struct sk_buff * *
        skb, const struct net_device * in, const struct net_device *
        out, int ( * okfn)(struct sk_buff * ))
{ ……
  struct nlmsghdr * nlh;
  char * data;
  int ret;
  int size= NLMSG_SPACE(ntohs(sb->nh. iph->tot_len)); //获
    得 netlink 消息队列空间
  struct sk_buff * sk_copy=alloc_skb(size, GFP_ATOMIC); // 获
    得 skb 空间
  if(! sk_copy)goto nlmsg_failure; //如果失败则转到 netlink 消息
    失败宏处理
  unsigned char * old_tail; //定义字符型指针
  old_tail = sk_copy->tail; //将指针指向获取的 skb 空间
  nlh = NLMSG_PUT(sk_copy, 0, 0, CAP_MSG, size-sizeof
  (struct nlmsghdr)); //将数据消息发送到 netlink 消息空间
  data=NLMSG_DATA(nlh); //封装数据，计算包括 netlink 消息头
    后的总长度
  memcpy(data, sb->nh. iph, ntohs(sb->nh. iph->tot_len)); //拷
    贝 IP 包头到 netlink 消息
  nlh->nlmsg_len = sk_copy->tail - old_tail; //计算消息长度
  NETLINK_CB(sk_copy). dst_groups = 0; //获取 sk_copy 的控制
    信息，并将其中的 det_group 参数设置为 0
  ret = netlink_unicast(nlfd, sk_copy, pid, MSG_DONTWAIT);
    //将数据包头信息发送到用户进程，通过用户进程号 pid 识别，
    将返回值作为发送成功标志
  if(ret<0)printk("errorcode=%d\n",ret); //如果不成功，则报错
  return NF_ACCEPT; //对数据包发送给协议栈做下一步处理
  nlmsg_failure: //内核自定义宏，处理 netlink 出错
  kfree_skb(sk_copy); //释放 skb 空间
}
```

内核和用户态通信接口模块 capture，这也是主程序，实现在用户态下创建 netlink 套接字对象实现进程通信接口，将数据包信息写入文件，并在控制台上显示每个数据包的大小和 IP 报头长度。为了使用户能够选择退出程序，考虑使用一个线程实现 netlink 套接字通信和数据包的信息写入。下面程序是在用户态下获取包信息的线程：cap_method 的核心代码。主要作为一个通信接口获取 netlink 消息，并将其写入文件中。

```
void * cap_method(void * arg)
{struct iphdr * phead;
 while(end)
{ char buf[2000]="\0";
  int count=recvfrom(sc, buf, 2000, 0, (struct sockaddr * ) &kpeer,
  &kpeerlen); //从套接字中获取消息，即数据报头信息
  phead=(struct iphdr * )(buf+sizeof(struct nlmsghdr)); //将信
    息保存到自定义数据 phead 中。
  int total_len=ntohs(phead->tot_len); //数据报头长度
  printf("count=%d,length=%d\n", count, total_len); //显示 IP
    数据报文长度和 IP 包数据部分的长度。
  packet_count++; //数据报文数目的总的统计量
  //以下操作实现讲相关信息写入文件。
  lseek(fd,0,SEEK_SET);
  write(fd,&packet_count,sizeof(int));
  lseek(fd,0,SEEK_END);
  time_t t=time(NULL);
  printf(t);
  write(fd,&t,sizeof(t));
  write(fd,&total_len,sizeof(total_len));
  write(fd,phead,total_len);
  }
  return Null;}
```

5 实验结果

在南京邮电大学主楼 903 教研室局域网内，已经使用所设计的数据包捕获程序网内的数据包进行捕获。提取的信息是每个 IP 报文的长度和 IP 数据部分长度。实验结果如图 4。

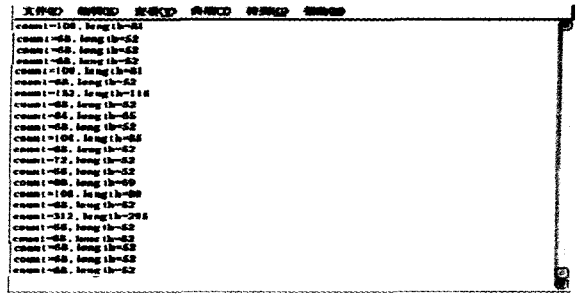

图 4 运行结果

包捕获系统的性能，主要受三方面影响：1)内存读写；2)内核态到用户态的拷贝；3)网卡中断。使用普通的包捕获平台如 libpcab，会有多次无用的内存读写，数据包从内核态到用户太的拷贝是通过系统调用来实现的。当每次拷贝字节很小的时候，每拷贝一次就需要通过调用系统函数获得拷贝，这样 cpu 效率非常低，当数据帧很小时，就已经会产生很大的丢包率了下图是华中科技大网络中心实验室得到的 libcap 的一个性能文件。

表 1 libpcap 平台下包捕获性能

背景数据	背景流量				
包帧长	0Mbps	25Mbps	50Mbps	75Mbps	100Mbps
1512Byte	100%	100%	100%	98%	80%
512Byte	100%	100%	100%	53%	17%
80Byte	100%	70%	35%	3%	0%

可见，在流量不是很大的时候，只要数据报帧长较小的情况下就会产生很大丢包率。而使用 Netfilter 框架，包的拷贝不通过系统调用，而使用 netlink 套接字，netlink 队列文件数据系统文件，所以数据的传输是通过内存→内存，对数据包的大小没有限制，同时在我们实现的数据包捕获模块中，对信息的提取放在内核态下，通过 netlink 队列传送的消息，仅仅是固定的数据结构，进一步提高了效率，由于设定了高优先级，所以使用 Netfilter 框架下的包捕获程序具有较高的实时性，在正常流量下不存在丢包率。当数据流量非常大的时候，由于优先处理钩子函数，所以会影响主机上的其他应用程序的处理。但总的来说，使用 Netfilter 框架，可以由用户自己设定数据包的捕获处理条件和流程，从而使框架更加简洁，具有高实时性，低丢包率。

通过对 Netfilter 框架的理解和学习，通过函数实现了框架下的数据包捕获，并集中解决了使用 Netlink 实现内核态进程和用户态进程之间的通信，并在局域网实现了数据包捕获，运行良好。下一步的工作是在捕获数据包的基础上，丰富内核函数的数据包处理功能，实现智能过滤功能。

参 考 文 献

1 博嘉科技. Linux 防火墙技术探秘[M]. 北京：国防工业出版社，2002

2 Wall K. GUN/Linux 编程指南(第二版)[M]. 清华大学出版社，2005

3 Russell R. Linux Netfilter Hacking HOWTO [EB/ OL]. http:// www. netfilter. org/ documentation/ HOWTO/ / netfilter - hacking - HOWTO. html. 2002 - 07 - 02

4 He K. Why and How to Use Netlink Socket. http://www. linuxjournal. com/article/7356

5 Linux Kernel 2. 4. 20 内核源代码. http://www. kernel. org/