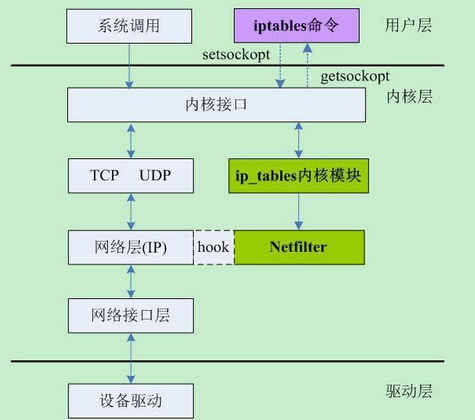
使用netfilter编程，不仅需要深入理解netfilter和iptables的工作原理和组织结构，对一些编程的细节也需要有了解。



Iptables内核模块就是利用netfilter提供的底层接口框架，提供给用户更容易使用的命令行或图形界面。

使用netfilter编程，除了一些套路的hook函数注册，最重要的就是sk\_buff这个数据结构了。

对于sk\_buff，很容易引起误解的一点是

unsigned int hook\_func(unsigned int hooknum,struct sk\_buff \*skb,const struct net\_device \*in,const struct net\_device \*out,int(okfn)(struct sk\_buff \*)){

skbuff = skb;

iph = ip\_hdr(skbuff);

printk("Network header is %u.\n",iph);

tcph = tcp\_hdr(skbuff);

printk("Transport header is %u.\n",tcph);

printk("Transport header is %u.\n",skbuff->transport\_header);

printk("Network header is %u.\n",skbuff->network\_header);

printk("MAC header is %u.\n",skbuff->mac\_header);

return NF\_ACCEPT;

}

参考skb.c.

此时输出的内容，很惊讶地除了MAC地址之外，所有的四个地址都是一样的，也即都指向IP头部。

所以在编程中直接使用tcp\_hdr都没用。

unsigned int hook\_func(unsigned int hooknum,struct sk\_buff \*skb,const struct net\_device \*in,const struct net\_device \*out,int(okfn)(struct sk\_buff \*)){

skbuff = skb;

iph = ip\_hdr(skbuff);

// tcph = tcp\_hdr(skbuff);

tcph = (struct tcphdr \*)(skb->data + (ip\_hdr(skb)->ihl \* 4));

if(iph->ihl == 5)

printk("ip header length is 20.\n");

if(iph->protocol == 6){

if(tcph->dest == htons(23)){

printk("drop telnet!\n");

return NF\_DROP;

}

return NF\_ACCEPT;

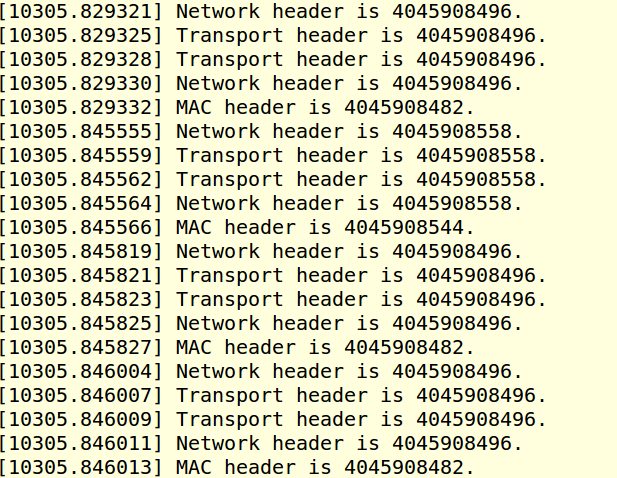
}

else

return NF\_ACCEPT;

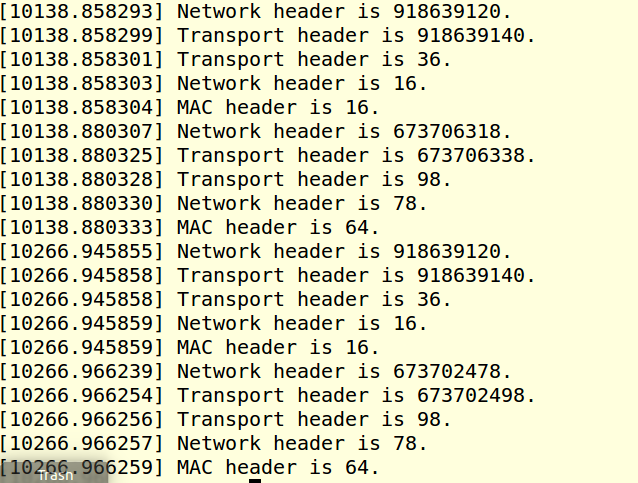
}

这段代码的目的是丢弃所有的telnet数据包。被注释的一行代码是直接使用系统提供的功能，但是似乎没用。

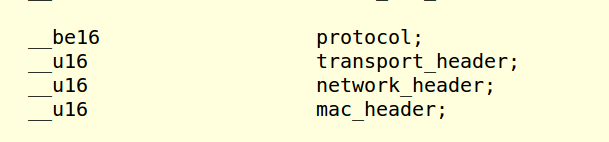


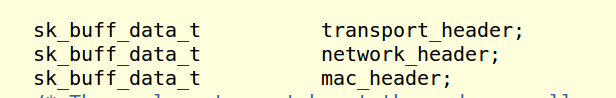
ubuntu12.04

另： 以上代码对ubuntu16以及64位系统无效。



在不同的内核中，sk\_buff的定义不同





由此可见，兼容性很不容易。

12.04上使用tcp\_header不能成功的代码，在16.04上没问题。