# 实验报告

## sniffer

### 原理

sniffer利用winpcap来访问网络底层，

选择好网卡后改为混杂模式，

截取获得帧（pkt\_data[]），

再通过构建结构体（\_iphdr/\_tcphdr/\_udphdr）来分析报文。

以下是包过滤的关键部分：

if (inum1 == 1){ //包过滤字符串

packet\_filter = "ip and tcp";

}

else if (inum1 == 2){

packet\_filter = "ip and udp";

}

else if (inum1 == 3){

packet\_filter = "arp";

}

pcap\_compile(adhandle, &fcode, packet\_filter, 1, netmask) //fcode是packet\_filter编译好后的过滤码

pcap\_setfilter(adhandle, &fcode) //设置这个过滤器

/\*判断是否是IP报文\*/

if (pkt\_data[13] == 0) // IP MAC帧的type字段是0x0800 所以第十四个字节是0

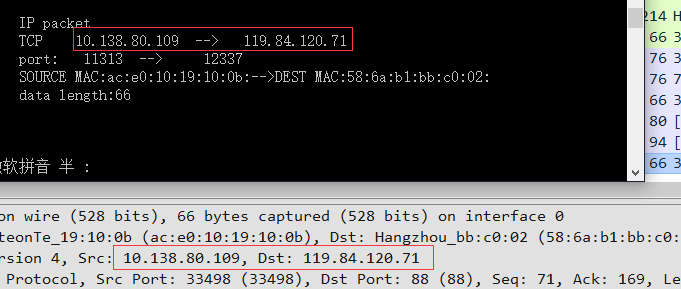
/\*如果等于6则是arp报文\*/

else if (pkt\_data[13] == 6) //ARP MAC帧的type字段是0x0806 所以第十四个字节是6

### 改进的地方

A.首先观察程序运行可以发现，端口号部分是存在问题的。

如下图所示，通过和wireshark进行对比，sniffer打印的端口号不正确。



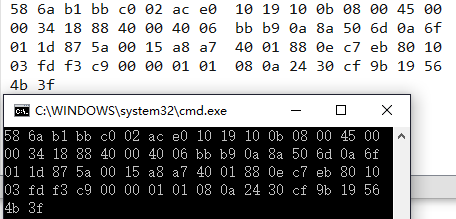
在代码中添加一段，打印 pkt\_data[]

for (i = 0; i <= header->len-1; i++)

{

printf("%.2x ", pkt\_data[i]);

}



发现数据帧有一致，没有问题，猜测是是头部结构体有问题。

添加一段，打印ip header 和tcp header

printf("%x\n", pip->h\_lenver); //4位首部长度+4位IP版本号

printf("%x\n", pip->tos); //8位服务类型TOS

printf("%x\n", pip->total\_len); //16位总长度（字节）

printf("%x\n", pip->ident); //16位标识

printf("%x\n", pip->frag\_and\_flags); //3位标志位

printf("%x\n", pip->ttl); //8位生存时间 TTL

printf("%x\n", pip->proto); //8位协议 (TCP, UDP 或其他)

printf("%x\n", pip->checksum); //16位IP首部校验和

printf("%x\n", pip->sourceIP); //32位源IP地址

printf("%x\n", pip->destIP); //32位目的IP地址

printf("-------------------\n");

printf("%x\n", ptcp->th\_sport); //16位源端口

printf("%x\n", ptcp->th\_dport); //16位目的端口

printf("%x\n", ptcp->th\_seq); //32位序列号

printf("%x\n", ptcp->th\_ack); //32位确认号

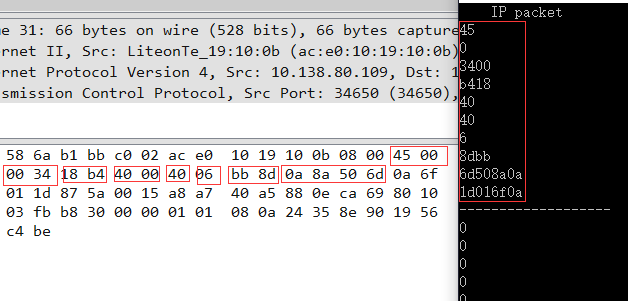
printf("%x\n", ptcp->th\_lenres); //4位首部长度/6位保留字

printf("%x\n", ptcp->th\_flag); //6位标志位

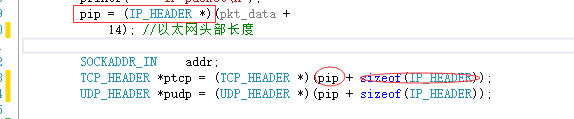
printf("%x\n", ptcp->th\_win); //16位窗口大小

printf("%x\n", ptcp->th\_sum); //16位校验和

printf("%x\n", ptcp->th\_urp); //16位紧急数据偏移量

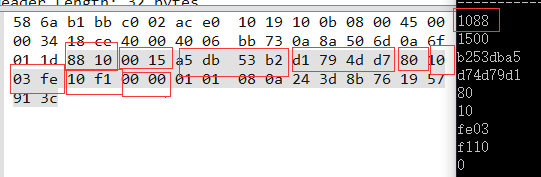


图中分别是网络字节顺序和主机字节顺序，可以发现IP报头没有问题但是TCP报头不正确，猜测是TCP报头指针有问题，检查代码发现：



pip是一个结构体的指针，由于结构体不是单字节等长的数组，加上一个常数后并没有指向正确的TCP头部地址（打印pip[i]即可证明），解决办法是改为 pkt\_data +14

TCP\_HEADER \*ptcp = (TCP\_HEADER \*)(pkt\_data +14 + sizeof(IP\_HEADER));

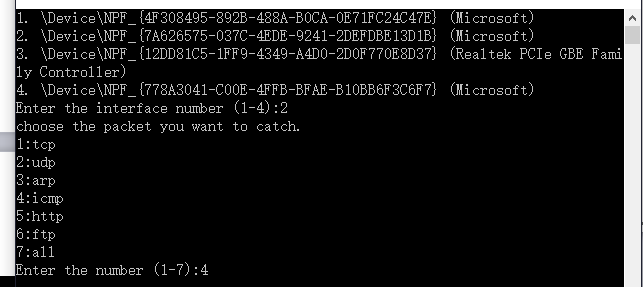


指针正常了。

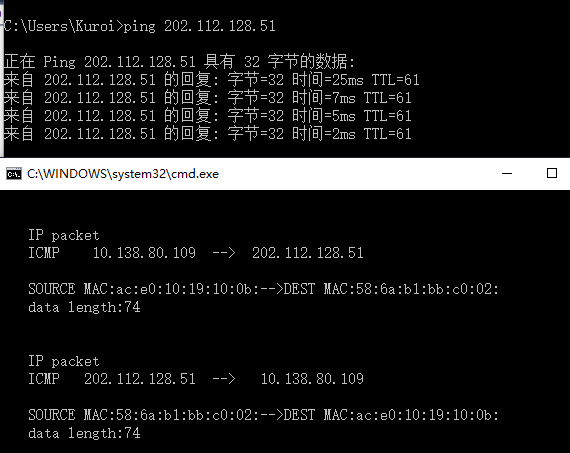
B.增加一些选项（ICMP、HTTP、FTP）

以下是各个新增选项的截图，具体代码见cpp文件

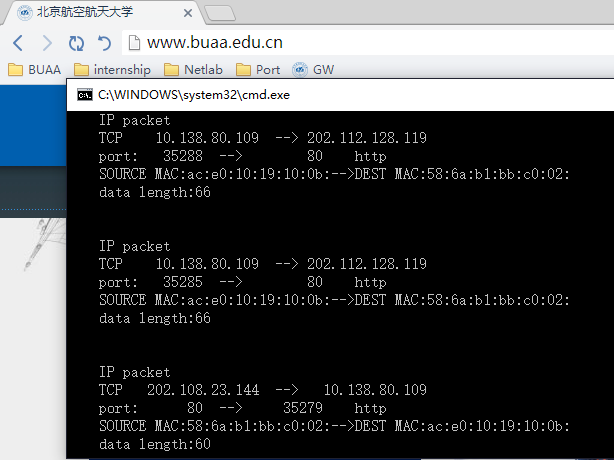
网卡选择/过滤选择



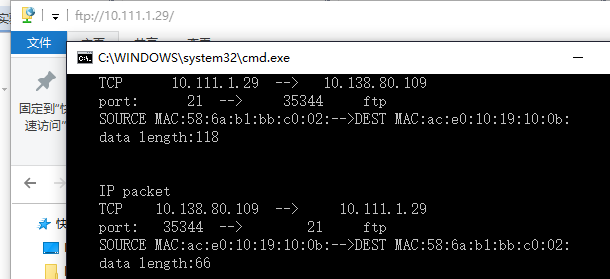
ICMP选项



HTTP选项



FTP选项



### 3.收获

对winpcap编程有了直观的认识，加深了IP、TCP、UDP报文帧结构的印象。

## CICSO网络仿真

### 1原理

### 2.改进的地方

### 3.收获