计算机网络

实验三　用 PCAP 库侦听并分析网络流量

20420192201952　庾晓萍

主要思路

一、 用侦听解析软件Wireshark观察数据格式。

（较简单，拿TCP协议数据包在wireshark上观察就可以）

二、 用侦听解析软件观察 TCP 机制（TCP的三次握手、四次挥手）

三、 用 WinPcap 库侦听网络数据

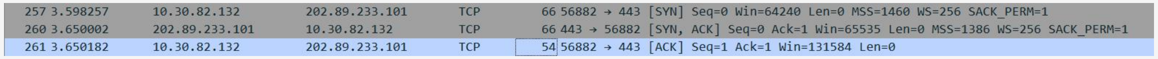
- WinPcap 库侦听网络数据（具体看b站视频）
- 解析MAC和 IP 地址，记录统计（在 winpcap 工程上修改）
- 解析侦听到的网络数据（以FTP密码侦听为例）

Wireshark观察 TCP 机制（不汇报，供大家参考）

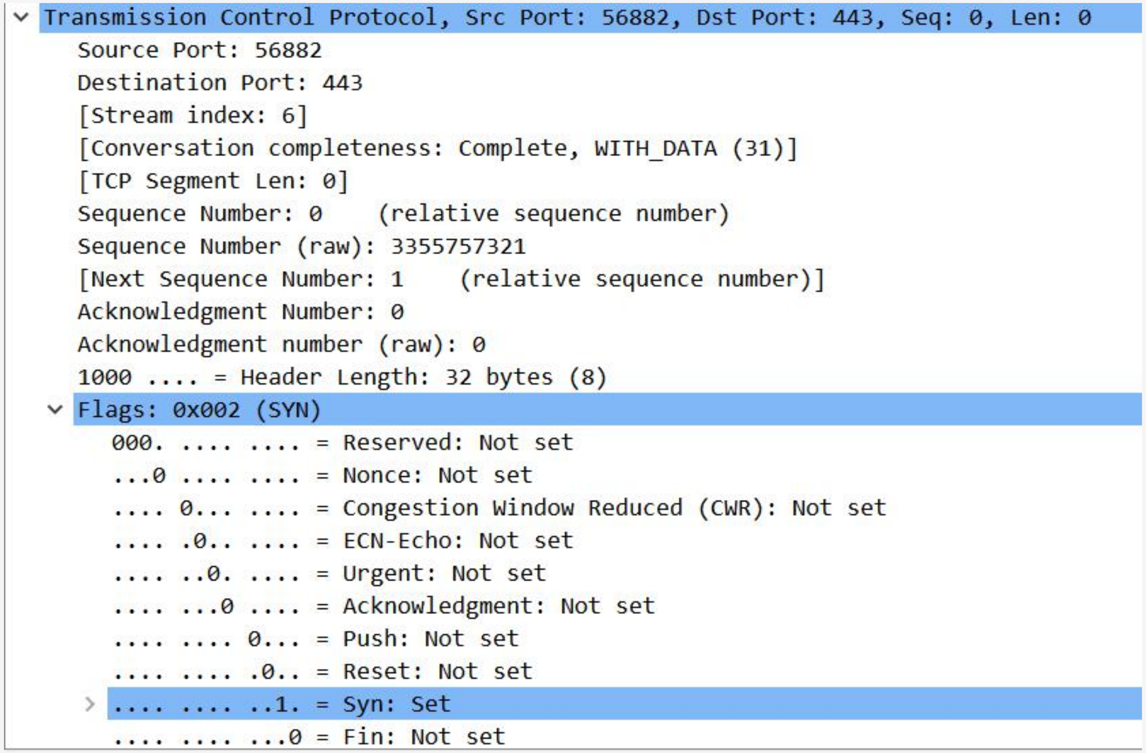
1、TCP三次握手

第一个握手：

客户端发送连接请求报文段，无应用层数据，标志位为同步比特SYN，用来同步序号。序列号seq为0，代表客户端请求建立连接。



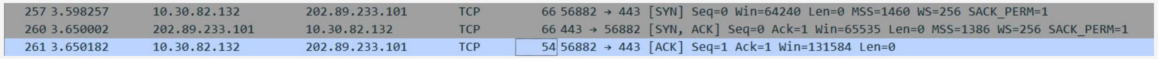
```
257 3.598257   10.30.82.132     202.89.233.101   TCP   66 56882 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
260 3.650002   202.89.233.101   10.30.82.132     TCP   66 443 → 56882 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1386 WS=256 SACK_PERM=1
261 3.650182   10.30.82.132     202.89.233.101   TCP   54 56882 → 443 [ACK] Seq=1 Ack=1 Win=131584 Len=0
```



```
v Transmission Control Protocol, Src Port: 56882, Dst Port: 443, Seq: 0, Len: 0
    Source Port: 56882
    Destination Port: 443
    [Stream index: 6]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 3355757321
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  v Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    > .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
```

Wireshark观察 TCP 机制（不汇报，供大家参考）

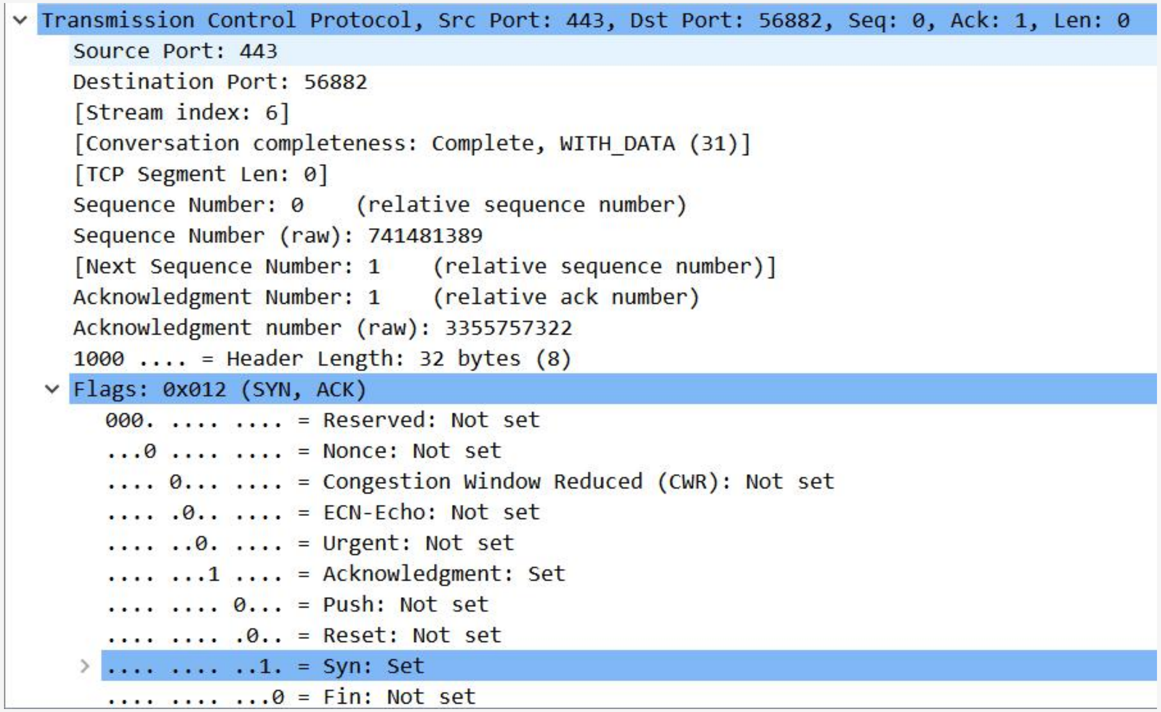
1、TCP三次握手



```
257 3.598257    10.30.82.132      202.89.233.101    TCP       66 56882 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
260 3.650002    202.89.233.101    10.30.82.132      TCP       66 443 → 56882 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1386 WS=256 SACK_PERM=1
261 3.650182    10.30.82.132      202.89.233.101    TCP       54 56882 → 443 [ACK] Seq=1 Ack=1 Win=131584 Len=0
```

第二个握手：

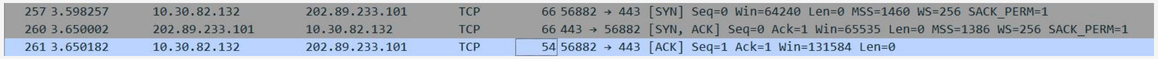
服务器端为该TCP连接分配缓存和变量，并向客户端返回确认报文段，表示允许连接，无应用层数据。标志位为SYN=1，ACK=1。将确认ack设置为1。



```
v Transmission Control Protocol, Src Port: 443, Dst Port: 56882, Seq: 0, Ack: 1, Len: 0
    Source Port: 443
    Destination Port: 56882
    [Stream index: 6]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 741481389
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3355757322
    1000 .... = Header Length: 32 bytes (8)
  v Flags: 0x012 (SYN, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    > .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
```

Wireshark观察 TCP 机制（不汇报，供大家参考）

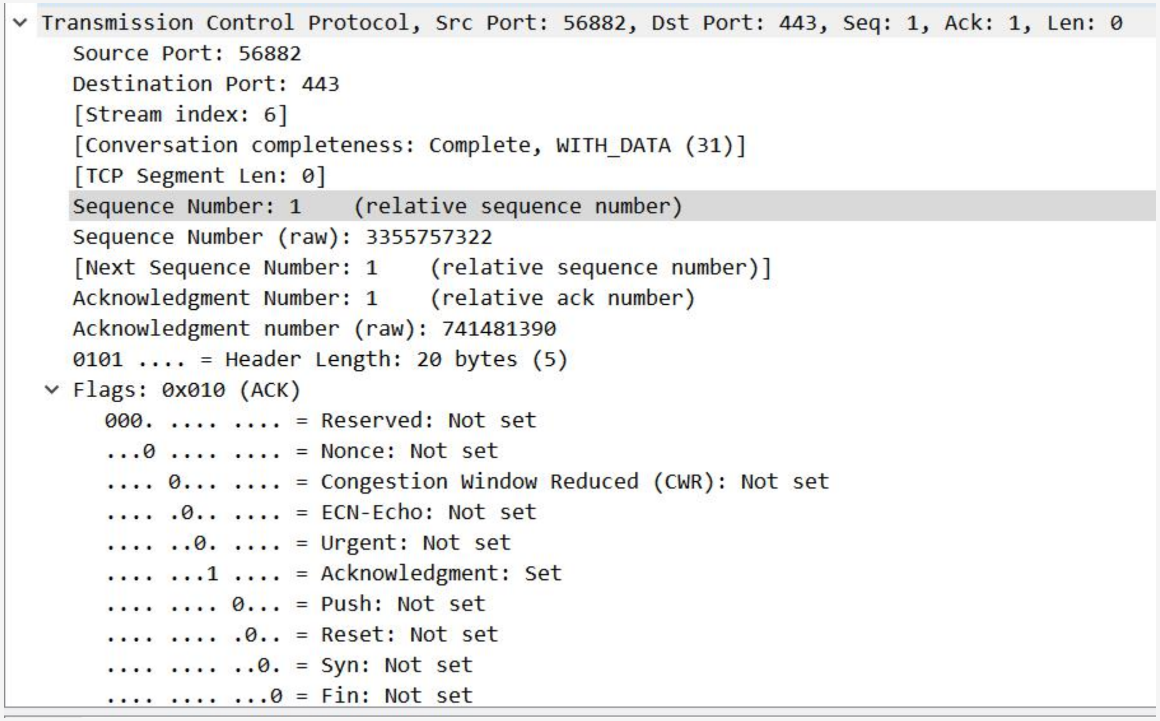
1、TCP三次握手



```
257 3.598257   10.30.82.132     202.89.233.101   TCP   66 56882 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
260 3.650002   202.89.233.101   10.30.82.132     TCP   66 443 → 56882 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1386 WS=256 SACK_PERM=1
261 3.650182   10.30.82.132     202.89.233.101   TCP   54 56882 → 443 [ACK] Seq=1 Ack=1 Win=131584 Len=0
```

第三个握手：

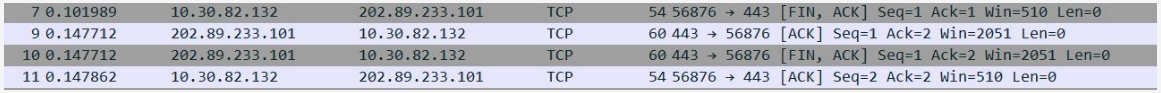
客户端为该TCP连接分配缓存和变量，并向服务器端再次发送确认包(ACK)，可以携带数据。SYN = 0，ACK=1。并且把序列号seq+1。



```
v Transmission Control Protocol, Src Port: 56882, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
    Source Port: 56882
    Destination Port: 443
    [Stream index: 6]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 3355757322
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 741481390
    0101 .... = Header Length: 20 bytes (5)
  v Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
```

Wireshark观察 TCP 机制（不汇报，供大家参考）

2、TCP四次挥手

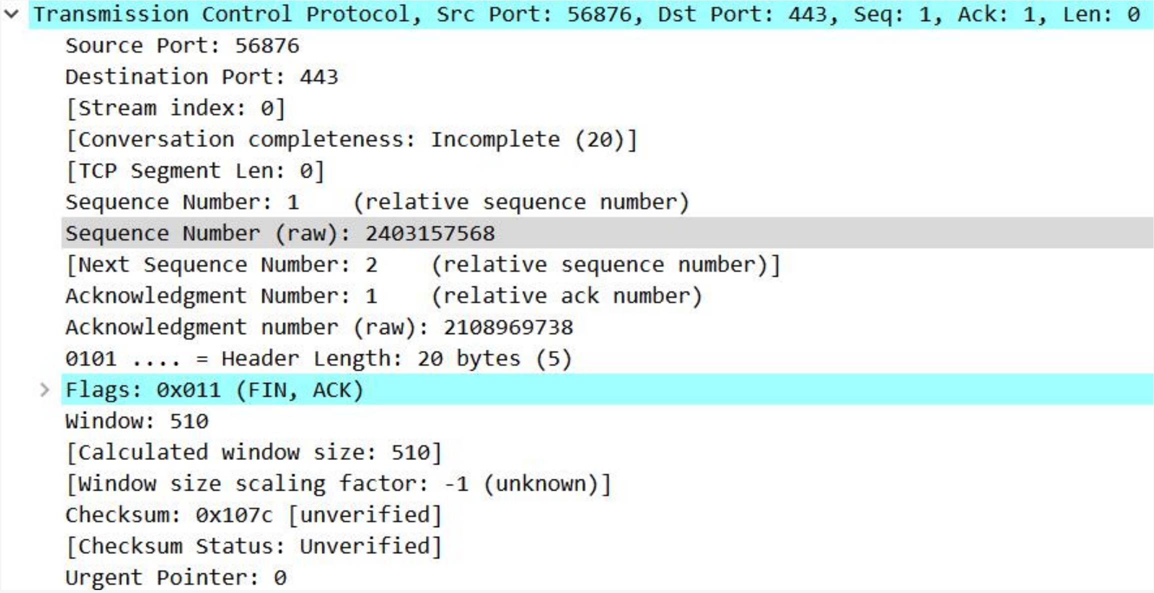


```
 7 0.101989   10.30.82.132     202.89.233.101   TCP    54 56876 → 443 [FIN, ACK] Seq=1 Ack=1 Win=510 Len=0
 9 0.147712   202.89.233.101   10.30.82.132     TCP    60 443 → 56876 [ACK] Seq=1 Ack=2 Win=2051 Len=0
10 0.147712   202.89.233.101   10.30.82.132     TCP    60 443 → 56876 [FIN, ACK] Seq=1 Ack=2 Win=2051 Len=0
11 0.147862   10.30.82.132     202.89.233.101   TCP    54 56876 → 443 [ACK] Seq=2 Ack=2 Win=510 Len=0
```

第一次挥手：

客户端发送连接释放报文段，停止发送数据，主动关闭TCP连接。

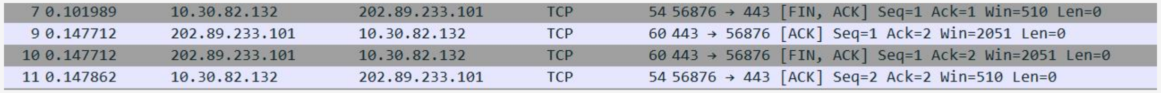
FIN = 1，ACK=1。



```
∨ Transmission Control Protocol, Src Port: 56876, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
    Source Port: 56876
    Destination Port: 443
    [Stream index: 0]
    [Conversation completeness: Incomplete (20)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2403157568
    [Next Sequence Number: 2    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 2108969738
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x011 (FIN, ACK)
    Window: 510
    [Calculated window size: 510]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x107c [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
```

Wireshark观察 TCP 机制（不汇报，供大家参考）

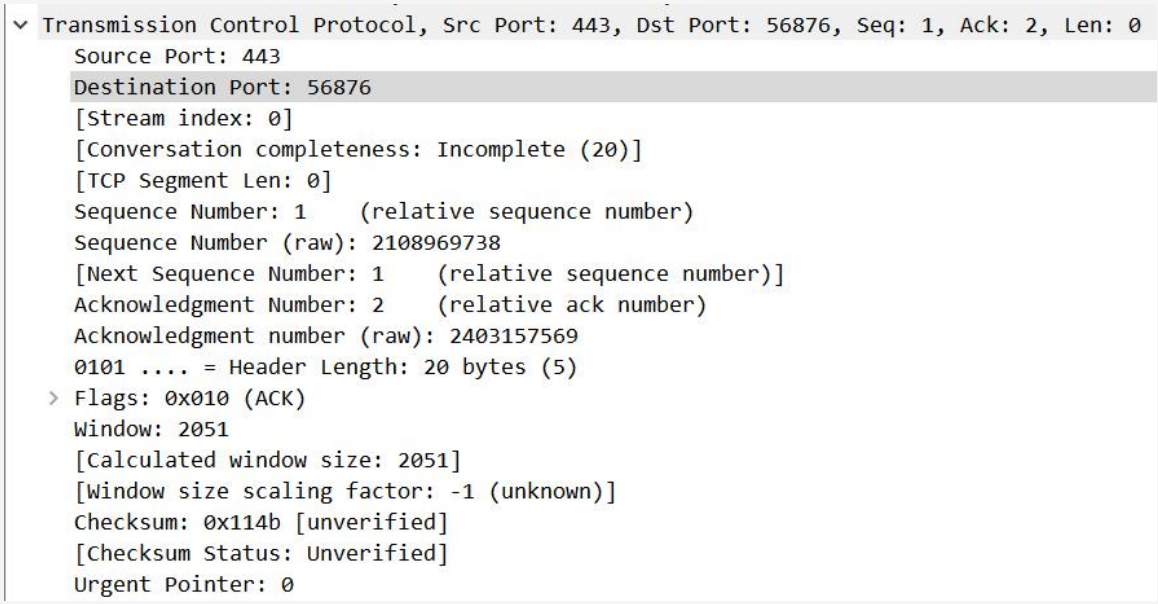
2、TCP四次挥手



```
 7 0.101989   10.30.82.132     202.89.233.101   TCP   54 56876 → 443 [FIN, ACK] Seq=1 Ack=1 Win=510 Len=0
 9 0.147712   202.89.233.101   10.30.82.132     TCP   60 443 → 56876 [ACK] Seq=1 Ack=2 Win=2051 Len=0
10 0.147712   202.89.233.101   10.30.82.132     TCP   60 443 → 56876 [FIN, ACK] Seq=1 Ack=2 Win=2051 Len=0
11 0.147862   10.30.82.132     202.89.233.101   TCP   54 56876 → 443 [ACK] Seq=2 Ack=2 Win=510 Len=0
```

第二次挥手：

服务器端回送确认报文段，客户到服务器的连接释放，半关闭状态。ACK = 1。序号为1，确认序号为2。



```
∨ Transmission Control Protocol, Src Port: 443, Dst Port: 56876, Seq: 1, Ack: 2, Len: 0
    Source Port: 443
    Destination Port: 56876
    [Stream index: 0]
    [Conversation completeness: Incomplete (20)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2108969738
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 2    (relative ack number)
    Acknowledgment number (raw): 2403157569
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 2051
    [Calculated window size: 2051]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x114b [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
```

第三次挥手：

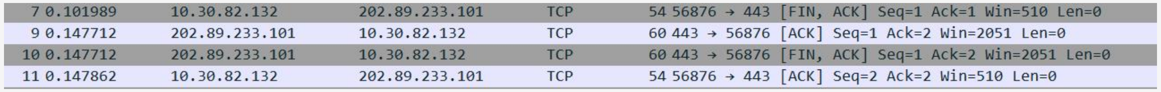
服务器端发送完数据，就发送连接释放报文段，主动关闭TCP连接。FIN = 1，ACK = 1。

Wireshark观察 TCP 机制（不汇报，供大家参考）

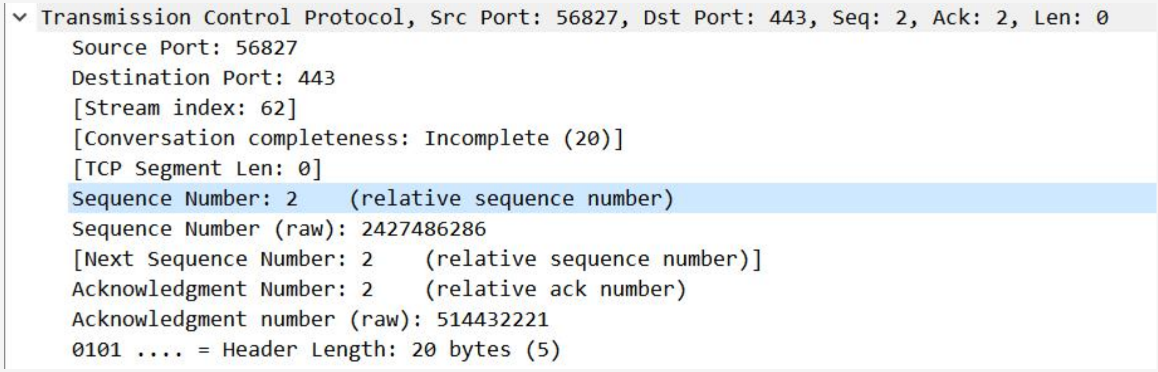
2、TCP四次挥手

第四次挥手：

客户端回送一个确认报文段，到时间等待计时器设置的最长报文段寿命后，连接彻底关闭。ACK=1，序号为2，确认序号为2。



```
 7 0.101989    10.30.82.132     202.89.233.101    TCP    54 56876 → 443 [FIN, ACK] Seq=1 Ack=1 Win=510 Len=0
 9 0.147712    202.89.233.101   10.30.82.132      TCP    60 443 → 56876 [ACK] Seq=1 Ack=2 Win=2051 Len=0
10 0.147712    202.89.233.101   10.30.82.132      TCP    60 443 → 56876 [FIN, ACK] Seq=1 Ack=2 Win=2051 Len=0
11 0.147862    10.30.82.132     202.89.233.101    TCP    54 56876 → 443 [ACK] Seq=2 Ack=2 Win=510 Len=0
```

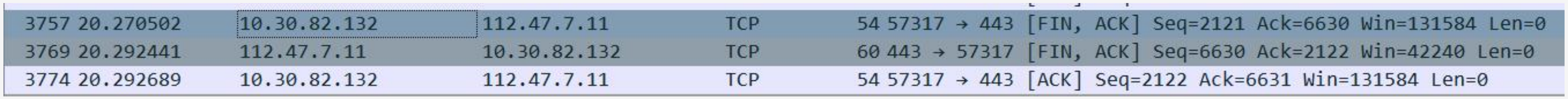


```
⌄ Transmission Control Protocol, Src Port: 56827, Dst Port: 443, Seq: 2, Ack: 2, Len: 0
    Source Port: 56827
    Destination Port: 443
    [Stream index: 62]
    [Conversation completeness: Incomplete (20)]
    [TCP Segment Len: 0]
    Sequence Number: 2    (relative sequence number)
    Sequence Number (raw): 2427486286
    [Next Sequence Number: 2    (relative sequence number)]
    Acknowledgment Number: 2    (relative ack number)
    Acknowledgment number (raw): 514432221
    0101 .... = Header Length: 20 bytes (5)
```

问题

发现很多抓到的 TCP 挥手是三次，而不是四次。

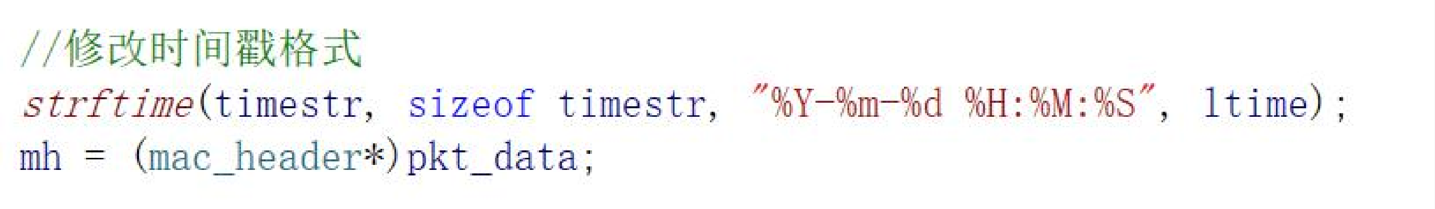
查资料是因为服务器端收到客户端的 FIN 后，服务器端同时也要关闭连接，这样就可以把 ACK 和 FIN 合并到一起发送，节省了一个包，变成了“三次挥手”。



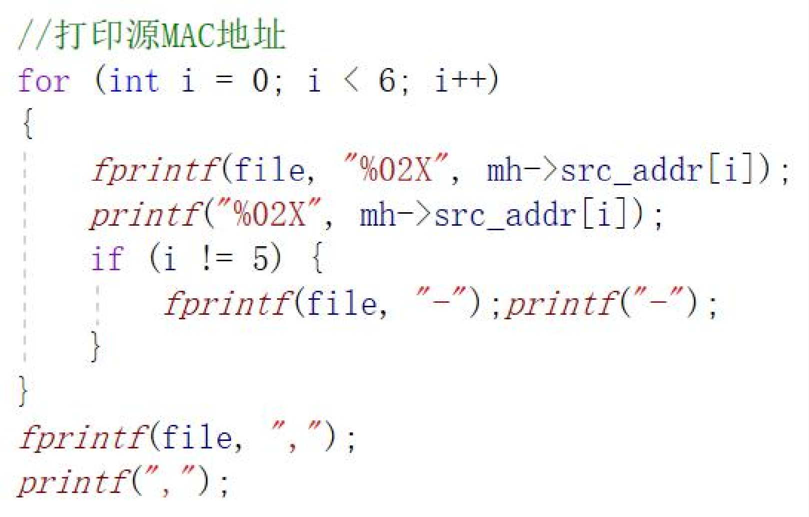
```
3757 20.270502   10.30.82.132   112.47.7.11    TCP   54 57317 → 443 [FIN, ACK] Seq=2121 Ack=6630 Win=131584 Len=0
3769 20.292441   112.47.7.11    10.30.82.132   TCP   60 443 → 57317 [FIN, ACK] Seq=6630 Ack=2122 Win=42240 Len=0
3774 20.292689   10.30.82.132   112.47.7.11    TCP   54 57317 → 443 [ACK] Seq=2122 Ack=6631 Win=131584 Len=0
```

解析MAC和 IP 地址，记录统计（核心代码）

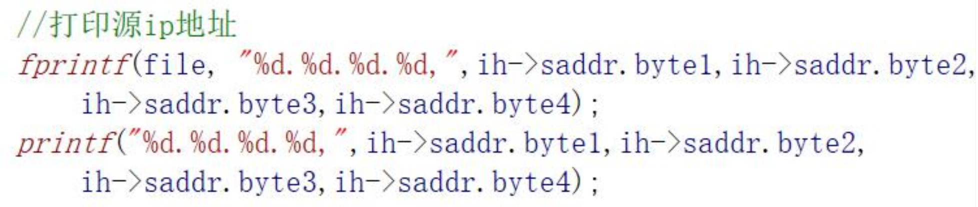
PART 1：修改输出到csv文件中的格式 （修改目的地址类似）



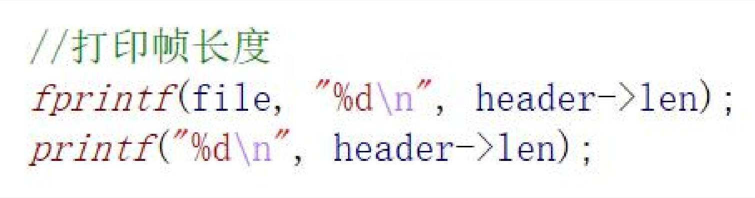
```
//修改时间戳格式
strftime(timestr, sizeof timestr, "%Y-%m-%d %H:%M:%S", ltime);
mh = (mac_header*)pkt_data;
```



```
//打印源MAC地址
for (int i = 0; i < 6; i++)
{
    fprintf(file, "%02X", mh->src_addr[i]);
    printf("%02X", mh->src_addr[i]);
    if (i != 5) {
        fprintf(file, "-");printf("-");
    }
}
fprintf(file, ",");
printf(",");
```



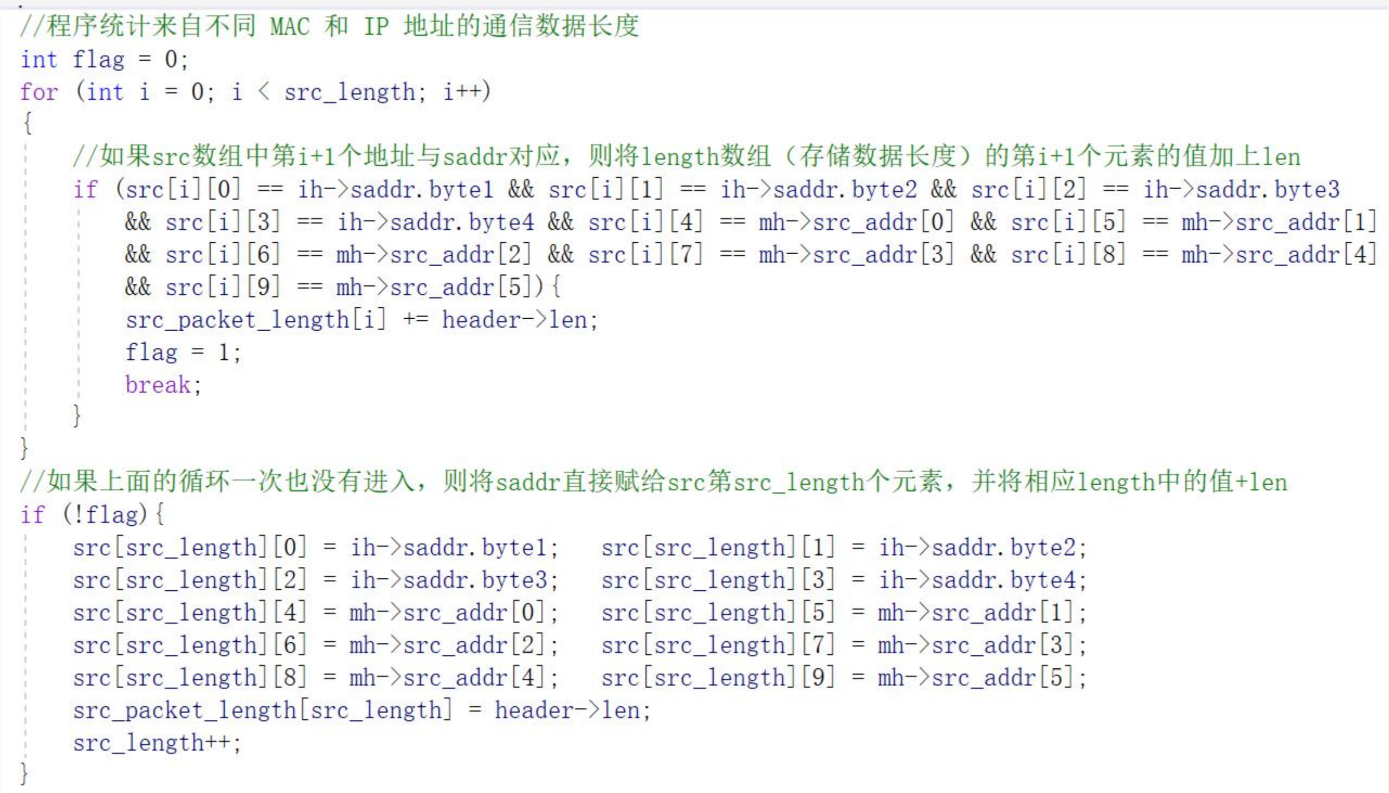
```
//打印源ip地址
fprintf(file, "%d.%d.%d.%d,",ih->saddr.byte1,ih->saddr.byte2,
    ih->saddr.byte3,ih->saddr.byte4);
printf("%d.%d.%d.%d,",ih->saddr.byte1,ih->saddr.byte2,
    ih->saddr.byte3,ih->saddr.byte4);
```



```
//打印帧长度
fprintf(file, "%d\n", header->len);
printf("%d\n", header->len);
```

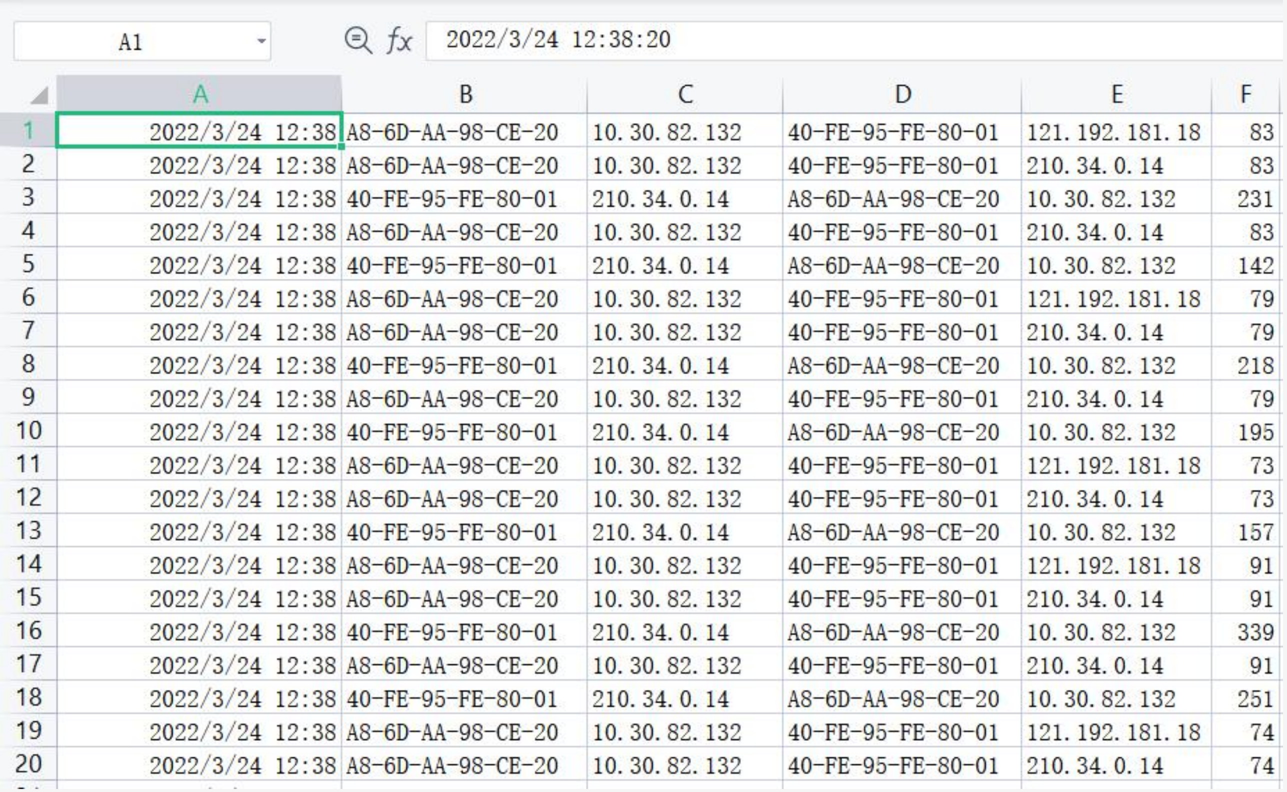
解析MAC和 IP 地址，记录统计（核心代码）

PART 2：对每分钟数据统计分析（统计发送到不同MAC和IP地址的通信数据长度的代码是类似）



```
//程序统计来自不同 MAC 和 IP 地址的通信数据长度
int flag = 0;
for (int i = 0; i < src_length; i++)
{
    //如果src数组中第i+1个地址与saddr对应，则将length数组（存储数据长度）的第i+1个元素的值加上len
    if (src[i][0] == ih->saddr.byte1 && src[i][1] == ih->saddr.byte2 && src[i][2] == ih->saddr.byte3
        && src[i][3] == ih->saddr.byte4 && src[i][4] == mh->src_addr[0] && src[i][5] == mh->src_addr[1]
        && src[i][6] == mh->src_addr[2] && src[i][7] == mh->src_addr[3] && src[i][8] == mh->src_addr[4]
        && src[i][9] == mh->src_addr[5]){
        src_packet_length[i] += header->len;
        flag = 1;
        break;
    }
}
//如果上面的循环一次也没有进入，则将saddr直接赋给src第src_length个元素，并将相应length中的值+len
if (!flag){
    src[src_length][0] = ih->saddr.byte1;    src[src_length][1] = ih->saddr.byte2;
    src[src_length][2] = ih->saddr.byte3;    src[src_length][3] = ih->saddr.byte4;
    src[src_length][4] = mh->src_addr[0];    src[src_length][5] = mh->src_addr[1];
    src[src_length][6] = mh->src_addr[2];    src[src_length][7] = mh->src_addr[3];
    src[src_length][8] = mh->src_addr[4];    src[src_length][9] = mh->src_addr[5];
    src_packet_length[src_length] = header->len;
    src_length++;
}
```

解析MAC和 IP 地址，记录统计（实验结果）

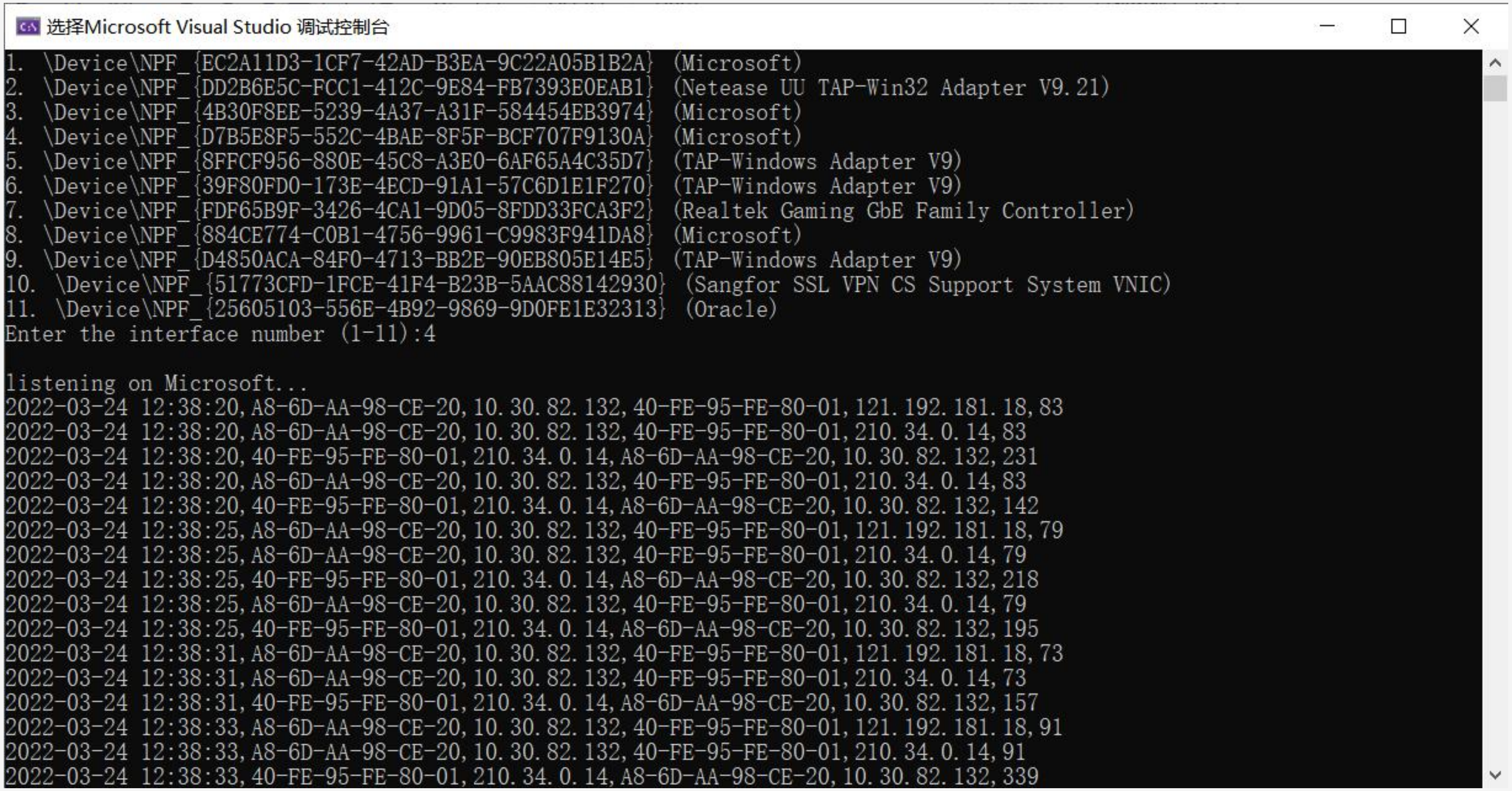


A1

2022/3/24 12:38:20

	A	B	C	D	E	F
1	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	121.192.181.18	83
2	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	83
3	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	231
4	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	83
5	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	142
6	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	121.192.181.18	79
7	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	79
8	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	218
9	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	79
10	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	195
11	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	121.192.181.18	73
12	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	73
13	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	157
14	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	121.192.181.18	91
15	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	91
16	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	339
17	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	91
18	2022/3/24 12:38	40-FE-95-FE-80-01	210.34.0.14	A8-6D-AA-98-CE-20	10.30.82.132	251
19	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	121.192.181.18	74
20	2022/3/24 12:38	A8-6D-AA-98-CE-20	10.30.82.132	40-FE-95-FE-80-01	210.34.0.14	74

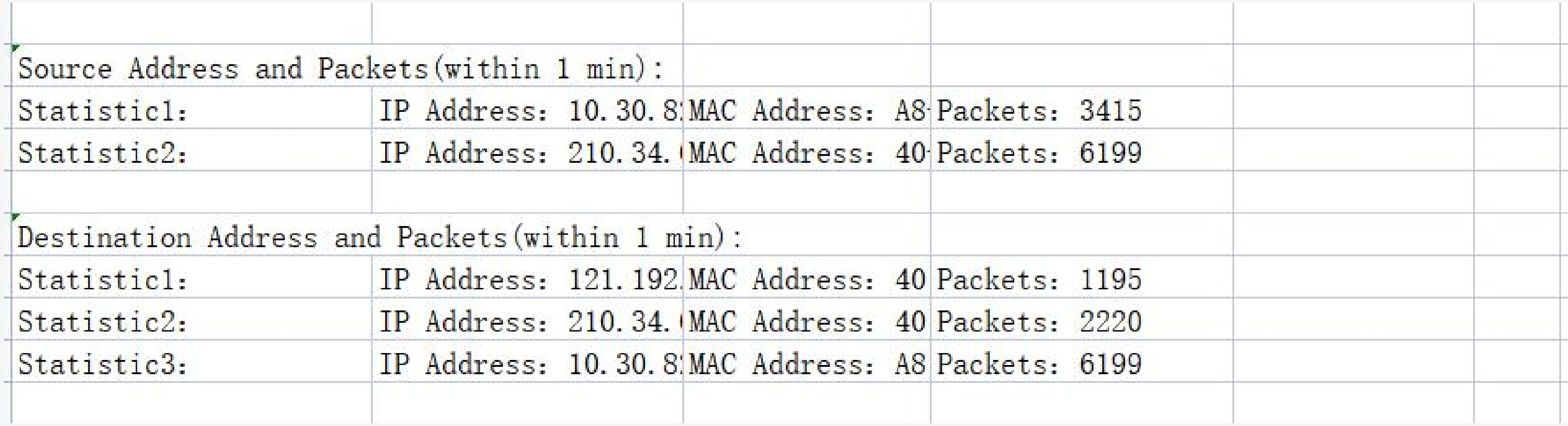
解析MAC和 IP 地址，记录统计（实验结果）



```
1. \Device\NPF_{EC2A11D3-1CF7-42AD-B3EA-9C22A05B1B2A} (Microsoft)
2. \Device\NPF_{DD2B6E5C-FCC1-412C-9E84-FB7393E0EAB1} (Netease UU TAP-Win32 Adapter V9.21)
3. \Device\NPF_{4B30F8EE-5239-4A37-A31F-584454EB3974} (Microsoft)
4. \Device\NPF_{D7B5E8F5-552C-4BAE-8F5F-BCF707F9130A} (Microsoft)
5. \Device\NPF_{8FFCF956-880E-45C8-A3E0-6AF65A4C35D7} (TAP-Windows Adapter V9)
6. \Device\NPF_{39F80FD0-173E-4ECD-91A1-57C6D1E1F270} (TAP-Windows Adapter V9)
7. \Device\NPF_{FDF65B9F-3426-4CA1-9D05-8FDD33FCA3F2} (Realtek Gaming GbE Family Controller)
8. \Device\NPF_{884CE774-C0B1-4756-9961-C9983F941DA8} (Microsoft)
9. \Device\NPF_{D4850ACA-84F0-4713-BB2E-90EB805E14E5} (TAP-Windows Adapter V9)
10. \Device\NPF_{51773CFD-1FCE-41F4-B23B-5AAC88142930} (Sangfor SSL VPN CS Support System VNIC)
11. \Device\NPF_{25605103-556E-4B92-9869-9D0FE1E32313} (Oracle)
Enter the interface number (1-11):4

listening on Microsoft...
2022-03-24 12:38:20,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,121.192.181.18,83
2022-03-24 12:38:20,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,83
2022-03-24 12:38:20,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,231
2022-03-24 12:38:20,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,83
2022-03-24 12:38:20,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,142
2022-03-24 12:38:25,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,121.192.181.18,79
2022-03-24 12:38:25,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,79
2022-03-24 12:38:25,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,218
2022-03-24 12:38:25,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,79
2022-03-24 12:38:25,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,195
2022-03-24 12:38:31,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,121.192.181.18,73
2022-03-24 12:38:31,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,73
2022-03-24 12:38:31,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,157
2022-03-24 12:38:33,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,121.192.181.18,91
2022-03-24 12:38:33,A8-6D-AA-98-CE-20,10.30.82.132,40-FE-95-FE-80-01,210.34.0.14,91
2022-03-24 12:38:33,40-FE-95-FE-80-01,210.34.0.14,A8-6D-AA-98-CE-20,10.30.82.132,339
```

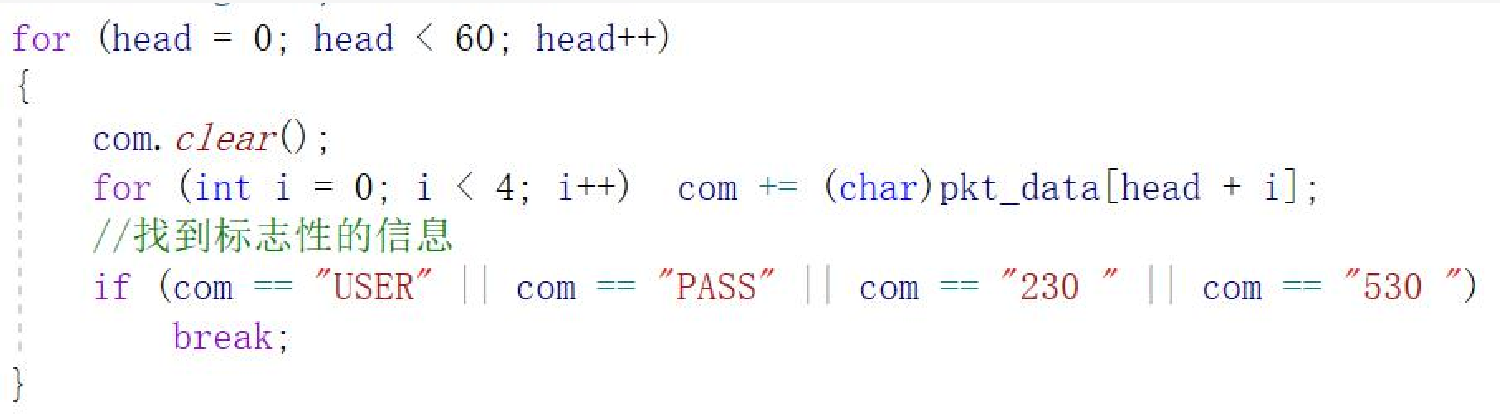
解析MAC和 IP 地址，记录统计（实验结果）



Source Address and Packets(within 1 min):			
Statistic1:	IP Address: 10.30.8	MAC Address: A8	Packets: 3415
Statistic2:	IP Address: 210.34.	MAC Address: 40	Packets: 6199
Destination Address and Packets(within 1 min):			
Statistic1:	IP Address: 121.192.	MAC Address: 40	Packets: 1195
Statistic2:	IP Address: 210.34.	MAC Address: 40	Packets: 2220
Statistic3:	IP Address: 10.30.8	MAC Address: A8	Packets: 6199

FTP侦听解析（核心代码）

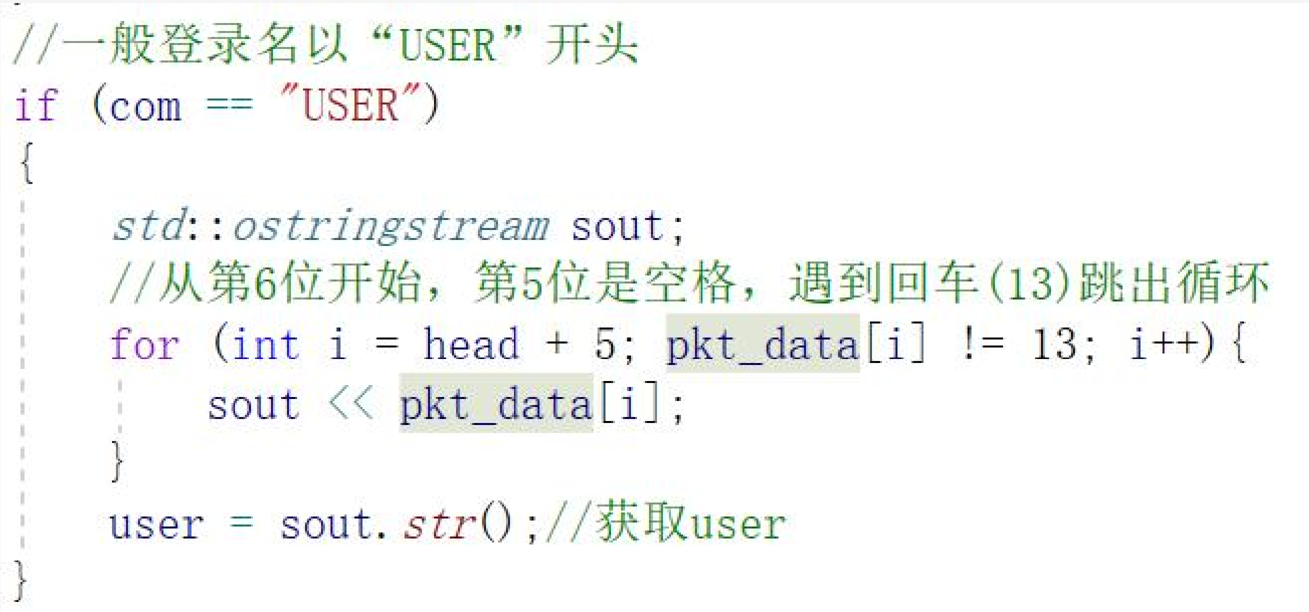
一般登录名以“USER”开头，口令以“PASS”开头，登录成功以“230”开头，失败以“530”开头。



```
for (head = 0; head < 60; head++)
{
    com.clear();
    for (int i = 0; i < 4; i++)  com += (char)pkt_data[head + i];
    //找到标志性的信息
    if (com == "USER" || com == "PASS" || com == "230 " || com == "530 ")
        break;
}
```

FTP侦听解析（核心代码）

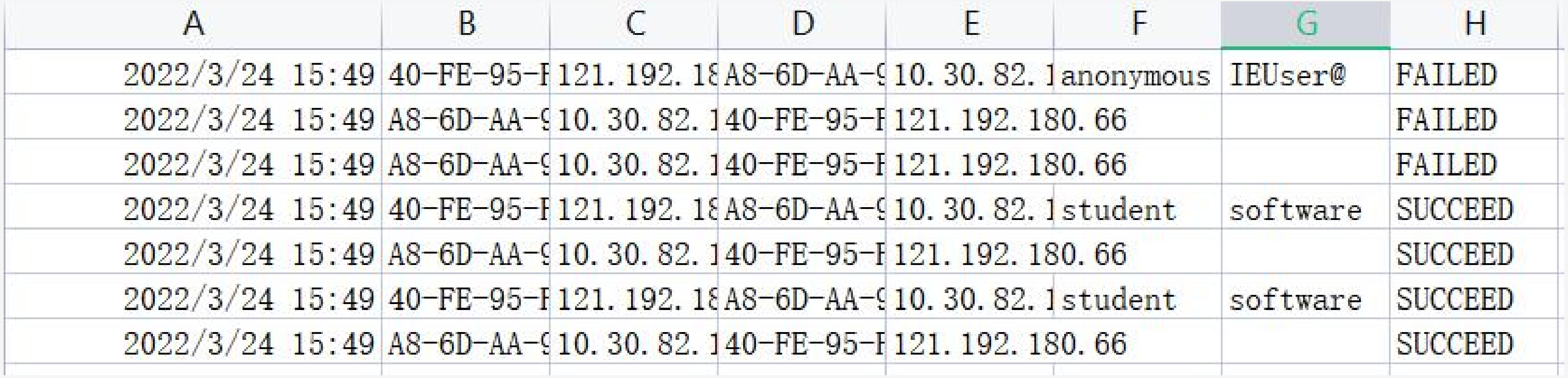
获取USER信息，其他信息获取类似



```
//一般登录名以“USER”开头
if (com == "USER")
{
    std::ostringstream sout;
    //从第6位开始，第5位是空格，遇到回车(13)跳出循环
    for (int i = head + 5; pkt_data[i] != 13; i++){
        sout << pkt_data[i];
    }
    user = sout.str();//获取user
}
```

FTP侦听解析（实验结果）

侦听得到系FTP的用户名和密码



A	B	C	D	E	F	G	H
2022/3/24 15:49	40-FE-95-F	121.192.18	A8-6D-AA-9	10.30.82.1	anonymous	IEUser@	FAILED
2022/3/24 15:49	A8-6D-AA-9	10.30.82.1	40-FE-95-F	121.192.180.66			FAILED
2022/3/24 15:49	A8-6D-AA-9	10.30.82.1	40-FE-95-F	121.192.180.66			FAILED
2022/3/24 15:49	40-FE-95-F	121.192.18	A8-6D-AA-9	10.30.82.1	student	software	SUCCEED
2022/3/24 15:49	A8-6D-AA-9	10.30.82.1	40-FE-95-F	121.192.180.66			SUCCEED
2022/3/24 15:49	40-FE-95-F	121.192.18	A8-6D-AA-9	10.30.82.1	student	software	SUCCEED
2022/3/24 15:49	A8-6D-AA-9	10.30.82.1	40-FE-95-F	121.192.180.66			SUCCEED

注意

1、运行程序时要把csv文件关上，不然会读空

2、使用科来数据包生成器找不到网卡适配器，是软件本身与Win10系统的兼容性问题，更改电脑兼容性并以管理员身份运行。

3、第二个实验会出现自定义inline导致和系统文件发生冲突的问题（Error：The C++ Standard Library forbids macroizing keywords），预处理器定义中加入“_XKEYCHECK_H”来避免。

参考资料

一、Wireshark相关

https://www.cnblogs.com/HOsystem/p/13170860.html

（《WireShark——IP协议包分析》）

http://c.biancheng.net/view/6379.html

（《Wireshark下载安装和使用教程》）

https://www.cnblogs.com/huanxiyun/articles/6553440.html

《wireshark捕获/过滤指定ip地址数据包》

https://cloud.tencent.com/developer/article/1538191

《Wireshark抓包分析 TCP三次握手/四次挥手详解》

参考资料

二、TCP相关

https://www.cnblogs.com/xiaolincoding/p/12732052.html

《30张图解： TCP 重传、滑动窗口、流量控制、拥塞控制》

https://blog.csdn.net/m0_52586092/article/details/119743299

《TCP四次挥手，状态码》

https://blog.csdn.net/qq_35733751/article/details/80552037

《tcp连接——初始化序列号(ISN)》

三、其他参考资料

b站实验三视频

实验三　用 PCAP 库侦听并分析网络流量

谢谢大家！

20420192201952　庾晓萍