厦門大學

软件学院

《计算机网络》实验报告

题　　目 用 WinPCAP 监听并解析 FTP

姓　　名 黄勖

学　　号 22920212204392

班　　级 计算机网络 2021 级 2 班

实验时间 2023.4.17

2023 年 4 月 17 日

1 实验目的

通过捕获并解析 FTP 连接的建立于与断开过程的数据帧，达到以下目的：

熟悉并学会分析 TCP 连接的建立、断开过程；

熟悉常见的 FTP 命令，熟悉并学会分析 FTP 连接的建立、断开过程；

掌握使用代码实现解析以太网数据帧的技能，提高编码能力。

2 实验环境

操作系统：Windows10 21H2

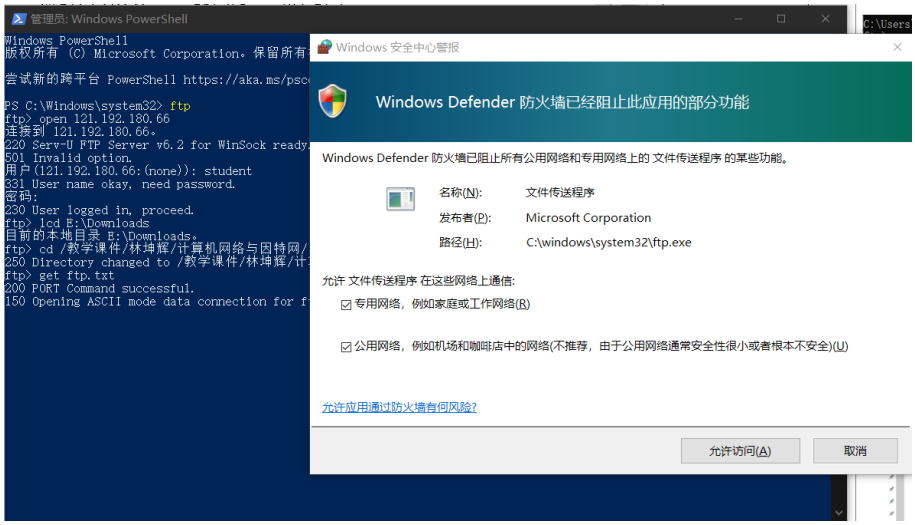
编程语言：C++ with Visual Studio 2022 Community

3 实验结果

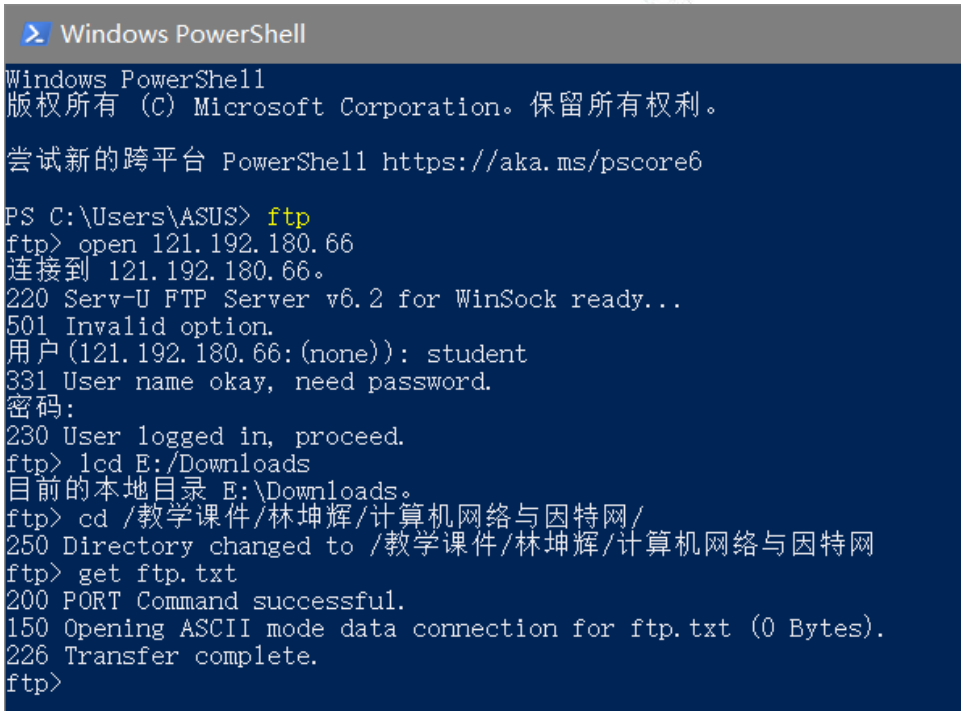
实验项-2：以命令行的方式登录 FTP 服务器，下载文件“/教学课件 /林坤辉/计算机网络 与因特网/ftp.txt”

过程：

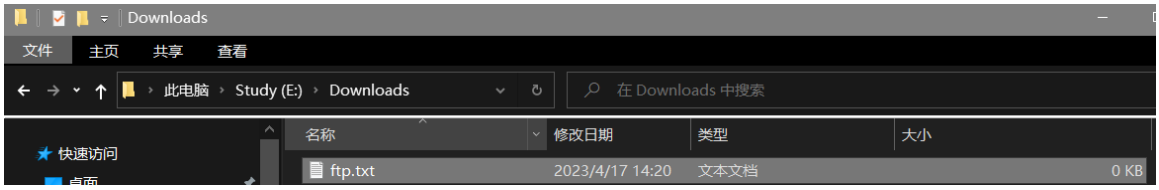
1)允许防火墙访问



2）命令行输入指令



3）下载的文件



实验项-1：使用 Wireshark 等抓包工具监听网络上的数据流，定位出 TCP 连接建立、断开过程，根据要求找出软件抓取到的对应数据帧，如 3 次握手中的 第 2 次握手数据帧；

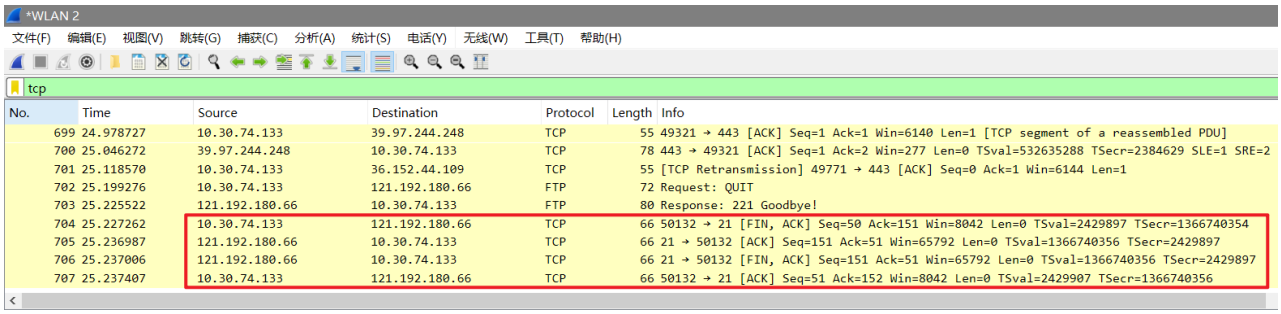
（1）在控制面板查出此时主机 IP 地址

（2）打开 Wireshark 过滤 TCP 包，结果如下

Wireshark 抓到的 3 次握手包：

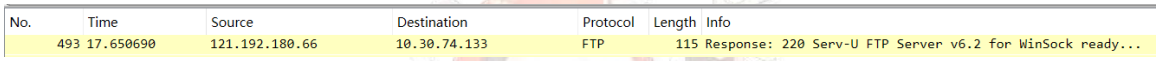
Source	Destination	Protocol	Length	Info
10.30.74.133	121.192.180.66	TCP	74	65172 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=1 SACK_PERM
121.192.180.66	10.30.74.133	TCP	74	21 → 65172 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1386 WS=2
10.30.74.133	121.192.180.66	TCP	66	65172 → 21 [ACK] Seq=1 Ack=1 Win=8192 Len=0 TSval=1417717 TSec

Wireshark 抓到的 4 次挥手包：

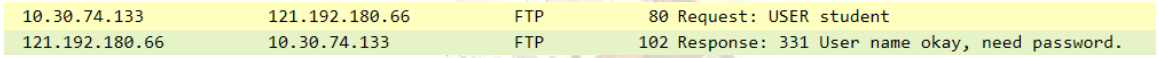


实验项 0：使用 Wireshark 等抓包工具监听网络上的数据流，定位出 FTP 连接建立过程；根据要求找出软件抓取到的对应数据帧，如包含登录密码的那一帧；

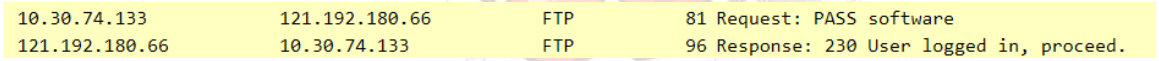
（1）服务就绪（应答码 220）



（2）用户 student 通过，请输入密码（应答码 331）



（3）密码 software 正确，登陆成功（应答码 230）



（4）收到退出指令，断开与 ftp 连接（应答码 221）

实验项 1：配置好实验环境；设置相应的过滤条件；分析 TCP/FTP 连接建立过程更加方便

1）本实验在上一个实验的 catchMac.cpp 代码的基础上进行改写

将过滤帧类型的数组修改，将过滤包类型改为 tcp

```
56    char packet_filter[] = "tcp";//————————过滤帧类型————————
```

2）可以看到本次连接分配的的本地端口为 65172

Destination	Protocol	Length	Info
121.192.180.66	TCP	74	65172 → 21
10.30.74.133	TCP	74	21 → 65172
121.192.180.66	TCP	66	65172 → 21

临时端口的分配与 tcp227 号报文的第 5、6 位有关，记为 n5、n6，则临时端口为 256*n5+n6

在编码中添加对应转换：

```
307                 int num = 0;        //Entering Passive Mode取第5位和第6位
308                 for (; i < header->len; i++) {
309                     if (num == 4) {
310                         char s1[10] = { 0 };
311                         char s2[10] = { 0 };
312                         int z = 0;
313                         while (data_header[i] != ',') {
314                             s1[z] = data_header[i];
315                             z++;
316                             i++;
317                         }
318                         num++;
319                         i++;
320                         z = 0;
321                         while (data_header[i] != ')') {
322                             s2[z] = data_header[i];
323                             z++;
324                             i++;
325                         }
326                         port = pstr_to_int(s1, s2);//修改端口
327                     }
328                     if (data_header[i] == ',') {
329                         num++;
330                     }
```

```
345     /*求被动打开时服务器的端口*/
346     int pstr_to_int(char* s1, char* s2) {
347         int n1 = atoi(s1);
348         int n2 = atoi(s2);
349         return 256 * n1 + n2;
350     }
```

实验项 2：编写程序，捕获到 TCP 连接建立的数据帧，并能够解析出目的端口、源端口。

（1）在输出语句中加入输出端口语句

```
/* 从网络字节顺序转换为主机字节顺序 */
sport = ntohs(th->sport);
dport = ntohs(th->dport);
```

```
printf("TCP.sport:%d\n", sport);
printf("TCP.dport:%d\n", dport);
```

（2）判断握手包和挥手包

A. 得到 6 个标识位的值

B. 识别握手包和挥手包

```
u_char s = th->sign;
unsigned short s1 = 0;//0010 0000,urg
unsigned short s2 = 0;//0001 0000,ack
unsigned short s3 = 0;//0000 1000,psh
unsigned short s4 = 0;//0000 0100,rst
unsigned short s5 = 0;//0000 0010,syn
unsigned short s6 = 0;//0000 0001,fin
if ((s & 32) == 32)//0010 0000
    s1 = 1;
if ((s & 16) == 16)//0001 0000
    s2 = 1;
if ((s & 8) == 8)//0000 1000
    s3 = 1;
if ((s & 4) == 4)//0000 0100
    s4 = 1;
if ((s & 2) == 2)//0000 0010
    s5 = 1;
if ((s & 1) == 1)//0000 0001
    s6 = 1;
printf("URG=%d\tACK=%d\tPSH=%d\tRST=%d\tSYN=%d\tFIN=%d\n", s1, s2, s3, s4, s5, s6);
fprintf(file, "URG=%d\tACK=%d\tPSH=%d\tRST=%d\tSYN=%d\tFIN=%d\n", s1, s2, s3, s4, s5, s6)
/*找握手包和挥手包,SYN==1是握手将建立连接，FIN==1是挥手将释放连接*/
```

```
if (s5 == 1) {
    if (s2 == 0) {//000010
        printf("ftp - 第一个握手包\n");
        fprintf(file, "ftp - 第一个握手包\n");
    }
    else { //010010
        printf("ftp - 第二个握手包\n");
        fprintf(file, "ftp - 第二个握手包\n");
    }
}
if (s6 == 1) {
    if (dport == 21) { //目的端口号为21
        printf("ftp - 挥手包，客户端向FTP服务端发出\n");
        fprintf(file, "ftp - 挥手包，客户端向FTP服务端发出\n");
    }
    else {
        printf("ftp - 挥手包，FTP服务端向客户端发出\n");
        fprintf(file, "ftp - 挥手包，FTP服务端向客户端发出\n");
    }
```

依据：①6 个标识位的值。以此判定是握手包还是挥手包

② FTP 服务端端口号是 21，大于 21 的端口号是客户端的端口号。以此判定包是客户端发送给服务端的还是服务端发送给客户端的

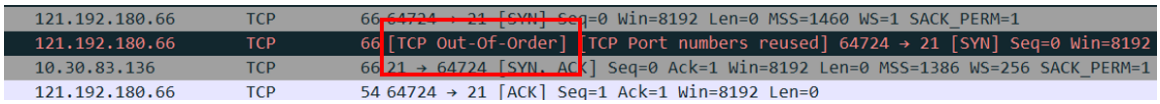
```
/*抓取ftp包 过滤21 and port*/
if ((sport != 21 && dport != 21) && (sport != port && dport != port))
    return;
```

备注部分的解答与实现：

①对遇到的异常情况分析；如为什么会出现 “TCP Out-of Order”；这部分看实际过程中遇到的异常：

使用 TCP 传输较大数据时，都会将数据分片，然后按照顺序依次传输。当 Wireshark 抓包时候，数据包标记为 TCP Out-Of-Order，意思是该数据包的发送顺序不对。这意味着网络状况不好。当然，Wireshark 有时会重传的包识别为乱序包。（在实际操作中没有遇到这个问题）

实际过程中异常：



②在程序的输出能明确指出握手包的前 2 个：

```
250     /*找握手包和挥手包,SYN==1是握手将建立连接，FIN==1是挥手将释放连接*/
251     if (s5 == 1) {
252         if (s2 == 0) {//000010
253             printf("ftp - 第一个握手包\n");
254             fprintf(file, "ftp - 第一个握手包\n");
255         }
256         else { //010010
257             printf("ftp - 第二个握手包\n");
258             fprintf(file, "ftp - 第二个握手包\n");
259         }
260     }
```

第一个握手包：

```
E:\VS Project\ComputerNetwork_Lab3\x64\Debug\ComputerNetwork_Lab4.exe
6. \Device\NPF_{E4887CA5-3292-41F2-A836-D781314417A9} (Microsoft)
7. \Device\NPF_{C606816D-49CB-4901-B61F-4241A21E347C} (Microsoft)
Enter the interface number (1-7):7

listening on Microsoft...
Time:08:13:46.404613 len:74
DEST MAC 40.fe.95.fe.80.1
SRC MAC 28.ea.1f.a5.7f.92
DEST IP:121.192.180.66
SRC IP:10.30.86.117
TCP.sport:65512
TCP.dport:21
6比特标志位:
URG=0   ACK=0   PSH=0   RST=0   SYN=1   FIN=0
ftp - 第一个握手包
Client--->Server
```

第二个握手包：

```
E:\VS Project\ComputerNetwork_Lab3\x64\Debug\ComputerNetwork_Lab4.exe

Time:08:13:46.445852 len:74
DEST MAC 28.ea.1f.a5.7f.92
SRC MAC 40.fe.95.fe.80.1
DEST IP:10.30.86.117
SRC IP:121.192.180.66
TCP.sport:21
TCP.dport:65512
6比特标志位：
URG=0   ACK=1   PSH=0   RST=0   SYN=1   FIN=0
ftp - 第二个握手包
Server--->Client
```

③在程序输出能明确指出客户端与 FTP 服务器、FTP 服务器与客户端断开 TCP 连接而发出的挥手包

```
261     if (s6 == 1) {
262         if (dport == 21) { //目的端口号为21
263             printf("ftp - 挥手包，客户端向FTP服务端发出\n");
264             fprintf(file, "ftp - 挥手包，客户端向FTP服务端发出\n");
265         }
266         else {
267             printf("ftp - 挥手包，FTP服务端向客户端发出\n");
268             fprintf(file, "ftp - 挥手包，FTP服务端向客户端发出\n");
269         }
270
271     }
```

```
E:\VS Project\ComputerNetwork_Lab3\x64\Debug\ComputerNetwork_Lab4.exe

Time:08:13:54.899569 len:66
DEST MAC 40.fe.95.fe.80.1
SRC MAC 28.ea.1f.a5.7f.92
DEST IP:121.192.180.66
SRC IP:10.30.86.117
TCP.sport:65512
TCP.dport:21
6比特标志位：
URG=0   ACK=1   PSH=0   RST=0   SYN=0   FIN=1
ftp - 挥手包，客户端向FTP服务端发出
Client--->Server
```

```
E:\VS Project\ComputerNetwork_Lab3\x64\Debug\ComputerNetwork_Lab4.exe
6比特标志位:
URG=0   ACK=1   PSH=0   RST=0   SYN=0   FIN=0
Server--->Client

Time:08:13:54.912572 len:66
DEST MAC 28.ea.1f.a5.7f.92
SRC MAC 40.fe.95.fe.80.1
DEST IP:10.30.86.117
SRC IP:121.192.180.66
TCP.sport:21
TCP.dport:65512
6比特标志位:
URG=0   ACK=1   PSH=0   RST=0   SYN=0   FIN=1
ftp - 挥手包，FTP服务端向客户端发出
Server--->Client
```

实验项 3：编写程序，捕获到 FTP 连接建立的数据帧，并能够解析出对应的 FTP 命令

（1）获取命令

（2）判断应答类型并给出相应操作

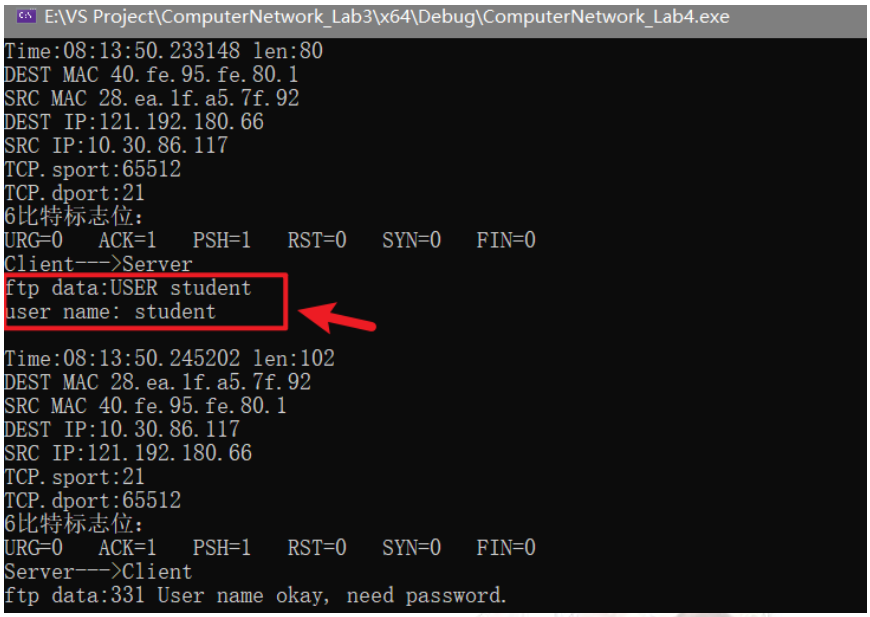
依据：看口令开头是什么判断口令类型

```
295         /*读取ftp指令*/
296         if (sport == 21 || dport == 21) {
297             char str[100] = { 0 };
298
299             int j = 0;
300
301             for (int i = 34 + tcp_len * 4; i < header->len; i++) {
302                 if (data_header[i] == ' ') {
303                     if (strcmp(str, "USER") == 0) {
304                         printf("user name:");
305                         fprintf(file, "user name:");
306                         for (; i < header->len; i++) {
307                             printf("%c", data_header[i]);
308                             fprintf(file, "%c", data_header[i]);
309                         }
310                     }
311                     else if (strcmp(str, "PASS") == 0) {
312                         printf("password:");
313                         fprintf(file, "password:");
314                         for (; i < header->len; i++) {
315                             printf("%c", data_header[i]);
316                             fprintf(file, "%c", data_header[i]);
317                         }
318                     }
319                     else if (strcmp(str, "227") == 0) {
320                         int num = 0;        //Entering Passive Mode取第5位和第6位
321                         for (; i < header->len; i++) {
322                             if (num == 4) {
323                                 char s1[10] = { 0 };
324                                 char s2[10] = { 0 };
325                                 int z = 0;
326                                 while (data_header[i] != ',') {
327                                     s1[z] = data_header[i];
328                                     z++;
329                                     i++;
330                                 }
331                                 num++;
332                                 i++;
333                                 z = 0;
334                                 while (data_header[i] != ')') {
335                                     s2[z] = data_header[i];
```

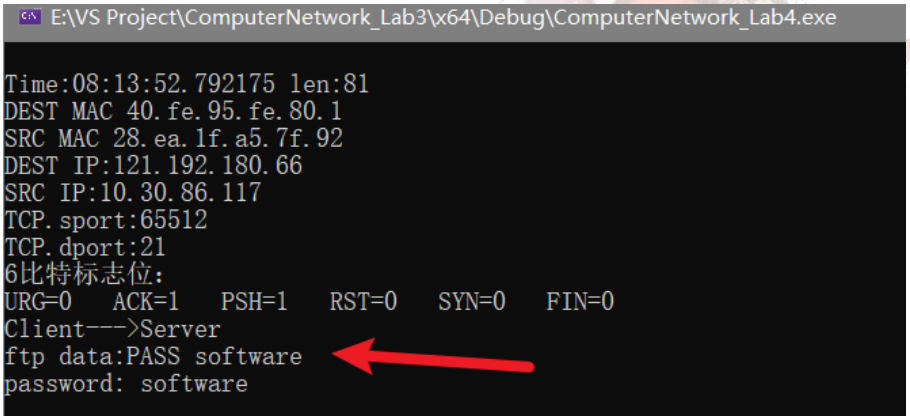
备注部分的实现：

①知道常用 FTP 命令的含义：如 USER，同时打印出对应的用户名；如 PASS，同时打印出对应密码；

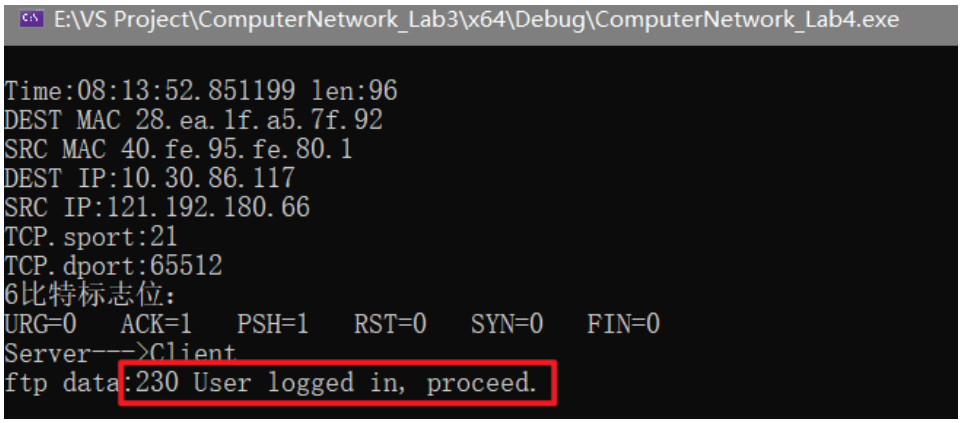
USER 命令，打印出对应的用户名：



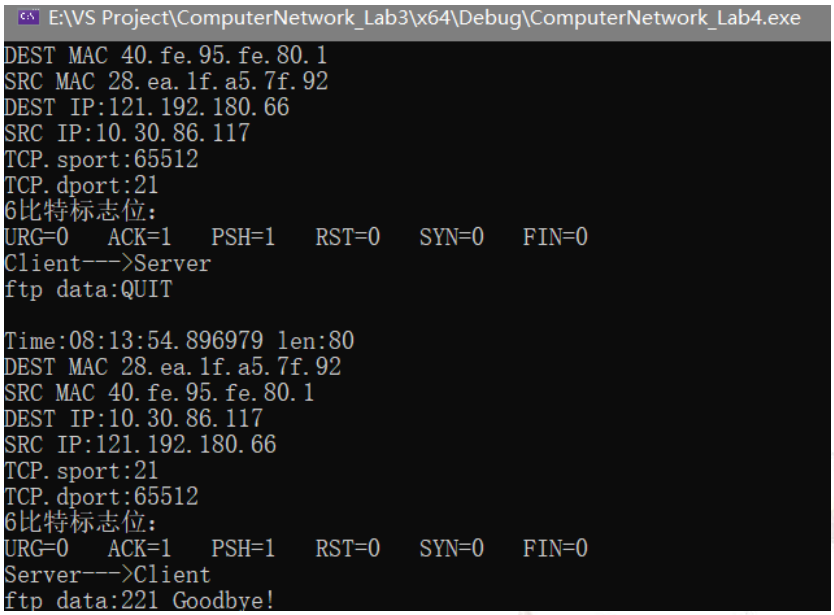
PASS 命令，打印出对应密码：



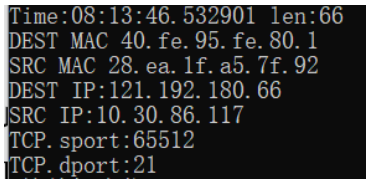
230 代表登陆成功



QUIT 命令，表示断开连接



②对于数据连接，解析出对应的客户端、 FTP 服务器使用的端口号（或者能从你的输出结果中找到判断依据）



通过谁向谁送和源端口、目的端口号确认。

判断依据：若是 FTP 服务端向客户端发送，则源端口号（sport）为 FTP 服务器使用的端口号，目的端口号（dport）为客户端使用的端口号。反之同理。

```c
/*服务器与客户端*/
if (dport == 21 || dport == port) {
    printf("Client--->Server\n");
    fprintf(file, "Server--->Client\n");
}
else if (sport == 21 || sport == port) {
    printf("Server--->Client\n");
    fprintf(file, "Server--->Client\n");
}
```

此时验证：

FTP 服务器使用的端口号为源端口号（sport）：21

客户端使用的端口号为目的端口号（dport）：65512

实验项 4：实验已经实现写入文件功能，具体查看 data.txt 文件

```c
    /* 对应输出文件 */
    fprintf(file, "DEST MAC %x.%x.%x.%x.%x.%x \n", mac_header[0], mac_header[1], mac_header[2], mac_header[3], mac_header[4], mac_header[5]
    fprintf(file, "SRC MAC %x.%x.%x.%x.%x.%x \n", mac_header[6], mac_header[7], mac_header[8], mac_header[9], mac_header[10], mac_header[1
    fprintf(file, "DEST IP:%d.%d.%d.%d\n",
        ih->daddr.byte1,
        ih->daddr.byte2,
        ih->daddr.byte3,
        ih->daddr.byte4
    );
    fprintf(file, "SRC IP:%d.%d.%d.%d\n",
        ih->saddr.byte1,
        ih->saddr.byte2,
        ih->saddr.byte3,
        ih->saddr.byte4
    );
    fprintf(file, "TCP.sport:%d\n", sport);
    fprintf(file, "TCP.dport:%d\n", dport);
    fprintf(file, "6比特标志位：\n");
```

data.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
6比特标志位:
URG=0   ACK=0   PSH=0   RST=0   SYN=1   FIN=0
ftp - 第一个握手包
Server--->Client

Time:08:13:46.445852 len:74
DEST MAC 28.ea.1f.a5.7f.92
SRC MAC 40.fe.95.fe.80.1
DEST IP:10.30.86.117
SRC IP:121.192.180.66
TCP.sport:21
TCP.dport:65512
6比特标志位:
URG=0   ACK=1   PSH=0   RST=0   SYN=1   FIN=0
ftp - 第二个握手包
Server--->Client
```

第 5 行，第 20 列　100%　Windows (CRLF)　ANSI

4 实验总结

通过这次实验，我利用 Wireshark 和 WinPCAP 监听并解析 FTP 命令。进一步地，利用 C 语言文件读写函数将侦听到的内容按格式保存至 txt 文件中，通过这一次实验，我对其中的知识了解的更加深刻了。

以下是参考的文章：

用 winpacp 监听并分析 FTP 协议并记录 IP、用户名、密码和登陆是否成功

https://blog.csdn.net/qq_38782152/article/details/80291329