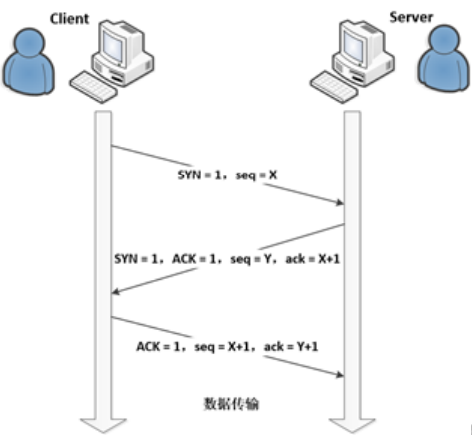
TCP报文捕获与三次握手分析（gcc实现）

## 一、实验思路

对于TCP报文的捕获，因为要捕获TCP报文且对三次握手的过程进行分析，所以在创建socket时要将第一个参数指定为AF\_PACKET，是因为AF\_PACKET可以捕获到自己发的TCP包，而AF\_INET捕获不到。先将TCP报文信息存储提前定义好的buffer中，再将该buffer格式化为TCP的格式，可以使用linux下定义好的TCP结构体和IP结构体。捕捉到TCP报文后，跳过以太网帧的头部和IP头部即可得到TCP报文，然后依次对结构体中的各字段进行分析。对于三次握手过程的分析对TCP结构体中的seq字段和seq\_ack字段进行分析即可，seq\_ack是对对方seq+1的确认。如果对应的seq\_ack为对方seq的值加1，则表明双方建立了TCP三次握手，同时使用wireshark抓包，如果用程序抓到的TCP三次握手的信息和wireshark抓包的信息相同则表明实验成功。



## 二、实验环境及步骤

1、Linux下gcc环境

2、gcc -o Catchtcp Catchtcp.c

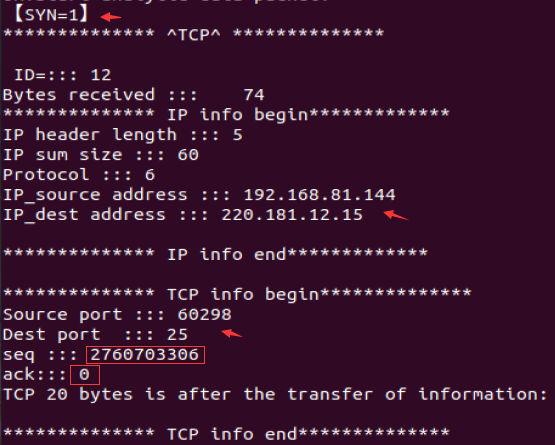
3、./Catchtcp 打开浏览器刷新，同时使用wireshark抓包

4、对比两者信息是否相同

## 三、实验结果及分析

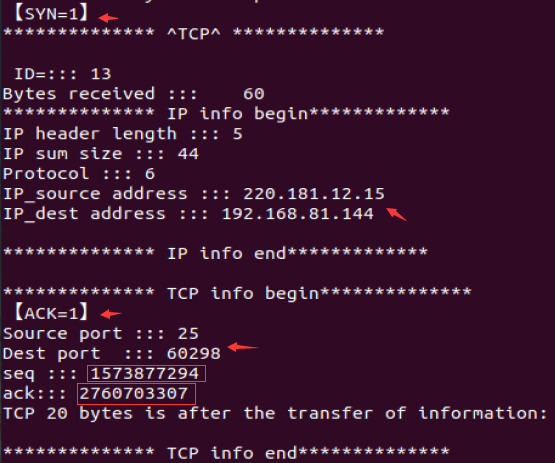
**1、使用Catchtcp.c程序抓包的结果**

① 第一次握手：



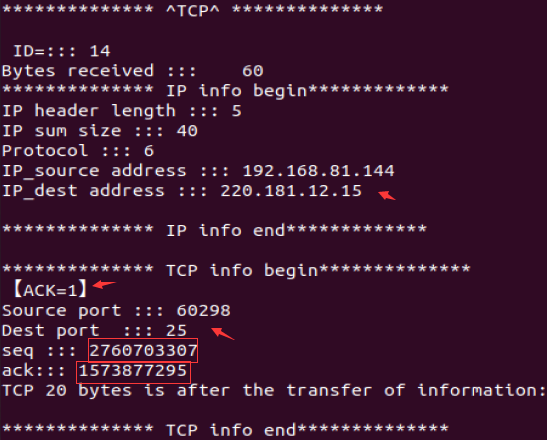
第一次握手由Client（192.168.81.144）建立，在本次实验中为我的虚拟机，此时SYN=1，seq为自己生成的2760703306，第一次握手无ack，默认为0，源端口为60298，目的端口为25；

② 第二次握手：



第二次握手由Server（220.181.12.15）建立，此时SYN=1，ACK=1，seq=1573877294，ack=2760703307是对Clinet的seq+1的确认，源端口为25，目的端口为60298；

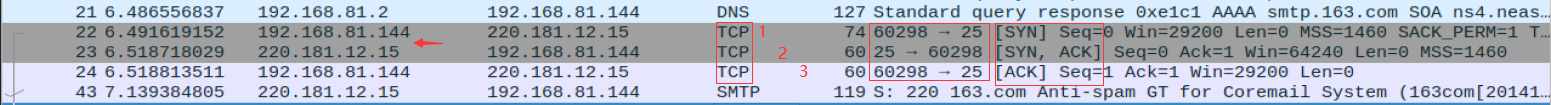
③ 第三次握手：



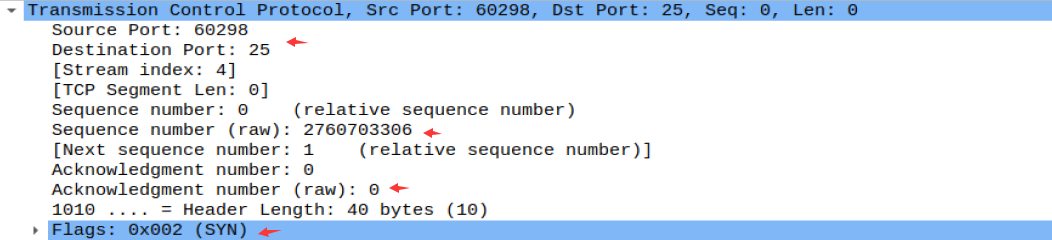
第三次握手由Client建立，只需ACK=1，且seq+1，即seq=2760703307，ack为对Server的seq+1确认，即ack=1573877295，源端口为60298，目的端口为25，至此三次握手建立成功！

**2、使用wireshark抓包的结果**

① 三次握手报文

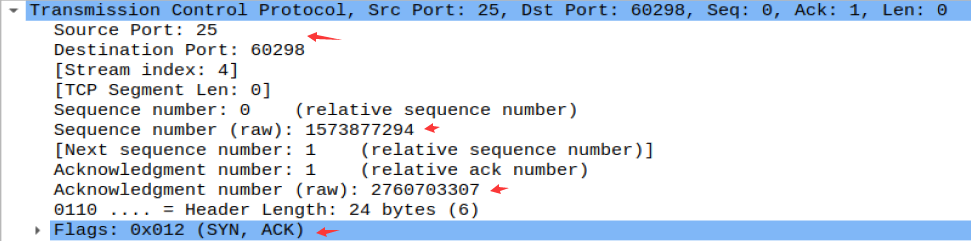


② 第一次握手：



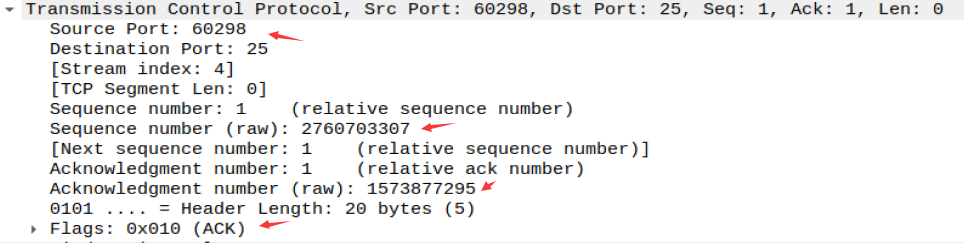
可以看出第一次握手Client的seq以及源端口号和目的端口号和程序捕获的完全一致，且也只是SYN;

③ 第二次握手：



seq、端口号、ack以及SYN和ACK与程序捕获一致

③ 第三次握手：



seq、端口号、ack以及ACK与程序捕获一致

## 四、实验代码

|  |
| --- |
| /\*\* TCP和IP结构体 \*\*/  // 首先，定义IP首部结构体：  **typedef** struct iphdr**{**  unsigned char h\_lenver**;** //4 位IP版本号+4位首部长度  unsigned char tos**;** //8位服务类型TOS  unsigned short total\_len**;** //16位总长度（字节）  unsigned short ident**;** //1 6位标识  unsigned short frag\_and\_flags**;** //3位标志位+13位偏移位, 用于IP分片  unsigned char ttl**;** //8位生存时间TTL  unsigned char proto**;** //8位协议号（TCP, UDP或其他）  unsigned short checksum**;** //16位IP首部校验和  unsigned int sourceIP**;** //32位源IP地址  unsigned int destIP**;** //32位目的IP地址  **}**IP\_HEADER**;**  // 其次定义TCP首部结构体  **typedef** struct tcphdr  **{**  unsigned short src\_port**;** //源端口号  unsigned short dst\_port**;** //目的端口号  unsigned int seq\_no**;** //序列号  unsigned int ack\_no**;** //确认号  #if LITTLE\_ENDIAN  unsigned char reserved\_1**:**4**;** //保留6位中的4位首部长度  unsigned char thl**:**4**;** //tcp头部长度  unsigned char flag**:**6**;** //6位标志  unsigned char reseverd\_2**:**2**;** //保留6位中的2位  #else  unsigned char thl**:**4**;** //tcp头部长度  unsigned char reserved\_1**:**4**;** //保留6位中的4位首部长度  unsigned char reseverd\_2**:**2**;** //保留6位中的2位  unsigned char flag**:**6**;** //6位标志  #endif  unsigned short wnd\_size**;** //16位窗口大小  unsigned short chk\_sum**;** //16位TCP检验和  unsigned short urgt\_p**;** //16为紧急指针  **}**TCP\_HEADER**;**  /\*\* 总体代码 \*\*/  #include <stdio.h>  #include <sys/socket.h>  #include <netinet/in.h>  #include <arpa/inet.h>  #include <netinet/ip.h>  #include <netinet/udp.h>  #include <netinet/tcp.h>  #include<netinet/if\_ether.h>  #include <sys/types.h>  int main**()** **{**  int sock**,** bytes\_recieved**,** fromlen**,**n**,**id**=**1**,**on**=**1**,**s**;**  unsigned char buffer**[**65535**];**  struct sockaddr\_in from**;**  struct ip **\***ip**;**  struct tcphdr **\***tcp**;**  struct ethhdr **\***ethh**;**    /\* 建立原始TCP包方式收到 以太网帧头+IP+TCP信息包 \*/  sock **=** socket**(**AF\_PACKET**,** SOCK\_RAW**,** htons**(**ETH\_P\_ALL**));**  printf**(**" The IPPROTO\_TCP value is %d \n"**,**IPPROTO\_TCP**);**  **if** **(**sock**>**0**)** **{**  printf**(**"Prepare caught!! \n"**);**  **}** **else**  **return(**0**);**  id**=**1**;**  **while(**1**){**  bytes\_recieved **=** recvfrom**(**sock**,** buffer**,** **sizeof(**buffer**),**0**,NULL,NULL);**//int bytes\_recieved  **if** **(**bytes\_recieved**>**0**)** **{**  printf**(**"OK!Start analytic data packet!\n"**);**  ethh **=** **(**struct ethhdr**\*)**buffer**;**  **if** **(!(**htons**(**ethh**->**h\_proto**)** **==** ETH\_P\_IP**))**  **{**  **continue;**  **}**  ip **=** **(**struct ip **\*)(**buffer**+**14**);**  /\* tcp从 buffer + 14+(4\*ip->ip\_hl) 地址处开始 \*/  tcp **=** **(**struct tcphdr **\*)(**buffer **+**14**+** **(**4**\***ip**->**ip\_hl**));**  **if** **(**tcp**->**syn**)**  **{**  printf**(**"【SYN】\n"**);** // tcp的syn标志为1表示为前两次握手，再根据ack判断是第一次握手还是第二次握手；  **}**  **if** **(**ntohs**(**tcp**->**dest**)!=**23 **)** **{** /\*23为Telnet端口，也可改为其它端口\*/  printf**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\* ^TCP^ \*\*\*\*\*\*\*\*\*\*\*\*\*\* \n"**);**  printf**(**"\n ID=::: %d\n"**,**id**);**  printf**(**"Bytes received ::: %5d\n"**,**bytes\_recieved**);**  printf**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\* IP info begin\*\*\*\*\*\*\*\*\*\*\*\*\* \n"**);**  printf**(**"IP header length ::: %d\n"**,**ip**->**ip\_hl**);**  printf**(**"IP sum size ::: %d\n"**,**ntohs**(**ip**->**ip\_len**));**  printf**(**"Protocol ::: %d\n"**,**ip**->**ip\_p**);**  printf**(**"IP\_source address ::: %s \n"**,**inet\_ntoa**(**ip**->**ip\_src**));**  printf**(**"IP\_dest address ::: %s \n"**,**inet\_ntoa**(**ip**->**ip\_dst**));**  printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* IP info end\*\*\*\*\*\*\*\*\*\*\*\*\* \n"**);**  printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* TCP info begin\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n"**);**  **if** **(**tcp**->**ack**)**  **{**  printf**(**"【ACK】\n"**);**  **}**  printf**(**"Source port ::: %d\n"**,**ntohs**(**tcp**->**source**));**  printf("Dest port ::: %d\n",ntohs(tcp->dest));  printf("seq ::: %u\n",ntohl(tcp->seq));  printf("ack::: %u\n",ntohl(tcp->ack\_seq));  printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* TCP info end\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  id=id+1;  } /\*>23 end \*/  } /\*>0 end \*/  } /\*while end \*/  } |