AMINA QADEER

CE-42-A

359607

COMPUTER NETWORKS

ASSIGNMENT 01

****

Use the full-duplex TCP chat codes you have developed to make client/server communication sessions. Use Wireshark to capture different types of messages (SYN, SYN-ACK, Data, ACK, and Fin messages etc.) during the client/server chat. Please highlight (include) the following things in your report:

1) Apply CORRECT filter on Wireshark to only display the messages exchanged between your TCP client/server applications. Also show a snippet of the filter in assignment submission.

2) Display Wireshark screenshots of All TYPES of messages captured during your chat, and highlight the values in the TCP header, specifying a particular message types.

3) For each message type, display corresponding lines of code, that triggered that message!

****

FULL -DUPLEX COMMUNICATION

****

SERVER	CLIENT
CODE:	**CODE:**
#include<sys/types.h> #include<sys/socket.h> #include<arpa/inet.h> #include<netinet/in.h> #include<string.h> #include<iostream> #include<pthread.h> //Pthreads header include file #include<unistd.h> #include<stdlib.h>	#include<sys/types.h> #include<arpa/inet.h> #include<sys/socket.h> #include<netinet/in.h> #include<iostream> #include<pthread.h> //Pthreads header include file #include<unistd.h> #include<cstring> #define THREAD_COUNT 2 //Controls the

```
#include<cstring>
#include<string>
#define THREAD_COUNT 2
//Controls the number of threads
char buffer[100];
char buf[100] = "hello server here";
char buf1[100];
void* SendM(void *rank);
void* RcvM(void *rank);
using namespace std;
int main()
{
int
fd=socket(AF_INET,SOCK_STREAM,
0);
```

```
if(fd==-1)
{
perror("Socket not made");
exit(-1);
}
struct sockaddr_in addr;
addr.sin_addr.s_addr=INADDR_ANY;
addr.sin_family=AF_INET;
addr.sin_port=htons(80);
if(bind(fd,(struct
sockaddr*)&addr,sizeof(addr))==-1)
{
perror("error-BINDING FAILED ON
SOCKET");
exit(-1);
}
int backlog=10;
if(listen(fd,backlog)==-1)
{
perror("listen failed on socket:");
exit(-1);
}
int connfd;
struct sockaddr_in cliaddr;
socklen_t cliaddr_len=sizeof(cliaddr);
connfd=accept(fd,(struct
sockaddr*)&cliaddr,&cliaddr_len);
if(connfd==-1){
cout<<"error";
exit(-1);
}
```

```
number of threads
char buffer[100];
char buf1[100];
void* SendCM(void *rank);
void* RcvCM(void *rank);
using namespace std;
int main()
{
int fd=socket(AF_INET,SOCK_STREAM,0);
if(fd==-1)
{
perror("socket creation failed");
exit(-1);
}
struct sockaddr_in s_addr;
s_addr.sin_family=AF_INET;
s_addr.sin_port=htons(80);
inet_aton("127.0.0.1",&s_addr.sin_addr);
connect(fd,(struct
sockaddr*)&s_addr,sizeof(s_addr));
pthread_t
thread_handles[THREAD_COUNT];
//Create THREAD_COUNT number of
Pthreads
for (long thread=0;
thread<THREAD_COUNT; thread++) {
pthread_create(&thread_handles[thread],
NULL, SendCM, (void*)&fd);
pthread_create(&thread_handles[thread],
NULL, RcvCM, (void*)&fd);
}
cout<<"Hello from the Client
chatbox\n"<<endl;
//Join all created Pthreads
for (long thread=0;
thread<THREAD_COUNT; thread++) {
pthread_join(thread_handles[thread], NULL);
}
//This line executes after completion of joined
threads
cout<<"Client chat room closing\n";
return 0;
```

```
}
//Thread function, Check the return type and
parameter list
void* SendCM(void *rank) {
```

```
pthread_t
thread_handles[THREAD_COUNT];
//Create THREAD_COUNT number of
Pthreads
for (long thread=0;
thread<THREAD_COUNT; thread++)
{
pthread_create(&thread_handles[thread
], NULL, SendSM, (void*)&connfd);
pthread_create(&thread_handles[thread
], NULL, RcvSM, (void*)&connfd);
}
cout<<"Hello from the server
chatbox\n"<<endl;
//Join all created Pthreads
for (long thread=0;
thread<THREAD_COUNT; thread++)
{
pthread_join(thread_handles[thread],
NULL);
}
//This line executes after completion of
joined threads
cout<<"Server chat room closing\n";
return 0;
}
```

```
//Thread function, Check the return
type and parameter list
void* SendSM(void *rank) {
long connfds = *((int*) rank);
while (1) {
fgets(buffer,100,stdin);
send(connfds,buffer,strlen(buffer),0);
// sleep(2);s
}
return NULL;
}
void* RcvSM(void *rank) {
long connfdr = *((int*) rank);
while (1) {
int check = recv(connfdr,buffer,100,0);
if (check < 1){
cout << "Client disconnected" << endl;
break;
}
cout << "Message rcvd on server is: "
```

```
long connfds = *((int*) rank);
while (1) {
fgets(buffer,100,stdin);
send(connfds,buffer,strlen(buffer),0);
//sleep(2);
}
return NULL;
}
void* RcvCM(void *rank) {
long connfdr = *((int*) rank);
while (1) {
int check = recv(connfdr,buf1,100,0);
if (check < 1){
cout << "Server Disconnected" << endl;
break;
}
cout << "Message rcvd on client is: " << buf1;
//sleep(2);
}
return NULL;
}
```

```
<< buffer;
// sleep(2);
}
return NULL;
}
```

1.SOCKET: TCP

```
int main() {

        int fd = socket(AF_INET, SOCK_STREAM, 0);
        if (fd == -1) {
                perror("Socket Creation failed\n");
                return -1;
        }
```

2.BIND

Socket address association

```
        struct sockaddr_in addr;

        addr.sin_addr.s_addr    = INADDR_ANY;
        addr.sin_family = AF_INET;
        addr.sin_port           = htons(SERVER_PORT_NO);
if (bind(fd, (struct sockaddr*) &addr, sizeof(addr)) == -1) {
                perror("Bind failed on socket\n");
                return -1;
        }
```

3.LISTEN

```
int backlog = 1;
if (listen(fd, backlog) == -1) {
        perror("Listen Failed on server: \n");
        return -1;
}
```

4.ACCEPT

```
int connfd;
struct sockaddr_in cliaddr;
socklen_t cliaddr_len;
connfd = accept(fd, (struct sockaddr *) &cliaddr, &cliaddr_len);

if (connfd <=0 ) {
        perror("accept failed on socket: ");
}
```

5.CLOSE

1.SOCKET:TCP

```
#define SERVER_PORT_NO 80

using namespace std;

int main() {

        int fd = socket(AF_INET, SOCK_STREAM, 0);
        if (fd == -1) {
                perror("Socket Creation failed\n");
                return -1;
        }
```

2.CONNECT:

```
struct sockaddr_in s_addr;

s_addr.sin_family       = AF_INET;
s_addr.sin_port = htons(80);
inet_aton("127.0.0.1", &s_addr.sin_addr);
if (connect(fd, (struct sockaddr *) &s_addr, sizeof(s_addr)) == -1) {
        perror("Socket Connect failed\n");
        return -1;
}
```

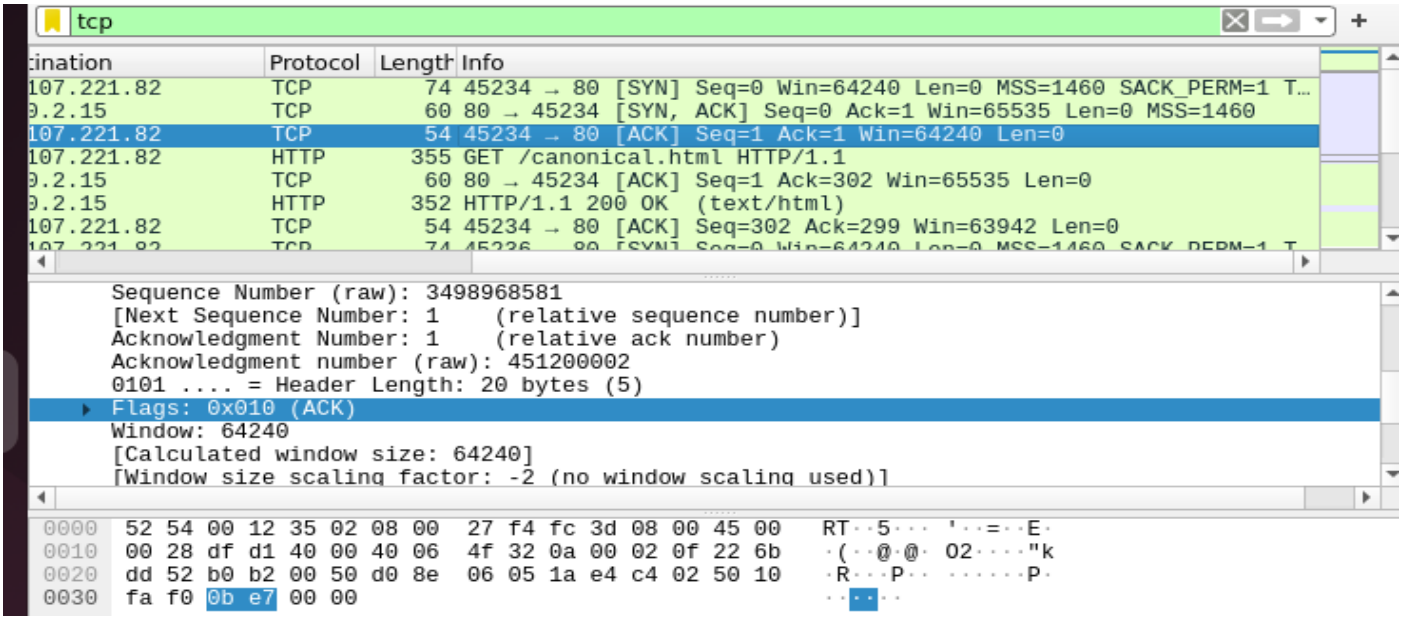
3.DATA

```
char buffer[] = "Hello, This is a Sample Message from Client";

send(fd, buffer, strlen(buffer), 0);


cout<<"Data Sent, Client Exiting!"<<endl;
return 0;
```

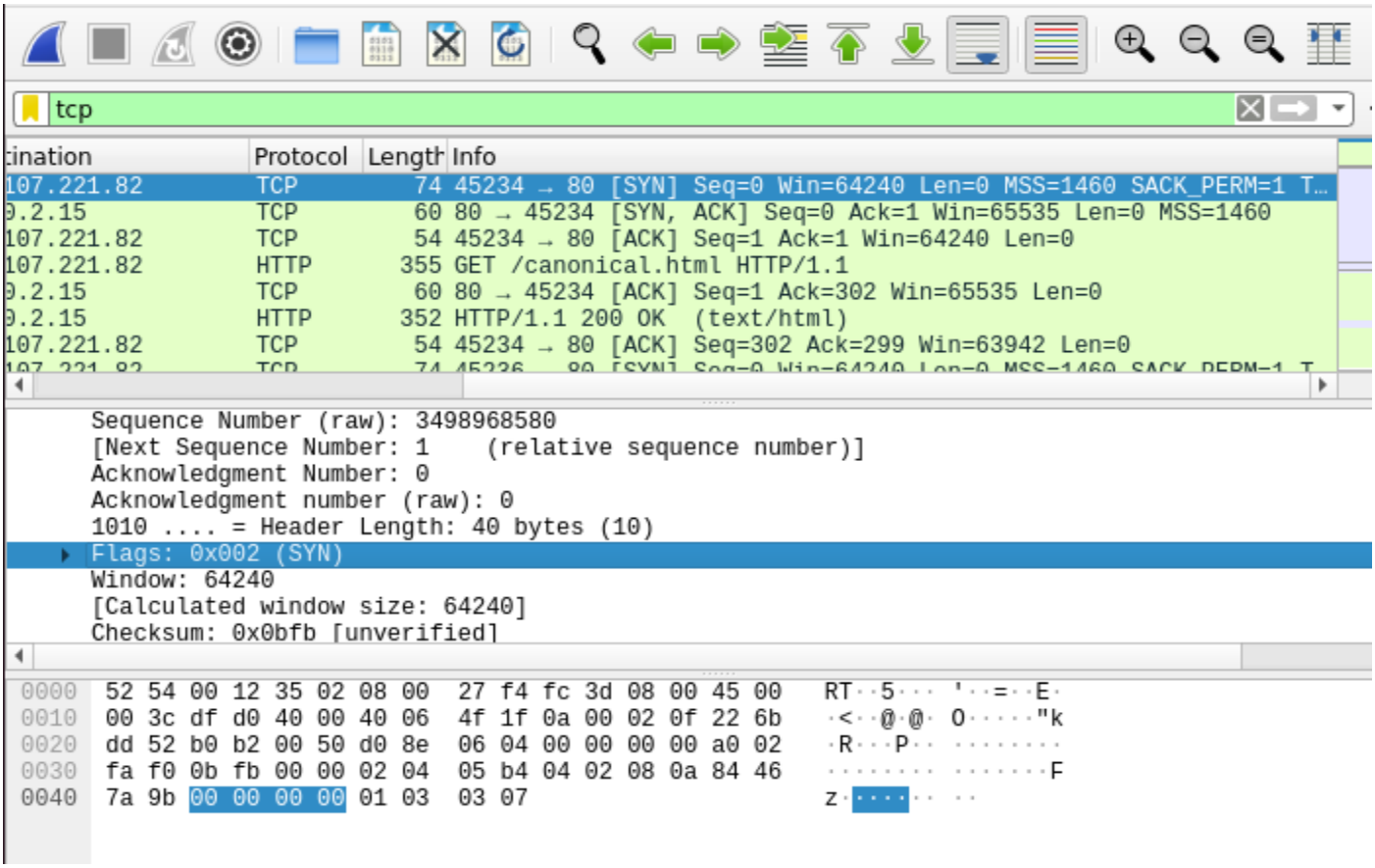
```
    char buffer[100];
    bzero(buffer, 100);
    recv(connfd, buffer, 100, 0);

    cout<<"server received message : "<<buffer;
    cout<<"\nServer Exiting!\n";

    return 0;
```

WIRESHARK DISPLAY:

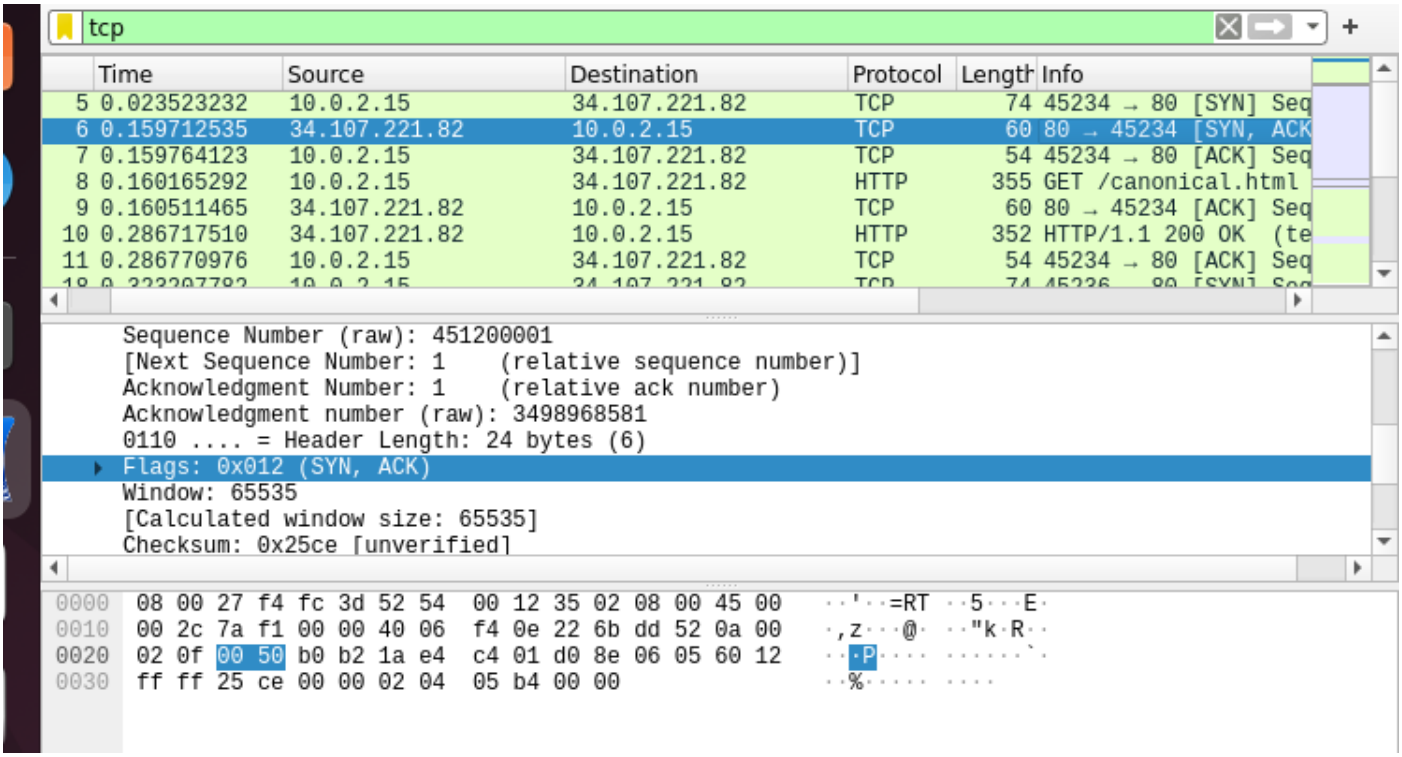
ACK:



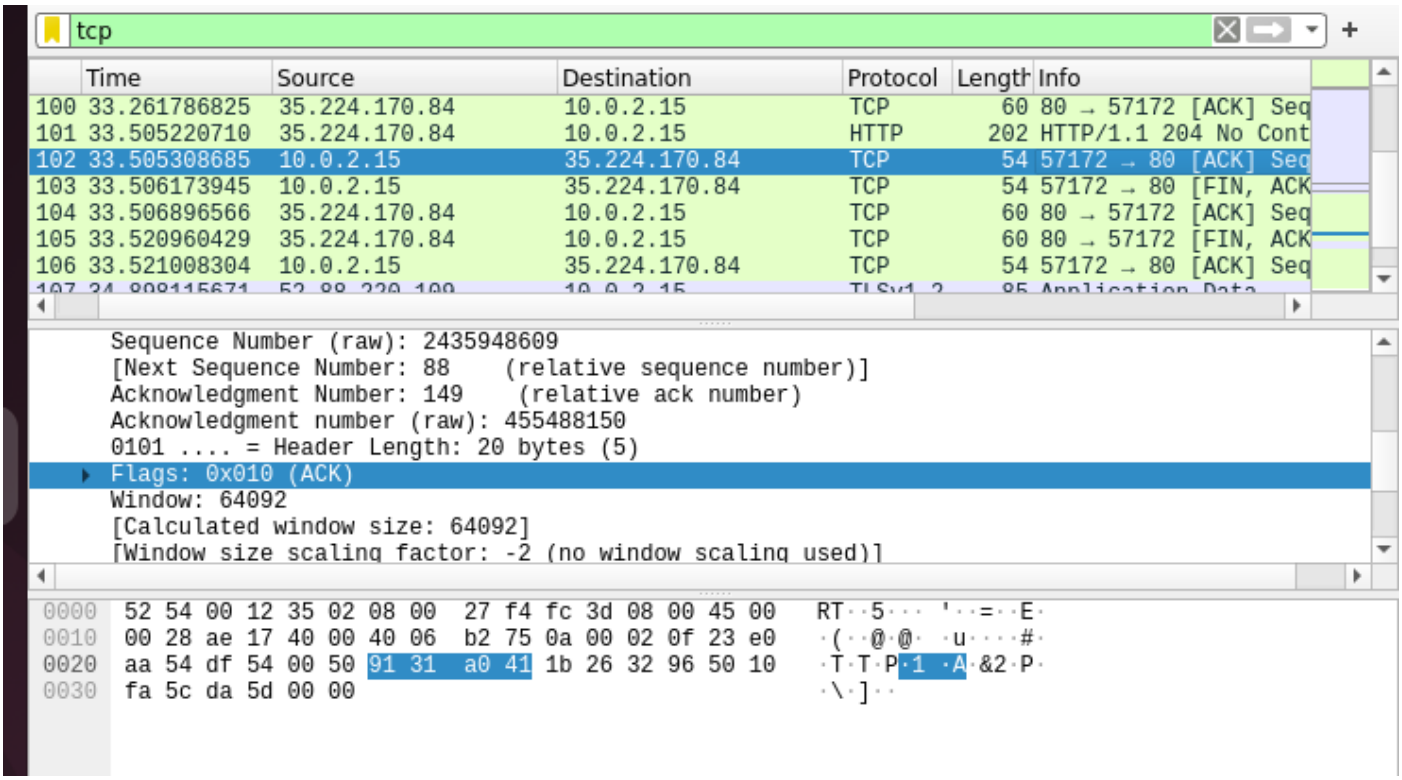
SYN:


tcp
ination Protocol Length Info
107.221.82 TCP 74 45234 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…
9.2.15 TCP 60 80 → 45234 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
107.221.82 TCP 54 45234 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
107.221.82 HTTP 355 GET /canonical.html HTTP/1.1
9.2.15 TCP 60 80 → 45234 [ACK] Seq=1 Ack=302 Win=65535 Len=0
9.2.15 HTTP 352 HTTP/1.1 200 OK (text/html)
107.221.82 TCP 54 45234 → 80 [ACK] Seq=302 Ack=299 Win=63942 Len=0
Sequence Number (raw): 3498968580
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1010 .... = Header Length: 40 bytes (10)
Flags: 0x002 (SYN)
Window: 64240
[Calculated window size: 64240]
Checksum: 0x0bfb [unverified]
0000 52 54 00 12 35 02 08 00 27 f4 fc 3d 08 00 45 00
0010 00 3c df d0 40 00 40 06 4f 1f 0a 00 02 0f 22 6b
0020 dd 52 b0 b2 00 50 d0 8e 06 04 00 00 00 00 a0 02
0030 fa f0 0b fb 00 00 02 04 05 b4 04 02 08 0a 84 46
0040 7a 9b 00 00 00 00 01 03 03 07

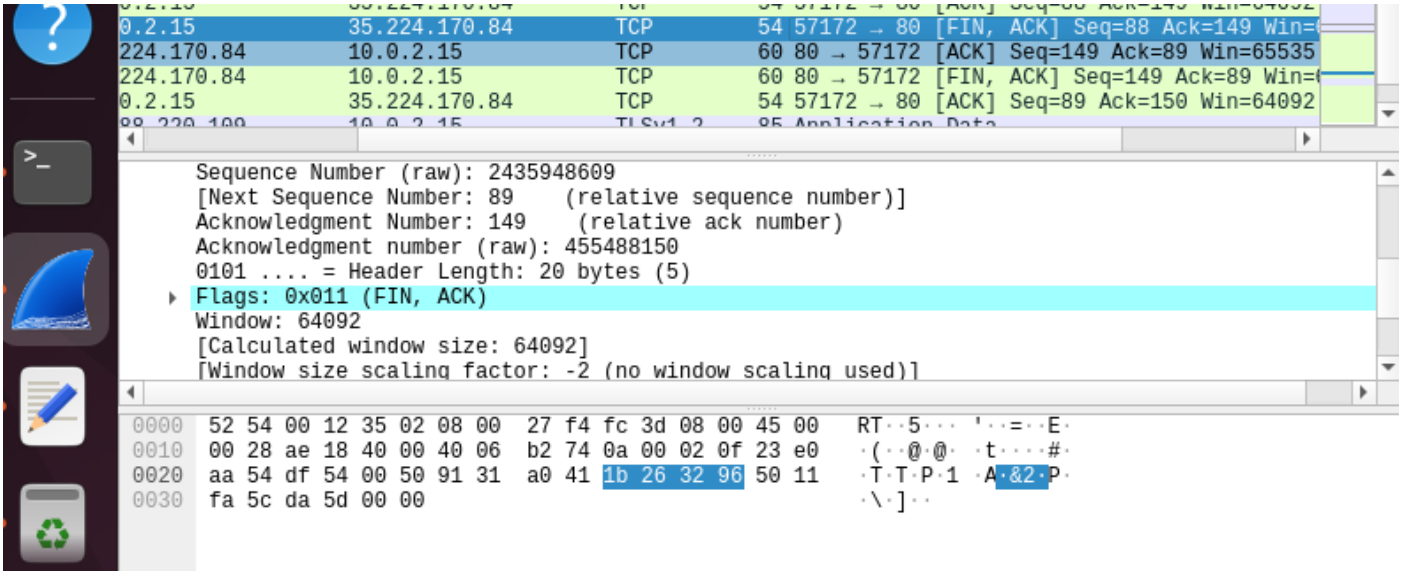
SYN-ACK


tcp
Time Source Destination Protocol Length Info
5 0.023523232 10.0.2.15 34.107.221.82 TCP 74 45234 → 80 [SYN] Seq
6 0.159712535 34.107.221.82 10.0.2.15 TCP 60 80 → 45234 [SYN, ACK
7 0.159764123 10.0.2.15 34.107.221.82 TCP 54 45234 → 80 [ACK] Seq
8 0.160165292 10.0.2.15 34.107.221.82 HTTP 355 GET /canonical.html
9 0.160511465 34.107.221.82 10.0.2.15 TCP 60 80 → 45234 [ACK] Seq
10 0.286717510 34.107.221.82 10.0.2.15 HTTP 352 HTTP/1.1 200 OK (te
11 0.286770976 10.0.2.15 34.107.221.82 TCP 54 45234 → 80 [ACK] Seq
Sequence Number (raw): 451200001
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 3498968581
0110 .... = Header Length: 24 bytes (6)
Flags: 0x012 (SYN, ACK)
Window: 65535
[Calculated window size: 65535]
Checksum: 0x25ce [unverified]
0000 08 00 27 f4 fc 3d 52 54 00 12 35 02 08 00 45 00
0010 00 2c 7a f1 00 00 40 06 f4 0e 22 6b dd 52 0a 00
0020 02 0f 00 50 b0 b2 1a e4 c4 01 d0 8e 06 05 60 12
0030 ff ff 25 ce 00 00 02 04 05 b4 00 00

ACK:


tcp
Time Source Destination Protocol Length Info
100 33.261786825 35.224.170.84 10.0.2.15 TCP 60 80 → 57172 [ACK] Seq
101 33.505220710 35.224.170.84 10.0.2.15 HTTP 202 HTTP/1.1 204 No Cont
102 33.505308685 10.0.2.15 35.224.170.84 TCP 54 57172 → 80 [ACK] Seq
103 33.506173945 10.0.2.15 35.224.170.84 TCP 54 57172 → 80 [FIN, ACK
104 33.506896566 35.224.170.84 10.0.2.15 TCP 60 80 → 57172 [ACK] Seq
105 33.520960429 35.224.170.84 10.0.2.15 TCP 60 80 → 57172 [FIN, ACK
106 33.521008304 10.0.2.15 35.224.170.84 TCP 54 57172 → 80 [ACK] Seq
Sequence Number (raw): 2435948609
[Next Sequence Number: 88 (relative sequence number)]
Acknowledgment Number: 149 (relative ack number)
Acknowledgment number (raw): 455488150
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 64092
[Calculated window size: 64092]
[Window size scaling factor: -2 (no window scaling used)]
0000 52 54 00 12 35 02 08 00 27 f4 fc 3d 08 00 45 00
0010 00 28 ae 17 40 00 40 06 b2 75 0a 00 02 0f 23 e0
0020 aa 54 df 54 00 50 91 31 a0 41 1b 26 32 96 50 10
0030 fa 5c da 5d 00 00

FIN:


0.2.15 35.224.170.84 TCP 54 57172 → 80 [FIN, ACK] Seq=88 Ack=149 Win=
224.170.84 10.0.2.15 TCP 60 80 → 57172 [ACK] Seq=149 Ack=89 Win=65535
224.170.84 10.0.2.15 TCP 60 80 → 57172 [FIN, ACK] Seq=149 Ack=89 Win=
0.2.15 35.224.170.84 TCP 54 57172 → 80 [ACK] Seq=89 Ack=150 Win=64092
Sequence Number (raw): 2435948609
[Next Sequence Number: 89 (relative sequence number)]
Acknowledgment Number: 149 (relative ack number)
Acknowledgment number (raw): 455488150
0101 .... = Header Length: 20 bytes (5)
Flags: 0x011 (FIN, ACK)
Window: 64092
[Calculated window size: 64092]
[Window size scaling factor: -2 (no window scaling used)]
0000 52 54 00 12 35 02 08 00 27 f4 fc 3d 08 00 45 00
0010 00 28 ae 18 40 00 40 06 b2 74 0a 00 02 0f 23 e0
0020 aa 54 df 54 00 50 91 31 a0 41 1b 26 32 96 50 11
0030 fa 5c da 5d 00 00