Lab Write-Up: Client-Server TCP Chat with TCP Packet Spoofing

Gabriel Velez

Part A: Setting up the Client-Server Chat

1. **Implement the Server:**
   - Write a C program for the server to listen on a specified port for incoming TCP connections.
   - The server should be able to handle multiple client connections and relay messages received from one client to all other connected clients.

2. **Implement the Client:**
   - Write a C program for the client to connect to the server using TCP.
   - The client should be able to send messages to the server and display messages received from the server.

3. **Testing the Chat System:**
   - Compile the server and client programs using a C compiler.
   - Run the server program on the server machine.
   - Run the client program on two different client machines and connect them to the server.
   - Send chat messages from both clients and ensure they are received and relayed correctly.
   - The relayed message should include the IP address of the client that sent the original message to the server.

Following screenshot of the terminal depicts the creation of the server and the three connections made by three different IP addresses.

```
avanzatech@avanzatech-pc-System-Product-Name:~/Desktop/Avanzatech Etapas/Etapa 6. OS & networks/lab_n
etwork$ ./server.output 127.00.00.1 8080
Configuring local address...
Creating socket...
Binding socket to local address...
Listening...
Waiting for connections...
New connection from 10.10.110.138
New connection from 10.10.110.122
New connection from 10.10.110.133
```

Now wireshark shows the three hand-shakes made when each of the three IPs connected to the server.

tcp.port==8080

No.	Time	Source	Destination	Protocol	Length	Info
81	6.716523998	10.10.110.138	10.10.110.138	TCP	76	42534 → 8080 [SYN]
82	6.716546530	10.10.110.138	10.10.110.138	TCP	76	8080 → 42534 [SYN,
83	6.716568411	10.10.110.138	10.10.110.138	TCP	68	42534 → 8080 [ACK]
117	10.838120232	10.10.110.122	10.10.110.138	TCP	76	36464 → 8080 [SYN]
118	10.838172409	10.10.110.138	10.10.110.122	TCP	76	8080 → 36464 [SYN,
119	10.838252338	10.10.110.122	10.10.110.138	TCP	68	36464 → 8080 [ACK]
176	16.977053109	10.10.110.133	10.10.110.138	TCP	76	50252 → 8080 [SYN]
177	16.977105527	10.10.110.138	10.10.110.133	TCP	76	8080 → 50252 [SYN,
178	16.977806482	10.10.110.133	10.10.110.138	TCP	68	50252 → 8080 [ACK]

This screenshot was taken from Juan`s terminal

```
avanzatech@avanzatech-pc:~/repositorios/lab_redes/gabo$ ./tcp_clientc_gabo 10.10.110.138 8080
Configuring remote address...
Remote address is: 10.10.110.138 http-alt
Creating socket...
Connecting...
Connected.
To send data, enter text followed by enter.
Received from IP        10.10.110.138 --> Muchachos las mejores en Avanzatech =)
Received from IP        10.10.110.122 --> El mejor eres tú
Vamos con toda!!
Sending: Vamos con toda!!
Received from IP        10.10.110.122 --> Saludos cordiales 👋🏽
```

This screenshot was taken from Mario`s terminal

```
❯ gcc tcp_client.c -o client.o && ./client.o 10.10.110.138 8080
Configuring remote address...
Remote address is: 10.10.110.138 http-alt
Creating socket...
Connecting...
Connected.
To send data, enter text followed by enter.
Received from IP        10.10.110.138 --> Muchachos las mejores en Avanzatech =)
El mejor eres tú
Sending: El mejor eres tú
Received from IP        10.10.110.133 --> Vamos con toda!!
Saludos cordiales 👋🏾
Sending: Saludos cordiales 👋🏾
```

Following screenshot of wireshark shows all communication between the IPs connected to the server.

375	41.496125449	10.10.110.138	10.10.110.138	TCP	107 42534 → 8080 [PSH, ACK] S
376	41.496169040	10.10.110.138	10.10.110.138	TCP	68 8080 → 42534 [ACK] Seq=1
377	41.496248478	10.10.110.138	10.10.110.122	TCP	120 8080 → 36464 [PSH, ACK] S
378	41.496277643	10.10.110.138	10.10.110.133	TCP	120 8080 → 50252 [PSH, ACK] S
379	41.496482264	10.10.110.133	10.10.110.138	TCP	68 50252 → 8080 [ACK] Seq=1
380	41.496660245	10.10.110.122	10.10.110.138	TCP	68 36464 → 8080 [ACK] Seq=1
478	49.220945066	10.10.110.122	10.10.110.138	TCP	86 36464 → 8080 [PSH, ACK] S
479	49.221021017	10.10.110.138	10.10.110.122	TCP	68 8080 → 36464 [ACK] Seq=53
480	49.221138145	10.10.110.138	10.10.110.138	TCP	99 8080 → 42534 [PSH, ACK] S
481	49.221162821	10.10.110.138	10.10.110.138	TCP	68 42534 → 8080 [ACK] Seq=40
482	49.221190242	10.10.110.138	10.10.110.133	TCP	99 8080 → 50252 [PSH, ACK] S
483	49.222518496	10.10.110.133	10.10.110.138	TCP	68 50252 → 8080 [ACK] Seq=1
772	55.027462723	10.10.110.133	10.10.110.138	TCP	85 50252 → 8080 [PSH, ACK] S
773	55.027505933	10.10.110.138	10.10.110.133	TCP	68 8080 → 50252 [ACK] Seq=84
774	55.027575313	10.10.110.138	10.10.110.138	TCP	98 8080 → 42534 [PSH, ACK] S
775	55.027589379	10.10.110.138	10.10.110.138	TCP	68 42534 → 8080 [ACK] Seq=40
776	55.027605679	10.10.110.138	10.10.110.122	TCP	98 8080 → 36464 [PSH, ACK] S
777	55.028091033	10.10.110.122	10.10.110.138	TCP	68 36464 → 8080 [ACK] Seq=19
1159	79.268200220	10.10.110.122	10.10.110.138	TCP	95 36464 → 8080 [PSH, ACK] S
1160	79.268277334	10.10.110.138	10.10.110.122	TCP	68 8080 → 36464 [ACK] Seq=83
1161	79.268400653	10.10.110.138	10.10.110.138	TCP	108 8080 → 42534 [PSH, ACK] S
1162	79.268424417	10.10.110.138	10.10.110.138	TCP	68 42534 → 8080 [ACK] Seq=40
1163	79.268455836	10.10.110.138	10.10.110.133	TCP	108 8080 → 50252 [PSH, ACK] S
1164	79.268954214	10.10.110.133	10.10.110.138	TCP	68 50252 → 8080 [ACK] Seq=18
1240	91.904570608	10.10.110.138	10.10.110.138	TCP	68 42534 → 8080 [FIN, ACK] S
1241	91.904664402	10.10.110.138	10.10.110.138	TCP	68 8080 → 42534 [FIN, ACK] S
1242	91.904698125	10.10.110.138	10.10.110.138	TCP	68 42534 → 8080 [ACK] Seq=41
1253	94.103521467	10.10.110.138	10.10.110.133	TCP	68 8080 → 50252 [FIN, ACK] S
1254	94.103583953	10.10.110.138	10.10.110.122	TCP	68 8080 → 36464 [FIN, ACK] S
1255	94.103853175	10.10.110.133	10.10.110.138	TCP	68 50252 → 8080 [FIN, ACK] S
1256	94.103899411	10.10.110.138	10.10.110.133	TCP	68 8080 → 50252 [ACK] Seq=12
1257	94.104109783	10.10.110.122	10.10.110.138	TCP	68 36464 → 8080 [FIN, ACK] S
1258	94.104133517	10.10.110.138	10.10.110.122	TCP	68 8080 → 36464 [ACK] Seq=84

Part B: TCP Packet Spoofing

1. **Analyzing the TCP Connection:**
   - Use Wireshark to capture the TCP packet sequence of the chat message from the first client to the server.
   - Analyze the packet sequence to understand the TCP three-way handshake and message structure.

2. **Implement the Faker Script:**
   - Write a C program that constructs a fake TCP packet mimicking the first client's packet sequence.
   - The program should set the source IP and port to that of the first client and use raw sockets to send the packet to the server.

3. **Testing Packet Spoofing:**
   - Run the faker script from the second client machine.
   - Verify that the server receives the message and relays it as if it were from the first client.
   - The relayed message should include the IP address of the client that sent the original message to the server. (The faked message should be shown as if it were sent from the other client.)

Here we can see the IPs being connected to the server. For the completion of this exercise only two IPs connected to it, although you will notice that we disconnected and then joined back the server a second time.

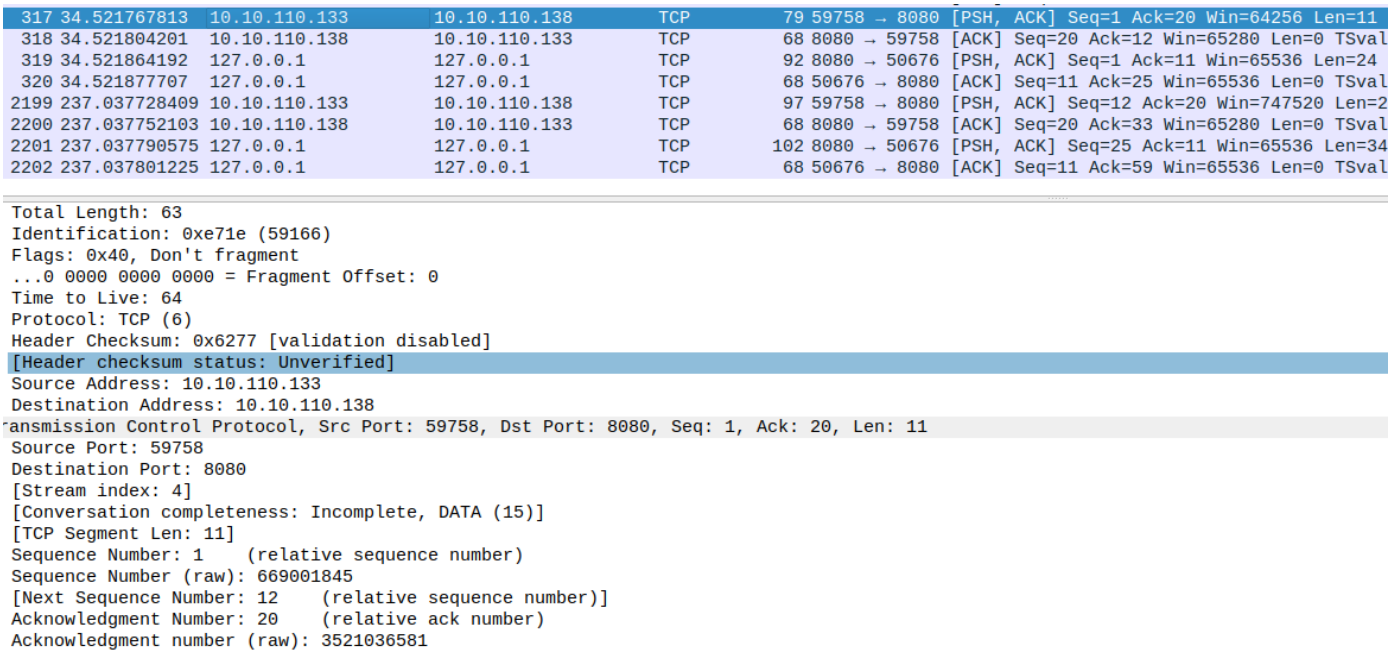
```
avanzatech@avanzatech-pc-System-Product-Name:~/Desktop/Avanzatech Etapas/Etapa 6. OS & networks/lab_n
etwork$ ./server.output
Configuring local address...
Creating socket...
Binding socket to local address...
Listening...
Waiting for connections...
New connection from 127.0.0.1
New connection from 10.10.110.133
New connection from 127.0.0.1
New connection from 10.10.110.133
```

On Juan terminal we can see that I initiated the conversation with “Hola Juan”, to what he replied “Hola Gabo”. Up until here the chat is working as it is meant to do.

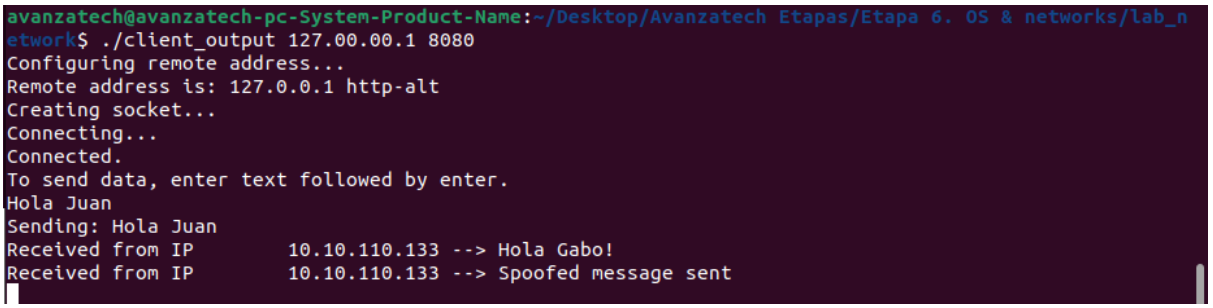
```
avanzatech@avanzatech-pc:~/repositorios/lab_redes/gabo$ ./tcp_clientc_gabo 10.10.110.138 8080
Configuring remote address...
Remote address is: 10.10.110.138 http-alt
Creating socket...
Connecting...
Connected.
To send data, enter text followed by enter.
Received from IP        127.0.0.1 --> Hola Juan
Hola Gabo!
Sending: Hola Gabo!
```

The message that Juan sent me allowed me to capture the info that I needed to hardcode my “raw_socket.c” file and by doing so, be able to spoof a custom message, pretending that it was him the one that sent it.
Following screenshot depicts the information that I used from the captured message packet that he had sent.
The info I used is:

- Identification = 59166. To which I added one in the file.
- Sequence Number(raw): 669001845. To which I added 11, that refers to the length of the last push that IP made.
- Acknowledgment number(raw): 3521036581
- Source port = 59758



On my terminal we can see the initial greeting message exchange, then we can see the spoofed message, which is shown to be sent by Juan`s IP number.
Note the spoofed message is not shown on Juan’s terminal.



Discussion:

Analyze the implications of the success of the TCP spoofing attack. Discuss potential security vulnerabilities in the client-server model and methods to mitigate such attacks. Reflect on the learning outcomes and the effectiveness of the methods used.

Successful TCP spoofing can lead to unauthorized access to sensitive information exchanged between clients and server, attackers can modify the data being sent between the client and server, leading to potential integrity issues. These attacks can be used to hijack established sessions, allowing hackers to impersonate legitimate users. These fake packets can be crafted to disrupt communication, causing a denial of service for legitimate users.If the client-server model relies only on IP addresses for authentication, spoofing becomes a significant threat. As we could prove during the lab, the predictable number sequences increase the likelihood of successful spoofing attacks. By using Wireshark we could capture packet messages, getting access to info that might do great damage in the wrong hands. TCP protocol lacks proper validation and verification mechanisms which allow attackers to inject malicious data into the communication stream or even worse.

Methods such as implementing strong authentication mechanisms beyond relying only on IP addresses and the use of cryptographic protocols gives you a step ahead of those who are not using them. Generating unpredictable number sequences is a good tool too, as it makes it difficult for attackers to predict and hijack sessions. Encapsulating traffic within secure VPN tunnels to protect data from interception and tampering is also something to take into consideration.

The content of this laboratory opened a new door for us as it introduced us to a critical topic within this journey. Talking to my programmer friends, they assured me that these types of labs put us ahead of many programmers nowadays that are being taught by bootcamps that definitely do not go this deep in their curriculum.

This exercise was very revealing, and even though we were surprised by it, we could see it is just the tip of the iceberg, as the hacking skills out there go far beyond what we put in practice.

Conclusion:

Summarize the findings of the lab, emphasizing the importance of understanding network protocols and security measures in the context of client-server communication.

In conclusion, the importance of network protocols and security measures cannot be overstated in our daily work as programmers. Network protocols lay the groundwork for smooth communication between devices and systems, serving as the common language that enables the exchange of information. These protocols not only facilitate the functioning of the internet but also form the backbone of client-server interactions.

On the other hand, security measures ensure the confidentiality, integrity, and availability of data. From encryption methods that shield sensitive information to authentication procedures that verify the legitimacy of users and systems, these security measures are essential in cultivating trust and mitigating the risks associated with cyber threats.

As our reliance on digital communication intensifies, the importance of understanding and implementing effective network protocols and security measures becomes increasingly important. It is not merely a technical necessity but a fundamental aspect of protecting the privacy and reliability of everyone.