PDI - TP 1 - B

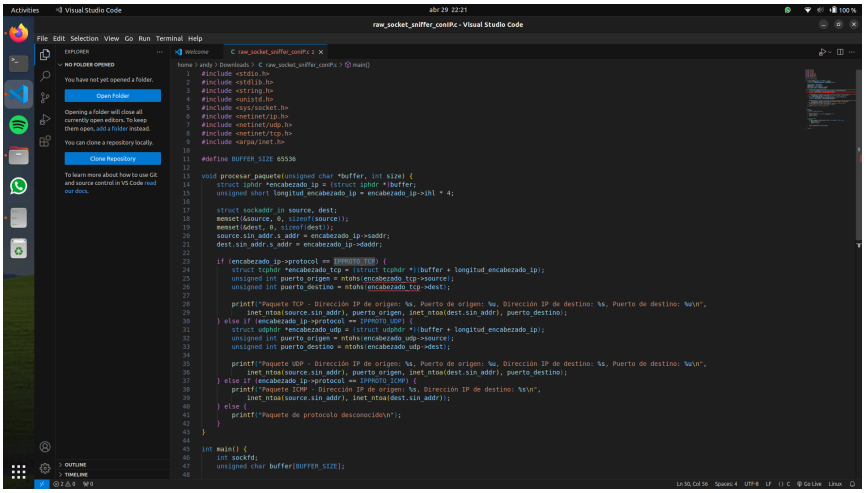
Integrantes: Lucas Debarbieri, Gonzalo Crucitta, Andres Luza, Sebastian Lernoud

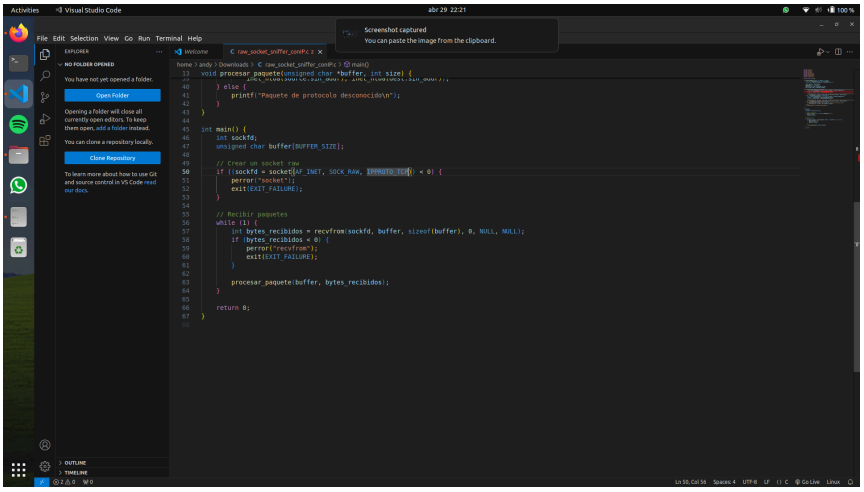
Consigna:

1. Utilizando el código raw.c como base escribir un "sniffer" que es un programa que muestra el contenido del tráfico que llega.
2. Enviar tráfico al "sniffer" desde el cliente escrito en la parte A del TP1
3. Enviar tráfico ICMP al "sniffer" y mostrar los resultados del LOG con comentarios.
4. Mostrar resultados.

PARTE 1

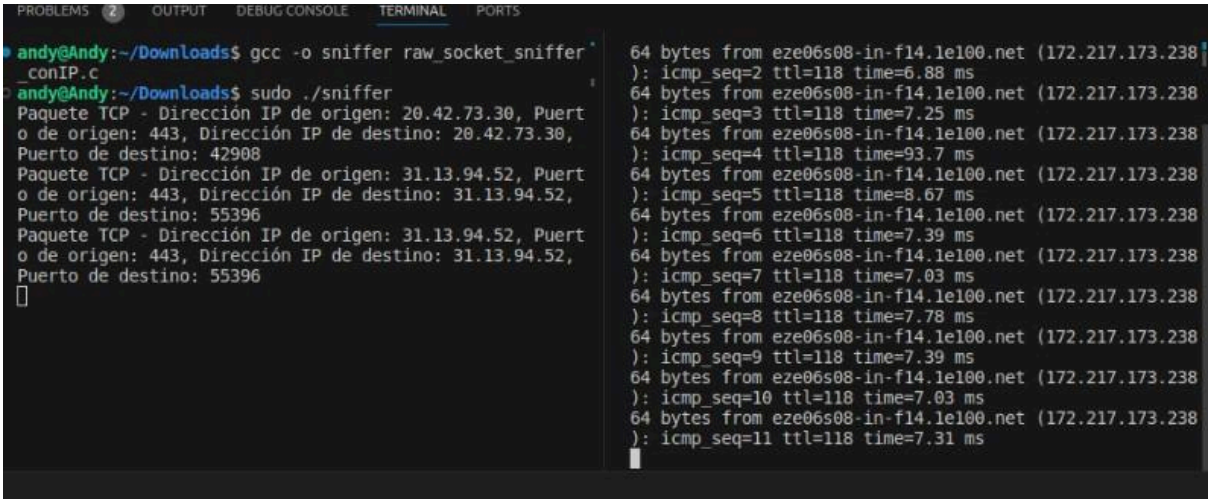
Usamos el codigo del archivo raw.c para armar un sniffer que captura el trafico del cliente y el servidor. A continuacion proporcionamos el codigo del sniffer.





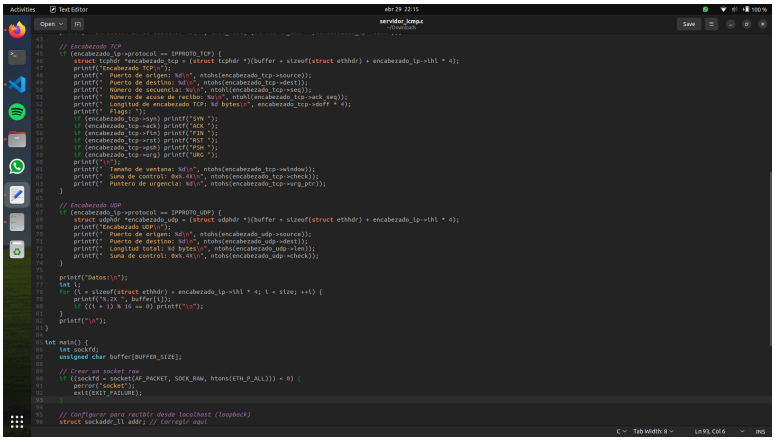
PARTE 2

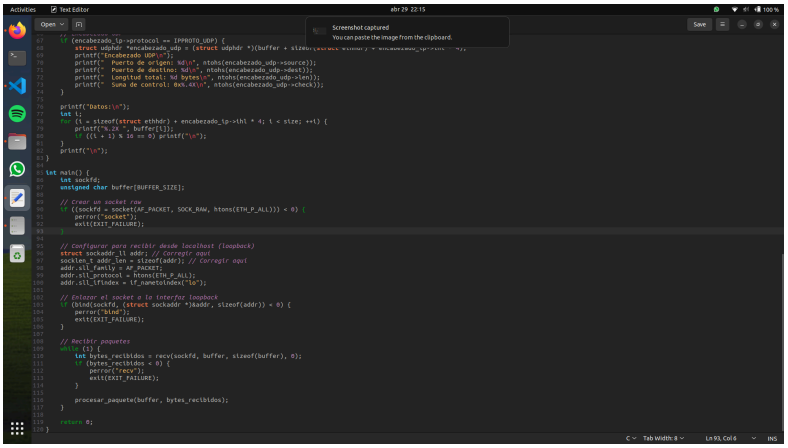
Acá en estas imágenes vamos a ver como el sniffer lee los paquetes de la web y vemos como funciona para detectar los paquetes TCP y mostrar su información.



En estas imágenes se puede apreciar como un cliente manda un paquete ICMP al servidor y el sniffer es capaz de captarlo. Para eso cambiamos el tipo de protocolo en el sniffer y ahora puede interpretar paquetes ICMP y mostrar su información.

A continuación mostraremos el código utilizado para el Servidor





A continuación mostraremos el código utilizado para el cliente.

UCA

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>

#define MAX 80
#define SERVERPORT 53490
#define SA struct sockaddr

void func(int sockfd)
{
    char buff[MAX];
    int n;
    for (;;)
    {
        bzero(buff, sizeof(buff));
        printf("Ingrese texto : ");
        n = 0;
        while ((buff[n++] = getchar()) != '\n')
            ;
        write(sockfd, buff, sizeof(buff));
        bzero(buff, sizeof(buff));
        read(sockfd, buff, sizeof(buff));
        printf("Servidor : %s", buff);
        if ((strncmp(buff, "exit", 4)) == 0)
        {
            printf("Cliente cierra conexión...\n");
            break;
        }
    }
}
```

```c
int main()
{
    int sockfd;
    struct sockaddr_in server_addr;
    char buffer[1024];
    int bytesRead;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1)
    {
        perror("Failed to create socket");
        exit(1);
    }

    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(SERVERPORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1"); // localhost

    if (connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) == -1)
    {
        perror("Failed to connect to server");
        exit(1);
    }

    bytesRead = recv(sockfd, buffer, sizeof(buffer) - 1, 0);
    if (bytesRead == -1)
    {
        perror("Failed to receive data from server");
        exit(1);
    }

    buffer[bytesRead] = '\0';
    printf("Received from server: %s\n", buffer);

    close(sockfd);

    return 0;
}
```

PARTE 3

Utilizamos el sniffer que creamos en la parte 1, cambiando el protocolo a IPPROTO_ICMP
para poder recibir los paquetes, y hacemos ping desde otra terminal para poder recibir los paquetes desde el sniffer. En el socket cambiamos el ultimo parametro, cambiando el IPPROTO_TCP.

```
// Crear socket RAW
sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);
if (sockfd < 0) {
```