



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 código del archivo raw.c para armar un sniffer que captura el tráfico del cliente y el servidor. A continuación proporcionamos el código del sniffer.



```
1 // raw_socket_sniffer_conIP.c
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 #include <sys/socket.h>
7 #include <sys/types.h>
8 #include <netinet/in.h>
9 #include <netinet/tcp.h>
10 #include <arpa/inet.h>
11 #include <unistd.h>
12
13 #define BUFFER_SIZE 65536
14
15 void process_packet(unsigned char *buffer, int size) {
16     struct iphdr *encabezado_ip = (struct iphdr *)buffer;
17     unsigned short longitud_encabezado_ip = ntohs(encabezado_ip->ihl) * 4;
18
19     struct sockaddr_in source, dest;
20     memset(&source, 0, sizeof(source));
21     memset(&dest, 0, sizeof(dest));
22     source.sin_addr.s_addr = encabezado_ip->saddr;
23     dest.sin_addr.s_addr = encabezado_ip->daddr;
24
25     if (encabezado_ip->protocol == IPPROTO_TCP) {
26         struct tcphdr *encabezado_tcp = (struct tcphdr *)buffer + longitud_encabezado_ip;
27         unsigned int puerto_origen = ntohs(encabezado_tcp->source_port);
28         unsigned int puerto_destino = ntohs(encabezado_tcp->dest_port);
29
30         printf("Paquete TCP - Dirección IP de origen: %s, Puerto de origen: %d, Dirección IP de destino: %s, Puerto de destino: %d\n",
31               inet_ntoa(source.sin_addr), puerto_origen, inet_ntoa(dest.sin_addr), puerto_destino);
32     } else if (encabezado_ip->protocol == IPPROTO_UDP) {
33         struct udphdr *encabezado_udp = (struct udphdr *)buffer + longitud_encabezado_ip;
34         unsigned int puerto_origen = ntohs(encabezado_udp->source_port);
35         unsigned int puerto_destino = ntohs(encabezado_udp->dest_port);
36
37         printf("Paquete UDP - Dirección IP de origen: %s, Puerto de origen: %d, Dirección IP de destino: %s, Puerto de destino: %d\n",
38               inet_ntoa(source.sin_addr), puerto_origen, inet_ntoa(dest.sin_addr), puerto_destino);
39     } else if (encabezado_ip->protocol == IPPROTO_ICMP) {
40         printf("Paquete ICMP - Dirección IP de origen: %s, Dirección IP de destino: %s\n",
41               inet_ntoa(source.sin_addr), inet_ntoa(dest.sin_addr));
42     } else {
43         printf("Paquete de protocolo desconocido\n");
44     }
45 }
46
47 int main() {
48     int sockfd;
49     unsigned char buffer[BUFFER_SIZE];
50 }
```

```
51
52 // raw_socket_sniffer_conIP.c
53
54 void process_packet(unsigned char *buffer, int size) {
55     // ... (previous code) ...
56 }
57
58 int main() {
59     int sockfd;
60     unsigned char buffer[BUFFER_SIZE];
61
62     // Crear un socket
63     if ((sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)) < 0) {
64         perror("socket");
65         exit(EXIT_FAILURE);
66     }
67
68     // Recibir paquetes
69     while (1) {
70         int bytes_recibidos = recvfrom(sockfd, buffer, sizeof(buffer), 0, NULL, NULL);
71         if (bytes_recibidos < 0) {
72             perror("recvfrom");
73             exit(EXIT_FAILURE);
74         }
75
76         process_packet(buffer, bytes_recibidos);
77     }
78
79     return 0;
80 }
```

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

[illegible]

A continuación mostraremos el código utilizado para el cliente.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <sys/types.h>
5  #include <sys/socket.h>
6  #include <netinet/in.h>
7  #include <arpa/inet.h>
8  #include <unistd.h>
9
10 #define MAX 80
11 #define SERVERPORT 53490
12 #define SA struct sockaddr
13
14 void func(int sockfd)
15 {
16     char buff[MAX];
17     int n;
18     for (;;)
19     {
20         bzero(buff, sizeof(buff));
21         printf("Ingresa texto : ");
22         n = 0;
23         while ((buff[n++] = getchar()) != '\n')
24             ;
25         write(sockfd, buff, sizeof(buff));
26         bzero(buff, sizeof(buff));
27         read(sockfd, buff, sizeof(buff));
28         printf("Servidor : %s", buff);
29         if ((strcmp(buff, "exit", 4)) == 0)
30         {
31             printf("Cliente cierra conexión...\n");
32             break;
33         }
34     }
35 }
36
37 int main()
38 {
39     int sockfd;
40     struct sockaddr_in server_addr;
41     char buffer[1024];
42     int bytesRead;
43
44     sockfd = socket(AF_INET, SOCK_STREAM, 0);
45     if (sockfd == -1)
46     {
47         perror("Failed to create socket");
48         exit(1);
49     }
50
51     memset(&server_addr, 0, sizeof(server_addr));
52     server_addr.sin_family = AF_INET;
53     server_addr.sin_port = htons(SERVERPORT);
54     server_addr.sin_addr.s_addr = inet_addr("127.0.0.1"); // localhost
55
56     if (connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) == -1)
57     {
58         perror("Failed to connect to server");
59         exit(1);
60     }
61
62     bytesRead = recv(sockfd, buffer, sizeof(buffer) - 1, 0);
63     if (bytesRead == -1)
64     {
65         perror("Failed to receive data from server");
66         exit(1);
67     }
68
69     buffer[bytesRead] = '\0';
70     printf("Received from server: %s\n", buffer);
71
72     close(sockfd);
73
74     return 0;
75 }
```

[illegible]