

Packet Sniffer using Raw Socket- TCP UDP DNS

Write the C language code for packet sniffer made using raw socket which used the packets from Promiscuous mode of os and stores in a buffer. Apply filter for packet types "TCP", "UDP" and "DNS" and show the port of entry and length of packets as well.

- TCP PACKETS

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <net/ethernet.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <sys/ioctl.h>
#include <net/if.h>

void process_packet(unsigned char *buffer, int size);

int main() {
    int raw_socket, data_size;
    char interface_name[IFNAMSIZ];
    unsigned char *buffer = (unsigned char *)malloc(65536);

    // Prompt for the network interface name
    printf("Enter the network interface name: ");
    scanf("%s", interface_name);

    raw_socket = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    if (raw_socket == -1) {
        perror("Socket creation error");
        exit(EXIT_FAILURE);
    }

    struct ifreq ifr;
    strncpy(ifr.ifr_name, interface_name, IFNAMSIZ - 1);
    if (ioctl(raw_socket, SIOCGIFFLAGS, &ifr) < 0) {
        perror("Error while getting interface flags");
        exit(EXIT_FAILURE);
    }
```

```

}

ifr.ifr_flags |= IFF_PROMISC;
if (ioctl(raw_socket, SIOCSIFFLAGS, &ifr) < 0) {
    perror("Error while setting interface to promiscuous mode");
    exit(EXIT_FAILURE);
}

while (1) {
    data_size = recvfrom(raw_socket, buffer, 65536, 0, NULL, NULL);
    if (data_size < 0) {
        perror("Packet receive error");
        exit(EXIT_FAILURE);
    }

    process_packet(buffer, data_size);
}

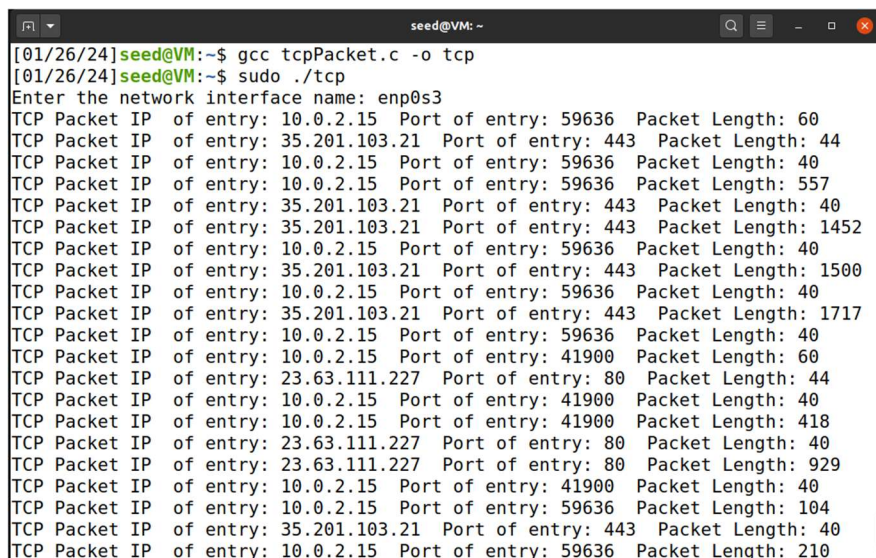
close(raw_socket);
free(buffer);

return 0;
}

void process_packet(unsigned char *buffer, int size) {
    struct ethhdr *eth = (struct ethhdr *)buffer;
    struct iphdr *iph = (struct iphdr *) (buffer + sizeof(struct ethhdr));
    struct tcphdr *tcph = (struct tcphdr *) (buffer + sizeof(struct ethhdr) + (iph->ihl *
4));

    if (eth->h_proto == htons(ETH_P_IP) && iph->protocol == IPPROTO_TCP) {
        printf("TCP Packet ");
        printf("IP of entry: %s ", inet_ntoa(*(struct in_addr *)&iph->saddr));
        printf("Port of entry: %u ", ntohs(tcph->source));
        printf("Packet Length: %d\n", ntohs(iph->tot_len));
    }
}

```



```

seed@VM: ~
[01/26/24]seed@VM:~$ gcc tcpPacket.c -o tcp
[01/26/24]seed@VM:~$ sudo ./tcp
Enter the network interface name: enp0s3
TCP Packet IP of entry: 10.0.2.15 Port of entry: 59636 Packet Length: 60
TCP Packet IP of entry: 35.201.103.21 Port of entry: 443 Packet Length: 44
TCP Packet IP of entry: 10.0.2.15 Port of entry: 59636 Packet Length: 40
TCP Packet IP of entry: 10.0.2.15 Port of entry: 59636 Packet Length: 557
TCP Packet IP of entry: 35.201.103.21 Port of entry: 443 Packet Length: 40
TCP Packet IP of entry: 35.201.103.21 Port of entry: 443 Packet Length: 1452
TCP Packet IP of entry: 10.0.2.15 Port of entry: 59636 Packet Length: 40
TCP Packet IP of entry: 35.201.103.21 Port of entry: 443 Packet Length: 1500
TCP Packet IP of entry: 10.0.2.15 Port of entry: 59636 Packet Length: 40
TCP Packet IP of entry: 35.201.103.21 Port of entry: 443 Packet Length: 1717
TCP Packet IP of entry: 10.0.2.15 Port of entry: 59636 Packet Length: 40
TCP Packet IP of entry: 10.0.2.15 Port of entry: 41900 Packet Length: 60
TCP Packet IP of entry: 23.63.111.227 Port of entry: 80 Packet Length: 44
TCP Packet IP of entry: 10.0.2.15 Port of entry: 41900 Packet Length: 40
TCP Packet IP of entry: 10.0.2.15 Port of entry: 41900 Packet Length: 418
TCP Packet IP of entry: 23.63.111.227 Port of entry: 80 Packet Length: 40
TCP Packet IP of entry: 23.63.111.227 Port of entry: 80 Packet Length: 929
TCP Packet IP of entry: 10.0.2.15 Port of entry: 41900 Packet Length: 40
TCP Packet IP of entry: 10.0.2.15 Port of entry: 59636 Packet Length: 104
TCP Packet IP of entry: 35.201.103.21 Port of entry: 443 Packet Length: 40
TCP Packet IP of entry: 10.0.2.15 Port of entry: 59636 Packet Length: 210

```

- **UDP PACKET**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <net/ethernet.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <sys/ioctl.h>
#include <net/if.h>

void process_packet(unsigned char *buffer, int size);

int main() {
    int raw_socket, data_size;
    char interface_name[IFNAMSIZ];
    unsigned char *buffer = (unsigned char *)malloc(65536);

    // Prompt for the network interface name
    printf("Enter the network interface name: ");
    scanf("%s", interface_name);

    raw_socket = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    if (raw_socket == -1) {
        perror("Socket creation error");
        exit(EXIT_FAILURE);
    }

    struct ifreq ifr;
    strncpy(ifr.ifr_name, interface_name, IFNAMSIZ - 1);
    if (ioctl(raw_socket, SIOCGIFFLAGS, &ifr) < 0) {
        perror("Error while getting interface flags");
        exit(EXIT_FAILURE);
    }

    ifr.ifr_flags |= IFF_PROMISC;
    if (ioctl(raw_socket, SIOCSIFFLAGS, &ifr) < 0) {
        perror("Error while setting interface to promiscuous mode");
        exit(EXIT_FAILURE);
    }

    while (1) {
        data_size = recvfrom(raw_socket, buffer, 65536, 0, NULL, NULL);
        if (data_size < 0) {
            perror("Packet receive error");
            exit(EXIT_FAILURE);
        }

        process_packet(buffer, data_size);
    }
}
```

```

}

close(raw_socket);
free(buffer);

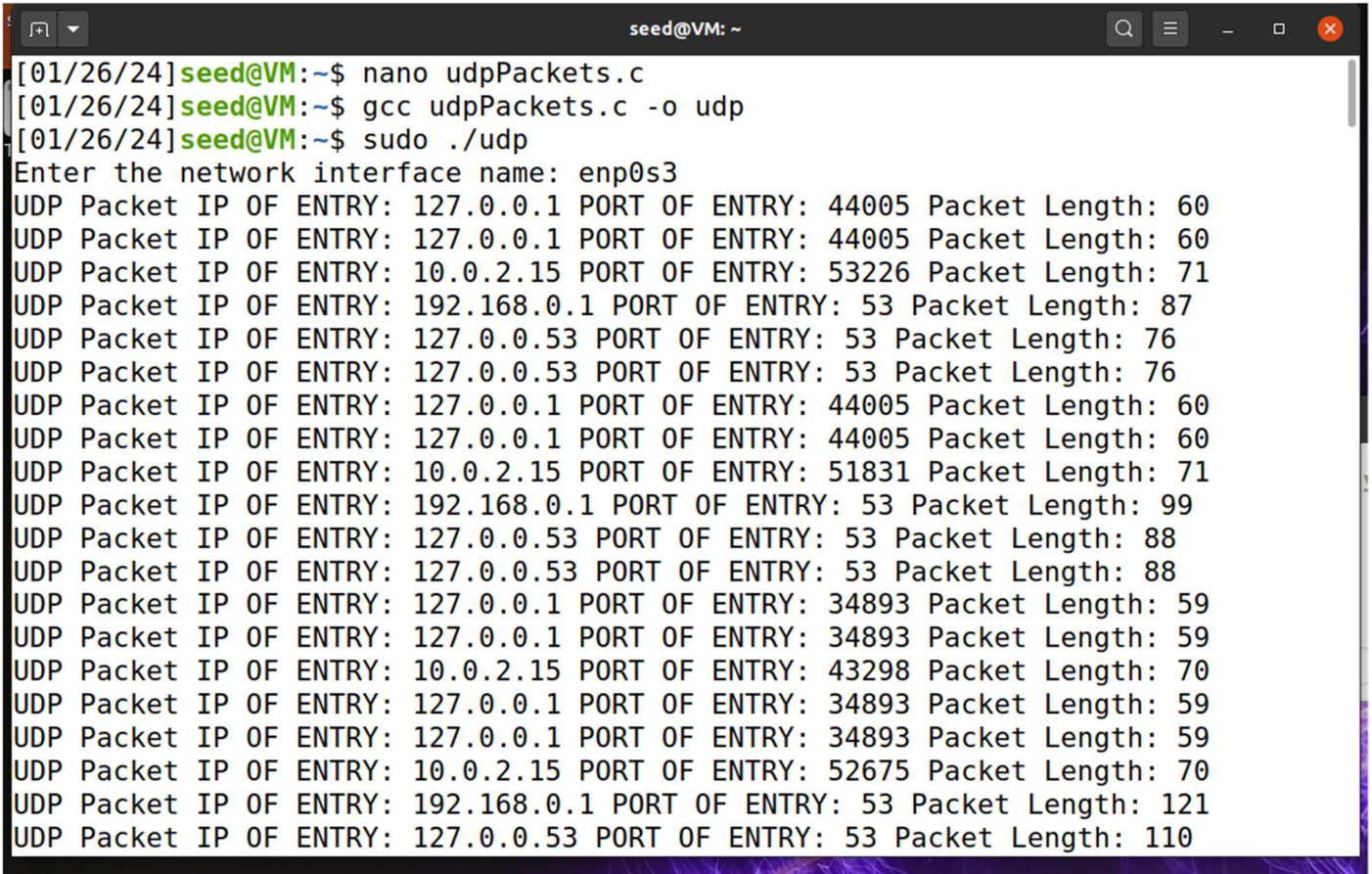
return 0;
}

void process_packet(unsigned char *buffer, int size) {
    struct ethhdr *eth = (struct ethhdr *)buffer;
    struct iphdr *iph = (struct iphdr *) (buffer + sizeof(struct ethhdr));
    struct udphdr *udph = (struct udphdr *) (buffer + sizeof(struct ethhdr) + (iph->ihl *
4));

    if (eth->h_proto == htons(ETH_P_IP) && iph->protocol == IPPROTO_UDP) {
        printf("UDP Packet ");
        printf("IP OF ENTRY: %s ", inet_ntoa(*(struct in_addr *)&iph->saddr));
        printf("PORT OF ENTRY: %u ", ntohs(udph->source));

        printf("Packet Length: %d\n", ntohs(iph->tot_len));
    }
}

```



```

seed@VM: ~
[01/26/24] seed@VM:~$ nano udpPackets.c
[01/26/24] seed@VM:~$ gcc udpPackets.c -o udp
[01/26/24] seed@VM:~$ sudo ./udp
Enter the network interface name: enp0s3
UDP Packet IP OF ENTRY: 127.0.0.1 PORT OF ENTRY: 44005 Packet Length: 60
UDP Packet IP OF ENTRY: 127.0.0.1 PORT OF ENTRY: 44005 Packet Length: 60
UDP Packet IP OF ENTRY: 10.0.2.15 PORT OF ENTRY: 53226 Packet Length: 71
UDP Packet IP OF ENTRY: 192.168.0.1 PORT OF ENTRY: 53 Packet Length: 87
UDP Packet IP OF ENTRY: 127.0.0.53 PORT OF ENTRY: 53 Packet Length: 76
UDP Packet IP OF ENTRY: 127.0.0.53 PORT OF ENTRY: 53 Packet Length: 76
UDP Packet IP OF ENTRY: 127.0.0.1 PORT OF ENTRY: 44005 Packet Length: 60
UDP Packet IP OF ENTRY: 127.0.0.1 PORT OF ENTRY: 44005 Packet Length: 60
UDP Packet IP OF ENTRY: 10.0.2.15 PORT OF ENTRY: 51831 Packet Length: 71
UDP Packet IP OF ENTRY: 192.168.0.1 PORT OF ENTRY: 53 Packet Length: 99
UDP Packet IP OF ENTRY: 127.0.0.53 PORT OF ENTRY: 53 Packet Length: 88
UDP Packet IP OF ENTRY: 127.0.0.53 PORT OF ENTRY: 53 Packet Length: 88
UDP Packet IP OF ENTRY: 127.0.0.1 PORT OF ENTRY: 34893 Packet Length: 59
UDP Packet IP OF ENTRY: 127.0.0.1 PORT OF ENTRY: 34893 Packet Length: 59
UDP Packet IP OF ENTRY: 10.0.2.15 PORT OF ENTRY: 43298 Packet Length: 70
UDP Packet IP OF ENTRY: 127.0.0.1 PORT OF ENTRY: 34893 Packet Length: 59
UDP Packet IP OF ENTRY: 127.0.0.1 PORT OF ENTRY: 34893 Packet Length: 59
UDP Packet IP OF ENTRY: 10.0.2.15 PORT OF ENTRY: 52675 Packet Length: 70
UDP Packet IP OF ENTRY: 192.168.0.1 PORT OF ENTRY: 53 Packet Length: 121
UDP Packet IP OF ENTRY: 127.0.0.53 PORT OF ENTRY: 53 Packet Length: 110

```

- DNS PACKET

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <net/ethernet.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <sys/ioctl.h>
#include <net/if.h>

#define DNS_PORT 53

void process_packet(unsigned char *buffer, int size);

int main() {
    int raw_socket, data_size;
    char interface_name[IFNAMSIZ];
    unsigned char *buffer = (unsigned char *)malloc(65536);

    // Prompt for the network interface name
    printf("Enter the network interface name: ");
    scanf("%s", interface_name);

    raw_socket = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    if (raw_socket == -1) {
        perror("Socket creation error");
        exit(EXIT_FAILURE);
    }

    struct ifreq ifr;
    strncpy(ifr.ifr_name, interface_name, IFNAMSIZ - 1);
    if (ioctl(raw_socket, SIOCGIFFLAGS, &ifr) < 0) {
        perror("Error while getting interface flags");
        exit(EXIT_FAILURE);
    }

    ifr.ifr_flags |= IFF_PROMISC;
    if (ioctl(raw_socket, SIOCSIFFLAGS, &ifr) < 0) {
        perror("Error while setting interface to promiscuous mode");
        exit(EXIT_FAILURE);
    }

    while (1) {
        data_size = recvfrom(raw_socket, buffer, 65536, 0, NULL, NULL);
        if (data_size < 0) {
            perror("Packet receive error");
            exit(EXIT_FAILURE);
        }
    }
}
```

```

        process_packet(buffer, data_size);
    }

    close(raw_socket);
    free(buffer);

    return 0;
}

void process_packet(unsigned char *buffer, int size) {
    struct ethhdr *eth = (struct ethhdr *)buffer;
    struct iphdr *iph = (struct iphdr *)(buffer + sizeof(struct ethhdr));
    struct udphdr *udph = (struct udphdr *)(buffer + sizeof(struct ethhdr) + (iph->ihl *
4));

    if (eth->h_proto == htons(ETH_P_IP) && iph->protocol == IPPROTO_UDP) {
        if (ntohs(udph->dest) == DNS_PORT || ntohs(udph->source) == DNS_PORT) {
            // DNS packet
            printf("DNS Packet ");
            printf("IP of entry : %s ", inet_ntoa(*(struct in_addr *)&iph->saddr));

            printf("port of entry : %u ", ntohs(udph->source));

            printf("Packet Length: %d\n", ntohs(iph->tot_len));
        }
    }
}

```


[01/26/24] seed@VM: ~\$ sudo ./dns

Enter the network interface name: enp0s3

ONS Packet IP of entry : 127.0.0.1 port of entry : 46421 Packet Length: 61
ONS Packet IP of entry : 127.0.0.1 port of entry : 46421 Packet Length: 61
ONS Packet IP of entry : 10.0.2.15 port of entry : 45735 Packet Length: 72
ONS Packet IP of entry : 127.0.0.1 port of entry : 46421 Packet Length: 61
ONS Packet IP of entry : 127.0.0.1 port of entry : 46421 Packet Length: 61
ONS Packet IP of entry : 10.0.2.15 port of entry : 39830 Packet Length: 72
ONS Packet IP of entry : 127.0.0.1 port of entry : 60672 Packet Length: 63
ONS Packet IP of entry : 127.0.0.1 port of entry : 60672 Packet Length: 63
ONS Packet IP of entry : 10.0.2.15 port of entry : 41640 Packet Length: 74
ONS Packet IP of entry : 127.0.0.1 port of entry : 60672 Packet Length: 63
ONS Packet IP of entry : 127.0.0.1 port of entry : 60672 Packet Length: 63
ONS Packet IP of entry : 10.0.2.15 port of entry : 59114 Packet Length: 74
ONS Packet IP of entry : 127.0.0.1 port of entry : 39503 Packet Length: 60
ONS Packet IP of entry : 127.0.0.1 port of entry : 39503 Packet Length: 60
ONS Packet IP of entry : 10.0.2.15 port of entry : 42957 Packet Length: 71
ONS Packet IP of entry : 127.0.0.1 port of entry : 39503 Packet Length: 60
ONS Packet IP of entry : 127.0.0.1 port of entry : 39503 Packet Length: 60
ONS Packet IP of entry : 10.0.2.15 port of entry : 37607 Packet Length: 71
ONS Packet IP of entry : 192.168.0.1 port of entry : 53 Packet Length: 218
ONS Packet IP of entry : 192.168.0.1 port of entry : 53 Packet Length: 362
ONS Packet IP of entry : 127.0.0.53 port of entry : 53 Packet Length: 207
ONS Packet IP of entry : 127.0.0.53 port of entry : 53 Packet Length: 207
