CENG393 Computer Networks
Labwork 8

1 Socket Programming: Raw Sockets

1.1 Definition

Previously, we have studied TCP and UDP sockets. Remember that a TCP socket is defined by requesting a connection-oriented socket from the operating system via:

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

And a UDP socket is defined by requesting a connectionless datagram socket:

```
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
```

When the third parameter of socket system call is left 0, the operating system associates the default protocol available to the chosen socket type. Default protocol for SOCK_STREAM is TCP and for SOCK_DGRAM is UDP. When these types of sockets are requested, the operating system handles various protocol related duties itself, such as preparing the TCP / UDP headers.

In this laboratory, you will be introduced to raw sockets. Raw sockets do not define a default protocol; it is the responsibility of the user for defining and controlling transport protocol. It is because of this reason that the operating system will prevent unauthorized users to execute programs that utilize raw sockets. In order to run your raw socket programs, you must temporarily elevate your privileges (by sudo command) or use a privileged account (root).

1.2 Exercise

You will find a sample raw socket program below. Use it together with the UDP server program we have used before.

1. Modify both programs so that when raw socket program sends a message to UDP server, the UDP server must send back a reply message and this message must be printed on screen by raw socket program.
2. What are the risks of using raw sockets in a program? Why does the operating system prohibit usage of raw sockets by unprivileged user accounts? Can you demonstrate an example?

1.3 raw.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <linux/in.h>
#include <linux/ip.h>
#include <linux/udp.h>

struct my_packet {
  struct iphdr ip;
  struct udphdr udp;
  unsigned char data[8];
};

unsigned short ip_cksum(unsigned short *buff, int len);
unsigned short udp_check(struct udphdr *th, unsigned short len,
    unsigned long saddr, unsigned long daddr);

unsigned long syn_daddr;
unsigned long syn_saddr;
int sockfd;

int main() {
  struct my_packet out;
  unsigned char in[65000];
  int size;
  unsigned short local_port = htons(3000);
  unsigned short syn_port = htons(2000);
  struct sockaddr_in to = { AF_INET, 2000 }, from = { AF_INET, 3000 };
  int one = 1;
  syn_daddr = inet_addr("127.0.0.1");
  syn_saddr = inet_addr("127.0.0.1");
  to.sin_addr.s_addr = syn_daddr;
  from.sin_addr.s_addr = syn_saddr;
  if ((sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_UDP)) < 0) {
    printf("socket creation error\n");
    exit(-1);
  }
  bind(sockfd, (struct sockaddr *) &from, sizeof(from));
  bzero(&out, sizeof(out));
  strcpy(out.data, "testing");
  out.ip.ihl = 5;
  out.ip.version = 4;
  out.ip.tos = 0;
  out.ip.tot_len = htons(sizeof(out));
  out.ip.id = getpid();
  out.ip.frag_off = 0;
  out.ip.ttl = 255;
  out.ip.protocol = IPPROTO_UDP;
  out.ip.saddr = syn_saddr;
  out.ip.daddr = syn_daddr;
  out.ip.check = 0;
  out.ip.check = ip_cksum((unsigned short *) &out.ip, 20);
  out.udp.source = local_port;
  out.udp.dest = syn_port;
  out.udp.len = htons(sizeof(struct udphdr) + 8);
  out.udp.check = 0;
  out.udp.check = udp_check(&out.udp, sizeof(struct udphdr) + 8, out.ip.saddr,
      out.ip.daddr);
  if (setsockopt(sockfd, IPPROTO_IP, IP_HDRINCL, &one, sizeof(one)))
    printf("setsockopt error\n");
  if (sendto(sockfd, &out, sizeof(out), 0, (struct sockaddr *) &to,
      sizeof(to)) < 0)
    printf("sendto error\n");
  printf("%d bytes packet sent to ...\n", sizeof(out));
  if (recvfrom(sockfd, &in, 4096, 0, (struct sockaddr *) &from, &size) < 0)
    printf("recvfrom error \n");

  return 0;
}

unsigned short ip_cksum(unsigned short *buff, int len) {
  unsigned long sum = 0;
  while (len > 1) {
    sum += *buff++;
    len -= 2;
  }
  if (len == 1)
    sum += (*buff & 0xff);
  sum = (sum >> 16) + (sum & 0xffff);
  sum += (sum >> 16);
  sum = ~sum;
  return (sum & 0xffff);
}

unsigned short udp_check(struct udphdr *th, unsigned short len,
    unsigned long saddr, unsigned long daddr) {
  unsigned long sum = 0;
  unsigned short *buff;
  buff = (unsigned short *) &saddr;
  sum += *buff++;
  sum += *buff;
  buff = (unsigned short *) &daddr;
  sum += *buff++;
  sum += *buff;
  sum += IPPROTO_UDP * 256;
  sum += htons(len) & 0xffff;
  buff = (unsigned short *) th;
  while (len > 1) {
    sum += *buff++;
    len -= 2;
  }
  if (len == 1)
    sum += (*buff & 0xff);
  sum = (sum >> 16) + (sum & 0xffff);
  sum += (sum >> 16);
  return ((~sum) & 0xffff);
}
```