**Wojskowa Akademia Techniczna**

**Wydział Elektroniki**

**Instytut Telekomunikacji**

**Studia II°**

**Projektowanie aplikacji sieciowych**

Materiały pomocnicze do zajęć

Cz.5.

**dr inż. Jarosław Krygier**

Warszawa 2017

Spis treści

[1 Gniazdo surowe RAW – sniffer 3](#_Toc496606791)

[2 Gniazdo surowe RAW – generator 8](#_Toc496606792)

# Gniazdo surowe RAW – sniffer

/\*

\* Oprogramownie na podstawie:

\*

\* Packet Sniffer Code in C using sockets | Linux

\* Author: By Silver Moon

\* Apr 26, 2009

\* http://www.binarytides.com/packet-sniffer-code-c-linux/

\*/

#include <stdio.h> //For standard things

#include <stdlib.h> //malloc

#include <string.h> //memset

#include <netinet/ip\_icmp.h> //Provides declarations for icmp header

#include <netinet/udp.h> //Provides declarations for udp header

#include <netinet/tcp.h> //Provides declarations for tcp header

#include <netinet/ip.h> //Provides declarations for ip header

#include <sys/socket.h>

#include <arpa/inet.h>

#include <net/if.h>

#include <unistd.h>

void ProcessPacket(unsigned char\* , int);

void print\_ip\_header(unsigned char\* , int);

void print\_tcp\_packet(unsigned char\* , int);

void print\_tcp\_packet\_to\_terminal(unsigned char\* , int);

void print\_udp\_packet(unsigned char \* , int);

void print\_icmp\_packet(unsigned char\* , int);

void PrintData (unsigned char\* , int);

int sock\_raw;

FILE \*logfile;

int tcp=0,udp=0,icmp=0,others=0,igmp=0,total=0,i,j;

struct sockaddr\_in source,dest;

int main()

{

socklen\_t saddr\_size;

int data\_size;

struct sockaddr saddr;

//struct in\_addr in;

unsigned char \*buffer = (unsigned char \*)malloc(65536); //Its Big!

logfile=fopen("log.txt","w");

if(logfile==NULL) printf("Unable to create file.");

printf("Starting...\n");

//Create a raw socket that shall sniff

sock\_raw = socket(AF\_INET , SOCK\_RAW , IPPROTO\_TCP);

struct ifreq ifr;

memset(&ifr, 0, sizeof(ifr));

snprintf(ifr.ifr\_name, sizeof(ifr.ifr\_name), "lo");

setsockopt(sock\_raw, SOL\_SOCKET, SO\_BINDTODEVICE, (void \*)&ifr, sizeof(ifr));

if(sock\_raw < 0)

{

printf("Socket Error\n");

return 1;

}

while(1)

{

saddr\_size = sizeof saddr;

//Receive a packet

data\_size = recvfrom(sock\_raw , buffer , 65536 , 0 , &saddr , &saddr\_size);

if(data\_size <0 )

{

printf("Recvfrom error , failed to get packets\n");

return 1;

}

//Now process the packet

ProcessPacket(buffer , data\_size);

}

close(sock\_raw);

printf("Finished");

return 0;

}

void ProcessPacket(unsigned char\* buffer, int size)

{

//Get the IP Header part of this packet

struct iphdr \*iph = (struct iphdr\*)buffer;

++total;

switch (iph->protocol) //Check the Protocol and do accordingly...

{

case 1: //ICMP Protocol

++icmp;

//PrintIcmpPacket(Buffer,Size);

break;

case 2: //IGMP Protocol

++igmp;

break;

case 6: //TCP Protocol

++tcp;

//print\_tcp\_packet(buffer , size);

print\_tcp\_packet\_to\_terminal(buffer , size);

break;

case 17: //UDP Protocol

++udp;

print\_udp\_packet(buffer , size);

break;

default: //Some Other Protocol like ARP etc.

++others;

break;

}

printf("TCP : %d UDP : %d ICMP : %d IGMP : %d Others : %d Total : %d\r",tcp,udp,icmp,igmp,others,total);

}

void print\_ip\_header(unsigned char\* Buffer, int Size)

{

unsigned short iphdrlen;

struct iphdr \*iph = (struct iphdr \*)Buffer;

iphdrlen = iph->ihl\*4;

memset(&source, 0, sizeof(source));

source.sin\_addr.s\_addr = iph->saddr;

memset(&dest, 0, sizeof(dest));

dest.sin\_addr.s\_addr = iph->daddr;

fprintf(logfile,"\n");

fprintf(logfile,"IP Header\n");

fprintf(logfile," |-IP Version : %d\n",(unsigned int)iph->version);

fprintf(logfile," |-IP Header Length : %d DWORDS or %d Bytes\n",(unsigned int)iph->ihl,((unsigned int)(iph->ihl))\*4);

fprintf(logfile," |-Type Of Service : %d\n",(unsigned int)iph->tos);

fprintf(logfile," |-IP Total Length : %d Bytes(Size of Packet)\n",ntohs(iph->tot\_len));

fprintf(logfile," |-Identification : %d\n",ntohs(iph->id));

//fprintf(logfile," |-Reserved ZERO Field : %d\n",(unsigned int)iphdr->ip\_reserved\_zero);

//fprintf(logfile," |-Dont Fragment Field : %d\n",(unsigned int)iphdr->ip\_dont\_fragment);

//fprintf(logfile," |-More Fragment Field : %d\n",(unsigned int)iphdr->ip\_more\_fragment);

fprintf(logfile," |-TTL : %d\n",(unsigned int)iph->ttl);

fprintf(logfile," |-Protocol : %d\n",(unsigned int)iph->protocol);

fprintf(logfile," |-Checksum : %d\n",ntohs(iph->check));

fprintf(logfile," |-Source IP : %s\n",inet\_ntoa(source.sin\_addr));

fprintf(logfile," |-Destination IP : %s\n",inet\_ntoa(dest.sin\_addr));

}

void print\_tcp\_packet(unsigned char\* Buffer, int Size)

{

unsigned short iphdrlen;

struct iphdr \*iph = (struct iphdr \*)Buffer;

iphdrlen = iph->ihl\*4;

struct tcphdr \*tcph=(struct tcphdr\*)(Buffer + iphdrlen);

fprintf(logfile,"\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TCP Packet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

print\_ip\_header(Buffer,Size);

fprintf(logfile,"\n");

fprintf(logfile,"TCP Header\n");

fprintf(logfile," |-Source Port : %u\n",ntohs(tcph->source));

fprintf(logfile," |-Destination Port : %u\n",ntohs(tcph->dest));

fprintf(logfile," |-Sequence Number : %u\n",ntohl(tcph->seq));

fprintf(logfile," |-Acknowledge Number : %u\n",ntohl(tcph->ack\_seq));

fprintf(logfile," |-Header Length : %d DWORDS or %d BYTES\n" ,(unsigned int)tcph->doff,(unsigned int)tcph->doff\*4);

//fprintf(logfile," |-CWR Flag : %d\n",(unsigned int)tcph->cwr);

//fprintf(logfile," |-ECN Flag : %d\n",(unsigned int)tcph->ece);

fprintf(logfile," |-Urgent Flag : %d\n",(unsigned int)tcph->urg);

fprintf(logfile," |-Acknowledgement Flag : %d\n",(unsigned int)tcph->ack);

fprintf(logfile," |-Push Flag : %d\n",(unsigned int)tcph->psh);

fprintf(logfile," |-Reset Flag : %d\n",(unsigned int)tcph->rst);

fprintf(logfile," |-Synchronise Flag : %d\n",(unsigned int)tcph->syn);

fprintf(logfile," |-Finish Flag : %d\n",(unsigned int)tcph->fin);

fprintf(logfile," |-Window : %d\n",ntohs(tcph->window));

fprintf(logfile," |-Checksum : %d\n",ntohs(tcph->check));

fprintf(logfile," |-Urgent Pointer : %d\n",tcph->urg\_ptr);

fprintf(logfile,"\n");

fprintf(logfile," DATA Dump ");

fprintf(logfile,"\n");

fprintf(logfile,"IP Header\n");

PrintData(Buffer,iphdrlen);

fprintf(logfile,"TCP Header\n");

PrintData(Buffer+iphdrlen,tcph->doff\*4);

fprintf(logfile,"Data Payload\n");

PrintData(Buffer + iphdrlen + tcph->doff\*4 , (Size - tcph->doff\*4-iph->ihl\*4) );

fprintf(logfile,"\n###########################################################\n");

}

void print\_tcp\_packet\_to\_terminal(unsigned char\* Buffer, int Size)

{

unsigned short iphdrlen;

struct iphdr \*iph = (struct iphdr \*)Buffer;

iphdrlen = iph->ihl\*4;

struct tcphdr \*tcph=(struct tcphdr\*)(Buffer + iphdrlen);

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TCP Packet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

//print\_ip\_header(Buffer,Size);

printf("\n");

printf("TCP Header\n");

printf(" |-Source Port : %u\n",ntohs(tcph->source));

printf(" |-Destination Port : %u\n",ntohs(tcph->dest));

printf(" |-Sequence Number : %u\n",ntohl(tcph->seq));

printf(" |-Acknowledge Number : %u\n",ntohl(tcph->ack\_seq));

printf(" |-Header Length : %d DWORDS or %d BYTES\n" ,(unsigned int)tcph->doff,(unsigned int)tcph->doff\*4);

//fprintf(logfile," |-CWR Flag : %d\n",(unsigned int)tcph->cwr);

//fprintf(logfile," |-ECN Flag : %d\n",(unsigned int)tcph->ece);

printf(" |-Urgent Flag : %d\n",(unsigned int)tcph->urg);

printf(" |-Acknowledgement Flag : %d\n",(unsigned int)tcph->ack);

printf(" |-Push Flag : %d\n",(unsigned int)tcph->psh);

printf(" |-Reset Flag : %d\n",(unsigned int)tcph->rst);

printf(" |-Synchronise Flag : %d\n",(unsigned int)tcph->syn);

printf(" |-Finish Flag : %d\n",(unsigned int)tcph->fin);

printf(" |-Window : %d\n",ntohs(tcph->window));

printf(" |-Checksum : %d\n",ntohs(tcph->check));

printf(" |-Urgent Pointer : %d\n",tcph->urg\_ptr);

printf("\n");

//printf(" DATA Dump ");

//printf("\n");

//fprintf(logfile,"IP Header\n");

//PrintData(Buffer,iphdrlen);

//fprintf(logfile,"TCP Header\n");

//PrintData(Buffer+iphdrlen,tcph->doff\*4);

//fprintf(logfile,"Data Payload\n");

//PrintData(Buffer + iphdrlen + tcph->doff\*4 , (Size - tcph->doff\*4-iph->ihl\*4) );

printf("\n###########################################################\n");

}

void print\_udp\_packet(unsigned char \*Buffer , int Size)

{

unsigned short iphdrlen;

struct iphdr \*iph = (struct iphdr \*)Buffer;

iphdrlen = iph->ihl\*4;

struct udphdr \*udph = (struct udphdr\*)(Buffer + iphdrlen);

fprintf(logfile,"\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*UDP Packet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

print\_ip\_header(Buffer,Size);

fprintf(logfile,"\nUDP Header\n");

fprintf(logfile," |-Source Port : %d\n" , ntohs(udph->source));

fprintf(logfile," |-Destination Port : %d\n" , ntohs(udph->dest));

fprintf(logfile," |-UDP Length : %d\n" , ntohs(udph->len));

fprintf(logfile," |-UDP Checksum : %d\n" , ntohs(udph->check));

fprintf(logfile,"\n");

fprintf(logfile,"IP Header\n");

PrintData(Buffer , iphdrlen);

fprintf(logfile,"UDP Header\n");

PrintData(Buffer+iphdrlen , sizeof udph);

fprintf(logfile,"Data Payload\n");

PrintData(Buffer + iphdrlen + sizeof udph ,( Size - sizeof udph - iph->ihl \* 4 ));

fprintf(logfile,"\n###########################################################");

}

void print\_icmp\_packet(unsigned char\* Buffer , int Size)

{

unsigned short iphdrlen;

struct iphdr \*iph = (struct iphdr \*)Buffer;

iphdrlen = iph->ihl\*4;

struct icmphdr \*icmph = (struct icmphdr \*)(Buffer + iphdrlen);

fprintf(logfile,"\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ICMP Packet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

print\_ip\_header(Buffer , Size);

fprintf(logfile,"\n");

fprintf(logfile,"ICMP Header\n");

fprintf(logfile," |-Type : %d",(unsigned int)(icmph->type));

if((unsigned int)(icmph->type) == 11)

fprintf(logfile," (TTL Expired)\n");

else if((unsigned int)(icmph->type) == ICMP\_ECHOREPLY)

fprintf(logfile," (ICMP Echo Reply)\n");

fprintf(logfile," |-Code : %d\n",(unsigned int)(icmph->code));

fprintf(logfile," |-Checksum : %d\n",ntohs(icmph->checksum));

//fprintf(logfile," |-ID : %d\n",ntohs(icmph->id));

//fprintf(logfile," |-Sequence : %d\n",ntohs(icmph->sequence));

fprintf(logfile,"\n");

fprintf(logfile,"IP Header\n");

PrintData(Buffer,iphdrlen);

fprintf(logfile,"UDP Header\n");

PrintData(Buffer + iphdrlen , sizeof icmph);

fprintf(logfile,"Data Payload\n");

PrintData(Buffer + iphdrlen + sizeof icmph , (Size - sizeof icmph - iph->ihl \* 4));

fprintf(logfile,"\n###########################################################");

}

void PrintData (unsigned char\* data , int Size)

{

for(i=0 ; i < Size ; i++)

{

if( i!=0 && i%16==0) //if one line of hex printing is complete...

{

fprintf(logfile," ");

for(j=i-16 ; j<i ; j++)

{

if(data[j]>=32 && data[j]<=128)

fprintf(logfile,"%c",(unsigned char)data[j]); //if its a number or alphabet

else fprintf(logfile,"."); //otherwise print a dot

}

fprintf(logfile,"\n");

}

if(i%16==0) fprintf(logfile," ");

fprintf(logfile," %02X",(unsigned int)data[i]);

if( i==Size-1) //print the last spaces

{

for(j=0;j<15-i%16;j++) fprintf(logfile," "); //extra spaces

fprintf(logfile," ");

for(j=i-i%16 ; j<=i ; j++)

{

if(data[j]>=32 && data[j]<=128) fprintf(logfile,"%c",(unsigned char)data[j]);

else fprintf(logfile,".");

}

fprintf(logfile,"\n");

}

}

}

# Gniazdo surowe RAW – generator

//Description : Gniazdo surowe TCP

// ============================================================================

\*/

// UWAGA! gniazdo surowe wymaga uprwnien administratora

/\*

Na podstawie:

Silver Moon (m00n.silv3r@gmail.com)

http://www.binarytides.com/raw-sockets-c-code-linux/

\*/

/\*

Naglowek IPv4:

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|Version| IHL |Type of Service| Total Length |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Identification |Flags| Fragment Offset |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Time to Live | Protocol | Header Checksum |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Address |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Destination Address |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Naglowek TCP:

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Source Port | Destination Port |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Sequence Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Acknowledgment Number |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Data | |U|A|P|R|S|F| |

| Offset| Reserved |R|C|S|S|Y|I| Window |

| | |G|K|H|T|N|N| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Checksum | Urgent Pointer |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options | Padding |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| data |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

\*/

#include <stdio.h> //for printf

#include <string.h> //memset

#include <sys/socket.h> //for socket ofcourse

#include <stdlib.h> //for exit(0);

#include <errno.h> //For errno - the error number

#include <netinet/tcp.h> //Provides declarations for tcp header

#include <netinet/ip.h> //Provides declarations for ip header

#include <arpa/inet.h> // inet\_addr

#include <unistd.h> // sleap()

/\*

96 bit (12 bytes) pseudo header needed for tcp header checksum calculation

\*/

struct pseudo\_header

{

u\_int32\_t source\_address;

u\_int32\_t dest\_address;

u\_int8\_t placeholder;

u\_int8\_t protocol;

u\_int16\_t tcp\_length;

};

/\*

Generic checksum calculation function

\*/

unsigned short csum(unsigned short \*ptr,int nbytes)

{

register long sum;

unsigned short oddbyte;

register short answer;

sum=0;

while(nbytes>1) {

sum+=\*ptr++;

nbytes-=2;

}

if(nbytes==1) {

oddbyte=0;

\*((u\_char\*)&oddbyte)=\*(u\_char\*)ptr;

sum+=oddbyte;

}

sum = (sum>>16)+(sum & 0xffff);

sum = sum + (sum>>16);

answer=(short)~sum;

return(answer);

}

int main (void)

{

//Create a raw socket

int s = socket (PF\_INET, SOCK\_RAW, IPPROTO\_TCP);

if(s == -1)

{

//socket creation failed, may be because of non-root privileges

perror("Failed to create socket");

exit(1);

}

//Datagram to represent the packet

char datagram[4096] , source\_ip[32] , \*data , \*pseudogram;

//zero out the packet buffer

memset (datagram, 0, 4096);

//IP header

struct iphdr \*iph = (struct iphdr \*) datagram;

//TCP header

struct tcphdr \*tcph = (struct tcphdr \*) (datagram + sizeof (struct ip));

struct sockaddr\_in sin;

struct pseudo\_header psh;

//Data part

data = datagram + sizeof(struct iphdr) + sizeof(struct tcphdr);

strcpy(data , "ABCDEFGHIJKLMNOPQRSTUVWXYZ");

//some address resolution

strcpy(source\_ip , "127.0.0.1"); //adres zrodlowy

sin.sin\_family = AF\_INET;

sin.sin\_port = htons(80);

sin.sin\_addr.s\_addr = inet\_addr ("127.0.0.1"); //adres docelowy

//Fill in the IP Header

iph->ihl = 5;

iph->version = 4;

iph->tos = 0;

iph->tot\_len = sizeof (struct iphdr) + sizeof (struct tcphdr) + strlen(data);

iph->id = htonl (54321); //Id of this packet

iph->frag\_off = 0;

iph->ttl = 255;

iph->protocol = IPPROTO\_TCP;

iph->check = 0; //Set to 0 before calculating checksum

iph->saddr = inet\_addr ( source\_ip ); //Spoof the source ip address

iph->daddr = sin.sin\_addr.s\_addr;

//Ip checksum

iph->check = csum ((unsigned short \*) datagram, iph->tot\_len);

//TCP Header

tcph->source = htons (1234);

tcph->dest = htons (80);

tcph->seq = 0;

tcph->ack\_seq = 0;

tcph->doff = 5; //tcp header size

tcph->fin=0;

tcph->syn=1;

tcph->rst=0;

tcph->psh=0;

tcph->ack=0;

tcph->urg=0;

tcph->window = htons (5840); /\* maximum allowed window size \*/

tcph->check = 0; //leave checksum 0 now, filled later by pseudo header

tcph->urg\_ptr = 0;

//Now the TCP checksum

psh.source\_address = inet\_addr( source\_ip );

psh.dest\_address = sin.sin\_addr.s\_addr;

psh.placeholder = 0;

psh.protocol = IPPROTO\_TCP;

psh.tcp\_length = htons(sizeof(struct tcphdr) + strlen(data) );

int psize = sizeof(struct pseudo\_header) + sizeof(struct tcphdr) + strlen(data);

pseudogram = malloc(psize);

memcpy(pseudogram , (char\*) &psh , sizeof (struct pseudo\_header));

memcpy(pseudogram + sizeof(struct pseudo\_header) , tcph , sizeof(struct tcphdr) + strlen(data));

tcph->check = csum( (unsigned short\*) pseudogram , psize);

//IP\_HDRINCL to tell the kernel that headers are included in the packet

// Zamiast tego mozna uzyc gniazda:

// int s = socket (AF\_INET, SOCK\_RAW, IPPROTO\_RAW);

// IPPROTO\_RAW - oznacza ze zarowno IP jak i naglowki warstw wyzszych trzeba zbudowac

int one = 1;

const int \*val = &one;

if (setsockopt (s, IPPROTO\_IP, IP\_HDRINCL, val, sizeof (one)) < 0)

{

perror("Error setting IP\_HDRINCL");

exit(0);

}

//wyzylanie w petli

while (1)

{

//Send the packet

if (sendto (s, datagram, iph->tot\_len , 0, (struct sockaddr \*) &sin, sizeof (sin)) < 0)

{

perror("sendto failed");

}

//Data send successfully

else

{

printf ("Packet Send. Length : %d \n" , iph->tot\_len);

}

sleep (1);

}

return 0;

}