第一部分

#define HAVE\_REMOTE

#include <pcap.h>

#include <Packet32.h>

#include <ntddndis.h>

#pragma comment(lib, "Packet")

#pragma comment(lib, "wpcap")

#pragma comment(lib, "WS2\_32")

using namespace std;

int main()

{

PIP\_ADAPTER\_INFO pAdapterInfo = NULL;

ULONG ulLen = 0;

pAdapterInfo = (PIP\_ADAPTER\_INFO)malloc(ulLen);

GetAdaptersInfo(pAdapterInfo, &ulLen);

GetAdaptersInfo(pAdapterInfo, &ulLen);

int count = 0;

while (pAdapterInfo)

{

printf("NIC %d: \n", ++count);

printf("\tIP: %s; Mask: %s; Gateway: %s\n", pAdapterInfo -> IpAddressList.IpAddress.String, pAdapterInfo->IpAddressList.IpMask.String, pAdapterInfo->GatewayList.IpAddress.String);

printf("\tName: %s; Desc: %s\n", pAdapterInfo->AdapterName, pAdapterInfo -> Description);

printf("\tMAC: ");

for (size\_t i = 0; i < pAdapterInfo->AddressLength; i++)

{

printf("%02X", pAdapterInfo->Address[i]);

}

printf("\n");

pAdapterInfo = pAdapterInfo->Next;

}

system("pause");

if (pAdapterInfo)

{

free(pAdapterInfo);

}

}

第二部分

/\*

\* Copyright (c) 1999 - 2005 NetGroup, Politecnico di Torino (Italy)

\* Copyright (c) 2005 - 2006 CACE Technologies, Davis (California)

\* All rights reserved.

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions

\* are met:

\*

\* 1. Redistributions of source code must retain the above copyright

\* notice, this list of conditions and the following disclaimer.

\* 2. Redistributions in binary form must reproduce the above copyright

\* notice, this list of conditions and the following disclaimer in the

\* documentation and/or other materials provided with the distribution.

\* 3. Neither the name of the Politecnico di Torino, CACE Technologies

\* nor the names of its contributors may be used to endorse or promote

\* products derived from this software without specific prior written

\* permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

\* "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

\* LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR

\* A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT

\* OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

\* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT

\* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,

\* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY

\* THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

\* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

\* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*

\*/

#ifdef \_MSC\_VER

/\*

\* we do not want the warnings about the old deprecated and unsecure CRT functions

\* since these examples can be compiled under \*nix as well

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#endif

#include "pcap.h"

/\* 4 bytes IP address \*/

typedef struct ip\_address

{

u\_char byte1;

u\_char byte2;

u\_char byte3;

u\_char byte4;

}ip\_address;

typedef struct mac\_header {

u\_char dest\_addr[6];

u\_char src\_addr[6];

u\_char type[2];

} mac\_header;

/\* IPv4 header \*/

typedef struct ip\_header

{

u\_char ver\_ihl; // Version (4 bits) + Internet header length (4 bits)

u\_char tos; // Type of service

u\_short tlen; // Total length

u\_short identification; // Identification

u\_short flags\_fo; // Flags (3 bits) + Fragment offset (13 bits)

u\_char ttl; // Time to live

u\_char proto; // Protocol

u\_short crc; // Header checksum

u\_char saddr[4]; // Source address

u\_char daddr[4]; // Destination address

u\_int op\_pad; // Option + Padding

}ip\_header;

/\* UDP header\*/

typedef struct udp\_header

{

u\_short sport; // Source port

u\_short dport; // Destination port

u\_short len; // Datagram length

u\_short crc; // Checksum

}udp\_header;

/\* prototype of the packet handler \*/

void packet\_handler(u\_char \*param, const struct pcap\_pkthdr \*header, const u\_char \*pkt\_data);

int main()

{

pcap\_if\_t \*alldevs;

pcap\_if\_t \*d;

int inum;

int i=0;

pcap\_t \*adhandle;

char errbuf[PCAP\_ERRBUF\_SIZE];

u\_int netmask;

char packet\_filter[] = "ip and udp";

struct bpf\_program fcode;

/\* Retrieve the device list \*/

if(pcap\_findalldevs(&alldevs, errbuf) == -1)

{

fprintf(stderr,"Error in pcap\_findalldevs: %s\n", errbuf);

exit(1);

}

/\* Print the list \*/

for(d=alldevs; d; d=d->next)

{

printf("%d. %s", ++i, d->name);

if (d->description)

printf(" (%s)\n", d->description);

else

printf(" (No description available)\n");

}

if(i==0)

{

printf("\nNo interfaces found! Make sure WinPcap is installed.\n");

return -1;

}

printf("Enter the interface number (1-%d):",i);

scanf("%d", &inum);

/\* Check if the user specified a valid adapter \*/

if(inum < 1 || inum > i)

{

printf("\nAdapter number out of range.\n");

/\* Free the device list \*/

pcap\_freealldevs(alldevs);

return -1;

}

/\* Jump to the selected adapter \*/

for(d=alldevs, i=0; i< inum-1 ;d=d->next, i++);

/\* Open the adapter \*/

if ((adhandle= pcap\_open\_live(d->name, // name of the device

65536, // portion of the packet to capture.

// 65536 grants that the whole packet will be captured on all the MACs.

1, // promiscuous mode (nonzero means promiscuous)

1000, // read timeout

errbuf // error buffer

)) == NULL)

{

fprintf(stderr,"\nUnable to open the adapter. %s is not supported by WinPcap\n",d->name);

/\* Free the device list \*/

pcap\_freealldevs(alldevs);

return -1;

}

/\* Check the link layer. We support only Ethernet for simplicity. \*/

if(pcap\_datalink(adhandle) != DLT\_EN10MB)

{

fprintf(stderr,"\nThis program works only on Ethernet networks.\n");

/\* Free the device list \*/

pcap\_freealldevs(alldevs);

return -1;

}

if(d->addresses != NULL)

/\* Retrieve the mask of the first address of the interface \*/

netmask=((struct sockaddr\_in \*)(d->addresses->netmask))->sin\_addr.S\_un.S\_addr;

else

/\* If the interface is without addresses we suppose to be in a C class network \*/

netmask=0xffffff;

//compile the filter

if (pcap\_compile(adhandle, &fcode, packet\_filter, 1, netmask) <0 )

{

fprintf(stderr,"\nUnable to compile the packet filter. Check the syntax.\n");

/\* Free the device list \*/

pcap\_freealldevs(alldevs);

return -1;

}

//set the filter

if (pcap\_setfilter(adhandle, &fcode)<0)

{

fprintf(stderr,"\nError setting the filter.\n");

/\* Free the device list \*/

pcap\_freealldevs(alldevs);

return -1;

}

printf("\nlistening on %s...\n", d->description);

/\* At this point, we don't need any more the device list. Free it \*/

pcap\_freealldevs(alldevs);

/\* start the capture \*/

pcap\_loop(adhandle, 0, packet\_handler, NULL);

return 0;

}

/\* Callback function invoked by libpcap for every incoming packet \*/

void packet\_handler(u\_char\* param, const struct pcap\_pkthdr\* header, const u\_char\* pkt\_data)

{

mac\_header\* mh;

ip\_header\* ih;

struct tm\* ltime;

char timestr[16];

udp\_header\* uh;

u\_int ip\_len;

u\_short sport, dport;

time\_t local\_tv\_sec;

int i = 0;

/\*

\* unused parameter

\*/

(VOID)(param);

/\* convert the timestamp to readable format \*/

local\_tv\_sec = header->ts.tv\_sec;

ltime = localtime(&local\_tv\_sec);

strftime(timestr, sizeof timestr, "%H:%M:%S", ltime);

/\* print timestamp and length of the packet \*/

printf("%s.%.6d len:%d ", timestr, header->ts.tv\_usec, header->len);

/\* retireve the position of the ip header \*/

ih = (ip\_header\*)(pkt\_data +

14); //length of ethernet header

/\* retireve the position of the udp header \*/

ip\_len = (ih->ver\_ihl & 0xf) \* 4;

uh = (udp\_header\*)((u\_char\*)ih + ip\_len);

/\* convert from network byte order to host byte order \*/

sport = ntohs(uh->sport);

dport = ntohs(uh->dport);

/\* print ip addresses and udp ports \*/

printf("%d.%d.%d.%d.%d -> %d.%d.%d.%d.%d\n",

ih->saddr[0],

ih->saddr[1],

ih->saddr[2],

ih->saddr[3],

sport,

ih->daddr[0],

ih->daddr[1],

ih->daddr[2],

ih->daddr[3],

dport);

int length = sizeof(mac\_header) + sizeof(ip\_header);

for (int i = 0; i < length; i++)

{

printf("%02X ", pkt\_data[i]);

if ((i & 0xF) == 0xF)

printf("\n");

}

printf("\n");

mh = (mac\_header\*)pkt\_data;

printf("mac\_header:\n");

printf("\tdest\_addr: ");

for (int i = 0; i < 6; i++)

{

printf("%02X ", mh->dest\_addr[i]);

}

printf("\n");

printf("\tsrc\_addr: ");

for(int i = 0; i < 6; i++)

{

printf("%02X ", mh->src\_addr[i]);

}

printf("\n");

printf("\ttype: %04X", ntohs(mh->type[i]));

printf("\n");

ih = (ip\_header\*)(pkt\_data + sizeof(mac\_header)); //length of ethernet header

printf("ip\_header\n");

printf("\t%-10s: %02X\n", "ver\_ihl", ih->ver\_ihl);

printf("\t%-10s: %02X\n", "tos", ih->tos);

printf("\t%-10s: %04X\n", "tlen", ntohs(ih->tlen));

printf("\t%-10s: %04X\n", "identification", ntohs(ih -> identification));

printf("\t%-10s: %04X\n", "flags\_fo", ntohs(ih->flags\_fo));

printf("\t%-10s: %02X\n", "ttl", ih->ttl);

printf("\t%-10s: %02X\n", "proto", ih->proto);

printf("\t%-10s: %04X\n", "crc", ntohs(ih->crc));

printf("\t%-10s: %08X\n", "op\_pad", ntohs(ih->op\_pad));

printf("\t%-10s: ", "saddr:");

for (int i = 0; i < 4; i++)

{

printf("%02X ", ih->saddr[i]);

}

printf(" ");

for (int i = 0; i < 4; i++)

{

printf("%d.", ih->saddr[i]);

}

printf("\n");

printf("\t%-10s: ", "daddr");

for (int i = 0; i < 4; i++)

{

printf("%02X ", ih->daddr[i]);

}

printf(" ");

for (int i = 0; i < 4; i++)

{

printf("%d.", ih->daddr[i]);

}

printf("\n");

}