#include "pcap.h"

#include <sys/types.h>

#include <pcap/pcap.h>

#include <netinet/in.h>

#include <string.h>

extern pid\_t pid;

extern u16 icmp\_req;//unsigned short = u16;

static const char\* dev = "eth0";

static char\* net;

static char\* mask;

static char filter\_string[FILTER\_STRING\_SIZE] = "";//FILTER\_STRING\_SIZE 100

struct in\_addr addr\_net;

static pcap\_t \*p;

static struct pcap\_pkthdr hdr;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

struct pcap\_pkthdr

{

struct timeval ts; ts是一個結構struct timeval，它有兩個部分，第一部分是1900開始以來的秒數，第二部分是當前秒之後的毫秒數

bpf\_u\_int32 caplen; 表示抓到的數據長度

bpf\_u\_int32 len; 表示數據包的實際長度

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* This function is almost completed.

\* But you still need to edit the filter string.

\*/

void pcap\_init( const char\* dst\_ip ,int timeout )

{

int ret;

char errbuf[PCAP\_ERRBUF\_SIZE];

bpf\_u\_int32 netp;

bpf\_u\_int32 maskp;

struct in\_addr addr;

/\*

struct in\_addr就是32位IP地址。

struct in\_addr {

unsigned long s\_addr;

};

\*/

struct bpf\_program fcode;

ret = pcap\_lookupnet(dev, &netp, &maskp, errbuf);//函數返回給定介面的IP位址和子網路遮罩。

if(ret == -1){

fprintf(stderr,"%s\n",errbuf);

exit(1);

}

addr.s\_addr = netp;

addr\_net.s\_addr = netp;

net = inet\_ntoa(addr);

if(net == NULL){

perror("inet\_ntoa");

exit(1);

}

addr.s\_addr = maskp;

mask = inet\_ntoa(addr);

if(mask == NULL){

perror("inet\_ntoa");

exit(1);

}

p = pcap\_open\_live(dev, 8000, 1, timeout, errbuf);//獲得用於捕獲網絡數據包的數據包捕獲描述字。

if(!p){

fprintf(stderr,"%s\n",errbuf);

exit(1);

}

/\*

pcap\_t \*pcap\_open\_live(char \*device, int snaplen, int promisc, int to\_ms, char \*ebuf)

device：上節中我們制定的監聽設備介面；

snaplen：制定pcap捕獲的最大數目的網路資料包；

promisc：>0指定device介面工作在混雜模式（promiscous Mode）；

to\_ms：制定經過特定時間（ms）後讀超時；0表示遇到錯誤退出，-1指定永不超時；

ebuf：制定用來存儲出錯資訊的字串

pcap\_t：返回值為用於監聽的pcap會話。

\*/

/\*

\* you should complete your filter string before pcap\_compile

\*/

memset(filter\_string,'\0',FILTER\_STRING\_SIZE);

sprintf(filter\_string,"%s%s","icmp and src host ",dst\_ip);

if(pcap\_compile(p, &fcode, filter\_string, 0, maskp) == -1){//將str參數指定的字符串編譯到過濾程序中。

pcap\_perror(p,"pcap\_compile");

exit(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int pcap\_compile(pcap\_t \*p, struct bpf\_program \*fp, char \*str, int optimize, bpf\_u\_int32 netmask)

p：表示pcap會話控制碼；

fp：存放編譯以後的規則；

str：規則運算式格式的過濾規則（filter），同tcpdump中的filter；

optimize：制定優化選項：0 false, 1 true；

netmask：監聽介面的網路遮罩；

返回值：-1表示操作失敗，其他值表成功。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(pcap\_setfilter(p, &fcode) == -1){ //指定一個過濾程序。

pcap\_perror(p,"pcap\_setfilter");

exit(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*

int pcap\_setfilter(pcap\_t \*p, struct bpf\_program \*fp)

p：表示pcap的會話控制碼；

fp：表示經過編譯後的過濾規則；

返回值：-1表示操作失敗，其他值表成功

\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

int pcap\_get\_reply( int sequence\_num ){

const u\_char \*ptr;

int i;

struct timeval tm;

ptr = pcap\_next(p, &hdr); //返回指向下一個數據包的u\_char指針

/\*

\* google "pcap\_next" to get more information

\* and check the packet that ptr pointed to.

\*/

pcap\_packet \*packet = NULL;

if(ptr != NULL){

packet = (pcap\_packet \*)(ptr-2) ;

}

while(ptr != NULL){

if(packet->header.icmp\_hdr.un.echo.id == pid){

if(packet->header.icmp\_hdr.un.echo.sequence == sequence\_num){

return (int)(hdr.ts.tv\_usec);

}

}

ptr = pcap\_next(p, &hdr);

if(ptr != NULL){

packet = (pcap\_packet \*)(ptr-2) ;

}

}

return 0;

}