**BORIS LOZA** 

# finding trojans for fun and profit



Boris Loza, Ph.D., is a founder of Tego System Inc. and HackerProof Technology, in addition to being a contributor to many industry magazines. He holds several patents and is an expert in computer security.

■ bloza@hackerproofonline.com

#### "THE TROJAN HORSE" ORIGINALLY

referred to the ploy used by the ancient Greeks to attack the city of Troy. Today, it's fairly common knowledge that a trojan horse is an application a cheeky hacker tries to install on your hard disk to get easy access to your computer. A trojan can be part of a rootkit while masquerading as a legitimate application such as ls, df, or ps. In this article I will show you how to find rootkits and trojans using handy little utilities and a couple of tricks.

#### **Checking Inodes**

One of the ways to find trojan files in a current directory is to check inode numbers. Many rootkits modify the access and modification time of the files they replace, so at a glance a file may appear to be unchanged or even untouched. What remains is to check an inode number of a file in question.

Most installs will install files sequentially. For example, the output below shows inode numbers for files in the /etc directory:

\$ Is -ai /etc | sort | more

. . . . . .

183491 TIMEZONE

183492 autopush

183493 cfgadm

183494 clri

183495 crash

183496 cron

. . . . . .

The –i option of |s lists the files' inode numbers. As you can see from the output, most of the inode numbers are in sequence.

A broken number sequence indicates the possibility that those files were installed after the main installation took place. Look for out-of-place entries, either very high or very low. Also look for new groupings, as all the rootkit pieces were probably installed at the same time

Note: The newfs command uses fsirand(1M) to install random inodes when creating a new file system. Also, if you use fsirand periodically, your system inode numbers will not be in sequence. For this reason, you may want to create a master database of all inode numbers for all your files. You can use something like the following to collect this information into a file:

# Is -aiR / > my\_inodes

Put this database aside and check the inode numbers of files in question against it. Update the database after installing new patches or system upgrades.

Check closely the /usr, /usr/bin, /sbin, /usr/sbin, and your X Window binaries directory, because rootkits are usually hiding in these places.

If an attack was successful, a hacker may install a rootkit. This is a suite of applications that can be used for many nasty things (creating back doors, root shells, etc.). It also helps to hide its own presence by modifying system commands that, for example, list all files in the directory (Is, dir (on Linux)) or find any file (find). Therefore, if you suspect that an attacker is on your system, you may not want to trust the Is or find commands, because they most likely have been replaced. How, then, do you list all files in the directory/s to find the rootkit's files?

#### Alternative Ways to List Files in a Directory

If a rootkit has been installed on your system, it replaced the Is and find commands with trojan versions that will not show a real list, one that includes the rootkit's files.

If you suspect that the current directory may contain a hidden directory or file, do one of the following. If you use Korn shell (ksh), press "ESC=" to list all files in a directory. For example, on Solaris OS:

```
$ ksh -o vi
$.<ESC>=
1) ../
2) ./
3) .Xauthority
4) .dt/
5) .dtprofile
6) .hushlogin
7) .netrc
8) .rhosts
9) .sh_history
Or:
1) TT_DB/
              12) mnt/
2) bin/
           13) net/
3) cdrom/
              14) opt/
4) dev/
         15) platform/
5) devices/ 16) proc/
           17) sbin/
6) etc/
7) export/
              18) tmp/
8) home/
              19) usr/
9) kernel/
              20) var/
           21) vol/
10) lib/
11) lost+found/ 22) xfn/
```

While your Is might be trojaned and not be able to see the hidden files, your Korn shell will see them.

You can also use the echo(1) command, which lists all files in a directory. For example:

\$ echo \*

TT\_DB bin cdrom core dev devices downloads etc export home kernel lib lost+found mnt mynes.txt net nsmail opt patches platform proc projects sbin test tmp ts1 typescript usr var vol xfn

Note: Echo will not show hidden files, such as files starting with ".".

On Linux systems, you may use the less(1) command to display all files in the directory:

```
$ less .
2 drwxr-xr-x 3 boris other 512 Jun 4 20:59 ./
2 drwxr-xr-x 43 root sys 1024 May 26 20:05 ../
2 -rw———— 1 boris other 90 Jun 4 21:14 .bash_profile ...
```

To list all files in the directory you can also use the tar(1) utility. Use the -w option for "wait for user confirmation." Answer "y" for the first entry and "n" for the rest of the list:

#### Using od and cat Commands

Because a directory is also a file, we will use commands that can be used to look inside files: od and cat. (You cannot use od(1) or cat(1) to display directories on Linux).

First, we will try to get the octal dump (using the od command) of the directory to look for all files. Let's try the following:

The od command with -c option displays single-byte characters. Certain non-graphic characters appear as C-language escapes:

```
null \0 backspace \b form-feed \f new-line \n return \r tab \t
```

Others appear as 3-digit octal numbers.

The od -c command starts each line with the number of bytes, in octal, shown since the start of the file. The first line starts at byte 0. The second line starts at byte 20 (that's byte 16 in decimal, the way most of us count). And so on. One can easily find file names on this output (., .., Project, webstat.log, and status).

The cat -v -t -e turns nonprintable characters into a printable form.

A directory usually has some long lines, so it's a good idea to pipe cat's output through fold:

You may try to filter out nonprintable characters. You can use something like the following (or use your own method):

 $\ cat -v -t -e \ . \ | fold | sed "s/[\x00-\x08\x80-\x88\x0B-\x19\ x8B-\x99\x7F\xFF]"//g$ 

Or:

 $d - c. | sed "s/[\001-\010\013-\037\177-\377]//g"$ 

Note: Be aware that some files shown by od or cat may be files that have already been deleted.

#### Conclusion

Obviously, if your Is command doesn't show the same files as any of these commands, beware (remember not to count deleted files!). This may be because your Is has been replaced by a hacker. Do not panic! This is not the end of the world. Open your favorite book on incident response and OS recovery.

You can find more hands-on tips and tricks of this kind in my new book *UNIX*, *Solaris and Linux: A Practical Security Cookbook*, which deals with securing UNIX OS without any of the third-party tools. It can be reviewed at www.amazon.com.

### **NEW!**

# ;login: Surveys

## To Help Us Meet Your Needs

*;login:* is the benefit you, the members of USENIX, have rated most highly. Please help us make this magazine even better.

Every issue of *;login:* online now offers a brief survey, for you to provide feedback on the articles in *;login:* . Have ideas about authors we should—or shouldn't—include, or topics you'd like to see covered? Let us know. See

http://www.usenix.org/publications/login/2005-10/ or go directly to the survey at

https://db.usenix.org/cgi-bin/loginpolls/oct05login/survey.cgi