4 May 2006

Unix Security: Diagnostics and Forensics

This document is intended to help Unix/Linux sys-admins with the diagnostic and forensic examination of a machine that has been hacked — or help determine whether a suspect machine has been. Specifically the document describes:

- immediate steps to take — a compromise between destroying forensic data and restoring service;
- further steps to take in order to determine what has been done and how;
- tools available to help.

1. Don't Panic! Don't Switch the Machine Off

So your machine has got at — or could have been. You have two conflicting tasks:

- preserve as much forensic data as possible so as to determine who the bad guys got in, what damage they did, what software they installed and what they were up to;
- stop the machine doing bad things and get it back into services a.s.a.p.

Ideally, do as little as possible to disturb the bad guys — this way you will be able to get most information. (The last thing you want is for them to shut down operations and clean up, leaving you with no evidence of how they got in or what they were up to.)

Minimum Immediate Network-Related Checks

If, for example, CERT or others are breathing down your neck, try to do the following before removing the network cable from the machine — do not switch it off as you may lose a lot of information (about e.g., running processes, loaded kernel modules, listening daemons): determine and understand every TCP connection [Page 2], determine and understand every open port [Page 3] and survey all network traffic [Page 5], in that order.

2. On what you should be looking for

Look for examples of the following which you cannot identify and/or are not familiar with:

- running processes, especially those associated with network connections and
- listening daemons, which could be running a service (such as IRC) or operating a back door;
- TCP connections — not just those on locally-privileged ports;
- outbound UDP and ICMP traffic, which could be part of a (distributed) denial of service attack or a scanning exercise.

3. Have you been rooted?

From the Wikipedia[1] page on "Rootkit":

A root kit is a set of tools used by an intruder after cracking a computer system. These tools can help the attacker maintain his or her access to the system and use it for malicious purposes.

A root kit typically hides logins, processes, and logs and often includes software to intercept data from terminals, network connections, and the keyboard...

A rootkit may also include utilities, known as backdoors to help the attacker subsequently access the system...

If their is a possibility that an intruder has gained `root` privileges on your machine, they may have installed a root kit. If so, you should not trust any of the standard tools such as `ls`, `ps`, `netstat`, `login` or `syslogd`: `ls`, `ps` and `netstat` may simply hide certain files, processes and connections; `syslogd` may not log certain events.

You will need a set of uncompromised utilities [Page 2].

4. Uncompromised tools and utilities

It's always worth a quick and dirty investigation of your machine with utilities what are already installed and may therefore have been compromised, but in order to carry out the investigations described in this document thoroughly you will need a set of utilities that you can trust [Page 1] not to omit or filter out information. Do both of the following:

1. Carry out at least the *Minimum Immediate Network-Related Checks* [Page 1] using a toolkit of CD-R-mounted statically-linked utilites [Page 10]. If possible carry out all checks listed in *First Steps*.
2. Carry out a full set of checks after booting from a *live CD* [Page 8].

5. Find and understand every TCP connection

If at all possible, use statically-linked tools and utilities mounted from a CD-R [Page 2] for this investigation.

List all TCP connections to/from the machine using `netstat` (see below). For all that you don't recognise, use `fuser` and/or `lsof` [Page 11], to determine which processes are responsible.

[1] `http://en.wikipedia.org`

On Linux it is possible to select TCP connections:

```
netstat -t

Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0     80 localhost.localdo:38436 localhost.localdoma:ssh ESTABLISHED
tcp        0     80 localhost.localdo:38862 localhost.localdoma:ssh ESTABLISHED
tcp        0      0 localhost.localdom:6012 localhost.localdo:40404 ESTABLISHED
tcp        0      0 mctalby.mc.ma:httpproxy CPE-67-48-233-44.n:2243 ESTABLISHED
tcp        0      0 mctalby.mc.ma:httpproxy 61.175.228.137:44104    ESTABLISHED
tcp        0      0 mctalby.mc.man.ac:55914 darkstar.umist.ac.u:ssh ESTABLISHED
tcp        0      0 mctalby.mc.man.ac:48994 bohrg2.man.ac.uk:484    ESTABLISHED
.
.
```

On Solaris, simply scroll down until the TCP header:

```
netstat -a | less

 TCP
    Local Address          Remote Address         Swind Send-Q Rwind Recv-Q  State
 --------------------   --------------------     ----- ------ ----- ------ -------
      *.*                    *.*                        0      0     0      0 IDLE
      *.sunrpc               *.*                        0      0     0      0 LISTEN
      *.*                    *.*                        0      0     0      0 IDLE
      *.892                  *.*                        0      0     0      0 BOUND
      *.32771                *.*                        0      0     0      0 LISTEN
      *.32772                *.*                        0      0     0      0 LISTEN
cosmos.umist.ac.uk.6051 bm2.csu.umist.ac.uk.1623 17443      0  8760      0 ESTABLISHED
cosmos.umist.ac.uk.6051 bm2.csu.umist.ac.uk.1624 17520      0  8760      0 ESTABLISHED
      *.*                    *.*                        0      0  8576      0 IDLE
      *.*                    *.*                        0      0  8576      0 IDLE
cosmos.umist.ac.uk.42376 130.88.211.29.ldap       8977      0  8760      0 ESTABLISHED
cosmos.umist.ac.uk.54164 sylo2.mc.man.ac.uk.22   33120      0  8760      0 ESTABLISHED
cosmos.umist.ac.uk.22 printer3.ma.man.ac.uk.3961 64511      0  8760      0 ESTABLISHED
.
.
```

6. Determine and understand every open port

If at all possible, use statically-linked tools and utilities mounted from a CD-R [Page 2] for this investigation.

You can do this locally with `netstat` and with `lsof`; you should also use a port-scanner, such as `nmap`, to do this remotely. *All three methods should agree.*

For all open ports that you don’t recognise, and for differences between the results of the three methods, use `fuser` and/or `lsof` [Page 11], to determine which processes are responsible.

On Linux

```
root>netstat -l

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:48132                 *:*                     LISTEN
tcp        0      0 *:8999                  *:*                     LISTEN
tcp        0      0 *:1066                  *:*                     LISTEN
tcp        0      0 *:httpproxy             *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
.
.
udp        0      0 localhost.localdoma:ntp *:*
.
.
```

On Solaris `netstat -a | grep LIST` will show up all daemons listening for TCP connections; to see all open ports, including UDP, fall back to `netstat -a | less`.

Thankfully, with `lsof`, the options for Linux and Solaris are the same:

```
root> lsof -i

./lsof -i
COMMAND     PID     USER   FD   TYPE        DEVICE   SIZE/OFF NODE NAME
inetd       325     root    4u  inet 0x30000721748       0t0  TCP *:time (LISTEN)
inetd       325     root    5u  inet 0x30000720988       0t0  UDP *:time (Idle)
inetd       325     root    6u  inet 0x300007208
.
.
sshd2     28972     root    7u  inet 0x300043b8a30      0t240  TCP \
                           sol.umist.ac.uk:49130->ldap3.ds.man.ac.uk:ldap (ESTABLISHED)
sshd2     28972     root    8u  inet 0x3000397f3b8      0t240 \
                       TCP sol.umist.ac.uk:49131->ldap3.ds.man.ac.uk:ldap (ESTABLISHED)
```

The third tool to use is `nmap`. You can use this to scan locally, i.e.,

```
noddy> nmap noddy.toytown.england
```

but it is better to scan for a trusted host — either turn off the firewall on the suspect host *temporarily*, or ensure the trusted host can get through on all ports, before scanning:

```
trusted> nmap noddy.toytown.england
trusted> nmap -vv -sT -p 1-1023 noddy.toytown.england
trusted> nmap -vvv -sU noddy.toytown.england
```

There are many options to `nmap`. Sample output:

```
nmap -vvv 127.0.0.1

Initiating Connect() Scan against \
                                localhost.localdomain (127.0.0.1) [1663 ports] at 16:19
Discovered open port 22/tcp on 127.0.0.1
Discovered open port 80/tcp on 127.0.0.1
.
.
The Connect() Scan took 0.12s to scan 1663 total ports.
Host localhost.localdomain (127.0.0.1) appears to be up ... good.
Interesting ports on localhost.localdomain (127.0.0.1):
(The 1657 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http

Nmap finished: 1 IP address (1 host up) scanned in 0.245 seconds
```

7. Survey all network traffic

If at all possible, use statically-linked tools and utilities mounted from a CD-R [Page 2] for this investigation.

You need to understand *all* traffic going to and from your machine — this can be time-consuming! `tcpdump` and `ethereal` are your friends here. Any traffic which you do not recognise should be treated as suspicious — use `lsof` [Page 11] to determine the process responsible for such traffic.

Example

Darkstar has one network interface, `hme0`:

```
tcpdump -i hme0 -n | egrep -v "130.88.99.10.22"
                   | egrep -v "130.88.119.67.53|130.88.120.67.53"
    # grep out things we already know about

11:18:55.486997 130.88.99.10.47865 > 130.88.124.69.6000: P 420:436(16) \
                                                            ack 161 win 8760 (DF)
11:18:55.487853 130.88.124.69.6000 > 130.88.99.10.47865: . ack 436 win 61304 (DF)
    # why are these people not tunnelling X traffic?

tcpdump -i hme0 -n | egrep -v "130.88.99.10.22"
                   | egrep -v "130.88.119.67.53|130.88.120.67.53"
                   | egrep -v "130.88.\d\d\d.\d\d.6000"

tcpdump -i hme0 -n | egrep -v "130.88.99.10.22"
                   | egrep -v "130.88.119.67.53|130.88.120.67.53"
                   | grep -v "130.88.[1-9][0-9][0-9].[1-9][0-9].6000"

13:35:49.729925 130.88.119.65.59549 > 130.88.99.10.25: S \
        3404736403:3404736403(0) win 5840 <mss 1460,sackOK,timestamp \
                                                           1011349486[|tcp]> (DF)
13:35:49.729967 130.88.99.10.25 > 130.88.119.65.59549: S \
        1333966013:1333966013(0) ack 3404736404 win 10136 <nop,nop,timestamp \
                                            267392917 1011349486,nop,[|tcp]> (DF)
    # email from UMIST email routers

tcpdump -i hme0 -n | egrep -v "130.88.99.10.[22|25]"
                   | egrep -v "130.88.119.67.53|130.88.120.67.53"
                   | grep -v "130.88.[1-9][0-9][0-9].[1-9][0-9].6000"

13:40:51.600950 130.88.99.10.2049 > 130.88.99.9.1007: . ack 2922688 win 8760 (DF)
13:40:51.600999 130.88.99.9.1007 > 130.88.99.10.2049: P \
                                    2922688:2924148(1460) ack 14829 win 8760 (DF)
    # 2049 is nfsd, so this is NFS traffic to/from eric

/usr/local/sbin/tcpdump -i hme0 -n | egrep -v "130.88.99.10.[22|25|123|2049]\
        |10.98.96.1|arp\ who|arp\ reply|802.1d\ config|130.88.1[1-2][0-9].67.53\
        |130.88.[1-9][0-9][0-9].[1-9][0-9].6000|130.88.1[1-2][0-9].6[5-6].25\
        |130.88.120.194.514"

14:12:05.498637 CDP v2, ttl=180s DevID ’TBA03170480(sw-umain)’ Addr (1): \
                                                          IPv4 130.88.98.2[|cdp]
14:13:05.507362 CDP v2, ttl=180s DevID ’TBA03170480(sw-umain)’ Addr (1): \
                                                          IPv4 130.88.98.2[|cdp]
```

. . . which leaves only CDP-related stuff.

8. Find and understand every process

If at all possible, use statically-linked tools and utilities mounted from a CD-R [Page 2] for this investigation.

There are two or three easy ways to do this:

- use the standard utility `ps auxww` or `ps -ef`;
- use `lsof` [Page 11]
- if you `/proc` filesystem is up to it (e.g., as on Linux) try something like this:

```
for J in 1 2 3 4 5 6 7 8 9; do ls -d  /proc/$J* | sort ; done > /tmp/proclist
for I in ‘cat /tmp/proclist‘; do cat $I/cmdline && echo "" ; done
```

Any process which you do not recognise should be treated as suspicious — Google it; any differences in the results between the two (or three) sets of results should be treated with equal suspicion.

9. Test for a rootkit

9.1. Standard Utilities

Any thorough search for a rootkit will begin with a boot from clean media [Page 8]. However, there are easy-to-use utilities which can help *without* the necessity of a reboot:

- chkrookit — see below [Page 14], or go to the homepage[2];
- Rootkit Hunter — see below [Page 15], or go to the homepage[3].

9.2. Kernel-Related Utilities

There are many other approaches to rootkit detection which usually require, in practice, a sysadmin to compilation up source code (as a minimum) and often knowledge of C and some kernel-level programming to tweak the code for a particular kernel. Details are beyond the scope of this document, but we mention two methods for interest sake:

- System-call fingerprinting: many rootkits work by wrapping system-calls — if data such as the address of each call is stored on a newly installed machine and periodically compared to the current state such wraps can often be detected.

- Loadable kernel module (LKM) scanning: many rootkits work by loading a kernel module which contains system-call wrappers — LKMs can be detected by scanning `/dev/kmem` for certain structures. Comparison of the results to those returned from `lsmod` can show up hidden — rootkit-related — LKMs.

10. Take a low-level filesystem dump

You want to get your hacked machine back in service as soon as possible; you don't want to lose any available forensic data; you want to determine how the intruder go access to you system: make a low-level copy/dump of your disks/partitions/slices — a copy using `rsync`, `cp -pr`, `tar` or similar high-level tools will not do, as any decent rootkit will have hidden itself, perhaps by wrapping system-calls.

[2] http://www.chkrootkit.org/

[3] http://www.rootkit.nl/

The plan: take a low-level copy; mount the copy on a trusted host and investigate — there should no longer be any rootkit-hidden files.

10.1. Standard Dump Tools

The standard tools for this kind of backup, or *dump* are: for Solaris, `ufsdump`; for Linux, `dump`. The latter, however is the subject of some discussion:

Pros:

- bypasses the kernel's filesystem interface — reads the filesystem through the block device — and therefore possibly wrapped/intercepted system calls;
- can handle unmounted filesystems;

Cons:

- handles Ext2 and Ext3 filesystems only (e.g., there is no ReiserFS dump utility);

10.2. Ghost

Norton/Symantec[4]'s Ghost can be used to make a copy of a filesystem. However, Ghost supports only Ext2 and Ext3, not ReiserFS, XFS, JFS...

10.3. `dd` or `cat /dev/[hs]d[a-z]\d+`

Enough! Any *real* Unix/Linux sysadmin will simply use `dd` or even `cat`. I have tried and tested both the following procedures. On the hacked machine

```
dd if=/dev/hda1 of=/scratch/hda1image
```

or

```
cat /dev/hda1 > /scratch/hda1.cat
```

then copy the image to a trusted machine and

```
cd /scratch
mkdir hda1
mount hda1image hda1 -t ext3 -o loop=/dev/loop2
```

Supports all filesystems that the kernel on the trusted machine can understand; works.

It is plausible that `dd` and/or `cat`, or something they depend on, will have been compromised in a way which will interfere with these procedures, so the truly paranoid should boot the hacked machine from a "live" CD, or attach its hard-disks to a trusted machine (as slave devices).

[4] `http://www.symantec.com`

11. Boot from a clean medium — Live CDs

To make a proper investigation of a machine that may have been `root`ed you need to boot from clean media. One option is to physically move the system disk from the hacked machine to another machine and mount it as a slave. A simpler way is to boot the machine from a "live CD". Suitable CDs include:

- Knoppix[5] or Knoppix-STD[6];
- Gnoppix[7]

There are many others.

12. Check both local and remote logs

Check the copy of your system logs (and kernel logs) on your remote syslog server (or, if no remote copy is available, your local logs, though these will almost certainly have been tampered with if your intruder has `root` access):

- anything interesting such as repeated attempts to authenticate to, for example SSH, or attempts at a buffer overrun:

  ```
  May 16 19:38:33 server rpc.statd[353]: gethostbyname error for ^Y^Y^[^[
  bffff760 8049710 8052c20687465676274736f6d616e7972652065207 26f7220726f66
  ```

- are there logins/authentications at unusual times or from unexpected hosts or IP addresses?

Also, check for *differences* between your local logs and the copy on your syslog server —

13. Establish the date/time of the intrusion and use it

If the date and time of the intrusion and/or of rootkit or other software installation can be determined the task of clearing up the damage is made much easier. The datestamp on a file can be changed to mislead; nevertheless this procedure is frequently worthwhile.

There are three obvious ways to determine the critical date and time:

- by noting the time/date when problems were noticed (e.g, from the details contained within the CERT report);
- from evidence in the (remote copy of the) system logs, or from *differences* in the logs — entries might be missing from the local copy;
- By finding (at least) one file which has been added or modified which should not have been: look for new libraries in `/lib` and `/usr/lib`, and utilities with new datestamps in `/bin`, `/usr/bin`, `/sbin` and `/usr/sbin`; look for recent changes in `/etc`.

Additionally, `chkrootkit` [Page 14] or Rootkit Hunter [Page 15] could spot, for example, changes to `/var/log/wtmp`.

[5] `http://www.knoppix.org`

[6] `http://www.knoppix-std.org`

[7] `http://www.gnoppix.org/`

Given an approximate date/time to work with, say 3 days ago, 2005 Jun 19, try

```
find / -ctime -2 -print
    # atime :  access --- the file was last accessed;
    # ctime :  change --- changes were made to the file’s inode;
    # mtime :  modify --- actual file contents changed;
```

or one of these (depending on the output format of `ls`)

```
ls -lR / |  grep "Jun 19" | egrep -v "2004|2003"
ls -lR / |  grep "2005-06-19"
```

For reliable results, this procedure should be done again after booting from reliable media [Page 8].

14. Check for a rootkit — again

Check for a rootkit again. Given that you have now booted from clean media and mounted the hacked system’s disk as a slave (or mounted a low-level dump), we are no longer looking for suspicious processes, connections or traffic; we focus on the filesystem. This time we have no potentially-wrapped/intercepted system-calls to worry about so can have more confidence in the results:

- verify all files, by using MD5 checksums, against your local database [Page 18], or via the the Solaris Fingerprint Database [Page 19], or similar, or, if you don’t have an MD5 database available, against another (clean) machine at the same patch-level.

- given the approximate date of the intrusion, use `find` and/or `ls -lR` [Page 9], again, to check for recent changes to files;

- retry `chkrootkit` and `rkhunter` — assuming the hacked disk is mounted at `/mnt/hacked`, then

```
chkrootkit -r /mnt/hacked
rkhunter --rootdir /mnt/hacked
```

15. Statically-Linked Binaries on a CD-R

On any machine which has likely been hacked, the installed utilities such as `ls`, `ps`, `top`, `netstat`, `ifconfig`, `stat`, `fuser`, `find`, `lsof`... should not be trusted: executables, or shared-object libraries on which they depend could easily have been trojanned. So statically-linked utilities from a mounted CD-R should always be used.

In fact, if system-calls are being intercepted/wrapped by an installed rootkit, then even this paranoia is not sufficient — in this case it is necessary to boot from clean media [Page 8], but its a good start, especialy if one is able to fingerprint system calls [Page 7] or otherwise check the kernel and rebooting your machine (server?) is not an option for a while.

To build statically-linked binaries you'll need the source:

`ftp.gnu.org/pub/gnu/bash`[8]	/bin	bash
`ftp.gnu.org/pub/gnu/coreutils`[9]	/bin	cat, dd, df, echo, ls, pwd,
	/usr/bin	du, stat, users, who
`ftp.gnu.org/pub/gnu/procps`[10]	/bin	kill, ps
	/usr/bin	free, pgrep, pkill, top, vmstat
`freshmeat.net/projects/net-tools`[11]	/bin	hostname, netstat
	/sbin	ifconfig, route
	/usr/sbin	arp, rarp
`ftp.gnu.org/pub/gnu/findutils`[12]	/usr/bin	find, locate, xargs
`ftp.gnu.org/pub/gnu/acct`[13]	/usr/bin	last

16. `lsof` and `fuser`

To build statically-linked binaries you'll need the source:

freshmeat.net[14] or perdue.edu[15]	lsof
sourceforge.net[16]	fuser, killall, pstree, pidof

16.1. `fuser`

`fuser` identifies processes using a given file. On Linux, use `fuser -m`.

To determine which processes are accessing the current working directory

```
fuser .        # Solaris
fuser -m .     # Linux

724c  1463c  1532c  1675c  5129cm...
```

To determine which process is responsible for this

```
tcp        0      0 localhost.localdom:6012 localhost.localdo:40404 ESTABLISHED
```

TCP connection on port 6011

```
prompt>fuser -n tcp 6011      # Linux, not Solaris
6011/tcp:      9060

prompt>ps auxw | grep 9060
umits     9060  0.0  0.3   9412  1592 ?        S    Jun02   4:38 sshd: umits@pts/36
```

16.2. `lsof`

`lsof` lists open files — that is “regular” files, network connections, directories… Output shows processes and their open files.

Example output:

```
COMMAND   PID     USER   FD    TYPE   DEVICE     SIZE      NODE NAME
init        1     root  cwd     DIR      3,2     1024         2 /
init        1     root  rtd     DIR      3,2     1024         2 /
init        1     root  txt     REG      3,2    31432    105874 /sbin/  init
init        1     root  mem     REG      3,2    90248     85518 /lib/ld-2.3.2.so
init        1     root  mem     REG      3,2  1244688     85606 /lib/libc-2.3.2.so
init        1     root   10u   FIFO      3,2              73308 /dev/initctl
keventd     2     root  cwd     DIR      3,2     1024         2 /
.


.
ssh       724   simonh  cwd     DIR      3,8     4096     32129 /home/simonh
ssh       724   simonh  rtd     DIR      3,2     1024         2 /
ssh       724   simonh  txt     REG      3,5   226168    244449 /usr/bin/ssh
ssh       724   simonh  mem     REG      3,2    90248     85518 /lib/ld-2.3.2.so
ssh       724   simonh  mem     REG      3,2    64924     85644 /lib/libresolv-2.3.2.so
ssh       724   simonh  mem     REG      3,5  1042728    358474 \
                                                  /usr/lib/i686/cmov/libcrypto.so.0.9.7
.


.
ssh       724   simonh    1u    CHR    136,9                 11 /dev/pts/9
ssh       724   simonh    2u    CHR    136,9                 11 /dev/pts/9
ssh       724   simonh    3u   IPv4  2027003                TCP \
                     mctalby.mc.man.ac.uk:55914->darkstar.umist.ac.uk:ssh (ESTABLISHED)
.


.
bash     1463   simonh  cwd     DIR      3,8     4096     32129 /home/simonh
bash     1463   simonh  rtd     DIR      3,2     1024         2 /
bash     1463   simonh  txt     REG      3,2   667180     69229 /bin/bash
bash     1463   simonh  DEL     REG      3,2              85599 /lib/ld-2.3.2.so.dpkg-new
.
.
```

Output can be restricted to only network connections:

```
lsof -i

COMMAND     PID        USER   FD   TYPE  DEVICE SIZE NODE NAME
ssh         724      simonh    3u  IPv4 2027003       TCP \
                       mctalby.mc.man.ac.uk:55914->darkstar.umist.ac.uk:ssh (ESTABLISHED)
ssh        1532      simonh    3u  IPv4 1054576       TCP \
                           mctalby.mc.man.ac.uk:60035->bohrg3.man.ac.uk:484 (ESTABLISHED)
ssh        1690      simonh    3u  IPv4 1055280       TCP \
                           mctalby.mc.man.ac.uk:60069->bohrg3.man.ac.uk:484 (ESTABLISHED)
.


.
firefox-b  2125         si2    3u  IPv4 2000944       TCP \
                    localhost.localdomain:54229->localhost.localdomain:6013 (ESTABLISHED)
firefox-b  2125         si2   27u  IPv4 2029291       TCP \
                    localhost.localdomain:56045->localhost.localdomain:6013 (ESTABLISHED)
sshd       2947        root    3u  IPv4 2628966       TCP *:ssh (LISTEN)
emacs      3201       umits    4u  IPv4 2480726       TCP \
                    localhost.localdomain:52828->localhost.localdomain:6011 (ESTABLISHED)
XFree86    3448        root    1u  IPv4   10336       TCP *:x11 (LISTEN)
httpproxy  4801        root    0u  IPv4 2635892       TCP \
                       mctalby.mc.man.ac.uk:httpproxy->61.175.228.137:44104 (ESTABLISHED)
httpproxy  4801        root    1u  IPv4 2635892       TCP \
                       mctalby.mc.man.ac.uk:httpproxy->61.175.228.137:44104 (ESTABLISHED)
.
.
```

To determine which process is responsible for this

```
  tcp        0      0 localhost.localdom:6012 localhost.localdo:40404 ESTABLISHED
```

try

```
lsof -i -n | grep 6012

firefox-b 10930  mc   3u  IPv4 2230197  TCP 127.0.0.1:40404->127.0.0.1:6012 (ESTABLISHED)
sshd      15330  mc  10u  IPv4   68837  TCP 127.0.0.1:6012 (LISTEN)
sshd      15330  mc  12u  IPv4 2230199  TCP 127.0.0.1:6012->127.0.0.1:40404 (ESTABLISHED)
sshd      15330  mc  13u  IPv4  863595  TCP 127.0.0.1:6012->127.0.0.1:49477 (ESTABLISHED)
emacs     18021  mc   4u  IPv4  863593  TCP 127.0.0.1:49477->127.0.0.1:6012 (ESTABLISHED)
firefox-b 19079  mc   3u  IPv4 2230197  TCP 127.0.0.1:40404->127.0.0.1:6012 (ESTABLISHED)
firefox-b 19090  mc   3u  IPv4 2230197  TCP 127.0.0.1:40404->127.0.0.1:6012 (ESTABLISHED)
firefox-b 19091  mc   3u  IPv4 2230197  TCP 127.0.0.1:40404->127.0.0.1:6012 (ESTABLISHED)
firefox-b 19093  mc   3u  IPv4 2230197  TCP 127.0.0.1:40404->127.0.0.1:6012 (ESTABLISHED)
```

17. `chkrootkit` and `rkhunter`

There is no way to be certain whether or not a machine has been rooted without booting from clean media. But these machines help — alot.

17.1. `chkrootkit`

From the `www.chkrootkit.org`[17] website:

`chkrootkit` is a tool to locally check for signs of a rootkit. It contains:

- `chkrootkit` shell script that checks system binaries for rootkit modification.
- `ifpromisc.c` checks if the interface is in promiscuous mode.
- `chklastlog.c` checks for lastlog deletions.
- `chkwtmp.c` checks for wtmp deletions.
- `check_wtmpx.c` checks for wtmpx deletions. (Solaris only)
- `chkproc.c` checks for signs of LKM trojans.
- `chkdirs.c` checks for signs of LKM trojans.
- `strings.c` quick and dirty strings replacement.
- `chkutmp.c` checks for utmp deletions.

Read the `man` page, or simply:

```
root> chkrootkit -h
Usage: /usr/sbin/chkrootkit [options] [test ...]
Options:
    -h                show this help and exit
    -V                show version information and exit
    -l                show available tests and exit
    -d                debug
    -q                quiet mode
    -x                expert mode
    -r dir            use dir as the root directory
    -p dir1:dir2:dirN path for the external commands used by chkrootkit
    -n                skip NFS mounted dirs
```

Quiet mode is good for a daily cron job.

17.2. RK Hunter

From the www.rkhunter.org[18] (also rootkit.nl[19]) website:

Rootkit Hunter

- Shell script
- No program dependencies (except optional Perl modules)
- Works on almost every UNIX-alike operating system (BASH shell preferred)

`rkhunter` is written in Perl so its easy to get a good idea of what it's doing as it performs it's tests.

[17] `http://www.chkrootkit.org`

[18] `http://www.rkhunter.org/`

[19] `http://rootkit.nl/`

There are many usage options; here are some:

```
 rkhunter <parameters>

   --checkall (or -c)
       Check the system, performs all tests.

   --createlogfile*
       Create a logfile (default /var/log/rkhunter.log)

   --cronjob
       Run as cronjob (removes colored layout)

   --help (or -h)
       Show help about usage

   --nocolors*
       Don't use colors for output (some terminals don't like
       colors or extended layout characters)

   --report-mode*
       Don't show uninteresting information for reports, like
       header/footer.  Interesting when scanning from crontab or with
       usage of other applications.

   --skip-keypress*
       Don't wait after every test (makes it non-interactive)
```

18. strace — Spying on Processes and Users

strace is your friend. To see what a suspicious process is doing try this

```
strace -p <process id>
```

To spy on a pseudoterminal, identify the process-id associated with it and

```
strace -e read,write -p <process id>
```

A Perl wrapper called ttylog [Page 16] is available for the above which nicely formats the output.

19. ttylog: a pty spy gizmo

ttylog is a Perl script written by Rob Brown intended for attaching to currently running pty or tty sessions. It is essentially a pretty-printing wrapper for

```
strace -e read,write -s16384 -x -o $write -p $pid
```

where $pid is the process-id of a running pty. It can be used for logging or helping users. Great for spying on suspect users or possible hackers.

The source is available from CPAN, rather oddly under Apache/BBB[20].

[20] http://www.cpan.org/modules/by-module/Apache/BBB/

(This utility should not to be confused with the binary utility called `ttylog` which copies output from `/dev/tty*` to `stdout`.)

20. tcpdump, ngrep and ethereal

`tcpdump` and `ethereal` are network traffic analysis tools; the former is a rough-and-ready command-line utility, while the latter has a GUI and can in addition analyse many protocols.

20.1. tcpdump

`tcpdump`[21] prints out the headers of packets on a given network interface which match the given boolean expression.

Some examples:

```
tcpdump -i hme0
    # print all packet headers arriving at or leaving interface hme0

tcpdump talby.csu.umist.ac.uk
    # print all packet headers arriving from or leaving for talby
    # on the default interface

tcpdump 'tcp port 80'
    # print tcp traffic on local port 80

tcpdump 'host not talby.csu.man.ac.uk'
    # print everything except packets to/from
```

20.2. Ethereal

Ethereal[22] is like `tcpdump` with a GUI and (more importantly) a protocol analyser — `tcpdump` on steroids.

20.3. ngrep

From the home page:

`ngrep`[23] strives to provide most of GNU grep's common features, applying them to the network layer. `ngrep` is a pcap-aware tool [cf. `tcpdump`] that will allow you to specify extended regular or hexadecimal expressions to match against data payloads of packets. It currently recognizes TCP, UDP, ICMP... and understands bpf filter logic in the same fashion as... `tcpdump`....

[21] `http://www.tcpdump.org`

[22] `http://www.ethereal.com/`

[23] `http://ngrep.sourceforge.net/`

Examples:

```
ngrep -d any port 25
    # ...any device

ngrep -i -d any 'error' port syslog
    # ...monitor SysLog traffic (port 514) for the string "error"
    #    (case-insensitive)

ngrep -wi -d any 'user|pass' port 21
    # ...traffic on src/dest port 21, look (case-insensitive) for
    #    "user" and "pass" as word-expressions (must have non-alphanumeric
    #    delimiting characters) --- sniff out credentials
```

21. `ntop` and `bandwidthd`

`ntop` and BandwidthD are yet further utilities built on `libpcap`. Both are more useful for long term monitoring of network traffic as a general policy of looking for suspect packets than for speedy forensics.

21.1. `ntop`

From the homepage:

`ntop`[24] is a network traffic probe ... uses a Web browser for the interface... configurable via the browser...

21.2. BandwidthD

From the homepage:

BandwidthD[25] tracks usage of TCP/IP network subnets and builds HTML files with graphs to display utilization. Charts are built by individual IPs, and by default display utilization over 2 day...

22. MD5 and Fingerprint Database

A simple and powerful way to determine if executable binaries or scripts, or libraries have been trojanned is to maintain and use a MD5 checksum database. Several popular utilities are available which implement this idea including Tripwire[26] (commercial software), AIDE[27] and Cheesewire[28]; inode values can also be usefully stored.

[24] `http://www.ntop.org/`

[25] `http://bandwidthd.sourceforge.net/`

[26] `http://www.tripwire.com`

[27] `http://www.cs.tut.fi/~rammer/aide.html`

[28] `http://talby.csu.umist.ac.uk/~isd/_cheesewire/`

Usage is simple: update the database each time the system is patched; copy the database to a remote, secure location (or burn to CD-R); periodically compare MD5 values (and inode values) of installed files to those in the database — or check after a suspected intrusion. Differences from the database indicate trojanned files. For most reliable results, mount the suspect filesystem as slave after booting from clean media.

23. Solaris Fingerprint Database

Sun Microsystems offer a MD5 fingerprint database at their sunsolve.sun.com[29] site. This can be used to test the integrity of utilities and libraries in a Solaris installation and answer the question — are these files trojanned? For reliable results, mount the susect filesystems as slave after booting from clean media.

There are two ways to use the system: simply enter MD5 values into the form on the web page and these will be checked for you. For example,
To use this facility, simply obtain the MD5 checksum by some means (e.g., by using `/opt/md5/md5-sparc`, for example). Following are two examples. First, `/usr/bin/netstat`:

```
41f06010aba241ea34e86a130fded6d4 -  - 2 match(es)

       * canonical-path: /usr/bin/netstat
       * package: SUNWcsu
       * version: 11.7.0,REV=1998.09.01.04.16
       * architecture: sparc
       * source: Solaris 7/SPARC

       * canonical-path: /usr/bin/netstat
       * package: SUNWcsu
       * version: 11.7.0,REV=1998.10.06.00.59
       * architecture: sparc
       * source: Solaris 7/SPARC
```

Second, `/bin/ps`:

```
120397cfdd451d448d3094042e7c473b -  - 1 match(es)

       * canonical-path: /usr/lib/isaexec
       * package: SUNWcsu
       * version: 11.7.0,REV=1998.10.06.00.59
       * architecture: sparc
       * source: Solaris 7/SPARC
       * patch: 106541-40
```

In the second case, a generic answer is produced — as long as some result of this type is returned, the file is genuine (see the Sunvolve-provided FAQ for details).

To help checking large numbers of files, Sun make the complete MD5 list available for download as a compressed tar file[30].

[29] `http://sunsolve.sun.com/pub-cgi/fileFingerprints.pl`

[30] `http://sunsolve.sun.com/md5/md5.tar.Z`

24. rpm and apt-get

Both RedHat's rpm package management system and Debian's system contain some in-build integrity checking.

The command rpm -Va will verify all packages installed, that is: file size, MD5 sum, permissions, type, owner and group of all files in all installed packages is compared against the metadata stored in the RPM database. Discrepancies are displayed. The obvious weakness of this approach is that the metadata is stored in a local database which can itself be hacked.

Debian's package system comes with the debsums utility which checks the MD5 sums of installed Debian packages against metadata in the local database. Again, a weakness is that metadata is stored locally and is open to intruders. However, debsums can be told to generate MD5 sums from .deb files (rather than use local metadata) and these can be freshly downloaded to /var/cache/apt/archives:

```
apt-get clean
    # ...ensure don't use old .deb packages for fear of corruption

apt-get --reinstall -d install `dpkg -l | grep ii | awk '{print $2}'`
    # ..."reinstall all packages" --- actually "-d" ensures each is
    #    downloaded to local cache only (don't overwrite installed files)

debsums -g -p /var/cache/apt/archives
    # ...check MD5 sums against those generated from .deb files just
    #    downloaded downloaded to local cache.
```

As usual, discrepancies indicate trojanned files. Given discrepancies, it is best to reinstall everything:

```
dpkg -l | grep ii | awk '{print $2}' | xargs -i{} apt-get -y --reinstall install {}
```

About this page:

Produced from the SGML: /home/mc/public_html/_unix_security/_reml_grp/diagnostic_forensic_tools.reml

On: 4/5/2006 at 13:50:22

Options: reml2 -l nolong -o tex -p single