To appear in *Proceedings of the Third Usenix UNIX Security Sympisum*, Baltimore, September 1992.

There Be Dragons

Steven M. Bellovin
AT&T Bell Laboratories
Murray Hill, NJ
smb@ulysses.att.com

August 15, 1992

Abstract

Our security gateway to the Internet, `research.att.com`, provides only a limited set of services. Most of the standard servers have been replaced by a variety of trap programs that look for attacks. Using these, we have detected a wide variety of pokes, ranging from simple doorknob-twisting to determined assaults. The attacks range from simple attempts to log in as `guest` to forged NFS packets. We believe that many other sites are being probed but are unaware of it: the standard network daemons do not provide administrators with either appropriate controls and filters or with the logging necessary to detect attacks.

1 Introduction

"Queer things you do hear these days, to be sure," said Sam.

"Ah,", said Ted, "you do, if you listen. But I can hear fireside-tales and children's stories at home, if I want to."

"No doubt you can," retorted Sam, "and I daresay there's more truth in some of them than you reckon. Who invented the stories anyway? Take dragons now."

"No thank 'ee," said Ted, "I won't. I heard tell of them when I was a youngster, but there's no call to believe in them now. There's only one Dragon in Bywater, and that's Green," he said, getting a general laugh.

J.R.R. Tolkien, *Lord of the Rings*

By now, it is widely accepted that, among other denizens of the Internet, lurk crackers.[1] For whatever reason, these folks enjoy breaking into various computer systems. AT&T appears to be a tempting target. Our approach to this problem is two-fold. First, most machines here are not directly connected to the Internet. Rather, we rely on application-level gateways and *proxy servers*[Che90]. Second, we employ a variety of monitors and phony daemons. Instead of providing services useful to both legitimate users and crackers,

[1] Some call them "crackers", and some call them "hackers". A compromise term might be "chrackers". We think that "vandals" is more appropriate, though those of a classical bent may prefer "Vandals", or even "Goths" or "Visigoths".

these log the request, and initiate *counterintelligence* strategies to learn something about the source of the request.

We are certainly not the first ones to attempt to trick attackers[Sto88, Sto89, HM91]. But our motivation is somewhat different. We do not expect to prosecute, because (we hope) no damage will occur to our machines. (This is not to say that the attackers do not try such things; see, for example, [Che92].) Nor, in general, do we care much about the identity of any particular attacker. Rather, we wish to study the attackers' strategies, tools, and techniques. Our goal is to learn what kinds of attacks are employed, both to warn others and to protect our own networks from internal crackers or from outsiders who have already gained a foothold within our network.

A word on the alarm messages shown. All of them are genuine, taken straight from our log files. However, the domain names, user names, logins, and IP addresses have been changed to protect the privacy of those concerned.

2 Tools and Traps

Our basic strategy is simple: except for the few servers we actually need — mail, `ftp`, and `telnet` — we run dummy servers for likely services. Some of these are quite specialized; others are generic packet suckers. All of them, though, log the incoming data, attempt to trace back the call, and — when feasible — try to distinguish between legitimate users and outside attackers.

The `finger` server is a good example. Attempts to `finger` a particular user are usually benign attempts to learn an electronic mail address. But that would not work even without our monitor program, since most users do not have logins on the gateway machine. Instead, we print a message explaining how to send mail by name. Generic `finger` attempts, though, are often used to gather login names for cracking attempts. Therefore, completely bogus output is returned, showing that `guest` and `berferd` — a dummy user name — are logged in. Counterintelligence moves, which include "reverse `finger`s", are not done in this case, for fear of triggering a `finger` war. And all attempts are logged, for later analysis.

The so-called "`r`-commands" also merit a special server, because of the extra information they provide. For `rlogin` and `rsh`, the protocol includes both the originating user's login name and the login name desired on our gateway. Thus, we can do a precisely-aimed reverse `finger`, and we can assess the level of the threat. A login attempt by some user `foo`, and requesting the same login on `research.att.com`, is probably a harmless error. On the other hand, an attempt by `bin` to execute the `domainname` command as `bin` — see Figure 1 — represents enemy action. (It also suggests that the attacking machine has been compromised. Note, too, that all of the people shown as logged in are idle.) Attempts to `rlogin` as `guest` from a legitimate account usually fall in the doorknob-twisting category.

For most other services, we rely on a simple packet sucker. That is, a program invoked by `inetd` sits on the socket, reading and logging anything that comes along. While that is happening, counterintelligence moves are initiated. The TCP packet sucker exits when the connection is closed; the UDP version relies on a timeout, but will also exit if a packet arrives from some other source. The information gained from such a simple technique can be quite interesting; see Figure 2. It shows an attempt to grab our password file via `tftp`.

Experience with the packet sucker showed us that there were a significant number of requests for the `portmapper` service. The `portmapper`, part of Sun Microsystem's RPC package, maps a program identifier to a dynamically-assigned port number[Sun90]. The

```
From: adm@research.att.com
To: trappers

Attempted rsh to inet[24640]
Call from host Some.Random.COM (176.75.92.87)
remuser: bin
locuser: bin
command: domainname


(/usr/ucb/finger @176.75.92.87; /usr/ucb/finger bin@176.75.92.87) 2>&1
[176.75.92.87]
Login       Name              TTY Idle    When    Where
rel      R. Locke              co   4d Sat 11:26
afu      Albert Urban          p0  10: Fri 13:51  seed.random.com
rlh      Richard L Hart        p2 3:18 Sat 20:27  fatso1.random.c
rel      R. Locke              p4   3d Mon 09:05  taxi.random.com
[176.75.92.87]
Login name: bin
Directory: /bin
Never logged in.
No unread mail
No Plan.
```

Figure 1: An attack via `rsh`.

```
From: adm@research.att.com
To: trappers
Subject: udpsuck tftp(69)

UDP packet from host some.small.edu (125.76.83.163): port 1406, 23 bytes
    0:   00012f65 74632f70 61737377 64006e65   ../etc/passwd.ne
   16:   74617363 696900                       tascii.
/usr/ucb/finger @125.76.83.163 2>&1
[125.76.83.163]
No one logged on

4 more packets received
```

Figure 2: Spoor of an attack detected by the UDP packet sucker.

usual protocol is for the client to contact the server's `portmapper` to learn what port that service is currently using. The `portmapper` supplies that information, and the client proceeds to contact the server directly. This meant, though, that we were seeing only the identifier of the service being requested, and not the actual call to it. Accordingly, we decided to simulate the `portmapper` itself.

Our version, called the `portmopper`, does not keep track of any real registrations. Rather, when someone requests a service, a new socket is created, and its (random) port number is used in the reply. Naturally, we attach a packet sucker to this new port, so we can capture the RPC call.

Figure 3 shows excerpts from a typical session. We print and decode all the goo in the packet, because we do not know if someone might try RPC-level subversion. The first useful datum is delimited by `***` lines; it shows a request for the mount daemon, using TCP. Our reply (not shown) assigned port `0x691` to this session. Finally, the input on that port shows that procedure 2 is being called, with no parameters. There is currently no code to interpret the procedure numbers, but a quick glance at `/usr/include/rpcsvc/mount.h` shows that it's a dump request, i.e., a request for a list of all machines mounting any of our file systems. It is also worth noting that our counterintelligence attempt failed; the machine in question is not running a `finger` daemon.

An alternate approach would have been to use the standard `portmapper`, and to have packet suckers registered for each interesting service. We rejected this approach for several reasons. First and foremost, we have no reason to trust the security of the `portmapper` code or the associated RPC library. We are not saying, of course, that they have security holes; rather, we are saying that we do not know if they do. And we are morally certain that legions of would-be crackers are studying the code at this very moment, looking for holes. To be sure, we do not know that our code is bug-free; it is, however, smaller and simpler, and hence less likely to be buggy. (It is also relatively unknown, a non-trivial advantage.)

A second reason for eschewing the `portmapper` is that we do not know what the "interesting" services are. Our approach does not require that we know in advance; instead, we can detect requests for anything.

A third reason is that by its nature, the RPC library provides a high-level abstraction to the actual packets. This is useful for programmers, but bad for us; if, say, someone is playing games with the authenticators, we want to know about it.

Finally, we wanted our code to be very portable. In particular, we want it to run on Plan 9 machines[PPTT90]. As of now, no one has ported RPC to Plan 9. Doing so might not be a lot of work, but it is not work we are interested in performing.

2.1 Address Space Probes

Our gateway, `research.att.com`, is a well-known machine, and hence attracts crackers. A clever cracker, though, might investigate further, looking for other likely machines to try. There seemed to be two possibilities: blind probing of the address space, or examination of our domain name system (DNS) data[Moc87]. We decided to monitor for such attempts.

The obvious way to do such monitoring is to put a network controller into promiscuous mode and watch the packets fly by. Indeed, we did do just that; however, the solution was not at all straight-forward. The gateway machine runs RISC/os[2]; to our knowledge, it has no user-level mechanisms analagous to Sun's `nit` driver. We did have a SPARCstation[3]

[2] RISC/os is a trademark of MIPS Computer Corporation

[3] SPARCstation is a trademark of SPARC International, Inc.

```
From: adm@research.att.com
To: trappers
Subject: UDP portmopper from Another.COM (176.143.143.175)

Request:
    0:   2974eaca 00000000 00000002 000186a0   )t..............
   16:   00000002 00000003 00000000 00000000   ................
   32:   00000000 00000000 000186a5 00000001   ................
   48:   00000006 00000000                     ........
xid: 2974eaca msgtype: 0 (call)
rpcvers: 2 prog: 100000 (portmapper)  vers: 2 proc: 3 (getport)
Authenticator: credentials
Authtype: 0 (none) length: 0
Authenticator: verifier
Authtype: 0 (none) length: 0

***
reqprog: 100005 (mountd) vers: 1 proto: 6 port: 0
***
...
/usr/ucb/finger @176.143.143.175 2>&1
[176.143.143.175]
connect: Connection refused

Server input:
    0:   2976c57d 00000000 00000002 000186a5   )v.}............
   16:   00000001 00000002 00000000 00000000   ................
   32:   00000000 00000000                     ........
xid: 2976c57d msgtype: 0 (call)
rpcvers: 2 prog: 100005 (mountd)  vers: 1 proc: 2
Authenticator: credentials
Authtype: 0 (none) length: 0
Authenticator: verifier
Authtype: 0 (none) length: 0
Parameters:
```

Figure 3: Output from the `portmopper`.

that we could connect to the net; since that machine is not adequately secure, we had a wire cutter introduce itself to the transmit leads on the drop cable.

Although we could now listen, we could not learn as much as we would like. Upon seeing a packet for a new machine, our router's instinct is to issue an ARP request[Plu82]. For non-existent machines, of course, no one can answer. Ideally, the monitoring machine would pick up such requests and provide a proxy ARP reply. Unfortunately, our security measures rendered that idea impractical. We thus have `research.att.com` handling proxy ARP for non-existent machines to point them towards the monitoring machine, a bizarre situation indeed. A final problem was that the ARP table is limited in size, so we could not provide complete coverage of the address space. We settled for the machines listed in the DNS, and for a few machines at either end of the range to detect counting up or counting down. Finally, we used the `tcpdump` program to do the monitoring; there was no point to building a special-purpose packet decoder when a very nice general one already existed.

The results of this trap have been rather curious. We have noticed a large number of `ftp` connection requests to `192.20.225.1`, a machine that has not existed for quite some time. Furthermore, the large majority of these connection attempts have come from abroad. We speculate that some old databases still list its address.

We have noticed a few attempts to connect to other machines. For the most part, these have been to DNS-listed addresses, rather than to random places on our network, and the one or two exceptions appear to be accidental. This log file is not examined in real time, so we have not been able to engage in our usual counterintelligence measures. Comparing the source addresses and timestamps with our other log files tends to show other forms of snooping going on. Such probes should likely be considered as hostile.

One set of probes was especially alarming. Immediately following the arrest of two alleged non-U.S. system crackers, someone else from that country launched a systematic probe of our network's address space. Our known machine was ignored. We believe that this was an attempt at revenge, and that our well-instrumented gateway machine was ignored because the attackers knew it for what it was.

Of late, we have seen concerted attempts to connect to random addresses of ours. The pattern does not suggest an attack; rather, it suggests hosts that are quite confused about our proper IP address. The problem appears to be corrupted DNS entries, which we have also experienced, rather than any security problem. This problem is discussed further in [Bel92].

2.2 Counterintelligence

When a probe occurs, we try to learn as much about the originating machine and user as we can. Thus far, the only generally-available mechanism to do that is the `finger` command. While far better than nothing, it has some weaknesses. Clever crackers have any number of ways to cover their tracks, such as overwriting `/etc/utmp` (it is world-writable on many systems) or using the appropriate options to `xterm`. And indeed, we have seen attacks from machines that claim to have no one logged in, viz. Figure 2.

There is also the problem of pokes originating from security-conscious sites. Often, these sites restrict or disable the `finger` daemon, for all the obvious reasons. Figure 3 shows an example. (That particular probe turned out to be an experiment by a friend.) To be sure, security-conscious sites are probably the least-likely to be penetrated. But no one is immune; one of our own theoretically-secure gateways was successfully attacked over a weekend, due to operator error.

Some sites take their own security precautions. One (unsolicited) prober noticed our reverse `finger` attempt, and congratulated us on it. Others who thought we were running a "cracker challenge contest" were able to detect our activities when specifically looking for them. The worst possibility would be an active response to our probe; it could easily trigger a recursive `finger`ing contest. For this reason, among others, we do not currently do reverse `finger`s in response to `finger` queries, but the problem could still arise. For example, an `rusers` query to us would trigger the `portmopper`'s counterintelligence probes; these in turn could cause the remote site to query our `rusers` daemon. It may be necessary to add some locking to our daemons.

We have contemplated adding other arrows to our counterintelligence quiver, but there are few choices available. The `rusers` command is an obvious possibility, but it offers less information than `finger` does. To be sure, because it goes through the `portmapper`, it is harder to block or monitor; unfortunately, many sites block all outside calls to the `portmapper` because of (valid) concerns about the security of some RPC-based services. Another choice would be the Authentication Server[Joh85], but our experiments show that very few sites support it. And SNMP[CFSD90] is generally implemented on routers, not hosts.

A totally different set of investigations are performed using DNS data. First of all, we attempt to learn the host name associated with the prober's IP address, which should be a trivial matter. In theory, all addresses should listed in the inverse mapping tree; in practice, many are not. This problem seems to be especially commonplace overseas, probably due to the newness of the connections. In such cases, we have to look for the SOA and NS records associated with the inverse domain; using them, we attempt a zone transfer of the inverse domain, and scan it for any host names at all. That, finally, gives the zone name; we then transfer the forward-mapping zone and search for the target's address.

On a few occasions, this procedure has failed; we have been forced to resort to the use of `traceroute`s, manual `finger` attempts, and even a few `telnet` connections to various ports to see if any servers announce the host and domain name. Needless to say, none of this is automated; if a simple `gethostbyaddr()` call fails, we perform any further investigations ourselves.

There is one DNS-related check that we do automate, however. It is by now well-known that evil games can be played with the inverse mapping tree of the DNS. To detect this, we perform a cross-check; using the returned name, we do a forward check to learn the legal addresses for that host. If that name is not listed, or if the addresses do not match, alarms, gongs, and tocsins are sounded.

2.3 Log-Based Monitoring Tools

A number of our monitors are based on periodic analyses of logs. For example, attempts to grab a (phony) password file via `ftp` are detected by a `grep` job run via `cron`. We thus cannot engage in counterintelligence activity in response to such pokes. Nevertheless, they remain very useful. These monitors — and a serious attack discovered via them — are described more fully in [Che92].

We also discovered that our gateway machine was being used as a repository for (presumably stolen) PC software. Assorted individuals would store such programs under a directory named "`..^T`", where "`^T`" represents the control-T character; others would retrieve it at their leisure. We idly discussed replacing these files with programs that printed nasty warnings, but settled for clearing out the incoming `ftp` area at least daily. That seems to have stopped the problem for now, though a better solution would be to add the notion of "inside versus outside" to the daemon, and to prohibit transfers that did not cross the boundary. (Other sites report similar incidents, often involving digitized erotic images. We leave to the readers' imagination what we could insert in place of these files.)

We are currently adding real-time analyzers to some of our logs. The implementation is simple:

`tail -f` *logfile* `| awk -f` *script*

This is an especially useful techique for the `ftp` daemon's logs; attempts to add more sophisticated mechanisms to the daemon itself would run afoul of the `chroot` environment it currently runs in.

There is danger lurking here. Our early versions could easily have fallen victim to a sophisticated attacker who used file names containing embedded shell commands. For this reason, among others, we run all of our traps with as few privileges as possible. In particular, where possible we do not run them as `root`.

3 Attacks Discovered

Thus far, we have seen a wide variety of attacks. Some of them are well-known, of course; there is nothing novel about password-guessing crackers. A typical scenario starts with a `finger` attempt; our pseudo-server returns output indicating that `guest` and `berferd` are logged in. Both of these accounts have obvious passwords; if the cracker takes the bait, we initiate counterintelligence measures. An attempt to log in as `guest` is in some sense less serious; one can make a plausible argument that sites that do not want guests should not have a `guest` account. No such excuse can be offered for trying to log in as an apparent genuine user.

The next level up are folks who want our password file. Our `ftp` daemon provides a dummy one (see [Che92] for details); a packet sucker catches `tftp` requests for it. We have contemplated the idea of distributing the same dummy file via `tftp`, but have rejected it; the benefit to us would be minimal, and we would have to expose ourselves to possible bugs in the `tftp` daemon.

There have been a fair number of attempts to `rlogin` to our machine. Most of these appear to be innocent, though curious nevertheless: why would anyone expect to be able to log in to another company's machines? Sometimes, we see attempts to connect as `netlib`, or to `rcp` the `netlib` distribution[DG87]; these most likely denote a somewhat-naive attempt to avoid the use of `ftp` when retrieving the `netlib` package we distribute. For other connections, we believe that fingers are faster than brains; the real intent was to use `ftp` or `telnet` to reach us. Regardless, such attempts represent noise in the log files.

Other connection requests have not been so genteel. We have seen attempts to `rlogin` as `root` coming from military sites. Figure 1 shows an attempt to execute the `domainname` command; apart from the obvious problem that exists if `bin` can connect to our machine, we suspect that the attacker planned mischief involving Sun's NIS.

The `portmopper`, and before that the UDP packet sucker, have picked up a number of RPC-related probes; the intent of some of these is unclear. We have no idea, for example, why someone would try to contact the `rstatd` daemon. There may be security problems lurking there. Other requests are most likely malicious; when someone tries to contact our (non-existent) `NFS` mount daemon, we assume that they are looking for file systems exported to the world. (Yes, there are many sites with that problem.)

```
From: adm@research.att.com
To: trappers
Subject: udpsuck nfs(2049)

UDP packet from host a.non-us.edu (173.46.173.146): port 804, 40 bytes
    0:   2964e5a6 00000000 00000002 000186a3   )d..............
   16:   00000002 00000000 00000000 00000000   ................
   32:   00000000 00000000                     ........
/usr/ucb/finger @173.46.173.146 2>&1
[173.46.173.146]
Login       Name              TTY Idle    When    Where
lu       Lee User              a  8:41 Fri 12:55  direct to room 101
ano      A.N. One              h6   3d Tue 00:49  direct to 719
nsa      Nun Atall             p0   36 Thu 18:56  eqg01:0.0
nsa      Nun Atall             p1   24 Thu 18:57  eqg01:0.0
```

Figure 4: A captured NFS request

There have been some connection requests to more obscure services. Several people have poked a packet sucker sitting on the whois port. Those have been innocent; generally, the captured data showed that the caller wanted the email address of researchers here. When feasible, we reply by email, doubtless causing much confusion and puzzlement. We will likely disable that trap in the near future. Other probers have connected to things like like the nntp port. We do not know for certain what they had in mind; likely guesses include attempts to read newsgroups not carried at their own sites, or attempts to forge netnews postings.

The most sophisticated pokes have been attempted NFS operations[Sun90]. They may have been hand-crafted, as most normal NFS operations are preceded by mount requests. A sample alarm message is shown as Figure 4. Perhaps not surprisingly, the users shown as logged in have all been idle for quite some time.

Thus far, all of the NFS packets we have captured have been no-ops. In a few instances, we have been able to contact the individuals responsible; they generally replied that they were checking to see if our archives were accessible by NFS as well as by ftp. (A number of sites do provide this option; we marvel at their courage.) In fact, at least one popular program — the amd auto-mounter[Pen] — apparently generates NFS no-ops automatically.

We are starting to see worrisome levels of such queries. Given the existence of public NFS archives, checking to see if we offer such a service cannot be considered a hostile act. On the other hand, what we see with our current tools — NFS no-ops and queries to the mount daemon — are not distinguishable from a genuine attack. Our choices are either to ignore all such requests, or to emulate more of the protocol, so we can see what is really intended. Neither alternative is appealing.

We have recently seen several determined attempts to grab our password file via NIS (Figure 5). The attackers' programs made repeated attempts to guess our NIS domain name, which is need in order to perform the transfer. Perhaps not surprisingly, these attempts occurred just a few weeks after the appropriate program was posted to a newsgroup.

There are several likely services where we have not, or not yet, received any serious pokes, such as bootp or X11. (Actually, we have seen a few connection attempts to our X11

```
From: adm@research.att.com
To: trappers
Subject: UDP portmopper from several.different.places (230.154.230.241)

Request:
....
***
reqprog: 100004 (ypserv) vers: 2 proto: 6 port: 0
***

...

/usr/ucb/finger @230.154.230.241
[230.154.230.241]
No one logged on


Server input:
    0:   2a36be5f 00000000 00000002 000186a4   *6._............
   16:   00000002 00000004 00000001 0000001c   ................
   32:   2a3b6cfa 00000004 69736673 00000000   *;l.....isfs....
   48:   00000000 00000001 00000000 00000000   ................
   64:   00000000 0000000c 3139322e 32302e32   ........192.20.2
   80:   32352e32 0000000d 70617373 77642e62   25.2....passwd.b
   96:   796e616d 65000000 80000060 2a36be5e   yname......`*6.^
  112:   00000000 00000002 000186a4 00000002   ................
  128:   00000004 00000001 0000001c 2a3b6cfa   ............*;l.
  144:   00000004 69736673 00000000 00000000   ....isfs........
  160:   00000001 00000000 00000000 00000000   ................
  176:   00000003 31393200 0000000d 70617373   ....192.....pass
  192:   77642e62 796e616d 65000000 80000064   wd.byname......d
  208:   2a36be5d 00000000 00000002 000186a4   *6.]............
  224:   00000002 00000004 00000001 0000001c   ................
  240:   2a3b6cfa 00000004 69736673 00000000   *;l.....isfs....
  256:   00000000 00000001 00000000 00000000   ................
  272:   00000000 00000008 32302e32 32352e32   ........20.225.2
  288:   0000000d 70617373 77642e62 796e616d   ....passwd.bynam
  304:   65000000 80000060 2a36be5c 00000000   e......`*6.\....
  320:   00000002 000186a4 00000002 00000004   ................
  336:   00000001 0000001c 2a3b6cfa 00000004   ........*;l.....
  352:   69736673 00000000 00000000 00000001   isfs............
  368:   00000000 00000000 00000000 00000002   ................
  384:   32300000 0000000d 70617373 77642e62   20......passwd.b
  400:   796e616d 65000000 80000064 2a36be5b   yname......d*6.[
...
```

Figure 5: Part of the alert message from an NIS attack.

monitor; investigation showed that they were innocent.) Perhaps the cracker community has not yet achieved a sufficient level of sophistication, or perhaps the traps have not been around long enough (the packet suckers were first deployed in mid-December of 1991). The frequency of attacks seems to be linked to the academic calendar; we saw a considerable upsurge in early January, when students would be returning to their campuses (in the U.S., at least), and a drop-off as their workload presumably increased.

When we detect an intrusion, we send a casual note to the system administrator. Generally, it says something like "someone from your site did <x> yesterday, and while we don't care much, we thought you might like to know, since such probes often come from stolen accounts." Responses are mixed. Some administrators respond immediately, ask for all the details we can provide, and take immediate action to track down the party responsible. Others never answer us. Perhaps they do not care, perhaps they never check `postmaster`'s mailbox, or perhaps the intruder has detected and deleted the mail. That last would seem to be a plausible explanation; one would think that sites would care that their own machines had been compromised. Commercial sites generally react the most; academic sites the least. On at least three occasions, we have had to notify administrators at (U.S.) military sites; to our surprise, we never received any response at all. Copies of all alarm messages and all administrator notifications are kept on an optical disk; additionally, CERT sees these notes.

4 Where the Wild Things Are

Not surprisingly, most of the attacks we have seen come from universities, both in the U.S. and abroad.[4] The distribution is highly non-linear; a few sites account for a high percentage of the misbehavior we see. One should not conclude, though, that the attackers are actually at those sites; very often, we see evidence of connection-laundering. This may take place because of open terminal servers, which permit hop-on/hop-off access, or because of a liberal attitude towards guest accounts, or because their own machines have been penetrated. We have seen evidence for all three explanations. (One persistent offender also hosts a well-known source archive accessible via NFS. We wonder if there is a connection. We also wonder about the integrity of the code in the archive.)

Table 1 shows the frequency of probes during February and March of 1992. The "ARP checks" indicate an address space probe judged to be suspicious enough to log; the other entries are based on a count of the automated trap messages generated. The `ftp` and `tftp` entries are of particular interest, since they are rarely, if ever, innocent. Other incidents, i.e., the `whois` connections, a few of the `portmopper` traps, and the `SNMP` messages, turned out to be benign.

The essential fact, though, is that the Internet can be a dangerous place. Individuals attempted to grab our password file at a rate exceeding once every other day. Suspicious RPC requests, which are difficult to filter via external mechanisms, arrived at least weekly. Attempts to connect to non-existent bait machines occurred at least every two weeks. It is worth noting that during the "Berferd" incident[Che92], we attempted, without success, to lure the intruders to that machine, which actually existed at the time. Now, connection requests have become commonplace. We do not know if there are that many more crackers, or if they have simply gotten more sophisticated in their targeting.

[4]This section is based on data compiled by Bill Cheswick.

Table 1: Frequency of Attacks During February and March

Incident	Number
guest/demo/visitor logins	296
rlogins	62
ftp passwd fetches	27
nntp	16
portmopper	11
whois	10
snmp	9
x11	8
tftp	5
ARP checks	4
systat	2
nfs	2
Number of evil sites	95

5 Ethical Concerns

To some, our activities are of dubious ethical character. The claim has been made that the existence of some of our monitors amount to entrapment. We welcome — and share — their sensitivity to ethical issues, but not their conclusions. We are comfortable with what we are doing.

We do not regard it as at all wrong to monitor our own machine. It is, after all, *ours*; we have the right to control how it is used, and by whom. (More precisely, it is a company-owned machine, but we have been given the right and the responsibility to ensure that it is used in accordance with company guidelines.) Most other sites on the Internet feel the same way. We are not impressed by the argument that idle machine cycles are being wasted. Most individuals' needs for computing power can be met at a remarkably modest cost. Furthermore, given the current abysmal state of host security, we know of no other way to ensure that our gateway itself is not compromised.

Equally important, we are not attempting to prosecute anyone. Our goal is to understand what is happening, and to shoo away nuisances. The reaction from system administrators whom we have contacted has generally been quite positive. In most cases, we have been told that either the probe was innocent, in which case nothing is done, or that the attacker was in fact a known troublemaker. In that case, the very concept of entrapment does not apply, since by definition it is an inducement to commit a violation that the victim would not otherwise have been inclined to commit. In a few cases, a system administrator has learned, through our messages, that his or her system was itself compromised.

The most problematic monitor is that on the `guest` login. We have been told that its existence is itself a lure. We do not agree. Most attempts to use it are blind; the individual has no reason to believe that we provide such a service. Rather, we are simply one of many systems that is searched for open accounts. To be sure, such a search is likely to be futile; guest login accounts have become quite rare on the Internet, even on historically open systems. This is in marked contrast to the ARPANET of 15 years ago. The change was likely inevitable; the vastly-increased access to the Internet has also increased the number of users who do not share the same moral credo with respect to proper behavior. Few sites, if any, are willing to expose themselves to unknown individuals. Even sites well-known for championing the principles of universal access have been forced to close down, because of abuses by a few guests.

The area of counterintelligence raises other serious issues. What sorts of network connections to other sites are proper? We must be very careful here not to step over the line. Given that we log `finger` attempts, and trace back `rusers` calls, are we justified in using those protocols ourselves? What about the aforementioned `telnet` operations? On occasion, we have had mail to a site administrator bounce; we have had to resort to things like hand-entered `VRFY` commands on the SMTP port to determine where the mail should be sent. Is that proper?

To carry matters a step farther, the suggestion has been made that in the event of a successful attack in progress, we might be justified in penetrating the attacker's computers under the doctrine of "immediate pursuit". That is, it may be permissible to stage our own counterattack in order to stop an immediate and present danger to our own property. The legal status of such an action is quite murky, though analagous precedents do exist. Regardless, we have not carried out any such action, and we would be extremely reluctant to; if nothing else, we would prefer to adhere to a higher moral standard than might be strictly required by law.

We do not claim to know definitive answers to these ethical questions. Thus far, we are comfortable with what we have done. If nothing else, our actions are (a) harmless, and (b) undertaken *only* in response to a "first strike" from the other site. But we are willing to listen to arguments that we have gone too far.

6 Future Extensions

There are several interesting ways to extend the current set of monitors. The most important change would be to monitor all requests for TCP or UDP services, and not just a select few. Currently, the gateway machine is blind to such probes, but the TCP listener on a Plan 9 machine has picked up requests for some very unusual port numbers, as part of an apparent attack[Bel92]. The ideal way to implement this monitoring would be for the kernel to pass unwanted packets to a user-level daemon, rather than issuing its own rejections. That daemon could do what it wanted — fork a child process to handle the connection, issue a reject, log the incident, etc. Unfortunately, no such mechanism exists at present in the systems we use. We may perform the necessary kernel surgery some day.

Our packet suckers could gather much more information if they had more ability to respond. We do not wish to write custom code for every possible service; however, a simple script interpreter might be useful. For example, the `nntp` listener could emit the proper greetings, thereby eliciting further input that might show the real location of the presumed security hole.

Along the same lines, we need better facilities for interpeting RPC requests. The current analysis program contains a lot of messy code; it should be fairly easy to write a `printf`-style interpreter for the messages. A better reply creator would be useful; for that, though, we might be best off using the real RPC library, our concerns notwithstanding. It might be useful to beef up the `portmopper` to respond to `rpcinfo -p`; we have seen a few such queries, and our own simulated attack scenarios have relied on it.

The DNS server (`named`) needs to have logging added as well. While it is probably inadvisable to note every single request, zone transfers can and should be logged. In theory, very few sites have legitimate reasons for examining our zone data, but we have seen evidence that crackers are already doing so. Some sites, in fact, already restrict zone transfers, though dodging bugs is the usual reason given for such policies.

We would like to hear about the results of similar monitoring at other sites. Our experiences may be atypical, for a number of reasons. We are in the "`.com`" domain, our machine is listed in the official `hosts.txt` file, some people still think we are "the phone company", and we have published several papers describing our security arrangements. A small university machine might see a very different pattern of attacks. On the other hand, we have seen enough connections that were apparently laundered through small university machines that we advise against complacency. Others report similar phenomena; see, for example, [Ran92].

For serious investigations of cracker behavior, a dedicated sacrificial machine is probably a better idea than installing trap programs. As noted, we made such a machine available when trying to track Berferd, but it attracted little interest. Our new monitors show much more interest in it today than we saw then.

Despite all this, it is important to view security in its proper perspective. The purpose of our gateway machine is to pass messages, not to entice crackers. We do not want to spend more effort fighting them than is necessary.

7 Recommendations

> *It does not do to leave a live dragon out of your calculations, if you live near him. Dragons may not have much real use for all their wealth, but they know it to an ounce as a rule.*
>
> J.R.R. Tolkien, *The Hobbit*

It is, of course, no surprise to anyone that crackers are active on the Internet. What is surprising, we think, is the level of activity. We see at least one hostile action a week, plus several doorknob checks a day. *Furthermore, we know of most of these solely because of our monitoring programs. No standard host software we are aware of provides an adequate level of monitoring.* More precisely, if you never look out the window, you will never see any dragons. And you will never know if one has decided that your passwords are just the things to add to its treasure hoard underneath the Mountain. The Internet appears to be lousy with dragons.... (N.B. We must confess that we do not visualize these dragons as grandiose or magnificent. Tolkien, of course, sometimes refers to dragons as "worms".)

The most important thing that can and should be done is for vendors to add logging to network software. Much more information needs to be logged, at the option of the site administrator. It is useful to be able to log *all* incoming connections, with some precis of the parameters passed. These need not be as detailed as our traps, of course, but should contain the essential information. Naturally, success or failure should be indicated as well.

While much of the logging can and should be done in `inetd`, that is not sufficient. Other programs need to create network log entries as well. For example, `named` should note the source of all zone transfer requests. (Optionally, such requests should be denied if not from known secondary servers for the zone. Some reasons were presented above; others are discussed in [Bel89].) The `ftp` daemon, `login`, and anything else that does authentication should note any session that does not end in a successful login. (Truly paranoid machines should log every attempt to log in, successful or not. But caution is indicated; experience suggests that one is likely to collect passwords that way[GM84].)

We urge the creation of a standardized logging interface. Do not confuse this interface with the `syslog` daemon. The daemon is a mechanism for collecting entries, not for creating them. The messages we wish should be in a form suitable for manipulation by `grep`, `awk`, `join`, and other standard tools, and that will only happen if they are created by a single subroutine.

Standardized filtering mechanisms are also useful. Given the number of daemons that are useful internally, but are susceptible to attack from outside, many administrators wish to deny access to them to outsiders. Router-level filtering is insufficient, if for no other reason than that the routers may be run by different organizations. Some vendors support filtering in `inetd`; most do not.

Unless and until standard logging and filtering mechanisms are created, use of outboard programs is a useful stopgap. There are a number of programs available to do that. One lists them in `/etc/inetd.conf` instead of the actual server; they create the log message, filter based on origin address, and only then pass control to the actual server.

8 Conclusions

"Never laugh at live dragons, Bilbo you fool!" he said to himself.

J.R.R. Tolkien, *The Hobbit*

It is all well and good to decry computer security, and to preach the religion of open access. Unfortunately, there are an increasing number of people with access to the Internet who do not share the morality necessary to make such schemes work. One can assume that one is being attacked; the only questions are how, and how often. (Just who the attackers are is in some sense uninteresting; if one group passes on, another is sure to take its place.)

Our goal is to provide information to the community, and to the proper authorities, on just how the crackers are operating. Our specific methods are not for everyone, but our lessons — and our warnings — are.

9 Availability

At this time, neither the gateway code nor the various monitors are available outside of AT&T. That may change in the future. Then again, it may not.

10 Acknowledgements

Bill Cheswick and Diana D'Angelo implemented the first hacker traps on our gateway machine[Che92]. Bill also did a lot of work collecting and collating log file data for this paper. He and Dave Presotto designed our overall security architecture.

Testing the traps described here required machines from which to launch simulated attacks. A number of sites granted us access to their systems; we thank them.

References

[Bel89] Steven M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communications Review*, 19(2):32–48, April 1989.

[Bel92] Steven M. Bellovin. Packets found on an internet, 1992. In preparation.

[CFSD90] J.D. Case, M. Fedor, M.L. Schoffstall, and C. Davin. *Simple Network Management Protocol (SNMP)*, May 1990. RFC 1157.

[Che90] W.R. Cheswick. The design of a secure internet gateway. In *Proc. Summer USENIX Conference*, Anaheim, June 1990.

[Che92] W.R. Cheswick. An evening with Berferd, in which a cracker is lured, endured, and studied. In *Proc. Winter USENIX Conference*, San Francisco, January 1992.

[DG87] Jack J. Dongarra and Eric Grosse. Distribution of mathematical software via electronic mail. *Communications of the ACM*, 30:403–407, 1987.

[GM84] Fred T. Grampp and Robert H. Morris. Unix operating system security. *AT&T Bell Laboratories Technical Journal*, 63(8, Part 2):1649–1672, October 1984.

[HM91] Katie Hafner and John Markoff. *Cyberpunk : Outlaws and Hackers on the Computer Frontier*. Simon & Schuster, 1991.

[Joh85] Mike St. Johns. *Authentication Server*, January 1985. RFC 931.

[Moc87] P.V. Mockapetris. *Domain Names — Concepts and Facilities*, November 1987. RFC 1034.

[Pen] Jan-Simon Pendry. `Amd` — An automounter. Department of Computing, Imperial College, London.

[Plu82] D.C. Plummer. *Ethernet Address Resolution Protocol*, November 1982. RFC 826.

[PPTT90] Rob Pike, Dave Presotto, Ken Thompson, and Howard Trickey. Plan 9 from Bell Labs. In *Proceedings of the Summer 1990 UKUUG Conference*, pages 1–9, London, July 1990. UKUUG.

[Ran92] Marcus J. Ranum. A network firewall. In *Proc. World Conference on System Administration and Security*, Washington, D.C., July 1992.

[Sto88] C. Stoll. Stalking the wiley hacker. *Communications of the ACM*, 31(5):484, May 1988.

[Sto89] C. Stoll. *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*. Doubleday, 1989.

[Sun90] Sun Microsystems, Inc., Mountain View, CA. *Network Interfaces Programmer's Guide*, March 1990. SunOS 4.1.

[Tol65] J.R.R. Tolkien. *Lord of the Rings*. Ballantine Books, 1965.

[Tol66] J.R.R. Tolkien. *The Hobbit*. Ballantine Books, 1937, 1938, 1966.