Packets Found on an Internet

Steven M. Bellovin*

August 23, 1993

Abstract

As part of our security measures, we spend a fair amount of time and effort looking for things that might otherwise be ignored. Apart from assorted attempted penetrations, we have also discovered many examples of anomalous behavior. These range from excessive `ICMP` messages to nominally-local broadcast packets that have reached us from around the world.

1 Introduction

For security reasons, AT&T's connection to the Internet is via a pair of application gateways[Che90]. To maintain the security of the gateways, we monitor them for attempted intrusions[Che92]. Recently, we have also started looking for more inventive penetration attempts[Bel92b]. We have indeed found such behavior. While looking, though, we noticed a surprising amount of other anomalous behavior, packets that do not appear to indicate an attempted break-in, but are worthy of attention nevertheless.

We are currently running three types of broad-spectrum monitors. First, a workstation with an Ethernet controller in "promiscuous mode" looks for packets not destined for any legal machine. Second, we run "packet suckers" on a variety of potentially-interesting ports. Third, we have recently deployed an `ICMP`[Pos81] monitor; it logs most `ICMP` messages received by the machine. Each of these has detected odd behavior. Currently, we cannot detect attempts to connect to random `TCP` or `UDP` ports, though we are contemplating adding that ability.

2 Address Space Oddities

Our setup for monitoring address space probes is fairly awkward. The monitoring machine is located in a part of the Murray Hill complex far removed from the live Internet cable. Accordingly, the link we are using includes a bridge, which filters out some packets. (This may be just as well, as it reduces the load.) Furthermore, since the monitor is not armored the way `research.att.com` is, we cannot allow it to talk to the Internet. Accordingly, we had a wire cutter introduce itself to the transmit leads on the drop cable. But this created a problem for ARP entries[Plu82]: the router will not transmit the packets until it has a valid Ethernet[1] address, and the monitoring machine is too crippled to supply one. The next obvious choice is to have `research.att.com` answer; unfortunately, it has no "raw" driver that would let an application program field ARP requests. We resorted to populating its kernel's tables as best we could; unfortunately, these tables are not large enough to permit complete coverage. Our selections, though adequate to detect security incidents, will likely miss attempts to reach random addresses. In the future, we hope to use a Plan 9 machine[PPTT90] to act as our ARP agent.

*AT&T Bell Laboratories. `smb@ulysses.att.com`

[1]Ethernet is a registered trademark of Xerox Corporation.

Reprinted from *Computer Communications Review*, July 1993, Vol. 23, No. 3, pp. 26–31.

2.1 Anomalous Broadcasts

None of this was necessary to detect the strangest packets we have seen: those addressed to host `255.255.255.255`, the IP broadcast address. That in itself would be quite ordinary, were they locally generated. They were not. On at least three occasions, we have received broadcast name server packets from other companies; more recently, we received a series of broadcasts intended for an local application. The first instance was from another company connected to the same regional network as our gateway; the other three involved traversals of the NSFnet backbone on the way to us.

We cannot explain why such packets would reach us. If router bugs permit such things to happen, we should see more broadcast packets, and for a wider range of ports. But all four incidents involved several packets, over a period ranging from minutes to hours. In one case where we were able to contact the site's administrator, we were told that their primary name server had crashed about the time of the first burst from their site. No theories were propounded to explain another broadcast packet from them several hours later.

Some commercial routers can be configured to forward such packets, if destined for the name server or selected other services. But the paths followed, in at least two of the cases, would have involved other types of routers. Other suggested causes include too much blind reliance on default routes.

We are continuing to monitor our network for such packets. Our timestamps are synchronized to WWV, in case anyone else has any logs they wish to match against ours.

2.2 Non-Existent Machines

We also see attempts to connect to odd addresses on our network. Some of these are very clearly security events — when the connections requests are only to non-existent machines nevertheless listed in the Domain Name Server (DNS) database[Moc87], the word "random" does not apply. Similarly, systematic attempts to probe the entire network's address space are likely carried out with hostile intent. But discounting those, we still see packets we cannot easily explain, packets destined for random addresses of ours.

In at least one case, the cause was determined to be repeated corruption of the sender's DNS cache. Somehow, a particular machine repeatedly acquired a variety of different incorrect addresses for `research.att.com`. As of this writing, we do not know where these addresses are coming from. The frequency of change is high enough that we do not think it is random contamination from an incorrect database; they would seem to be generated locally. A backup machine, running the same hardware and software, has displayed the same symptoms.

We have also seen numerous `ftp` requests for our old gateway machine, which has not existed for at least three years. As best we can tell, there are old host tables being passed around, even to new sites. Most of these requests have come from non-U.S. sites, where the DNS seems to be used less. Given that, it would seem to be worthwhile to re-advertise the existence of the standard `hosts.txt` file. Yes, the DNS is much better, but even a static host table is better than not being able to communicate at all.

3 Strange Application Requests

Some strange behavior occurs at the application layer. For example, we have seen a number of requests to connect to inexplicable port numbers. We know of no standard `TCP` daemons that listen on ports `2`, `42`, `70`, or `525`. Nor are those ports listed in the latest *Assigned Numbers* RFC[RP90]. While these particular requests appeared to part of an apparent break-in attempt, it is unclear to us why attackers should bother probing unused ports. Conceivably, these are standard back doors deployed and used by the hacker community (and documented, no doubt, in their own RFCs — "Resources For Crackers"); if so, it is especially unfortunate that most systems cannot log attempts to connect to unused ports. We were lucky to notice these requests; the attacker tried to connect to `9net.att.com`, a Plan 9 machine, and its design philosophy made detection quite easy.

On a number of occasions, we have seen attempts to connect to our NNTP port[KL86]. Since we do not run NNTP, such requests are definitely out of line. As best we can tell, the usual motivation is a desire to read newgroups disallowed by local administrative policy. Other reasons include a desire to submit forged articles, and — in one instance — a purported desire to determine whether or not a news article had been passed on. Certainly, there may be security flaws in the standard NNTP daemon. We have no evidence for or against this proposition.

On several occasions, our RPC[Sun90, Sun88] monitors have detected attempts to send "`wall`" broadcast messages to our machine. On at least one occasion, the request came from a site in Germany. Investigation of the code for the `rwall` command showed that if an entry in the `netgroup` file was not a valid host name, it was presumed to be a wild card. This in turn caused the broadcast message to be sent to every machine listed in the host file. The combination of this property of the code, and the apparent persistence of host tables, can cause a mind-boggling number of messages to be sent.

3.1 Wild and Crazy SNMP Agents

The most amusing application-level oddity we have seen was an SNMP message[CFSD90] from a distant university. Investigation showed that this was a case of an overly-helpful network management system. Apparently, several such systems have automatic or semi-automatic topology discovery mechanisms. This is useful — creating a network map is hard work for any entity large enough to need a management system — but such features need to be controlled. Internet-wide broadcasts are distressing enough; the thought of implementing them by stepping through the entire address space is horrifying.

This was not an isolated incident. We described what happened in the RISKS Digest[Bel92a], and received several reports of similar incidents elsewhere. Indeed, we have had runaways bother us since then, including once from the control center of a regional networks.

4 ICMP Peculiarities

A recent glance at the output of the `netstat` command showed several peculiarities. We were seeing non-zero counters for "bad code fields" and for "routing redirects". The latter was especially strange, since we have only one router on that network. Given the oddities, and given the theoretical possibility of an attack via `ICMP Redirect`[Bel89] messages, we wrote a monitor to log all `ICMP` messages. As usual, we saw more than we were looking for.

The `Redirect` messages were a bit elusive; they only seemed to come from certain sites. We eventually trapped a burst of them. Apparently, a dial-up IP server of some sort will emit them, possibly if the remote end is not available. The messages said, in effect, "to reach host `X`, use `X` as the gateway". Such a message is clearly erroneous even if `Redirect`s were legal when sent from other than the first-hop router. Not only that, the connection information returned was erroneous, with constant (and incorrect) values given for the local and remote port numbers, and even the remote host number, i.e., the value `X` referred to above. Attempts to trace the route showed that the server was indeed confused; a routing loop appeared as well, though that may be an artifact of the `traceroute` program.

We eventually learned that the target address that caused the trouble is in reality the broadcast address for a subnet. This explains some of what we saw; a broadcast storm can certainly confuse routers. And why were we trying to send messages to a broadcast address? Because `our` DNS cache was corrupted; it listed 15 incorrect addresses (and 2 correct ones) for a very popular mail relay host.

Our `ICMP` monitor also detected the source of at least some of the "bad code field" messages. Some routers, including a few that appear to be part of the NSFnet backbone, emitted `Source Quench` messages with a non-zero code field. This appears to be an ancient bug that was part of early releases of 4.3`BSD`. Unfortunately, many popular `ICMP` implementations will ignore messages with invalid code fields, recent RFC's notwithstanding[Bra89]. Thus, at the precise time when a router is strapped for resource, it is sending useless `Source Quench` messages.

4.1 Firewall Routers

Many of the `Destination Unreachable` messages we received came from so-called "firewall routers". These are routers with very restrictive access control lists; their purpose is to protect hosts within an organization, much as our gateways do. Unfortunately, the precise configuration of such gateways can and does cause trouble.

We encountered problems with a number of these routers. Attempts to send mail to destinations beyond the firewall generated large flurries of `Host Unreachable` messages. Analysis showed that the problem stemmed from the desire to present a different face to the inside than to the outside. For example, DNS `NS` records pointed to both the internal servers, to which access was blocked, as well as to the permitted gateway machines. For whatever reasons, our resolver tended to make large numbers of queries to the internal DNS servers. The resolver did not see the `ICMP` rejections, and perceived the problem only as a timeout. Eventually, it would switch to the next server in the `NS` list; until then, retransmissions to the original server would generate new bounce messages. A similar situation existed in the `MX` records. A moderately-large number of gateways were shown; only the least-desirable ones, by the included metrics, were reachable from the outside. Thus, mail deliveries to this site were quite expensive; a long list of failures had to be endured before a successful connection was established. The problem was compounded by the apparent inability of our local `TCP` to process `Destination Unreachable` messages at this point; instead, the connection attempts had to time out, a lengthy process.

The root cause of these failures is not, strictly speaking, a protocol problem. Rather, there is an operational weakness in the existing name server implemenations. Clearly, the administrator did not want us to try to reach the blocked hosts. Ideally, the answers returned by their DNS servers should be filtered: outsiders should never receive `NS` or `MX` records naming such hosts. But there is no easy way to do this. What is needed is some sort of general filtering language for the name server, specifying communities of interest and what records they are allowed to see.

We are not claiming that such a mechanism is a security feature. Unless and until authentication is added to the DNS, the level of security it could provide is fairly low. Rather, we are looking for performance improvements, and for the elimination of these unneeded and unwanted packets aimed at inside hosts.

4.2 A DNS Virus?

As noted above, incorrect DNS information exists. It is not clear why this happens; that it does happen is indisputable. Worse yet, the incorrect information can spread. If a site that has a bad resource record is queried about it, the server will blithely return the erroneous information, thereby contaminating the cache of another site. We thus have something with characteristics akin to a virus: a mutant record that uses standard facilities to reproduce itself. It "wins" if it can infect a high-level server, thereby causing it to spread to almost anyone who tries to find out the correct address for the destination.

The word "mutant" may, in fact, be literally correct. One possible explanation for the origin of such records is undetected corruption of DNS data while in transit. This is not at all unlikely, especially since at least one major vendor ships machines with `UDP` checksum validation and generation disabled. Other causes include address and name changes to name servers. If not done carefully, at both the delegating site and the primary and secondary servers for the zone, records containing the union of both old and new information will be propagated. We have seen this failure mode in our internal network.

4.3 Rejected DNS Messages

Our `ICMP` detected a surprisingly large number of `Port Unreachable` messages, most of them to or from port 53, the DNS port. Such messages suggest that a DNS response has arrived after the querying process has terminated. Most likely, these indicate that more than one packet was sent out for a given query, with some responses delayed overly long in transit. About 65% of the `ICMP` messages were to and from processes on the local machine, indicating that our own queries were repeated. The rest indicate that the local DNS server sent back late responses to outside inquiries.

DNS queries are in some sense a parasitic load. There is generally no value to the information per se; rather, a DNS query generally indicates a desire to open up a useful connection for things like mail delivery. A locally-generated query means that our host wishes to send information; a remote query means that someone wants to send something to us. Accordingly, we compared the number of locally-generated rejections with the number of outgoing calls during the same period, and the number of remote rejections of our DNS responses with the number of incoming calls. The results were alarmingly high, on the order of 50-60%. That is, about half the connections made involved repeated DNS inquiries, with the repetition due to CPU or network load. Given that many calls can be made solely with reference to the DNS cache, it would seem that either resolver retransmit timers are set much too low, or that there is some unsuspected name server bug that causes unneeded transmissions.

The quantitative aspects of this analysis are somewhat suspect. There are two servers for our domain, and two gateways; we are only monitoring one of each. Conceivably, our server is seeing a disproportionate number of DNS queries compared with the number of inbound mail messages. But we did see the same results for outbound messages, when the confounding factor was absent. We plan to investigate this further, deploying appropriate monitors on both machines. We also plan to trap and analyze sequences of DNS queries, responses, and rejections, an investigation our current monitoring configuration does not permit.

5 Conclusions

To some, our observations can be summarized succinctly as "bugs happen". That certainly is not news. But dismissing our results so cavalierly misses the point. Yes, bugs happen. But bugs can be fixed — *if* they are detected. The Internet is, as a whole, working remarkably well. Huge software packages (i.e., `X11R5`) can be distributed electronically. Connections span the globe. But the very success of the Internet makes some bugs invisible.

Because of our monitoring, we are able to spot certain classes of misbehavior that are, in general, not seen. Unfortunately, unlike our security logging recommendations[Bel92b], many of the techniques discussed here are not practical elsewhere. Trying to analyze bogus IP destination addresses on a busy Ethernet cable does not work, for example. But the underlying problems they are symptomatic of have not thereby gone away. We therefore suggest that, difficulties notwithstanding, others make similar efforts to instrument at least portions of their networks. That is the only way some of these subtle (and not so subtle) problems will be detected and eliminated.

References

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

[Bel92a] Steven M. Bellovin. "Helpful" self-configuring programs. *RISKS Digest*, 13(25), March 5 1992.

[Bel92b] Steven M. Bellovin. There be dragons. In *Proc. UNIX Security Symposium III*, pages 1–16, Baltimore, September 1992.

[Bra89] R.T. Braden, ed. *Requirements for Internet hosts - communication layers.*, October 1989. RFC 1122.

[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.

[KL86] Brian Kantor and Phil Lapsley. *Network News Transfer Protocol*, February 1986. RFC 977.

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

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

[Pos81] Jon B. Postel. *Internet Control Message Protocol*, September 1981. RFC 792.

[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.

[RP90] Joyce K. Reynolds and Jon B. Postel. *Assigned numbers*, March 1990. RFC 1060.

[Sun88] Sun Microsystems, Inc. *RPC: Remote Procedure Call Protocol Specification Version 2*, June 1988. RFC 1057.

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