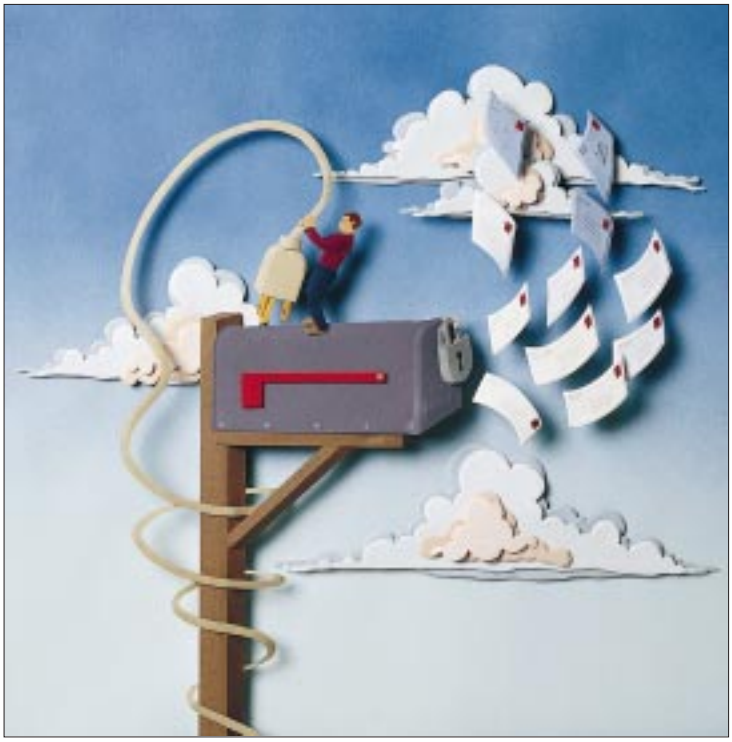
by Peter Collinson, **Hillside Systems**



DENISE ORTAKALES

Stop the Spam!

I'm becoming increasingly annoyed by spam, or to give it a somewhat less trademark-intensive name, unsolicited commercial email. I've been on the Net for about 10 years now, and my email address always seems to find its way into the clutches of the spam dispatchers. Over the weekend, my mailbox is deluged with tons of unwanted mail, all of which costs me time to scan and delete.

There's a huge danger that the world email network will simply become clogged with all this junk, that it will become less and less useful as a way of communicating with real people. One way to deal with the flood of messages is to simply ignore them, and I guess this seems the line of least resistance for many people. Ignoring the message is probably better than replying to the invitation to excise oneself from the list by returning a "remove" request. Many people are deeply suspicious of the efficacy of asking to be removed. After all, it's a good way for the spammer to see that the mail has arrived.

I suppose that what's most annoying is the inability to stop the mail from flowing in. I've spent some part of this year being essentially mail-bombed by a company in the Midwest who wanted to collect my debts for me. One message was sufficient for me to realize that I didn't need this debt collection service. At one point, I was getting five or six messages each Friday, Saturday and Sunday from this company. It's certain that the person doing this was extremely naive about the Net.

Anyway, this experience has caused me to be more proactive about spam. I've installed antispam measures on my machines and, for the harder cases, I've started to mail the companies who sell the IP service that is used by the spammers, asking them to stop the flow of mail. Many of the larger ISPs now have an `abuse` mail alias intended to allow people to complain, and when I do, they tell me that they have taken action to remove that customer from the Net. Before getting into that, let's look at what you can do at your site.

Most people running UNIX use Sendmail to send and receive mail. If your site uses a vanilla version from Sun Microsystems Inc., then you are well advised to upgrade to the latest version (which is Version 8.8.8 at this writing, see `http://www.sendmail.org`). You should install the newer version anyway because it is certainly more secure than the version you were supplied by your software vendor. Also, the more recent versions have hooks that allow for considerable checking of inbound mail, permitting you to defend yourself against a proportion of spam.

Incidentally, if you are worried about configuration, then Sendmail 8 has an automatic configuration system that works trivially. The new configuration system passes a three- or four-line setup file through the m4 macro processor to create your Sendmail configuration file, `sendmail.cf`.

Sendmail 8 should also be able to use your current configuration file, so from an engineering standpoint, you can drop

in the new binary /usr/lib/sendmail and check that your mail works. You can then create a new configuration file using m4 and check again.

Once you have converted to using the m4 version, life becomes simple. Eric Allman, the author and maintainer of Sendmail, sometimes changes the way the configuration actually works when he releases a new version, and you get these improvements for free by simply reprocessing your m4 configuration file when you install the new release. I've been tracking new releases of Sendmail for some time and have never had any problem. I've always been able to drop in the new release, and things have continued to work as advertised.

The new antispam hooks in Sendmail work by looking at the incoming SMTP connection, so the mail isn't received by your system; it's rejected before you see it. The hooks can be used in several ways. First, you can simply refuse to accept mail from a set of sites that can be entered by name or IP address. I now instantly add the domain name of sites that originate spam, and so I never get more than one piece of mail from that domain.

Second, most sites will relay mail automatically by default. I can very probably send mail to some system on the Net using your machine as a relay. Your mail system will accept the mail and resend it to its destination, and you are completely unaware that this is happening. Free relaying is a legacy of the original precommercial networking days, when mail was moved around the world from machine to machine.

Spammers make considerable use of this legacy. They want to hide the origin of the mail and prevent their mail systems from being clogged by people complaining about spam. They pick some poor unsuspecting machine and send it a single message with multiple recipients. The unsuspecting relay will fan the mail out all over the world, and probably some proportion of the recipients will think that the mail originated on that relay. Notice also that the spammer has sent a single message and has essentially got the relay machine to do all the work of spamming people.

On the whole, there is no reason why your machine should do any relaying now, except to relay mail from machines on your site to the outside world. If suddenly we all stopped relaying mail, then we would force the spammers to originate mail themselves, and this is much easier to stop.

Third, spammers usually forge mail headers to hide their origin. Some of these forgeries create a piece of mail that apparently comes from a site that doesn't exist. Many of the do-it-yourself spam packages select a random number and send mail purporting to come from somewhere like 27793739@8383826.com. The new Sendmail hooks can use the Domain Name Service and refuse to accept mail from addresses that cannot be found. It looks up the name in the DNS and rejects the mail if the name doesn't exist. The rejection is somewhat imperfect because the DNS can take time to obtain information, so the lookup can fail to return a machine name when it is needed. In the case of a lookup failure, Sendmail cannot reject the mail absolutely, so it sends a temporary failure message, and the sender (usually an unsuspecting relay) will keep retrying that message until it times out, usually a week or so later.

At the moment, I find that rejecting mail from addresses that don't exist is a huge antispam win, although the folks that supply me with IP service are perhaps less happy. The mail that I reject ends up on their machine, because they advertise themselves as my secondary mail forwarder. When I reject the mail, it ends up on their relay machine, which tries to send it to me. I continue to reject it, until it eventually times out.

If you want to take these actions with your mail system, or

Antispam Recipes

Eric Allman distributes his own m4 configuration file with the Sendmail release: Look for cf/cf/knecht.mc. The bottom half of this file, from LOCAL_CONFIG to the end, contains the antispam measures that he has installed on his machine.

On a vanilla Sun Microsystems Inc. Solaris system, you'll need to make some slight changes to the way that files are configured. Eric's default setup places control files in /etc, and you will want to put them into /etc/mail. Also, his default file uses the Berkeley db package to provide fast keyword lookup on data files. If, like me, you haven't bothered with this and are using the dbm database system instead, then you'll need to change the word hash in your copy of the section from the knecht.mc file to dbm. Eric was talking about supplying the db system as part of the Sendmail release (for other reasons), so this need may go away.

You'll now need to create two new control files: /etc/mail/sendmail.cR and /etc/mail/domaincheck. The sendmail.cR file contains a list of machines, one per line, naming the machines for which you are happy to provide relaying. If you change this list, then you'll need to restart the running sendmail on your machine.

Domains are accepted or rejected with the domaincheck file. The file contains lines, each holding a domain name and a rejection message:

```
.abuser.com   553 Domain restricted
abuser.com    553 Domain restricted
```

There should be a tab character after the domain name. This file is processed by the makemap program. If you are using the dbm package then

```
% makemap dbm domaincheck < domaincheck
```

will create the necessary dbm access files. You can add lines to this file without having to restart the running sendmail.

Now check that your mail works. You must ensure that relaying for internal machines is working properly and also that it fails when someone tries to relay mail through your machine from outside.–*pc*

only some of them, then it's not too difficult. I believe that you should stop relaying mail at the very least. There are some recipes for code to add to your m4 configuration file on the `http://www.sendmail.org` Web pages, and these pages also point elsewhere to other prescriptions. I've found that some of these recipes don't work too well, so I am using Eric Allman's solution, which is distributed with the Sendmail release (see "Antispam Recipes").

Interpreting Mail Headers

These solutions don't make spam go away, but they stop the easy cases from ending up on your machine. The hard cases are where the mail has been relayed and the header has been forged. As we have seen, if the forgery maps to a machine that doesn't exist, then we can ignore it. It's harder to discriminate against mail purporting to come from a real site.

As I said, I've taken to complaining, very politely, to the owners of the machines that originated the message. If you want to do this, then you need to know how to decode the headers that you can see on the message to determine the machine that was used to send it. Looking at the `From:` line isn't good enough; it's probably been forged anyway.

The first thing you need is the complete message with all its headers. Sadly, many mail programs suppress this information. On UNIX, you'll find that the information is very probably available, but you may need to edit your mailbox file to get it. There are so many different mail systems available, it's hard to talk generally about what you need to do to get the actual message that was received by your system. Beware that some systems strip headers when they save messages, so storing the mail may eliminate the information you need when making a complaint.

If you complain, then the ISP will need all the header information to prove to themselves that they are at fault and also to track down the villain. I've also taken to annotating the headers when I send complaint mail, because I find that the first-line support at some ISPs is unable to perform correct decoding.

On my machine, the very first line of the mail was added by my Sendmail program when it received the mail. It will look something like this:

```
Received: from relay.unsuspecting.com
    (relay.unsuspecting.com [279.254.254.254])
    by craggy.hillside.co.uk (8.8.7/8.8.7)
    with ESMTP id IAA06271 for <pc@hillside.co.uk>;
    Sun, 7 Dec 1997 08:11:21 GMT
```

Of course, the line is wrapped for printing. The parts in italics show that I've edited the original mail to protect the people involved in sending me this particular spam message. The line tells me that I got this mail from a machine calling itself `relay.unsuspecting.com`. But there's some confidence that this name is correct because the section in the round brackets is a decode of the IP address from where the mail arrived, and here the DNS reverse lookup succeeded, giving me a name that matched the name of the site. So, assuming I can trust my machine's DNS, I can determine the machine that sent me the mail.

The next two lines get more interesting:

```
From: someone@relay.unsuspecting.com
Received: from relay.unsuspecting.com
    (some.ipseller.net [299.53.52.8])
    by relay.unsuspecting.com (8.7.5/8.7.3) with
    SMTP id BAA28421;
    Sun, 7 Dec 1997 01:06:18 -0700 (MST)
```

The `From:` line is added by the relay mail system using the information that was carried in the SMTP call when it received the mail. The contents of this line are completely bogus and forged by the sender, who has configured their mail-sending program to lie–the `Received:` line underneath it tells us that. The DNS reverse lookup gave a different value for the real name of the IP address of the sending machine. When the sender originated the mail, they claimed to be the machine that was receiving it! Setting things up like this seems common practice with spammed mail. However, we can detect the forgery. The reverse lookup result tells us the IP address of the machine that was faking the mail.

Underneath this active `Received:` line is another one. This is also a fake and was added by the sender in an attempt to hide the originating machine. The body of the mail message and the actual spam follows all the header lines.

Incidentally, don't think that it's always the second `Received` line that is of interest. What's interesting is the point at which the spammer injected the mail into the network, and you need to look for inconsistency between the quoted address and the reverse lookup to see that.

Finding the ISP

We can usually deduce one truth: the IP address used by the sender when the mail was originated. Sometimes, as in the example above, we are also told the name that IP address resolves to. What next?

Well, now we need to find some contact address to send polite complaining mail. It's a good idea to do reverse lookup on the IP address yourself to check that the name to which it's supposed to resolve is correct. If it doesn't resolve, then there's a good chance the line is faked. I use the `dig` program to do reverse lookups on IP addresses. Giving the program the `-x` flag and the IP address will do this trivially. However, you may not have this on your machine because it's not standard.

Your machine will have the `nslookup` program (tucked away in `/usr/sbin` on Solaris) and you can use that to deduce the address, with some work. IP-to-domain name address mappings are stored in the DNS by reversing the IP numbers and then placing the resulting name in the DNS in the `in-addr.arpa` domain. So to look up the invented address above, you'd do this:

```
$ /usr/sbin/nslookup
> set type=PTR
> 8.52.53.299.in-addr.arpa
```

and hopefully you would be returned the relevant information.

The `set type` line tells `nslookup` to look for reverse addresses, which are called `PTR` records. Incidentally, you have to type a Control-D to get out of the `nslookup` program.

Another check that can be worthwhile for investigating IP addresses is to use the `traceroute` command. Again, this isn't readily available on your machine, but copies are floating around on the Net. I got my Solaris version from the Sun Freeware folks (see below).

The `traceroute` command is intended to help you determine the route your packets are taking through the Internet from your machine to some other system. Each step of the way will generate an IP address, which is looked up in the DNS, so you can see whose machines the packet has been through. The command can be invaluable for finding who owns addresses that are not registered in the DNS. These days, addresses are allocated in blocks, and if `traceroute` shows that the packet originated from a machine behind the gateway of an ISP, then it's a good bet your mail originated from that ISP's customer.

We've checked that the mail came from someone's system and now need to find the address of someone to whom we should send a complaint. The address in the example is `some.ipseller.net`, and it's a good bet that sending mail to `postmaster@ipseller.net` will work. Many ISPs have `support` addresses, and the larger ones are setting up `abuse` addresses. You can check that a domain name is likely to work for mail by looking for an MX record in the DNS:

```
$ /usr/sbin/nslookup
> set type=mx
> ipseller.net
```

which should give a response like this:

```
ipseller.net preference = 10,
  mail exchanger = postoffice.ipseller.net
```

If you don't get a response, try looking for any record for the site:

```
$ /usr/sbin/nslookup
> set type=any
> ipseller.net
```

and if the name exists you should be able to send mail and get a response.

If you cannot make up your mind about where to send the mail, then another valuable tool is the `whois` command. This tells you who owns a domain name for the main US domains. Typing

```
$ whois ipseller.net
```

will give you a bunch of useful information about the site you are trying to contact, including the email address of the person who is technically responsible for the domain. Other countries can be somewhat less open with this information. For European sites, consult the RIPE database (`http://www.ripe.net`).

This may all seem long-winded, but, most of the time, it's easy to find the name of the owner of the machine that was used to originate the spam. Send them polite mail–being rude doesn't help–and if they are responsible they will have an acceptable use policy that will allow them to remove the offender from the Net. I also tend to send mail to the owner of the relay machine, suggesting that they should stop relaying mail.

Some people use a program called `procmail` to eliminate spam. The `procmail` program started life as a way for users to sort their mail into identifiable folders, perhaps filtering mail from mailing lists into folders based on the mail header. It's now being used to filter out undesirable mail, looking for common strings in the mail that has been received and placing possible spam mail in some folder that contains junk. Because it looks at the whole mail message and not just the header, it can perhaps do a better job.

Paul Vixie has gone even further. He provides router and DNS information that forces IP traffic from known IP addresses into a black hole. The Real Time Blackhole lists networks that are known to be friendly to spammers, and by friendly, Paul means that these networks tolerate the sending of spam over their services. Various other networks are now making use of this black hole list to stop the activities of spammers.

Where to Find out More

A good place to start to look for antispam resources is `http://www.spam.abuse.net`. The Sendmail site is `http://www.sendmail.org`. The Sun Freeware project, where you can find binaries of `dig` and `traceroute` for Solaris, is `http://sunfreeware.com/`. A good place to start for `procmail` filtering is `http://www.best.com/~ariel/nospam/`.

All the names used in my examples are fictitious. The spam examples are taken from real mail. The domains `abuser.com`, `unsuspecting.com` and `someisp.net` were not registered at the time I wrote this. If they are registered by the time you read this, then the examples do not relate to mail processed by those machines.

Peter Collinson *runs his own UNIX consultancy, dedicated to earning enough money to allow him to pursue his own interests: doing whatever, whenever, wherever… He writes, teaches, consults and programs using Solaris running on a SPARCstation 2. Email:* `pc@cpg.com`.