Keep smiling, waste spammers' time

Peter N. M. Hansteen

When you are in the business of building the networks people need and the services they need to run on them, you may also be running a mail service. If you do, you will sooner or later need to deal with spam. This article is about how to waste spammers' time and have a good time while doing it.

Assembling the parts: To take part of the fun and useful things in this article, you need a system with PF, the OpenBSD packet filter. If you are reading this magazine you are likely to be running all important things on a BSD already, and all the fully open source BSDs by now include PF, developed by OpenBSD but also ported to the other BSDs. On OpenBSD, it is *the* packet filter, and if you are running FreeBSD, NetBSD or DragonFlyBSD it is likely to be within easy reach, either as a loadable kernel module or as a kernel compile-time option.

Getting started with PF is surprisingly easy. The official documentation such as the PF FAQ is very comprehensive, but you may be up and running faster if you buy *The Book of PF* [1] or do what about 30,000 others have done before you: Download or browse the free forerunner from *http://home.nuug.no/~peter/pf/*. Or do both, if you like.

Network design issues

A PF setup can be, and to my mind should be, quite unobtrusive. For the activities in this article it does not matter much where you run your PF filtering, as long as it is somewhere in the default path of your incoming SMTP traffic. A gateway with PF is usually an excellent choice, but if it suits your needs better, it is quite feasible to do the filtering needed for this article on the same host your SMTP server runs.

Enter spamd

OpenBSD's spamd, *the spam deferral daemon* (not to be confused with the program with the same name from the SpamAssassin content filtering system), first appeared in OpenBSD 3.3. The original spamd was a tarpitter with a very simple mission in life. Its spamd-setup program would take a list of known bad IP addresses, that is, the IP addresses of machines known to have sent spam recently, and load it into a table. The main spamd program would then have any smtp traffic from hosts in that table redirected to it, and spamd would answer those connections *s-l-o-w-l-y*, by default one byte per second.

A minimal PF config

As man spamd will tell you, the bare minimum to get spamd running in a useful mode on systems with PF version 4.1 or later is:

```
    table <spamd-white> persist
    no rdr inet proto tcp from
<spamd-white> to any port smtp
   rdr pass inet proto tcp from any to
any port smtp -> 127.0.0.1 port spamd
```

This means, essentially, that any smtp traffic from hosts that are not already in the table spamd-white will be redirected to localhost, port spamd, where you have set up the spam deferral daemon spamd to listen for connections. Enabling spamd, on the other hand, is as easy as adding `spamd_flags=""` to your `/etc/rc.conf.local` if you run OpenBSD or `/etc/rc.conf` if you run FreeBSD [2], and starting it with:

```
$ sudo /usr/libexec/spamd
```

or if you are on FreeBSD,

```
$ sudo /usr/local/libexec/spamd
```

It is also worth noting that if you add the `-d` for Debug flag to your spamd flags, spamd will generate slightly more log information, of the type shown in the log excerpts later in this article.

While earlier versions of spamd required a slightly different set of redirection rules and ran in blacklists-only mode by default, spamd from OpenBSD 4.1 onwards runs in greylisting mode by default. Let's have a look at what greylisting means and how it differs from other spam detection techniques before we exlore the finer points of spamd configuration.

Content versus behavior: Greylisting

When the email spam deluge started happening during the late 1990s and early 2000s, observers were quick to note that the messages in at least some cases could be fairly easily classified by looking for certain keywords, and the bulk of the rest fit well in familiar patterns.

Various kinds of content filtering have stayed popular and are the mainstays of almost all proprietary and open source antispam products. Over the years the products have develped from fairly crude substring match mechanisms into multi-level rule based systems that incorporate a number of sophisticated statistical methods. Generally the products are extensively customizeable and some even claim the ability to *learn* based on the users' preferences.

Those sophisticated and even beautiful algorithms do have a downside, however: For each new trick a spam producer chooses to implement, the content filtering becomes incrementally more complex and computationally expensive.

In sharp contrast to the content filtering, which is based on message content, *greylisting* is based on studying spam senders' behavior on the network level. The 2003 paper [3] by Evan Harris noted that the vast majority of spam appeared to be sent by software specifically developed to send spam messages, and those systems typically operated in a *fire and forget* mode, only trying to deliver each message once. The delivery software on real mail servers, however, are proper SMTP implementations, and since the relevant RFCs state that you MUST retry delivery in case you encounter some classes of delivery errors, in almost all cases real mail servers will retry *after a reasonable amount of time*. *Spammers do not retry*. So if we set up our system to say essentially *My admin told me not to talk to strangers* – we should be getting rid of anything the sending end does not consider important enough to retry delivering. The practical implementation is to record for each incoming delivery attempt at least:

1) Sender's IP address
2) The From: address
3) The To: address
4) Time of first delivery attempt matching 1) through 3)
5) Time delivery of retry will be allowed
6) Time to live for the current entry

At the first attempt, the delivery is rejected with temporary error code, typically *451 temporary local problem, try again later*, and the data above is recorded. Any

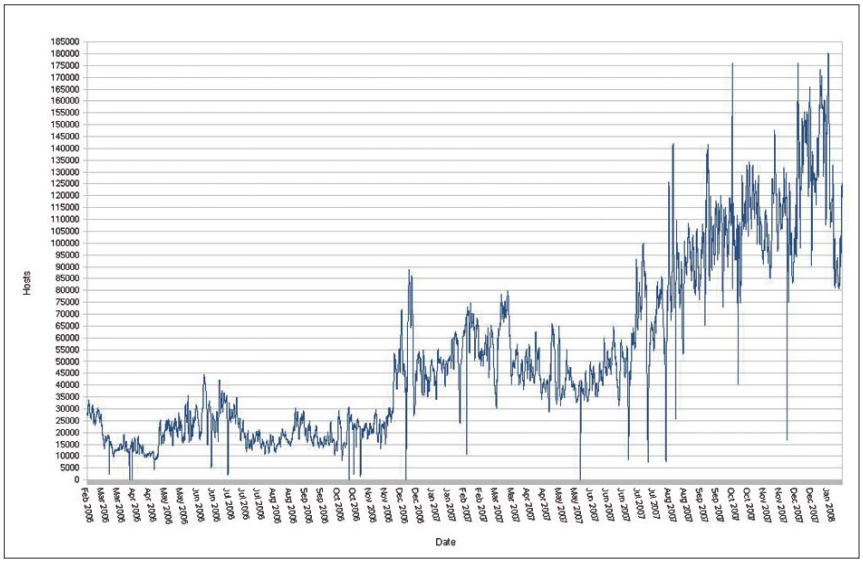

Figure 1. Number of hosts in the uatraps list

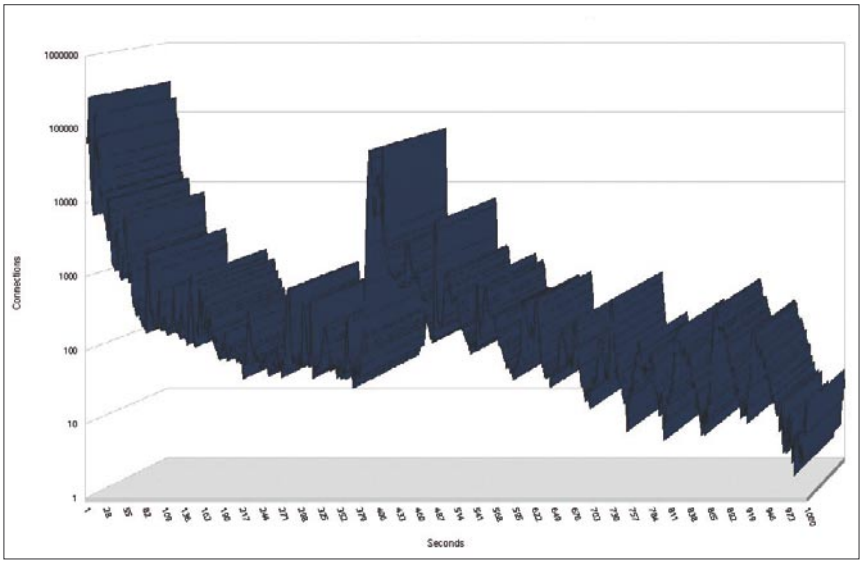

Figure 2. SMTP connections to smamd by connetion length

subsequent delivery attempts matching fields 1) through 3) that happen before the time specified in field 5) are essentially ignored, treated to the same temporary error. When a delivery matching fields 1) through 3) is attempted after the specified time, the IP address (or in some implementations, the whole subnet) is *whitelisted*, meaning that any subsequent deliveries from that IP address will be passed on to the mail service.

The first release of OpenBSD's spamd to support greylisting was OpenBSD 3.5. spamd's greylisting implementation operates only on individual IP addresses, and by default sets the minimum time before a delivery attempt passes to 25 minutes, the time to live for a greylist entry to 4 hours, while a whitelisted entry stays in the whitelist for 36 days after the delivery of the last message from that IP address. With a properly configured setup, machines that receive mail from your outgoing mail servers will automatically be whitelisted, too.

The great advantage to the greylisting approach is that mail sent from correctly configured mail servers will be let through. New correspondents will experience an initial delay for the first message to get through and their IP address is added to the whitelist. The initial delay will vary depending on a combination of the length of your minimum time before passing and the sender's retry interval. Regular correpondents will find that once they have cleared the initial delay, their IP addresses are kept in the whitelist as long as email contact is a regular affair.

And the technique is amazingly effective in removing spam. 80% to 95% or better reduction in the number of spam messages is frequently cited, but unfortunately only a few reports with actual numbers have been published. An often-cited report is Steve Williams' message on opensd-misc [4], where Steve describes how he helped a proprietary antispam device cope with an unexptected malware attack. He notes quite correctly that the blocked messages were handled without receiving the message body, so their apparently metered bandwidth use was reduced.

Even after more than four years, greylisting remains extremely effective. Implementing greylisting greatly reduces the load on your content filtering systems, but since messages sent by real mail servers will be let through, it will sooner or later also let a small number of unwanted messages through, and unfortunately it does not eliminate the need for content filtering altogether. Unfortunately you will still occasionally encounter some sites that do not play well with greylisting, see the references for tips on how to deal with those.

Do we need blacklists?

With greylisting taking care of most of the spam, is there still a place for blacklists? It's a fair question. The answer depends in a large part on how the blacklists you are considering are constructed and how much you trust the people who generate them and the methods they use.

The theory behind all good blacklists is that once an IP address has been confirmed as a source of spam, it is unlikely that there will be any valid mail send from that IP address in the foreseeable future.

With a bit of luck, by the time the spam sender gets around to trying to deliver spam to addresses in your domain, the spam sender will already be on the blacklist and will in turn treated to the *s-l-o-w* SMTP dialogue.

Knowing how a host makes it into a blacklist is important, but a clear policy for checking that the entries are valid and for removing entries is essential too. Once spam senders are detected, it is likely that their owners will do whatever it takes to stop the spam sending. Another reason to champion *aggressive maintenance* of blacklists is that it is likely that IP addresses are from time to time reassigned, and some ISPs do in fact not guarantee that a certain physical machine will be assigned the same IP address the next time it comes online.

Your spamd.conf file contains a few suggested blacklists. You should consider carefully which ones to use. Take the time you need to look up the web pages listed in the list descriptions in the *spamd.conf* file and then decide which lists fit your needs. If you decide to use one or more blacklists, edit your spamd.conf to include those and set up a cron job to let spamd-setup load updated blacklists at regular intervals.

The lists I consider the more interesting ones are the *nixspam* list, with a 4 day expiry and the *uatraps* list, with a 24-hour exiry. The nixspam list is maintained by ix.de, based on their logs of hosts that have verifiably sent spam to their mail servers. The uatraps list is worth looking into too, mainly because it is generated automatically by greytrapping.

Behavior based response: Greytrapping

Greytrapping is yet another useful technique that grew out of hands-on empirical study of spammer behavior, taken from the log data available at ordinary mail servers. You have probably seen spam messages offering lists of *millions of verified email addresses* available. However, verification goes only so far. You can get a reasonable idea of the quality of that verification if you take some time to actually browse mail server logs for failed deliveries to addresses in your domain. In most cases you will find a number of attempts at delivering to addresses that either have never existed or at least have no valid reason to receive mail.

The OpenBSD spamd developers saw this too. They also realized that what addresses are deliverable or not in your own domain is something you have complete control over, and they formulated the following rule to guide a new feature to be added to spamd:

"*if we have one or more addresses that we are quite sure will never receive valid email, we can safely assume that any mail sent to those addresses is spam*"

that feature was dubbed greytrapping, and was introduced in spamd in time for the OpenBSD 3.7 release. The way it works is, if a machine that is already greylisted tries to deliver mail to one of the addresses on the list of known bad email addresses, that machine's IP address is added to a special local blacklist called *spamd-greytrap*. The address stays in the spamd-greytrap list for 24 hours, and any SMTP traffic from hosts in that blacklist is treated to the tarpit for the same period. This is the way the uatraps list is generated. Bob Beck put a list of addresses he has referred to as *ghosts of usenet postings past* on his local greytrap list, and started exporting the IP addresses he collects automatically to a freely available blacklist. As far as I know Bob has never published the list of email addresses in his spamtrap list, but the machines at University of Alberta appear to be targeted by enough spammers to count. At the time this article was written, the uatraps list typically contained roughly 120,000 addresses, and the highest number of addresses I have seen reported by my spamd-setup was just over 180,000. See Figure 1.

By using a well maintained blacklist such as the uatraps list you are likely to add a few more percentage points to the amount of spam stopped before it reaches your content filtering or your users, and you can enjoy the thought of actively wasting spammers' time.

A typical log excerpt for a blacklisted host trying to deliver spam looks like what you see in Listing 1.

This particular spammer hung around at a rate of 1 byte per second for 403 seconds

(six minutes, forty-three seconds), going through the full dialogue all the way up to the DATA part before my spamd rejected the message back to the spammer's queue.

That is a fairly typical connection length for a blacklisted host. Statistics from my sites (see Figure 2) show that most connections to spamd last from 0 to 3 seconds, a few hang on for about 10 seconds, and the next peak is at around 400 seconds. Then there's a very limited number that hang around for anywhere from 30 minutes to several hours, but those are too rare to be statistically significant.

Interaction with a running spamd: spamdb

Your main interface to the contents of your spamd related data is the spamdb administration program. The command:

```
$ sudo spamdb
```

without any parameters will give you a complete listing of all entries in the database, whether WHITE, GREY or others. In addition, the program supports a number of different operations on entries in spamd's data, such as adding or deleting entries or changing their status in various ways. For example:

```
$ sudo spamdb -a 192.168.110.12
```

will add the host 192.168.110.12 to your spamd's whitelist or update its status to WHITE if there was an entry for that address in the database already. Conversely, the command:

```
$ sudo spamdb -d 192.168.110.12
```

will delete the entry for that IP address from the database. For greytrapping purposes, you can add or delete spamtrap email addresses by using a command such as:

```
$ sudo spamdb -T -a wkitp98zpu.fsf@da
tadok.no
```

to add that address to your list of spamtrap addresses. To remove the address, you substitute `-d` for the `-a`. The `-t` flag lets you add or delete entries for TRAPPED addresses manually.

Hitting back, poisoning their well: Summary of my field notes

Up util July 2007, I ran my spamd installations with greylisting, supplemented by hourly updates of the uatraps blacklist and a small local list of greytrapping addresses like the one in the previous section, which is obviously a descendant of a message-id, probably harvested from a news spool or from some unfortunate malware victim's mailbox. Then something happened that made me take a more active approach to my greytrapping.

My log summaries showed me an unusually high number of attempted deliveries to non-existent addresses in the domains I receive mail for. Looking a little closer at the actual logs showed spam *backscatter*: Somebody, somewhere had sent a large number of messages with made up addresses in one of our domains as the `From:` or `Reply-to:` addresses, and in those cases the `To:` address was not deliverable either, the bounce messages were sent back to our servers. The fact that they were generating bounces to the spam messages indicates that any copies of those messages directed at actually deliverable addresses in those domains would have been delivered to actual users' mailboxes, not too admirable in itself.

Another variety that showed up when I browsed the spamd logs was the type illustrated in Listing 2. Which could only mean that the administrators at that system had not yet learned that spammers no longer use their own `From:` addresses. Roughly at that time it struck me:

- Spammers, one or more groups, are generating numerous fake and nondeliverable addresses in our domains.
- Adding those generated addresses to our local list of spamtraps is mainly a matter of extracting them from our logs
- If we could make the spammers include those addresses in their to: addresses, too, it gets even easier to stop incoming spam and shift the spammers to the one-byte-at-a-time tarpit. Putting the trap addresses on a web page we link to from the affected domains' home pages will attract the address slurping robots sooner or later.
- Or the short version: Let's poison their well![5]

Over the following weeks and months I collected addresses from my logs and put them on the web page at *http://www.bsdly.net/~peter/traplist.shtml*.

After a while, I determined that harvesting the newly generated soon-to-be-spamtrap addresses directly from our greylist data was more efficient and easier to script than searching the mail server logs. Using spamdb, you can extract the current contents of the greylist with

```
# spamdb | grep GREY
```

which produces output in the format you see in Listing 3.

Listing 1. Spamd's SMTP dialogue with a blacklisted host

```
Jan 16 19:55:50 skapet spamd[27153]: 82.174.96.131: connected (3/2),
lists: uatraps
Jan 16 19:59:33 skapet spamd[27153]: (BLACK) 82.174.96.131: <bryonRoe@boxe
rdelasgargolas.com> -> <schurkoxektk@ehtrib.org>
Jan 16 20:01:17 skapet spamd[27153]: 82.174.96.131: From: "bryon Roe"
<bryonRoe@boxerdelasgargolas.com>
Jan 16 20:01:17 skapet spamd[27153]: 82.174.96.131: To:
schurkoxektk@ehtrib.org
Jan 16 20:01:17 skapet spamd[27153]: 82.174.96.131: Subject: vresdiam
Jan 16 20:02:33 skapet spamd[27153]: 82.174.96.131: disconnected after
403 seconds. lists: uatraps
```

Listing 2. Spamd encounters clulessness

```
Jul 13 14:36:50 delilah spamd[29851]: 212.154.213.228: Subject: Consi-
dered UNSOLICITED BULK EMAIL, apparently from you Jul 13 14:36:50 deli-
lah spamd[29851]: 212.154.213.228: From: "Content-filter at srv77.kit.kz"
<postmaster@srv77.kit.kz> Jul 13 14:36:50 delilah spamd[29851]:
212.154.213.228: To: <skulkedq58@datadok.no>
```

Listing 3. Spamdb greylist output format

```
GREY|96.225.75.144|Wireless_Broadband_Router|<aguhjwilgxj@bn.cam
com.it>|<bsdly@bsdly.net>|1198745212|1198774012|1198774012|1|0
GREY|206.65.163.8|outbound4.bluetie.com|<>|<leonard159@data
dok.no>|1198752854|1198781654|1198781654|3|0
GREY|217.26.49.144|mxin005.mail.hostpoint.ch|<>|<earle@data
dok.no>|1198753791|1198782591|1198782591|2|0
```

Where GREY is what you think it is, the IP address is the sending host's address, the third entry is what the sender identified as in the SMTP dialogue (HELO/EHLO), the fourth is the `From:` address, the fifth is the `To:` address. The next three are date values for first contact, when the status will change from GREY to WHITE and when the entry is set to expire, respectively. The final two fields are the number of times delivery has been blocked from that address and the number of conntections passed for the entry. For our purpose, extracting the made up `To:` addresses in our domains from backscatter bounces, it is usually most efficient to search for the `"<>"` indicating bounces, then print the fifth field. Or, expressed in grep and awk:

```
$ sudo spamdb | grep "<>" | awk -F\|
'{print $5}' |  tr -d '<>' | sort |
uniq
```

will give you a sorted list of unique intended bounce-to addresses, in a format ready to be fed to a corresponding script for feeding to spamd. The data in Listing 3 and the command line here would produce:

- *earle@datadok.no*
- *leonard159@datadok.no*

In some situations, the list will be a tad longer than in this illustration. This does not cover the cases where the spammers apparently assume that any mail with `From:` addresses in the local domain will go through, even when they come from elsewhere. Extracting the fourth column instead:

```
# spamdb | grep GREY | awk -F\|
'{print $4}' | grep mydomain.tld | tr
-d '<>' | sort | uniq
```

will give you a list of `From:` addresses in your own domain to weed out a few more bad ones from.

After a while, I started seeing very visible and measurable effects. At short intervals, we see spam runs targeting the addresses in the published list, working their way down in more or less alphabetical order. For example, in my field notes dated November 25, 2007, I noted earlier this month the address *capitalgain02@gmail.com* started appearing frequently enough that it caught my attention in my greylist dumps and log files.

The earliest contact as far as I can see was at Nov 10 14:30:57, trying to spam *wkzp 0jq0n6.fsf@datadok.no* from 193.252.22.241 (apparently a France Telecom customer). The last attempt seems to have been ten days later, at Nov 20 15:20:31, from the Swedish machine 217.10.96.36.

My logs show me that during that period 6531 attempts had been made to deliver mail from *capitalgain02@gmail.com* via bsdly.net, from 35 different IP addresses, to 131 different recipients in our domains. Those recipients included three deliverable addresses, mine or aliases I receive mail for. None of those attempts actually succeeded, of course.

It is also worth noting that even a decreipt the Pentium III 800MHz at the end of the unexciting DSL line to my house has been able to handle about 190 simultaneous connections from TRAPPED addresses without breaking into a sweat. For some odd reason, the number of simultaneous connection a the other sites I manage with better bandwidth have not been as high as the ones from my home gateway. During the months I have been running the trapping experiment, the number of spamtrap addresses in the published list has grown to more than 10,000 addresses. Oddly enough, a my greylist scans still show up a few more every few days. Meanwhile, my users report that spam in their mailboxes is essentially non-existent. On the other side of the fence, there are indications that it may have dawned on some of the spammers that generating random addresses in other people's domains might end up poisoning their own well, so they started introducing patterns to be able to weed out their own made up addresses from their lists. I take that as a confirmation that our harvesting and republishing efforts have been working rather well.

The method they use is to put some recognizable pattern into the addresses they generate. One such pattern is to take the victim domain name, prepend `"dw"` and append `"m"` to make up the local part and then append the domain, so starting from *sia.com* we get *dwsiam@sia.com*.

There is one other common variation on that theme, where the prepend string is `"lin"` and the append string is `"met"`, producing addresses like *linhrimet@hri.de*. Then again when they use that new, very recognizable, address to try to spam my spamtrap address *malseeinvmk@bsdly.net*, another set of recognition mechanisms are activated, and the sending machine is quietly added to my spamd-greytrap. And finally, there are clear indications that spammers use slightly defective relay checkers that tend to conclude that a properly configured spamd is an open relay, swelling my greylists temporarily. We already know that the spammers do not use `From:` addresses they actually receive mail for, and consequently they will never know that those messages were in fact never delivered. If you've read this far and you are still having fun, you can find other anecdotes I would have had a hard time believing myself a short time back in my field notes at *http://bsdly.blogspot.com/*. By the time the magazine has been printed and distributed, there might even be another few tall tales there.

References

[1] The Book of PF, by Peter N. M. Hansteen, No Starch Press December 2007, available in better bookshops or from the publisher at: *http://www.nostarch.com/pf.htm*.

[2] Note that on FreeBSD, spamd is a port, so you need to install that before proceeding. Also, on recent FreeBSDs, the `rc.conf` lines are `obspamd_enable="YES"` to enable spamd and `obspamd_flags=""` to set any further flags.

[3] The Next Step in the Spam Control War: Greylisting, by Evan Harris available at *http://greylisting.org/articles/whitepaper.shtml*.

[4] Available at *http://marc.info/?l=openbsd-misc&m=116136841831550&w=2*.

[5] Actually in the first discussions about this with my BLUG user group friends, we referred to this as *brnnpissing* in Norwegian, which translates as *urinating in their well*. The more detailed descriptions of the various steps in the process can be tracked via blog entries at *http://bsdly.blogspot.com*, starting with the entry dated Monday, July 9th, 2007, *http://bsdly.blogspot.com/2007/07/hey-spammer-heres-list-for-you.html*.

About the Author

Peter N. M. Hansteen is the author of The Book of PF (No Starch Press, December 2007). Peter has been tinkering with computers and networks since the mid-1980s, found the Freenixes in the early 1990s and is a frequent lecturer on PF and other OpenBSD and FreeBSD topics. He is a consultant, sysadmin and writer based in Bergen, Norway and occasionally blogs at *http://bsdly.blogspot.com/*.