**Spam Prevention**

Between when this sentence is written and when it is published, the Internet-wide spam problem will probably double in severity—or at least it will feel that way. There was a time, not so long ago, when one could run a mailing list without taking any spam-prevention measures at all. The occasional stray post would still show up, but infrequently enough to be only a low-level annoyance. That era is gone forever. Today, a mailing list that takes no spam prevention measures will quickly be submerged in junk emails, to the point of unusability. Spam prevention is mandatory.

We divide spam prevention into two categories: preventing spam posts from appearing on your mailing lists, and preventing your mailing list from being a source of new email addresses for spammers' harvesters. The former is more important, so we examine it first.

**Filtering posts**

There are three basic techniques for preventing spam posts, and most mailing list software offers all three. They are best used in tandem:

1. **Only auto-allow postings from list subscribers.**

This is effective as far as it goes, and also involves very little administrative overhead, since it's usually just a matter of changing a setting in the mailing list software's configuration. But note that posts which aren't automatically approved must not be simply discarded. Instead, they should be passed along for moderation, for two reasons. First, you want to allow non-subscribers to post. A person with a question or suggestion should not need to subscribe to a mailing list just to make a single post there. Second, even subscribers may sometimes post from an address other than the one by which they're subscribed. Email addresses are not a reliable method of identifying people, and shouldn't be treated as such.

2. **Filter posts through spam-filtering software.**

If the mailing list software makes it possible (most do), you can have posts filtered by spam-filtering software. Automatic spam-filtering is not perfect, and never will be, since there is a never-ending arms race between spammers and filter writers. However, it can greatly reduce the amount of spam that gets through to the moderation queue, and since the longer that queue is the more time humans must spend examining it, any amount of automated filtering is beneficial.

There is not space here for detailed instructions on setting up spam filters. You will have to consult your mailing list software's documentation for that (see the section called “Software” later in this chapter). List software often comes with some built-in spam prevention features, but you may want to add some third-party filters. I've had good experiences with these two: SpamAssassin (http://spamassassin.apache.org/) and SpamProbe (http://spamprobe.sourceforge.net/). This is not a comment on the many other open source spam filters out there, some of which are apparently also quite good. I just happen to have used those two myself and been satisfied with them.

3. **Moderation.**

For mails that aren't automatically allowed by virtue of being from a list subscriber, and which make it through the spam filtering software, if any, the last stage is *moderation*: the mail is routed to a special address, where a human examines it and confirms or rejects it.

Technical Infrastructure

Confirming a post takes one of two forms: you can accept the post just this once, or you can tell the list software to allow this and all future posts from the same sender. You almost always want to do the latter, in order to reduce the future moderation burden. Details on how to confirm vary from system to system, but it's usually a matter of replying to a special address with the command "accept" (meaning accept just this one post) or "allow" (allow this and future posts).

Rejecting is usually done by simply ignoring the moderation mail. If the list software never receives confirmation that something is a valid post, then it won't pass that post on to the list, so simply dropping the moderation mail achieves the desired effect. Sometimes you also have the option of responding with a "reject" or "deny" command, to automatically disapprove future mails from the same sender without even running them through moderation. There is rarely any point doing this, since moderation is mostly about spam prevention, and spammers tend not to send from the same address twice anyway.

Be sure to use moderation *only* for filtering out spams and clearly off-topic messages, such as when someone accidentally posts to the wrong mailing list. The moderation system will usually give you a way to respond directly to the sender, but don't use that method to answer questions that really belong on the mailing list itself, even if you know the answer off the top of your head. To do so would deprive the project's community of an accurate picture of what sorts of questions people are asking, and deprive them of a chance to answer questions themselves and/or see answers from others. Mailing list moderation is strictly about keeping the list free of junk and off-topic emails, nothing more.

**Address hiding in archives**

To prevent your mailing lists from being a source of addresses for spammers, a common technique is for

the archives to obscure people's email addresses, for example by replacing jrandom@somedomain.com

with

jrandom\_AT\_somedomain.com

or

jrandomNOSPAM@somedomain.com

or some similarly obvious (to a human) encoding. Since spam address harvesters often work by crawling through web pages—including your mailing list's online archives—and looking for sequences containing "@", encoding the addresses is a way of making people's email addresses invisible or useless to spammers. This does nothing to prevent spam from being sent to the mailing list itself, of course, but it does avoid increasing the amount of spam sent directly to list users' personal addresses.

Address hiding can be controversial. Some people like it a lot, and will be surprised if your archives don't do it automatically. Other people think it's too much of an inconvenience (because humans also have to translate the addresses back before using them). Sometimes people assert that it's ineffective, because a harvester could in theory compensate for any consistent encoding pattern. However, note that there is empirical evidence that address hiding *is* effective, see http://www.cdt.org/speech/spam/030319spamreport.shtml.

Ideally, the list management software would leave the choice up to each individual subscriber, either through a special yes/no header or a setting in that subscriber's list account preferences. However, I don't know of any software which offers per-subscriber or per-post choice in the matter, so for now the list

Technical Infrastructure

manager must make a decision for everyone (assuming the archiver offers the feature at all, which is not always the case). I lean very mildly toward turning address hiding on. Some people are very careful to avoid posting their email addresses on web pages or anywhere else a spam harvester might see it, and they would be disappointed to have all that care thrown away by a mailing list archive; meanwhile, the inconvenience address hiding imposes on archive users is very slight, since it's trivial to transform an obscured address back to a valid one if you need to reach the person. But keep in mind that, in the end, it's still an arms race: by the time you read this, harvesters might well have evolved to the point where they can recognize most common forms of hiding, and we'll have to think of something else.

**Identification and Header Management**

List subscribers often want to put mails from the list into a project-specific folder, separate from their other mail. Their mail reading software can do this automatically by examining the mail's *headers*. The headers are the fields at the top of the mail that indicate the sender, recipient, subject, date, and various other things about the message. Certain headers are well known and effectively mandatory:

From: ... To: ... Subject: ... Date: ...

Others are optional, though still quite standard. For example, emails are not strictly required to have the

Reply-to: sender@email.address.here

header, but most do, because it gives recipients a foolproof way to reach the author (it is especially useful when the author had to send from an address other than the one to which replies should be directed).

Some mail reading software offers an easy-to-use interface for filing mails based on patterns in the Subject header. This leads people to request that the mailing list add an automatic prefix to all Subjects, so they can set their readers to look for that prefix and automatically file the mails in the right folder. The idea is that the original author would write:

Subject: Making the 2.5 release.

but the mail would show up on the list looking like this:

Subject: [discuss@lists.example.org] Making the 2.5 release.

Although most list management software offers the option to do this, I strongly recommend against turning the option on. The problem it solves can easily be solved in much less obtrusive ways, and the cost of eating space in the Subject field is far too high. Experienced mailing list users typically scan the Subjects of the day's incoming list mail to decide what to read and/or respond to. Prepending the list's name to the Subject can push the right side of the Subject off the screen, rendering it invisible. This obscures information that people depend on to decide what mails to open, thus reducing the overall functionality of the mailing list for everyone.

Instead of munging the Subject header, teach your users to take advantage of the other standard headers, starting with the To header, which should say the mailing list's name:

Technical Infrastructure

To: <discuss@lists.example.org>

Any mail reader that can filter on Subject should be able to filter on To just as easily.

There are a few other optional-but-standard headers expected for mailing lists. Filtering on these is even more reliable than using the "To" or "Cc" headers; since these headers are added to each post by the mailing list management software itself, some users may be counting on their presence:

list-help: <mailto:discuss-help@lists.example.org>

list-unsubscribe: <mailto:discuss-unsubscribe@lists.example.org> list-post: <mailto:discuss@lists.example.org>

Delivered-To: mailing list discuss@lists.example.org

Mailing-List: contact discuss-help@lists.example.org; run by ezmlm

For the most part, they are self-explanatory. See http://www.nisto.com/listspec/list-manager-intro.html for more explanation, or if you need the really detailed, formal specification, see http://www.faqs.org/rfcs/rfc2369.html.

Notice how these headers imply that if you have a mailing list named "list", then you also have administrative addresses "list-help" and "list-unsubscribe" available. In addition to these, it is normal to have "list-subscribe", for joining, and "list-owner", for reaching the list administrators. Depending on the list management software you use, these and/or various other administrative addresses may be set up; the documentation will have details. Usually a complete explanation of all these special addresses is mailed to each new user as part of an automated "welcome mail" on subscribing. You yourself will probably get a copy of this welcome mail. If you don't, then ask someone else for a copy, so you know what your users are seeing when they join the list. Keep the copy handy so you can answer questions about the mailing list functions, or better yet, put it on a web page somewhere. That way when people lose their own copy of the instructions and post to ask "How do I unsubscribe from this list?", you can just hand them the URL.

Some mailing list software offers an option to append unsubscription instructions to the bottom of every post. If that option is available, turn it on. It causes only a couple of extra lines per message, in a harmless location, and it can save you a lot of time, by cutting down on the number of people who mail you—or worse, mail the list!—asking how to unsubscribe.

**The Great Reply-to Debate**

Earlier, in the section called “Avoid Private Discussions”, I stressed the importance of making sure discussions stay in public forums, and talked about how active measures are sometimes needed to prevent conversations from trailing off into private email threads; furthermore, this chapter is all about setting up project communications software to do as much of the work for you as possible. Therefore, if the mailing list management software offers a way to automatically cause discussions to stay on the list, you would think turning that feature on would be the obvious choice.

Well, not quite. There is such a feature, but it has some pretty severe disadvantages. The question of whether or not to use it is one of the hottest debates in mailing list management—admittedly, not a controversy that's likely to make the evening news in your city, but it can flare up from time to time in free software projects. Below, I will describe the feature, give the major arguments on both sides, and make the best recommendation I can.

The feature itself is very simple: the mailing list software can, if you wish, automatically set the Replyto header on every post to redirect replies to the mailing list. That is, no matter what the original sender puts in the Reply-to header (or even if they don't include one at all), by the time the list subscribers see the post, the header will contain the list address:

Technical Infrastructure

Reply-to: discuss@lists.example.org

On its face, this seems like a good thing. Because virtually all mail reading software pays attention to the Reply-to header, now when anyone responds to a post, their response will be automatically addressed to the entire list, not just to the sender of the message being responded to. Of course, the responder can still manually change where the message goes, but the important thing is that *by default* replies are directed to the list. It's a perfect example of using technology to encourage collaboration.

Unfortunately, there are some disadvantages. The first is known as the *Can't Find My Way Back Home* problem: sometimes the original sender will put their "real" email address in the Reply-to field, because for one reason or another they send email from a different address than where they receive it. People who always read and send from the same location don't have this problem, and may be surprised that it even exists. But for those who have unusual email configurations, or who cannot control how the From address on their mails looks (perhaps because they send from work and do not have any influence over the IT department), using Reply-to may be the only way they have to ensure that responses reach them. When such a person posts to a mailing list that he's not subscribed to, his setting of Reply-to becomes essential information. If the list software overwrites it, he may never see the responses to his post.

The second disadvantage has to do with expectations, and in my opinion is the most powerful argument against Reply-to munging. Most experienced mail users are accustomed to two basic methods of replying: *reply-to-all* and *reply-to-author*. All modern mail reading software has separate keys for these two actions. Users know that to reply to everyone (that is, including the list), they should choose reply-to-all, and to reply privately to the author, they should choose reply-to-author. Although you want to encourage people to reply to the list whenever possible, there are certainly circumstances where a private reply is the responder's prerogative—for example, they may want to say something confidential to the author of the original message, something that would be inappropriate on the public list.

Now consider what happens when the list has overridden the original sender's Reply-to. The responder hits the reply-to-author key, expecting to send a private message back to the original author. Because that's the expected behavior, he may not bother to look carefully at the recipient address in the new message. He composes his private, confidential message, one which perhaps says embarrassing things about someone on the list, and hits the send key. Unexpectedly, a few minutes later his message appears *on the mailing list!* True, in theory he should have looked carefully at the recipient field, and should not have assumed anything about the Reply-to header. But authors almost always set Reply-to to their own personal address (or rather, their mail software sets it for them), and many longtime email users have come to expect that. In fact, when a person deliberately sets Reply-to to some other address, such as the list, he usually makes a point of mentioning this in the body of the message, so people won't be surprised at what happens when they reply.

Because of the possibly severe consequences of this unexpected behavior, my own preference is to configure list management software to never touch the Reply-to header. This is one instance where using technology to encourage collaboration has, it seems to me, potentially dangerous side-effects. However, there are also some powerful arguments on the other side of this debate. Whichever way you choose, you will occasionally get people posting to your list asking why you didn't choose the other way. Since this is not something you ever want as the main topic of discussion on your list, it might be good to have a canned response ready, of the sort that's more likely to stop discussion than encourage it. Make sure you do *not* insist that your decision, whichever it is, is obviously the only right and sensible one (even if you think that's the case). Instead, point out that this is a very old debate, there are good arguments on both sides, no choice is going to satisfy all users, and therefore you just made the best decision you could. Politely ask that the subject not be revisited unless someone has something genuinely new to say, then stay out of the thread and hope it dies a natural death.

Someone may suggest a vote to choose one way or the other. You can do that if you want, but I personally do not feel that counting heads is a satisfactory solution in this case. The penalty for someone who is surprised by the behavior is so huge (accidentally sending a private mail to a public list), and the inconvenience for everyone else is fairly slight (occasionally having to remind someone to respond to the whole list instead of just to you), that it's not clear that the majority, even though they are the majority,

Technical Infrastructure

2 Shortly after this book appeared, Michael Bernstein [http://www.michaelbernstein.com/] wrote me to say: "There are other email clients that implement a reply-to-list function besides Mutt. For example, Evolution has this function as a keyboard shortcut, but not a button (Ctrl+L)."

3 Since I wrote that, I've learned that there is at least one list management system that offers this feature: Siesta [http://siesta.unixbeard.net/]. See also this article about it: http://www.perl.com/pub/a/2004/02/05/siesta.html

should be able to put the minority at such risk.

I have not addressed all aspects of this issue here, just the ones that seemed of overriding importance. For a full discussion, see these two canonical documents, which are the ones people always cite when they're having this debate:

• **Leave Reply-to alone**, *by Chip Rosenthal*

http://www.unicom.com/pw/reply-to-harmful.html

• **Set Reply-to to list**, *by Simon Hill*

http://www.metasystema.net/essays/reply-to.mhtml

Despite the mild preference indicated above, I do not feel there is a "right" answer to this question, and happily participate in many lists that *do* set Reply-to. The most important thing you can do is settle on one way or the other early, and try not to get entangled in debates about it after that.

**Two fantasies**

Someday, someone will get the bright idea to implement a *reply-to-list* key in a mail reader. It would use some of the custom list headers mentioned earlier to figure out the address of the mailing list, and then address the reply directly to the list only, leaving off any other recipient addresses, since most are probably subscribed to the list anyway. Eventually, other mail readers will pick up the feature, and this whole debate will go away. (Actually, the Mutt [http://www.mutt.org/] mail reader does offer this feature.2)

An even better solution would be for Reply-to munging to be a per-subscriber preference. Those who want the list to set Reply-to munged (either on others' posts or on their own posts) could ask for that, and those who don't would ask for Reply-to to be left alone. However, I don't know of any list management software that offers this on a per-subscriber basis. For now, we seem to be stuck with a global setting.3

**Archiving**

The technical details of setting up mailing list archiving are specific to the software that's running the list, and are beyond the scope of this book. When choosing or configuring an archiver, consider these qualities:

Prompt updating

People will often want to refer to an archived post made within the last hour or two. If possible, the archiver should archive each post instantaneously, so that by the time a post appears on the mailing list, it's already present in the archives. If that option isn't available, then at least try to set the archiver to update itself every hour or so. (By default, some archivers run their update processes once per night, but in practice that's far too much lag time for an active mailing list.)

Referential stability

Once a message is archived at a particular URL, it should remain accessible at that exact same URL forever, or as close to forever as possible. Even if the archives are rebuilt, restored from backup, or otherwise fixed, any URLs that have already been made publicly available should remain the same. Stable references make it possible for Internet search engines to index the archives, which is a major boon to users looking for answers. Stable references are also important because mailing list posts

Technical Infrastructure

and threads are often linked to from the bug tracker (see the section called “Bug Tracker”) later in this chapter or from other project documents.

Ideally, mailing list software would include a message's archive URL, or at least the message-specific portion of the URL, in a header when it distributes the message to recipients. That way people who have a copy of the message would be able to know its archive location without having to actually visit the archives, which would be helpful because any operation that involves one's web browser is automatically time-consuming. Whether any mailing list software actually offers this feature, I don't know; unfortunately, the ones I have used do not. However, it's something to look for (or, if you write mailing list software, it's a feature to consider implementing, please).

Backups

It should be reasonably obvious how to back up the archives, and the restoration recipe should not be too difficult. In other words, don't treat your archiver as a black box. You (or someone in your project) should know where it's storing the messages, and how to regenerate the actual archive pages from the message store if it should ever become necessary. Those archives are precious data—a project that loses them loses a good part of its collective memory.

Thread support

It should be possible to go from any individual message to the *thread* (group of related messages) that that original message is part of. Each thread should have its own URL too, separate from the URLs of the individual messages in the thread.

Searchability

An archiver that doesn't support searching—on the bodies of messages, as well as on authors and subjects—is close to useless. Note that some archivers support searching by simply farming the work out to an external search engine such as Google [http://www.google.com/]. This is acceptable, but direct search support is usually more fine-tuned, because it allows the searcher to specify that the match must appear in a subject line versus the body, for example.

The above is just a technical checklist to help you evaluate and set up an archiver. Getting people to actually *use* the archiver to the project's advantage is discussed in later chapters, in particular the section called “Conspicuous Use of Archives”.

**Software**

Here are some open source tools for doing list management and archiving. If the site where you're hosting your project already has a default setup, then you may not ever have to decide on a tool at all. But if you must install one yourself, these are some possibilities. The ones I have actually used are Mailman, Ezmlm, MHonArc, and Hypermail, but that doesn't mean the others aren't good too (and of course, there are probably other tools out there that I just didn't happen to find, so don't take this as a complete list).

Mailing list management software:

• **Mailman** — http://www.list.org/

(Has built-in archiver, and hooks for plugging in external archivers.) • **SmartList** — http://www.procmail.org/

(Meant to be used with the Procmail mail processing system.) • **Ecartis** — http://www.ecartis.org/ • **ListProc** — http://listproc.sourceforge.net/ • **Ezmlm** — http://cr.yp.to/ezmlm.html

Technical Infrastructure

(Designed to work with the Qmail [http://cr.yp.to/qmail.html] mail delivery system.) • **Dada** — http://mojo.skazat.com/

(Despite the web site's bizarre attempts to hide the fact, this is free software, released under the GNU General Public License. It also has a built-in archiver.)

Mailing list archiving software:

• **MHonArc** — http://www.mhonarc.org/ • **Hypermail** — http://www.hypermail.org/ • **Lurker** — http://sourceforge.net/projects/lurker/ • **Procmail** — http://www.procmail.org/

(Companion software to SmartList, this is a general mail processing system that can, apparently, be configured as an archiver.