DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033—Computer System Engineering Handout #25—April 7, 1999

Additional reading: an e-mail pseudonym server

In the schedule that appeared in handout zero, ***6.033 at a glance***, the reading for recitation 18, April 13, is listed as "TBD" (To Be Decided). We traditionally leave at least one slot open in this fashion, for a reading that is timely and, if possible, related to the subject of computer systems and society.

Attached is this year's reading, the paper

> David Mazières and M. Frans Kaashoek. The design, implementation and operation of an email pseudonym server. *Proceedings of the 5th ACM Conference on Computer and Communications Security*, 1998, pages 27–36.

Details on how to read it will appear in the usual weekly reading assignment. However, note well the following extra feature: on the same day that this paper is assigned and discussed in class, there will be a panel discussion in Kresge auditorium on this very subject, including Prof. Kaashoek and Scott Charney, of the United States Justice Department. Mr. Charney is expected to take a somewhat negative view of the service described in the paper. The panel discussion is part of the Laboratory for Computer Science 35th anniversary celebration, and Kresge is expected to be packed full (it is an invitation–only affair), but the discussion is to be carried live on the M.I.T. cable system. So locate a convenient access point for the cable broadcast and plan to watch at 4:00 p.m. Watching the panel is not a required assignment, but it is so timely and pertinent that we highly recommend it.

The Design, Implementation and Operation of an Email Pseudonym Server

David Mazières and M. Frans Kaashoek
MIT Laboratory for Computer Science
545 Technology Square, Cambridge MA 02139

Abstract

Attacks on servers that provide anonymity generally fall into two categories: attempts to expose anonymous users and attempts to silence them. Much existing work concentrates on withstanding the former, but the threat of the latter is equally real. One particularly effective attack against anonymous servers is to abuse them and stir up enough trouble that they must shut down.

This paper describes the design, implementation, and operation of nym.alias.net, a server providing untraceable email aliases. We enumerate many kinds of abuse the system has weathered during two years of operation, and explain the measures we enacted in response. From our experiences, we distill several principles by which one can protect anonymous servers from similar attacks.

1 Introduction

Anonymous on-line speech serves many purposes ranging from fighting oppressive government censorship to giving university professors feedback on teaching. Of course, the availability of anonymous speech also leads to many forms of abuse, including harassment, mail bombing and even bulk emailing. Servers providing anonymity are particularly vulnerable to flooding and denial-of-service attacks. Concerns for the privacy of legitimate users make it impractical to keep usage logs. Even with logs, the very design of an anonymous service generally makes it difficult to track down attackers. Worse yet, attempts to block problematic messages with manually-tuned filters can easily evolve into censorship—people unhappy with anonymous users will purposefully abuse a server if by doing so they can get legitimate messages filtered. Nonetheless, careful design can make a large difference in how well an anonymous server resists abuse.

This paper describes our experience in designing, implementing, and operating nym.alias.net, an email pseudonym server. Nym.alias.net allows anyone to create an email alias without revealing his identity. Such an alias, called a *nym* (short for pseudonym), appears as an ordinary email address to the rest of the world.

From *Proceedings of the 5th ACM Conference on Computer and Communications Security*, 1998.

Anonymous services get used for more and less popular reasons. Protecting unpopular speech is one of the fundamental purposes of anonymity. However, certain types of use can either force an anonymous server to shut down or else destroy its utility to other people. We classify such use as *abuse*. Our experience with nym.alias.net shows that controlling abuse is as important as protecting the identities of anonymous users. Both considerations should play a central role in the design of an anonymous server. Moreover, since people invent very creative forms of abuse, one must actually deploy an anonymous server to measure its viability.

This paper tackles the question of how to build anonymous servers that can survive in the real world. Our ideas apply equally to systems based on more advanced theoretical work (such as [1, 3, 6, 10, 12]), but such systems would likely not work with off-the-shelf software. Consequently, they would draw fewer users and fewer attacks of the kind we are concerned with studying.

1.1 History and usage

Nym.alias.net began operation in June 1996. To facilitate use of the system, we soon contributed support for it to the *premail* package, which provides encryption for popular Unix mail readers. Since then, others have built several DOS and Windows programs for managing nyms. Two other sites currently run our server software.

The anonymity of our users and the lack of mail logging make it impossible to know exactly how heavily the system is used. However, the number of active accounts has remained between 2,000 and 3,000 over the past 18 months. Statistics from the back end of our server suggest that the nyms on the system receive over 1,000 email messages per day. From the size of a replay cache the system keeps, we estimate that users send over 500 messages per day from nym addresses. Finally, Usenet search engines reveal many news postings from nym.alias.net addresses covering a large number topics.

We sent a survey to users of nym.alias.net asking them why they use the service. The survey encouraged people to answer as frankly as possible, and to reply anonymously. We received over 200 replies listing a wide range of uses. The reasons can broadly be categorized in order of decreasing need for privacy:

- In countries with oppressive governments, people use nyms to make public political statements, to hide the identities of their correspondents, and to encrypt the contents private mail (particularly when exchanging mail with people who don't use encryption).

- In more tolerant political environments, many people use nyms for purposes that might otherwise lead to embarrassment, harassment, or even loss of their jobs. These include discussing alcoholism, depression, and being a sexual minority, as well as meeting people through personal ads. A few people said they had used nyms to blow the whistle on illegal activities. Others express radical political views through nyms, while still others use them to fight harmful cults. Finally, a small minority of respondents admitted to using nyms for more legally marginal purposes, including discussing marijuana cultivation, publishing programs to exploit security holes, virus development, and software piracy.

- In companies that monitor email, some people report using nym addresses to encrypt all mail they receive before it enters the company. Nyms also keep the addresses of correspondents out of system log files. Some people similarly use nyms because they distrust their Internet service providers.

- Some people worry that seemingly innocuous Usenet posts will have unforeseen future ramifications. One response described a job interview at which the candidate's Usenet posting became a topic of discussion.

- Some nym users simply want their statements to be judged on their own merit. These people fear their reputations would bias readers for or against any messages bearing their real addresses.

- Finally, a surprisingly high number of people just use nym.alias.net for a free email address—either to avoid junk email by changing addresses frequently, or to keep a permanent address when switching Internet service providers. The fact that people use nyms without needing the privacy speaks well for the reliability of the system.

Of course, though we asked users to be frank, those who abuse the service had little incentive to answer our survey.

1.2 Design goals

We designed nym.alias.net with three goals: to build a real system that would see real use (and abuse) by people outside of computer science research, to protect the secrecy of users' identities in the face of compromised servers, and to provide a robust email service people can rely on.

We achieved the first goal, attracting users, by building on existing infrastructure. To use nym.alias.net, one only needs a copy of PGP [15], the most widely used encryption program. Moreover, nym.alias.net exploits a preexisting network of anonymous remailers—servers that strip identifying information from mail and forward it, after optionally decrypting, encrypting, or delaying it. While a clean-slate pseudonym server design would have permitted greater security at an equal level of deployment, it would also have enjoyed considerably less acceptance.

To achieve the second goal, preventing compromised servers from disclosing users' identities, nym.alias.net uses the anonymous remailer network as a *mix-net* [4]: It forwards mail received for a nym to its final destination through a series of independently operated remailers. Only by compromising multiple remailers can one uncover the full path taken by such a message. Thus, even the administrators of the nym server have no way to expose the identity of someone making proper use of the system.

The third goal, reliability, we achieve through solid software and redundancy. The nym.alias.net software itself is carefully written and does not lose mail—a claim substantiated by people using the server for permanent email addresses. One of the authors of this paper actually uses a nym as his primary email address for all correspondence about the server. Reliability does become more of a challenge when messages travel through many remailers. However, as described later in Section 3.3, nym.alias.net can lessen this problem with redundant messages.

1.3 The rest of this paper

In the rest of this paper, we describe the nym.alias.net pseudonym server and few related services the machine provides. We then discuss several kinds of abuse nym.alias.net has weathered. In each case, we explain how the machine fared and what changes, if any, we made in response to the abuse. Finally, we classify the abuse of anonymous services into three general categories, and suggest principles by which one can develop solutions.

2 Related work

Our work on nym.alias.net was largely motivated by the problems of previous unpublished anonymous mail systems. A good summary of past and present systems (including nym.alias.net) can be found in [8].

The first email pseudonym system open to the public was anon.penet.fi. Penet kept a database linking real and pseudonymous email addresses. It replaced a user's real email address with her pseudonym in outgoing mail, and routed incoming mail received for a pseudonym back to the appropriate address.

Unfortunately, penet did not use encryption—all messages went over the network in cleartext and were vulnerable to eavesdropping. Moreover, by design, the operator of the service knew the identities of all users. Only one machine needed to be compromised to violate the privacy of every user on the system. Penet also severely restricted the size and number of messages any given user could send, and imposed a delay of several days on any pseudonymous communication. These properties protected the system from abuse at the cost of usefulness. Finally, penet automatically provided double-blind communication. This could potentially cause users to send pseudonymous email unknowingly (particularly to pseudonymous mailing list subscribers), and thus to reveal their identities through the context of a message not intended to be pseudonymous.

Penet shut down most of its operation when the operator faced the risk of having to turn the user database over to authorities. It later shut down completely when it became overloaded with unsolicited commercial mail.

Type-1 anonymous remailers, also called cypherpunk remailers, were developed to address many shortcomings of the penet system. Type-1 remailers have public keys with which incoming messages can be encrypted. A message can be sent through a *chain* of type-1 remailers, having been successively encrypted for each one. Each remailer in a chain knows only the identity of the previous remailer and the next. Type-1 remailers alone serve mostly for anonymous, rather than pseudonymous mail. However, they do allow messages to be sent to unknown destinations. As described later, nym.alias.net exploits this property to provide email addresses to users whose identities it does not know.

The alpha.c2.org pseudonym server provided untraceable pseudonyms through type-1 remailers, and was part of the inspiration for nym.alias.net. However, alpha was vulnerable to replay attacks, did not use public keys to identify pseudonyms, did not provide forward secrecy of messages received for pseudonyms, could not tolerate an unreliable type-1 remailer network, developed serious reliability problems of its own under high load, and finally was shut down for using too much CPU time.

Type-2 or mixmaster remailers [5] offer several improvements in security over type-1 remailers. These improvements in general make hop-by-hop traffic analysis considerably harder. They include fixed size messages, replay detection, and better reordering of messages at remailers. Type-2 remailers do not, however, allow replies to unknown destinations, and thus cannot be used to provide pseudonyms.

Experimental versions of the type-2 remailer have incorporated hash cash [2], a scheme that deals with service abuse; it allows providers of unmetered Internet services to charge for usage in burnt CPU time. Hash cash requires users of a service to find partial hash collisions under a cryptographic hash function—an expensive operation that can be efficiently verified. Hash cash has the potential to limit certain kinds abuse to free anonymous servers.

Babel [9], an anonymous remailer developed at IBM Zurich Research Laboratory, incorporates a number of features to foil traffic analysis. Unlike nym.alias.net, Babel provides a distributed architecture with no central server maintaining nyms; instead, each email message includes specially encoded instructions for how to respond through the remailer network. The disadvantage of this approach is that a person who receives such email must to understand how to use encryption software. Nym.alias.net has no such requirement. Accounts at our server behave like regular email accounts. Users responding to email from a nym account can do so using a standard mail reader. Our survey shows that many users consider this an important feature.

Recently, some systems have provided anonymity in areas other than email, including interactive network connections [14] and web browsing [13]. Anonymous web browsing should allow pseudonymous email though web-based email providers. We don't know how many people are using it for that purpose, or what kind of abuse, if any, these systems have suffered from.

An interesting question is whether users have the right to anonymity. The question is complex and its answer is likely to vary from country to country. In the Unites States, there is no law making services such as nym.alias.net illegal. In fact, there have been a number of court cases that link anonymous speech directly to freedom of speech, in particular for political anonymous speech. However, whether the U.S. constitution directly protects anonymous communication is an open legal question. This paper does not address this question; we point the reader to Froomkin [7] on the legal issues of anonymous on-line speech.

3 Pseudonym server

This section describes the workings of the pseudonym server. The nym.alias.net help file [11] gives more complete details of the system's operation, including down-to-the-byte descriptions of message formats. For those wanting even more detail, we have always made the system's source code freely available for use and inspection.

Nym.alias.net uses a type-1 remailer network similar to Chaum's *mix-nets* [4]. A *mix* is a computer that forwards batches of messages, using encryption to conceal the relationship between incoming and outgoing ones. Mixes can be cascaded so that multiple mixes must be compromised to expose the path of a message. While type-1 remailers do not offer the full security of mixes, they do permit the nym server to send mail to users without knowing their real email addresses.

3.1 Nymserv

The pseudonym server consists principally of the program *nymserv*, which is invoked by the system mail software (e.g., sendmail or qmail) whenever it must deliver mail to an address at nym.alias.net. Nymserv, in turn, remails messages addressed to nyms in such a way that they will eventually reach the owners of those nyms. A few reserved addresses cause special processing of incoming mail. For instance, to send email from a pseudonymous address, one sends it through `send@nym.alias.net`. Requests to create and delete nyms go to `config@nym.alias.net`. Of course, any mail sent to an unused address at nym.alias.net will bounce as usual.

Nymserv keeps three pieces of information on file for every nym: a public key, a reply block, and some configuration data. The public key authenticates messages from the owner of the nym. All mail sent from a nym address must be signed by that nym's private key, as must requests to delete a nym or modify its configuration settings. By default, nymserv also encrypts any mail sent to a nym with that nym's public key. This ensures the forward secrecy of remailed messages; someone who compromises the server and learns that a nym forwards mail to a news group still cannot recover the contents of previously received messages.

The reply block contains instructions for getting mail from the nym server to the owner of a nym. These instructions are successively encrypted for a series of type-1 remailers in such a way that each remailer can only see the identity of the next hop. The innermost encrypted instructions, visible only to the last remailer, contain the final destination of mail sent to a nym.

While people generally choose their real email addresses as a final destination, they can alternatively use broadcast messages pools such as the Usenet group alt.anonymous.messages. Sending mail to a newsgroup that propagates to so many machines makes it virtually impossible to track a user down from a reply block alone (though most news servers keep logs that will permit one confirm a guess about the identity of a nym).

Thus, one need never communicate directly with nym.alias.net to use a pseudonym. Digital signatures prove the authenticity of messages to the server, allowing them to come from anywhere. In particular, requests to create nyms and send mail from them usually arrive through a chain of anonymous remailers. Likewise, mail sent sent from the server to a user leaves through a chain of anonymous remailers. The nym server administrators have no easy way to find the real identity of someone using the service in this way.

3.2 Reply block details

Reply-blocks use type-1 remailers to conceal the destination of mail messages. A type-1 remailer message begins with a preamble specifying the email address of a next hop. This preamble can also contain a delay time, a symmetric encryption key, and mail headers like Subject and Newsgroups to paste into the remailed message. Type-1 remailers strip identifying headers from any mail they receive. Then, depending on the preamble, they can conventionally encrypt everything after a marker line, paste mail headers, and delay messages. Finally, they forward messages onto their next hops. Every type-1 remailer has a public key. The beginning of a type-1 remailer message or the entire message may be encrypted with the remailer's public key. This allows the nym server to construct valid type-1 messages by prepending an encrypted reply block to a mail message received for a nym. Symmetric encryption below the reply block makes it difficult for eavesdroppers to correlate incoming and outgoing messages at a remailer. PGP is used for both the public-key and symmetric key encryption.

Figure 1a shows the process of creating a reply block with two hops. The user encrypts her real email address, `usr@a.com`, and a symmetric key, "key1," with the public key of remailer `rem@b.edu`. She then prepends the address of that remailer and another key, "key2," to the resulting cyphertext and encrypts that with the public key of a second remailer, `rem@isp.nl`. Finally, she prepends `rem@isp.nl` and a third key, "key3," to the second cyphertext. In all cases she has specified a random delay of up to one hour.

Figure 1b shows the encryptions undergone by a message delivered to a nym with this reply block. Nymserv always starts by adding some explicit context to any message it receives, including the name of the pseudonym receiving the message (not necessarily obvious from the message itself), the date, a unique identifier, and a disclaimer. It then digitally signs the message with its own private key and encrypts the message with the nym's public key. It prepends the reply block to the resulting cyphertext, and feeds the result to a type-1 remailer running on the local machine. That remailer then super-encrypts the message with "key3," randomly delays it for up to an hour, and forwards it to `rem@isp.nl`. `rem@isp.nl` in turn super-encrypts the message with "key2," delays it, and forwards it to `rem@b.edu`, which likewise super-encrypts the message using "key1" and sends it on to the user.

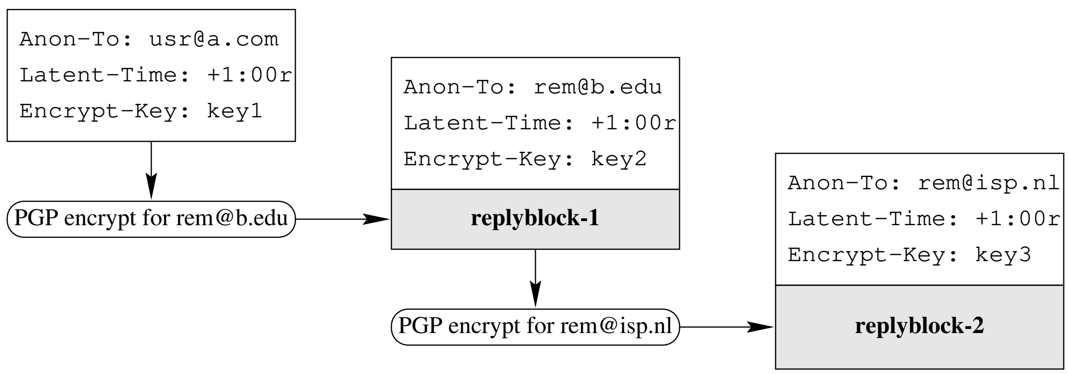
Figure 1c shows the actual data sent across the network when a nym with this reply block receives mail. One can immediately see that the security of the system is far from optimal: Identical reply block cyphertexts travel across the network each time a particular nym receives mail. Messages crossing the network have non-constant size. Nothing prevents message replays or reuse of inner reply block cyphertexts; an attacker can grab a reply block cyphertext off the network and reuse it to send either a huge message or a large number of small messages—facilitating hop-by-hop traffic analysis in either case. This was the price we paid to attract real users.

Nonetheless, the secrecy of nyms doesn't entirely depend on type-1 remailers. One can still achieve strong privacy through broadcast message pools. Thus, nym.alias.net does permit virtually untraceable nyms, albeit inefficiently and inconveniently. More importantly, most attacks on reply blocks, though theoretically possible, are beyond the means of the nym server operators. Even in cases where we might actually have wanted to trace a nym—for instance when a very distressed sounding teen-ager discussed suicide in a newsgroup—revealing the person's identity was never an option. Thus the weaknesses of type-1 remailers have probably had little effect on our experience of running the server.

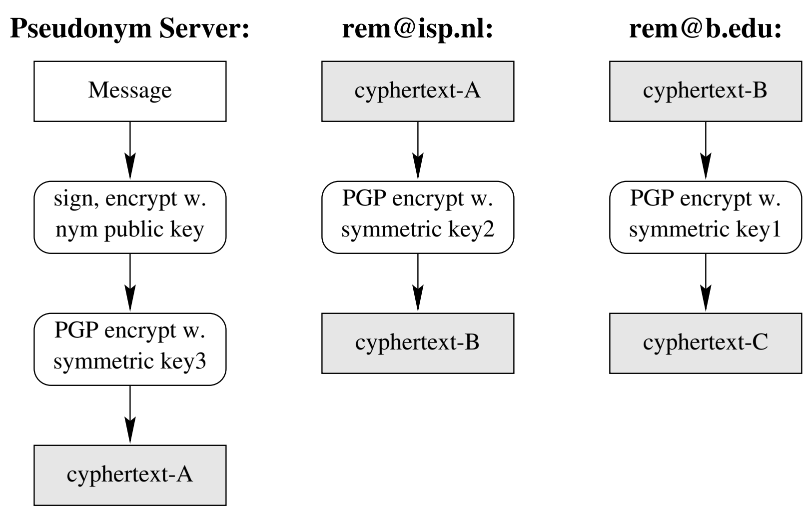
3.3 Reliability, replay and redundancy

Nym.alias.net's pseudonym server does not lose mail[1]. The machine has a good network connection and high uptime, and the nymserv software has proven robust. The same cannot, unfortunately, be said of all anonymous remailers. Remailers come and go, often with little warning. A large number of independently run remailers give users more options for remailer chains, but

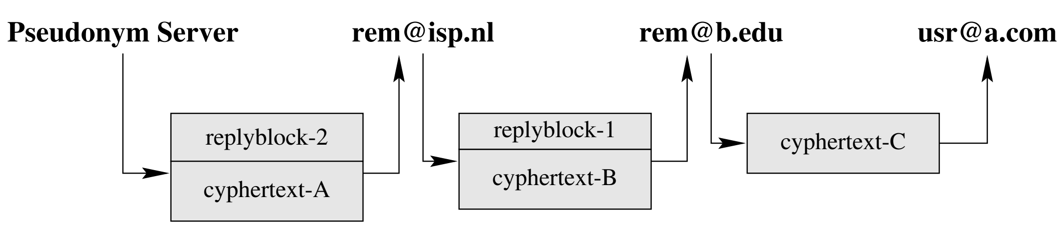
[1]We must mention a single, painful, and glaring exception to this statement. An Internic billing error for alias.net led to the disappearance of the entire domain for a period of several weeks. The authors have no affiliation with alias.net beyond having use of the nym.alias.net host name, and consequently could do nothing to hasten resolution of the problem. We nonetheless continue to believe that nym servers should in principle be highly reliable.



a. Steps performed by a user to construct a reply block with two hops



b. Encryptions performed on messages at each remailer



c. The actual data that traverses the network

Figure 1: Forwarding messages to anonymous users

not everyone willing to run a remailer can do so reliably. ISPs sometimes shut down customers' remailers when controversial usage surfaces or traffic levels get too high. "Disposable remailers" running on free email services like juno.com periodically exceed their mail quotas. Disks fail in cases where operators avoided backing up private keys. Machines crash when remailer operators have gone on vacation and no one else has access to the machine. In short, what's good for security may hurt reliability. Pseudonym servers should therefore tolerate an unreliable remailer network.

Two types of mail risk getting lost in the remailer network: messages from users to the nym server, and those from the nym server to users. Redundancy can address both risks. Nymserv keeps a replay cache to thwart certain attacks, but this cache additionally allows users to send duplicate copies of any message to the nym server. Nymserv also permits pseudonyms to have multiple reply blocks, which lets users receive several copies of mail to their nyms through distinct chains of remailers.

Attackers may try to replay old configuration messages or cause duplicate copies of outgoing mail. Nymserv consequently keeps a replay cache of all such messages (i.e. anything sent to `config@nym.alias.net` or `send@nym.alias.net`). Both types of message carry PGP signatures. Nymserv caches the MD5 hash of these signatures to detect replay. It will process the same message twice only if the user has signed it twice. Fortunately, the replay cache need not grow without bounds. PGP embeds a date and time in every signature. Nymserv discards incoming messages with signatures older than a week and those dated too far in the future. It can therefore delete any MD5 hashes corresponding to signatures more than a week old. Note that configuration requests and and outgoing mail, while both signed by the user, have distinct message formats nymserv cannot confuse. A "config" or "send" request delivered to the wrong address does not affect the replay cache.

As mentioned above, nyms can have multiple reply blocks. To increase reliability, more than one reply block can deliver mail to the nym's owner. Since nymserv adds a unique identifier to each message it remails, client software can easily discard the duplicate messages generated by such a scheme. Of course, not all reply blocks have to go to the nym's owner. Some may simply discard mail after passing it through a chain of remailers. Such "fake" reply blocks can increase the average number of remailers an attacker must compromise without incurring the reliability penalty of lengthening the real reply block.

3.4 Miscellaneous features

Users may abandon nyms without deleting them, or even lose nym private keys. In such cases the nyms may nonetheless continue to receive mail. Indeed, they will likely do so given the pervasiveness of unsolicited commercial mail. To detect abandoned accounts, then, nymserv keeps track of the date on which it last verified a valid PGP signature by each nym's private key. We consider an account idle if we see no evidence of the existence of its private key for 90 days. Idle accounts receive a warning message every 10 days for 30 days, after which the software deletes them.

Finally, nymserv also functions as a finger daemon. Nym owners can optionally publish their nym PGP keys in their finger information.

3.5 Related servers

While nymserv provides the core functionality of nym.alias.net, several related servers on the machine deserve mention. A type-1 remailer, *remail*, functions as the back end to nymserv and the first hop in every reply chain. A mail-to-news gateway, *mail2news*, allows posting to news groups from nyms and anonymous remailers (though it has plenty of non-anonymous users, too). *smtpd*, a custom-built mail server, handles connections from remote mail clients and helps control abuse. Finally, nym.alias.net runs an ordinary type-2 remailer. This remailer processes over 500 messages a day and can be used as a final hop for mail sent to `config@nym.alias.net` and `send@nym.alias.net`.

4 Attacks and Abuse

Attacks on anonymous servers generally fall into two categories: attempts to expose anonymous users and attempts to silence them. Most existing work on such systems concentrates on withstanding the former—the more important of the two to resist. In practice, however, the threat of the latter is equally real. Users of an anonymous service often express unpopular opinions, which incite efforts to silence them or even shut down the service. One of the most effective means of closing an anonymous service is to abuse it. If, by abusing the service, one can stir up sufficient trouble, people will eventually no longer tolerate its existence.

This section discusses many forms of abuse we have anticipated and encountered while running nym.alias.net, and gives solutions we have implemented or envisaged to counter the abuse. In designing solutions, our goal was to avoid blocking problematic messages with manually-tuned filters. Such filters would constitute censorship, make us liable for messages we did not block, and even provide incentive for abuse. Other remailers' experience has shown that people unhappy with anonymous users will purposefully abuse a server if by doing so they can stop legitimate messages.

4.1 Harassment

Virtually every anonymous remailer periodically gets used to send offensive or harassing email to someone who does not want to receive it. The sender of such messages can never be tracked down, but the recipient of the mail can be blocked from receiving any further anonymous correspondence. Such blocking is known as *destination-blocking*.

Rather than manually process requests to be blocked, we implemented a destination-blocking scheme for our type-2 remailer that requires no intervention on our part. When a user `x@y.com` sends mail to `dstblk-request@nym.alias.net`, the system first sends mail to a few addresses like `owner-x@y.com` to try to reach the list administrator in case `x@y.com` is a mailing list. Each message is sent from a unique address containing random data. If someone replies to any of the mail messages, `x@y.com` gets blocked. Otherwise, if all the messages bounce, another message is sent to `x@y.com` asking the user to confirm the block request. This lets users block their own addresses, but requires the consent of mailing list administrators to block mailing lists from receiving anonymous mail.

Surprisingly, despite being prepared to apply this blocking system to nymserv, it has never proven necessary. In two years of operating the pseudonym server, we have not destination-blocked a single person. Most content-based complaints we receive about nym addresses concern postings to public forums such as Usenet news groups. People sometimes ask us to terminate nym accounts. However, messages from a troublesome nym user always come from a particular email address. This makes them easy to ignore with news reader killfiles. We believe that canceling accounts of obnoxious users would only make matters worse by driving them to post in other ways less easily filterable. We therefore have never closed a nym account.

4.2 Exponential mail loop

Nyms can have multiple reply blocks. Since the nym server does not know where any of the reply blocks point, two of a nym's reply blocks could very well point back to the nym itself. Such a configuration causes an exponential mail loop. To prevent such loops from overwhelming the server, nymserv limits the amount of mail a nym can receive each day. It keeps a running count of the total number of message chunks remailed for each nym in the current 24-hour period. When a nym with C reply blocks receives a message B bytes long, that nym's chunk count increases by $C \cdot \lceil B/32K \rceil$. If a nym's chunk count ever exceeds 512, nymserv disables the account: No more mail can be sent from the account, and any mail to the nym bounces. The user then receives a warning that the account has been disabled. At that point the user must wait a day and send a PGP signed configuration message to the pseudonym server to reenable the account.

Nym.alias.net never suffered from an exponential mail loop. We anticipated the attack and built message limits into the first version of the software. Unfortunately, message limits do open nym users to a denial of service attack: An attacker can disable a nym by flooding it with messages. Fortunately, someone maliciously flooding the system with messages cannot easily remain anonymous, so such behavior can be dealt with as a traditional denial of service attack. As described in Section 4.4, our mail server also offers some protection against such mail bombing. Finally, message limits can actually increase security in some cases. Someone wishing to confirm a guess about the identity of a nym could otherwise attempt to fill up the real person's mail box by flooding the nym.

4.3 Bulk mailing

Early on in the history of the nym server, someone mailed some sort of chain letter pyramid scheme to tens of thousands of users. While we received a number of angry complaints to postmaster, the effects did not seem particularly bad. Enough angry people sent mail to the pseudonym itself that the chunk count exceeded the daily limit and nymserv disabled the account. Those complaining to the sender may have been satisfied to see their complaints bounce back with the message "account disabled." The disabled account probably appeared more like the result of a policy decision than an incidental consequence of exponential mail loop protection.

Some weeks later, a nym user filled up the mail queue—the area on disk where the server temporarily stores mail—with a number of 25 Megabyte outgoing mail messages. The messages contained a single line of text, repeated over and over. Such messages compress extremely well when encrypted with PGP, so anonymously mailing them as cyphertext to `send@nym.alias.net` did not pose any problems. Because a full mail spool disrupts service for other users, we modified nymserv to prevent a repeat of the incident: We began counting outgoing message chunks, per recipient, against the daily limit. This change also prevents the kind of bulk emailing done for the pyramid scam.

4.4 Mail-bomb

Every once in a while, someone decides to send as much mail as possible to an address at nym.alias.net. The perpetrator of such a mail-bomb can easily generate messages at a faster rate than the system can process them. Serious delays and overloading can therefore result, not to mention undesirable consequences from actually processing all the messages—often an advertisement going to tens of thousands of news groups.

To prevent mail bombs, we limit the rate at which any given person can send mail to the server. Of course, we must do so without compromising people's privacy. Users of nym.alias.net may never have to send mail directly to the machine, but many do so anyway—for instance when requesting help files from autoresponders or using our type-2 remailer as the first hop in a chain. We therefore cannot keep a database file with per-user message counts, as such a file might accidentally get copied, backed up, or leaked, and at some later point provide a list of potential users of the system. We can, however, use short-term sender statistics to limit incoming mail rates if we keep those statistics in memory and out of the file system[2].

These limits are enforced by the mail server we built, smtpd. Smptd uses non-blocking I/O to handle all connections from remote mail clients in a single Unix process. This structure makes sharing data structures across client connections trivial. It also makes the overhead of accepting network connections considerably smaller than for traditional servers that create one process per connection. Smptd imposes per-sender and per-host quotas on mail deliveries, periodically decaying usage counts to permit a steady but controlled inflow of messages. When clients exceed quotas, the server returns temporary error codes. This ensures that large but short bursts of traffic do not cause any lost mail—only delays. The server also limits the number of recipients per-message to 5, as mail-bombers will try to generate many copies of a message for each one they have to transfer over the network. The Internet mail protocol, SMTP, specifies a minimum limit of 100 recipients per message, but imposing a limit of 5 doesn't seem to cause problems so long as attempts to deliver more only result in temporary error codes.

In practice, these simple limits on mail traffic have proven quite effective. When mail-bombs come from different sender addresses, they usually come from one or a small number of machines running special mail bombing software. When mail-bombs come from a large number of hosts, they typically originate from a large service provider like aol.com. Such providers apparently make it more difficult to forge sender addresses. Sometimes mail-bombs get relayed through other people's mail servers. In such cases, the perpetrator cannot feel back pressure from our mail quotas; instead, the relay machine's mail queue simply fills up—perhaps not inappropriate punishment for running an open mail relay.

[2]This information must reside other places in memory, anyway. Moreover, we consider an adversary unlikely to seize our machine and pore over the swap partition for information just recently available through network eavesdropping.

4.5 Reverse mail-bomb

One day, we started receiving many complaints of the form, "I don't want to use your system, leave me alone," and "Why do you keep sending me this crap? I didn't request it." It turns out that someone was mounting reverse mail-bombs against people he did not like—forging hundreds of messages from his victims' email addresses to `help@nym.alias.net`, an address that replies to any mail with a copy of the nym.alias.net help file.

Without logs, we had no idea who was sending the forged help requests. Moreover, we certainly did not want to keep the kind of databases necessary to implement any kind of one copy per email address per day policy. We solved the problem simply: We modified nymserv to quote the headers of any mail sent to the autoresponder and send them back with the response. This informed victims of the reverse mail-bomb attack of where the forgeries were coming from, and let them deal directly with the administrators of those machines. The reverse mail-bombs subsided soon after this change.

4.6 Encrypted mail-bomb

We don't know if this attack has occurred, as victims would not know to complain to us. Someone could create a nym with a reply block pointing to a victim's email address and subscribe that nym to some high-traffic mailing lists. The victim would subsequently receive large numbers of PGP-encrypted messages through the remailer network.

We discourage this abuse by requiring users to confirm reply blocks. When a user submits a new reply block as part of account creation or reconfiguration, nymserv sends a confirmation request to the user via the new reply block, embedding a nonce in the Reply-To address. The new reply block does not become active until the user replies to the confirmation request. This scheme is not fool-proof, as the user must only confirm one reply block in a set of several. However, we suspect this confirmation process complicates encrypted mail-bombs enough that other misuses of the system become easier. Victims can always get destination-blocked at the last type-1 remailer in a reply block, if necessary.

Reply block confirmation has an added benefit. Users often submit reply blocks without testing them, and sometimes those reply blocks don't work. If nymserv requires reply block confirmation, it can garbage-collect new accounts with unconfirmed reply blocks after only a week, rather than waiting 120 days. Users who reconfigure working accounts with broken reply blocks can also continue to receive mail with the old reply block.

4.7 Creating many accounts

One evening we noticed a large jump in the number of nym accounts. A small script confirmed that about 80 recent accounts had just been created with the same PGP key. We worried that an attacker might try to create a huge number of accounts, maybe even running the file system out of i-nodes (each account requires 3 files). At this point, we began requiring reply block confirmation, which apparently slowed the person down enough that the problem did not continue.

We don't consider this line of attack particularly worrisome, however. First of all, with available software, PGP key generation requires CPU time and manual attention. Thus, people creating many accounts will tend to use the same PGP key for all of them, making the accounts easily detectable. Moreover, if necessary, a more challenging reply block confirmation process could thwart an automated attack with multiple PGP keys. For instance, to require manual intervention, confirmation requests could contain a GIF image of the confirmation nonce (perhaps in an OCR-proof font) rather than an ASCII representation.

4.8 Spam

Given the complexity of decrypting nym mail without good client software, many nym users begged us to do something to reduce the amount of unsolicited commercial email or *spam* they received. Of course, we couldn't filter mail based on content, as this would amount to censorship. However, we tried several approaches with some success.

First, we added a per-account configuration option, *nobcc* (no blind carbon copies), that tells nymserv to reject all messages delivered to a nym but not addressed to it. Many bulk emailers send spam through mail relays. They try to get as many recipients as possible out of each copy of the message they must transmit. Thus, the headers they send usually do not reflect all the recipients. People using nobcc have expressed much enthusiasm for the option and reported a 90% or better reduction in spam. Unfortunately, one cannot subscribe to mailing lists from a nym with nobcc set, as mailing list headers reflect the address of the list rather than that of the subscribers.

Second, we tried throttling the flow of spam. We created a number of "spam-trap" accounts on the nym server, and then began posting news articles from some of them. Mail delivered to a spam-trap account caused the mail server to delay future messages from the same sender by returning temporary error codes. This scheme had the nice property of making it virtually impossible to send mail to every single nym on the system. The delays would add up and eventually cause messages to bounce. We now believe this approach was a mistake, however. Someone sent mail to a spam-trap account through a remailer, and suddenly mail from the remailer started getting delayed. Fortunately, we caught this before losing any mail and disabled the mechanism.

Third, we modified our smtpd to refuse mail it cannot bounce. The server attempts to verify the sender address before processing the sender's mail. It does so by performing a hostname lookup on the sender address. If it gets a temporary error from the Domain Name System (DNS), it returns a temporary error code. If it gets a permanent DNS error, it returns a permanent error code. An examination the spam-trap logs around the time of the change indicates this may have reduced spam by 30–50%, though we did not calculate an exact number.

4.9 Spam-baiting

Interestingly enough, the worst problems we ever encountered resulted from spam mail that never even passed through nym.alias.net. One fantastically effective way to receive spam is to post to a newsgroup such as misc.entrepreneurs, biz.mlm, or alt.sex.erotica.marketplace. A single article in one of those newsgroups can bring the sender dozens of unsolicited commercial email messages in the weeks to come.

One day, someone apparently resolved to drive away non-spamming customers of what he or she considered spam-friendly Internet service providers, and to do so with spam. The person somehow obtained lists of customers, and started posting *spam-bait*—forged news articles from those customers' addresses in the newsgroups most likely to draw spam.

The attacker forged the articles through our mail-to-news gateway, which allowed anonymous remailers to set their own From headers[3]. To add insult to injury, this person created From lines with fake names, for instance:

```
From: customer@isp.under.attack
       (My^ISP^spams^I^should^switch)
```

Which sometimes resulted in personalized spam messages with lines like:

```
Dear My^ISP^spams^I^should^switch,
```

People became furious, but did not initially understand what had precipitated all this spam. Bulk emailers do the best they can to conceal their electronic identities, so victims could not easily complain to the senders of the spam. When someone finally did figure out what was going on, people turned on the remailer and mail-to-news gateway operators with a vengeance, and began bombarding us with complaints. Then, someone developed a daemon that automatically alerted victims of spam-bait to the situation, and incited them to action against the remailers. Curiously enough, these alerts were sent anonymously through the remailer network. Some remailers shut down because of too many complaints. Eventually, people started calling the official technical contact of our network to complain. Then they started calling him at home, in the middle of the night. At that point, we modified our mail-to-news gateway so that anonymous articles could not carry arbitrary From headers.

The perpetrator of this spam baiting was not yet through, however. Having lost the ability to paste From headers, the person began posting long lists of email addresses in the bodies of anonymous mail messages. Though people reacted angrily to this, too, these messages seemed to draw much less spam. We couldn't very well censor people posting lists of email addresses, so we reacted by posting more invalid addresses to those newsgroups than the spam-baiter was posting valid ones. Eventually the spam-baits subsided.

Was this really an attack against bulk emailers, or could it possibly have been intended to close down remailers? We cannot answer this question. One of the most vocal critics of remailers during this period had a history of digging up and publishing private information about people he did not like. As a consequence, he frequently met with anonymous criticism in public forums. This person demanded we filter all news articles containing his email address—allegedly to reduce the amount of spam he received. Of course, such a filter would also have had the effect of blocking anonymous followups to his postings. Coincidence? Either way, this story drives home the point that abuse is one of the most effective attacks on an anonymous service.

4.10 INN exploit

One day, someone posted the news article shown in Figure 2. This malformed control message exploits a bug in the Unix news server software INN to mail a copy of the system's password file to

[3]Anonymous remailers allow users to paste arbitrary headers into outgoing mail, but not to modify the standard ones. Pasting a From header has the bizarre effect of creating a message with two From headers—something illegal for news articles. Our mail-to-news gateway simply removed all but the last From header of news articles.

```
From: root@ed.com
Newsgroups: alt.test
Control: newgroup
   `/usr/bin/sed:-n:'/^#+/,/^#-/p':${ARTICLE}|/bin/sh`
   moderated
Date: 9 Aug 1997 03:00:01 -0700
Message-ID: <m0wx8tn-0017nnC@www.state.or.us>
Apparently-From: root@ed.com

#+
/bin/cat /etc/passwd | /bin/mail voodoo@nym.alias.net
#-
```

Figure 2: This article collected the password files of news servers running INN.

voodoo@nym.alias.net. The article was neither posted through an anonymous remailer, nor through our mail-to-news gateway. (In fact, our mail-to-news gateway disallows newgroup control messages.) The first 512 news servers to receive the message probably mailed their password files to that address without incident, but then the exponential mail loop defeater kicked in and disabled the account. Subsequent password files then began bouncing back to news server administrators, some of whom were shocked to learn of the existence of a service like nym.alias.net.

While it's unfortunate that the pseudonym server participated in such an attack, the perpetrator didn't need nym.alias.net to steal the password files. He could instead have posted the stolen passwords to a newsgroup or mailed them to an unmoderated mailing list. At least the pseudonym server encrypted the password files before forwarding them on, so that only the owner of voodoo could read them.

4.11 Child pornography

Our worst nightmare came true. Someone allegedly posted child pornography from a nym. The FBI contacted us. They sent us a subpoena. We complied, and disclosed the reply block for the nym. Of course, a reply block doesn't necessarily give one the identity of a user. What we turned over to the FBI can only have helped them if they used it to issue more subpoenas.

The experience was not as bad as we had feared. The FBI did not seize our equipment. They did not threaten us or try to intimidate us. They did not ask us to start keeping logs, or try to convince us to shut down. We feared child pornography more than anything, but this happened and nym.alias.net survived.

5 Discussion

The types of abuse faced by anonymous servers fall roughly into three categories: conventional attacks, content-based abuse, and overloading. We discuss each type of attack in turn and suggest general principles that we have developed to deal with them.

Conventional attacks apply equally to machines without anonymous services. They include SYN bombs, mail bombs, and any attempts to exploit vulnerabilities in the server's operating system. Conventional attacks can be dealt with through conventional means, with the slight complication that anonymous servers may lack system logs.

What cannot go into logs may go elsewhere. When missing logs present a problem, one should try to record equivalent information where it can be retrieved in case of abuse but will not hurt the privacy of users. We did precisely this to solve the reverse mail-bomb problem of Section 4.5: We couldn't log the source help requests, so we instead started returning the information with the help file.

Avoid censorship. Content-based abuse consists of anonymously antagonizing people to turn them against the service providing the anonymity (whether justifiably or not). The first solution that comes to mind for fighting content-based abuse is often censorship. Unfortunately, no practical way of censoring anonymous servers exists. Manually inspecting anonymous traffic requires too much effort, and probably calls for judgements beyond the competence of the administrators. Automatic filters can simply be circumvented by abusers once they understand the blocking criteria. Moreover, filters risk blocking traffic from legitimate users. When people make unpopular statements through a server, this incidental blocking of legitimate users can actually provide an incentive for abuse. Finally, in the United States, censorship opens service providers up to legal liability for content they do not block.

Make it easy for people to filter anonymous messages. Of course, no one has the right to force himself on unwilling listeners, whether anonymously or not. Thus, a provider of anonymous speech must help unwilling recipients avoid anonymous messages. With email, one can accomplish this by clearly labeling anonymous messages and providing automated destination blocking, as described in section 4.1. In public forums such as Usenet, anonymous messages should have some property that lets people easily ignore them automatically with mechanisms such as killfiles.

Keep the filtering secret from the attacker. In either case, someone engaging in content-based abuse should have no way to know who ignores what messages. Otherwise the perpetrator can try to work around whatever mechanisms people use to ignore him.

Interestingly enough, the nym.alias.net pseudonym server has received considerably less abuse than our anonymous type-2 remailer. In fact, after two years of operation we have still not needed to implement destination blocking in nymserv. We can in part attribute less abuse to greater complexity of using the service, but good software does now exist for creating nyms. Pseudonymity may also just be a less appealing tool for harassment than anonymity, particularly since one can filter one pseudonymous user without filtering them all.

Anonymous servers also face the threat of being anonymously overloaded, for instance with bulk email or Usenet posting. Abuse involving large amounts of traffic differs from content-based abuse in two ways: First, techniques such as destination blocking and killfiles can no longer adequately resolve the problem; considerably many resources may still be wasted. Second, it is difficult to remain anonymous while overloading a server. During a mail-bomb attack, for example, even without mail logs, one can list open network connections and conclude that the site with 20 connections is the one causing trouble.

Recent history may suffice to prevent overload. Servers can prevent overloading by applying back pressure to aggressive clients. Because knowlege of current and very recent activity suffices to detect network overloading, anonymous servers need not sacrifice privacy to apply back pressure.

Smtpd, the mail server described in Section 4.4, exemplifies this fact. Though we keep no mail logs, smtpd keeps recent usage statistics in memory and uses them to limit the rate at which any given client can send mail. We found single-process, non-blocking network servers particularly amenable to this application, as they are highly efficient and permit easy sharing of data accross connections.

Put the human in the loop. Where direct network connections are not involved, demands can be imposed on clients to slow them down. For example, we would like it to remain hard for abusers to create huge numbers of pseudonyms. Currently, the nym creation process is slow enough and (thanks to PGP) requires enough human intervention that the difficulty has remained sufficient. Should the situation change, however, we could increase the burden of creating a pseudonym by charging hash cash. Ultimately, however, the most effective currency in which to charge for open services is human effort. When simpler techniques fail, this may be accomplished by requiring people to type in confirmation texts from images of OCR-proof fonts.

6 Conclusion

In practice, anonymous servers face more serious attempts to silence users than to expose them. Anonymous users can be silenced through denial of service attacks, which can be more difficult to prevent or stop in the presence of anonymity. A particularly vicious form of attack involves abusing anonymous servers until they must shut down. Nonetheless, we have run a pseudonymous email service for two years and easily survived the abuse.

We conclude that abuse must be factored into the design of any anonymous server, but the problem is not insurmountable. A variety of techniques can be used to slow down abusers or force them to reveal their identities.

Acknowledgements

We thank the many people at MIT who have actively supported nym.alias.net. There are few places in the world where one can count on such generous support.

We would also like to thank all those who have taken news feeds from nym.alias.net, making our mail-to-news gateway such a fast and reliable way of posting to Usenet.

References

[1] Masayuki Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In *EUROCRYPT 98*, May 1998.

[2] Adam Back. Hash cash. From `http://www.dcs.ex.ac.uk/~aba/hashcash/`.

[3] David Chaum. The dining cryptographers problem: *unconditional sender and recipient untraceability*. *Journal of Cryptology*, 1(1):65–75, 1988.

[4] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), Feb 1981.

[5] Lance Cotrell. Mixmaster and remailer attacks. From `http://www.obscura.com/~loki/remailer/remailer-essay.html`, 1995.

[6] Shlomi Dolev and Rafail Ostrovsky. Efficient anonymous multicast and reception. In *Advances in Cryptology—CRYPTO '97*, 1997.

[7] Michael Froomkin. Flood control on the information ocean: Living with anonymity, digital cash, and distributed databases. *U. Pittsburgh Journal of Law and Commerce*, 395(15), 1996. `http://www.law.miami.edu/~froomkin/articles/ocean1.htm`.

[8] Ian Goldberg, David Wagner, and Eric Brewer. Privacy-enhancing technologies for the Internet. In *COMPCON '97*, February 1997.

[9] Ceki Gülcü and Gene Tsudik. Mixing E-mail with BABEL. In *Proceedings of the ISOC Symposium on Network and Distributed System Security*, February 1996.

[10] Markus Jacobsson. A practical mix. In *EUROCRYPT 98*, May 1998.

[11] David Mazières. Instructions for nym.alias.net. Available by finger or email autoresponder from `help@nym.alias.net`.

[12] Charles Rackoff and Daniel R. Simon. Cryptographic defense against traffic analysis. In *Proceedings of the 25th annual ACM Symposium on Theory of Computing*, pages 672–681, 1993.

[13] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. Technical Report 97-15, DIMACS, April 1997.

[14] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *Proceedings of the 18th annual Symposium on Security and Privacy*, pages 44–54, Oakland, CA, May 1997. IEEE.

[15] Phil Zimmermann. *PGP: Source Code and Internals*. MIT Press, 1995.