Shibboleth: Private Mailing List Manager

Matt Curtin
Interhack Corporation
`http://www.interhack.net/people/cmcurtin/`

June 15, 2000

Abstract

We describe *Shibboleth*, a program to manage private Internet mailing lists. Differing from other mailing list managers, *Shibboleth* manages lists or groups of lists that are closed, or have membership by invitation only. So instead of focusing on automating the processes of subscribing and unsubscribing readers, we include features like SMTP forgery detection, prevention of outsiders' ability to harvest usable email addresses from mailing list archives, and support for cryptographic strength user authentication and nonrepudiation.

1 Introduction

> Then Jephthah gathered together all the men of Gilead, and fought with Ephraim: and the men of Gilead smote Ephraim, because they said, Ye Gileadites are fugitives of Ephraim among the Ephraimites, and among the Manassites. And the Gileadites took the passages of Jordan before the Ephraimites: and it was so, that when those Ephraimites which were escaped said, Let me go over; that the men of Gilead said unto him, Art thou an Ephraimite? If he said, Nay; Then said they unto him, Say now Shibboleth: and he said Sibboleth: for he could not frame to pronounce it right. Then they took him, and slew him at the passages of Jordan: and there fell at that time of the Ephraimites forty and two thousand.
>
> Judges 12:4–6, KJ

Shibboleth was conceived in early 1995 as a system that would allow a group of people to communicate freely with one another without concern about outsiders being able to infiltrate the group or to address the group by impersonating one of its members. At the time, widely available mailing list software had no means of effectively addressing these security and privacy requirements.

Since that time, the Internet has seen explosive growth, which has unfortunately increased significantly the number of unscrupulous and greedy marketeers among us online. These have devised many avenues of finding people, arguably invading their privacy "to market to them more effectively", and engaging in abusive practices like spamming [16, 10]. On today's Internet, it is difficult to participate in a forum of any type—even "private" ones—without being exposed to the risk of making oneself known to the outside world and having one's address published for all to see and to abuse.

Shibboleth has managed to avoid falling victim to the abusive behavior that is sadly becoming increasingly popular on the Internet today.

More information about the system, including its source code, is available on the Web at `http://www.interhack.net/projects/shibboleth/` .

1.1 Terminology

It will be easy to get lost in these interactions unless we explicitly state exactly what we mean by the these terms.

Family A group of mailing lists on the same machine, managed by the same installation of *Shibboleth*.

List A specific mailing list.

Moderator Human who approves messages and manages elements of a list. Each list can have a single moderator or a set of moderators. A moderator can be a moderator for several lists.

Administrator A human who deals with irregularities, such indications of mail forgery and digital signature failures. Additionally, administrators are responsible for the management of the *Shibboleth* installation as a whole, including such things as global configuration options.

Hosts A group responsible for the operation of the family of lists. Typically, this is the moderators and administrators as a single group.

Outsider A network user who has no association with the family of lists in the particular *Shibboleth* installation.

Insider A user who does have association with this particular family of lists; someone with a profile in the members database.

Subscriber A particular type of insider: one who is subscribed to a particular list.

Nym The name by which an insider is known. This might be some construction of the insider's name (like "first_last"), or it could be a handle by which he is known.

1.2 Differentiating Ourselves

It is important to note that *Shibboleth* is vastly different from most mailing list managers because most are designed to take care of routine issues of subscriptions; our software makes no attempt to automate this process to the same degree. Subscriptions are by invitation only. That is, the hosts initiate the subscription process by sending an invitation. If the user accepts, a profile is created, and he is welcomed to the family of lists.

2 High-Level Design

Before construction began, we wrote about the system's design goals and requirements.

2.1 List Structure

Shibboleth thinks of lists in groups, which we call families. If a group of security folks wants to work together, it can do so by defining a "family", which might be called "White Hats". When we refer to a White Hats member, we mean only that *Shibboleth* has a profile for that user in its member database. That name is typically the basis of deciding what prefix to use to reference the family of lists as a whole. In this example, we'll use "WH".

Within each family are any number of lists that belong to that family. The only lists that *Shibboleth* expects to find by default are

- an "-all" list (which includes everyone in the database) and
- a list for the list's hosts, those responsible for the operation of the lists.

There is nothing inherently special about either of these lists. Any insider may submit a message to the "-all" list; whether these require moderator approval or are allowed at all is purely a matter of configuration, as is true with any other list.

Any number of other lists can be created. Their names consist of the prefix ("WH" in our example) and its separator ("-" in our example) followed by a keyword to identify the list. A WH list to discuss projects might be "wh-projects", and another to handle otherwise off-topic traffic might be "wh-chat".

Each list has its own privacy level. That is, some lists can be available for anyone associated with that list's family. Others can require approval from a moderator. Thus, if there's a topic that would not be open for all WH members, it can be discussed on a list marked "private". Members may only retrieve archival postings from non-private lists, or private lists to which they are currently subscribed.

In no case is mailing to a combination of insiders and outsiders supported. *Shibboleth* will trigger an error for such messages, requiring administrator approval.

2.2 Design Goals

High-level goals for the system include

Members-Only Access Only insiders can send mail through the relay to any of the family's lists or other insiders.

Resistant to Forgery The system should be resistant to SMTP [14] mail forgeries [19]. Some basic header checking should be done. Additionally, the system should be able to verify and to generate digital signatures in order to make the possibility of convincing forgeries computationally infeasible.

Configurable Access Each member of the list will have an "access level" associated with his account. These are

- **Admin** List owner; can do anything.
- **User** Regular list member. Can read and post messages without special restrictions.
- **Novice** List member, but read-only. (Novices may submit articles, but they must be approved by a moderator, irrespective of the list's configuration.)

Timeliness Users should not have to wait "long periods of time" for processing of their mail.

Minimal Overhead The administrative burden must be reasonably manageable.

- Day-to-day tasks pursuant to the operation of the mailing list should be minimal. It shouldn't require a lot of time to manage a mailing list.
- Processing of the list should not be so expensive that it bogs down the machine running the system.
- A simple-as-possible configuration file should exist which would allow configuration changes to be made easily, either manually or by some mechanism in the software itself.

Usefulness The system needs to provide general utility that would be expected from a mailing list package, including

- Ease of use: the learning curve for users should be gentle;
- Archival of old messages;
- Digest creation: send digests to those who want to receive only specified articles from the archive;
- File server: a means for files of interest to the user community to be sent via email.

2.3 System Requirements

Specific system requirements. Features of this section indicate the feature must be part of the original implementation. Features that can be added in a subsequent version of the software are listed in the next section.

2.3.1 Moderation

A list of moderators is assigned for each list managed by the software. Every time a message requiring moderator intervention is processed, one copy of the message is sent to each address in the list of moderators.

This design is suboptimal. We have found that it can work in cases where there are few moderators and they have some agreement whereby they can decide who will process which messages. Nevertheless, this is relatively cumbersome, and would best be replaced by a mechanism to allow a moderator to fetch a number of messages in the queue, or to inquire as to the number of messages in the queue.

Each list can be configured for one of several moderation modes.

Unmoderated Moderate nothing: let all messages pass;

Moderate new threads Require moderator approval for only the first message in any thread;

Taboos Moderate messages having a header matching a given pattern;

Unproven Require moderator approval, except for messages whose authors have been "proven" using an authentication mechanism;

Fully Moderated Require moderator approval for everything.

"Taboos" is actually a special case: one can, for example, employ both "moderate new threads" and "taboos". If any taboo patterns are specified, they'll be used. Any `Subjects` matching one of the taboo patterns will trigger the moderation rule, irrespective of any other moderation configurations for that list.

2.3.2 Sender verification

Each member of the list has a list of patterns used to identify his known addresses. When a message arrives, the `From` header is compared to patterns in the profiles in the database so that the user who sent the message can be identified.

If a message comes from an unknown address, it can be spooled for a moderator to approve or to reject the message. Additionally, in order to prevent an outsider who mailed an insider from getting the idea that the address he used is valid, a "user unknown" bounce message is sent.

As part of the verification process, the `Received` and `Message-Id` SMTP headers [6] are examined to decrease the possibility of forging a message that appears to come from a known (legitimate) user. If a forgery is suspected, the system spools the message for an administrator to peruse. In practice, this rule is most often triggered by administrative changes in the user's Internet Service Provider (ISP), such as the addition of previously unknown mail relays, or changes in the user's behavior, such as the use of a new ISP for IP connectivity without changes in the user's email address. (As an example, someone might have an "address for life" from a university and always use that. As far as anyone who sees only `From` and `Reply-To` headers is concerned, there is no change when such a person switches ISPs. However, someone looking at `Received` headers will be able to identify that mail is definitely coming from a different source when such a person changes ISPs.) The administrator simply replies to mail *Shibboleth* sent, updating the profile to include the new relay, or a new pattern that will cover the relay.

The optional `X-Password` field is used as an additional means of convincing the system of the message's authenticity. Thus, if the `X-Password`'s value matches the user's password in his profile, SMTP header errors are ignored.

Note that remote MTAs, like any other users, are outsiders. That means that if Alice, an insider, sends mail to Bob, another insider, via *Shibboleth* and Bob's MTA sends any kind of a message to Alice (perhaps to report a delay in delivery, a bounce, or some other error condition), Alice will never see the bounce. Such a message would have a `From` header like `postmaster` or `mailer-daemon` at Bob's site, which will not be in the members database. The bounce message that Bob's MTA tried to send will therefore generate a "user unknown" error condition for the administrator's attention and generate a "user unknown" bounce. This prevents attacks against the system where someone would be able to send a message to a *Shibboleth* user by generating a false bounce.

2.3.3 Address Standardization or Shadowing

Each user should have a standardized address, in the form of "prefix-nym". Some might want their nyms to be their first and last name. Others might like their nyms to be some sort of unique token. The prefix should make it clear that the intended target is a *Shibboleth* user. A private installation called "white hats" might have a prefix of "wh", thus if Alice is a member of "white hats", her address to other "white hats" members would be `wh-alice@example.com`.

This serves two purposes.

- Insiders can easily mail each other by knowing only the first and last names (or the nym) of their addressee.
- All mail sent this way is subject to the same defenses as mail sent to a list is. Hence, snoops seeing mail in transit from the list to its recipients will not be able to gather email addresses (which might later be used to target for mailings or for some other purpose unrelated to the specific message from which it was harvested). For this to work effectively, it is necessary to ensure that headers from the original message are not passed through the system, especially `Received`, `Message-Id`, `X-*`, and `Reply-To`.

It should be noted that mail from an insider to the system can still be snooped. The effects of gathering addresses this way is far smaller than from the system to the list members, simply because the number of users affected is much smaller. The only effective way to gather all of the addresses of the system's users is to snoop all of the incoming traffic, which can only be effectively done on a point in the network that the system will exclusively use for its network connection.

2.3.4 Header Canonicalization

All mail from the system has a consistent `From` header format, which easily identifies the kind of mail that's been sent to the user by *Shibboleth*. For example:

._. Alice mail from Alice to a *Shibboleth* mailing list.

.^. Alice Mail from Alice to you via *Shibboleth*.

.#. Alice Mail from Alice to the mailing list for list managers.

.!. List Managers Mail from the List Managers via the "-all" list (the broadcast channel).

This format makes it possible to score articles easily in software in addition to identify visually why the article was received. ("Scoring" is a method of having the user agent mark messages, either as likely to be of interest to the user or likely not to be of interest. Scoring is described in various FAQs on Usenet and Usenet client software [1, 11, 18].)

2.3.5 Archives

Insider-only retrieval of messages posted to a given list. Further, although "public" lists—those available to any insider—will be retrievable by all insiders, only subscribers of a private list can retrieve messages from that list's archive.

2.3.6 Portability

Should run unmodified on any Unix machine where Perl 5.004 can be found. In theory, it would take very little work for the system to run under alien systems like Windows NT and MacOS, provided that the local Mail Transfer Agent (MTA) has some means of running a process as a given user.

2.3.7 Logging

The complete and unmodified headers of each message are logged. This was implemented to aid in determining precisely which headers triggered particular rules and whatnot after the fact. At runtime, if an error is reported, the headers causing the error are included, but we thought there might be some value in being able to examine the headers after the fact, or to be able to examine the data for patterns over a period of time. This is also consistent with the belief that where security is a concern, it is better not to need what one has than not to have what one needs.

2.3.8 White Pages

This feature allows each user to have a bit of text that would be intended as a biographical sketch available to other insiders. Even among insiders, individual users have control over who sees their white pages entries, by providing their preferred means of handling requests to see their white pages entries. Available options are:

Deny Do not let anyone (but administrators) retrieve the entry.

Ask Respond to each attempt to see the white pages entry by mailing the entry's owner and asking for approval. If the owner approves the request (by replying to the request with a "yes"), forward the entry to the requester. Refusals (made by replying to the request with a "no") result in the requester being told that the request has been denied by the entry's owner. If no answer is received in a configurable amount of time, the requester is told that the request has expired without a response from the entry's owner.

Allow Let anyone who is a subscriber of the list view the entry.

2.3.9 Remote Administration

While the software should be able to be administered directly from the machine where it's running, a set of commands should be available to the administrators, to allow them to administer the list without having to login to the machine running the software. Additionally, if there are changes that need to be made on more than one machine, the software should accept the command, and then sync the other systems to ensure that all systems are configured to agree with each other.

Because of the nature of Internet email and the potential effects of accepting commands from attackers, the software does not execute commands unless it can be proven that they originated from an administrator. For this purpose, a valid PGP digital signature is required. Recently, we have added the ability for administrators and moderators to authenticate themselves via the use of the `X-Password` header mechanism. We don't recommend use of this mechanism; if it's at all possible, use PGP.

2.3.10 File serving

Having files (such as system documentation, or other files of interest) available for insiders can be useful. Making them available via HTTP [7, 3] or FTP [15] could be problematic, as these are not subject to the same sorts of authentication mechanisms that exist in *Shibboleth*. Making *Shibboleth* do this itself has been most useful; requests are sent to *Shibboleth* as are any other requests, and responses are sent via email, with the requested files sent as MIME [8] attachments.

2.3.11 PGP Signature Generation

Each list has the option of having all of its traffic PGP signed [2]. That is, before *Shibboleth* sends a message, it PGP signs the message with its own key. This is an important option: all of the header checking in the world won't do us any good if a user can be fooled by a simple forgery that never went through the *Shibboleth* server. By enabling the option to have all messages PGP signed, such forgeries wouldn't be possible to perpetrate in a way that would fool most of the users. They'd want to know where the signature is, or why the signature doesn't verify properly.

2.3.12 Additional Sender Verification

In addition to SMTP header checks and the use of the `X-Password` header, *Shibboleth* supports PGP. A message signed with PGP will be verified on the basis of the signature's correctness. SMTP header checks will not be performed.

2.4 Additional Features

These are features that we should have, but have not yet implemented.

2.4.1 Peer support

Because of the processing requirements of the system, it is desirable to have the ability to share the load among several machines. Perhaps one machine receives incoming mail and handles verification, while another machine receives the mail from the first machine and handles subsequent processing. Once a more final version of the features are available, we'll describe each of the parts' interfaces, which will show where load can be split among machines.

2.4.2 Configurable Load

The software should be configurable to be a system pig (i.e., handle redundant tasks with parallel processes or threads) or to be nice (i.e., process everything sequentially; do not take up extra cycles.)

2.4.3 Digest Generation

Shibboleth's digest is different from most systems' digests. A digest can be requested by any user and an index of all of the articles which have been posted to the specified list will be returned. The user then chooses which articles he wants by quoting the lines containing the articles he wants to read and sending that mail back to the digester. The digester then sends those articles to the requester, separately.

Something we're considering is the ability for users to receive mail in batches, perhaps through some sort of digest capability as described in RFC 1153 [21].

3 Implementation

We'll limit our focus on implementation details that we believe to be the most relevant to our goals of privacy and security, particularly where we do things that aren't known to be done by other mailing list managers.

3.1 Language

Perl was chosen as the implementation language because it satisfied some important criteria for our application.

Portability Well-written Perl code will run unchanged on essentially any Unix implementation, and even on non-Unix platforms.

Safety Perl frees the programmer from dealing with the sorts of problems that are unrelated to the project at hand and historically the most problematic, including management of memory and of fixed-length buffers. Additionally, Perl's taint-checking allows us to run *Shibboleth* without worrying about the likelihood of unverified user data being handled unsafely.

Rich Library Perl has a rich library of modules that allows us to work with simple interfaces to protocols and systems we'll be using.

Pattern Matching Perl's sophisticated pattern-matching capabilities are well suited to the sort of analysis of text data that we do.

3.2 Sender Verification

Erring on the side of paranoia, we begin with the assumption that a message is from an outsider. We'll consider the verification process first and then consider the noteworthy parts in more detail.

1. If a digital signature is present it is evaluated.
   (a) If the digital signature is good, the sender's identity is accepted without need for further verification.
   (b) If the digital signature is bad, the message is sent to the administrator for review, with the note that the digital signature failed.
2. If the user can be identified, his nym is associated with the message. Otherwise, the message is sent to the administrator for review with the note that the sender could not be identified as an insider and a "user unknown" bounce is returned to the sender.
3. The `Received` and `Message-ID` headers in the message are examined. If any hosts that are not in the user's profile were involved in sending the message, the message is sent to the administrator for review, with the note that a header didn't match the user's profile.
4. If the user's profile contains a password, the message is checked for the presence and correctness of an `X-Password` header.

3.3 Getting Mail to *Shibboleth*

Mailing lists typically require a few aliases to ensure that mail directed to the list's address will be delivered properly, as well as such variations as `listname-owner`. Because we need *Shibboleth* to process not only things directed specifically to it or to any of its lists, but also to any other insider, we would require a larger number of aliases.

Shibboleth has the support necessary to generate and to maintain a separate MTA alias file.

We actually favor a newfangled option[1] to all of this alias maintenance. Most MTAs support a feature for aliasing multiple addresses to a single mailbox. For example, in recent versions of *Sendmail,* mail addressed to `user`, `user+foo`, and `user+bar` all get delivered to the same place. A Usenet FAQ describes how to do this in detail [13]. By using this feature, a site can create a user with a prefix that will be used for all of the family's lists. Then, addresses will be `user+list` or `user+nym`. *Shibboleth* will generate bounces for anything that doesn't exist. That is, if "nym" does not exist, it is be

[1] This options is not fully implemented.

Shibboleth's responsibility to inform the sender of the incoming message that the intended recipient address does not exist.

By having mail go to a user account, we can also place the burden of managing the ID of the user running the process on the MTA. Rather than having to write our own `setuid` front-end for *Shibboleth*, the MTA will perform the `setuid` for us. This saves us some headaches in permissions related to aliases and other files we need to access. It's also much more easy for us to run from a single unprivileged account this way, making all of our database files, archives, and other data unreadable to all other system users.

3.3.1 Identifying the User

Each user's profile, as is shown in Figure 1 contains a list of Perl patterns; an address that matches the pattern is associated with the user. This allows users to send mail from any of their accounts, without revealing how many such accounts they have, or even giving anyone any idea that they have more than one. This is a convenience issue, as mailing lists that accept mail only from subscribers are typically less intelligent and require that one send mail from the exact address that one uses to subscribe to the list. For those who do a lot of contract work (thus changing daytime address often), those who send mail from both home and work, or those who use more than one machine, our solution is much more workable.

Headers that we examine for this case are the SMTP envelope's `From` (sometimes called `From_` because MTAs have historically stored the value in a `From` header without a separating colon, but instead followed by a space and a timestamp) and the message's `From` header.

3.3.2 SMTP Header Checking

In an effort to prevent outsiders from being able to send mail by impersonating an insider and to prevent insiders from impersonating each other by means of trivial SMTP forgery, we examine each of the headers put in place by mail relays used to deliver the message as well as the "domain part" (right side: that which follows `@`) of the `Message-ID`.

```
wh-matt_curtin is Matt Curtin

Destination Email: cmcurtin@interhack.net
Valid addresses:   cmcurtin@interhack\.net
             cmcurtin@\w+\.interhack\.net
             cmcurtin@cis\.ohio-state\.edu
MX servers:  \w+\.interhack\.net
             \w+\.cis\.ohio-state\.edu
X-Password:
White page access:    allow
PGP key ID:           DEADBEEF
User is:
User subscribed:      wh-hosts, wh-all,
            wh-chat, wh-security
User is admin in:     wh-all
User is moderator in: wh-all
```

Figure 1: Typical User Profile

Again, each insider's profile contains a list of Perl patterns used to identify known SMTP relays. If any relay does not match one of the patterns in the profile, an error is signaled.

For convenience, administrators can specify "clusters", a token that will be associated with a list of patterns. So, for example, if an installation has some number of insiders who are all AOL subscribers, each can have the `*aol*` cluster in his profile, and then the `*aol*` cluster can be made to recognize hosts that fit the pattern `\S+\.(mx|mail)\.aol\.com` and `mrin[0-9]+`.

3.3.3 X-Password

We recognize that PGP—our preference for authentication—is not available to all. Nontechnical members on less capable platforms might especially have difficulty using PGP correctly and finding tools that allow them to use it conveniently. Lastly and ironically, there are companies whose security organizations have banned the use of PGP.

To provide an option for additional authentication for those who have no capability to PGP, *Shibboleth* supports a user-defined header that contains a password. This isn't considered a "high-security" option, as it's susceptible to replay attacks [9], just as is any reusable-password authentication scheme whose credentials are sent in cleartext.

It is also important to note that use of PGP also provides the feature of nonrepudiation. If the `X-Password` feature is used instead of PGP, there is no nonrepudiation feature in the system.

3.4 Address Standardization

Several benefits are realized by the address standardization feature. A common problem with mailing lists is that responses to messages sent to mailing lists will have both the mailing list itself and the author of the message that prompted a response included on the copy-to list, resulting in some subscribers receiving several copies of messages in response to theirs or of messages even further down the thread.

Implementation of the address standardization mechanism requires that all messages—both insider-to-insider and insider-to-list—be processed by *Shibboleth*. As such, we can prevent subscribers from receiving multiple copies of the same message, even if the author specifies a specific user's address and a list to which he subscribes.

Sometimes, a *Shibboleth* user's shadowed address will fall into the hands of outsiders, either by way of oversight, perhaps a careless person forwarding the mail to outsiders without removing such headers, or intentionally. Perhaps a subscriber has been ejected for some reason and has saved addresses of other insiders.

An outsider sending a message to an insider or addressing one of the *Shibboleth* lists will receive a "user unknown" bounce. A copy of the message will also be sent to the list administrators. This has proven to be an effective means of preventing unwanted traffic from finding its way to insiders. Never has unsolicited bulk email ("spam") ever made it to a subscriber, though some of our addresses appear to have been sold as part of at least one spam software package; the administrators got copies of the spam, as they would any message from an apparent outsider, but the insiders had no idea that the spam was ever directed their way. Individuals who unscrupulously add others to their lists without confirmation occasionally add an insider's shadowed address. After receiving the bounce, the list operator who added the shadowed address will typically remove the address. (If not, he will just keep getting bounces!)

If an insider sends mail that includes outsiders, the message goes to an administrator for handling. If the administrator approves the message, *Shibboleth* will remove the outsiders' addresses, thus preventing any replies from insiders also being directed to outsiders. Replies from outsiders that include insider addresses will, of course, bounce.

3.5 Cryptographic Strength Moderation

A potential weakness for any moderation scheme is the authentication mechanism used for the moderators to identify themselves and the messages that they approve. As *Shibboleth* is fully integrated with PGP (albeit "classic" IDEA/RSA/MD5 PGP 2.6.x), we can easily require cryptographic strength moderation. Thus, defeating the moderation scheme would require a crack of a moderator's key, or the ability to forge an MD5 hashed signature, both of which are currently computationally infeasible.

Administrative functions also require the same level of authentication.[2]

Messages to be moderated appear to be bounces to most mail user agents (MUA). MUAs that support a feature like "retry bounce", such as Kyle Jones's excellent *VM* for XEmacs and GNU Emacs, will work best, as the submissions can be read from the moderation queue, brought into a composition buffer, PGP signed, and then sent on their way.

Shibboleth, upon seeing that the message has a valid PGP signature belonging to a moderator, will send the message on its way. It is noteworthy that only the part of the message that was signed will be passed. Anything outside of the delimiters for the signed message will not be included; neither will the PGP signature itself.

Because we use a version of PGP that is not MIME-aware, PGP does not interfere with any of the MIMEisms that are in the original message. Its `Content-type` header is left unchanged, and the signature engulfs the entire message body, including all attachment data. Thus, when *Shibboleth* verifies the

[2] Again, we recently added the ability to allow the simple `X-Password` header authentication for moderators and administrators. This isn't something we recommend that administrators enable, but provide it as a means of authentication where PGP isn't an option.

signature, it is verifying not only the content of the message, but also of all of the attachments. Any changes to the attachments in transit (such as, for example, the addition of a virus-infected file) would cause the signature to fail, causing the attempted approval message to go to the administrator for review.

3.6 Load Sharing

Instead of requiring that all processing and all outbound messages be sent via the same host, *Shibboleth* provides the ability to direct its outbound mail to another host. This allows the system running *Shibboleth* to share the burden of running the list: one machine can receive all of the incoming messages and handle that processing; another can be used to send all of the outbound messages.

Other features that would enable a more granular level of load sharing have been planned.

3.7 Outgoing Messages

To prevent the leakage of header data that could provide information about a poster's address, *Shibboleth* does not simply forward the message to the appropriate users. *Shibboleth* actually creates a new message altogether. *Shibboleth's* configuration contains a list of Perl patterns whose headers should be preserved. This allows us to include things like `Subject`, without having to worry about what data might bleed in other headers.

Outgoing messages have their own `Message-ID` headers generated, thus preventing the poster's MTA's domain name from being present in outgoing messages. Thus, all `Message-ID`s have the site name of the MTA that *Shibboleth* uses. As such, it is safe to configure *Shibboleth* to pass headers like `In-Reply-To` and `References`, which can be used for building threads. These headers should contain only `Message-ID`s that were generated by *Shibboleth*.

In practice, we also allow such personal touches that are contained in headers like `X-Face` and `X-Attribution`. MIME works by passing such headers as `MIME-Version` and `Content-type`. This is all completely under the control of the list family administrator.

For the sake of safety, *Shibboleth* adds an `X-Loop` header and will recognize its own token as a means of preventing mail loops. If the header and token are present in an incoming message, *Shibboleth* will not deliver the message; it will exit, logging the detection of a mail loop.

4 Administration

Administration of a *Shibboleth* installation tends to be somewhat intensive at first, while the system learns different addresses the insiders will use and what mail relays will deliver their mail. Once the system has been running for a while, the regular posters will have their profiles set properly, and the system will not report problems for them except in the cases of real errors or in major system configuration changes (such as an ISP buying another and changing an insider's address).

The additional administration comes from the need to train the system to recognize the legitimate mail relays that are used by various insiders. Because we use Perl patterns, one need not list every possible relay host for each user: to allow any host with an alphanumeric name inside of the zone `example.com`, we can specify the pattern `\w+\.example\.com`.

Beyond this, there is little difference in the administrative operation of a *Shibboleth* list, as compared to another list manager, such as *Majordomo* [5]. Because of our authentication system's requirement for strong authentication, it isn't practical for us to implement a web-based interface like that of *Mailman* [20].

5 Future Work

We have identified some areas where *Shibboleth* could be improved.

5.1 Other Secure Mail Standards

Support for emerging standards, such as OpenPGP [4] (which supports newer, sometimes more desirable options for ciphers and hashing algorithms) and perhaps S/MIME [17], would be useful. There are some tricky matters to be resolved here, especially in allowing a moderator to sign a signed message and sanely preserving MIME `Content-type` data, but it seems well within reach.

An additional benefit to support of these standards would be that one could reasonably sign all outgoing messages without the need to be concerned whether such signing would create problems for MIME-formatted messages. (Our RFC 1991-compliant PGP, though convenient for approving MIME-formatted messages, isn't practical for signing things that will be reviewed by most MUAs. Our option to have *Shibboleth* sign all outgoing messages would prevent most MUAs from being able to decode MIME-formatted data properly. If we could perform the signing in a way that's friendly with MIME, the sign-outgoing-mail feature would be much more useful.)

5.2 Better Peer Support

Especially in an installation where many users digitally sign their submissions to the mailing list and where the system is signing each message it relays, CPU overhead could be significant. It would be nice if there were a more intelligent way to spread the load among some set of hosts, rather than placing all of the processing burden on one host and only providing the option of giving the delivery burden to another.

5.3 Better Moderation Scheme

Currently, there is a race condition for lists where there are multiple moderators: all moderators for a particular list get all copies of messages for that list to be moderated. As a result, multiple approval (and thus, multiple posts) are possible. We have managed this problem by convention, but a more intelligent system for moderation would include the ability for a moderator to get some number of messages from the moderation queue and then work on them himself, without leaving the possibility of duplicating another moderator's work.

5.4 Tolerance of SMTP Irregularities

Our current implementation treats any irregularity as an error that could indicate forgery, thus requiring administrative attention. We have been thinking about ways of lightening this load somewhat without losing the benefits of SMTP header verification.

The most interesting solution to this problem we've considered so far is the addition of a runtime parameter that will specify the number of irregularities that can be tolerated per message. Another runtime parameter could be used to indicate whether *Shibboleth* should automatically add the previously unknown relays to the user's profile if they are fewer than the specified number. Messages that have come through more unknown relays than that number could continue to be handled in the manner that today's errors are: administrative attention. Indeed, if they are not, messages will either mysteriously "disappear", or we'll override any benefits derived from checking the headers at all.

5.5 Local User Dilemma

People with local access via normal user accounts to systems that provide network services can often create problems for those network services. *Shibboleth* is no exception. Specifically, users who can inject a message locally can fool our SMTP header verification. Today, we do a simple enumeration of relays and compare those hosts to patterns in a user's profile. If any pattern in a user's profile does not match, no error is signaled. This is a mildly difficult problem, since the primary use for these lists of patterns is the allowance of posters to send either from home or from work. Were we to trigger an error if a pattern does not match, a message from such a user would need to be sent from home *and* work at the same time.

The best solution to this problem is to improve the intelligence of the SMTP header parsing. Rather than being a simple enumeration of hosts to be checked against a pattern, the headers should be parsed, thus telling us which host injected the message into the network and which hosts merely relayed the message.

Even in such a scenario, we're at the mercy of MTAs that are out of our control. Perhaps in practice, this will be good enough to provide us much more security than we have now. Whether this is true will probably depend on each installation, where its users originate their mail, and which MTAs those systems run.

Depending on the MTA and its configuration, local users also have the ability to determine if a "user unknown" bounce is legitimately generated by the MTA or if it has been tricked into returning such an error.

For the time being, we have to consider local users "trusted", and reducing such trust is an area for further study.

5.6 Reducing Necessary Trust in Administrators

As implemented, *Shibboleth* users must trust the administrators. Though insiders are protected from outsiders and from each other, they are not protected from administrators. Also, all administrators have full administrative access to the system.

We're particularly intrigued by the possibility of future releases of *Shibboleth* actually distrusting administrators and hiding sensitive information—such as the members database—from them. Rather than having any single administrator being able to query the database for full user profiles, for example, what if we were to require that multiple administrators' signatures would be necessary to have *Shibboleth* reveal such information or perform especially "risky" operations?

Another interesting possibility is where hosts would be used for initially introducing new insiders to the system, after which users would be individually responsible for maintaining their own profiles. (It seems obvious that this would work only for rather experienced or technical users, but could the system be made to be smart enough that this is no longer true?)

6 Conclusions

We have shown that it is possible for a group of people who wishes to keep to itself can do so, even in today's Internet. Despite the lack of strong authentication mechanisms for email at any level other than application, it is possible to identify mail from insiders, letting it flow normally, without requiring that mail from outsiders flow just as freely. Even less-than-perfect schemes, like SMTP's simple headers, can be employed to determine reasonably the authenticity of a message.

Forcing all mail to insiders, public and private, to run through *Shibboleth* solves the problem of posters receiving multiple copies of posts later in the threads to which they contribute.

Because only *Shibboleth* nyms are known, someone cannot infiltrate the group, collect messages, and then use those addresses for his own nefarious purposes later. Once someone is no longer an insider, those addresses immediately become useless to him.

A simple checklist of features and requirements will show that we have satisfied our requirements. Our experience with running mailing lists with *Shibboleth* and enduring attacks against it gives us evidence that these safeguards have in fact worked.

On a daily basis, we hear about the difficulty of privacy and security on the Internet. Though not a substitute for the needed support for security in our infrastructure through efforts like IPsec [12], making better use of what we have available can move us in the right direction and bring us much closer to the sort of well-behaved system that does our bidding, and only our bidding.

7 Acknowledgments

Much of the first implementation of the original design owes its existence to Eugene Sandulenko (Женя Сандуленко). Thanks to Ed Sheppard for suggesting the name and to Carolyne J. Butler for careful proofreading.

References

[1] Usenet software: History and sources. Usenet FAQ, February 1998. [online] http://www.faqs.org/faqs/usenet/software/part1/.

[2] D. Atkins, W. Stallings, and P. Zimmermann. PGP Message Exchange Formats. RFC 1991, August 1996.

[3] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol – HTTP/1.0. RFC 1945, May 1996.

[4] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP Message Format. RFC 2440, November 1998.

[5] D. Brent Chapman. Majordomo: How I manage 17 mailing lists without answering "-request" mail. In *Systems Administration (LISA VI) Conference*, pages 135–143, Long Beach, CA, October 19-23 1992. USENIX.

[6] D. Crocker. Standard for the format of ARPA Internet text messages. RFC 822, August 1982.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, June 1999.

[8] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, November 1996.

[9] N. Haller and R. Atkinson. On internet authentication. RFC 1704, October 1994.

[10] S. Hambridge and A. Lunde. DON'T SPEW A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam*). RFC 2635, June 1999.

[11] Peter J. Kappesser. trn newsreader FAQ, part 2: Advanced. Usenet FAQ, August 1998. [online] http://www.faqs.org/faqs/usenet/software/trn-faq/part2/.

[12] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, November 1998.

[13] Eli Pogonatus. Email addressing FAQ (how to use user+box@host addresses). Usenet FAQ, December 1998. [online] http://www.faqs.org/faqs/mail/addressing/.

[14] J. Postel. Simple Mail Transfer Protocol. RFC 821, August 1982.

[15] J. Postel and J.K. Reynolds. File Transfer Protocol. RFC 959, October 1985.

[16] Jon Postel. On the Junk Mail Problem. RFC 706, November 1975.

[17] B. Ramsdell (Ed.). S/MIME Version 3 Message Specification. RFC 2633, June 1999.

[18] Justin Sheehy. Gnus (emacs newsreader) FAQ. Usenet FAQ, January 2000. [online] http://www.faqs.org/faqs/gnus-faq/.

[19] Bob Thomas. On the Problem of Signature Authentication for Network Mail. RFC 644, July 1974.

[20] John Viega, Barry Warsaw, and Ken Manheimer. Mailman: The GNU mailing list manager. In *Twelfth Systems Administration Conference (LISA '98)*, page 309, Boston, Massachusetts, December 6-11 1998. USENIX.

[21] F. Wancho. Message Digest Format. RFC 1153, April 1990.