12

In this chapter:
- *General Issues*
- *Authentication*
- *Security*
- *UW IMAP Utilities*

UW System Administration

The UW IMAP server is so much of a plug-and-play package, there's not a lot of system administration to talk about. A few things should be mentioned, though, so we'll cover them before going on to bigger issues, like security.

General Issues

Some broad issues should be addressed up front. As we've mentioned several times already, you would probably have a perfectly good IMAP server setup if you just slapped the location of the prebuilt binary that probably came with your operating system into *inetd.conf* and ran it. Unix system administrators being what they are, however, we'd be remiss in our duties if we didn't give you a handful of "nerd knobs" to tweak on the UW server, so here goes.

IMAP Alerts

The UW server supports IMAP alerts, *motd* type messages that all IMAP users see in their IMAP client when a connection to the server is first established.

To set up an alert, put the alert message text in the file */etc/imapd.alert*.

Because many clients don't handle IMAP alerts perfectly, keep the alert text down to one line. Also, keep in mind that the alert message is displayed every time a user connects to the server, so it's best to use alerts only for critical messages.

Disabling the mbox Driver

If you compiled the UW server to support the mbox driver (it's enabled by default), the server's behavior when an INBOX is opened may not be what you

expect or desire. With mbox support, when INBOX is opened, the server checks the user's home directory for a Unix mailbox format file called *mbox*. If it exists, the server selects that file as the INBOX and transfers all mail from the user's mail spool file to the *mbox* file.

The mbox driver can be disabled by editing the top-level Makefile and removing mbox from the `EXTRADRIVERS` list and rebuilding the UW server. Rebuilding and reinstalling the server will not take care of messages that were already transferred from the mail spool file to the *mbox* file. Those messages have to be moved back to the spool file by hand, or the user will not be able to read them unless she explicitly opens *mbox*.

Alternative Default Subdirectory for User Mailboxes

Many users of host-based mail programs will find their mailboxes sitting under *~/mail/*. If you want UW to default to that directory and have it appear as "~/" to IMAP clients, go to the file *src/osdep/unix/env_unix.c* in your source distribution, change some code in the function `env_init()`, and rebuild the server.

In the function `env_init()` in *env_unix.c*, you'll find the following line assigning a value to `myHomeDir`:

```
myHomeDir = cpystr (home);       /* use real home directory    */
```

Add a line so the assignment is made to something more of your liking. Something like this would do nicely:

```
sprintf (tmp,"%s/mail",home) ;/* I'd rather have this value */
      myHomeDir = cpystr (tmp) ;/* original string copy line  */
```

Changing Location of INBOX

The only easy way to contain users' disk usage on your UW server is through OS quotas, because UW doesn't support IMAP-specific quotas. For this reason, you might want to move the default INBOX for your users to somewhere in their home directory structure so it can be contained by the same quota as the rest of their personal files.

You may, for example, want to change your delivery agent so that it delivers to *~someuser/mail/incoming* instead of */var/spool/mail/someuser*. To make that change, go to the function `sysinbox()` in *env_unix.c*:

```
/*  Return system standard INBOX
 *  Accepts: buffer string
 */
```

```
char *sysinbox ()
{
  char tmp[MAILTMPLEN];
  if (!sysInbox) {              /* initialize if first time */
    sprintf (tmp,"%s/%s",MAILSPOOL,myusername ());
    sysInbox = cpystr (tmp);    /* system inbox is from mail spool */
  }
  return sysInbox;
}
```

and change the `sprintf` line, making the assignment to `tmp` something like this:

```
sprintf ( tmp,"%s/mail/incoming",myhomedir() );
```

Permissions

/tmp must be world-writable. If it's not world-writable, the UW server will log syslog messages like this one, complaining that it cannot open mailbox lock files:

```
Apr  2 14:44:51 serverhost imapd[5759]: \
    Mailbox lock file /tmp/.80001d.29bc2 open failure: Permission denied
```

If you're using Unix mailbox format, you'll only be able to open mailboxes in read-only mode until the permissions are fixed. To correct the permissions, use the command:

```
# chmod 1777 /tmp
```

The permissions of individual lock files under */tmp* should be read-write by world (0666). If the permissions are set to something other than 0666, then they were changed either in error or with malicious intent (see the section "Security" later in this chapter).

Mailbox Formats

The question of maildir format comes up more and more often as MTA alternatives to sendmail, such as Qmail, take on more of a following. Although UW supports a variety of mailstore formats, Qmail's maildir format is not supported. There are no plans to add support for maildir.

Authentication

One of the appeals of UW IMAP is its flexibility with regard to authentication and mailbox format schemes. Here's a brief overview of some of the more common areas of concern with UW authentication.

Disabling Plaintext Passwords

It's possible to disable plaintext passwords before they disable you. Doing so involves rebuilding the server with an alternate authentication method and the password type set to nul:

```
% make lnx EXTRAAUTHENTICATORS=gss PASSWDTYPE=nul
```

At least one `EXTRAAUTHENTICATOR` (`gss`, in the previous example) must be specified, or the server will have no mechanism for users to log in.

Enabling Anonymous Login

Anonymous IMAP login is disabled by default. To enable anonymous login, create an empty file called */etc/anonymous.newsgroups*. Note that anonymous user access is limited to mailboxes in the News (*#news/*), FTP (*#ftp/*), and Public (*#public/*) namespaces.

Using PAM for Plaintext Passwords

UW includes a port for Linux distributions that include support for PAM (Pluggable Authentication Modules). If your Linux distribution supports PAM, then rebuild the UW server with the Linux-PAM (`lnp`) port:

```
% make lnp
```

Solaris systems prior to Solaris 8 have PAM implementations that vary slightly from the Linux PAM implementation. To support plaintext passwords with PAM, Solaris sites should rebuild UW as follows:

```
% make clean
% make sol PASSWDTYPE=pmb
```

Other systems should build with `PASSWDTYPE=pam`:

```
% make systemtype PASSWDTYPE=pam
```

System types are too numerous to list here. They're listed as comments in the top-level UW IMAP Makefile and discussed briefly in Chapter 10, *Introduction to the UW IMAP Server*.

CRAM-MD5

The UW server supports CRAM-MD5* (Challenge-Response Authentication Mechanism). To enable CRAM authentication, all that needs to be done is to create the

* *http://www.imap.org/docs/rfc2195.html*

CRAM-MD5 authentication database. If the CRAM-MD5 authentication database exists, then plaintext password authentication via the IMAP LOGIN command will use the CRAM-MD5 passwords instead of Unix passwords.

The default location for the CRAM-MD5 authentication database is */etc/cram-md5.pwd*. The format of the database is shown in Example 12-1. The first field of each line is the username; the second field is the password. Fields are separated by a tab. Lines beginning with # are comments.

Example 12-1. CRAM-MD5 Authentication Database

```
# CRAM-MD5 authentication database
# Format: <username><tab><password>
johndoe234&xx7
msmithmypasswd
qpublicbad!pass
```

You may have noticed that the password appears in the authentication database unencrypted. To protect the database, the file permissions should be restricted to read and write access by root only. Use the following command to change the permissions on the database:

```
# chmod 0400 /etc/cram-md5.pwd
```

Every CRAM-MD5 database entry must correspond to an entry in */etc/passwd*. Think of *cram-md5.pwd* as a variation on */etc/shadow*. Each holds only username and password information. It is left to */etc/passwd* to hold the UID, GID, GECOS, and home directory information. It is permitted, and probably recommended in some configurations, for the user's entry in */etc/passwd* to be disabled either by locking the username in the */etc/passwd* or */etc/shadow* or by assigning the user a non-functional shell.

Kerberos

To build the UW server to support Kerberos V5, use:

```
% make systemtype EXTRAAUTHENTICATORS=gss
```

The *gss* authentication module supports Kerberos V5. Note that users will also be able to log in using plaintext Unix authentication, unless you specifically disable plaintext authentication (see earlier in this chapter). The UW server does not include support for Kerberos V4.

Security

As with any other package, there are some security housekeeping items that you ought to address when you install it.

SSL and TLS

SSL and TLS will be supported in IMAP 2000 (due to be released during production of this book), but they are not supported in earlier versions. The University of Washington has an SSL patch kit for the UW IMAP server, which adds SSL and TLS server support to POP and IMAP. Unfortunately, UW cannot make it available even with the recently relaxed U.S. government export restrictions, because of lingering governmental restrictions with regard to which countries still may not receive encryption technology from the U.S., and certain peripheral issues, such as the distribution of crypto-binaries.

As an alternative approach using freely available open source software, IMAP can be tunneled through SSL or SSH using the techniques discussed in Appendix B, *Adding SSL Support to IMAP*.

Permissions on Files Under /tmp

As we mentioned before, in addition to */tmp* permissions needing to be set to 1777 (`drwxrwxrwt`), all lock files created in */tmp* by IMAPD must have the permissions of 0666 (`-rw-rw-rw-`). Yes, that does open up the possibility of malicious or accidental denial of service by changing or removing the lock files, but any permissions other than 0666 will keep shared mailboxes from working. On this issue, you have two choices. One, to go on living with the problem and knowing that it may be fairly easy to track down lock file vandals. Two, to use this opening as a good excuse for running a dedicated mail server without general CGI or shell access.

Mail Spool Directory Permissions

C-Client and, by extension, the UW IMAP server run completely without privileges, and thus require that the spool directory (in most cases, */var/spool/mail*) have 1777 protection (`drwxrwxrwt`). That enables read, write, and execute for user, group, and other, and it enables the sticky bit on the directory. That means that, although everyone can see the names of everyone else's lock files, they can't change them or otherwise mess with them. About the only lock-file-specific attack malicious users could do would be to preemptively create huge numbers of lock files in an attempt to block users' read-write access to their mailboxes.

UW IMAP Utilities

Chapter 18, *IMAP Tools*, lists a variety of tools to perform nearly every common IMAP task. The UW IMAP Utilities are available at *ftp://ftp.cac.washington.edu/imap/imap-utils.tar.Z*. Download and unpack the tools at the same level as your UW IMAP source:

```
% lynx -dump ftp://ftp.cac.washington.edu/imap/imap-utils.tar.Z | uncompress |
tar xvf -
```

You'll see several subdirectories. Each subdirectory has the source code for a separate utility. Change to the utility subdirectory and build the utility:

```
% cd chkmail
% make
```

In each utility subdirectory, you'll find a manual page describing the purpose and usage of the utility. Here's a brief description of the utilities:

chkmail

chkmail is a utility for checking how many messages in a given mailbox have the Recent flag set. Any mailbox that can be described in C-Client mailbox syntax (as in the PINE configuration) can be checked for new messages with this utility.

dmail

dmail is a drop-in replacement for *binmail*. It's intended for use as a local delivery agent in every case *except* as an MDA used by the transport agent. Essentially, *dmail* is intended for use in user processes, and *tmail* is intended for use in server processes.

icat

An IMAP version of */bin/cat*, *icat* can be used to display all the messages in a given folder in a single output stream. Users may also filter based on flags, dates, or other header values. *icat* is an excellent tool for migrating mailboxes away from proprietary or monolithic mailstores that happen to have marginal support for IMAP.

ifrom

ifrom gives a directory listing of the specified IMAP mailbox with message number, status, sender, date, and subject.

imapcopy

imapcopy and *imapmove* copy (or move) messages from an IMAP INBOX to an existing local mailbox.

imapxfer

The *imapxfer* utility uses IMAP to copy entire mailboxes from one host to another, creating them on the destination host if need be.

mbxcopy
: *mbxcopy* and *mbxmove* copy (or move) messages from one mailbox (IMAP or local) to another existing mailbox (IMAP or local).

mbxcreat
: *mbxcreate* creates new mailboxes on the local system.

mbxcvt
: *mbxcvt* copies messages from a mailbox (IMAP or local) to a new mailbox in any desired format.

tmail
: Whereas *dmail* is the preferred agent to use in user scripts and procmail-type applications, *tmail* is the preferred server-called delivery agent. *tmail* is a privileged program and, consequently, is very paranoid about its arguments. *dmail* is an unprivileged program and is not as paranoid.

This example shows how the *ifrom* utility can be used to list the contents of a mailbox:

```
% ./ifrom -s -n -t -l ~msmith/mail/test
John Doe <johndoe@theman.hoopie.net>   1-Feb-2000  Re: Career Option
```

The *–s* and *–n* flags omit the *Status* field and *Message* number fields from the output. Omitting those fields increases the width of the *From* and *Subject* fields. The *–t* flag omits the time from the data field, and the *–l* field tells *ifrom* not to restrict the output to 80 columns. The folder, *~msmith/mail/test*, contains a single message from John Doe.

Here's an example that shows how to use the *mbxcopy* utility to copy an entire mailbox to a new mailbox:

```
% ./mbxcopy INBOX ~johndoe/mail/saved-mail
%Mailbox vulnerable - directory /var/spool/mail must have 1777 protection
/var/spool/mail/johndoe [38 message(s)] => /home/johndoe/mail/saved-mail
[Ok]
%
```