In this chapter:

- *IMAP Administration Tools*
- *Authentication Tools*
- *Monitoring and Testing Tools*
- *IMAP Clustering*
- *IMAP APIs*

18

IMAP Tools

A great advantage of embracing open source and open standard software is that all those nagging little—and some not-so-little—features that you would like to have, but don't, can be addressed by a handful of free software tools. Some of the tools are invasive, patching the source to your server to get their work done. Others rely only on the IMAP protocol to get their work done and would work equally well with closed source IMAP servers. All of them, however, could be used to provide a bridge between the capabilities of your server and the demands of your workload.

IMAP Administration Tools

Few things can positively affect your ability to manage your mail server like a good set of administration tools. We've found a few packages that could easily be the Leatherman Tool for your IMAP server—something you keep close by so you can use it on a moment's notice to perform some random act of maintenance. We've split these tools into two types: general server administration, and those that would be used in different load-balancing schemes.

General IMAP Server Administration

Those of our readers who are transitioning from POP3 mail servers to IMAP know that there's not a whole lot to using POP3 to administer user mailboxes. Nearly anything you might want to do in POP3 is easily accomplished by telnetting to the POP3 port and issuing commands manually. Because IMAP is a much richer protocol than POP3, there's much to be gained by having an intermediate piece of software to assist you in administering your IMAP server. That's the role played by the packages in this section.

IMAP-Admin

Author
Eric Estabrooks

URL
http://www.cpan.org/modules/by-module/IMAP/

Major features
Includes methods *create*, *delete*, *list*, *capability*, *get_quotaroot*, *get_quota*, *set_quota*, *get_acl*, *set_acl*, *delete_acl*.

Installation
Standard Perl module installation. See *http://www.cpan.org/modules/INSTALL.html*.

Documentation
Documentation is included within the module and is available at *http://theoryx5.uwinnipeg.ca/CPAN/by-name/IMAP-Admin.html*.

Status
Current version is 1.0.1 as of February 2000.

Licensing
Unknown (free).

IMAP-Admin is a Perl module for basic IMAP server administration, such as creating or deleting mailboxes and setting such mailbox options as ACLs and quotas. The module is RFC 2060–compliant, and thus works with both the Cyrus and UW servers. If you feel more comfortable supporting Perl than Tcl, IMAP-Admin makes a good substitute for *cyradm*, the Tcl-based system administration utility that is distributed with the Cyrus IMAP server. An example script for creating an IMAP account using the IMAP-Admin module is shown in Example 18-1.

Example 18-1. IMAP-Admin Example Script

```
#!/usr/local/bin/perl

use IMAP::Admin;

$imap = IMAP::Admin->new('Server' => 'imap.themullets.net',
                         'Port' => 143,
                         'Login' => 'admin',
                         'Password' => 'XXXXXXXX');

#  Create johndoe's mailbox
#
$rc = $imap->create("user.johndoe");
if ($rc != 0) {
  #print "$imap->{'Error'}\n";
}
```

Example 18-1. IMAP-Admin Example Script (continued)

```
#  Confirm johndoe's mailbox is there
#
@list =  $imap->list("user.johndoe");

#  Set johndoe's mailbox quota to 5 MB
#
$rc =   $imap->set_quota("user.johndoe", 5120);
@quota = $imap->get_quotaroot("user.johndoe");

#  Set the ACL on johndoe's mailbox
#
$rc =   $imap->set_acl("user.johndoe", "admin", "d", "mary", "lrs");
@acl =   $imap->get_acl("user.johndoe");

print "Created user: "; print join ' ', @list, "\n";
print "Quota: ";        print join ' ', @quota, "\n";
print "ACL: ";          print join ' ', @acl, "\n";

# Close the IMAP connection
#
$imap->close;
```

The output of the example is as follows:

```
Created user: user.johndoe
Quota: user.johndoe 0 5120
ACL: johndoe lrswipcda admin d mary lrs
```

PHP Cyrus-Tools

Authors

Didi Rieder (*adrieder@sbox.tu-graz.ac.at*) and Gernot Stocker

URL

ftp://ftp.hr.vc-graz.ac.at/cyrus-tools/

Major features

Performs most aspects of Cyrus IMAP server system administration on the Web.

Installation

Installation involves unpacking the distribution into your web server's document root directory, customizing a configuration file, and adding your IMAP admin username and password to the web server's authentication file.

Documentation

Documentation is contained in the *INSTALL* file included in the distribution.

Status

Current version is 1.0.1 as of February 2000.

Dependencies
PHP 3.0.11 or greater with IMAP support and a web server that supports PHP (such as Apache).

Licensing
Public domain.

PHP Cyrus Tools actually consist of two packages: *php-cyradm* and *php-mailsettings*. *php-cyradm* provides administration of a Cyrus IMAP server via a web interface. It includes the familiar *cyradm* functions that allow mailbox management (create, rename, delete mailboxes, modify ACLs) and also allows configuration of the Cyrus server by editing the *imapd.conf* file. *php-mailsettings* allows Cyrus IMAP users to set up mail forwarding and vacation auto-responses via the Web. You're advised to enable SSL on your web server to prevent your administrator password from crossing the network in cleartext.

Balancing Users Across Cyrus Partitions

move_imap_users

Author
J.T. Chiodi (*squeegy@squeegy.org*)

URL
http://www.squeegy.org/programs/move_imap_users.tar.gz

Major features
Balances Cyrus IMAP users between IMAP partitions or allows a set of users to be moved manually from one partition to another.

Installation
The package consists of two standalone scripts. No special installation is required.

Documentation
A *README* file is included with the package.

Status
Current version is 1.1 as of February 2000.

Licensing
GNU General Public License.

An absolute treasure for Cyrus administrators, the *move_imap_users* package contains a pair of tools for moving users between IMAP partitions on a Cyrus IMAP system.

balance.imap

balance.imap "load balances" a set of existing user mailboxes between two Cyrus IMAP partitions based on the available space on the partitions. It moves the mailboxes from the partition with the least available space to the partition with the most available space, until a determined threshold is reached.

fast.imap

fast.imap is used to move an existing mailbox or a set of user mailboxes from one partition to another, manually.

fast.imap takes three arguments: the source partition, the destination partition, and a range of usernames. Range can be either a letter (for example, "a" would be used to move all users whose usernames begin with the letter "a") or a range of letters ("[a-m]" would be used to move all users whose usernames begin with letters ranging from "a" to "m"). When using a range, be sure to escape the square brackets so that they won't be misinterpreted by the shell:

```
% cyradm -file fast.imap imap1 imap2 a
% cyradm -file fast.imap imap1 imap2 \[a-m\]
```

balance.imap could be run periodically out of *cron* to keep mailboxes evenly distributed across partitions.

Running Multiple Instances of Cyrus on a Single Machine

cyrus-imapd-configfiles

Author

David M. Zendzian (*dmz@dmzs.com*)

URL

http://www.dmzs.com/~dmz/projects/cyrus-config/cyrus-config.diff

Major features

Allows Cyrus to support more than one configuration file.

Installation

cyrus-imapd-configfiles is a patch that is applied to the Cyrus IMAP source.

Documentation

Documentation on setting up Cyrus to use more than one configuration file is located at *http://www.dmzs.com/~dmz/projects/cyrus-config/cyrus-virtual.txt*.

Status

Current version is 1.0 as of February 2000.

Dependencies
Depends on Cyrus 1.6.20.

Licensing
Unknown (free).

cyrus-imapd-configfiles is a patch for the Cyrus *imapd* that adds support for a command line flag that accepts an IMAP configuration file as an option, for example:

```
imapd -c /usr/local/etc/virtual/domain1.imapd.conf
imapd -c /usr/local/etc/virtual/domain2.imapd.conf
```

Being able to use multiple configuration files comes in handy when you need to run IMAP servers for more than one domain and want to keep the data for each domain separate, but on a single machine. There is some documentation on how to set up such a scheme at *http://www.dmzs.com/~dmz/projects/cyrus-config/cyrus-virtual.txt,* but it doesn't cover the bigger picture of how to map a group of users to a particular IMAP server.

Although that bigger issue is really a networking topic and beyond the scope of this book, here are a couple of quick sketches of how it can be done. Both ways require you to install a variation of *inetd* that allows binding a service to a particular IP address and allows you to restrict access to a service based on the originating IP address. There are several such programs freely available; the one used in our examples is *xinetd* (*http://www.xinetd.org/*).

Before you go any further, Figure 18-1 is a high-level illustration of the setup we're suggesting.

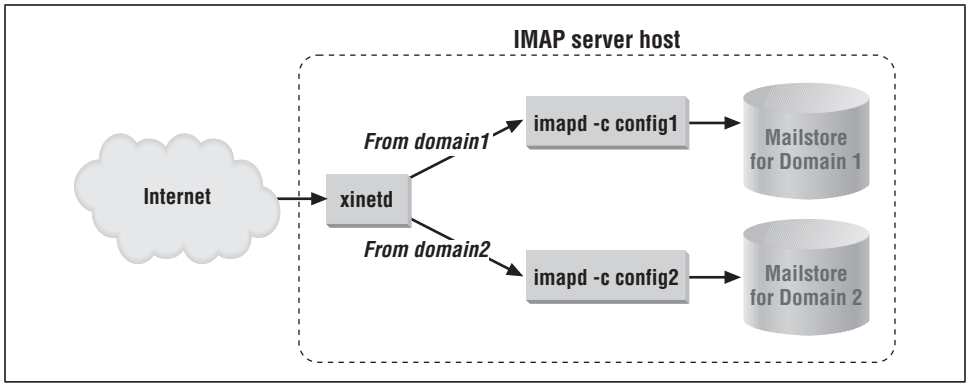

Figure 18-1. Multiple Cyrus configurations on a single host

IP traffic is filtered by *inetd* to one of two *imapd*'s based on the originating IP address. Each of the *imapd* servers operates on a separate mailstore.

Method 1: One IP, one port, multiple mailstores. Define two IMAP services in the *inetd* configuration file. Restrict one IMAP service to one domain, the other to another domain. The IP address and port are identical for the two servers. The only difference between them is that they handle IMAP requests only from a specific subnet and look at different configuration files. The configuration for *xinetd* would look something like Example 18-2.

Example 18-2. xinetd Configuration for Single Port/IP, Multiple Configurations

```
service imap
{
    socket_type     = stream
    protocol        = tcp
    wait            = no
    user            = cyrus
    only_from       = 129.120.1.0 localhost
    server          = /usr/cyrus/bin/imapd
    server_args     = -c /usr/local/etc/imap.subnet1.conf
}

service imap
{
    socket_type     = stream
    protocol        = tcp
    wait            = no
    user            = cyrus
    only_from       = 129.120.2.0 localhost
    server          = /usr/cyrus/bin/imapd
    server_args     = -c /usr/local/etc/imap.subnet2.conf
}
```

Method 2: Virtual IP per server. This method requires you to create a virtual interface for each domain for which you want to provide a separate IMAP service. Example 18-3 is the *xinetd* configuration entries for two virtual domains.

Example 18-3. xinetd Configuration for Two Virtual Domains

```
service imap
{
    socket_type     = stream
    protocol        = tcp
    wait            = no
    user            = cyrus
    only_from       = 129.120.1.0 localhost
    server          = /usr/cyrus/bin/imapd
    server_args     = -c /usr/local/etc/imap.subnet1.conf
    bind            = 129.120.1.1
}
```

Example 18-3. xinetd Configuration for Two Virtual Domains (continued)

```
service imap
{
    socket_type     = stream
    protocol        = tcp
    wait            = no
    user            = cyrus
    only_from       = 129.120.2.0 localhost
    server          = /usr/cyrus/bin/imapd
    server_args     = -c /usr/local/etc/imap.subnet2.conf
    bind            = 129.120.1.2
}
```

The abilities to balance your storage demands and customize your server-side configuraton for different classes of users are elements frequently demanded in large IMAP installations. Another element frequently stressed in large systems is user authentication. Both the performance and security of an IMAP server can be improved by moving beyond native Unix authentication.

Authentication Tools

It's becoming increasingly popular to opt out of using standard Unix authentication and to use database-based authentication instead. The advantages of alternative authentication methods are many. In many cases, you won't necessarily want your users to have regular Unix accounts on your IMAP server, and storing usernames and passwords in an alternate database allows you to get around setting up Unix accounts for everyone. Another advantage, in the case of the UW server, is that you can run your IMAP daemon as a non-privileged user. This section introduces tools that allow alternative authentication methods.

Authenticating Against a SQL Database (UW)

getpg/UW-IMAP

Author
: Alex Howansky (*alex@wankwood.com*)

URL
: *http://www.wankwood.com/getpg/uw-imap-latest.tar.gz*

Major features
: Allows UW IMAP to authenticate users against a PostgreSQL database.

Installation
: Involves applying a patch to the UW IMAP source distribution.

Documentation
Documentation is contained in the *README* file included in the *getpg/UW-IMAP* package.

Status
Current version is 0.54 as of February 2000.

Dependencies
PostgreSQL, UW IMAP. Tested using *qmail* as the MTA.

Licensing
Unknown (free).

The *getpg/UW-IMAP* patch utilizes the *getpg* functions to allow the UW IMAP server to authenticate users against a PostgreSQL database. *getpg*, included in the *getpg/UW-IMAP* package, consists of a pair of C functions that are drop-in replacements for the standard `getpwnam` and `getpwuid` functions, which do Unix authentication. The replacement functions `getpgnam` and `getpguid`, authenticate users against a PostgreSQL database. The replacement functions can be used not only to patch IMAP servers, but also to patch any application that authenticates against the Unix password file. More information on the *getpg* functions can be found at *http://www.wankwood.com/getpg/*.

Authenticating Against an SQL Database (Cyrus)

Authcheck

Author
Vladimir Ivaschenko (*vi@maks.net*)

URL
http://www.hazard.maks.net/cyrus/sql-auth.tar.gz

Major features
Tools for authenticating Cyrus users against a PostgreSQL database. Includes CGI administration tool.

Installation
Standalone Perl script. No special installation. Some IMAP configuration required.

Documentation
Documentation is contained in the *INSTALL* file included with the distribution.

Status
Current version is 0.01 as of February 2000.

Dependencies
Cyrus IMAP 1.6.20 or higher, Cyrus SASL built with *pwcheck* support, Perl modules IO::Socket, DBI, Unix::Syslog, Digest::MD5, and POSIX.

Licensing
GNU General Public License.

Authcheck is a daemon written in Perl that is used to authenticate IMAP users against an PostgreSQL database.

Authcheck requires Cyrus Version 1.6.20 or higher and Cyrus SASL Version 1.5.13 or higher built to include support for *pwcheck* authentication. Prebuilt RPMs (both source and binary) that include *pwcheck* support are available from *http://www.hazard.maks.net/cyrus/*.

Authcheck presumes that the server uses MD5 digests with its passwords. Two CGI scripts are included in the Authcheck distribution to simplify management of MD5 passwords. *imap-chpasswd.cgi.pl* is a CGI interface by which a user can change her MD5 password. *imapadmin.cgi.pl* is a CGI interface for administration of IMAP accounts that includes support for MD5 passwords. The administration CGI requires that the Cyrus admin user exist in the SQL database.

Configuring Authcheck is limited to the Authcheck daemon itself (*authcheck.pl*)—it should be modified to include your DBI DSN, database username, and password. If your Cyrus server does not use *pwcheck* authentication, then you'll have to modify */etc/imapd.conf* to include:

```
sasl_pwcheck_method: pwcheck
```

authcheck.pl should be run as the *cyrus* user, never as *root*.

cyrus-sasl-mysql patch

Author
David M. Zendzian (*dmz@dmzs.com*)

URL
http://www.dmzs.com/~dmz/projects/cyrus-sasl-mysql/

Major features
Allows Cyrus authentication via SASL against a MySQL database.

Installation
Involves patching and rebuilding Cyrus SASL.

Documentation
http://www.dmzs.com/~dmz/projects/cyrus-sasl-mysql/

Status
Current version is 0.8.0 as of March 2000.

Dependencies
Cyrus SASL 1.5.13 and Cyrus IMAP Version 1.6 or higher.

Licensing
Unknown (free).

cyrus-sasl-mysql is a patch for the Cyrus SASL library that allows the Cyrus IMAP server to authenticate IMAP users against a *mysql* database. SASL authentication is supported only in Cyrus IMAP Versions 1.6 and higher. Sites running Cyrus 1.5.x wishing to use an alternative authentication method should use the *pwcheck* variant (*pwcheck_mysql*) covered in this section.

Using the *cyrus-sasl-mysql* patch involves installing the patch and rebuilding Cyrus SASL. A rebuild of Cyrus IMAP is not required, because the SASL libraries are completely separate from Cyrus IMAP and have no dependencies on Cyrus. Once you've installed the patch and rebuilt SASL, there's no going back, so it's advisable to save a copy of your original source tree before installing the patch. The author of the patch intends to add a configure option that allows SASL to be built with *mysql* support enabled or disabled.

Configuration for *mysql* authentication exists entirely in */etc/imapd.conf*. The configuration options are described at *http://www.dmzs.com/~dmz/projects/cyrus-sasl-mysql/*.

pwcheck_mysql

Author
Aaron Newsome (*anewsome@anewsome.com*)

URL
http://www.mysql.com/Contrib/pwcheck_mysql-0.1.tar.gz

Major features
Allows IMAP users to be authenticated against a MySQL database.

Installation
Drop a source file into the Cyrus distribution, then configure and build Cyrus.

Documentation
Documentation is contained in the *README* file included with distribution.

Status
Current version is 0.1 as of March 2000.

Licensing
GNU General Public License.

With *pwcheck_mysql*, you can authenticate your IMAP users against a MySQL database. Installation is not complicated. From the top level of your Cyrus source tree, copy *pwcheck_mysql.c* into the *pwcheck/* subdirectory.

Configure Cyrus IMAP as follows:

```
% ./configure --with-pwcheck=mysql --with-login=unix_pwcheck
```

Edit *pwcheck/Makefile* so that your `CPPFLAGS`, `LIBS`, and `LDFLAGS` definitions look something like this (the exact definitions will vary slightly by operating system):

```
CPPFLAGS = -I/usr/local/include/mysql -I. -I$(srcdir) \
    -I$(srcdir)/../lib -I$(srcdir)/../et
LIBS = -lmysqlclient -ldb -lndbm -ldl
LDFLAGS = -g -L/usr/local/lib/mysql
```

Finally:

```
% make depend
% make
# make install
```

pwcheck_pgsql

Author
Maxim Gorkov

URL
ftp://ftp.kstu.edu.ru/pub/unix/cyrus/pwcheck_pgsql-0.1.tgz

Major features
Allows Cyrus IMAP users to authenticate against a PostgreSQL database.

Installation
Copy a C source file into Cyrus source tree, edit a Makefile, then configure and rebuild Cyrus.

Documentation
README file included in package.

Status
Current version is 0.1 as of February 2000.

Dependencies
PostgreSQL, Cyrus IMAP.

Licensing
Freeware.

pwcheck_pgsql is a replacement for the Cyrus IMAP server's external authentication daemon, *pwcheck*. *pwcheck_pgsql* authenticates users against a PostgreSQL database instead of the Unix shadow password file.

To use *pwcheck_pgsql*, copy *pwcheck_pgsql.c* into the *pwcheck* subdirectory of the Cyrus source tree. Then, configure Cyrus:

```
% ./configure -with-pwcheck=pgsql -with-login=unix_pwcheck
```

Open *pwcheck/Makefile* and edit the definitions of `LIBS` and `LDFLAGS` to include the PostgreSQL libraries (the exact definitions will vary slightly by operating system):

```
LIBS =  -lsocket -lnsl -lcrypt -lutil -lpg
LDFLAGS = -L/usr/local/lib -R/usr/local/lib -L/usr/local/pgsql/lib -L
```

And finally, from the top level of the Cyrus source directory:

```
% make
# make install
```

Authenticating Against an LDAP Directory (Cyrus)

pwcheck_ldap

Author
: Clayton Donley (*donley@cig.mot.com*)

URL
: *http://www.linc-dev.com/Files/pwcheck_ldap.c.txt*

Major features
: Replacement for Cyrus *pwcheck* that authenticates users against an LDAP directory.

Installation
: Involves copying a C source file into the Cyrus source tree, modifying some preprocessor commands and a Makefile, and reconfiguring/rebuilding Cyrus.

Documentation
: *http://www.linc-dev.com/Files/using-pwcheck.html*

Status
: Latest version is 1.01 as of February 2000.

Licensing
: Unknown (free).

pwcheck_ldap is a replacement for the Cyrus IMAP server's external authentication daemon, *pwcheck*. *pwcheck_ldap* authenticates users against an LDAP directory instead of the Unix shadow password file.

To install *pwcheck_ldap*, download *pwcheck_ldap.c* and copy it into the Cyrus IMAP source tree under the *pwcheck* subdirectory.

Edit the following two lines of *pwcheck_ldap.c* to suit your site:

```
#define MY_LDAP_SERVER "localhost"
#define MY_LDAP_BASEDN "o=Motorola, c=US"
```

Find the definition of `DEPLIBS` in *pwcheck/Makefile.in* and change it as follows:*

```
DEPLIBS = ../lib/libcyrus.a -lldap -llber @DEPLIBS
```

Finally, configure Cyrus and rebuild it:

```
% ./configure -with-pwcheck=ldap -with-login=unix_pwcheck
% make
# make install
```

Users with *uid* and *userPassword* attributes in the LDAP directory will be able to log in to the IMAP server.

Monitoring and Testing Tools

So, now that you've brought up your load-balanced, SQL-authenticated, awesome-performing IMAP server (at least you *think* it's awesome-performing), it would probably be good if you could back up your claims of near-permanent uptime and blazing performance with more than anecdotal tales of how well it works for you and the handful of users you've polled since the last upgrade. That's where monitoring and testing tools come in. They take system performance (which you and your users understand) and turn it into numbers (which "the suits" understand) and outage notifications (which you and your operations center understand). Refer to Chapter 16, *Server Performance Tuning*, for more on performance monitoring.

SMT

Authors
: Brian Hill and Joel Loudermilk

URL
: *http://www.doodlabs.com/smt/smt-1.05.tar.gz*

Major features
: Monitors IMAP, POP, SMTP, DNS, and HTTP services and notifies a user if a service is down.

Installation
: Typical GNU-type software installation.

---

* If you're using the Netscape LDAP libraries, use `-lldap10`.

Documentation
: General features are documented at *http://www.doodlabs.com/smt/*, and specifics are documented in manpages included in the SMT distribution.

Status
: Current version is 1.05 as of February 2000.

Licensing
: GNU General Public License.

SMT is a daemon that tests a service, such as IMAP or SMTP, by executing a test program. There are prebuilt tests included with SMT that monitor IMAP, POP3, SMTP, DNS, and HTTP. SMT can also run a user-defined test—anything goes, as long as it can be run from a shell prompt.

If the test program reports a failure, the SMT daemon executes a customized notification program to notify someone of the test failure. SMT maintains state information and makes notifications based on the change in state, not just on test failure. In other words, if a service is down, the notification will be sent out only once—the first time the service failure is detected.

SMT is light on resource utilization, using only the resources required for a test itself. Multiple tests can be executed concurrently.

To build and install SMT, unpack the distribution and from the distribution directory, type the commands:

```
% ./configure
% make
# make install
```

By default, the *smt* daemon is installed under */usr/local/bin/*, the tests under */usr/local/share/smt/*, and the manpages under */usr/local/man/*. You will have to create your own configuration file and notification script. Example 18-4 is an example configuration file for monitoring IMAP and SMTP.

Example 18-4. SMT Configuration File for IMAP and SMTP

```
logfile "/var/log/smt.log"
logdetail 4                          # Show test results in logs

test "IMAP server" {
     command "/usr/local/share/smt/imap imap.themullets.net"
     message "The IMAP server is %U"
     successcode 0
     timeout 25
}

test "SMTP" {
     command "/usr/local/share/smt/smtp mail.themullets.net"
     message "The SMTP server is %U"
```

Example 18-4. SMT Configuration File for IMAP and SMTP (continued)

```
     successcode 0
     timeout 25
}

group "Mail_Tests" {
     notification "/usr/local/etc/notify.pl dianna@pager.themullets.net"
     frequency 300
     test "SMTP"
     test "IMAP server"
}
```

Example 18-5 is the example notification script as defined in the configuration file in the previous example.

Example 18-5. SMT Notification Script

```
#!/usr/bin/perl
# Usage: notify.pl user@address messagefile
#
# messagefile is passed to the notify script by SMT as the last argument in the
# argument list.

$addressee     = "$ARGV[0]";
$messagefile   = "$ARGV[1]";

if (! $addressee && ! $messagefile) {

    print "Usage: notify user\@address messagefile\n";
    exit;
}

`/usr/bin/elm -s IMAPtest $addressee < $message`;
```

To start the SMT daemon, issue the command:

```
# smt /etc/smt.conf
```

tcpflow

Author

Jeremy Elson (*jelson@circlemud.org*)

URL

ftp://ftp.circlemud.org/pub/jelson/tcpflow/tcpflow-0.12.tar.gz

Major features

Records the conversational transactions that take place in TCP-based protocols in a format much easier to read than the default output of *snoop* or *tcpdump*. Useful only for TCP-based protocol debugging and analysis.

Installation
Standard GNU software installation.

Documentation
http://www.circlemud.org/~jelson/software/tcpflow/

Status
Current version is 0.12 as of March 2000.

Dependencies
LBL Packet Capture Library (*libpcap.a*), available from *ftp://ftp.ee.lbl.gov/libpcap.tar.Z*.

Licensing
GNU General Public License.

An excellent tool for observing, analyzing, and troubleshooting connections between IMAP client and server, *tcpflow* captures and stores data transmitted over a TCP connection. *tcpflow* is a more narrowly focused and approachable alternative to *tcpdump*. Both programs are based on the LBL Packet Capture Library, but *tcpflow* can store captured data in separate files named for the source, TCP port, and destination IP addresses. For example, the data transmitted over an IMAP connection from a client, 129.120.156.155 port 2156, to the server, 129.120.1.1, on port 143 would be stored in the file 129.120.156.155.02156-129.120.001.001.00143.

In Example 18-6, *tcpflow* is used to observe an IMAP client (Mulberry) log in to the IMAP server running on the local host, flag a message as "important," then log out. *diannapc.themullets.net* is the hostname of the client.

Example 18-6. tcpflow

```
localhost# tcpflow 'host diannapc.themullets.net and (port imap)'
129.120.210.004.00143-129.120.217.045.02516: * OK localhost IMAP4rev1 v12.250
server ready

129.120.217.045.02516-129.120.210.004.00143: A00001 CAPABILITY

129.120.210.004.00143-129.120.217.045.02516: * CAPABILITY IMAP4 IMAP4REV1
NAMESPACE IDLE SCAN SORT MAILBOX-REFERRALS LOGIN-REFERRALS AUTH=LOGIN
AUTH=ANONYMOUS THREAD=ORDEREDSUBJECT
A00001 OK Completed

129.120.217.045.02516-129.120.210.004.00143: A00002 LOGIN johndoe "XXXXXXXX"

129.120.210.004.00143-129.120.217.045.02516: A00002 OK Completed

Intermediate details omitted...

129.120.217.045.02529-129.120.210.004.00143: A00008 STORE 1 +FLAGS (\Flagged)
```

Example 18-6. tcpflow (continued)

```
129.120.210.004.00143-129.120.217.045.02529: * 1 FETCH (FLAGS (\Seen \Flagged))
A00008 OK Completed

129.120.217.045.02529-129.120.210.004.00143: A00009 LOGOUT

129.120.210.004.00143-129.120.217.045.02529: * BYE localhost IMAP4rev1 server
terminating connection
A00009 OK Completed
```

tm: A Stress Tester

Authors

Andrew Sutton (*Andrew.Sutton.1@ohio.edu*)
Todd Acheson (*acheson@ohio.edu*)

URL

http://redbud.cats.ohiou.edu/tm/tm.tar

Major features

Stress-tester for Cyrus IMAP server.

Installation

Unpack the *tar* archive and build.

Documentation

The technical description of *tm* is at *http://redbud.cats.ohiou.edu/tm/*.

Status

Current version is 1.1 as of March 2000.

Requirements

Requires the *pthread*[*] library.

Licensing

Free, can be redistributed for non-commercial use (*http://redbud.cats.ohiou.edu/tm/copyright.html*).

tm is a stress tester for the Cyrus IMAP server. It's designed to test the robustness of a Cyrus system by simulating over-exaggerated use. The results of a *tm* test can be used to predict how a Cyrus system will perform under heavy loads and to identify bottlenecks on the server. Currently, *tm* has been ported only to Sun OS, IRIX and DGUX.

---

[*] The GNU *pthread* library is available at *ftp://ftp.cis.ohio-state.edu/mirror/gnu/ptb/*.

Before attempting to run *tm* on a production system, be aware that the loads generated by *tm* are far in excess of normal system loads and could lead to downtime.

In order to simulate the effect of multiple users sharing resources, *tm* needs to perform IMAP operations asynchronously. To meet that requirement, *tm* is multithreaded and uses a separate thread for each set of write-read-verify operations. The number of threads is configured in the *tm.conf* file. A simple *tm.conf* file is shown in Example 18-7.

Example 18-7. tm.conf

```
# Example tm.conf file

{write1 read1}
{write2 read2}

# Spawn 10 noise threads
10 {noise}
```

The {writer reader} notation designates a writer-reader pair of scripts. The writer performs an operation, such as SETACL. The reader grabs the result of the operation by performing the companion operation (e.g., if the writer performs a SETACL operation, the reader would perform a GETACL). The noise thread calls another script, *noise*, which simulates work, such as users logging on to and out of the IMAP server, clients sending CAPABILITY or NOOP commands, or users fetching the contents of a mailbox.

The name of each member of the writer-reader pair corresponds to the actual name of the script that contains the IMAP commands to be executed. In the example, there are four scripts: *write1*, *write2*, *read1*, and *read2*. The *write1* and *read1* scripts are shown in Example 18-8 and Example 18-9.

Example 18-8. write1 (Example Writer Script)

```
open imap imap.themullets.net
command login testuser xxxxxxxx
loop 20
command setacl mailbox inbox johndoe lprs
command setacl mailbox inbox johndoe lwci
end
logout
```

Example 18-9. read1 (Example Reader Script)

```
open imap imap.themullets.net
command login testuser xxxxxxxx
loop 20
command getacl mailbox inbox
command getacl mailbox inbox
end
logout
```

The *tm* program compares the results of the writer and reader and logs the results of the comparison to *tm.log*. The IMAP commands and their output are written to the debug log file, *out.log*.* The test is complete when all scripts have finished running and the last operations have been logged.

Of course, having the ability to create system load is more meaningful if you also have the capacity to engineer ways to meet that load. One way to address that need is through clustering your IMAP servers.

IMAP Clustering

Unfortunately, there is no comprehensive open source package that provides IMAP server clustering. We've avoided covering commercial software tools in this chapter, but in light of the lack of free clustering software, we'll break the trend and mention one commercial product and one freely available package.

BLUETAIL Mail Robustifier

Author
: Bluetail AB

URL
: *http://www.bluetail.com/products/*

Major features
: Adds load balancing, scalability, and fault tolerance to IMAP services.

Installation
: *http://www.bluetail.com/big/common/bmr11.pdf*

Documentation
: Available at Bluetail AB's site: whitepaper (*http://www.bluetail.com/big/common/wp11.pdf*), overview (*http://www.bluetail.com/big/common/wpm.pdf*), and user manuals (*http://www.bluetail.com/big/common/bmr11.pdf*).

Status
: Current version is 1.1 as of February 2000.

* Use *tm*'s *–n* option to turn off debug logging to *out.log*.

Licensing
Commercial (free 30-day trial available).

BLUETAIL Mail Robustifier is a software proxy system that adds load balancing, scalability, and fault tolerance to IMAP, POP, and SMTP services running on Unix platforms. The supported platforms are Linux, Solaris, FreeBSD, and BSDi.

The software runs on a pair of dedicated proxy servers that sit between the IMAP server farm and the IMAP clients and load-balance traffic across the servers in the farm. If a proxy server dies, BLUETAIL fails over to the other node.

BLUETAIL adds scalability by allowing additional IMAP server machines to be added transparently while IMAP services are online. Likewise, machines can be removed or upgraded while IMAP is up. If an IMAP server dies, its traffic can either be routed to another server or rejected, depending on the configuration. BLUETAIL can be configured dynamically and the software can be upgraded while the system is running.

An overload control mechanism can be configured to kick in when traffic reaches a certain threshold. Once that threshold is met, any further connections will be refused. The overload control mechanism can also be used to prioritize groups of users, i.e., give premium users priority over normal users when the system begins to reach its threshold.

Additional features of BLUETAIL include RFC 2505 SPAM filtering, GUI and command-line interfaces, and monitoring and statistics gathering.

Of particular interest to Cyrus sites, BLUETAIL is capable of handling both infrastructures that share a mailstore between IMAP servers and infrastructures that have a separate mailstore per server. If the mailstore is shared, Robustifier balances the traffic between the servers. In cases where each server has a dedicated mailstore, BLUETAIL routes traffic to the appropriate server, either per-user or per-domain.

Cyrus IMAP Aggregator

Author
Project Cyrus

URL
http://asg.web.cmu.edu/cyrus/ag.html

Major features
Divides IMAP mailstore load across an arbitrary number of servers. Provides near high-availability for IMAP processing (see below).

Installation
Not released when we went to press.

Documentation
http://asg.web.cmu.edu/cyrus/ag.html

Status
Unreleased and pre-alpha as of June 2000.

Licensing
Freely available.

The Cyrus IMAP Aggregator is very early into its life cycle. As we go to press, it's pre-alpha and not even released to the public. Upon release, however, the IMAP Aggregator promises to provide robust multitier reliability for IMAP systems.

An IMAP Aggregator consists of two rows of IMAP servers. A row of frontend servers take all the incoming IMAP connections from clients and act solely as proxies between the end users and the backend servers. A row of backend servers provides direct access to the mailstore. Each frontend server provides duplicate proxy services. Lose one or more of your front-row servers, and as long as some still remain, you're still completely able to provide IMAP services. Each backend server provides mailstore services for a given percentage of your users. ACAP or LDAP help the frontend servers (which act as proxies to the back servers) locate the mailbox for a particular user.

To be a true high-availability system, rather than simple multiplexing, the IMAP Aggregator would need to provide duplicate mailstore services on each of the backend servers.

IMAP APIs

You may be the type that is only be satisfied with IMAP utilities that you write yourself. Or perhaps you've got a particular need that isn't adequately met by the other packages described here. In either case, here are some IMAP APIs that might make your life a bit easier and your development time a bit shorter.

C-Client API

Author
Mark Crispin (*mrc@cac.washington.edu*)

URL
ftp://ftp.cac.washington.edu/imap/imap.tar.Z

Major features
IMAP API and supporting libraries for messaging applications bundled with the UW IMAP server distribution.

Installation
No special installation required (*make <platform>*).

Documentation
Documentation on C-Client is included in the UW IMAP distribution, in the file *docs/internal.txt*.

Status
Current version is 4.7a as of March 2000.

Licensing
Free, can be redistributed, copyright restricted (see comments in source files for details).

C-Client is the granddaddy of IMAP APIs. Written by Mark Crispin, the IMAP originator, C-Client has been used at the core of many IMAP servers and clients beyond just PINE and UW-IMAP. It's probably fair to say that C-Client is a reference API for most IMAP protocol features and, as such, is more appropriate for full-scale projects than quick-and-dirty utilities.

C-Client may be found in the *src/c-client* directory of the UW-IMAP distribution, and extensive, well-written, detailed documentation on C-Client may be found in *docs/internal.txt*.

Perl APIs

Mail-IMAPClient

Author
David J. Kernen (*kernen@erols.com*)

URL
http://www.perl.com/CPAN/authors/id/D/DJ/DJKERNEN/Mail-IMAPClient-1.08.tar.gz

Major features
Perl API that simplifies the conversation between a Perl script and an IMAP server.

Installation
Standard Perl module installation. See *http://www.cpan.org/modules/INSTALL.html*.

Documentation
http://search.cpan.org/doc/DJKERNEN/Mail-IMAPClient-1.08/README

Status
Current version is 1.08 as of March 2000.

Dependencies

Requires Perl module IO::Socket.

Licensing

GNU General Public License.

Mail-IMAPClient (shown in Example 18-10) is an RFC 2060–compliant API that simplifies the interaction between a Perl script and an IMAP server. Virtually all client commands defined in RFC 2060 are supported. Mail-IMAPClient has been tested successfully with both the Cyrus and UW IMAP servers.

To verify that your system has IO::Socket installed, enter on the command line:

```
% perl -e 'use Socket; use IO::Socket;'
```

The module source distribution includes examples, including *imap-to-inbox.pl*, which reads a user's IMAP folders and converts them to *mbox* format. *imap-to-inbox.pl* can be used as an intermediate step converting from a proprietary IMAP server to one of the public-domain servers.

Example 18-10. Mail-IMAPClient

```
#!/usr/local/bin/perl

use Mail::IMAPClient;
$| = 1;

$server = "imap.themullets.net";
$port   = 143;
$user   = "johndoe";
$password = "XXXXXXXX";

&connect;
&count_messages;

sub connect {

    $imap = Mail::IMAPClient->new(
        Server      => "$server",
        User        => "$user",
        Password    => "$password",
        Port        => "$port",
    )
    || die ("Could not connect to $server:$port: $! $?");
}

sub count_messages {

    my (@folders, $folder, $count);

    @folders = $imap->folders;
```

Example 18-10. Mail-IMAPClient (continued)

```
    push(@folders, "INBOX");

    foreach $folder (@folders) {

        $count = $imap->message_count($folder);

        if (! $count) {
                print "$folder is empty.\n";
                next;
        }
       print "$count messages in $folder.\n";
    }
}
```

Net-IMAP-Simple

Author
Joao Fonseca (*joao_g_fonseca@yahoo.com*)

URL
http://www.perl.com/CPAN/authors/id/J/JP/JPAF/Net-IMAP-Simple-0.93.tar.gz

Major features
Perl IMAP API, limited to a subset of IMAP commands.

Installation
Standard Perl module installation. See *http://www.cpan.org/modules/INSTALL.html*.

Documentation
http://search.cpan.org/doc/JPAF/Net-IMAP-Simple-0.93/Simple.pm.

Status
Current version is 0.93 as of February, 2000.

Licensing
Unstated (free).

This module is a simple way to access IMAP accounts. Net-IMAP-Simple is limited to the following commands:

login	*create_mailbox*
select	*rename_mailbox*
seen	*delete_mailbox*
get	*copy*
getfh	*quit*
mailboxes	

Example 18-11 is an example script using Net-IMAP-Simple.

Example 18-11. Net-IMAP-Simple Example

```
#!/usr/local/bin/perl

use Net::IMAP::Simple;

# Open a connection to the IMAP server
#
$server = new Net::IMAP::Simple( 'imap.unt.edu' );

# Log in to the server
#
$server->login( 'johndoe', 'XXXXXXXX' );


# Get a list of all folders
#
@folders = $server->mailboxes();

foreach $folder (@folders) {

    $number_of_messages = $server->select( $folder );

    if ($number_of_messages) {
        print "There are $number_of_messages in $folder.\n";
    } else {
        print "$folder is empty.\n";
    }
}

# close the connection
$server->quit();
```

NetxAP

Author
: Kevin Johnson (*kjj@pobox.com*)

URL
: *http://www.perl.com/CPAN/authors/id/KJOHNSON/NetxAP-0.02.tar.gz*

Major features
: Perl API featuring full set of RFC 2060 IMAP commands along with commands from many of the IMAP extensions (such as namespace, ACL, and quota).

Installation
: Standard Perl module installation. See *http://www.cpan.org/modules/INSTALL.html*.

Documentation
: *http://search.cpan.org/doc/KJOHNSON/NetxAP-0.02/Net/IMAP.pm*

Status
Current version is 0.02 as of February 2000.

Dependencies
MIME::Base64 and Digest-MD5 modules.

Licensing
Freeware.

Net::IMAP is a Perl module included in the NetxAP package. It consists of a complete set of core RFC 2060 protocol commands, with a few extras thrown in. The list is too extensive to list here, but is documented at *http://search.cpan.org/doc/KJOHNSON/NetxAP-0.02/Net/IMAP.pm*. The NetxAP modules are still in alpha release. Although they look very promising, the author of the modules cautions against using them in production environments. Two examples that use Net::IMAP are included in the NetxAP distribution.

PHP

Author
Many contributors (*http://www.php.net/credits.php3*)

URL
http://www.php.net/

Major features
PHP IMAP API based on the C-Client library.

Installation
Standard GNU-type software installation.

Documentation
http://www.php.net/manual/ref.imap.php3

Status
Current version is 4.0 as of March 2000.

Dependencies
Requires C-Client library.

Licensing
GNU General Public License (*http://www.php.net/license.html*).

PHP includes a built-in IMAP API that's a port of the C-Client library. To use the IMAP functions, you must first install the C-Client library (refer to the section on C-Client earlier in this chapter for download information). Build the C-Client library, then copy *c-client/c-client.a* to */usr/local/lib/* (or wherever you keep your locally installed libraries). Copy *c-client/rfc822.h*, *c-client/mail.h* and *c-client/linkage.h* to */usr/local/include* (or another directory in your include path). Once

the C-Client library is installed, compile PHP with the *--with-imap* option. Complete documentation on the API functions is available in the PHP manual at *http://www.php.net/manual/ref.imap.php3*.

JavaMail

Author

Sun Microsystems

URL

http://www.javasoft.com/products/javamail/index.html

Major features

Comprehensive API for developing mail clients in Java; supports SMTP, POP, and other protocols in addition to IMAP.

Installation

Standard Java installation.

Documentation

http://www.javasoft.com/products/JavaMail-1.1.pdf (documentation also available online)

Status

Current version is 1.1.3 (February 2000); 1.2 spec draft available.

Dependencies

Requires JavaBeans Activation Framework (*javax.activation*).

Licensing

SCSL (Sun Community Source License).