AfNOG 2007 Exim Practical

Objectives

Part 1 is building Exim from source, and installing it.

- Download Exim source and documentation
- Unpack the source and documentation
- Build Exim from the generic distribution
- Install Exim
- Replace Sendmail with Exim

Part 2 is running basic tests. You don't need to modify the Exim configuration to do this.

- Test a standard installation and default configuration
- Inspect and manage the mail queue
- Check relay control
- Process log data

Part 3 involves some simple modification of the runtime configuration.

- Modify the runtime configuration to send undeliverable mail to postmaster
- Add some simple virtual domains

Part 4 sets up your host as a mail relay

- Allow relaying from another host
- Allow relaying to another domain
- Configure SMTP authentication

Part 5 more advanced things to try for those who have time.

- Configure TLS
- Demonstrate retry mechanisms
- Configure and test address rewriting
- Interface Exim to SpamAssassin
- Interface Exim to ClamAV

☞ This sign is used in the text to mark an action that you need to take if you want to do the things that are suggested in this practical.

Common mistakes

In past workshops, these are the most common mistakes that have been made:

- *Doing everything as root.* You only need to be *root* to install Exim and change its configuration. Otherwise, you should do everything (including building Exim) under your normal

login. In the sample commands, the command prompt is shown as **#** for commands that must be run as *root*, and **$** otherwise.

In particular, running email tests as *root* is a bad idea, because *root* has privileges. You want to test that Exim is working when an ordinary, unprivileged user calls it.

You are going to be wanting to switch backwards and forwards between *root* and your own login quite a lot. A convenient way of doing this is to run X windows, and set up two *xterm* windows side by side, one under your login and the other as *root*.

- *Forgetting the dot that terminates a message.* When you type a message directly into Exim, it needs a line containing just a dot to terminate it. Until you type that line, all input is taken as part of the message.
- *Setting PATH incorrectly.* Whenever you change your PATH setting, be sure to check what you have typed carefully before pressing RETURN. If you mess up with PATH, you'll find that lots of commands "disappear".
- *Adding dots to email domains.* You should now have got used to inserting trailing dots in fully qualified domains in DNS zones. Unfortunately, in email configurations, trailing dots are *not* used and will cause problems if you use them.

1. Installing Exim

We are going to install Exim from the generic source distribution. This way of doing it allows you to make your own choices at build time. It also shows you that installing software from source can be quite easy.

You do not need to be *root* to build Exim, and it is best practice if you are not. However, you do need to be *root* to install Exim.

Reminder: In the sample commands below, the command prompt is shown as **#** for commands that must be run as *root*, and **$** otherwise.

1.1 Preliminary preparation

☞ If Sendmail is running, kill it off. Use this command:

```
# /etc/rc.d/sendmail stop
```

You must also stop Sendmail from restarting at the next boot. Do this by editing the file */etc/rc.conf* and adding this line at the end:

```
sendmail_enable="NONE"
```

Use vi to create the file */etc/periodic.conf* file and add these lines:

```
daily_status_include_submit_mailq="NO"
daily_clean_hoststat_enable="NO"
```

This disables the daily housekeeping commands that Sendmail uses.

❄ **Note:** On other operating systems, the way you kill off **sendmail** and stop it from restarting may be different.

You should previously have created a user and a group called *exim*. These will be used for running Exim when it does not need to be *root*. If you have not done this yet, you must do it before trying to build Exim.

You should also have arranged for your personal (non-root) account to be in the *exim* group so that you can be an administrator for Exim. Check by running

```
$ groups
```

If the *exim* group is not listed, edit */etc/group* (as *root*). Find the line that contains

```
exim:*:90:
```

and add your login name to the end of it. You will then need to login again for this change to take effect.

☞ Ensure that the */var/mail* directory has the 'sticky' bit set on it. If you don't understand this, don't worry, just do it:

```
# chmod 1777 /var/mail
```

This is so that the default Exim configuration will work without having to be changed.

1.2 Download the source and documentation

☞ As *root*, make a directory in which to build Exim, say */usr/exim*, and give yourself access to it:

```
# mkdir /usr/exim
# chown yourname:yourname /usr/exim
```

You can now fetch and build Exim from your own account (not *root*).

☞ Fetch the source of Exim and the HTML documentation and a test utility from the workshop ftp site:

```
$ cd /usr/exim
$ ftp ftp.e1.ws.afnog.org
```

Log in as *anonymous*.

```
ftp> cd /pub/sources
ftp> get exim-4.67.tar.gz
ftp> get exim-html-4.67.tar.gz
ftp> get smtpc
ftp> bye
```

☞ Unzip and untar the source and the HTML documentation:

```
$ cd /usr/exim
$ gunzip exim-4.67.tar.gz
$ tar -xf exim-4.67.tar
$ gunzip exim-html-4.67.tar.gz
$ tar -xf exim-html-4.67.tar
```

1.3 Check the documentation

Before moving on, make sure you can access the Exim documentation, so that you can look things up if you have problems. If you have a web browser running, point it at:

```
file:///usr/exim/exim-html-4.67/doc/html/index.html
```

The FAQ has a useful keyword-in-context index to help you find your way about. There should also be a file called */usr/exim/exim-4.67/doc/spec.txt*. It contains a copy of the manual in ASCII format which can be searched with a text editor.

1.4 Building Exim

☞ Now we can get ready to build Exim. You have to set up two configuration files. Go into the toplevel source directory:

```
$ cd /usr/exim/exim-4.67
```

Copy the file *src/EDITME* to *Local/Makefile* and *exim_monitor/EDITME* to *Local/eximon.conf*. You then have to edit *Local/Makefile*, following the instructions inside it:

```
$ cp src/EDITME Local/Makefile
$ cp exim_monitor/EDITME Local/eximon.conf
$ vi Local/Makefile
```

There are lots of instructions inside the file, but you do not have to make many changes. You can leave almost all of the settings at the defaults, but you will need to set `EXIM_USER` to the user for

running Exim. You also need to request 'maildir' support, TLS support, RADIUS support and content scanning support for use later in the workshop. Find the lines that contain these capitalized keywords and change them to be like this:

```
EXIM_USER=exim
SUPPORT_MAILDIR=yes
WITH_CONTENT_SCAN=yes
AUTH_PLAINTEXT=yes
SUPPORT_TLS=yes
TLS_LIBS=-lssl -lcrypto
RADIUS_CONFIG_FILE=/etc/radius.conf
RADIUS_LIB_TYPE=RADLIB
```

You must also add the following line after the RADIUS_LIB_TYPE line:

```
EXTRALIBS_EXIM=-lradius
```

(Do not do this at the workshop.) When you build Exim on your own hosts back home, you may want to change `BIN_DIRECTORY` and `CONFIGURE_FILE` from their default values of */usr/exim/bin* and */usr/exim/configure*. For example, these settings match what the FreeBSD port uses:

```
BIN_DIRECTORY=/usr/local/sbin
CONFIGURE_FILE=/usr/local/etc/exim/configure
```

However, for this exercise, we assume that you didn't change the default values.

You do not need to edit *Local/eximon.conf* because the default settings will be OK.

☞ Now you can run *make*:

```
$ make
```

You should see a lot of output while Exim builds, ending with the line:

```
>>> exim binary built
```

When you see that line, you have successfully built Exim. Easy, wasn't it?

1.5 Installing Exim

☞ You need to be *root* to install Exim:

```
# cd /usr/exim/exim-4.67
# make install
```

You should end up with the Exim binaries in */usr/exim/bin/* and a default configuration file in */usr/exim/configure*.

☞ Test that Exim has been installed by running:

```
$ /usr/exim/bin/exim -bV
```

which should tell you Exim's version number and some other information about which features are included.

1.6 Replace Sendmail with Exim

All the MUAs call */usr/sbin/sendmail* to pass messages to the MTA. We want them to call Exim instead of Sendmail.

☞ On FreeBSD, there is a file called */etc/mail/mailer.conf* that selects the MTA. To change the MTA, you must edit this file (as *root*):

```
# vi /etc/mail/mailer.conf
```

Comment the existing lines, and insert these new lines:

```
sendmail        /usr/exim/bin/exim
send-mail       /usr/exim/bin/exim
mailq           /usr/exim/bin/exim -bp
newaliases      /usr/bin/true
```

☞ Now try that basic test again, but this time using the standard path name:

```
$ /usr/sbin/sendmail -bV
```

You should get the same output as before, which shows that Exim is now being used instead of Sendmail.

If you are doing a real installation on a live system, you might want to work on the configuration and do lots of testing before removing Sendmail and replacing it with Exim.

❄ (Do not do this at the workshop.) On operating systems that don't have */etc/mailer.conf*, you can just make */usr/sbin/sendmail* into a symbolic link that points directly to the Exim binary. First, however, you must move the old Sendmail binary out of the way:

```
# mv /usr/sbin/sendmail /usr/sbin/sendmail.sendmail
# ln -s /usr/exim/bin/exim /usr/sbin/sendmail
```

---

2. Testing Exim

2.1 Test the standard installation and configuration

Make sure you substitute a real local user name for *localuser* in what follows. Remember, you should *not* be *root* when running these tests.

☞ To save typing, adjust your `PATH` variable so that the command *exim* can be used to run the Exim binary. **Take great care** when you do this, because messing up your PATH will make many commands "vanish". The way to adjust `PATH` depends on which shell is running. If you are using a Bourne-compatible shell such as *bash* or *ksh*, type this command exactly, taking care with the colon and dollar in the middle:

```
$ export PATH=/usr/exim/bin:$PATH
```

If you are using *csh* or *tcsh*, use this command, noting that there is a space (not an =) after `PATH`:

```
$ setenv PATH /usr/exim/bin:$PATH
```

☞ First, check what Exim will do with a local address:

```
$ exim -bt localuser
```

This tests the delivery routing for a local account. See what output you get.

☞ Try with a non-existent local user and see what happens:

```
$ exim -bt junkjunkjunk
```

☞ Try something that is in */etc/aliases*:

```
$ exim -bt postmaster
```

Exim will not normally deliver mail to a *root* mailbox (for security reasons) so what people usually do is to make *root* an alias for the sysadmin. In FreeBSD, all the default aliases point to *root*. Therefore, you need to add a new alias to */etc/aliases*. Add this line (as *root*):

```
root: yourname
```

Now try this again:

```
$ exim -bt postmaster
```

☞ Now we are going to try a real local delivery. You can pass a message directly to Exim without using an MUA:

```
$ exim -v -odf localuser
This is a test message.
.
```

Note: the message is terminated by a line that just contains a dot. Be sure to type it! (Alternatively, you can send "end of file" by pressing CTRL-D.)

The `-v` option turns on user verification output, which shows you copies of Exim's log lines.

The `-odf` option requests 'foreground' delivery, which means that the ***exim*** command won't return until the delivery is complete. (This avoids your shell prompt getting mixed up with Exim's output.)

☞ Check what is in Exim's logs:

```
$ cat /var/spool/exim/log/mainlog
$ cat /var/spool/exim/log/paniclog
```

If you get a *permission denied* error, it is probably because you have not put yourself in the *exim* group, or not logged in again after editing */etc/group*.

If the delivery succeeded, you should see two lines in the main log, one containing <= for the message arriving, and one containing => for the delivery.

The panic log should normally be empty, and if nothing has ever been written to it, it will not even exist, so you may get a *No such file or directory* error. **Tip**: On a live system it is helpful to set up a ***cron*** job that mails you a warning if it ever finds a non-empty panic log.

☞ Now check the contents of the local user's mailbox:

```
$ ls -l /var/mail/localuser
$ cat /var/mail/localuser
```

If the delivery didn't succeed, you need to find out why. If the information in the log doesn't help, you can try the delivery again, with debugging turned on:

```
$ exim -d -odf localuser
<there will be output from Exim here>
This is another test message.
.
```

The `-d` option turns on debugging, which gives a lot more information than `-v`. You need to be an Exim administrator to use `-d`. If you get a *Permission denied* error, check that you are a member of the Exim group.

☞ If you are logged on as *localuser*, you can use the *mail* command to read the mail in the usual way. You could also try sending a message from the *mail* command.

The next thing is to test whether Exim can send to a remote host. The speed of this may vary, depending on the state of the network connection. In what follows, replace *user@remote.host* with your home email address.

☞ First, check that Exim can route to the address:

```
$ exim -bt user@remote.host
```

☞ Now send a message to the remote address:

```
$ exim -v -odf user@remote.host
This is a test message.
.
```

This time, the -v option causes Exim to display the SMTP dialogue as well as the log lines. If you can, check that the message arrived safely. If there are problems, see if you can figure out what went wrong and why.

☞ You won't be able to receive messages from a remote host until you start the Exim daemon:

```
$ /usr/exim/bin/exim -bd -q20m
```

The -bd option causes the daemon to listen for incoming SMTP calls, and the -q20m option causes it to start a queue runner process every 20 minutes.

☞ We also want the daemon to start automatically on a reboot. The */etc/rc.local* file is a script that is run at boot time. Add this line (the same as the command you have just run) to the file */etc/rc.local*:

```
/usr/exim/bin/exim -bd -q20m
```

The file */etc/rc.local* may not exist. If it does not exist, you should create it.

Next time you reboot, check that Exim has started.

☞ Use telnet to check that the daemon is accepting SMTP calls:

```
$ telnet localhost 25
```

You should see an Exim greeting message. Use QUIT to exit.

☞ Now check that a remote host can send a message to your host, and see how Exim logs what happens. If that succeeds, you have a working basic installation correctly installed.

☞ Try sending to an invalid address from a remote host, and see what error message you get, and how Exim logs this case. Look in both *mainlog* and *rejectlog*.

2.2 Starting the Exim Monitor

You need to have an X-windows session running before you start the monitor, and you need to have logged in with your own id, *not* as root. If you try to start the monitor as root you may have permission problems accessing the display.

☞ Start the monitor:

```
$ /usr/exim/bin/eximon
```

The upper window shows a 'tail' of the main log; the lower window shows the messages that are waiting in the queue. Expect both to be empty to start with. Send a few messages and watch what the monitor displays.

The lower window is updated automatically only every 5 minutes, because it is expensive to scan a large queue. The "Update" button can be used to force an update.

The "Size" button in the upper window reduces the whole monitor window to show just the queue length stripchart, which is convenient for keeping in a corner of your screen.

2.3 Queue management tests

There are several command line options (and equivalent menu items in the monitor) for doing things to messages.

☞ To put a message on the queue without its being delivered, run

```
$ exim -odq address1 address2 ...
Test message.
.
```

The message stays on the queue until a queue runner process notices it.

☞ List the messages on the queue:

```
$ exim -bp
```

☞ Do a manual queue run, with minimal verification output:

```
$ exim -v -q
```

(Without `-v` you won't see any output at all on the terminal, but there will be entries in the log.)

2.4 Checking relay control

☞ To demonstrate that Exim will relay by default via the loopback interface, try the following sequence of SMTP commands. Wait for Exim to respond to each command before typing the next one. Substitute the number of your host for *nn*:

```
$ telnet 127.0.0.1 25
ehlo localhost
mail from:<localuser@pcnn.e1.ws.afnog.org>
rcpt to:<localuser@pcnn.e1.ws.afnog.org>
rcpt to:<user@some.remote.domain>
```

You should get an OK response to all the SMTP commands. Type 'quit' to end the SMTP session without actually sending a message.

☞ Now try the same thing, but use your host's IP address instead of 127.0.0.1.

```
$ telnet xx.xx.xx.xx 25
ehlo localhost
mail from:<localuser@pcnn.e1.ws.afnog.org>
rcpt to:<localuser@pcnn.e1.ws.afnog.org>
rcpt to:<user@some.remote.domain>
```

In this case, you should get the error message

```
550 relay not permitted
```

for the second `RCPT` command, which is the one that is trying to relay. The first `RCPT` command should be accepted, because it specifies a local delivery. You could also try telnetting from an external host and running the same check.

2.5 Processing log data

☞ Run *exigrep* to extract all information about a certain message, or a certain user's messages, or messages for a certain domain. For example:

```
$ exigrep localuser /var/spool/exim/log/mainlog
```

That extracts all the log information for all messages that have any log line containing '*localuser*'. It's a Perl pattern match, so you can use Perl regular expressions.

☞ To extract simple statistics from a log, run

```
$ eximstats /var/spool/exim/log/mainlog | more
```

There are options for selecting which bits you don't want. Details are in the manual. If you have time, experiment with the options for outputting the statistics as HTML.

3. Changing the configuration

To change Exim's runtime configuration, you must edit */usr/exim/configure* and then HUP the Exim daemon (as *root*). The daemon stores its process id (pid) in a file, in order to make this easy. You can find out the daemon's process id by running:

```
$ cat /var/spool/exim/exim-daemon.pid
```

You can use the contents of this file as part of a command to restart the Exim daemon. If you are using the *bash* shell, you can use this command:

```
# kill -HUP $(cat /var/spool/exim/exim-daemon.pid)
```

The shell first runs the command inside `$(...)`, and then uses its output as part of the main command line.

If you are using the C-shell (*csh*) rather than *bash*, you cannot use the `$(...)` construction. Instead, you must put the nested command inside "backticks" (grave accent characters).

You can confirm that the daemon has restarted by checking the main Exim log.

You are going to be restarting the Exim daemon a lot, so make yourself a script to save typing. Use *vi* to create a file called */usr/local/bin/hupexim*, containing these lines:

```
#! /usr/local/bin/bash
kill -HUP $(cat /var/spool/exim/exim-daemon.pid)
```

Note that the # character in the first line is part of the file (it's not a prompt). Now make the new file into an executable script:

```
# chmod a+x /usr/local/bin/hupexim
```

If you are using the C-shell (*csh*) rather than *bash*, after creating */usr/local/bin/hupexim* you must run this command:

```
# rehash
```

This causes the internal hash table of the contents of the directories in the PATH variable to be recomputed. This is not necessary if you are using *bash*.

Once you have created */usr/local/bin/hupexim* you can restart Exim just by running:

```
# hupexim
```

The following sections contain some suggestions for configuration modifications that you can try, just to get a feel for how the configuration file works. You do not have to stick rigidly to these examples; use different domain names or user names if you want to.

3.1 Adding more local domains

As *root*, edit the configuration file (*/usr/exim/configure*), and change the **local_domains** setting so that it looks like this:

```
domainlist local_domains = @ : subdomain.afnogws.gh
```

where *subdomain* is the domain you chose during the DNS exercise. Remember to HUP the daemon afterwards. Now you have a new local domain. Try sending it some mail:

$ `mail` *yourname*`@subdomain.afnogws.gh`

Check that it arrives in your mailbox.

Note: The domains that we are adding now can only be used from your own host, because there are no DNS records for them. When you are adding domains to a production host, you must of course also add MX records for them.

If you want to add a lot of domains, or if you want to keep changing them, it is easier to keep the list of domains in a file instead of in the Exim configuration. (You can also keep them in several different kinds of database, such as LDAP or MySQL, but we don't cover that in this workshop.) We are now going to add some domains like this, and then make them into *virtual domains*.

As *root*, use *vi* to create a file called */usr/exim/vdomains* that contains a list of domains (as many as you like):

```
vdom1.afnog.org
vdom2.afnog.org
...
```

As *root*, edit */usr/exim/configure* to change the local domains setting:

```
domainlist local_domains = @ : subdomain.afnogws.gh : \
                           lsearch;/usr/exim/vdomains
```

Note: There is no space following the semicolon. This change makes all the new domains into local domains.

Now add a new router to the configuration to handle these domains as virtual domains. Put this router *first*, before all the other routers, immediately after the line "begin routers":

```
virtual_domains:
  driver = redirect
  domains = lsearch;/usr/exim/vdomains
  data = ${lookup{$local_part}lsearch{/usr/exim/aliases-$domain}}
  no_more
```

There must be no space after the semicolon in the "domains" line. (Remember to HUP the daemon.)

Create an alias file for the first virtual domain – as *root*, use *vi* to make the file */usr/exim/aliases-vdom1.afnog.org* containing these lines:

```
castro:  odoom@ghana.com
yourname:  your email address
```

The local parts *castro* and *yourname* should now be valid for the first virtual domain.

Test that Exim recognizes the virtual addresses (*not* as *root*):

$ `exim -bt castro@vdom1.afnog.org`

Please don't actually send test mail to that address – I get too much junk already!

☞ Now create a different alias file for the second virtual domain, */usr/exim/aliases-vdom2.afnog.org*. This time, alias *tony* to somebody else's address, and check (with **-bt**) that Exim treats that address differently.

Note: It is always important to test that incorrect addresses are handled the way you want. So you need to run this test:

```
$ exim -bt unknown@vdom1.afnog.org
```

3.2 Catching undeliverable mail

☞ Add a **redirect** router that sends all undeliverable mail in your domain to the postmaster. Where in the list of routers should this go? See if you can work out how to do this on your own without looking at the answer below. Do you think that having a router like this is a good idea on a busy host?

Here is a sample router that does this job:

```
unknown_to_postmaster:
  driver = redirect
  data = postmaster
```

It should be placed last, after all the other routers. Test it by sending mail to an unknown user.

4. Relaying from another host

In section 2.4 above, there is test to demonstrate that relaying is blocked if you connect to your host's IP address.

☞ We are now going to remove this block by changing a line in the configuration to let all the classroom hosts relay through your host. Change this line:

```
hostlist   relay_from_hosts = 127.0.0.1
```

to

```
hostlist   relay_from_hosts = 127.0.0.1 : xx.xx.xx.xx/mm
```

where *xx.xx.xx.xx/mm* is the classroom network. (Don't forget to HUP the daemon.) Then try the telnet test from section 2.4 again. This time it should accept the request to relay. Ask one of the other students in the classroom to try relaying through your host – it should work. If you can, telnet from a host outside the classroom network, and confirm that relaying is still blocked.

4.1 Allowing relaying to specific domains

The default configuration contains the line

```
domainlist relay_to_domains =
```

This defines domains to which your host will relay, wherever the message comes from. As you can see, the default list is empty, so no domains match.

☞ Add some domains to this line. For example, add the domain of your home email. In my case, this would be:

```
domainlist relay_to_domains = cam.ac.uk
```

Now we need to test that Exim will indeed relay to those domains (but not to others) from a host that does not match **relay_from_hosts**. Exim has a testing facility that lets you simulate an SMTP call from a remote host. Run it like this:

```
$ exim -bh 192.168.1.1
```

You will see some debugging output, and then an SMTP greeting line. Now type SMTP commands, waiting for a response between each one:

```
ehlo testhost
mail from:<localuser@pcnn.e1.ws.afnog.org>
rcpt to:<user@your.home.domain>
rcpt to:<user@some.other.domain>
```

You will see the tests that Exim is making as it runs the ACL after each RCPT command. Check that it allows relaying to the right domains, and not to any others. End the SMTP session with QUIT.

4.2 Configuring SMTP authentication

This is a simple exercise that uses RADIUS to check usernames and passwords with an unencrypted authentication mechanism. If you have enough time, the next exercise shows how to configure TLS to protect usernames and passwords from being snooped.

☞ Edit the Exim configuration, and find the line, almost at the end, that contains:

```
begin authenticators
```

Insert the following after this line:

```
LOGIN:
  driver = plaintext
  server_prompts = <| Username: | Password:
  server_condition = ${if radius{$1:$2} }
  server_set_id = $1
```

(Remember to HUP the daemon.) This authenticator passes the username "$1" and password "$2" to RADIUS for verification.

☞ Now try the previous test again:

```
$ exim -bh 192.168.1.1
```

Type the following command after Exim prints the greeting line:

```
ehlo testhost
```

The response to the EHLO command should include `AUTH LOGIN`, indicating that the LOGIN authentication echanism is available. The authentication exchange is difficult to do manually because it is base64-encoded, so end the SMTP session:

```
quit
```

Instead, we'll use the SMTP test utility which you downloaded earlier. First install it with these commands (as root):

```
# cp /usr/exim/smtpc /usr/exim/bin/smtpc
# chmod +x /usr/exim/bin/smtpc
```

Now you can run the following command (not as root):

```
$ smtpc -dvv -a LOGIN:username 127.0.0.1 "" your@emailaddress
```

The *username* is the one you created at the start of the week, and which you got working with RADIUS and ssh. You will see a `Password:` prompt. Type in your password. The program should then automatically connect, issue the `EHLO` command then `AUTH LOGIN` – the user name and password will be hidden – and finally `MAIL` and `RCPT` before quitting. The output should include:

```
smtpc: verified your@emailaddress
```

Try it again with an incorrect password, and the output should include:

```
smtpc: SMTP AUTH LOGIN failed: 535 Incorrect authentication data
```

☞ If you have time, you can try the same test from an external host, and actually send a real message for relaying.

5. More advanced configuration

These are ideas for things to do for those who have time. Don't worry if you do not have time to do this part in the workshop.

5.1 Configuring TLS

The plaintext authentication that we set up in the previous exercise is not secure against hackers snooping passwords, which is particularly easy on a wireless network, for example. So in a real configuration passwords should only be used with TLS.

☞ Edit the Exim configuration, and find the `LOGIN` authenticator. Add the following line after the `driver` line:

```
server_advertise_condition = ${if def:tls_cipher }
```

This ensures that the authenticator can only be used with TLS. Scroll up to the start of the configuration, and find the `acl_smtp_data` option. Add these options in the following lines:

```
tls_advertise_hosts = *
tls_certificate = /usr/local/etc/apache22/server.crt
tls_privatekey = /usr/local/etc/apache22/server.key
```

This makes Exim advertise TLS support, and tells it to use the same certificate and key as the web server. Save the configuation file and HUP the daemon.

☞ Now try the authentication test again:

```
$ exim -bh 192.168.1.1
```

Type the following command after Exim prints the greeting line:

```
ehlo testhost
```

The response to the EHLO command should not include "AUTH LOGIN", so LOGIN authentication is no longer available without encryption. Instead it should include "STARTTLS", indicating that you can use TLS. End the SMTP session with QUIT, because it isn't possible to do TLS manually:

```
quit
```

Instead, we'll use the SMTP test utility again. Run the following command – note that, compared to the command in the previous exercise, it has an extra `-s` flag to turn on TLS:

```
$ smtpc -dsvv -a LOGIN:username 127.0.0.1 "" your@emailaddress
```

Use the same *username* and *password* as in the prevous exercise. This should automatically connect, issue the `EHLO` and `STARTTLS` commands, do TLS negotiation, then `AUTH LOGIN` - the user name and password will be hidden - and finally `MAIL` and `RCPT` before quitting. The output should include:

```
smtpc: verified your@emailaddress
```

5.2 Demonstrate retry mechanisms

The easiest way to demonstrate what happens when Exim cannot deliver a message is to force connections to remote hosts to fail.

☞ Edit the configuration, and change the **remote_smtp** transport to be like this:

```
remote_smtp:
  driver = smtp
  port = 3456
```

(Remember to HUP the daemon.) This makes Exim try port 3456 instead of the SMTP port (25) when delivering, causing the remote host to refuse the connection (assuming you've chosen an unused port!)

☞ Send a message to a remote address and see what happens.

☞ Start a queue run

```
$ exim -q
```

and see what happens and what gets logged. Have a look at the message's own *msglog* file, which you can do from the monitor or by using the `-Mvl` option. For example:

```
$ exim -Mvl 19EdUm-00016A-IA
```

(That is an example message ID; you must use the real one for the message that is on your queue.)

☞ Use *exinext* to see when Exim is next scheduled to deliver to the host that failed:

```
$ exinext remote.domain
```

☞ Remember to remove the setting of `port` when you have finished playing with retries (and HUP the daemon).

5.3 Configure some address rewriting

☞ Find the rewriting section of the configuration (the part that starts with "begin rewrite"). Then add this line:

```
othername@otherdomain.com   postmaster@your.domain
```

☞ Now send a message to *othername@otherdomain.com* and see what happens.

☞ You can test rewriting rules with the `-brw` command line option:

```
$ exim -brw othername@otherdomain.com
```

5.4 Interface Exim to SpamAssassin

SpamAssassin is a content-based filtering system. It is written in Perl, and is very CPU-intensive. SpamAssassin can perform a number of network-based lookup checks that can take a long time to complete. It may therefore not be suitable for high-volume mail systems. The configuration is complex, though a basic installation using the ports system is straightforward.

☞ Install SpamAssassin:

```
# cd /usr/ports/mail/p5-Mail-SpamAssassin
# make
# make install
# make clean
```

In the directory `/usr/local/share/doc/p5-Mail-SpamAssassin`, the files `INSTALL` and `USAGE` give more detailed information.

☞ Set up the SpamAssassin configuration to disable the most expensive network-based tests.

```
# cd /usr/local/etc/mail/spamassassin
# cp local.cf.sample local.cf
# vi local.cf
```

Add the following lines:

```
use_dcc 0
use_pyzor 0
use_razor2 0
skip_rbl_checks 1
use_bayes 0
```

☞ Before starting the SpamAssassin daemon, you must edit */etc/rc.conf* to enable it. Add the following line to */etc/rc.conf*:

```
spamd_enable="YES"
```

This also ensures that the daemon will automatically be started when the system is rebooted.

☞ Now you can start the SpamAssassin daemon, which is called *spamd*:

```
# sh /usr/local/etc/rc.d/sa-spamd.sh start
```

☞ Check that it is running:

```
$ ps -ax | grep spamd
```

Next time you reboot, check that *spamd* has started automatically.

☞ You can test the *spamd* daemon manually using *spamc*, a client that sends mail to *spamd* for analysis:

```
$ spamc -R
subject: penis enlargement

Great new pills available!!!!!
```
Ctrl-D

The output should look something like this:

```
-2.0/5.0
Spam detection software, running on the system "xxx.xxx.xx.xx",
has identified this incoming email as possible spam. The
original message has been attached to this so you can view it
(if it isn't spam) or label similar future email. If you have
any questions, see the administrator of that system for details.

Content preview:  Great new pills available!!!!! [...]

Content analysis details:   (-2.0 points, 5.0 required)

 pts rule name          description
---- ------------------ ----------------------------------
 0.0 MISSING_DATE       Missing Date: header
-2.8 ALL_TRUSTED        Did not pass through any untrusted hosts
 0.8 BODY_ENHANCEMENT2  BODY: Information on getting larger body
                        parts
```

Despite its content, this message has been strongly tagged as 'not spam' because it has not been through any 'untrusted' hosts. You may not see exactly this output: it depends on the SpamAssassin release and the default SpamAssassin configuration that has been installed.

We are now going to change the Exim configuration so that every message is passed to SpamAssassin. At first, we won't block any messages. Instead, we will put the spam score and other SpamAssassin output into new headers that are added to the message.

☞ Edit */usr/exim/configure* and find the line that contains:

```
acl_smtp_rcpt = acl_check_rcpt
```

☞ Add these new lines:

```
acl_smtp_data = acl_check_data
acl_not_smtp  = acl_check_data
```

The first line asks Exim to run an ACL check when a message's data has been received in an SMTP transaction. The second line asks for the same ACL to be run on non-SMTP messages. This ensures that all incoming messages are scanned. We must now define the ACL.

☞ Find the configuration line that contains:

```
begin acl
```

☞ Insert the following lines immediately after that line:

```
acl_check_data:
  warn     spam    = nobody
           message = X-is-spam: over spam threshold
  warn     message = X-Spam_score: $spam_score\n\
                     X-Spam_score_int: $spam_score_int\n\
                     X-Spam_bar: $spam_bar\n\
                     X-Spam_report: $spam_report
  accept
```

The **warn** verb in an ACL doesn't accept or reject, but if its conditions are true, it can add headers to the message. The first **warn** passes the message to SpamAssassin, and if the spam score is over

the threshold, an `X-is-spam:` header is added. The second **warn** adds some more headers containing information from SpamAssassin. These are always added, unconditionally.

☞ Now send yourself a spammy message. This example uses the local SMTP interface:

```
$ exim -bs
mail from:<>
rcpt to:<yourname@yourhostname>
data
message-id: abcd
subject: BUY VIAGRA HERE!!!

<html><p>Dear Friend</p>
<p>VIAGRA $10.99</p>
<p>RISK FREE</P></HTML>
.
quit
```

Take a look at the headers of the message that you receive.

☞ Change the first **warn** to **deny**, and try the test again. If the spam score is at least 5, the message should be rejected.

5.5 Interface Exim to ClamAV

ClamAV is free virus-scanning software. You can interface Exim to it in much the same way as for SpamAssassin. The documentation is online at *http://clamav.sourceforge.net/doc/*. Installing ClamAV is very similar to the way you installed SpamAssassin.

☞ Install ClamAV from the ports tree:

```
# cd /usr/ports/security/clamav
# make
```

At this point, a dialog box pops up; use TAB to move to 'OK', then hit ENTER.

```
# make install
```

If you are using a C-shell (*csh* or *tcsh*), you must run the command

```
# rehash
```

to make the *clamdscan* command available. This is not necessary if you are using *bash* or another Bourne-compatible shell (there is no *rehash* command in these shells).

ClamAV needs it own user, called *clamav*, which must be in the *exim* group so that it can access Exim's spool files. The ports system creates the user *clamav*, but it does not add it to the *exim* group.

☞ Add *clamav* to the *exim* group:

```
# vi /etc/group
```

Find the line that starts

```
exim:*:90:
```

Add *clamav* to the end of that line, using a comma to separate it from any other user names. Your own login name should be there already, so the final line looks like this:

```
exim:*:90:yourid,clamav
```

Warning: Make sure you use a comma separator, and no spaces.

ClamAV has two daemons: one is the actual virus scanner, and the other (called *freshclam*) updates the virus database periodically over the Internet. New viruses are being created all the time. When you run anti-virus software, it is important to keep it updated. The *freshclam* daemon makes this very easy. It is also possible to run the *freshclam* command manually (see the man page for details).

The configuration files for the ClamAV daemons are */usr/local/etc/clamd.conf* and */usr/local/etc/freshclam.conf*. Suitable defaults should ahve been installed, so you do not need to change these files.

☞ Before starting the two ClamAV daemons, you must edit */etc/rc.conf* to enable them:

```
# vi /etc/rc.conf
```

Add these lines:

```
clamav_clamd_enable="YES"
clamav_freshclam_enable="YES"
```

The daemons should now start automatically whenever you reboot. Next time you reboot, check that the ClamAV daemons have started.

☞ Now we can start the ClamAV daemons manually:

```
# sh /usr/local/etc/rc.d/clamav-clamd.sh start
# sh /usr/local/etc/rc.d/clamav-freshclam.sh start
```

If all has worked, we can now run some tests of the scanner. How can you test an anti-virus scanner? You don't want to be sending yourself a real virus! Luckily, a test virus called "eicar" exists. It consists of a short string of printing characters.

☞ Create a file that contains this string:

```
$ cat >/tmp/eicar
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

The third character is the letter O, not the digit 0.

☞ We can use the *clamdscan* command to check an individual file (or a directory of files) for viruses:

```
$ clamdscan /tmp/eicar
```

You should see output like this:

```
/tmp/eicar: Eicar-Test-Signature FOUND

----------- SCAN SUMMARY -----------
Infected files: 1
Time: 0.003 sec (0 m 0 s)
```

Now that we have ClamAV working, we can edit Exim's configuration so that every message is scanned for viruses.

☞ Edit */usr/exim/configure* and insert this line somewhere near the top, in the main section, before the first "begin" line:

```
av_scanner = clamd:/var/run/clamav/clamd
```

This option tells Exim where to find its anti-virus scanner: */var/run/clamav/clamd* is a socket that the ClamAV daemon creates for communication.

☞ Now add some more lines to the **acl_check_data** ACL that you created for SpamAssassin. Add these lines at the start:

```
  deny     message = This message contains \
                     a virus ($malware_name).
           malware = *
```

For reference, the entire ACL should now read like this:

```
acl_check_data:
  deny     message = This message contains \
                     a virus ($malware_name).
           malware = *
  warn     spam    = nobody
           message = X-is-spam: over spam threshold
  warn     message = X-Spam_score: $spam_score\n\
                     X-Spam_score_int: $spam_score_int\n\
                     X-Spam_bar: $spam_bar\n\
                     X-Spam_report: $spam_report
  accept
```

☞ Send yourself the eicar test and see what happens (use copy-paste to copy the test data):

```
$ exim -bs
mail from:<>
rcpt to:<yourname@yourhostname>
data
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
.
quit
```

You could also try using your MUA to send the test virus to yourself as an attachment.

5.6 What next?

If you have got this far in the available time, you are probably starting to understand the basics of Exim pretty well. You can either start reading the book, or help out other students who are having problems.

* * *