Installing a Mail System on Debian Sarge

Introduction

I wanted to run a complete mail server for myself and any of my friends or family who would prefer to have a regular mail account instead of one from one of the free web based services. I decided to encrypt and authenticate connections for both sending and receiving mail, and to handle receiving mail with IMAP, rather than POP. The final setup can handle large amounts of both incoming and outgoing mail and is easy to use...

Overview

Accounts

I had two basic options when selecting how to handle accounts. Either every valid mail account could be tied to a valid user account on the mail server, or I could set up a backend such as an SQL or LDAP database to handle authentication that wouldn't be tied to the server.

Based on my needs and relatively small number of potential users, I decided to go with mail delivery to local user accounts, while retaining support for virtual domains. This way I can support any domain with an MX record pointing to the mail server, and not just my own domain, but mail will be delivered to and authenticated against local user accounts.

Software

Since I have a host on the internet that is running Debian Sarge (Testing), I decided to use that as the host system, and tailored the rest of the system around that. All packages except the SquirrelMail SIEVE plugin are straight out of the Debian package management system.

The software used is:

- Postfix [http://www.postfix.org/] (MTA), and TLS patch.
- Spamassassin [http://www.spamassassin.org] (Spam Filter)
- Cyrus-SASL [http://asg.web.cmu.edu/sasl/] (Authentication)
- Cyrus-IMAP [http://asg.web.cmu.edu/cyrus/imapd/] (IMAP and SIEVE server)
- Apache [http://httpd.apache.org] (For webmail)
- SquirrelMail [http://www.squirrelmail.org] (Webmail system)
- Plugins for SquirrelMail: AvelSIEVE [http://email.uoa.gr/projects/squirrelmail/avelsieve.php] (SIEVE script creator for Squirrelmail)

Installation

Before We Begin

Before installing software make sure the current system is up to date:

```
# apt-get update
```

```
# apt-get upgrade
```

At this point everything should be ready to go. The default MTA on debian, Exim, should be uninstalled by apt automatically when postfix is installed. Everything else is probably new software.

Postfix

```
# apt-get install postfix postfix-tls
```

This installs a postfix system that is ready to be setup for TLS (SSL) encrypted connections. I chose reasonable defaults to the questions the debian installer asks. Anything that needs to be changed can be changed later without much trouble. At this point local delivery probably works, but the system is far from complete. I didn' t worry about it at this point, since the way the system will be set up I also need to tie postfix into SASL and IMAP, and probably send it to Spamassassin for filtering as well.

A note for users of etch and unstable

In etch and unstable, the tls code has been moved into the main postfix install, so there is no need to install the second package. The following will install a SSL and TLS aware postfix in etch or unstable.

```
# apt-get install postfix
```

Cyrus SASL and IMAP

Next install all the needed packages from project Cyrus, namely the SASL library, the Cyrus admin and client programs, and the Cyrus IMAP daemon. Although it is available, I will not install the corresponding Cyrus POP3 server. This is an IMAP only setup.

```
# apt-get install libsasl2 libsasl2-modules sasl2-bin
# apt-get install cyrus21-admin cyrus21-clients cyrus21-common cyrus21-imapd
```

SpamAssassin

Since spam is such an annoyance these days installing a decent spam filter seems to be a prudent choice.

```
# apt-get install spamassassin spamc
```

Later on, when I get to SIEVE script creation and webmail options, I' ll cover Apache and Squirrelmail. For now, and to get the server working the way we want it, everything is ready to go.

Configuration

Start with configuring Cyrus-SASL for authentication in general. Both the SMTP server, postfix, and the IMAP server, Cyrus-IMAP, will authenticate using another piece of daemon software, saslauthd. Saslauthd will handle the actual authentication of users. With this in mind, start first by configuring SASL to authenticate against the system accounts using PAM.

SASL

Edit /etc/default/saslauthd to enable the daemon and set the authentication mechanism.

```
START=yes
MECHANISMS="pam"
```

If you want to use another authentication mechanism, this is where you would set it. The system here authenticates against user accounts, so pam is the relevant method.

Testing SASL

Start up the server now to test it.

```
# /etc/init.d/saslauthd start
```

Now test it using a known working user account.

```
# testsaslauthd -u username -p password
```

If it works, you should see

```
0: OK "Success."
```

as a result. If so, saslauthd can talk to pam and authenticate users. Now shut it back down.

```
# /etc/init.d/saslauthd stop
```

Finish SASL Configuration

Note: see Update #2 below for a much simpler method to achieve the same end effect.

Edit the file /etc/init.d/saslauthd and make the following changes:

Add the line `PARAMS=" -m /var/spool/postfix/var/run/saslauthd"` to the variable declarations at the top of the file. Change `PWDIR` to `/var/spool/postfix/var/run/saslauthd`, and `PIDFILE` to `" /var/spool/postfix/var/run/${NAME}/saslauthd.pid"` . Then go down to the `start)` case and add the lines

```
rm -rf /var/run/saslauthd
ln -s /var/spool/postfix/var/run/saslauthd /var/run/saslauthd
```

after `test -z " $dir" ; || createdir $dir;`.

The following commands will finish up the SASL configuration.

```
# dpkg-statoverride --remove /var/run/saslauthd
# dpkg-statoverride --add --update root sasl 710 /var/spool/postfix/var/run/saslauthd
```

These steps move the default location of the saslauthd mux file from `/var/run/saslauthd`, which is outside the postfix chroot, to `/var/spool/postfix/var/run/saslauthd`, inside the postfix chroot. The additions to the `/etc/init.d/saslauthd` file will remove the folder if it exists and create a link between the mux location in the chroot and the default location. We do this so that both postfix and cyrus-imapd can authenticate against the same daemon, even though one server is inside the chroot and one is not.

The daemon can now be restarted and it should give no more problems.

Update: Using SASL while running Postfix in a chroot is a tricky proposition. While I have explained how to do so with

good success, it is my understanding that the Postfix author recommends instead not running Postfix chrooted and leaving SASL in the default location. If you would rather do this, do not change the running location of the saslauthd daemon, and instead remove the chroot from the Postfix master.cf.

To do so, change the ' -' character under in the fifth column of master.cf to ' n' for each of the services from which you wish to remove the chroot.

Update #2: Although the listed method works quite well, it is needlessly complicated to implement. The same basic thing can be accomplished by creating by mounting /var/run/saslauthd inside the postfix chroot. This can be accomplished by using a bind mount. To implement this, add the following line to /etc/fstab.

```
/var/run/saslauthd    /var/spool/postfix/var/run/saslauthd    none    rw,bind    0 0
```

Create the directory the mount will reside in as well.

```
# mkdir -p /var/spool/postfix/var/run/saslauthd
```

The same location is now accessable outside the chroot at /var/run/saslauthd and inside the chroot at /var/spool/postfix/var/run/saslauthd (which postfix sees as /var/run/saslauthd). This avoids editing the saslauthd init script, messing with dpkg, and creating symbolic links. The only visible side effect is an extra listing from the `df` command which doesn't know that the bind mount isn't an actual separate disk.

Postfix

There are a lot of steps necessary to get postfix up and running. It needs to be set up to connect to saslauthd, to use SSL, and to link with cyrus-imapd and Squirrelmail. This section will go over the first stages of configuring postfix, but later sections will cover the other parts.

Edit the file /etc/postfix/main.cf and add the following lines to the file.

```
# SASL Auth Settings
smtpd_sasl_local_domain =
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
broken_sasl_auth_clients = yes
smtpd_recipient_restrictions = permit_mynetworks, permit_sasl_authenticated, reject_unauth_dest
```

Also edit /etc/postfix/sasl/smtpd.conf and put in the lines:

```
pwcheck_method: saslauthd
mech_list: PLAIN LOGIN
```

Reload postfix and check to see if auth is enabled.

```
# /etc/init.d/postfix reload
# telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 dev.ev-15.com ESMTP Postfix (Debian/GNU)
```

Type in `EHLO domain.com`. If the server responds with something like the following, then authentication is enabled in the server. Then type `quit` to exit.

```
250-mail.domain.com
250-PIPELINING
250-SIZE 10240000
```

```
250-VRFY
250-ETRN
250-AUTH LOGIN PLAIN
250-AUTH=LOGIN PLAIN
250 8BITMIME
```

You may also need to add postfix to the sasl group:

```
# adduser postfix sasl
```

Postfix should now tie authentication into itself using saslauthd. Note that the advertised authentication methods are only PLAIN and LOGIN, both of which are plain text based. To enable one of the shared secret methods, such as CRAM-MD5 or DIGEST-MD5, a different authentication method would need to be used. To avoid delivering these plain passwords over the internet, wrap everything up in an encrypted layer using SSL/TLS.

SSL

Certificates are required to use TLS and SSL. There are generally two classes of certificates: self-signed certificates and certificates signed by one of the major certificate authorities. If you need implicitly trusted certificates that you can use and let your customers be comfortable with, you' ll probably want to go to a CA and purchase them. With smaller sites and users who understand certficate warnings, self-signed certificates are easy to create and still allow encryption. They will throw up warnings in mail clients and web browsers, but the certificates can usually be permanently accepted by users.

If the primary goal is encryption and the trust issues are not major, then a self-signed certificate is adequate.

Certificate Creation

The following steps will create a new CA, certificate request, and certificate.

Start with making a new CA.

```
# cd /usr/lib/ssl/misc
# ./CA.pl -newca
```

Answer the questions as they come with reasonable information. The value for CN (Common Name) should be the hostname of the server that the certificates will be used on.

Now make the server certificate request.

```
# ./CA.pl -newreq-nodes
```

Now sign it

```
# ./CA.pl -sign
```

Copy the files to /etc/ssl/certs

```
# cp newcert.pem /etc/ssl/certs/
# cp newreq.pem /etc/ssl/certs/
# cp demoCA/cacert.pem /etc/ssl/certs/
```

Add SSL to Postfix

Edit /etc/postfix/main.cf again and add the following lines to the file.

```
# TLS Information
smtpd_use_tls = yes
#smtpd_tls_auth_only = yes
smtpd_tls_key_file = /etc/ssl/certs/newreq.pem
smtpd_tls_cert_file = /etc/ssl/certs/newcert.pem
smtpd_tls_CAfile = /etc/ssl/certs/cacert.pem
smtpd_tls_loglevel = 3
smtpd_tls_received_header = yes
smtpd_tls_session_cache_timeout = 3600s
tls_random_source = dev:/dev/urandom
```

Edit /etc/postfix/master.cf and uncomment the following lines.

```
tlsmgr     fifo   -   -  n  300   1  tlsmgr
smtps   inet   n   -   n   -   -   smtpd -o smtpd_tls_wrappermode=yes -o smtpd_sasl_auth_enable=yes
587     inet   n   -   n   -   -   smtpd -o smtpd_enforce_tls=yes -o smtpd_sasl_auth_enable=yes
```

This will enable TLS, the smtps port, and the dedicated port 587 for mail submission.

The line `#smtpd_tls_auth_only = yes` will be uncommented later so that users are required to encrypt their connections. For now though leave it alone until testing is complete. To test, telnet in again, as above, and see if the line `250-STARTTLS` is there. If so then TLS should be available.

Cyrus IMAP

This guide uses Cyrus-IMAP, largely because it is high performance, and because of SIEVE server-side filtering. One thing to note about this package is that it takes the " black box" approach to IMAP mail delivery. Mail is never delivered to a user' s home folder, but rather to a set of dedicated cyrus folders. If you would rather mail either be delivered to maildirs or mbox files, then the Courier- or UW-IMAP servers would be better choices. Later on SquirrelMail and a plugin to easily edit SIEVE scripts will be installed.

Open the file /etc/cyrus.conf for editing and choose which services to run. Unless POP access is desired, leave only IMAP, secure IMAP, and SIEVE. For extra security turn off IMAP to force users to connect over SSL. Under the SERVICES section, uncomment the line

```
imaps cmd="imapd -s -U 30" listen="imaps" prefork=0 maxchild=100
```

and comment out the pop3 lines.

This setup uses LMTP. The default socket set in the file should be fine. The line for sieve should already be uncommented, but check it just to be sure. Everything else in the file can stay the way it was.

Now open the file /etc/imapd.conf. Most of the settings can be left the way they are, but there are a few more to set. As always, read the comments for the various settings in the file. The directive `sieveusehomedir` should be set to false so that remote sieve management with SquirrelMail will work.

The way this setup is working, the following values should be set:

```
admins: cyrus
allowplaintext: yes
sasl_mech_list: PLAIN
sasl_pwcheck_method: saslauthd
```

IMAP and SSL

Set the following values in imapd.conf, leaving the rest alone.

```
tls_cert_file: /etc/ssl/certs/newcert.pem
tls_key_file: /etc/ssl/certs/newreq.pem
tls_ca_file: /etc/ssl/certs/cacert.pem
tls_ca_path: /etc/ssl/certs
```

Finish Cyrus Setup

Restart cyrus.

```
# /etc/init.d/cyrus21 restart
```

Then, start saslauthd again.

```
# /etc/init.d/saslauthd start
```

The way things are set up, only the cyrus user can administer the cyrus server. Set a password for the user cyrus and then su to the cyrus user.

```
server# passwd cyrus
server# su cyrus
```

Now add an IMAP user. Usernames are prefaced with `user.`, followed by the login name of the user. For a user account jimmy, the mailbox would be named `user.jimmy`.

Log into the cyrus admin tool and create a new mail user to match a local delivery name that postfix knows. Do this for each local account that receives mail, or postfix will throw errors.

```
$ cyradm localhost
cyradm> cm user.username
... repeat for all users ...
cyradm> quit
```

Cyrus and Postfix

Edit the file /etc/postfix/main.cf and add the following line to the file, removing or commenting out any other `mailbox_transport =` lines.

```
mailbox_transport = lmtp:unix:/var/run/cyrus/socket/lmtp
```

Create the lmtp group and add postfix to that group.

```
# addgroup lmtp
# adduser postfix lmtp
```

Fix the socket directory permissions and restart both mail servers.

```
# dpkg-statoverride --force --update --add cyrus lmtp 750 /var/run/cyrus/socket
# /etc/init.d/postfix restart
# /etc/init.d/cyrus21 restart
```

Cyrus should now be linked to Postfix. All mail from Postfix will be handed off to the Cyrus server for delivery. This will fail if Postfix tries to deliver mail for a user Cyrus doesn't know about, so make sure that when new users are added to the mail system that the corresponding mailboxes are also added with cyradm.

Postfix and Multiple Domains

If the mail server will handle more than one domain but deliver all mail to local users, then the following information will configure this behavior. This is based on the guide at http://www.postfix.org/VIRTUAL_README.html [http://www.postfix.org/VIRTUAL_README.html].

Create the file /etc/postfix/virtual and populate it with entries like the following

```
user1@domain1.com  realuser1
user2@domain1.com  realuser2
user3@domain2.com  realuser1
# send all mail for domain3 to realuser3
@domain3.com       realuser3
```

The left side maps to the email address people will send mail to, and the right side maps to the local system account that mail will be delivered to (through Cyrus IMAP). Once this file has been created, it needs to be hashed for postfix to use, so run the postmap command to do this:

```
# postmap /etc/postfix/virtual
```

Edit /etc/postfix/main.cf and add the following lines.

```
# Virtual Domain Settings
virtual_alias_domains = domain1.com, domain2.com, domain3.com
virtual_alias_maps = hash:/etc/postfix/virtual
```

This tells Postfix which domains to deliver to, and what file to use to check where to deliver the mail. It' s pretty simple to get going. Reload Postfix again and things should work. Basic system setup should be complete at this point.

Troubleshooting

The most useful information for troubleshooting is found in the /var/log/mail.log file. Open a console and use the command

```
tail -f /var/log/mail.log
```

to watch mail server activity. Send a few test mails to an address on the server, and use the server to send test mails to other accounts. If errors occur use the information in the mail log file to try to track them down.

After things are working well it is a good idea to uncomment the line `smtpd_tls_auth_only = yes` and change the value of `smtpd_tls_loglevel` to something lower, such as 2.

Extra Software

In addition to basic mail service, it is also possible to provide webmail service using the SquirrelMail package, spam filtering with SpamAssassin, and server-side mail filtering with SIEVE.

Spam Filtering

SpamAssassin was installed earlier. Configuring it now will hopefully cut down on the amount of junk mail that users have to deal with. First edit the file /etc/default/spamassassin and change `ENABLED=0` to `ENABLED=1`. Start the daemon.

```
# /etc/init.d/spamassassin start
```

Set up Postfix to run mail through the SpamAssassin filter. Edit /etc/postfix/master.cf and add the following lines to the bottom of the file. Be sure to start the second and third lines with white space so that postfix treats the whole thing as a single line.

```
spamassassin
          unix  -       n       n       -       -       pipe
     user=filter argv=/usr/bin/spamc -f -e /usr/sbin/sendmail -oi -f ${sender} ${recipient}
```

Near the top of the file, change

```
smtp      inet  n       -       -       -       -       smtpd
```

to

```
smtp      inet  n       -       -       -       -       smtpd -o content_filter=spamassassin
```

This setup runs the spam filter as the user `filter`. Since this user likely does not exist, add the user to the system with the `adduser` script. The login of the filter user can be safely disabled without affecting mail delivery, which can be done by done by editing the password - file in /etc and inserting an asterisk `*` at the beginning of the second field where the password itself is written down.

Webmail

To use webmail, first set up Apache if it is not already working. To install Apache[1)]:

```
# apt-get install apache2 libapache2-mod-php4
```

Since users will log in to SquirrelMail using a web form it is a good idea to enable SSL on the web server. Apache uses a certificate of a slightly different form than has been used in this guide so far. It combines the site certificate with the private key from the certificate request. To create this file copy the private key from newreq.pem and the signed certificate from newcert.pem into a new file /etc/apache2/ssl/apache.pem.

Enable the SSL module.

```
# a2enmod ssl
```

Copy the default configuration file in /etc/apache2/sites-available/default to a new file in /etc/apache2/sites-available, such as default-ssl. Edit the file. Change the `<NameVirtualHost *>` line to `<NameVirtualHost *:443>` the `<VirtualHost *>` line to `<VirtualHost *:443>` and add the following lines to the file inside the VirtualHost directive:

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/apache.pem
```

Edit the default site file and change the line `<NameVirtualHost *>` to `<NameVirtualHost *:80>` and the line `<VirtualHost *>` to `<VirtualHost *:80>`.

Now open the file /etc/apache2/ports.conf and add the following line to the file.

```
Listen 443
```

Enable the new SSL-enabled site.

```
a2ensite default-ssl
```

Restart Apache.

```
/etc/init.d/apache2 force-reload
```

Now that Apache is up and running, install the SquirrelMail package.

```
# apt-get install squirrelmail
```

Run the configuration for SquirrelMail and adjust the settings for your system.

```
# squirrelmail-configure
```

The configuration is quite straightforward. Once that is done try logging into the newly created squirrelmail site (The line `Include /etc/squirrelmail/apache.conf` may need to be added to /etc/apache2/apache2.conf). If you can log in to the site then SM is up and running.

SIEVE Filters

SIEVE is a simple yet powerful way to filter mail on the server side. In this setup it is part of the Cyrus IMAP server package, and runs on port 2000. I waited this long to introduce it, because I find the nicest way to build sieve scripts is with the AvelSIEVE plugin for SquirrelMail. First download the plugin from the SquirrelMail site and untar in in the SquirrelMail plugins folder. Copy the sample configuration to the real file and edit it to your configuration. If you're like me you didn't have to change anything.

Run the SquirrelMail configure script again and select plugins to view the list of available plugins. Type the number of the avelsieve plugin to enable it. Save your prefs and exit, and go back to your SquirrelMail web page to test it.

Click on the new filters link on the top bar. If you get an error about connecting, check /etc/cyrus.conf. If you get a connection refused error test sieve using sivtest. If you can connect to localhost but not your machines dns name, you'll have to edit that config file to fix the problem. I removed the "localhost" parameter from the sieve init line.

If you get a dialog that will let you add a new rule, congratulations. You're set. I usually create rules based on the X-Spam-Flag header to move mail to INBOX.Junk, and a few others to move mailing lists to their respective folders. Always make sure you save your rules before you leave the page though, or they will not take effect.

The nice thing about the sieve filters is that even though I set them up through squirrelmail, they run at the IMAP server level, and all mail you get through IMAP is filtered according to your rules.

Conclusion

That's pretty much it. The mail server can now send and receive mail, authenticate users, encrypt connections with SSL, and provide webmail and server-side mail filtering. Good luck with it.

Steve Block, 29 January 2005

Acknowledgements

This guide wouldn't be possible without the developers of all the software packages used here. I would like to extend my thanks to all of the developers. Thanks are also due to the Debian developers and package maintainers for their sensible setup of the base Debian system and its corresponding packages.

This guide is originally based on the guide found at http://www.idealog.us/2004/10/helpful_guide_t.html [http://www.idealog.us/2004/10/helpful_guide_t.html], and has been tailored to fit my preferences, choice of software, and to solve some minor issues.

Comments and Updates

Feel free to add updates or comments on this guide in this section.

Hi and thank you very much for this very good how to. I dared to add some minor hints. And what I didn't correct was the ssl configuration of apache but I guess that it doesn't work without a SSLCertificateKeyFile which has to be encrypted. On the other hand the private key for postfix may not be encrypted as far as I know. I would also suggest to restart the server after the modification of the fstab. The steps may be to obvious for your knowledge but they are not for mine if I may say so.

yours Peter Guyan

---

1) This guide has been updated to use Apache 2.0