Implementation

This section describes how to implement a virtual mail solution. Not every little detail is covered, just what is needed above and beyond the „standard“ installations.

Prerequisites

Here is the list of software, with their version numbers, that we have tested this configuration with:

Software

- Red Hat Linux 6.2, 7.1, or 7.2
- Postfix 1.1.3
- OpenLDAP 2.0.21
- Courier-IMAP 1.4.1
- Procmail v3.22

Preparing the Unix System

To prepare the Unix system there are a few tasks you'll need to accomplish: create the vmail user and decide where you're going to store the virtual users email.

Creating the vmail user is just like creating any other system account. You'll want to have a UID and a GID that is used alone for vmail. You may also want to set its home directory to the location you've selected for the storage area of the virtual users' email.

In our system we used vmail as the user and group name. We also decided to store our virtual users email in `/home/vmail`.

The following example would work on a RedHat Linux distribution and result in a vmail user being created and an empty mail storage directory being created.

```
% useradd -k -m -r -d /home/vmail vmail
% mkdir ~vmail/domains
% chown vmail.vmail ~vmail/domains
```

You will also need to create an account and two groups for postfix, but that is covered in Postfix's INSTALL documentation.

OpenLDAP

You do not need to follow any special instructions for compiling and installing OpenLDAP, so please consult its documentation for full instructions. For a production environment, you should read up on how to run OpenLDAP as a non-root user, setup a chroot environment, and replication. This section

describes how to configure slapd for a single server, how to create the base tree structure, and how to insert some basic data into the LDAP directory. Please consult Figure 2 as this is the LDAP tree we will aim to setup.

Configuring slapd

All slapd configuration is in slapd.conf.

Adding Schemas

You need to make Courier's schema file available, so copy the file from `authlib/authldap.schema` in the Courier distribution to `/usr/local/etc/openldap/schema/courier.schema`. `courier.schema` depends on `cosine.schema` and `nis.schema`. Add these lines to `slapd.conf`:

```
include                 /usr/local/etc/openldap/schema/cosine.schema
include                 /usr/local/etc/openldap/schema/nis.schema
include                 /usr/local/etc/openldap/schema/courier.schema
```

Adding a Database Definition

Next, you need to set up a database definition. You can do this with the following lines:

```
database        ldbm
directory       /usr/local/var/openldap-ldbm
suffix          "dc=myhosting,dc=example"
```

The database directive specifies the back-end type to use. You should use LDBM as the back-end database. The directory directive specifies the path to the LDBM database. The suffix directive specifies the root suffix for this database.

Creating the root User

The next few lines set up the „super user“ or „root“ account:

```
rootdn          "cn=Manager,dc=myhosting,dc=example"
rootpw          {SSHA}ra0sD47QP32ASAlaAhF8kgi+8Aflbgr7
```

The rootdn entry has complete access to the database, which is why the password is stored outside the actual database. The password in rootpw should always be stored in hashed format. Do not store the password in clear text. To convert the clear text password secret to a hashed format, use the slappasswd command:

```
% slappasswd
New password: secret
Re-enter new password: secret
```

```
{SSHA}ra0sD47QP32ASAlaAhF8kgi+8Aflbgr7
```

Take the output from slappasswd, and copy that into slapd.conf, as we did above.

Defining Indexes

To speed up searches, you should create indexes for commonly searched attributes. In OpenLDAP, you can not only choose which attributes to index, but you can choose which types of searches to index on. For example, if you index the field mail, you have the option of creating indexes for presence, equality, approximate, and/or substring searches. We want to create the following index policy:

- Create presence and equality indexes on objectClass.
- Create equality and substring indexes on mail and cn.

To implement this policy, add these lines to `slapd.conf`:

```
index   objectClass  pres,eq
index   mail,cn      eq,sub
```

Setting up Access Control

The last part in `slapd.conf` is the access control. You can define your own policy, be here's the one we've adopted:

- The user can change any of their own attributes.
- Anyone in the postmaster group of the domain may change any user's attributes in their domain, including the password. This allows the postmaster to reset a users password if they forget it.
- Anonymous (non-authenticated) users may read all information, except the password attribute.

Access control statements are evaluated in order, so they should be defined from most specific to most general. Access to the password attribute, userPassword, is the most specific in our case, and hence it's specified first:

```
access to dn=".*,o=([^,]+),o=hosting,dc=myhosting,dc=example"
          attr=userPassword
        by self write
        by group/organizationalRole/roleOccupant=\
            "cn=postmaster,o=$1,o=hosting,dc=myhosting,dc=example" write
        by anonymous auth
        by * none
```

The access to line specifies what entries and attributes to which the following rules apply. The dn regular expression matches any entry in a domain of our hosting tree, and attr limits these rules to the userPassword attribute. Write access is granted to the user itself and anyone in the postmaster group. Anonymous users may only access this field when trying to authenticate. For all other cases, access is denied.

Next, all other attributes to entries in a domain's tree are specified:

```
access to dn=".*,o=([^,]+),o=hosting,dc=myhosting,dc=example"
        by self write
        by group/organizationalRole/roleOccupant=\
            "cn=postmaster,o=$1,o=hosting,dc=myhosting,dc=example" write
        by * read
```

This access to line is very similar the previous one, except that there is no attr specification. Hence, this matches all other attributes other than userPassword. Again, write access is granted to the user and anyone in the postmaster group. Everyone is granted read access.

Finally, we provide read access to all other elements in the database:

```
access to *
        by * read
```

Creating the Directory Tree

Now that slapd is configured, it's time to start adding data to the LDAP directory. We will use the command line tools that come with OpenLDAP and create LDIF files to modify the directory.

Creating the Base Directory

The first step is to create a base tree structure with our root node, the hosting organization, and an entry for the rootdn. Create a file called `base.ldif` with the following contents:

```
dn: dc=myhosting, dc=example
objectClass: top

dn: cn=Manager, dc=myhosting, dc=example
objectClass: top
objectClass: organizationalRole
cn: Manager

dn: o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: organization
o: hosting
```

Now use ldapadd, binding as the root user, to add this LDIF:

```
$ ldapadd -x -D "cn=Manager,dc=myhosting,dc=example" -w secret -f base.ldif
adding new entry "dc=myhosting, dc=example"
adding new entry "cn=Manager, dc=myhosting, dc=example"
adding new entry "o=hosting, dc=myhosting, dc=example"
```

Adding a Domain

Domains may now be added under the hosting tree. Each domain needs to have postmaster and abuse entries at a minimum. To create a tree for domain1.example, create a file called `domain1.example.ldif` with the following contents:

```
dn: o=domain1.example, o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: organization
o: domain1.example

dn: cn=postmaster, o=domain1.example, o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: organizationalRole
objectClass: CourierMailAlias
cn: postmaster
mail: postmaster@domain1.example
maildrop: postmaster

dn: mail=abuse@domain1.example, o=domain1.example, o=hosting, dc=myhosting,
 dc=example
objectClass: top
objectClass: CourierMailAlias
mail: abuse@domain1.example
maildrop: abuse
```

Notice that the maildrop attributes are local email accounts and will forward to the postmaster and abuse accounts in `/etc/aliases`. There are also no users in the postmaster role, so only the root user can create accounts at the moment. Add this domain with the following command:

```
$ ldapadd -x -D "cn=Manager,dc=myhosting,dc=example" -w secret \
    -f domain1.example.ldif
adding new entry "o=domain1.example, o=hosting, dc=myhosting, dc=example"
adding new entry "cn=postmaster, o=domain1.example, o=hosting, dc=myhosting,
dc=example"
```

Adding a User

Now, let's add a user with an email user1@domain1.example. Let's also grant this user postmaster privileges for domain1.example. Create a `user1.domain1.example.ldif` with the following contents:

```
dn: mail=user1@domain1.example, o=domain1.example, o=hosting, dc=myhosting,
 dc=example
objectClass: top
objectClass: CourierMailAccount
mail: user1@domain1.example
homeDirectory: /home/vmail/domains
uidNumber: 101
```

```
gidNumber: 101
mailbox: domain1.example/user1

dn: cn=postmaster, o=domain1.example, o=hosting, dc=myhosting, dc=example
changetype: modify
add: roleOccupant
roleOccupant: mail=user1@domain1.example, o=domain1.example, o=hosting,
 dc=myhosting, dc=example
```

The first section adds a new entry for the user. The home directory and mailbox point to the physical mailbox on the file system. The uidNumber and gidNumber attributes are required, but not used, so they are filled in with dummy values of 101. The second section modifies the postmaster entry by adding a roleOccupant attribute with the DN of `user1@domain1.example`. Let's create this user:

```
$ ldapadd -x -D "cn=Manager,dc=myhosting,dc=example" -w secret \
    -f user1.domain1.example.ldif
adding new entry "mail=user1@domain1.example, o=domain1.example, o=hosting,
dc=myhosting,dc=example"
modifying entry "cn=postmaster, o=domain1.example, o=hosting, dc=myhosting,
dc=example"
```

The user does not have a password yet, so even though he has been granted postmaster privileges, he cannot be authenticated. Use the ldappasswd command to set the initial password to user1:

```
$ ldappasswd -x -D "$DN" -w $PW -s user1 \
  "mail=user1@domain1.example, o=domain1.example, o=hosting, dc=myhosting,
dc=example"
Result: Success (0)
```

Other domains and users may be added with similar LDIF files. Creating LDIF files by hand can be cumbersome and error prone. We will talk about alternatives for administration in Section 5.

Postfix

We'll only cover the sections of Postfix that pertain to the mail hosting. To deal with other parts of Postfix setup, please visit the Postfix web page[14]. However, we do recommend that you set up as much of Postfix as you can to run in a chroot environment. In our experience, this means all of the Postfix daemons except pipe, local, and virtual.

Compiling Postfix with LDAP

This is covered fully in README_FILES/LDAP_README[15] that comes with the Postfix source. We'll just cover it briefly here.

Download the Postfix source and untar it. You need to rebuild the Postfix Makefiles to be aware of LDAP and link against it. To do this, execute the following command.

```
% make makefiles CCARGS="-I/usr/local/include -DHAS_LDAP" \
AUXLIBS="-L/usr/local/lib -lldap -L/usr/local/lib -llber"
```

At this point, follow the normal Postfix compiling and installing instructions as documented in its INSTALL file.

Configuring Postfix

While configuring Postfix for this task, we'll be mostly concerned with `/etc/postfix/main.cf`.

For most of the Postfix configuration, you will configure this in a way that makes the most sense for your site and you can follow the documentation contained in the Postfix source or on the Postfix web page[14]. In this document, we'll talk about the settings that are unique to and/or affected by this setup. If any of the configuration examples shown below aren't explicitly attributed to a specific file, assume they would be found in main.cf.

Procmail

Having Postfix use procmail for delivery is easy. All you need to do is define the `mailbox_command` parameter in `main.cf`.

```
mailbox_command = /usr/bin/procmail
```

The transport map

The transport table maps domains to message delivery transports (as specified in `/etc/postfix/master.cf`) and/or relay hosts. For our virtual domains, we want to map them to the virtual delivery agent that comes with Postfix. A transport table could look something like this.

```
domain1.example           virtual:
anotherdomain.example     virtual:
domain2.example           virtual:
```

After making your transport table in plain text, you need to make it into a binary DB file. To do this, you run the postmap program against the text file.

```
% cd /etc/postfix
% postmap transport
```

postmap can create three different database files: btree, dbm, and hash. For more information on this, see the postmap(1) man page. It should be noted that hash is the default on Linux systems.

At this point, you need to tell Postfix that there is a transport table and where to find it. You also need to let Postfix know that we accept mail for those domains. This is done through the transport_maps and mydestination directives in `main.cf`.

```
transport_maps = hash:/etc/postfix/transport
mydestination = $myhostname, localhost.$mydomain, $mydomain, $transport_maps
```

This information could be stored in LDAP as well, in theory. However, we aren't aware of an object class that fits our needs here. For now, we feel safer doing it as a map file.

Configuring LDAP sources

You can easily define multiple LDAP sources. LDAP source parameters are documented in README_FILES/LDAP_README[15]. The parameter names follow the pattern of ldapsource_parameter. The LDAP source name is defined when it is first used. In main.cf, you'll need one LDAP source definition per each lookup.

Aliases

```
aliases_server_host = localhost
aliases_search_base = o=hosting,dc=myhosting,dc=example
aliases_query_filter = (&(mail=%s)(objectClass=CourierMailAlias))
aliases_result_attribute = maildrop
aliases_bind = no
aliases_cache = yes
```

This first LDAP source definition is for virtual aliases. We've named this LDAP source aliases. In our configuration, as specified by the `server_host` line, our LDAP server is running on `localhost`. Our search base is the top of the hosting subtree we defined in our LDAP server. We're querying for items where the mail elements matches the email recipient as well as items that are of the *courierMailAlias* object class. The destination of the alias is the maildrop attribute. We do not want to bind to the LDAP server, we just want to do an anonymous lookup. Also, to help us over performance humps, we'll cache the results. The default cache has a life time of 30 seconds and has a size of 32K.

Accounts

```
accounts_server_host = localhost
accounts_search_base = o=hosting,dc=myhosting,dc=example
accounts_query_filter = (&(mail=%s)(objectClass=CourierMailAccount))
accounts_result_attribute = mailbox
accounts_cache = yes
accounts_bind = no
```

The accounts source is very similar to our aliases source. The big difference here is that we're looking for entries that have an object class of *courierMailAccount* and we're interested in the mailbox attribute of the resulting match.

The virtual alias maps

Now that the aliases LDAP source is defined, we need to let Postfix know to use it. This is taken care of using the `virtual_maps` parameter in `main.cf`

```
virtual_maps = ldap:alias
```

The virtual accounts

Telling Postfix about the virtual accounts is a bit trickier than the aliases. This is due to the fact that we need to define a lot of extra information about the virtual mail storage.

For this example, we assume that there is a vmail Unix account created that has a UID of 101, a GID of 101, and its home directory is `/home/vmail`. We will use the home directory of the vmail user as the place where we store our virtual mail repository.

```
virtual_mailbox_base = /home/vmail/domains
virtual_mailbox_maps = ldap:accounts
virtual_minimum_uid = 101
virtual_uid_maps = static:101
virtual_gid_maps = static:101
```

Most of the above is pretty straight forward, except for `virtual_minimum_uid`, `virtual_uid_maps`, and `virtual_gid_maps`. The Postfix documentation states „[virtual_minimum_uid] specifies a minimum UID that will be accepted as a return from a virtual_uid_maps lookup. Returned values less than this will be rejected, and the message will be deferred.“[16] Since we have decided that all mail for virtual accounts will be stored using the vmail Unix account, we set the `virtual_minimum_uid` to be the UID of vmail. Also, we set the `virtual_uid_maps` and `virtual_gid_maps` to a special static map and hard code it to the UID and GID of the vmail user. All of the parameters shown here are fully documented in README_FILES/VIRTUAL_README[16] that comes with the Postfix source.

We also need to edit the `local_recipient_maps` parameter to look at the `virtual_mailbox_maps` so Postfix knows who is a user our mail server supports and who is not. The main reason we do this is so Postfix can reject mail for unknown users.

```
local_recipient_maps = $alias_maps unix:passwd.byname $virtual_mailbox_maps
```

Courier

You do not need to follow any special instructions for installing Courier, so please see its documentation for full instructions. It should auto-detect LDAP and build it in. You should seriously consider passing the `–enable-workarounds-for-imap-client-bugs` option to `./configure`, otherwise Netscape mail users may have trouble interacting with your server. Yeah, this bends the IMAP protocol a little bit, but it's better to have happy users than a perfect protocol with unhappy users.

Configuring the Authentication Daemon

Courier uses an authentication daemon to keep authentication separate from the other parts of the system. We need to configure it so that a valid email user is either found in LDAP or in PAM. You specify this in authdaemonrc using the `authmodulelist` parameter:

```
authmodulelist="authldap authpam"
```

Configuring LDAP

All LDAP parameters are in `authldaprc`. Most parameters are self explanatory. To use the Courier schema, you actually have a few modifications to make, though. You also need to map all virtual users to the vmail account. Here is a summary of the updates you need to make to authldaprc:

```
LDAP_GLOB_UID           vmail
LDAP_GLOB_GID           vmail
LDAP_HOMEDIR            homeDirectory
LDAP_MAILDIR            mailbox
LDAP_CRYPTPW            userPassword
```

Three other settings we must concern ourselves with is `LDAP_AUTHBIND`, `LDAP_BINDDN`, and `LDAP_BINDPW`. These settings relate to authenicating the user. `LDAP_AUTHBIND` is mutually exclusive with `LDAP_BINDDN` and `LDAP_BINDPW`. We recommend using `LDAP_AUTHBIND`. A comment in authldaprc mentions a memory leak in OpenLDAP when using `LDAP_AUTHBIND`, but it has been fixed in OpenLDAP version 2.0.19.

If you use `LDAP_BINDDN` and `LDAP_BINDPW`, you are limited to the crypt, MD5, and SHA algorithms for passwords. SMD5 and SSHA are not available. Also, you also must put the root LDAP password in clear text in `authldaprc` when defining `LDAP_BINDPW`. There are security issues with putting the root LDAP password in clear text, so you should defintely use `LDAP_AUTHBIND` if you can.

The last change you need to do is to enable the IMAP server by setting the `IMAPDSTART` parameter to YES. You should now be able to use the `courier-imap.sysvinit` startup script to start and stop the IMAP daemon.

Setting up IMAP over SSL

If you had OpenSSL installed when compiling Courier, you will have support for IMAP over SSL. Courier can either do SSL using the STARTTLS extension or use a separate port for SSL connections. We prefer not to use STARTTLS, and have separate ports for SSL and non-SSL connections. You also need to give Courier the full path to your certificate file. And finally, you need to enable the SSL daemon, as it is off by default. Here is how we modified our imapd-ssl:

```
IMAPDSSLSTART=YES
IMAPDSTARTTLS=NO
TLS_CERTFILE=/usr/local/share/imapd.pem
```

SquirrelMail

Since SquirrelMail works directly against IMAP it does exactly what we want it to out of the box. Consider it to be the same as any other IMAP client. There is nothing special about SquirrelMail's setup in this configuration, just follow its install documentation.

We recommend that you set up SquirrelMail on a webserver that is running SSL. This will allow you to make sure that passwords are not going across in the clear.