Extending Use of Sendmail and Procmail for Virtual Mail Hosting

Randolph Langley
langley@cs.fsu.edu

April 2, 2007

Postfix [Ven] natively handles virtual domains and users better than Sendmail does. Perhaps the most needling problem with Sendmail's handling of virtual users and domains is the problem of "name folding": Sendmail doesn't properly allow for the same username in two different domains. If Sendmail tries to handle a single mail that has an envelope with both user `fred@home.com` and `fred@away.com`, it will just deliver the message to one of the users.

Also, although Sendmail lets you use an LDAP database, it doesn't allow you to directly store domain and user configuration in a relational database. One way to get around that is by using a relational database as a back-end for OpenLDAP, creating the metadata for that takes a good bit of effort and in practice it is not very fast.

I suggest in the final section a different means to access to a database via **sendmail**'s new socket map feature. That will allow you to keep all user and domain information in a database. With some caching and failover capabilities, it also allows a good bit of redundancy to be introduced.

The Sendmail Consortium's website has a very good introduction to using **sendmail** to host virtual domains at **http://www.sendmail.org/virtual-hosting.html**. The approach outlined here extends that approach in making a mail-hub:

1. It uses two cooperating sendmail processes, an "incoming" and an "outgoing" sendmail to solve the name "folding" problem.
2. It uses **procmail** to deliver either into mbox INBOXes (or into Maildirs) divided first by domain and then by username.
3. It uses dovecot 0.99 and squirrelmail for its clients.

4. Virtual user information is managed in a relational database, much like what is possible with Postfix's virtual users.

1 Other Approaches

The proposed approach is in contrast to such sendmail setups as the old userdb approach [AAS, pp. 83–85, section 5.12] (which has been deprecated in favor of virtusertable which is part of the outlined approach) and approaches where virtual domains are much more "real" in the sense of having separation to the point of having individual password files, mail queues, and other operating system level entities individuated. **http://howtos.linux.com/howtos/Virtual-Services-HOWTO-1.shtml**

While I haven't seen that approach taken to its logical extreme of actually having a single full virtual machines handling each virtual domain under, say, Xen or UML, it would also seem to be a possible solution.

2 The Basics

While the approach outlined here allows for virtual users who are only entries in a relational database, the hosted email itself remains in ordinary directories and files divided first by *DOMAIN* and then by *USER*. It would be possible to extend this method to directly deliver email into a database by changing the local delivery invocation to do so, but until client software is generally available to take advantage of email found in a standard relational database, the traditional idea of delivering incoming email either into a mbox INBOX or into a Maildir is more useful.

As **MailScanner** [Fie05] elegantly demonstrates, configuring running dual **sendmail** processes also provides a natural point for scanning email for spam and malware via separate incoming and outgoing queues. It turns out that it is also useful for solving the name folding problem.

3 Configuration

In the Sendmail Consortium recommended configuration, the MTA handles incoming email, it does local delivery, and it also directly relays any email that is destined to go off-box.

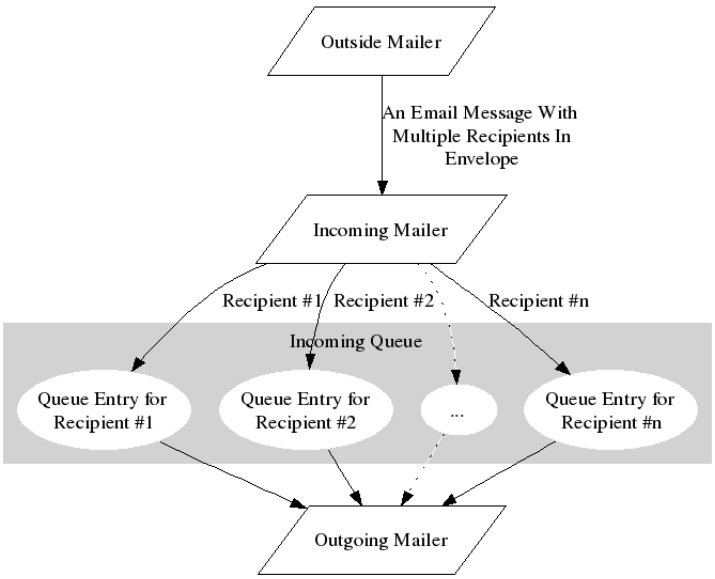

Figure 1: Splitting envelopes of multi-recipient email

When one utilizes **MailScanner**, the setup varies from the the Sendmail Consortium recommended setup by processing all email with **MailScanner** rather than just having the MTA immediately forward off-box email. In the approach developed here, even if we don't use **MailScanner**, then we will still force all email into a local processing queue rather than immediately forwarding email for domains that we do relay for. The reasons for this are

1. The incoming MTA **sendmail** process will not actually know about the virtual domains as local domains; it simply thinks that it is relaying email for the virtual domains, despite the fact that it is putting it into a queue rather doing the relay itself.
2. Since we force splitting of all envelopes into separate deliveries of disparate messages (see Figure 1), this splitting separates email in the same envelope with problematic addresses of the form *USER@DOMAIN1* and *USER@DOMAIN2*.

We will call the .mc file for the incoming processor *sendmail_in.mc* and for the outgoing processor *sendmail_out.mc*. (In the Sendmail Consortium recommendation, relaying is done by the MTA using *sendmail.cf*, rather than the MailScanner concept of placing incoming mail into */var/spool/mqueue.in* to first be

scanned. MailScanner does this by explicitly setting the option **-ODelivery-Mode=queueonly**, which forces all email to be queued, and implicitly by not turning on the usual **-q$QUEUETIME** option when the MTA **sendmail** is started, which prevents further processing by the incoming **sendmail** daemon.)

The particular idea behind handling local delivery is one that I first found in an email message where Mr. Andrew Fresh elucidated the idea of utilizing a pseudo-domain called *.virtuser.*, albeit in the email message this utilization was not in the context of two separate **sendmail** handlers. [1]

The crux of the scheme is to have each valid userid *USER@DOMAIN* listed in the virtusertable, mapped to *USER@DOMAIN.virtuser.* Those are then handled by **sendmail**'s ruleset 0 picking up the *.virtuser.* domain as one that invokes a particular local delivery. This local delivery will be via **procmail** invoked with arguments that allow **procmail** to easily deliver the email into the correct box. (In contrast, in the scheme presented at **howtos.linux.com** each virtual host is configured into ruleset 0.)

In the Fresh email, the mail was to be delivered to virtual mbox mailboxes distributed as */var/spool/vbox/DOMAIN/USER*, which allows for overlapping names in the various virtual domains to be handled cleanly. In this article, we discuss this and also discuss using Maildirs for delivery.

Four of the problems that needed to be solved were:

1. Those of pure virtual users, who have no reference in a password file.
2. Easy management of those pure virtual users, since the usual system administration tools are not set up to handle them.
3. Name "folding"; **sendmail** by default will, if it believes that *DOMAIN1* and *DOMAIN2* are both local domains it handles, will "fold" email to *USER@DOMAIN1* and *USER@DOMAIN2*, and deliver only one copy of the email.
4. Handling aliases and forwards.

4 Configuring sendmail_in.mc

If you are doing this along with a **MailScanner** installation, then as is usual in **MailScanner** installations, you should have lines in *sendmail_in.mc* along the lines of:

```
define('QUEUE_DIR', '/var/spool/mqueue.in')dnl
QUEUE_GROUP('mqueue', 'P=/var/spool/mqueue.in,
   F=f, r=1, R=8, I=2m')
```

If you are doing this without **MailScanner**, you would need to define your mqueue like this in *sendmail_in.mc*:

```
QUEUE_GROUP('mqueue', 'P=/var/spool/mqueue,
   F=f, r=1, R=8, I=2m')
```

The previous **QUEUE_GROUP** settings establish that you want to have your mail placed into */var/spool/mqueue.in* for **MailScanner** to find and handle, or in the ordinary queue */var/spool/mqueue* for a **sendmail**-only installation. Using a **QUEUE_GROUP** named **mqueue** means that this is defined for the default queue group for this **sendmail** process. (A third way would be to use both */var/spool/mqueue.in* and */var/spool/mqueue*, and have another **sendmail** daemon configured like the MSP version to move mail between the two queues.)

The **r=1** setting is particularly critical to our efforts – in order to handle the "folding" problem, we need to split our emails that are in a single envelope so that email to the same username *USER* but different domains, e.g. *smith@domain1.com* and *smith@domain2.com*, does not get delivered to just one mailbox.

Setting **r=1** sets the maximum number of recipients to 1 by the incoming **sendmail**. That means that if we send an email with n recipients in the envelope that it is split in n separate queue files.[AAS, pp. 54,81]

(Just to explicate the other flags, **F=f** means the "fork" flag is set so that **sendmail** will fork off parallel queue runner processes, **R=8** means there should be 8 parallel queue runners for this queue group, and **I=2m** means that the queue should be processed every two minutes. [AAS, pp. 81-82])

You also need to make sure that you aren't using the same */etc/mail/access.db* that the outgoing .mc file uses since we want the incoming process to think that it is RELAYing for the domain. To avoid confusion, we use */etc/mail/access_in* rather than the standard */etc/mail/access*.

```
FEATURE('access_db','hash -T<TMPF>
   /etc/mail/access_in')dnl
```

You must also make sure that you have **dnl**-ed any reference to */etc/local/local-host-names* that could confuse the incoming process, which needs to believe it is merely relaying. This is done via the *cw* file (named in reference to the .cf file's idea of **w** class of equivalent names for a machine, although these days it is now known as */etc/mail/local-host-names* rather than */etc/sendmail.cw*):

```
dnl FEATURE(use_cw_file)dnl
```

5 Configuring sendmail_out.mc

Here are the lines:

```
VIRTUSER_DOMAIN_FILE('-o
   /etc/mail/virtuserdomains')dnl
FEATURE('virtusertable',
   'hash -o /etc/mail/virtusertable.db')dnl
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl
dnl #
dnl #  THESE ARE USED IFF YOU DELIVER
dnl #         VIA LOCAL_RULE_0
dnl #
define('PROCMAIL_MAILER_PATH','/usr/bin/procmail,
   U=vmail:vmail')dnl
define('PROCMAIL_MAILER_ARGS',
   'procmail -t -Y -a $h -a $u')dnl
define('PROCMAIL_MAILER_FLAGS',
   'l0')dnl
dnl #
dnl #   THIS IS USED IFF YOU FALL THROUGH AND
dnl #     *DO*NOT*DELIVER* VIA LOCAL_RULE_0
dnl #
FEATURE(local_procmail,'/usr/bin/procmail',
   'procmail -t -Y -a $h -d $u')dnl
FEATURE('access_db','hash -T<TMPF>
   -o /etc/mail/access_out.db')dnl
#
#
#
MAILER(procmail)dnl
LOCAL_RULE_0
R$* < @ $* .virtuser. > $*
               $#procmail $@ $2 $: $1
```

(You can find a similar idea at **http://info.ccone.at/INFO/Mail-Archives/procmail/Jul-2002/msg00361.html**)

This is slightly intricate in order to achieve two types of functionality:

1. If we are receiving mail for a virtual domain via a .virtuser. mapping in the virtusertable, we want to pass the arguments via a pair of **-a** options, yielding arguments $1 and $2 in our **procmail** script. **$#procmail** uses the **define(PROCMAIL_MAILER_)** definitions to obtain its settings.

2. If we are receiving mail locally for some reason (such as for local root email delivery), then we fall back to the usual procmail invocation from **sendmail** using **-a** and **-d** options to indicate a */var/spool/mail/* mbox delivery. (**-d** indicates explicit delivery to a mailbox.) Local delivery uses **FEATURE(local_procmail)** to obtain its settings.

6 Configuring procmail

The easiest place to configure procmail is most likely */etc/procmailrc*, and that's where I put this code:

```
DOMAIN="<$1>"
RECIPIENT="<$2>"
ROOTHOMEDIR=/home/vmail-users
ROOTINBOXDIR=/var/spool/vbox

:0
* RECIPIENT ?? ()\/[^<]*@
* MATCH ?? ()\/.*[^@]
{
   USER=$MATCH
}

:0 a
* DOMAIN ?? ()\/[^<].*[^>]
{
   DOMAINNOBRACKET=$MATCH
}

:0 a:
${ROOTINBOXDIR}/${DOMAINNOBRACKET}/${USER}
```

That works for mboxes; for Maildirs (possible since **procmail** 3.15) located in home directories, the file is only different in the last two lines:

```
:0 a
${ROOTHOMEDIR}/${DOMAINNOBRACKET}/${USER}
```

The elimination of the final ":" is not a typo – it simply tells **procmail** that we don't need to lock the output file.

7 Configuring dovecot 0.99

7.1 PostgreSQL

The easy way to handle these virtual users is to keep their master data in a database. For instance, if you want to use PostgreSQL you can create a file */usr-/local/etc/dovecot-PostgreSQL.conf* that just has this configuration information:

```
db_host = [DBHOST]
db_port = 5432
db = [DATABASE]
db_user = [DBUSER]
db_passwd = [DBPASSWD]
db_client_flags = 0
```

where *[DBHOST]*, *[DATABASE]*, *[DBUSER]*, and *[DBPASSWD]* are all set to get you to a user database with a table that has at least fields for a username and a password. If you are going to use Maildirs, it would probably be a good idea to also have the name of the Maildir in there also.

7.2 mbox format

For mbox INBOXes, you can just define the following three lines in */etc/dovecot-.conf* something like this:

```
default_mail_env = mbox:/home/vmail-users/%d/%n/mail
   :INBOX=/var/spool/vbox/%d/%n
auth_passdb = PostgreSQL /usr/local/etc/dovecot-PostgreSQL.conf
auth_userdb = static uid=502 gid=502
   home=/home/vmail/%d/%n
```

and in /usr/local/etc/dovecot-PostgreSQL.conf, define something like

```
password_query = SELECT password FROM users
   WHERE username = '%u'
```

7.3 maildir format

If you want to use Maildirs, you can configure them this way:

```
default_mail_env = maildir:%h/Maildir
auth_passdb = PostgreSQL
   /usr/local/etc/dovecot-PostgreSQL.conf
auth_userdb = PostgreSQL
   /usr/local/etc/dovecot-PostgreSQL.conf
```

and in */usr/local/etc/dovecot-PostgreSQL.conf*

```
password_query = SELECT password FROM users
   WHERE username = '%u'
user_query = SELECT home, uid, gid FROM users
   WHERE username = '%u'
```

8 Squirrelmail configuration

Setting up squirrelmail once you have dovecot requires only adding these lines to */etc/squirrelmail/config_local.php*:

```
$imap_server_type               = 'courier';
$optional_delimiter             = '.';
$default_folder_prefix          = '';
```

9 Creating your database users table

Now you are up to creating your users table. A schema with fields as simple as these are adequate:

```
username  varchar(60)
password  varchar(32)
userpart  varchar(60)
domain    varchar(60)
```

You can define username to be the primary key and to not allow password, domain, and userpart to be NULL.

If you want to allow mail forwarding and aliases, you can add the following fields:

```
forward   varchar(128)
alias_for varchar(128)
```

It would be better practice to also define a domain table and make the field **domain** a foreign key.

10 Handling Users and Domains

Now you are set up, and ready to create your domains and your users. Let's suppose that you want to add domain *somedomain* with a user *user1@somedomain*.

10.1 Creating a new domain

You need to add support for this domain in */etc/mail/virtuserdomains*:

```
somedomain
somedomain.virtuser
somedomain.virtuser.
```

You need to have access_in (for *sendmail_in.mc*) understand that email is handled for this domain:

```
somedomain         RELAY
```

and then remake the access_in.db file:

```
cd /etc/mail ; make access_in.db
```

And finally, you need to have this added to class **w** for *sendmail_out.mc*; that's easily done by adding to */etc/mail/local-host-names*

```
somedomain
```

10.2 Creating a new user

The process of creating virtual user *user1@somedomain* has three steps:

10.2.1 Database

You need to add a row in the *users* table for this user:

```
INSERT INTO users (username,password,userpart,domain)
   VALUES ('user1@somedomain','PASSWORD',
           'user1','somedomain');
```

10.2.2 Files

You need to add an entry into */etc/mail/virtusertable* for this user:

```
user1@somedomain        user1@somedomain.virtuser.
```

and then remake the *.db* file:

```
cd /etc/mail ; make virtusertable.db
```

You need to create a home directory for the user:

```
mkdir /home/vmail-users/somedomain/user1
chown vmail:vmail /home/vmail-users/somedomain/user1
```

If you are using mbox INBOXes, you will also need a vbox for your user:

```
touch /var/spool/vbox/somedomain/user1
chown vmail:vmail
   /var/spool/vbox/somedomain/user1
```

10.2.3 Special Considerations: Aliases, Forwards, and Disabling Accounts

Setting a simple mail forward is as simple as changing the */etc/mail/virtusertable* to do the forward. For instance, you can change the usual entry for *user1@-domain1.com* from

```
user1@domain1.com    user1@domain1.com.virtuser.
```

to

```
user1@domain1.com    another-user@another-domain.com
```

Likewise, creating a simple alias can be done via the */etc/mail/virtusertable*:

```
alias@domain1.com          user1@offsitedomain.com
anotheralias@anotherdomain user2@domain1.com.virtuser.
```

The first takes a local alias and sends it off to another site; the second takes an alias and points it toward a local delivery via the pseudo-domain *.virtuser.*

If you need to for some reason to disable an account from receiving email, you can use */etc/mail/access_in*:

```
user@domain             550 Mailbox not available
```

Finally, if you want to suspend a user from reading email but still receive it while the user is in suspension, you can do this by just having the database return an alternative home directory that is not owned or writable by user and group *vmail*. If you are using mbox INBOXes rather than Maildirs, then you will need to also set up an non-writable INBOX.

Then populate that "suspension" area with a mail message explaining the problem. Since the user cannot read or write to the mailbox, the user cannot modify this area; it doesn't interfere with the user's ability to continue to receive email normally, just with the user's ability to read the email.

11 An alternative approach to handling users and domains

One possibility for handling users and domains is rather than modifying datafiles based on information in the PostgreSQL database, you can could use the new socket maps in **sendmail** to directly access them. You can use the Perl script **sockmapClient.pl** in sendmail's **contrib/** directory as a base to build your connector on.

To make sendmail use the maps, in sendmail_out.mc remove the lines for access_db, virtusertable, and virtuserdomains for change the specifications:

```
LOCAL_CONFIG
Kvirtuserdomains socket unix:/SOMEPATH/SOMESOCKETNAME_OUT
Kvirtusertable socket unix:/SOMEPATH/SOMESOCKETNAME_OUT
Kaccess_db socket unix:/SOMEPATH/SOMESOCKETNAME_OUT
```

and in sendmail_in.mc, remove the lines for access_db and use

```
LOCAL_CONFIG
Kaccess_db socket unix:/SOMEPATH/SOMESOCKETNAME_IN
```

References

[AAS] Eric Allman, Claus Assmann, and Gregory Neil Shapiro. Sendmail installation and operation guide. Version 8.706 for Sendmail 8.13.

[Fie05] Julian Field. *MailScanner: a guide to the world's most widely-used e-mail security and anti-spam system*. Julian Field, 2005.

[Ven] Wietse Venema. postfix.org (website). `http://www.postfix.org`.