

*By Tim*

Published: 2007-02-01 18:09

## Virtual Users With Postfix, PostfixAdmin, Courier, Mailscanner, ClamAV On CentOS

Written by Tim Haselaars, Trinux.

In this how to I will explain how to setup a Postfix virtual mailserver with Courier-IMAP, Maildrop and Postfix Admin GUI. We will secure our mailserver with Mailscanner and Clamav as anti-virus and Spamassassin as anti-spam.

3 parts:

- Installation of all software
- Configuration of mail server it self
- Configuration of anti-spam and anti-virus

I have written this tutorial, because it was very difficult to find a decent œhow-to• on how to configure a mail server on a Linux distribution like CentOS.

I have written this tutorial for CentOS 4.4 X86\_64, but I should work on all CentOS 4.4 distributions (i386, i) and Redhat-like clones.

First of all, it was a long painful road to walk, because as you all should know. CentOS is not the most progressive distribution, but never the less a very secure and stable one.

### Installation

Let(TM)s start with a minimal installation of CentOS. Look at the tutorial ~The Perfect Setup - CentOS 4.4(TM) ( [http://www.howtoforge.com/perfect\\_setup\\_centos\\_4.4](http://www.howtoforge.com/perfect_setup_centos_4.4)), with MySQL up and running.

Next we are going to install all basic needed packages.

```
yum install rpm-build pcre-devel
```

Next comes the Cyrus sasl packages, needed for the encrypted authentication.

```
yum install cyrus-sasl-sql cyrus-sasl-devel
```

Default has CentOS installed a couple of cyrus packages, we need the basic cyrus sasl packages but there are a couple of packages that can be deleted.

```
yum remove cyrus-sasl-gssapi.x86_64
```

We are going to use some packages that can(TM)t be found in the base repository of CentOS. Therefore we add the DAG repository. The DAG repository is a more progressive, but stable repository for CentOS. Here you can find some extra rpm's.

```
vi /etc/yum.repos.d/dag.repo
```

add the following lines.

```
[dag]
name=Dag RPM Repository for Red Hat Enterprise Linux
baseurl=http://apt.sw.be/redhat/el$releasever/en/$basearch/dag
gpgcheck=1
enabled=0
```

By default I have disabled (*enabled=0*) this repository, so you won(TM)t update any packages that you don(TM)t want to.

Next package will be our MTA (mail transport agent). I have chosen for Postfix, instead of the default Sendmail or the Qmail. I find Postfix easy to configure and stable.

I will install Postfix with MySQL support, because most of the mail server data will be stored in the MySQL database. This make it easier to maintain and manage.

```
yum install --enablerepo=centosplus postfix
```

As POP3/IMAP we have chosen for Courier. Again with the same reason, it(TM)s clean, quick and stable. It also provides MySQL support so that(TM)s handy. A downside is that doesn(TM)t come with Centos, so we are going to build its RPM(TM)s ourselves.

We start by making a non-root user. This will be needed to build some Courier packages and this user will own all the virtual mail. I have chosen for the user vmail.

```
groupadd vmail -g 1001  
useradd vmail -u 1001 -g 1001
```

It might be necessary to add this user temporary to the sudoers file.

```
vi /etc/sudoers
```

add the following line to sudoers file

```
vmail ALL=(ALL) ALL
```

Next we switch to this user

```
su vmail  
sudo yum install libtool postgresql-devel gdbm-devel pam-devel expect openldap-devel
```

These are dependencies for courier-authlib, so first install these. These should be normally available in the centos base repo.

```
sudo yum install gamin-devel openldap-servers
```

These are dependencies for courier-imap. These should be normally available in the CentOS base repo.

### Create RPM build directories

```
mkdir $HOME/rpm  
  
mkdir $HOME/rpm/SOURCES  
  
mkdir $HOME/rpm/SPECS  
  
mkdir $HOME/rpm/BUILD  
  
mkdir $HOME/rpm/SRPMs  
  
mkdir $HOME/rpm/RPMS  
  
mkdir $HOME/rpm/RPMS/i386
```

Finally:

```
echo "%_topdir    $HOME/rpm" >> $HOME/.rpmmacros
```

Next we make a directory where we store all our downloads.

```
mkdir $HOME/downloads  
cd $HOME/downloads
```

And start downloading the necessary courier packages.

```
wget http://surfnet.dl.sourceforge.net/sourceforge/courier/courier-authlib-0.58.tar.bz2
```

```
wget http://surfnet.dl.sourceforge.net/sourceforge/courier/courier-imap-4.1.1.tar.bz2
```

```
wget http://surfnet.dl.sourceforge.net/sourceforge/courier/mailedrop-2.0.2.tar.bz2
```

Start with installing the authlib. The Courier Authentication Library is a generic authentication API that encapsulates the process of validating account passwords. In addition to reading the traditional account passwords from `/etc/passwd`, the account information can alternatively be obtained from an LDAP directory; a MySQL or a PostgreSQL database; or a GDBM or a DB file. The Courier authentication library must be installed before building any Courier packages that needs direct access to mailboxes (in other words, all packages except for courier-sox and courier-analog).

```
sudo rpmbuild -ta courier-authlib-0.58.tar.bz2
```

Next enter the root password.

After compiling:

```
cd $HOME/rpm/RPMS/x86_64
```

Install the ones you need:

```
sudo rpm --install courier-authlib-0.58-1.x86_64.rpm
```

```
sudo rpm --install courier-authlib-devel-0.58-1.x86_64.rpm
```

```
sudo rpm --install courier-authlib-mysql-0.58-1.x86_64.rpm
```

Next we are gonna compile the courier-imap server.

Make sure the your user has WRITE access to `$HOME/rpm/RPMS/x86_64` and other directories that the build script might need (else `sudo chmod -R 777 $HOME/rpm/RPMS/`)

```
cd $HOME/downloads
rpmbuild -ta courier-imap-4.1.1.tar.bz2

cd $HOME/rpm/RPMS/x86_64

sudo rpm --install courier-imap-4.1.1-1.4.x86_64.rpm
```

At last we gonna install the maildrop, which is used to filter incoming mail and drop it at the correct mail directory.

```
cd $HOME/downloads

sudo rpmbuild -ta maildrop-2.0.2.tar.bz2

cd $HOME/rpm/RPMS/x86_64

sudo rpm --install maildrop-2.0.2.x86_64.rpm
```

**Create SSL certificates** SSL certificates will be used by Postfix (for SMTPS and TLS), Courier (for IMAPS and POP3S) and Apache (for HTTPS). We store all the certificates in one directory.

```
mkdir /usr/local/ssl
cd /usr/local/ssl
```

Generate the RSA private-key for the server. We don't want a pass phrase on this key, otherwise it will need to be entered every time courier/apache/postfix starts.

```
openssl genrsa -out mail.yourdomain.com.key 1024
```

*Generating RSA private key, 1024 bit long modulus*

```
.....+++++
.....+++++
e is 65537 (0x10001)
```

Tighten the permissions on this key file:

```
chmod 600 mail.yourdomain.com.key
```

Generate a certificate request:

```
openssl req -new -key mail.yourdomain.com.key -out mail.yourdomain.com.csr
```

*You are about to be asked to enter information that will be incorporated into your certificate request.*

*What you are about to enter is what is called a Distinguished Name or a DN.*

*There are quite a few fields but you can leave some blank*

*For some fields there will be a default value,*

*If you enter '.', the field will be left blank.*

-----

*Country Name (2 letter code) [GB]:*

*State or Province Name (full name) [Berkshire]:*

*Locality Name (eg, city) [Newbury]:*

*Organization Name (eg, company) [My Company Ltd]:*

*Organizational Unit Name (eg, section) []:*

*Common Name (eg, your name or your server's hostname) []:*

*Email Address []:*

*Please enter the following 'extra' attributes to be sent with your certificate request*

*A challenge password []:*

*An optional company name []:*

At this point you would send your CSR off to a Certificate Authority for signing (such as Verisign or Thawte) . However if you wanted to do some in-house testing, we can set ourselves up as a CA, and then sign the CSR ourselves :

Generate RSA private-key for the CA:

```
openssl genrsa -des3 -out ca.key 1024
```

```
Generating RSA private key, 1024 bit long modulus
```

```
.....+++++
```

```
.....+++++
```

```
e is 65537 (0x10001)
```

```
Enter pass phrase for ca.key:
```

```
Verifying - Enter pass phrase for ca.key:
```

Tighten permissions on this private key:

```
chmod 600 ca.key
```

Create a self signed CA certificate:

```
openssl req -new -x509 -days 365 -key ca.key -out ca.crt
```

```
Enter pass phrase for ca.key:
```

```
You are about to be asked to enter information that will be incorporated  
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
```

```
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
```

```
----- Country Name (2 letter code) [GB]:
```

```
State or Province Name (full name) [Berkshire]:
Locality Name (eg, city) [Newbury]:
Organization Name (eg, company) [My Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:
```

Use this test CA to sign our server cert:

```
openssl x509 -req -days 365 -CA ca.crt -CAkey ca.key -set_serial 01 -in mail.yourdomain.com.csr -out mail.yourdomain.com.crt
```

Signature ok

```
subject=/C=AU/ST=SomeState/L=SomePlace/O=Test CA Company/OU=SomeGroup/CN=CA Signing Biz/emailAddress=postmaster@nowhere
Getting CA Private Key
Enter pass phrase for ca.key:
```

Combine the server key and certificate into a single file. Postfix and Apache can deal with two separate files, but Courier needs them both in one. To try and keep things consistent we will use a single file with all 3 apps. Create the pem file in the format that courier wants (both the key and the cert in one file):

```
cat mail.yourdomain.com.key mail.yourdomain.com.crt > mail.yourdomain.com.pem
```

```
chmod 600 mail.yourdomain.com.pem
```

OK so you should now have something like this :

```
ls -al
```

```
total 36
```

```
drwxr-xr-x  2 root root 4096 Nov 28 22:02 .
drwxr-xr-x 14 root root 4096 Nov 20 21:50 ..
-rw-r--r--  1 root root 1371 Nov 28 21:50 ca.crt
-rw-----  1 root root  963 Nov 28 21:47 ca.key
-rw-r--r--  1 root root 1001 Nov 28 21:51 mail.yourdomain.com.crt
-rw-r--r--  1 root root  773 Nov 28 21:45 mail.yourdomain.com.csr
-rw-----  1 root root  887 Nov 28 21:45 mail.yourdomain.com.key
-rw-----  1 root root 1888 Nov 28 22:02 mail.yourdomain.com.pem
```

## Configuration

Make sure you are the ROOT user again and not the mail user (*exit* or *su root*).

To make things easy, I would advice to install phpmyadmin (<http://www.phpmyadmin.net/>). This will make it easier to work with the MySQL database.

I choose to work with Postfix Admin (<http://high5.net/page7.html>), an excellent PHP, multi-user postfix GUI. Postfix Admin enables: forwarding, vacation, mailbox creation, !. But you are free to change the structure to what you want. Install Postfix Admin: download the latest package from the download page. Make sure that you are in your WWW directory and then unarchive the Postfix Admin archive (whatever the filename is):

```
tar -zxvf postfixadmin-2.*.*.tgz
```

Since the database password is stored in the *config.inc.php* it's a good idea to change the permissions for Postfix Admin.

```
cd postfixadmin

chmod 640 *.php *.css

cd postfixadmin/admin/

chmod 640 *.php .ht*

cd postfixadmin/images/
```

```
chmod 640 *.gif *.png

cd postfixadmin/languages/

chmod 640 *.lang

cd postfixadmin/templates/

chmod 640 *.tpl

cd postfixadmin/users/

chmod 640 *.php
```

## Mysql db structure

In `DATABASE_MYSQL.TXT` you can find the table structure for MySQL that you need in order to configure Postfix Admin and Postfix in general to work with Virtual Domains and Users. In `DATABASE_PGSQL.TXT` you can find the table structure for PostgreSQL.

```
mysql -u root [-p] < DATABASE_MYSQL.TXT
```

Check the `config.inc.php` file. There you can specify settings that are relevant to your setup.

Postfix Admin contains 3 views of administration. There is the Site Admin view, located at <http://www.yourdomain.com/postfixadmin/admin/>. There is the Domain Admin view, located at <http://www.yourdomain.com/postfixadmin/>. And there is the User Admin View, located at <http://www.yourdomain.com/postfixadmin/users/>.

In order to do the initial configuration you have to go to the Site Admin view.

The default password for the Site Admin view of Postfix Admin is `admin/admin`.

This is specified in the `.htpasswd` file in the `/admin` directory. Make sure that the location of the `.htpasswd` file matches your path.

You can make a new domain and a test mailbox.

## SASL (for SMTP-AUTH)

Next we are going to configure our Postfix so that I will use the `authdaemon`.

```
vi /usr/lib64/sasl2/smtpd.conf
```

```
# smtpd.conf
pwcheck_method: authdaemond
log_level: 3
mech_list: PLAIN LOGIN
authdaemond_path:/usr/var/spool/authdaemon/socket
```

**NOTICE:** [/usr/lib64/sasl2/smtpd.conf](#) For i386 architecture please use this: [/usr/lib/sasl2/smtpd.conf](#)

```
chown root.vmail /usr/lib64/sasl2/smtpd.conf
```

```
chmod 640 /usr/lib64/sasl2/smtpd.conf
```

Secure the `smtpd` config file. Courier's `authdaemond` socket and `pid` directory must be readable by Postfix:

```
chmod 755 /usr/var/spool/authdaemon/
```

## Postfix > master.cf

The `master.cf` file contains all the directives concerning the daemons and network settings.

```
vi /etc/postfix/master.cf
```

```
=====
# service type private unpriv chroot wakeup maxproc command + args
#          (yes) (yes) (yes) (never) (100)
# =====
smtp      inet  n       -       n       -       -       smtpd
#submission inet  n       -       n       -       -       smtpd
#  -o smtpd_etrn_restrictions=reject
#  -o smtpd_client_restrictions=permit_sasl_authenticated,reject
#smtps    inet  n       -       n       -       -       smtpd
# -o smtpd_tls_wrappermode=yes -o smtpd_sasl_auth_enable=yes
```

We want to make it possible to let user access the SMTP over SSL (smtps), so all we have to do is remove the comment in front of the smtps line like so.

```
=====
# service type private unpriv chroot wakeup maxproc command + args
#          (yes) (yes) (yes) (never) (100)
# =====
smtp      inet  n       -       n       -       -       smtpd
#submission inet  n       -       n       -       -       smtpd
#  -o smtpd_etrn_restrictions=reject
#  -o smtpd_client_restrictions=permit_sasl_authenticated,reject
smtps    inet  n       -       n       -       -       smtpd
-o smtpd_tls_wrappermode=yes -o smtpd_sasl_auth_enable=yes
```

We also want our postfix to listen on an extra port, because some ISP block the usage of the default smtp port (25). This is done to prevent spam. So add an extra rule right below the first rule of smtp with the port we want to use, in our case port 567. Also make sure your firewall has enabled this port.

```
=====
# service type private unpriv chroot wakeup maxproc command + args
```

```
#      (yes) (yes) (yes) (never) (100)
```

```
# =====
```

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

```
567  inet n  -  n  -  -  smtpd
```

```
#submission inet n  -  n  -  -  smtpd#  -o smtpd_etrn_restrictions=reject#  -o
```

```
smtpd_client_restrictions=permit_sasl_authenticated,rejectsmtps  inet n  -  n  -  -  smtpd -o smtpd_tls_wrappermode=yes -o
```

```
smtpd_sasl_auth_enable=yes
```

Make also sure the path to the maildrop binary is correct, so at the bottom of the file change

```
flags=DRhu user=vmail argv=/usr/local/bin/maildrop -d ${recipient}
```

into

```
flags=DRu user=vmail argv=/usr/bin/maildrop -d ${recipient}
```

## Postfix > main.cf

The *main.cf* file contains all the directives concerning the postfix settings.

```
vi /etc/postfix/main.cf
```

```
# make the following changes :
```

```
myhostname = mail.yourdomain.com
```

```
mydomain = yourdomain.com
inet_interfaces = all
mydestination = $myhostname, localhost.$mydomain, localhost
local_recipient_maps = proxy:unix:passwd.byname $alias_maps
mynetworks = $config_directory/mynetworks
##relayhost = [smarthost.isp.be] #if you have a smarthost server
relay_domains = mysql:/etc/postfix/mysql_relay_domains_maps.cf
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
# Next, add all these to the bottom of the file :
#
# Virtual Mail Mysql settings
#
virtual_alias_maps = mysql:/etc/postfix/mysql_virtual_alias_maps.cf
virtual_uid_maps = static:1001
virtual_gid_maps = static:1001
virtual_mailbox_base = /opt/mail
virtual_mailbox_domains = mysql:/etc/postfix/mysql_virtual_domains_maps.cf
virtual_mailbox_limit = 51200000
virtual_mailbox_maps = mysql:/etc/postfix/mysql_virtual_mailbox_maps.cf
virtual_minimum_uid = 1001
# Who handles the mail delivery?
# POSTFIX = virtual
# MAILDROP = maildrop
#
#virtual_transport = virtual
virtual_transport = maildrop
maildrop_destination_recipient_limit = 1
# Transport map
transport_maps = hash:/etc/postfix/transport
vacation_destination_recipient_limit = 1
# Additional for quota support
virtual_create_maildirsize = yes
```

```

virtual_mailbox_extended = yes
virtual_mailbox_limit_maps = mysql:/etc/postfix/mysql_virtual_mailbox_limit_maps.cf
virtual_mailbox_limit_override = yes
virtual_maildir_limit_message = Sorry, the user's maildir has overdrawn his disk space quota, please try again later.

#####
### ENABLE SASL SUPPORT ( SMTP-AUTH )
# smtpd_sasl_auth_enable = yes
# Enable SASL support in postfix
# smtpd_sasl_security_options = noanonymous
# Anonymous logins will not be permitted
# broken_sasl_auth_clients = yes
# Allow RFC-broken mail clients like Outlook Express4 to use SMTP AUTH
# smtpd_sasl_path = smtpd
# Tells SASL to get the config from /usr/lib64/sasl2/smtpd.conf
# smtpd_sasl_local_domain =
# If the user fails to nominate a domain, don't auto append one
# smtpd_sasl_authenticated_header = yes
# Include the authenticated username in the message headers.
# Having this on will make it easier if a spammer cracks one of your user's weak passwords,
# and starts using SMTP-AUTH to relay spam through your server
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
broken_sasl_auth_clients = yes
smtpd_sasl_path = smtpd
smtpd_sasl_local_domain =
smtpd_sasl_authenticated_header = yes
smtpd_recipient_restrictions =
permit_mynetworks,permit_sasl_authenticated,reject_non_fqdn_hostname,reject_non_fqdn_sender,reject_non_fqdn_recipient,reject_unauth_destination,reject_unauth_pipelining,reject_invalid_hostname,reject_rbl_client opm.blitzed.org,reject_rbl_client list.dsbl.org,reject_rbl_client bl.spamcop.net,reject_rbl_client sbl-xbl.spamhaus.org
#####
### ENABLE TLS SUPPORT ( "STARTTLS" ... enables SSL to be negotiated during a SMTP connection )
# smtp_use_tls = no

```

```
# dont enable TLS for outbound SMTP connections
# smtpd_use_tls = yes
# announce TLS availability for incoming SMTP connections
# smtpd_tls_auth_only = no :
# TLS is optional, not enforced
# smtpd_tls_key_file :
# specify the private key ( must not be encrypted - ie no password)
# smtpd_tls_cert_file :
# specify the certificate
# smtpd_tls_session_cache_database :
# nominate a server-side TLS session cache. Improves performance.
# smtpd_tls_loglevel = 1 :
# log basic TLS handshake and cert info
# smtpd_tls_received_header = yes
# record some protocol/cipher etc info in the Received header smtp_use_tls = no
smtp_use_tls           = no
smtpd_use_tls          = yes
smtpd_tls_auth_only    = no
smtpd_tls_key_file     = /usr/local/ssl/mail.yourdomain.com.key
smtpd_tls_cert_file    = /usr/local/ssl/mail.yourdomain.com.crt
smtpd_tls_session_cache_database = btree:/etc/postfix/tls_smtpd_scache
smtpd_tls_loglevel     = 1
smtpd_tls_received_header = yes
```

Next we have to create all mysql-virtual files, like referenced in the *main.cf*.

Note "hosts = localhost" means Postfix will use sockets, "hosts = 127.0.0.1" means Postfix will use TCP. I would advise to use, sockets are faster than TCP.

My socket is located `/usr/local/mysql/data/mysql.sock` but your mysql socket can be different. So in my personal host file I would use `hosts = unix:/usr/local/mysql/data/mysql.sock` but you can use just `hosts = localhost`.

```
vi /etc/postfix/mysql_virtual_alias_maps.cf
```

```
user = postfix
password = postfix
hosts = localhost
dbname = postfix
query = SELECT goto FROM alias WHERE address='%s' AND active = 1
```

```
vi /etc/postfix/mysql_virtual_domains_maps.cf
```

```
user = postfix
password = postfix
hosts = localhost
dbname = postfix
query = SELECT domain FROM domain WHERE domain='%s'
#optional query to use when relaying for backup MX
#query = SELECT domain FROM domain WHERE domain='%s' and backupmx = '0' and active = '1'
```

```
vi /etc/postfix/mysql_virtual_mailbox_maps.cf
```

```
user = postfix
password = postfix
hosts = localhost
dbname = postfix
query = SELECT maildir FROM mailbox WHERE username='%s' AND active = 1
```

```
vi /etc/postfix/mysql_virtual_mailbox_limit_maps.cf
```

```
user = postfix
password = postfix
hosts = localhost
dbname = postfix
query = SELECT quota FROM mailbox WHERE username='%s'
```

```
vi /etc/postfix/mysql_relay_domains_maps.cf
```

```
user = postfix
password = postfix
hosts = localhost
dbname = postfix
query = SELECT domain FROM domain WHERE domain='%s' and backupmx = '1'
```

These files contain our database username/password, so tighten the security a bit:

```
chown root.postfix /etc/postfix/mysql_*.cf
```

```
chmod 640 /etc/postfix/mysql_*.cf
```

Now we need to populate the mynetworks file. This file lists the IPs that are able to "relay" mail through your server. We put localhost into this file, so that scripts running on this server can relay mail to the internet. For all other users who have mailboxes on your server, when sending mail they can either use SMTP-AUTH, or alternatively they could set their email client's SMTP server settings to point to their ISP's mail server.

```
echo '# Localhost' > /etc/postfix/mynetworks

echo '127.0.0.0/8' >>/etc/postfix/mynetworks

echo '' >>/etc/postfix/mynetworks
```

If you have workstations on a LAN, or other users on the internet with fixed-ip addresses, you can add them here as well, and these users will then be permitted to relay mail.

```
echo '# MyCompany blocks' >>/etc/postfix/mynetworks

echo 'xxx.xxx.xxx.xxx/24' >>/etc/postfix/mynetworks

echo 'yyy.yyy.yyy.yyy/24' >>/etc/postfix/mynetworks
```

Tweak the aliases file. These mappings are used for system related mails eg crontab messages, postfix bounces etc.

```
vi /etc/aliases
```

```
root: someone@yourdomain.com
```

## Directory Structure

As you know there are two common formats for storage of mail messages. The first one is Mbox and the other one is Maildir. Mbox format keeps all mail messages in a single file which has some disadvantages. Maildir format keeps all mails in separate files in special folders.

In our installation we will use Maildir format. Each user will have a mail directory. e.g: you have an e-mail *john@test.com*. Mail directory for this user will be */usr/local/vmail/test.com/john*. By this way, all of your users's mail directories will be created under their domain directory. This is so

great, you keep users of the same domain under the same directory.

Lets create directory for our virtual system.

```
mkdir /opt/mail  
  
chown vmail:vmail /opt/mail  
  
chmod 700 /opt/mail
```

## Courier-authlib

Courier-authlib provides user authentication services to Courier-IMAP, Courier-POP3.

```
vi /etc/authlib/authdaemonrc
```

Change the line starting with "authmodulelist=" as below.

```
authmodulelist="authmysql"  
authdaemonvar=/usr/var/spool/authdaemon
```

```
vi /etc/authlib/authmysqlrc
```

Modify it as described below:

```
MYSQL_SERVER localhost  
MYSQL_USERNAME vmailuser  
MYSQL_PASSWORD vmailpass
```

```
MYSQL_SOCKET /var/lib/mysql/mysql.sock
MYSQL_PORT 0
MYSQL_OPT 0
MYSQL_DATABASE postfix
MYSQL_USER_TABLE mailbox
MYSQL_CRYPT_PWFIELD password
MYSQL_CLEAR_PWFIELD password
#you can optionally enable this next setting if you want
#a particular domain to be appended when users haven't
#specified a domain during authentication
#DEFAULT_DOMAIN yourdomain.com
MYSQL_UID_FIELD '1001'
MYSQL_GID_FIELD '1001'
MYSQL_LOGIN_FIELD username
MYSQL_HOME_FIELD '/opt/mail'
MYSQL_NAME_FIELD name
MYSQL_MAILDIR_FIELD CONCAT("/opt/mail/",maildir)
MYSQL_QUOTA_FIELD concat(quota,'S')
```

Note: MYSQL\_CRYPT\_PWFIELD line can exist or not. Its existence doesn't cause any problem. Since we are going to put clear text passwords into our database.

Tweak the config to disable some unneeded features:

```
vi /etc/authlib/authdaemonrc
```

```
#if your server is going to be very busy, you might need to increase this one
daemons=5
# Disable some unneeded functionality.
# (Note that these could optionally be re-enabled per-user
# by adding appropriate columns to the mailbox database)
```

```
#
# wbnocchangeass : this option allows user to change their password through
# a webclient like a webmail, we are using a custom build
# control panel, so we want to disable this function
# wbussexsender : Include an X-Sender header to all outgoing mail
# ( allows you to track actual sender, even if
# user has altered their From address in a webmail client )
# disablesshared : We don't want shared folders, as this mail server is going
# to be used in ISP rather than corporate scenario
#
DEFAULTOPTIONS="wbnocchangeass=1,wbussexsender=1,disablesshared=1"
```

Secure the authmysqlrc file:

```
chmod 400 /etc/authlib/authmysqlrc
```

## Maildrop

Maildrop provides Postfix with a Maildir++ softquota-compatible way to deliver mail into user's mailboxes.

[Note : Instead of using maildrop, many people use the "Postfix VDA" patch instead. This patch hacks the Postfix virtual delivery agent to \(supposedly\) support Maildir++ softquotas. However I would strongly recommend you don't use that patch! The doco etc for the patch makes it sounds like it does everything you need. However when you actually inspect the code it is a total debacle zone. There are numerous logic errors - the patch fails to follow the Maildir++ specs, and will cause a ridiculous amount of needless load on your server. Maildrop does everything correctly, doesn't require the Postfix source code to be patched \(which is good for Postfix's security/reliability\), and gives additional features like quota warnings. Maildrop also has the huge bonus of being from the same author as Courier-imap/pop3d/sqwebmail so you are guaranteed excellent interoperability between all your tools that touch the Maildir.](#)

We want maildrop to send a warning message when the mailbox of the user is almost full.

```
vi /etc/quotawarnmsg
```

```
X-Comment: Rename/Copy this file to quotawarnmsg, and make appropriate changes
X-Comment: See deliverquota man page for more information
From: Mail Delivery System <Mailer-Daemon@calcom.com.mx>
Reply-To: postmaster@calcom.com.mx
To: Valued Customer;;
Subject: Mail quota warning
Mime-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 7bit
Your mailbox on the server is now more than 90% full. So that you can continue to receive mail you need to remove some messages from your mailbox.
```

Change in the *master.cf* the maildrop options like this:

```
vi /etc/postfix/master.cf
```

```
maildrop unix - n n - - pipe
 flags=DRhu user=vmail argv=/usr/bin/maildrop -w 90 -d ${recipient}
```

The `-w N` option places a warning message into the maildir if the maildir has a quota setting, and after the message was successfully delivered the maildir was at least *N* percent full. The warning message is copied from `/etc/quotawarnmsg` with the addition of the "Date:" and "Message-Id:" headers. The warning is repeated every 24 hours (at least), until the maildir drops below *N* percent full. After a change always reload postfix:

```
/etc/init.d/postfix reload
```

Create `/etc/maildroprc` file and save lines below to this file if you want to enable maildrop logging.

```
vi /etc/maildroprc
```

```
logfile "/var/log/maildroprc.log"
```

## Courier-IMAP / Courier-POP3

We now focus on configuring our IMAP/POP3 daemons.

```
vi /usr/lib/courier-imap/etc/imapd
```

```
# If you are going to run a busy IMAP-based webmail package, you will need to substantially increase this.
# The default value of 4 is insufficient even for servicing individual users, since clients like Thunderbird default to using up to 5 simultaneous connections
#
MAXPERIP=20
# Add our collection of supported auth methods to the advertised capability string
IMAP_CAPABILITY="IMAP4rev1 UIDPLUS CHILDREN NAMESPACE THREAD=ORDEREDSUBJECT THREAD=REFERENCES SORT QUOTA AUTH=CRAM-MD5 AUTH=CRAM-SHA1
AUTH=PLAIN AUTH=LOGIN IDLE"
# we want to turn off the announcement of IMAP ACL extensions,
# as we dont need this ( we arent using shared folders ),
# and the ACL stuff makes Thunderbird spit errors in some cases
IMAP_ACL=0
IMAP_CAPABILITY_TLS="$IMAP_CAPABILITY"
# Enabled the enhanced IDLE functionality
# This allows the IMAP server to notify your client when something has changed
# (eg a new message has arrived)
IMAP_ENHANCEDIDLE=1
# If you were going to have mainly Outlook Express based IMAP users, you can tell
# Courier-IMAP to name the trash folder "Deleted Items"
# However in our case we are expecting most IMAP users to be webmail,
# so sticking with the default "Trash" foldername is probably best.
#IMAP_TRASHFOLDERNAME="Deleted Items"
#IMAP_EMPTYTRASH="Deleted Items":7
```

```
# Enable the Courier-IMAP daemon
IMAPDSTART=YES
```

Please make sure the all the ~START(TM) states are *YES*.

```
vi /usr/lib/courier-imap/etc/imapd-ssl
```

```
# enable courier-imaps (port 993) daemon
IMAPDSSLSTART=YES
# enable STARTTLS extensions for IMAP. Enabling this means "STARTTLS"
# will be added to the IMAP CAPABILITY line
IMAPDSTARTTLS=YES
# nominate where the SSL key/certificate can be found
TLS_CERTFILE=/usr/local/ssl/mail.yourdomain.com.pem
```

```
vi /usr/lib/courier-imap/etc/pop3d
```

```
# you would likely have to increase this for a busy server
MAXDAEMONS=40
# Add out collection of supported auth methods to the advertised capability string
POP3AUTH="CRAM-MD5 CRAM-SHA1 PLAIN LOGIN"
POP3AUTH_TLS="$POP3AUTH"
# enabled the courier-pop3 daemon
POP3DSTART=YES
```

```
vi /usr/lib/courier-imap/etc/pop3d-ssl
```

```
# enable the courier-pop3s (port 995) daemon
POP3DSSLSTART=YES
# enable STARTTLS extensions for POP3.
POP3_STARTTLS=YES
# nominate where the SSL key/certificate can be found
TLS_CERTFILE=/usr/local/ssl/mail.yourdomain.com.pem
```

### Starting daemons the normal manner:

```
/etc/init.d/mysql.server start
/usr/sbin/authdaemond start

/usr/sbin/saslauthd start

/usr/lib/courier-imap/libexec/imapd.rc start

/usr/lib/courier-imap/libexec/pop3d.rc start

postfix start
```

### CentOS manner:

```
/etc/init.d/mysql.server start

/etc/init.d/saslauthd start

/etc/init.d/courier-authlib start
```

```
/etc/init.d/courier-imap start  
  
/etc/init.d/postfix start
```

## Postfix Admin Tweaking

To create the Maildirs on the disk you will need some commands like, we will be doing this automatically by tweaking the Postfix Admin.

Here are some examples to do it manually.

Create the maildir structure:

```
maildirmake /opt/mail/testdomain.com/user1
```

Create the softquota *maildirsize* file in the maildir. (If this file isn't present, no quotas will be enforced.)

```
maildirmake -q 20971520S /opt/mail/testdomain.com/user1
```

*maildirmake -q N*: this option set the quota size (*N* expressed as bytes, in our case 20971520S = 20MB [default quota in db structure])

[Notice the S at the end of the bytes.](#)

```
chmod g-r,o-r /opt/mail/testdomain.com/user1  
chown -R vmail.vmail /opt/mail/testdomain.com/user1
```

## Postfix Admin Tweaking

We need to tweak Postfix Admin so that it can work with Maildrop. Here for we needed 2 extra scripts.

```
vi /usr/sbin/mailedirmake.sh
```

```
#!/bin/bash
set -e
mail_home="/opt/mail"
if [ ! -d $mail_home/$1 ] ; then
    mkdir $mail_home/$1
    chown -R vmail:vmail $mail_home/$1
    chmod -R 700 $mail_home/$1
    #echo "$mail_home/$1 CREATED"
fi
if [ -d $mail_home/$1 ] ; then
    cd "$mail_home/$1"
    mailedirmake $2
    #echo "$mail_home/$1/$2 CREATED"
    mailedirmake -q "$3" $2
    #echo "$3 $2 QUOTA CREATED"
    chown -R vmail:vmail $mail_home/$1/$2
    chmod -R 700 $mail_home/$1/$2
fi
```

This script we will use for automatically creating a mailbox with a certain quota. We also need a script for deleting unwanted mailboxes.

```
vi /usr/sbin/mailedirdel.sh
```

```
# vi /usr/sbin/mailedirdel.sh
#!/bin/bash
set -e
mail_home="/opt/mail"
```

```
if [ -d $mail_home/$1/$2 ] ; then
    rm -Rf mkdir $mail_home/$1/$2
    #echo "$mail_home/$1/$2 DELETED"
fi
```

We want to execute these scripts through apache.

```
vi /etc/sudoers
```

```
apache your_hostname = NOPASSWD: /usr/sbin/mailedmake.sh
```

Next we will tweak Postfix Admin. Change both *create\_mailbox.php* in the main and admin folder.

```
system("sudo /usr/sbin/mailedmake.sh ".$fDomain." ".$fUsername." ". $quota);
db_log ($SESSION_USERNAME, $fDomain, "create mailbox", "$fUsername");
```

Change both *delete.php* in the main and admin folder.

```
db_query ("DELETE FROM vacation WHERE email='$fDelete' AND domain='$fDomain'");
$userarray=explode("@",$fDelete);
$user=$userarray[0];
$domain=$userarray[1];
system("sudo /usr/sbin/maileddel.sh ".$domain."â€•â€œ.$user");
db_log ($SESSION_USERNAME, $fDomain, "delete mailbox", $fDelete);
```

And your Postfix Admin now can make the mailbox directories with quota and remove them.

**Vacation Install** You need to have the following installed to be able to use Virtual Vacation.

- Perl5
- Perl DBI
- Perl DBD::mysql

Else

```
yum install perl perl-DBI perl-DBD-MySQL
```

Virtual Vacation is done with a local shell account that can receive email. The email is then handled by a Perl script which sends the vacation message back to the sender.

Create a dedicated local user account called *vacation*. This user handles all potentially dangerous mail content - that is why it should be a separate account.

```
useradd -d /var/spool/vacation -s /sbin/nologin vacation
```

Do not use *nobody*, and most certainly do not use *root* or *postfix*. The user will never log in, and can be given a \* password and non-existent shell and home directory.

This should look like this:

```
cat /etc/passwd
```

```
vacation:*:65501:65501::0:Virtual Vacation:/nonexistent:/sbin/nologin
```

Create a directory, for example `/var/spool/vacation`, that is accessible only to the `vacation` user. This is where the vacation script is supposed to store its temporary files.

```
mkdir /var/spool/vacation
```

Copy the vacation perl from your VACATION directory in Postfix Admin. Get the latest version (Version 3.2) of `vacation.pl` from the subversion repository. <http://dev.high5.net/trac/wiki>

```
cp vacation.pl /var/spool/vacation/vacation.pl
```

```
chown -R vacation:vacation /var/spool/vacation/  
chmod 700 /var/spool/vacation/*
```

Next we need to setup our transport, at the end of `master.cf` add the following:

```
vi /etc/postfix/master.cf
```

```
#  
# VIRTUAL VACATION  
#  
vacation unix - n n - - pipe  
flags=DRhu user=vacation argv=/var/spool/vacation/vacation.pl -f ${sender} -- ${recipient}
```

Tell Postfix to use a transport maps file, so add the following to your Postfix `main.cf`:

```
vi /etc/postfix/main.cf
```

```
transport_maps = hash:/etc/postfix/transport
```

Then add the transport definition to the newly created transport file. Obviously, change *yourdomain.com* to your own domain. This can be any arbitrary domain, and it is easiest if you just choose one that will be used for all your domains.

```
vi /etc/postfix/transport
```

```
autoreply.yourdomain.com vacation
```

Execute `postmap /etc/postfix/transport` to build the hashed database:

```
postmap /etc/postfix/transport
```

Change in your Postfix Admin `config.inc.php` file:

```
$CONF['vacation'] = 'YES';  
$CONF['vacation_domain'] = 'autoreply.yourdomain.com';
```

Execute `postfix restart` to complete the change:

```
/etc/init.d/postfix restart
```

**Queue Management** You might want to view the mailqueue, if some mails are still pending or so.

```
postqueue -p
```

Time to learn something about a helper application that comes with Postfix: `postsuper`. Set a mail on hold:

```
postsuper -h MESSAGEID
```

Every message has its unique ID provided by Postfix when it accepts a message. If you want to address all mails in the queue use *ALL*.

When you delete a mail it goes like this:

```
postsuper -d MESSAGEID
```

Delete all mails in the queue:

```
postsuper -d ALL
```

## Debugging Postfix logging

Postfix daemon processes run in the background, and log problems and normal activity to the syslog daemon. The syslogd process sorts events by class and severity, and appends them to logfiles. The logging classes, levels and logfile names are usually specified in */etc/syslog.conf*. At the very least you need something like:

```
vi /etc/syslog.conf
```

```
mail.err          /dev/console
mail.debug        /var/log/mail/maillog
```

OR

```
# Log all the mail messages in one place.
mail.*            -/var/log/mail/maillog
```

After changing the *syslog.conf* file, send a *HUP* signal to the syslogd process.

```
ps aux | grep sys
```

```
root      2585  0.0  0.0  3628  632 ?        Ss   Nov27   0:00 syslogd -m 0
```

```
kill -HUP 2585
```

**IMPORTANT:** many syslogd implementations will not create files. You must create files before (re)starting syslogd.

**IMPORTANT:** on Linux you need to put a "-" character before the pathname, e.g., `-/var/log/maillog`, otherwise the syslogd process will use more system resources than Postfix.

Another method is to check the postfix config files.

```
postfix check
```

```
egrep '(reject/warning/error/fatal/panic):' /some/log/file
```

The first line (`postfix check`) causes Postfix to report file permission/ownership discrepancies.

The second line looks for problem reports from the mail software, and reports how effective the relay and junk mail access blocks are. This may produce a lot of output. You will want to apply some postprocessing to eliminate uninteresting information.

The two major log files that you should check are:

- `messages` (or `messages.log`)
- `maillog` - located in `/var/log` and/or `/var/log/mail`

## Debugging authentication problems

Courier-authlib includes a couple of debugging tools. These can be handy if you are having problems eg auth'ing via POP3, but aren't sure if its your POP3 config that's broken or whether its actually the courier-authlib that's not working properly.

Display all accounts:

```
/usr/sbin/authenticate
```

Perform a test authentication, and show all values returned from courier-authlib:

```
/usr/sbin/authtest someuser@yourdomain.com somepassword
```

A common problem after installing the Courier authentication library is that authentication, using `authtest`, doesn't work. Below shows how to use courier's debugging features to pinpoint the problem.

Turn on debugging:

```
vi /etc/authlib/authdaemonrc
```

```
DEBUG_LOGIN=1 # turn on authentication debugging  
DEBUG_LOGIN=2 # turn on authentication debugging AND show passwords
```

This setting is located at the very end of the configuration file.

After changing this setting, restart the authentication daemon by running the `authdaemon stop` and `authdaemon start` commands.

```
/etc/init.d/courier-authlib restart
```

At this point, all debugging output goes to syslog at level `debug`, which is normally not shown. You will probably need to change your `/etc/syslog.conf` file to be able to see these messages. If you have an existing entry which says `mail.info` (which means facility `mail`, level `info` or higher) then you can

just change this to *mail.debug*. Alternatively you can add a new entry like this:

```
vi /etc/syslog.conf
```

```
*.debug          /var/log/debug
```

Don't forget to create this file, and to send a HUP signal to syslogd to make it re-read its configuration:

```
touch /var/log/debug
```

```
kill -HUP syslogd
```

If you don't want to mess around with your syslog configuration, you can also start authdaemond manually, and log its output to a file:

```
/usr/bin/authdaemond >filename 2>&1
```

Issue a manual login like listed below.

## Debug SMTP

If *AUTH* is listed you can log in to the server. This will usually allow some things which are normally restricted, for example relaying. You will need to use your username and password in Base64.

```
telnet localhost 25
```

```
AUTH LOGIN
```

```
334 VXN1cm5hbWU6
```

```
bmFtZQ==
```

```
334 UGFzc3dvcmQ6
```

```
U2VjcmlV0
```

```
235 2.0.0 OK Authenticated
```

## Debug POP3

```
telnet localhost pop3
```

```
user USERNAME  
pass PASSWORD  
stat  
quit
```

You can see if your POP3 server is working correctly. It should give back *+OK Hello there.* (Type *quit* to get back to the Linux shell.)

## Debug IMAP

```
telnet localhost imap
```

```
a login USERNAME PASSWORD  
a examine inbox  
a logout
```

## Debug POP3 over SSL

```
openssl s_client -connect x.x.x.x:995
```

(Then use same commands as in POP3 example.)

## Debug IMAP over SSL

```
openssl s_client -connect x.x.x.x:993
```

(Then use same commands as in IMAP example.)

## Debug Maildrop

```
maildrop -V9 -d someone@yourdomain.com
```

```
maildrop: authlib: groupid=1001
maildrop: authlib: userid=1001
maildrop: authlib: logname=someone@yourdomain.com, home=/var/vmail, mail=yourdomain.com/s/someone/Maildir/
maildrop: Changing to /opt/mail
<press CTRL-D here>
```

## Problems / Errors

After changing values in config files, please make sure that you restart the daemon in charge of these config files to reload your changed config file.

[error: Failed build dependencies: /usr/include/fam.h is needed by courier-imap-4.1.1-1.4.i386](#)

```
sudo yum install gamin-devel
```

[error: Failed dependencies: /usr/local/bin/perl is needed by courier-imap-4.1.1-1.4.i386](#)

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

[error: Failed build dependencies: pcre-devel is needed by maildrop-2.0.2-1.x86\\_64](#)

```
sudo yum install pcre-devel
```

[configure: WARNING: === Do not compile Courier-IMAP as root. Compile](#)

[\\_configure: WARNING: === Courier-IMAP as a non-root user then su to configure: WARNING: === root before running make install. You must now configure: WARNING: === remove this entire directory and then extract the configure: WARNING: === source code from the tarball as a non-root user configure: WARNING: === and rerun the configure script. If you have read configure: WARNING: === the INSTALL file you should have known this. So configure: WARNING: === you better read INSTALL again. configure: error: aborted. error: Bad exit status from /var/tmp/rpm-tmp.46388 \(%prep\)](#)

Do not run as root so try again. First remove old build dir:

```
sudo rm -rf $HOME/rpm/BUILD/courier-imap-4.1.1/

rpmbuild -ta courier-imap-4.1.1.tar.bz2
```

[error: make: \\*\\*\\* No rule to make target ` %{ smp mflags}'. Stop. error: Bad exit status from /var/tmp/rpm-tmp.XXXX \(%build\) This error can occur with the courier-authlib,the courier-imap and maildrop, so please change accordingly.](#)

```
cd $HOME/downloads

cp courier-authlib-0.58.tar.bz2 $HOME/rpm/SOURCES

bunzip2 courier-authlib-0.**

tar xvf courier-authlib-0.**.tar

cd courier-authlib-0.**

vi courier-authlib.spec
```

In the `.spec` file, change:

```
%{__make} %{_smp_mflags}
```

to

```
%{__make}
```

First clean out the old build data before rebuilding:

```
sudo rm -rf $HOME/rpm/BUILD/courier-authlib-0.58/

sudo rm -rf $HOME/rpm/SPECS/courier-authlib.spec

cp courier-authlib.spec $HOME/rpm/SPECS

cd $HOME/rpm/SPECS

sudo rpmbuild -bb courier-authlib.spec
```

courier-imap can(TM)t be build with SUDO but just as plain user.

[error: "File not found by glob:" /var/tmp/courier-authlib-XXX/usr/local/etc/init.d/\\*](#)

```
sudo vi /usr/lib/rpm/macros
```

In `/usr/lib/rpm/macros`, change

```
%_sysconfdir %{_prefix}/etc
```

to

```
%_sysconfdir /etc
```

[/usr/sbin/postconf: /usr/lib64/mysql/libmysqlclient.so.14: no version information available \(required by /usr/sbin/postconf\)](#)

```
vi /etc/ld.so.conf
```

Add as first your path to the mysql lib directory (in my case `/usr/local/mysql/lib/mysql`).

So my `ld.so.conf` looks like this:

```
/usr/local/mysql/lib/mysql  
include ld.so.conf.d/*.conf
```

Make sure your mysql lib location comes before the include.

After this save the file and do a

```
ldconfig
```

[ERR: authdaemon: s\\_connect\(\) failed: Permission denied /usr/bin/maildrop: Temporary authentication failure](#)

Courier Maildrop will get this message when the user running the maildrop program has no access to the socket that authdaemon is running. The socket is usually located at:

```
/usr/var/spool/authdaemon/
```

Check its permissions.

```
ls -al /usr/var/spool/
```

If it's vmail that's trying to run maildrop make sure `/etc/postfix/master.cf` has the user running maildrop as the one that has access to the socket mentioned above.

```
flags=DRu user=vmail argv=/usr/bin/maildrop -d ${recipient}
```

```
chown vmail.daemon /usr/var/spool/authdaemon/
```

### SMTP Problem

Sometimes your ISP provider may block all traffic directed to port 25. So you need to add an extra port in your master.cf file

```
vi /etc/postfix/master.cf
```

```
=====
# service type private unpriv chroot wakeup maxproc command + args
#          (yes) (yes) (yes) (never) (100)
# =====
smtp      inet  n       -       n       -       -       smtpd
567      inet  n       -       n       -       -       smtpd
```

In my case I added the port 567 besides the normal SMTP port (25).

Make sure your firewall has opened this port.

```
/etc/init.d/postfix restart
```

[warning: unknown smtpd restriction: "reject\\_invalid\\_helo\\_hostname"](#)

For Postfix Version < 2.3 change in the *main.cf* file *reject\_invalid\_helo\_hostname* to *reject\_invalid\_hostname*:

```
vi /etc/postfix/main.cf
```

```
smtpd_recipient_restrictions =  
check_client_access proxy:mysql:/etc/postfix/mysql-client-access.cf,  
check_sender_access proxy:mysql:/etc/postfix/mysql-sender-access.cf,  
check_recipient_access proxy:mysql:/etc/postfix/mysql-recipient-access.cf,  
permit_sasl_authenticated,  
permit_mynetworks,  
reject_unauth_destination,  
reject_invalid_hostname,  
reject_non_fqdn_sender,  
reject_non_fqdn_recipient,  
reject_unknown_sender_domain,  
reject_unknown_recipient_domain,  
reject_rbl_client list.dsbl.org,  
reject_rbl_client sbl-xbl.spamhaus.org,  
permit
```

[sql\\_select option missing auxprofunc error no mechanism available Make sure you are working with the correct smtpd.conf file. If working on a X86\\_64 machine. Please use the /usr/lib64/sasl2/smtpd.conf file.](#)

[Module IO::Multiplex is required for Multiplex](#)

```
yum install --enablerepo=dag perl-Net-Server perl-IO-Multiplex
```

## Mailscanner

MailScanner is an open-source E-mail program to secure against spam and viruses. I prefer it above amavisd, because it has clamav and spamassin support built in and it(TM)s easier to configure and to maintain.

Before beginning with the installation of Mailscanner, make sure your Postfix version is all working. Stop Postfix using the command:

```
/etc/init.d/postfix stop
```

Make sure you have the chroot jail set up in `/var/spool/postfix`. You should be able to see `etc`, `usr` and `lib` directories inside `/var/spool/postfix`. If you haven't got the chroot jail setup already, then look in the `examples` directory of the Postfix documentation and you will find a script in there to set up it up for your operating system. If you can't find that, then see the "Problems or Errors" section further down this page.

```
sh postfix-chroot.sh enable
```

[warning: SASL authentication failure: cannot connect to Courier authdaemond: No such file or directory Move saslauthd's socket dir inside Postfix's chroot and create a link to keep everybody happy:](#)

```
mkdir /var/spool/postfix/usr/var  
  
mkdir /var/spool/postfix/usr/var/spool  
  
mv /usr/var/spool/authdaemon/ /var/spool/postfix/usr/var/spool/authdaemon  
  
ln -s /var/spool/postfix/usr/var/spool/authdaemon/ /usr/var/spool/authdaemon
```

Restart Postfix and start saslauthd:

```
/etc/init.d/postfix restart
```

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

At this point, things change from the setup for other MTAs as we can make it run with just one copy of Postfix, and let Postfix do the "split MTA" setup for us. In the Postfix configuration file `/etc/postfix/main.cf` add this line:

```
vi /etc/postfix/main.cf
```

```
header_checks = regexp:/etc/postfix/header_checks
```

In the file `/etc/postfix/header_checks` add this line:

```
vi /etc/postfix/header_check
```

```
/^Received:/ HOLD
```

The effect of this is to tell Postfix to move all messages to the HOLD queue.

In your `MailScanner.conf` file (probably in `/etc/MailScanner` or `/opt/MailScanner/etc`), there are five settings you need to change. They are all really near the top of the file. The settings are:

```
vi /etc/MailScanner/MailScanner.conf
```

```
Run As User = postfix  
Run As Group = postfix  
Incoming Queue Dir = /var/spool/postfix/hold  
Outgoing Queue Dir = /var/spool/postfix/incoming  
MTA = postfix
```

You will need to ensure that the user *postfix* can write to */var/spool/MailScanner/incoming* and */var/spool/MailScanner/quarantine*:

```
chown postfix.postfix /var/spool/MailScanner/incoming
chown postfix.postfix /var/spool/MailScanner/quarantine
```

If you upgrade your copy of MailScanner, unfortunately these directories will be changed back to being owned by root, so you will have to do those two commands again.

```
/etc/rc.d/init.d/MailScanner start
```

That(TM)s it for mailscanner, you can config to your personal needs by edditing the */etc/Mailscanner/MailScanner.conf*.

## Perl ClamAV and Spamassassin Module

The easiest way to install ClamAV and Spamassassin is to download the latest Perl ClamAV - Spamassassin install script for mail scanner from <http://www.mailscanner.info/downloads.html> (<http://www.mailscanner.info/files/4/install-Clam-0.88.7-SA-3.1.7.tar.gz>):

```
wget http://www.mailscanner.info/files/4/install-Clam-0.88.7-SA-3.1.7.tar.gz

tar -xvzf install-Clam-0.88.7-SA-3.1.7.tar.gz

cd install-Clam-0.88.7-SA-3.1.7

sh install.sh

/etc/rc.d/init.d/MailScanner start
```

And you are all set and good to go for fighting spam and virus mails.

## Razor Agent

Download latest razor-agents-sdk and razor-agents.

```
wget http://surfnet.dl.sourceforge.net/sourceforge/razor/razor-agents-sdk-2.07.tar.bz2
```

```
wget http://surfnet.dl.sourceforge.net/sourceforge/razor/razor-agents-2.82.tar.bz2
```

Unzip and untar the packages:

```
bunzip2 razor-agents-sdk-2.07.tar.bz2
```

```
bunzip2 razor-agents-2.82.tar.bz2
```

```
tar -xf razor-agents-sdk-2.07.tar
```

```
tar -xf razor-agents-2.82.tar
```

Build the razor sdk:

```
cd razor-agents-sdk-2.07.tar
```

```
perl Makefile.PL
```

```
make
```

```
make test
```

```
make install
```

## Build the razor agent:

```
cd razor-agents-2.82

perl Makefile.PL

make

make test

make install
```

## Rules du jour

Download latest Rules du jour package and unpack:

```
wget http://www.fsl.com/support/Rules_Du_Jour.tar.gz

tar -xzf Rules_Du_Jour.tar.gz
```

```
cd rules_du_jour/

sh install.sh
```