

OpenBSD as a Mail Server

Author: [Daniele Mazzocchio](#)

Applies to: OpenBSD 4.6

Last update: Dec 5, 2009

Latest version: <http://www.kernel-panic.it/openbsd/mail/>

Table of Contents

| | |
|---|----|
| 1. Introduction..... | 2 |
| 2. Preliminary installation steps..... | 4 |
| 3. Postfix..... | 5 |
| 3.1 Configuration..... | 5 |
| 4. MySQL..... | 10 |
| 4.1 The socket dilemma..... | 10 |
| 4.2 Configuration..... | 11 |
| 5. Courier-IMAP..... | 14 |
| 5.1 Installation and configuration..... | 14 |
| 5.2 Adding POP3 access..... | 16 |
| 5.3 Managing disk space..... | 16 |
| 6. Content filtering..... | 18 |
| 6.1 SpamAssassin..... | 18 |
| 6.2 ClamAV..... | 19 |
| 6.3 Amavisd-new..... | 21 |
| 7. Advanced Postfix configuration..... | 24 |
| 7.1 Enabling TLS..... | 24 |
| 7.2 SMTP authentication with SASL..... | 25 |
| 8. SquirrelMail..... | 28 |
| 8.1 Preliminary steps..... | 28 |
| 8.2 Installation and configuration..... | 28 |
| 9. Appendix..... | 30 |
| 9.1 References..... | 30 |
| 9.2 Bibliography..... | 30 |

OpenBSD

the “*secure by default*” operating system, with “*only two remote holes in the default install, in a heck of a long time!*”;

Postfix

an MTA “*that started life at IBM research as an alternative to the widely-used [Sendmail](#) program*” and which “*attempts to be fast, easy to administer, and secure*”;

MySQL

the “*world's most popular open source database*”;

Courier-IMAP

a “*fast, scalable, enterprise IMAP server*” that supports MySQL and [maildirs](#);

Cyrus SASL

the [Cyrus](#) implementation of the [SASL](#) protocol;

Amavisd-new

a “*high-performance interface between mailer (MTA) and content checkers*” (antivirus and antispam), written in Perl and optimized for Postfix;

SpamAssassin

a Perl-based “*mail filter to identify Spam*”, using “*a variety of mechanisms including header and text analysis, Bayesian filtering, DNS blocklists, and collaborative filtering databases*”;

ClamAV

a fast and easy-to-use open-source virus scanner.

A good knowledge of OpenBSD is assumed, since we won't delve into system management topics such as [base configuration](#) or [packages/ports](#) installation.

2. Preliminary installation steps

Before delving into the installation and configuration of all the mail-handling software, we will take a brief look at the operating system that will host it.

As usual, my choice goes to [OpenBSD](#) for its proven security, reliability and ease of use. Needless to say, all these features are essential for a system that will have to handle a large volume of email traffic while still making life hard for spammers and malicious users.

We won't dwell upon the installation procedure here, which is documented in full detail on the [OpenBSD web site](#). Just a couple of notes:

- while partitioning the hard drive, bear in mind that we will [configure Postfix](#) to use [virtual domains](#) and, consequently, it will store all users' mail folders in a single directory (`/var/vmail`). Therefore, it is recommended to assign a (large) dedicated slice to this filesystem, in order to prevent mails from filling up any critical filesystem, should quotas fail. Furthermore, if you choose to install [MySQL](#) on the mail server itself, it is usually recommended to assign one of the first slices to `/var/mysql`, in order to allow for faster disk access by the database engine;
- the only file sets we will need to install are those marked as “**required**” on the [documentation](#), i.e. `bsd` (the kernel), `baseXX.tgz` (the base system), and `etcXX.tgz` (the configuration files in `/etc`) plus `compXX.tgz` (the C compiler), since we will also have to install some [ports](#) not available as precompiled packages for licensing reasons. *Note:* since leaving a compiler on a publicly accessible server is a definite security risk, it is recommended that you remove the compiler when the installation is over or that you compile on another machine.

After the first reboot, we can disable some default network services managed by [inetd\(8\)](#):

```
$ grep -v ^# /etc/inetd.conf
ident      stream    tcp      nowait    _identd  /usr/libexec/identd    identd -el
ident      stream    tcp6    nowait    _identd  /usr/libexec/identd    identd -el
127.0.0.1:comsat dgram    udp      wait      root     /usr/libexec/comsat    comsat
[::1]:comsat  dgram    udp6    wait      root     /usr/libexec/comsat    comsat
daytime     stream    tcp      nowait    root     internal
daytime     stream    tcp6    nowait    root     internal
time        stream    tcp      nowait    root     internal
time        stream    tcp6    nowait    root     internal
$
```

by commenting them out in [/etc/inetd.conf](#) and reloading [inetd\(8\)](#):

```
# pkill -HUP inetd
```

Anyway, OpenBSD is considered secure also with those services turned on and the mail server should be placed behind a firewall; nevertheless, I prefer staying on the safe side and disable them all (including [comsat\(8\)](#), since we won't have any interactive user receiving mail on the system).

To modify the server network configuration, please refer to the [related chapter](#) in the previous document about [redundant firewalls](#) or to the [networking](#) FAQ.

3. Postfix

[Postfix](#) is a MTA (Mail Transport Agent) developed by [Wietse Venema](#) “as an alternative to the widely-used [Sendmail](#) program”. It “attempts to be fast, easy to administer, and secure, while at the same time being [sendmail](#) compatible enough to not upset existing users. Thus, the outside has a [sendmail](#)-ish flavor, but the inside is completely different”. Postfix also comes with excellent [documentation](#) and a lot of [howtos](#).

Our mail server requirements will be quite simple: it will be final destination solely for its [canonical domains](#) and it will only relay mail from systems on the internal network (though we will also consider relaying from untrusted networks by means of [SMTP authentication](#)). Canonical domains include the hostname (in our case, "mail.kernel-panic.it") and the IP address (172.16.240.150) of the machine that Postfix runs on, and the parent domain of the hostname ("kernel-panic.it").

Canonical domains are usually implemented with the Postfix [local domain address class](#), which, unfortunately, has one major drawback for me: it requires that each e-mail account have a corresponding Unix account. On the contrary, I prefer:

1. keeping Unix and e-mail accounts apart and
2. having all mailboxes well-ordered inside a single directory.

Therefore, we will use Postfix [Virtual Domain Hosting](#), which is normally used for hosting multiple internet domains on the same server, but will also allow us to achieve our two goals.

3.1 Configuration

In this paragraph, we will configure Postfix to work standalone, with no back-end database. Then, in the [next chapter](#), when everything will be working fine, we will hook up Postfix to a MySQL database; this will allow us to centrally store configuration information that both Postfix and [Courier-IMAP](#) will need to access.

There are a few packages we need to install:

- `mysql-client-x.x.x.tgz`
- `pcre-x.x.tgz`
- `postfix-x.x.x-mysql.tgz`

Note: if you're planning to use [SMTP authentication](#), you will need to compile Postfix from the ports, because there's no pre-compiled package available with both MySQL and SASL support:

```
# cd /usr/ports/mail/postfix/snapshot
# env FLAVOR="mysql sasl2" make install
```

The installation will create the `/etc/postfix` directory, containing all the configuration files. Postfix has several hundred configuration parameters that are controlled via the `/etc/postfix/main.cf` file, but don't worry: for the vast majority of these parameters, the default value is the best option (see [postconf\(5\)](#) for a detailed list of all the available configuration parameters, their description and their default value) and we will only have to override a very small subset of them:

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

```
# Directory containing all the post* commands
command_directory = /usr/local/sbin
# Directory containing all the Postfix daemon programs
daemon_directory = /usr/local/libexec/postfix

# Full pathnames of various Postfix commands
sendmail_path = /usr/local/sbin/sendmail
```

```

newaliases_path = /usr/local/sbin/newaliases
mailq_path = /usr/local/sbin/mailq

# Directories containing documentation
html_directory = /usr/local/share/doc/postfix/html
manpage_directory = /usr/local/man
readme_directory = /usr/local/share/doc/postfix/readme

# The owner of the Postfix queue and of most Postfix daemon processes
mail_owner = _postfix
# The group for mail submission and queue management commands
setgid_group = _postdrop

# The myhostname parameter specifies the internet hostname of this mail system. It
# is used as default for many other configuration parameters (default = system's
# FQDN)
myhostname = mail.kernel-panic.it

# The internet domain name of this mail system. Used as default for many other
# configuration parameters (default = $myhostname minus the first component)
mydomain = kernel-panic.it

# The domain name that locally-posted mail appears to come from, and that locally
# posted mail is delivered to. As you can see, a parameter value may refer to other
# parameters
myorigin = $myhostname

# Network interface addresses that this mail system receives mail on
inet_interfaces = all

# Network interface addresses that this mail system receives mail on by way of a
# proxy or NAT unit
proxy_interfaces = router.kernel-panic.it

# List of domains that this machine considers itself the final destination for.
# Virtual domains must not be specified here
mydestination = $myhostname, localhost.$mydomain, localhost

# List of "trusted" SMTP clients allowed to relay mail through Postfix.
mynetworks = 127.0.0.0/8, 172.16.0.0/24, 172.16.240.0/24

# What destination (sub)domains this system will relay mail to
relay_domains = $mydestination

# The default host to send mail to when no entry is matched in the optional
# transport(5) table. Square brackets turn off MX lookups
relayhost = [smtp.isp.com]

# List of alias databases used by the local delivery agent
alias_maps = hash:/etc/postfix/aliases

# Alias database(s) built with "newaliases" or "sendmail -bi". This is a separate
# configuration parameter, because alias_maps may specify tables that are not
# necessarily all under control by Postfix
alias_database = hash:/etc/postfix/aliases

# SMTP greeting banner
smtpd_banner = $myhostname ESMTP $mail_name

# Postfix is final destination for the specified list of "virtual" domains
virtual_mailbox_domains = kernel-panic.it

# Virtual mailboxes base directory

```

```

virtual_mailbox_base = /var/vmail

# Optional lookup tables with all valid addresses in the domains that match
# $virtual_mailbox_domains.
virtual_mailbox_maps = hash:/etc/postfix/vmailbox

# The minimum user ID value accepted by the virtual(8) delivery agent
virtual_minimum_uid = 2000

# User ID that the virtual(8) delivery agent uses while writing to the recipient's
# mailbox
virtual_uid_maps = static:2000

# Group ID that the virtual(8) delivery agent uses while writing to the recipient's
# mailbox
virtual_gid_maps = static:2000

# Optional lookup tables that alias specific mail addresses or domains to other
# local or remote address
virtual_alias_maps = hash:/etc/postfix/virtual

```

Let's take a closer look at some of the above configuration parameters.

One of the goals we had was to avoid having a separate Unix account for each e-mail account. We have achieved this by configuring Postfix to write to the mailboxes using uid 2000 and gid 2000 (see the `virtual_uid_maps` and `virtual_gid_maps` parameters above). Now we only have to create a user with this pair of uid and gid:

```

# useradd -d /var/vmail -g =uid -u 2000 -s /sbin/nologin \
> -c "Virtual Mailboxes Owner" -m vmail

```

Our second goal was having all mailboxes grouped together in a single directory; this is achieved by setting the value of the `virtual_mailbox_base` parameter to the path of that directory (in our configuration, `/var/vmail`). In matter of fact, this parameter is a prefix that the [virtual\(8\)](#) agent prepends to all pathname results from `virtual_mailbox_maps` table lookups.

In our configuration, the `virtual_mailbox_maps` parameter refers to the `/etc/postfix/vmailbox` file, containing the list of all valid addresses in the virtual domains (`virtual_mailbox_domains` parameter) and the path to the corresponding mailboxes or maildirs (a mailbox is a single file containing all the emails; a [maildir](#), instead, is a directory, with a defined structure, containing all the emails in separate files):

```
/etc/postfix/vmailbox
```

```

info@kernel-panic.it      kernel-panic.it/info/
d.mazzocchio@kernel-panic.it  kernel-panic.it/d.mazzocchio/
[...]

```

Please pay attention to the trailing slashes: they tell Postfix that the pathname refers to a maildir instead of a mailbox file, and maildirs are our only option, since [Courier-IMAP](#) doesn't support mailbox files.

The `virtual_alias_maps` parameter allows to alias specific mail addresses or domains to other local or remote address. Its value is the pathname to a file (in our case `/etc/postfix/virtual`) containing the alias mappings:

```
/etc/postfix/virtual
```

```

root@kernel-panic.it      root@localhost.kernel-panic.it
postmaster@kernel-panic.it  postmaster@localhost.kernel-panic.it
abuse@kernel-panic.it     postmaster@localhost.kernel-panic.it

```

```
[...]
```

Finally, the `/etc/postfix/aliases` file contains the addresses to which Postfix will redirect mail for local recipients (see [aliases\(5\)](#)). Since many accounts point to root's email address, you should check root email frequently or forward it all to another account. E.g.:

```
/etc/postfix/aliases
```

```
root: d.mazzocchio@kernel-panic.it
MAILER-DAEMON: postmaster
postmaster: root
bin: root
[...]
```

Now we only have to reload Postfix lookup tables:

```
# /usr/local/sbin/postmap /etc/postfix/vmailbox
# /usr/local/sbin/postmap /etc/postfix/virtual
# /usr/local/sbin/newaliases
```

replace Sendmail:

```
# /usr/local/sbin/postfix-enable
old /etc/mailer.conf saved as /etc/mailer.conf.pre-postfix
postfix /etc/mailer.conf enabled

NOTE: do not forget to add sendmail_flags="-bd" to
      /etc/rc.conf.local to startup postfix correctly.

NOTE: do not forget to add "-a /var/spool/postfix/dev/log" to
      syslogd_flags in /etc/rc.conf.local and restart syslogd.

NOTE: do not forget to remove the "sendmail clientmqueue runner"
      from root's crontab.

#
```

and follow the above advice, by commenting out the "sendmail clientmqueue runner" in root's crontab:

```
# sendmail clientmqueue runner
#*/30 * * * * /usr/sbin/sendmail -L sm-msp-queue -Ac -q
```

and adding a couple of variables in the [/etc/rc.conf.local\(8\)](#) file.

```
/etc/rc.conf.local
```

```
# Specify a location where syslogd(8) should place an additional log socket
# for Postfix
syslogd_flags="-a /var/spool/postfix/dev/log"

# Make Postfix start in background and process queued messages every 30 min
sendmail_flags="-bd"
```

Now we can change a few permissions and restart the processes (or simply reboot):

```
# chgrp _postdrop /usr/local/sbin/postqueue /usr/local/sbin/postdrop
# chmod 2755 /usr/local/sbin/postqueue /usr/local/sbin/postdrop
# pkill syslogd
# syslogd -a /var/empty/dev/log -a /var/spool/postfix/dev/log
# pkill sendmail
# /usr/local/sbin/sendmail -bd
postfix/postfix-script: starting the Postfix mail system
```

and test our hard work!

```
# telnet mail.kernel-panic.it 25
Trying 172.16.240.150...
Connected to mail.kernel-panic.it.
Escape character is '^]'.
220 mail.kernel-panic.it ESMTP Postfix
HELO somedomain.org
250 mail.kernel-panic.it
mail from: someone@somedomain.org
250 Ok
rcpt to: d.mazzocchio@kernel-panic.it
250 Ok
data
354 End data with <CR><LF>.<CR><LF>
From: someone@somedomain.org
To: d.mazzocchio@kernel-panic.it
Subject: Test mail

It works!
.
250 Ok: queued as 548D7286
quit
221 Bye
Connection closed by foreign host.
# tail /var/log/maillog
Dec 16 10 15:26:35 mail postfix/smtpd[29212]: connect from ws1.lan.kernel-panic.it[172.16.0.15]
Dec 16 15:26:53 mail postfix/smtpd[29212]: 57076222: client=ws1.lan.kernel-panic.it[172.16.0.15]
Dec 16 15:27:02 mail postfix/cleanup[13428]: 57076222: message-id=<20070210142653.57076222@mail.kernel-panic.it>
Dec 16 15:27:02 mail postfix/qmgr[26776]: 57076222: from=<someone@somedomain.org>, size=392, nrcpt=1 (queue active)
Dec 16 15:27:02 mail postfix/virtual[14381]: 57076222: to=<d.mazzocchio@kernel-panic.it>, relay=virtual, delay=15, delays=15/0.28/0/0.03, dsn=2.0.0, status=sent (delivered to maildir)
Dec 16 15:27:02 mail postfix/qmgr[26776]: 57076222: removed
Dec 16 15:27:06 mail postfix/smtpd[29212]: disconnect from ws1.lan.kernel-panic.it[172.16.0.15]
# cat /var/vmail/kernel-panic.it/d.mazzocchio/new/1118146014.V3I9448M811660.mail.kernel-panic.it
Return-Path: <someone@somedomain.org>
X-Original-To: d.mazzocchio@kernel-panic.it
Delivered-To: d.mazzocchio@kernel-panic.it
Received: from somedomain.org (ws1.lan.kernel-panic.it [172.16.0.15])
    by mail.kernel-panic.it (Postfix) with SMTP id 57076222
    for <d.mazzocchio@kernel-panic.it> Sat, 16 Dec 2007 15:26:47 +0100 (CET)
From: someone@somedomain.org
To: d.mazzocchio@kernel-panic.it
Subject: Test mail
Message-Id: <20070210142653.57076222@mail.kernel-panic.it>
Date: Sat, 16 Dec 2007 15:26:47 +0100 (CET)

It works!
#
```

4. MySQL

If Postfix is working fine, we can proceed to the next step and install [MySQL](#). MySQL is “*the world's most popular open source database*”, combining performance, reliability and ease of use. It will ensure faster data access times and allow us to centralize configuration information that both [Postfix](#) and [Courier-IMAP](#) will need to access.

There are a few packages we need to install:

- p5-Net-Daemon-x.xx.tgz
- p5-PIRPC-x.xxxx.tgz
- p5-DBI-x.xx.tgz
- p5-DBD-mysql-x.xxxx.tgz
- mysql-server-x.x.xx.tgz

After the installation, you will find various sample configuration files in the `/usr/local/share/mysql` directory; choose the most suitable to your needs and copy it to `/etc/my.cnf`. E.g.:

```
# cp /usr/local/share/mysql/my-small.cnf /etc/my.cnf
```

4.1 The socket dilemma

Choosing a good location for the MySQL socket file is sometimes hard because of chrooted processes, which need to access it from inside their "reduced" filesystem. But Postfix goes even further: of its many processes, most are chrooted to the `/var/spool/postfix` directory, but a few are not! As a consequence, by default, part of the Postfix processes will look for the socket file in the `/var/run/mysql/` directory, while the others will look for it in the `/var/spool/postfix/var/run/mysql/` directory!

Anyway, there are many possible workarounds:

1. if the database runs on a remote server, there is no need to bother with the socket file! We will later see how to configure [Postfix](#) and [Courier-IMAP](#) for connecting to a remote database;
2. if you want to preserve the defaults as much as possible, you can create a symbolic link to the socket inside the chroot before the database startup:

```
# mkdir -p /var/spool/postfix/var/run/mysql/
# ln -f /var/run/mysql/mysql.sock /var/spool/postfix/var/run/mysql/mysql.sock
```

3. Remember to add the above commands to the `/etc/rc.local(8)` script to automatically create the link at boot time.
4. you can place the socket inside the Postfix chroot (by setting the value of the `socket` variable in the `[mysqld]` section of `/etc/my.cnf` to the path of the socket, e.g. `/var/spool/postfix/mysql/mysql.sock`), and give Postfix the possibility to choose between two distinct paths: `/var/spool/postfix/mysql/mysql.sock`, for non-chrooted processes, and `/mysql/mysql.sock`, for chrooted processes;
5. finally, you can forget about socket files and connect through the loopback network interface.

Mmmh... what to choose? After a moment's thought, I chose the latter solution, which is probably the simplest. Therefore, I left "skip networking" commented out in `/etc/my.cnf` and added the following line in the `[mysqld]` section :

```
/etc/my.cnf
```

```
bind-address = 127.0.0.1
```

thus preventing MySQL from listening on the external network interfaces.

4.2 Configuration

First and foremost, we need to install the default databases, change the password of the MySQL root user (don't take my passwords as an example!):

```
# /usr/local/bin/mysql_install_db
[ ... ]
# mysqld_safe &
[ ... ]
# /usr/local/bin/mysql_secure_installation
[ ... ]
Enter current password for root (enter for none): <Enter>
OK, successfully used password, moving on...
[ ... ]
Set root password? [Y/n] Y
New password: root
Re-enter new password: root
Password updated successfully!
[ ... ]
Remove anonymous users? [Y/n] Y
... Success!
[ ... ]
Disallow root login remotely? [Y/n] Y
... Success!
[ ... ]
Remove test database and access to it? [Y/n] Y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!
[ ... ]
Reload privilege tables now? [Y/n] Y
... Success!
[ ... ]
#
```

and configure the system to start MySQL on boot:

```
/etc/rc.local
```

```
if [ -x /usr/local/bin/mysqld_safe ]; then
    echo -n ' MySQL'
    /usr/local/bin/mysqld_safe >/dev/null 2>&1 &
fi
```

Next, we will hook Postfix up to the database. In particular, we will modify the value of a few parameters in the `/etc/postfix/main.cf` file:

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

```
virtual_mailbox_domains = mysql:/etc/postfix/mysql_virtual_domains.cf
virtual_mailbox_maps = mysql:/etc/postfix/mysql_virtual_mailboxes.cf
virtual_alias_maps = mysql:/etc/postfix/mysql_virtual_alias_maps.cf
```

We will see [in a moment](#) the contents of those files; but first, we are going to create the database. Tables

don't need to have any particular structure, since we will tell Postfix which queries to use to extract the data. Therefore, this will actually be just one among the many possible implementations: feel free to modify it according to your taste and needs.

Note: [Postfix](#) obtains the full pathname of the maildirs by joining the values of the `virtual_mailbox_base` and `virtual_mailbox_maps` parameters, while [Courier-IMAP](#) obtains it by joining the values of the `MYSQL_HOME_FIELD` and `MYSQL_MAILDIR_FIELD` parameters. As a consequence, we will create two separate fields in the `users` table (`home` and `maildir`) and make those variables point to them in order for Postfix and Courier-IMAP to get along.

```
# mysql -u root -p
password: root
mysql> CREATE DATABASE mail;
Query OK, 1 row affected (0.01 sec)

mysql> use mail
Database changed
mysql> CREATE TABLE domains (
->     id          INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
->     domain     VARCHAR(255) NOT NULL UNIQUE);
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE users (
->     id          INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
->     login      VARCHAR(255) NOT NULL UNIQUE,
->     name       VARCHAR(255) NOT NULL,
->     password   CHAR(13) NOT NULL,
->     uid        SMALLINT NOT NULL DEFAULT 2000,
->     gid        SMALLINT NOT NULL DEFAULT 2000,
->     home       VARCHAR(255) NOT NULL DEFAULT '/var/vmail',
->     maildir    VARCHAR(255) NOT NULL,
->     quota      VARCHAR(10) NOT NULL DEFAULT '10000000S');
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE alias_maps (
->     id          INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
->     account    VARCHAR(255) NOT NULL UNIQUE,
->     alias      VARCHAR(255) NOT NULL);
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT SELECT ON mail.* to 'vmail'@'localhost' IDENTIFIED BY 'vmail';
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO domains (domain) VALUES ('kernel-panic.it');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO users (login, name, password, maildir)
-> VALUES ('d.mazzocchio@kernel-panic.it', 'Daniele Mazzocchio',
-> ENCRYPT('danix'), 'kernel-panic.it/d.mazzocchio/');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO alias_maps (account, alias)
-> VALUES ('postmaster@kernel-panic.it',
-> 'postmaster@localhost.kernel-panic.it');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO alias_maps (account, alias)
-> VALUES ('root@kernel-panic.it', 'root@localhost.kernel-panic.it');
Query OK, 1 row affected (0.00 sec)
```

Now let's take a brief look at the new Postfix configuration files, which include the configuration settings for MySQL.

```
/etc/postfix/mysql_virtual_domains.cf
```

```
user = vmail
password = vmail

# solution 1:
# hosts = db_server_name
# Solution 2: skip this parameter
# Solution 3 (this file is required only by chrooted processes):
# hosts = unix:/mysql/mysql.sock
# Solution 4:
hosts = 127.0.0.1

dbname = mail
query = SELECT domain FROM domains WHERE domain='%s'
```

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

```
user = vmail
password = vmail

# solution 1:
# hosts = db_server_name
# Solution 2: skip this parameter
# Solution 3 (this file is required only by chrooted processes):
# hosts = unix:/mysql/mysql.sock
# Solution 4:
hosts = 127.0.0.1

dbname = mail
query = SELECT alias FROM alias_maps WHERE account='%s'
```

```
/etc/postfix/mysql_virtual_mailboxes.cf
```

```
user = vmail
password = vmail

# solution 1:
# hosts = db_server_name
# Solution 2: skip this parameter
# Solution 3 (this file is required by both chrooted and non-chrooted processes):
# hosts = unix:/mysql/mysql.sock unix:/var/spool/postfix/mysql/mysql.sock
# Solution 4:
hosts = 127.0.0.1

dbname = mail
query = SELECT maildir FROM users WHERE login='%s'
```

That's all: now we can reload Postfix configuration:

```
# postfix reload
postfix/postfix-script: refreshing the Postfix mail system
```

and test our work; everything should run exactly as before!

5. Courier-IMAP

Now that our server can send and receive email, it may be useful to let users read it! For this purpose, we're going to install [Courier-IMAP](#), “a fast, scalable, enterprise IMAP server that uses [Maildirs](#)”. This is the same IMAP server that comes with the [Courier mail server](#), but configured as a standalone IMAP server that can be used with other mail servers, such as Postfix.

5.1 Installation and configuration

The following is the list of the required packages:

- `gdbm-x.x.x.tgz`
- `libltdl-x.x.x.tgz`
- `tcl-x.x.x.tgz`
- `expect-x.x.x-no_tk.tgz`
- `courier-authlib-x.x.tgz`
- `courier-imap-x.x.x.tgz`
- `courier-authlib-mysql-x.x.x.tgz`

Once you have added all the packages, you will find a fresh new `/etc/courier/` directory containing Courier IMAP's configuration files. Let's take a brief look at each of them.

The `/etc/courier/authdaemonrc` configuration file sets several operational parameters for the `authdaemond` process (the resident authentication daemon); fortunately, we only need to edit the `authmodulelist` parameter, which specifies the list of the authentication modules available; set it to `authmysql` to allow for [MySQL](#) based authentication:

```
/etc/courier/authdaemonrc
```

```
[ ... ]
authmodulelist="authmysql"
[ ... ]
```

The `/etc/courier/authmysqlrc` configuration file contains the `authmysql` database connection parameters; below is a sample configuration file:

```
/etc/courier/authmysqlrc
```

```
MYSQL_SERVER          127.0.0.1
MYSQL_USERNAME         vmail
MYSQL_PASSWORD         vmail
# If you connect through the socket:
#MYSQL_SOCKET          /path/to/mysql.sock
#MYSQL_PORT            0
MYSQL_PORT            3306
MYSQL_OPT              0
MYSQL_DATABASE         mail
MYSQL_USER_TABLE       users
MYSQL_CRYPT_PWFIELD   password
MYSQL_DEFAULT_DOMAIN  kernel-panic.it
MYSQL_UID_FIELD        uid
MYSQL_GID_FIELD        gid
MYSQL_LOGIN_FIELD      login
MYSQL_HOME_FIELD      home
MYSQL_NAME_FIELD       name
MYSQL_MAILDIR_FIELD    maildir
MYSQL_QUOTA_FIELD      quota
# MYSQL_WHERE_CLAUSE  field=value AND field=value...
```

The next step is creating the SSL certificate for the IMAPS protocol. To make your life easier, Courier-IMAP comes with a script, [mkimapdcert\(8\)](#), which will create the certificate after reading all the necessary information from the `/etc/courier/imapd.cnf` configuration file. Therefore, you should first customize the latter file (in particular, pay close attention to the common name (CN) parameter, which must match the name of the server the users will connect to) and then run [mkimapdcert\(8\)](#):

```
# /usr/local/sbin/mkimapdcert
[ ... ]
```

Now we only have to start the daemons:

```
# mkdir -p /var/run/courier{,-auth}/
# /usr/local/sbin/authdaemon start
# /usr/local/libexec/imapd.rc start
# /usr/local/libexec/imapd-ssl.rc start
```

configure the system to start Courier-IMAP on boot:

```
/etc/rc.local
```

```
echo -n ' Courier-IMAP'
/bin/mkdir -p /var/run/courier{,-auth}/
[ -x /usr/local/sbin/authdaemon ] && /usr/local/sbin/authdaemon start
[ -x /usr/local/libexec/imapd.rc ] && /usr/local/libexec/imapd.rc start
[ -x /usr/local/libexec/imapd-ssl.rc ] && /usr/local/libexec/imapd-ssl.rc start
```

...and test our hard work! I suggest using a simple Python script, just to give our weary fingers a break:

```
IMAP_test.py
```

```
#!/usr/bin/env python

import imaplib

# Constants
IMAP_SRV = "mail.kernel-panic.it"
USER     = "d.mazzocchio@kernel-panic.it"
PASSWD   = "danix"

# Connect to server
imap_srv = imaplib.IMAP4(IMAP_SRV)
imap_srv.login(USER, PASSWD)

# Select the INBOX folder
imap_srv.select()

# Retrieve message list
msg_nums = imap_srv.search(None, 'ALL')[1]

# Print all messages
for num in msg_nums[0].split():
    msg = imap_srv.fetch(num, '(RFC822)')[1]
    print 'Message %s\n%s\n' % (num, msg[0][1])

# Disconnect from server
imap_srv.close()
imap_srv.logout()
```

5.2 Adding POP3 access

It is usually desirable that email users be offered the choice between IMAP and POP3 remote access; after all, POP3 users tend to use less disk space, bandwidth and resources on the server.

Adding POP3 support to our mail server is fairly simple; first, we need to add the appropriate package:

- `courier-pop3-x.x.x.tgz`

Then, we have to run [mkpop3dcert\(8\)](#) to generate the SSL certificate for POP3 over SSL (similarly to [mkimapdcert\(8\)](#), SSL parameters are read from a configuration file, `/etc/courier/pop3d.cnf`) and start the daemons:

```
# /usr/local/sbin/mkpop3dcert
[ ... ]
# /usr/local/libexec/pop3d.rc start
# /usr/local/libexec/pop3d-ssl.rc start
```

Add the following lines to [/etc/rc.local\(8\)](#) to start the POP3 server on boot:

```
/etc/rc.local
```

```
[ -x /usr/local/libexec/pop3d.rc ] && /usr/local/libexec/pop3d.rc start
[ -x /usr/local/libexec/pop3d-ssl.rc ] && /usr/local/libexec/pop3d-ssl.rc start
```

Finally, we can perform a quick test to make sure everything works as expected:

```
# telnet mail.kernel-panic.it 110
Trying 172.16.240.150...
Connected to mail.kernel-panic.it.
Escape character is '^]'.
+OK Hello there.
user d.mazzocchio@kernel-panic.it
+OK Password required.
pass danix
+OK logged in.
list
1 2531
[ ... ]
quit
+OK Bye-bye.
Connection closed by foreign host.
#
```

5.3 Managing disk space

Quotas allow you to specify the maximum size of maildirs, in order to prevent the `/var/vmail` filesystem from filling up. The best option would certainly be to use the operating system's built-in [quota support](#), but we can't, because we have a single user writing to all the maildirs. Therefore, we must rely on the mail software to get quota support on maildirs.

Courier-IMAP comes with built-in quota support, but this solves only one half of the problem: in fact, also Postfix must be able to reject mail sent to over-quota users. To achieve this, we will rely on the [deliverquota\(8\)](#) utility, which delivers mail taking into account any software-imposed quota on maildirs.

The first step is assigning a quota to each maildir with [maildirmake\(1\)](#). E.g.:

```
# /usr/local/bin/maildirmake -q 10000000S \
> /var/vmail/kernel-panic.it/d.mazzocchio
```

The above command installs an (approximately) 10MB quota on the `/var/vmail/kernel-panic.it/d.mazzocchio` maildir. *Note:* [maildirmake\(1\)](#) also allows you to create and initialize maildirs, thus allowing users to access them; otherwise, a user's maildir will be created upon receiving his first email.

Next, we need to define, in `/etc/postfix/master.cf(5)`, a special Postfix daemon for delivery through [deliverquota\(8\)](#):

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

```
[ ... ]
qdeliver unix - n n - - pipe
 flags=uh user=vmail argv=/usr/local/bin/deliverquota -c -w 90
 /var/vmail/${domain}/${user}
```

and tell Postfix to use this daemon for final delivery to virtual domains, by setting the value of the `virtual_transport` parameter in `/etc/postfix/main.cf`:

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

```
virtual_transport = qdeliver
```

[deliverquota\(8\)](#) will place a warning message into the maildir if, after the message is successfully delivered, the maildir is at least 90 percent full (`-w 90`). The body of the warning message is copied verbatim from the `/etc/courier/quotawarnmsg` file.

Please note that, as reported by Giovanni Bechis, [deliverquota\(8\)](#) fails to correctly deliver emails sent to an alias that maps to multiple accounts, one of which has the same name as the alias itself, unless you set the following parameters in `/etc/postfix/main.cf`:

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

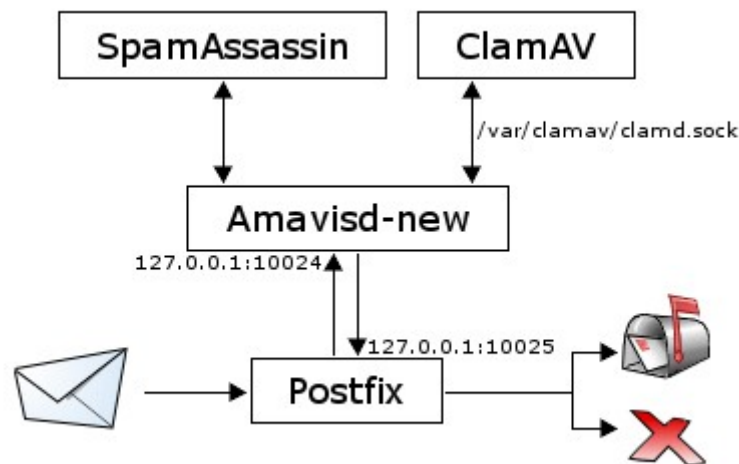
```
qdeliver_destination_concurrency_limit = 1
qdeliver_destination_recipient_limit = 1
```

Setting these parameters to "1" disables parallel deliveries to the same recipient.

6. Content filtering

Now we have a fully-functional mail server, able to send and receive email and providing remote access to users' mailboxes. However, if we don't want our server to become an immune carrier of computer viruses or to be drowned under a sea of spam, we need to install all the necessary content-filtering tools.

Though Postfix natively supports multiple [content inspection mechanisms](#), the [documentation](#) itself “*encourages the use of external filters and standard protocols because this allows you to choose the best MTA and the best content inspection software for your purpose*”. Therefore, we will rely on third-party software for content filtering; in particular, we will use [SpamAssassin](#) to filter spam, [ClamAV](#) to check emails for viruses and [Amavisd-new](#) to coordinate it all. Below is the outline of the whole architecture:



6.1 SpamAssassin

[SpamAssassin](#) is a “*mature, widely-deployed open source project that serves as a mail filter to identify Spam. SpamAssassin uses a variety of mechanisms including header and text analysis, Bayesian filtering, DNS blocklists, and collaborative filtering databases*”.

There are quite a few packages we need to install:

- p5-Compress-Raw-Zlib-x.x.tgz
- p5-IO-Compress-Base-x.x.tgz
- p5-IO-Compress-Zlib-x.x.tgz
- p5-Compress-Zlib-x.x.tgz
- p5-IO-Zlib-x.x.tgz
- p5-IO-String-x.x.tgz
- p5-Algorithm-Diff-x.x.tgz
- p5-Text-Diff-x.x.tgz
- p5-Archive-Tar-x.x.tgz
- re2c-x.x.tgz
- p5-Net-CIDR-Lite-x.x.tgz
- p5-Net-IP-x.x.tgz
- p5-Digest-SHA1-x.x.tgz
- p5-Digest-HMAC-x.x.tgz
- p5-Net-DNS-x.x.tgz
- p5-Sys-Hostname-Long-x.x.tgz
- p5-URI-x.x.tgz

- p5-Mail-SPF-Query-*x.x*.tgz
- p5-Socket6-*x.x*.tgz
- p5-IO-INET6-*x.x*.tgz
- bzip2-*x.x.x*.tgz
- libiconv-*x.x.x*.tgz
- gettext-*x.x.x*.tgz
- libidn-*x.x.x*.tgz
- curl-*x.x.x*.tgz
- gnupg-*x.x.x*.tgz
- p5-Net-SSLeay-*x.x*.tgz
- p5-IO-Socket-SSL-*x.x*.tgz
- p5-HTML-Tagset-*x.x*.tgz
- p5-HTML-Parser-*x.x*.tgz
- p5-Crypt-SSLeay-*x.x*.tgz
- libhttp-*x.x.x*.tgz
- p5-HTTP-GHTTP-*x.x*.tgz
- p5-libwww-*x.x*.tgz
- p5-Mail-SpamAssassin-*x.x.x*.tgz

After the packages installation, you will find the main SpamAssassin configuration file (`local.cf`) in the fresh new `/etc/mail/spamassassin` directory. The configuration phase can be very complex and goes beyond the scope of this document; anyway, you can find all the details in the man page ([Mail::SpamAssassin::Conf](mailto:SpamAssassin::Conf)).

Like Postfix, SpamAssassin has a lot of configuration parameters, although, in most cases, default values can be preserved and only a few parameters need to be overridden:

```
/etc/mail/spamassassin/local.cf
```

```
rewrite_header Subject ***** SPAM *****
report_safe 1
lock_method flock
required_score 8.0
```

6.2 ClamAV

[ClamAV](#) is an “*open source (GPL) anti-virus toolkit for UNIX*”; the main purpose of this software is the integration with mail servers (i.e. attachment scanning). All the antivirus tasks are handled by three processes:

freshclam

which automatically updates the virus definitions, by connecting to one of the ClamAV [mirrors](#); its configuration file is `/etc/freshclam.conf`;

clamd

a flexible and scalable multi-threaded antivirus daemon; its configuration file is `/etc/clamd.conf`;

clamscan

a command line antivirus scanner.

The required packages are:

- arc-*x.xx*.tgz
- lha-*x.xx.xxxxxx*.tgz
- unzip-*x.x*.tgz
- zoo-*x.x.x*.tgz
- gmp-*x.x.x*.tgz

- unarj-x.x (from the ports)
- unrar-x.x (from the ports)
- clamav-x.x.tgz

The `freshclam.conf` configuration file requires only a few parameters:

```
/etc/freshclam.conf
```

```
DatabaseDirectory    /var/db/clamav
DatabaseOwner        _clamav
DNSDatabaseInfo      current.cvd.clamav.net
DatabaseMirror       db.it.clamav.net
DatabaseMirror       database.clamav.net
MaxAttempts          3
checks               24
```

Now we can update the virus definition database by running the `freshclam` command. Please make sure you have installed the latest release of ClamAV, or you'll get warning messages about reduced functionality, like the following:

```
# freshclam
ClamAV update process started at Tue Dec 18 00:35:25 2007
WARNING: Your ClamAV installation is OUTDATED!
WARNING: Local version: 0.90.3 Recommended version: 0.92
DON'T PANIC! Read http://www.clamav.net/support/faq
Downloading main.cvd [100%]
main.cvd updated (version: 45, sigs: 169676, f-level: 21, builder: sven)
WARNING: Your ClamAV installation is OUTDATED!
WARNING: Current functionality level = 16, recommended = 21
DON'T PANIC! Read http://www.clamav.net/support/faq
Downloading daily.cvd [100%]
daily.cvd updated (version: 5160, sigs: 8698, f-level: 21, builder: sven)
WARNING: Your ClamAV installation is OUTDATED!
WARNING: Current functionality level = 16, recommended = 21
DON'T PANIC! Read http://www.clamav.net/support/faq
Database updated (178374 signatures) from db.it.clamav.net (IP: 193.206.139.37)
#
```

The reduced "functionality level" means that you may not be able to use all the available virus signatures and, consequently, fail to detect the latest viruses. To automatically update the database, we simply have to schedule `freshclam` in `crontab` every hour (preferably not on the hour, just to avoid traffic peaks):

```
16 * * * * /usr/local/bin/freshclam >/dev/null 2>&1
```

Also the `/etc/clamd.conf` configuration file needs editing only very few parameters:

```
/etc/clamd.conf
```

```
DatabaseDirectory    /var/db/clamav
LocalSocket          /var/clamav/clamd.sock
User                 _clamav
[...]
```

Now we can run `clamd`:

```
# touch /var/log/clamd.log
# chown _clamav /var/log/clamd.log
# clamd
Running as user _clamav (UID 539, GID 539)
```

and add the following lines to [/etc/rc.local \(8\)](#) to start it on system boot:

```
/etc/rc.local
```

```
if [ -x /usr/local/sbin/clamd ]; then
    echo -n ' clamd'
    [ -S /var/clamav/clamd.sock ] && rm -f /var/clamav/clamd.sock
    /usr/local/sbin/clamd >/dev/null 2>&1
fi
```

6.3 Amavisd-new

[Amavisd-new](#) is a high-performance interface between mailer (MTA) and content checkers. We will configure it to bind to port 10024 on the loopback interface, where Postfix will forward all incoming e-mails. If the e-mail successfully passes all the checks, it will be forwarded back to Postfix, listening on localhost port 10025; otherwise, mails may be deleted or quarantined and the administrator and recipients may be notified.

The following is the list of the required packages:

- cabextract-x.x.tgz
- freeze-x.x (from the ports)
- p5-Convert-BinHex-x.x.tgz
- p5-IO-stringy-x.x.tgz
- p5-Mail-Tools-x.x.tgz
- p5-Time-TimeDate-x.x.tgz
- p5-MIME-tools-x.x.tgz
- p5-Convert-TNEF-x.x.tgz
- p5-Convert-UUlib-x.x.tgz
- rpm2cpio-x.x.tgz
- p5-Net-Server-x.x.tgz
- p5-Unix-Syslog-x.x.tgz
- amavisd-new-x.x.x.tgz

The installation procedure creates a new user and group called `_vscan`; however, the easiest way to get Amavisd-new to cooperate with [ClamAV](#), is to run them both under the same user (`_clamav`). The configuration file is `/etc/amavisd.conf`, which is actually a perl script (so pay attention to the semi-colons at the end of the lines!); below are the options you will most likely want to tweak:

```
/etc/amavisd.conf
```

```
# COMMONLY ADJUSTED SETTINGS:

$max_servers = 2;
$daemon_user = '_clamav';      # Run under the same user as ClamAV
$daemon_group = '_clamav';    # Run under the same group as ClamAV

$mydomain = 'kernel-panic.it';

$MYHOME = '/var/amavisd';
$TEMPBASE = "$MYHOME/tmp";    # working directory, needs to be created manually
$ENV{TMPDIR} = $TEMPBASE;
$QUARANTINEDIR = '/var/clamav/quarantine';

[...]

# Leave only ClamAV uncommented
@av_scanners = (
    ['ClamAV-clamd',
     \&ask_daemon, ["CONTSCAN {}\n", "/var/clamav/clamd.sock"],
     qr/\bOK$/, qr/\bFOUND$/,
```

```

qr/^. *?: (?!Infected Archive)(.*) FOUND$/ ],
);
[...]

# Leave only ClamAV uncommented
@av_scanners_backup = (
  ['ClamAV-clamscan', 'clamscan',
   "--stdout --disable-summary -r --tempdir=$TEMPBASE {}"], [0], [1],
  qr/^. *?: (?!Infected Archive)(.*) FOUND$/ ],
);
1;

```

After manually creating Amavisd-new's working directory (`/var/amavisd/tmp`), we can start the daemon in debug mode (i.e. in foreground), just to check any errors:

```

# mkdir /var/amavisd/tmp
# chown -R clamav:clamav /var/amavisd/
# /usr/local/sbin/amavisd debug
Dec 18 22:07:11 mail.kernel-panic.it /usr/local/sbin/amavisd[24429]: starting.
/usr/local/sbin/amavisd at mail.kernel-panic.it amavisd-new-2.3.2 (20050629),
Unicode aware
Dec 18 22:07:11 mail.kernel-panic.it /usr/local/sbin/amavisd[24429]: user=,
EUID: 0 (0); group=, EGID: 0 31 20 5 4 3 2 0 (0 31 20 5 4 3 2 0)
Dec 18 22:07:11 mail.kernel-panic.it /usr/local/sbin/amavisd[24429]: Perl version
5.008008
[...]

```

Now we can configure the system to start Amavisd-new on boot:

```
/etc/rc.local
```

```

if [ -x /usr/local/sbin/amavisd ]; then
  echo -n ' amavisd'
  /usr/local/sbin/amavisd >/dev/null 2>&1
fi

```

The last step is to update Postfix [configuration](#) to enable interfacing between Postfix and Amavisd-new. To achieve this, we have to add a couple of services to the [/etc/postfix/master.cf\(5\)](#) configuration file: one to forward all incoming emails to Amavisd-new, and the other to get emails back again:

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

```

smtp-amavis unix - - - - 2 smtp
  -o smtp_data_done_timeout=1200
  -o smtp_send_xforward_command=yes
  -o disable_dns_lookups=yes
  -o max_use=20

127.0.0.1:10025 inet n - - - - smtpd
  -o content_filter=
  -o local_recipient_maps=
  -o relay_recipient_maps=
  -o smtpd_restriction_classes=
  -o smtpd_delay_reject=no
  -o smtpd_client_restrictions=permit_mynetworks,reject
  -o smtpd_helo_restrictions=
  -o smtpd_sender_restrictions=
  -o smtpd_recipient_restrictions=permit_mynetworks,reject
  -o mynetworks_style=host

```

```
-o mynetworks=127.0.0.0/8
-o strict_rfc821_envelopes=yes
-o smtpd_error_sleep_time=0
-o smtpd_soft_error_limit=1001
-o smtpd_hard_error_limit=1000
-o smtpd_client_connection_count_limit=0
-o smtpd_client_connection_rate_limit=0
-o receive_override_options=no_header_body_checks,no_unknown_recipient_checks
```

Finally, we need to tell Postfix to start forwarding all the emails it receives to amavisd-new for content inspection and reload the configuration.

```
# postfix -e 'content_filter=smtp-amavis:[127.0.0.1]:10024'
# postfix reload
postfix/postfix-script: refreshing the Postfix mail system
```

7. Advanced Postfix configuration

Postfix offers a lot of interesting features whose discussion would go beyond the scope of this document; we will examine just a few, mostly security-related, that may be necessary depending on the environment where Postfix is operating.

7.1 Enabling TLS

Enabling TLS support in Postfix allows you to encrypt SMTP sessions and [SASL authentication](#) exchanges and to optionally authenticate remote SMTP clients and/or servers. However, keep in mind that, by enabling TLS support, you also turn on thousands and thousands of lines of OpenSSL library code. Assuming that OpenSSL is written as carefully as Wietse's own code, every 1000 lines introduce one additional bug into Postfix [[TLS](#)]. Therefore, if you think you don't need these features, feel free to skip this paragraph.

TLS relies on public key certificates for authentication and therefore requires that you first set up a basic Public Key Infrastructure (PKI) for managing digital certificates. As a preliminary step, we will create the directories where certificates will be stored:

```
# install -m 700 -d /etc/postfix/ssl/private
```

The first step in setting up the PKI is the creation of the root CA certificate (`/etc/ssl/ca.crt`) and private key (`/etc/ssl/private/ca.key`) using [openssl\(1\)](#):

```
# openssl req -days 3650 -nodes -new -x509 -keyout /etc/ssl/private/ca.key \
> -out /etc/ssl/ca.crt
[ ... ]
Country Name (2 letter code) []: IT
State or Province Name (full name) []: Italy
Locality Name (eg, city) []: Milan

Organization Name (eg, company) []: Kernel Panic Inc.
Organizational Unit Name (eg, section) []: Postfix CA
Common Name (eg, fully qualified host name) []: ca.lan.kernel-panic.it
Email Address []: <enter>
#
```

The next step is the creation of the private key (`/etc/postfix/ssl/private/server.key`) and Certificate Signing Request (`/etc/postfix/ssl/private/server.csr`) for the mail server:

```
# openssl req -days 3650 -nodes -new -keyout /etc/postfix/ssl/private/server.key \
> -out /etc/postfix/ssl/private/server.csr
[ ... ]
Country Name (2 letter code) []: IT
State or Province Name (full name) []: Italy
Locality Name (eg, city) []: Milan

Organization Name (eg, company) []: Kernel Panic Inc.
Organizational Unit Name (eg, section) []: Postfix Server
Common Name (eg, fully qualified host name) []: mail.kernel-panic.it
Email Address []: <enter>

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: <enter>
An optional company name []: <enter>
#
```

Finally, the CA will generate the signed certificate out of the certificate request:

```
# openssl x509 -req -days 3650 -in /etc/postfix/ssl/private/server.csr \
> -out /etc/postfix/ssl/server.crt -CA /etc/ssl/ca.crt \
> -CAkey /etc/ssl/private/ca.key -CAcreateserial

Signature ok
subject=/C=IT/ST=Italy/L=Milan/O=Kernel Panic Inc./OU=Postfix Server/CN=mail.kernel-panic.it
Getting CA Private Key
#
```

If you want the mail server to authenticate SMTP clients, you can generate any number of client certificates by repeating the last two steps.

To actually turn on TLS support in Postfix, we need to add a few parameters to the `/etc/postfix/main.cf` configuration file:

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

```
# Enable (optional) TLS encryption
smtpd_tls_security_level = may
# Enable logging of TLS handshake and certificate information
smtpd_tls_loglevel = 1

# TLS certificates
smtpd_tls_cert_file = /etc/postfix/ssl/server.crt
smtpd_tls_key_file = /etc/postfix/ssl/private/server.key
smtpd_tls_CAfile = /etc/ssl/ca.crt

# External entropy source for the pseudo random number generator pool.
# Specify /dev/arandom when /dev/urandom gives timeout errors.
tls_random_source = dev:/dev/urandom
```

and uncomment the `smtps` service in [/etc/postfix/master.cf\(5\)](#):

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

```
smtps      inet  n       -       n       -       -       smtpd
  -o smtpd_tls_wrappermode=yes
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_client_restrictions=permit_sasl_authenticated,reject
  -o milter_macro_daemon_name=ORIGINATING
```

Finally, we can reload Postfix configuration to apply the changes made:

```
# postfix reload
postfix/postfix-script: refreshing the Postfix mail system
```

7.2 SMTP authentication with SASL

When [configuring Postfix](#), we have restricted mail relay to a limited number of trusted networks, i.e. the internal corporate LANs. Sometimes, however, such a relay policy may not fit your organization's requirements: a typical example is the need to let mobile users (such as sales people) send messages from anywhere over the Internet.

In these cases, the best solution to allow relay from legitimate users (while still blocking UCE software) is certainly using SMTP authentication, by means of the SASL protocol (defined in [\[RFC4954\]](#)). However, there are some downsides to enabling SASL authentication: it will force us to unchroot the [smtpd\(8\)](#) process and, since the Cyrus SASL library is a lot of code, Postfix will become as secure as

other mail systems that use the Cyrus SASL library (see [\[SASL\]](#)). So, if you don't need this feature, feel free to skip this paragraph.

To enable SASL authentication in Postfix, we just have to add a few parameters in the `/etc/postfix/main.cf` configuration file:

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

```
# Enable SASL authentication in the Postfix SMTP server
smtpd_sasl_auth_enable = yes

# Only accept mail from trusted networks, authenticated clients or mail with
# a 'RCPT TO' address that Postfix is forwarder or final destination for
smtpd_recipient_restrictions = permit_mynetworks permit_sasl_authenticated
    reject_unauth_destination

# Enable inter-operability with old SMTP clients
broken_sasl_auth_clients = yes

# Name of the Postfix SMTP server's local SASL authentication realm
smtpd_sasl_local_domain = $mydomain
```

SASL configuration parameters must be contained in a file named `smtpd.conf`, located in `/usr/local/lib/sasl2`. The SASL library will rely on Courier's `authdaemond` process as the authentication backend (the MySQL backend only supports passwords stored in clear-text):

```
/usr/local/lib/sasl2/smtpd.conf
```

```
pwcheck_method: authdaemond
authdaemond_path: /var/run/courier-auth/socket
mech_list: PLAIN LOGIN
```

Next, we need to edit the `/etc/postfix/master.cf(5)` file to make `smtpd(8)` run unchrooted (just put a "n" in the process's `chroot` field):

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

```
smtp      inet  n       -       n       -       -       smtpd
[ ... ]
```

and reload Postfix configuration.

```
# postfix reload
postfix/postfix-script: refreshing the Postfix mail system
```

Finally, we can perform a simple test to make sure everything works as expected. Authentication information is sent as the base64-encoding of the string `"\0username\0password"`, where `"\0"` is the null byte.

```
$ perl -MMIME::Base64 \
> -e 'print encode_base64("\0d.mazzocchio\@kernel-panic.it\0danix");'
AGQubWF6em9jY2hpb0BrZXJuzWwtcGFuaWMuaXQAZGFuaXg=
$ telnet mail.kernel-panic.it 25
Trying 172.16.240.150...
Connected to mail.kernel-panic.it.
Escape character is '^]'.
220 mail.kernel-panic.it ESMTP Postfix
EHLO somedomain.org
250-mail.kernel-panic.it
250-PIPELINING
250-SIZE 10240000
```

```
250-VERFY
250-ETRN
250-AUTH PLAIN LOGIN
250-AUTH=PLAIN LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
AUTH PLAIN AGQubWF6em9jY2hpb0BrZXJuzWwtcGFuaWMuaXQAZGFuaXg=
235 2.0.0 Authentication successful
quit
221 2.0.0 Bye
Connection closed by foreign host.
$
```

8. SquirrelMail

In addition to [IMAP](#) and [POP3](#), most end-users will be glad to have also web-based access to their mailbox. Therefore, we will install [SquirrelMail](#), a “standards-based webmail package written in PHP”, with built-in support for the IMAP and SMTP protocols. It is written for maximum compatibility across browsers and has very few requirements; SquirrelMail is very easy to [install](#) and [configure](#) and provides a lot of [plugins](#) to allow administrators to deeply customize its look and feel.

As an excellent alternative, you may want to have a look at [RoundCube](#), also available through OpenBSD [ports and packages](#).

Please note that the webmail software doesn't have to necessarily reside on the mail host: indeed, if you have a dedicated web server available, that's certainly the best place to install it.

8.1 Preliminary steps

The required packages are:

- `libxml-x.x.x.tgz`
- `php5-core.x.x.x.tgz`
- `php5-mbstring-x.x.x.tgz`

If [PHP](#) wasn't already installed on your system, don't forget to enable it, as well as the `mbstring` module:

```
# ln -s /var/www/conf/modules.sample/php5.conf /var/www/conf/modules
# ln -fs /var/www/conf/php5.sample/mbstring.ini /var/www/conf/php5/mbstring.ini
```

You also have to uncomment the following line in the Apache configuration file, `/var/www/conf/httpd.conf`:

```
/var/www/conf/httpd.conf
```

```
AddType application/x-httpd-php .php
```

and restart Apache:

```
# apachectl restart
/usr/sbin/apachectl restart: httpd restarted
```

8.2 Installation and configuration

Now we can [download](#) SquirrelMail, extract it inside Apache's chroot and give the newly-created directory a nicer name:

```
# tar -zxvf squirrelmail-x.x.x.tar.gz -C /var/www/htdocs
# mv /var/www/htdocs/squirrelmail-x.x.x /var/www/htdocs/webmail
```

Next, we need to create three directories:

- the data directory (`/var/www/squirrelmail/data`), which will contain user preferences and will have to belong to Apache's `www` user;
- the attachment directory (`/var/www/squirrelmail/attachments`), where email attachments will be uploaded and which will have to be readable only by the root user, but writable by the `www` user too;
- the temporary directory where session data will be stored (`/var/www/tmp`), which must be

accessible only to the www user.

```
# install -d -o www -g www /var/www/squirrelmail/data
# install -d -g www -m 730 /var/www/squirrelmail/attachments
# install -d -o www -g www -m 700 /var/www/tmp
```

Further configuration can be easily managed through a menu-driven Perl script, `/var/www/htdocs/webmail/config/conf.pl`. Some of the parameters you will certainly have to customize are:

Organization Preferences -> Organization Name

The organization name, which will appear here and there in the web pages.

Server Settings -> Domain

The domain name.

General Options -> Data Directory

The pathname to the directory in which user preferences will be stored (in our case `/squirrelmail/data/`).

General Options -> Attachment Directory

The pathname to the directory in which attachments will be temporarily stored (in our case `/squirrelmail/attachments/`).

Set pre-defined settings for specific IMAP servers -> courier

Enable the predefined settings optimized for Courier-IMAP.

9. Appendix

Special thanks to TomazZ for his detailed notes on configuring TLS in Postfix.

9.1 References

- [Redundant firewalls with OpenBSD, CARP and pfsync](#)
- [OpenBSD](#), a FREE, multi-platform 4.BSD-based UNIX-like operating system
- [Postfix](#), an Open source email server for Unix
- [\[TLS\]](#) - Postfix TLS Support
- [MySQL](#), the world's most popular open source database
- [Courier-IMAP](#), a fast, scalable, enterprise IMAP server that uses Maildirs
- [Cyrus SASL](#), the Cyrus SASL API implementation
- [Amavisd-new](#), a high-performance interface between mailer (MTA) and content checkers: virus scanners and/or SpamAssassin
- [SpamAssassin](#), a mail filter to identify spam using a wide range of heuristic tests on mail headers and body text
- [ClamAV](#), an open source (GPL) anti-virus toolkit for Unix
- [Postfix Virtual Domain Hosting Howto](#)
- [\[RFC4954\]](#) - RFC 4954, SMTP Service Extension for Authentication
- [\[SASL\]](#) - Postfix SASL Howto

9.2 Bibliography

- [OpenBSD Documentation and Frequently Asked Questions](#)
- [Postfix Configuration - UCE Controls](#)
- [Postfix Basic Configuration](#)
- [Postfix MySQL Howto](#)
- [Postfix Virtual Domain Hosting Howto](#)
- [Mail::SpamAssassin::Conf](#)
- [How to use amavisd-new with Postfix](#)
- [Clam AntiVirus 0.92 User Manual](#)
- [Fairly-Secure Anti-SPAM Gateway Using OpenBSD, Postfix, Amavisd-new, SpamAssassin, Razor and DCC](#)