
Published on Free Software Magazine (http://www.freesoftwaremagazine.com)

# Secure email servers from scratch with FreeBSD 6 (Part 2)

## Configuring the core components

By Yousef Ourabi

In the last article we parted ways after configuring a base FreeBSD system, enabling it with upgrades via `cvsup` and `portsupgrade`, and securing it with a simple `ipfw2` firewall. The previous article created a solid foundation which this article will build on, covering the configuration of Postfix, amavisd-new, ClamAV, SpamAssassin, MySQL and finally SquirrelMail for web mail. The final setup will have all the bells and whistles of a high end-mail setup: web-mail, anti-virus filtering, spam filtering, and hosting unlimited domains with virtual domains and users stored in MySQL.

**Postfix is released under the IBM Public License, and not the GNU Public License**

# Postfix

The first and most important component is Postfix, a well known mail transfer agent developed by Wietse Venema at IBM and initially known as the "IBM Secure Mailer". Venema, is a respected software engineer who also developed the popular security tool "S.A.T.A.N" (Security Administrator Tool for Analyzing Networks). Postfix is released under the IBM Public License, and not the GNU Public License; the "IPL" has been approved as an open source license by both the Free Software Foundation ("FSF") and the Open Source Initiative("OSI"); however, it has been declared incompatible with the GPL. Initially Postfix was created in reaction to a long list of security vulnerabilities in Sendmail, the then dominant "MTA". The direct result of the "security first" mind, Postfix has a well earned reputation for being easy to setup, fast and secure.

Postfix has two central configuration files:

- `main.cf`: which configures the "properties" of the mail server such as where user and domain information is stored, or which domains to accept mail for.
- `master.cf`: which configures the "behavior" of the Postfix daemon, such as configuring interfaces to non Postfix programs, and other configuration settings for the Postfix daemon itself.

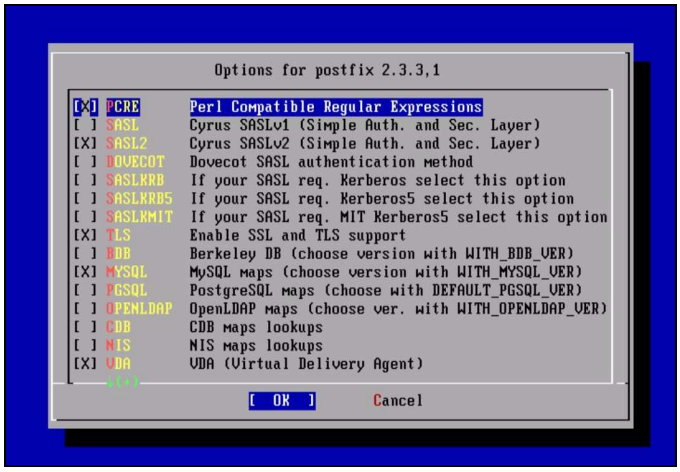
FreeBSD places the configuration files of packages installed from ports under `/usr/local/etc`; so normally you'll find the Postfix configuration files under `/usr/local/etc/postfix`. First of all, you should install Postfix from the ports tree, in the same way that MySQL was installed in the first article.

# Installing Postfix using ports

```
cd /usr/ports/mail/postfix

make install && make clean
```

You will then be presented with a dialog box: select *TLS* and *MYSQL*.




Postfix configuration dialog

Note that when MySQL functionality is selected, the default action is to install the MySQL 4.1 client library. If you plan on running a newer version of MySQL, such as 5.0, simply cancel the Postfix installation, install the MySQL client library of your choice, and then re-run the installation.

Example:

```
cd /usr/ports/databases/mysql50-client
make install && make clean
```

During the installation of the client libraries you may be prompted for options to the `gettext` package. It's not necessary to select any of the options, but feel free to do so if you wish.

After Postfix is built, you will be prompted asking if you want to activate it in the `mailer.conf` file: say "yes":

```
[Prompt] Would you like to activate Postfix in /etc/mail/mailer.conf [n]? y
```

# Configuring Postfix—`main.cf`

Now that Postfix is installed, it's time to dive into the most important of the two configuration files, `main.cf`. In this one file there are essentially two sets of directives: one for the domains the server will be hosting, which being with the `virtual` prefix; and one for the mail server itself, with lines that begin with `my` as in `myhostname`. I'll be giving in line commentary, so read the configuration file closely.

```
# These virtual_* directives configure the domains, users,
# and aliases this Postfix instance will handle.
# Use proxy: for performance
virtual_alias_maps = proxy:mysql:/usr/local/etc/postfix/mysql_virtual_alias_maps.cf
virtual_mailbox_domains = proxy:mysql:/usr/local/etc/postfix/mysql_virtual_domains_maps.cf
virtual_mailbox_maps = proxy:mysql:/usr/local/etc/postfix/mysql_virtual_mailboxes_maps.cf

proxy_read_maps = $local_recipient_maps $mydestination $virtual_alias_maps
  $virtual_alias_domains $virtual_mailbox_maps $virtual_mailbox_domains
```

```
  $relay_recipient_maps $relay_domains $canonical_maps $sender_canonical_maps
  $recipient_canonical_maps $relocated_maps $transport_maps $mynetworks
  $virtual_mailbox_limit_maps

# Where to store the mail
virtual_mailbox_base = /usr/local/virtual

# Ownership of the mail directory
virtual_uid_maps = static:125
virtual_gid_maps = static:125


smtpd_sasl_auth_enable = yes
smtpd_sasl_local_domain = $myhostname
smtpd_sasl_security_options = noanonymous

# Secure SMTP-AUTH
smtpd_use_tls = yes
smtpd_tls_enforce_tls = yes

# Uncomment the following line if you only want auth to happen over tsl
# smtpd_tls_auth_only = yes

# This setups the ssl certificates which I'll configure a little later
smtpd_tls_cert_file = /usr/local/etc/postfix/smtpd.crt
smtpd_tls_key_file = /usr/local/etc/postfix/smtpd.key

# Mostly for MS outlook clients
broken_sasl_auth_clients = yes

# Built in restrictions
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


# Enables virtual hosting
virtual_transport = virtual

# Filter with amavis-new which uses clam-av for
content_filter=smtp-amavis:[127.0.0.1]:10024
```

Now, look at the configuration for the actual host itself:

```
# Configure your hostname, and domain names here!
myhostname = myhost.mydomain.mytld
mydomain = mydomain.mytld
myorigin = $mydomain

# Which network interfaces to listen on
inet_interfaces = all
```

```
# This allows delivery for mail to root@[host] for system messages
mydestination = myhost, myhost.mydomain.mytld
unknown_local_recipient_reject_code = 550

# For other systems on network?
mynetworks_style = host

# This is what people will see if they telnet to port 25 on your mail server
# it might be worth placing a warning message, for legal reasons. Systems with
# banners outlining acceptable use, and the legal actions that will be taken
# have a slightly stronger case in court.

smtpd_banner = mysqlever.mydomain.mytld ESMTP (Insert witty message here)
```

The framework for the rest of the article is in place. Even though the system is un-usable in the sense the no users can login to send or receive mail, Postfix is itself configured enough to send and receive mail. The next step I'm going go take will be putting the finishing touches on the `master.cf` file. The default file may look something like that listed below, but you'll only need to edit a tiny bit of it, and as such I'll only show the relevent lines.

# Configuring `master.cf`

The file `master.cf` requires very little tweaking out of the box:

```
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
# ==========================================================================
smtp      inet  n       -       y       -       -       smtpd
[...]
virtual   unix  -       n       n       -       -       virtual
[...]



smtp-amavis unix -      -       n       -       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  -       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 smtpd_data_restrictions=reject_unauth_pipelining
    -o smtpd_end_of_data_restrictions=
    -o mynetworks=127.0.0.0/8
    -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
```

## Postfix Virtual Maps

Postfix has the notion of "maps", which is the mechanism used to map data from one form to another. In our case, it's mapping the data in a MySQL database, using a simple select statement, to certain configuration parameters. The map files contain all the information needed to connect to and query the MySQL database, which contains the user information. Each file contains only one query to the database. Create the files, copy and paste the content, and double check the user name and password used to access the database, which I'll cover setting up in the MySQL section.

Please note that this article is geared towards Postfix 2.3 or newer; however, if you plan on using the configuration presented with an earlier version, the format for `mysql_maps` may be different, and broken up into several lines, so make sure to double check which version you are working with.

The file `mysql_virtual_alias_maps.cf` looks like this:

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

`mysql_virtual_domains_maps.cf:`

```
user = mail_admin
password = mail_admin_passwd
dbname = mail
hosts = localhost
query = SELECT domain FROM domain WHERE domain='%s'
```

`mysql_virtual_mailboxes_maps.cf:`

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

**Postfix can store user and domain information in LDAP directories, databases such as MySQL and PostgreSQL or flat files**

# OpenSSL & Certificates

The Postfix configuration listed above makes use of SSL certificates for secure authentication over SSL/TLS.

To create an SSL Certificate:

```
openssl genrsa -des3 -out smtpd.key 1024
openssl req -new -key smtpd.key -out smtpd.csr

# Remove Passphrase from private key -- make optional?
cp smtpd.key smtpd.key.bak
```

```
openssl rsa -in smtpd.key.bak -out smtpd.key

# Generate Self-Signed Certificate
openssl x509 -req -days 365 -in smtpd.csr -signkey smtpd.key -out smtpd.crt
```

# MySQL

Postfix is extremely flexible, and affords a wide array of options in storing user and domain information. The right choice depends mostly on what's appropriate for the environment you plan on working in. If you only plan on hosting a single domain, with a handful of users it may make sense to avoid the overhead of maintaining a database; however, if you plan on maintaining multiple domains with hundreds or thousands of users, or simply want the flexibility of manipulating the data in an automated way, LDAP or MySQL is the way to go.

An in depth discussion of either MySQL or the way Postfix interacts with it is beyond the scope of this article. The only key concept is that of maps in Postfix, which is discussed towards the end of the Postfix section.

## Installing MySQL

```
cd /usr/ports/databases/mysql50-server
make install && make clean
```

You should now have the MySQL database installed. You may need to run the startup script, and make sure it has been enabled in the `/etc/rc.conf` file.

## Creating the mail databases

The `SQL` listing below creates the three tables used by Postfix. Simply follow the steps, create the mail database, and then copy and paste the table creation statements. After that, use the GRANT command to grant access to the `mail_admin` user configured in the Postfix maps described earlier.

Logon to the MySQL database, initially the MySQL root password isn't set:

```
mysql -u root
create database mail;
use mail;

CREATE TABLE `alias` (
  `address` varchar(255) NOT NULL default '',
  `goto` text NOT NULL,
  `domain` varchar(255) NOT NULL default '',
  `active` tinyint(1) NOT NULL default '1',
  PRIMARY KEY  (`address`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COMMENT='Postfix Admin - Virtual Aliases';

CREATE TABLE `domain` (
  `domain` varchar(255) NOT NULL default '',
  `description` varchar(255) NOT NULL default '',
  `aliases` int(10) NOT NULL default '0',
  `mailboxes` int(10) NOT NULL default '0',
  `maxquota` int(10) NOT NULL default '0',
  `transport` varchar(255) default NULL,
```

```
  `backupmx` tinyint(1) NOT NULL default '0',
  `active` tinyint(1) NOT NULL default '1',
  PRIMARY KEY  (`domain`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COMMENT='Postfix Admin - Virtual Domains';

CREATE TABLE `mailbox` (
  `username` varchar(255) NOT NULL default '',
  `password` varchar(255) NOT NULL default '',
  `name` varchar(255) NOT NULL default '',
  `maildir` varchar(255) NOT NULL default '',
  `quota` int(10) NOT NULL default '0',
  `domain` varchar(255) NOT NULL default '',
  `active` tinyint(1) NOT NULL default '1',
  PRIMARY KEY  (`username`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COMMENT='Postfix Admin - Virtual Mailboxes';
```

## Grant privileges

Run the following command:

```
GRANT SELECT on mail.* to mail_admin identified by 'mail_admin_password';
```

## Inserting information

An in depth introduction to SQL is beyond the scope of this article too, but here are some simple insert statements to get you started adding users, aliases, and domains

Inserting a user:

```
insert into mailbox (username, password, name, maildir) values ('my_user_name', 'my_password', 'm
```

Adding a new alias, for an existing user:

```
insert into alias (address, goto) values ('myaddress@mydomain.com', 'myalias@mydomain.com');
```

## Adding a new virtual hosted domain:

```
insert into domain (domain, description) values ('mydomain', 'my_domain_is_so_cool');
```

**Amavisd-new acts as a proxy, accepting the mail from Postfix, and filtering it through ClamAV and SpamAssassin**

# Amavisd-new & SpamAssassin

Amavisd-new is a high performance interface between Postfix and other mail components. Postfix, as well as most other MTAs can only reliably connect to one other content checker such as an anti-virus or spam package, so in order to use both spam and virus filtering we configure Postix to filter content through amavisd-new, which then filters the email through SpamAssassin and ClamAV.

SpamAssassin, is an Apache subproject and is probably the most famous and powerful free software spam filtering program. It uses a wide range of tests to assign an overall score to a piece of mail, if the score is

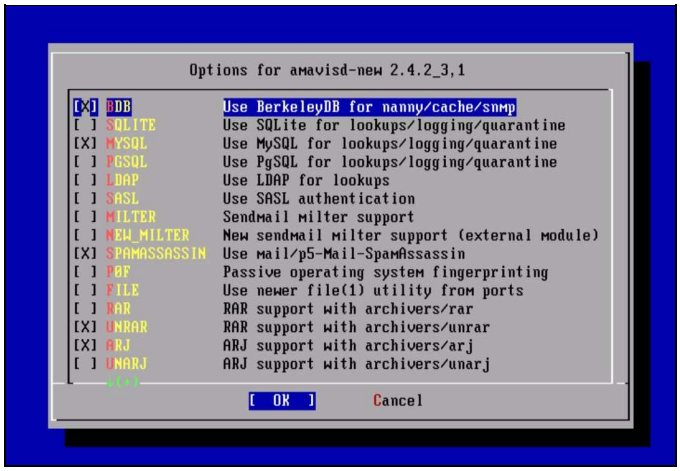
above the acceptable threshold it is considered spam. Amavisd-new comes with SpamAssassin embedded, rather than the stand alone daemon. This simplifies the configuration greatly, since SpamAssassin can be configured in the same place that amavisd-new can. The SpamAssassin directives always begin with `sa` as in `$sa_kill_level_deflt`, which sets the threshold above which SpamAssassin will simply drop an email instead of tagging it as spam.

Amavisd-new is under `/usr/ports/security/amavisd-new`. It should be built with the with the `spamassassin` option, which will automatically build and install SpamAssassin during the amavisd installation.

# Installing amavisd-new

```
cd /usr/ports/security/amavisd-new
make install && make clean
```

You'll be presented with a config dialog. The required options are MySQL, and SpamAssassin. Again, after selecting these, feel free to install other options if you feel adventurous.




Amavisd-new configuration dialog

*Tip:* If you've already installed amavisd-new, and wish to re-configure it, start the process with `make config`, which will re-present you with the configuration dialog.

# Configuring Postfix to work with amavisd-new

It's simple, yet important to configure Postfix to filter all mail through amavisd-new, which then acts as a broker and filters the mail through both ClamAV and SpamAssassin. To do so add the following lines to the `main.cf` file:

```
# filter with amavisd-new which proxies to ClamAV and SpamAssassin
content_filter=smtp-amavis:[127.0.0.1]:10024
```

# Configuring the amavisd-new daemon

Amavisd-new is now installed, and Postfix is filtering to it. The only thing left to do is configure the `amavisd-new` daemon itself. The ports package doesn't create an `amavisd-new` directory under `/usr/local/etc`, unlike most of the other packages. Instead, the `amavisd.conf` is directly under

/usr/local/etc. Similar to the Postfix configuration block above, I'll be mixing explanation tid-bits with the configuration, so make sure to read it thoroughly.

```
# Important lines to check, as I said above, I'm mixing
# in my own comments with the configuration file.

# Make sure these two lines are commented out, as they
# will disable spam and virus filtering

# uncomment to DISABLE anti-virus code
# @bypass_virus_checks_maps = (1);

# uncomment to DISABLE anti-spam code
# @bypass_spam_checks_maps  = (1);


# (no default;  customary: vscan or amavis), -u
$daemon_user  = 'vscan';

# (no default;  customary: vscan or amavis), -g
$daemon_group = 'vscan';

# Configure this to your domain
# a convenient default for other settings
$mydomain = 'mydomain.com';

# Configure the FQDN of your mail host
$myhostname = 'myhost.mydomain.com';

# Change the local_domains_maps to include all
# the mail domains you are hosting
@local_domains_maps = ( [ ".$mydomain", '.example2.com'] );

# IF you want email marked as spam to have a different
# subject header, change the following line
$sa_spam_subject_tag = '***SPAM*** ';

# Default is 6.31, if message is above this
# it will be dropped, and nothing sent to the user
$sa_kill_level_deflt = '20';


# Change the default tag level from 2.0 to undef
# This will insure all mail addressed to domains
# in @local_domains will get a spam score in the header, spam or not.
$sa_tag_level_deflt = undef;

# Make sure this matches what's in main.cf
# listen on this local TCP port(s) (see $protocol)
$inet_socket_port = 10024;

# Configure these two lines to receive email alerts
# when mail is blocked
$spam_admin = 'webmaster@mydomain.com';
$virus_admin = 'webmaster@mydomain.com';

### CLAM_AV INTEGRATION ###
['ClamAV-clamd',
  \&ask_daemon, ["CONTSCAN {}\n", "/usr/share/clamav/clamd.sock"],
  qr/\bOK$/, qr/\bFOUND$/,
  qr/^.*?: (?!Infected Archive)(.*) FOUND$/ ],
```

**ClamAV comes with `freshclam`, an automated process for staying up to date with the latest virus signatures**

# ClamAV

ClamAV is a widely used free software anti-virus package. It is released under the GPL license, and is available for most unix-like platforms. It has many features, but its main goal is an integrated attachment scanner for mail servers. ClamAV is used in a production setting across many private companies, large universities, and government agencies such as: Barracuda Networks, Michigan State University, Brandeis University, Webmail.us, Slashmail, and the US state of Vermont.

ClamAV's configuration file, like that of `amavisd-new` is directly under `/usr/local/etc`, and is called `clamd.conf`. The default configuration is more or less what we need, but it's good to take a peek through the file anyway.

Make sure that mail scanning is enabled:

```
# Example
LogFile /var/log/clamd.log
LogFileMaxSize 5M
DatabaseDirectory /usr/share/clamav

# UNIX Socket which other programs will use
# ie: amavisd-new, will use to connect to ClamAV
LocalSocket /usr/share/clamav/clamd.sock

# Obviously, the user who will own the ClamAV process.
User clamav
```

# Courier-IMAP

Courier-IMAP is the IMAP client which allows Squirelmail, and other mail clients such as Mozilla Thunderbird to connect and download messages. There are actually two components installed:

- `courier-authlib`, which manages user authentication; and
- `courier-imap`, which is the actual IMAP server.

However, this dependency is automatically resolved by FreeBSD's ports, all you need to do is select the `AUTH_MYSQL` option.

## Installing and configuring

```
/usr/ports/mail/courier-imap
make install && make clean
```

Make sure to enable the MySQL option, so the `authmysql` module is built with `courier-authlib`.


Courier configuration dialog

After the installation is complete, there will be two directories under `/usr/local/etc`: `authlib`, and `courier-imap`.

# courier-authlib

`authlib/authdaemonrc`:

```
authmodulelist="authmysql"
```

`authlib/authmysqlrc`:

```
# User Information
MYSQL_CRYPT_PWFIELD     password
MYSQL_GID_FIELD         '125'
MYSQL_UID_FIELD         '125'
MYSQL_HOME_FIELD        '/usr/local/virtual'
MYSQL_LOGIN_FIELD       username
MYSQL_MAILDIR_FIELD     maildir
MYSQL_NAME_FIELD        name

# MySQL Server information
MYSQL_SERVER            localhost
MYSQL_USERNAME          mail_admin
MYSQL_PASSWORD          mail_admin_password
MYSQL_DATABASE          mail
MYSQL_OPT               0
MYSQL_USER_TABLE        mailbox
```

# courier-imap

This puts in place the mechanism to authenticate IMAP requests against the information you have stored in your MySQL database.

`courier-imap/imapd`:

```
# Change this from to 127.0.0.1, so that only
# SquirrelMail has un-encrypted access to IMAP
ADDRESS=127.0.0.1
PORT=143
```

courier-imap/imapd-ssl:

```
# The port, and IP address to listen for ssl/tls encrypted connections.
SSLADDRESS=0
SSLPORT=993
```

# Webmail with SquirrelMail

SquirrelMail is a flexible, easy-to-configure webmail package written in PHP. Configuring Apache is beyond the scope of this article, and there are many other good tutorials out there, so I'm going to focus on installing the necessary packages, and then on configuring SquirrelMail.

## Configuring SquirrelMail

Install Apache:

```
/usr/ports/www/apache22
make install && make clean
```

Install PHP:

```
/usr/ports/lang/php5
make install && make clean
```

Install SquirrelMail:

```
/usr/ports/mail/squirrelmail
make install && make clean
```

Installing SquirrelMail plugins:

```
/usr/ports/mail/squirrelmail-compatibility-plugin
make install && make clean

/usr/ports/mail/squirrelmail-vlogin-plugin
make install && make clean
```

Configuring SquirrelMail:

```
cd /usr/local/www/squirrelmail
./configure
```

1. Select 2 to configure the server, make sure to configure the domain name, and the imap server settings.
2. Select R to return to the main menu.
3. Select 8 to enable plugins, make sure you've enabledthe vlogin plugin.
4. Save and quit!

# Conclusion

I've shown all the pieces required to put together a secure, full-featured, FreeBSD mail server. It was a lot to cover, and I thank you for staying with me. The first article took you through the steps of installing FreeBSD, managing updates, and installing the core server components for a fully functional and secure email server; this article covered configuring the core components for a robust, spam and virus filtering mail system. The journey is by no means over. I strongly encourage you to take the time to explore the packages used, and learn them thouroughly: they are powerful tools which will serve you well in the future.

# Bibliography

http://www.postfix.org/

http://www.clamav.net/

http://www.ijs.si/software/amavisd/

http://spamassassin.apache.org/index.html

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/

## Biography

Yousef Ourabi (/user/43" title="View user profile.): Yousef Ourabi (http://yousefourabi.com) is a developer in the San Francisco bay area. He is currently working at the startup he recently founded, Zero-Analog (http://www.zero-analog.com " title="Zero-Analog). Zero-Analog is currently developing an enterprise application, however, one of its stated goals is "to increase the rate of open source adoption in companies of all sizes, across all industries". Zero-Analog also offers consulting services, all based around open source tools, frameworks and applications.

## Copyright information



---

**Source URL:**
http://www.freesoftwaremagazine.com/articles/secure_email_servers_from_scratch_with_freebsd_6_part_2