

Self-Hosted Email services on OpenBSD

📅 2023-03-14 ➡ [openbsd](#) , [email](#) , [imap](#) , [opensmtpd](#) , [dovecot](#) , [rspamd](#) , [dkim](#)

1. Overview
2. Users
3. Email domains and addresses
4. TLS certificate
5. Receive and send emails
 - 5.1. rspamd
 - 5.2. smtpd
6. Read and manage email
7. It's not DNS
8. One more thing

Looking at [my notes](#) , it seems I haven't setup an email services server from scratch since 2015. Of course, mine have evolved following OpenBSD updates and upgrades.

Let's benefits from the fact that I'm migrating from Vultr to [OpenBSD Amsterdam](#) to write a few notes about the mail server (re)creation. At the time of writing, OpenBSD is available in version 7.2.

“Setting up a mail server” means all and nothing. So to clarify, here's what I want to be able to do:

- as a user, I want to receive emails from the Wild Wild World ;
- as a user, I want to send emails to the Wild Wild World ;
- as a user, I want to manage my mailboxes using several email clients ;
- as a user, I want to avoid unsolicited emails as much as possible ;
- as an administrator, I want to be able to manage several email accounts.

I also want to manage adressbooks and calendars, but that's another story ; and won't be convered here.

Overview

In the ancient times, I did a few things quite differently than I will today. So don't expect an updated version of the previous posts I already did. My configuration is now minimal and enough to deal with 5-10 users.

- Users and password are managed locally - hear in `/etc/passwd`. Every authentication is done this way. No LDAP, SQL database...
- Users are created by a local administrator using an SSH session.
- Passwords are modified by users using an SSH session.
- Email delivery is done by OpenSMTPD using the Maildir format. No use of Dovecot MDA or LMTP features.

- Email protection is done by OpenSMTPD and rspamd.
No use of spamd(8).
- Email submission is done by OpenSMTPD.
It requires the usage of TLS connexions.
It requires authentication to be processed.
Information from the client IP are wiped out.
- Archived emails - hear those not in INBOX - are compressed.

Users

I don't manage a lot of users. So I just create them by hand. Using an SSH connexion to the server, issue a simple classical command:

```
# useradd -m -c "John Doe" -g users -s /sbin/nologin -u 10000 jdoe
```

Then configure the initial user password.

```
# passwd jdoe
```

One way to allow users to change their own password is to create SSH keys for them, turn their shell to /bin/sh and let them run passwd.

Another way is to limit the commands they can run (read the SSH-BASED VIRTUAL PRIVATE NETWORKS section from ssh(1)) but allowing only passwd and configuring an SSH client that will run that command. But this is out of scope here.

All the mail services daemon use /etc/passwd to authenticate users.

Email domains and addresses

Big Tech and most companies offer authentication using email addresses. It is very user friendly. But I don't like this. Because since you know the name of a person, you can guess their login. And that's one less thing to guess for the attackers.

An OpenSMTPD table lists all email domains the server should accept for local delivery.

An OpenSMTPD table lists all email addresses that are legit. It also references every email aliases that I want / need to manage. All those email addresses point to one of the several local users ; those created previously. Read table(5) for more information.

```
# vi /etc/mail/vdomains
tumfatig.net
carnat.net
(...)
```

```
# vi /etc/mail/vusers
abuse@tumfatig.net      root@tumfatig.net
postmaster@tumfatig.net root@tumfatig.net
webmaster@tumfatig.net root@tumfatig.net
```

```
root@tumfatig.net      jdoe
(...)
john.doe@carnat.net    jdoe
jane.smith@carnat.net  jsmith
(...)
```

Note that email and DNS format is not correct. Read and learn, do not simply copy/paste ;)

TLS certificate

To secure the connexions to the server and provide a bit of privacy, a TLS certificate has to be obtained and used by the email daemons. I'll be using Let's Encrypt certificates. Read `acme-client(1)` for more detailed informations.

```
# cp /etc/examples/httpd.conf /etc/
# vim /etc/httpd.conf
# rcctl enable httpd
# rcctl start httpd

# cp /etc/examples/acme-client.conf /etc/
# vim /etc/acme-client.conf
# acme-client -v mail.carnat.net
(...)
acme-client: /etc/ssl/mail.carnat.net.crt: created
```

Receive and send emails

rspamd

Receiving email is dependant on rspamd, for spam checking. And sending email is depending on rspamd for DKIM signing. So first thing to do is install and configure rspamd.

Install the required packages. Note that redis is not strictly required but it will smooth rspamd reactions.

```
# pkg_add rspamd--hyperscan opensmtpd-filter-rspamd redis

# rcctl enable redis
# rcctl start redis
```

Configure rspamd to actually use redis.

```
# vi /etc/rspamd/local.d/redis.conf
servers = "127.0.0.1";
```

```
# vi /etc/rspamd/local.d/greylist.conf
servers = "127.0.0.1:6379";
```

Configure rspamd SPF feature.

```
# vi /etc/rspamd/local.d/spf.conf
spf_cache_size = 1k;
spf_cache_expire = 1d;
max_dns_nesting = 10;
max_dns_requests = 30;
min_cache_ttl = 5m;
```

Configure rspamd DKIM feature.

```
# vi /etc/rspamd/local.d/dkim_signing.conf
domain {
    carnat•net {
        path = "/etc/mail/dkim-carnat•net•key";
        selector = "default";
    },
    (...)
}
```

The documentation for DKIM key generation is available here .

With all this done, rspamd can be started and used.

```
# rcctl enable rspamd
# rcctl start rspamd
```

smtpd

OpenSMTPD uses rspamd and senderscore filters.

```
# pkg_add opensmtpd-filter-senderscore
```

The hostname of the server is not the one the SMTP server is known as. So let's tell OpenSMTPD to use the proper (HELO) name.

```
# vi /etc/mail/mailname
mail•carnat•net
```

Now configure OpenSMTPD.

```
# vim /etc/mail/smtpd.conf
# TLS certificate -----
pki mail cert "/etc/ssl/mail•carnat•net•crt"
pki mail key "/etc/ssl/private/mail•carnat•net•key"
```

```
# Incoming mail filters -----
filter "rdns" phase connect match !rdns disconnect "550 Nope."
filter "fcrdns" phase connect match !fcrdns disconnect "550 Nope."
filter "senderscore" proc-exec \
    "/usr/local/libexec/smtpd/filter-senderscore -blockBelow 10 -junkBelow 70 -slow"
filter "rspamd" proc-exec "filter-rspamd"

# Domain and email tables -----
table vdomains file:/etc/mail/vdomains
table vusers   file:/etc/mail/vusers

# Listeners -----
listen on socket
listen on all tls pki mail filter { "rdns", "fcrdns", "senderscore", "rspamd" }
listen on all port submission tls-require pki mail auth mask-src filter rspamd

# Actions to process -----
action "local_mail" maildir junk virtual <vusers>
action "outbound"   relay helo mail.carnat.net

match from any      for domain <vdomains>    action "local_mail"
match from any auth  for any                  action "outbound"
match from local     for local                action "local_mail"
match from local     for any                  action "outbound"
```

I thought DNS filtering was overkill. But looking at my logs, there are about 200 hits per day for my current rdns and fcrdns filters. So let's keep them.

Check configuration and start the daemon.

```
# smtpd -dvn
configuration OK
# rcctl restart smtpd
smtpd(ok)
smtpd(ok)
```

Emails will now be delivered to the ~/Maildir directory of the configured users. If rspamd is not up when a mail is received, OpenSMTPD will not accept the mail and issue a "451 Try again later" error message. The email will have an opportunity to be delivered later on. As no DNS is properly configured yet, expect to receive only SPAM. Also, email clients configured to use this server right now - to send emails - shall not pass DKIM/DMARC tests.

Read and manage email

Accessing emails with a decent mail client will be provided by Dovecot.

```
# pkg_add dovecot

# vi /etc/dovecot/conf.d/10-mail.conf
(...)
```

```
mail_location = maildir:~/Maildir

# vi /etc/dovecot/conf.d/10-master.conf
(...)
service imap-login {
    inet_listener imap {
        port = 0

# vi /etc/dovecot/conf.d/10-ssl.conf
(...)
ssl = yes
(...)
ssl_cert = </etc/ssl/mail•carnat•net.crt
ssl_key = </etc/ssl/private/mail•carnat•net.key

# vi /etc/dovecot/conf.d/20-imap.conf
(...)
mail_plugins = $mail_plugins imap_sieve zlib
(...)
mail_max_userip_connections = 50

# vi /etc/dovecot/conf.d/90-plugin.conf
(...)
plugin {
    zlib_save = zstd
}
```

Using the rspamd configuration described above, some email may be identified as SPAM and moved to the Junk folder, automatically. But what if there are false-positives. And what if rspamd does not recognise all the SPAM I receive and let some land to my INBOX. Well a way to train rspamd is by using sieve scripts that will be targetted when email clients move emails to / from the Junk mailbox. Moving a mail from INBOX to Junk (or tagging an email as SPAM) teaches rspamd that this is a SPAM. Moving a mail out of Junk teaches rspamd that this mail wasn't a SPAM.

```
# pkg_add dovecot-pigeonhole

# vi /etc/dovecot/conf.d/90-plugin.conf
(...)
plugin {
    zlib_save = zstd

sieve_plugins = sieve_imapsieve sieve_extprograms
sieve_global_extensions = +vnd.dovecot.pipe +vnd.dovecot.environment

# Spam: move from anywhere to Junk or flag changed in Junk
imapsieve_mailbox1_name = Junk
imapsieve_mailbox1_causes = COPY APPEND FLAG
imapsieve_mailbox1_before = file:/usr/local/lib/dovecot/sieve/report-spam.sieve

# Ham: move from Junk to anywhere
imapsieve_mailbox2_name = *
imapsieve_mailbox2_from = Junk
imapsieve_mailbox2_causes = COPY
imapsieve_mailbox2_before = file:/usr/local/lib/dovecot/sieve/report-ham.sieve
```

```
sieve_pipe_bin_dir = /usr/local/lib/dovecot/sieve
}

# vi /usr/local/lib/dovecot/sieve/report-spam.sieve
require ["vnd.dovecot.pipe", "copy", "imapsieve", "environment", "variables"];
if environment :matches "imap.user" "*" { set "username" "${1}"; }
pipe :copy "sa-learn-spam.sh" [ "${username}" ];

# vi /usr/local/lib/dovecot/sieve/report-ham.sieve
require ["vnd.dovecot.pipe", "copy", "imapsieve", "environment", "variables"];
if environment :matches "imap.user" "*" { set "username" "${1}"; }
pipe :copy "sa-learn-ham.sh" [ "${username}" ];

# vi /usr/local/lib/dovecot/sieve/sa-learn-spam.sh
exec /usr/local/bin/rspamc learn_spam

# vi /usr/local/lib/dovecot/sieve/sa-learn-ham.sh
exec /usr/local/bin/rspamc learn_ham

# cd /usr/local/lib/dovecot/sieve/
# chmod 0755 *sh
# sievec report-ham.sieve
# sievec report-spam.sieve
```

The previous magic spells tell Dovecot to run some sieve scripts when certain mail movement condition are met ; like mark as SPAM or move out of Junk folder.

The sieve scripts end piping the emails to some shell scripts.

The shell scripts call the rspamd commands required to learn a new SPAM or a new HAM message.

It is then time to use the complete Dovecot configuration.

```
# rcctl enable dovecot
# rcctl start dovecot
dovecot(ok)
```

The email clients can now be pointing at the IMAPS port of the server.

It's not DNS

Sometimes, it is DNS. To be able to receive email from the Wild Wild World, the mail domains will require proper MX records. And to be able to send email to the World lowering the chances of being rejected as SPAM, the mail domains need a couple of DNS records.

Setup the server's DNS records:

- an A record referencing the IPv4 server address.
- an AAAA record referencing the IPv6 server address.

- an IPv4 PTR record matching the A record.
- an IPv6 PTR record matching the AAAA record.

Tell the world that the server can receive emails for your domain(s):

- an MX record referencing the server's name.

If the server's name is resolvable using IPv4 and IPv6, then you shall receive email from servers using both stack.

Tell the world that the server is legit to send emails for your domain(s) ; and that others aren't.

- a TXT record referencing the DKIM domain key.
- a TXT record referencing the SPF domain rules.
- a TXT record referencing the DMARC domain policy.

You can have a look at what others do. At least to see if you understood the docs properly and planned TXT records that seem classical.

```
# dig +short openbsd.org txt
"v=spf1 mx a:mail.openbsd.org a:cvns.openbsd.org ~all"

# dig +short txt _dmarc.openbsd.org

# dig +short txt gmail.com
"v=spf1 redirect=_spf.google.com"
"globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="

# dig +short txt _dmarc.gmail.com
"v=DMARC1; p=none; sp=quarantine; rua=mailto:mailauth-reports@google.com"

# dig +short _spf.google.com TXT
"v=spf1 include:_netblocks.google.com include:_netblocks2.google.com
include:_netblocks3.google.com ~all"
```

Once done, you may use tools such as [DMARCLY](#) to verify that your DNS records are configured properly.

You can also use a free service like [DKIMValidator](#) to validate that the email you send are configured properly.

One more thing

rspamd has a nice reporting web GUI. By default, it only listens on localhost and access is granted unauthenticated. Steps are required to set a password. One way to access the Web page is to use SSH port forwarding feature.

```
# ssh -L 11334:localhost:11334 <mail server>
```

Browse to <http://localhost:11334/> and enjoy.

There probably are ways to grab those metrics and send them to InfluxDB so that they can be rendered using Grafana. So is the case of smtpd logs. But that's another story.

Happy mailing!

2023-04-26 Update: Dale Carstensen pointed out performance tricks that I actually implement but did not document here. Thank you Dale for notifying me. Here they are.

If you get "Too many open files" errors, you can create a dedicated daemon class and increase the default limits:

```
# cat > /etc/login.conf.d/dovecot
dovecot:\
:openfiles-cur=1024:\
:openfiles-max=2048:\
:tc=daemon:
^D

# usermod -L dovecot _dovecot
# userinfo _dovecot
(...)
class    dovecot
(...)

# rcctl restart dovecot
```

This can be done with every daemon that suffers from limit issues. You may use a single class for many daemons or configure a class per daemon.

Using rspamd(8) can be very DNS queries consuming. To improve responsiveness, you may run a local DNS cache. I'm using the stock unbound(8) daemon as a default resolver and fallback to an external one:

```
# vi /var/unbound/etc/unbound.conf
(...)
do-ip6: yes
(...)
tls-cert-bundle: "/etc/ssl/cert.pem"
(...)
forward-zone:
  name: "."
  forward-tls-upstream: yes
  forward-first: no
  forward-addr: 2620:fe::fe@853#dns.quad9.net
  forward-addr: 2620:fe::9@853#dns.quad9.net
  forward-addr: 9.9.9.9@853#dns.quad9.net
  forward-addr: 149.112.112.112@853#dns.quad9.net




# rcctl enable unbound
# rcctl start unbound

# vi /etc/resolv.conf
lookup file bind
```

```
nameserver 127.0.0.1
nameserver 9.9.9.9
nameserver 2620:fe::fe

# rcctl restart rspamd
```

Related posts:

-  [Back to the sea ; the Certificate Authority \(CA\), episode IV](#)
-  [Update Lenovo X260 BIOS with OpenBSD](#)
-  [Using Munstrap, Munin's theme, on OpenBSD](#)



© 2003-2024 Joel Carnat. Powered by [OpenBSD](#) , [httpd\(8\)](#) , [relayd\(8\)](#) and [Hugo](#).

