Setting up postfix

This is a document copied from https://www.tornevalls.se/2019/08/12/the-postfix-maildir-guide/

There is also an addon for smtpauth, which is included in the first post, reposted at https://www.tornevalls.se/2020/10/04/postfix-smtpauth-and-clients-without-imap-pop-auth/

Document content

- Document content
- Let's get started
- What we currently miss in this document
    - Details
        - The backsides of DigitalOcean
        - DigitalOcean and IPv6 - not allowed
- So how do I get postfix installed?
    - How about spamassassin checking?
- Next?
- Is this it?
    - Breaking in with complex structures
    - Let's continue...
    - Edit /etc/postfix/sasl/smtpd.conf and add this row to it:
    - Do not forget the certificates for letsencrypt
    - And DKIM?
    - Do you remember the milter rows?
- SpamAssassin
- Clients that does not support SMTP authentication via imap or pop
    - The solution?
    - The documented problem URLs

Greylisting
- Installing
- Are we done?

Let's get started

For a long time ago, I decided to install qmail. I loved it. It felt so easy and simple, running qmail and - at the time - courier-mta. But time flies by and I realized, all of a sudden, that I got stuck in something evil with qmail. At this moment, many people had already dropped qmail for something better.

But that world scared me. I had no spare time to really dig into a new project, new configuration and new user accounts to handle. The worst part was that qmail needed recompilation and new setups to work entirely safe. And I had no spare time for this either. However, this summer - I decided to try anyway. To go postfix. The configuration was however the same, so I guessed that odds that things should go easy to keep most of the prior "setup"; the courier-mta. In this case, it was the Maildir functions I still needed. A lot of other options was ready for me out there, but I had no doubts that courier-mta was "the shit".

Unfortunately, I have a feeling that people in common doesn't agree. So many postfix setups are written for other setups. And this does not cover my needs at all. Also, I tried to find documents that explained to me how to set up a remote spamassassin configuration. I found nothing. But those parts is not so bad either.

So what does this "guide" cover? Well, my needs are probably quite weird:

- Postfix with virtual domains and virtual users.
- Postfix with a proper TLS and certificates by letsencrypt.
- SpamAssassin.
- DKIM.
- SPF.
- DMARC.
- Moving spam into another folder.

Most of those docs could be found, by searching googling and spending half a life compiling documents, scripts, etc to make this an entire e-mail system. However, it is a real pain in the ass. So I'll start here with my own compilation. It may STILL lack information that YOU need, however, this is some kind of completion of what others failed to do: Collect information.

What we currently miss in this document

A relaying account. Meaning, when you want to use a relay smtp server that requires a login to be able to relay mail.

So what are the requirements?

Preferably something that you can just "apt" everything you need with. How about Ubuntu? Version? 18.04 or later. Which gives us not xenial, but bionic. Preferrably a lot of extras too. Remember that I'm a qmail idiot initially. I did run spamassassin remotely, which is - if you try a very default postfix installation - also bound to use sendmail as one of the core functions in postfix. This installation is based on a "core basic bionic ubuntu" installation. In other words VANILLA installation from a VPS point of view. I was aware that every VPS has their downsides, but this installation was initially made from a DigitalOcean instance. However, I also installed a simpler failover instance at GleSYS.

Details

Details and function is very important for me, so this post also describes how pedantic I am when it comes to function. There are nothing left to chance. If I see that my own visions of how a mailserver works don't work, I find another solution for it. Like the bottlenecks of DigitalOcean.

The backsides of DigitalOcean

I like DigitalOcean. However, as of July/August 2019, they seem to have some kind of bad reputation as e-mail provider. I recently found some graphs of the throttling spam reputation at DO, and I may post it here if I can find it again. However, it takes a bit of time to get DO ip's delisted from blacklists that has this provider marked as spammers. So, during this installation I decided to ditch the outgoing relay via DO and run an extra relay somewhere else. Using my own ISP is out of question, mostly because this requires extra authentication. Besides using my credentials with other users that wants to relay their mail may end up in that I get blamed is something happens. So, I created two failover sites: One hosted by GleSYS and one hosted by own VPS servers, via a network range at Spacedump AB. The failovers is this far known as pure and are allowed to do outbound SMTP.

DigitalOcean and IPv6 - not allowed

To be entirely sure that everything really works, if I decide to go outbound via DigitalOcean, I also created by own IPv6-tunnel to the VPS as DigitalOcean is NOT allowing outbound IPv6 SMTP traffic at all. The moments that this VPS really needs a proper IPv6-host to send mail by, I used my own tunnel to bounce all e-mail through my own allowed IPv6-prefix. So, no I do not use Hurricane Electric for this purpose and I can probably run this through GleSYS too. I'm not sure how HE handles IPv6-smtp today, and that's one of the reasons that makes me not want to dig into that hole.

So how do I get postfix installed?

Initially, in my own base document, I splitted up the installation instructions in a few blocks. The mailserver is installed with following components (also read below, since you may need more during):

- Postfix
- Spamassassin (for using spamc, not checking spam)
- Clamav for possible virus checks
- Courier-MTA (imap/pop/etc)
- Fetchmail (for fetching email from other providers)
- Procmail (for moving spam into specific folders)
- DKIM-tools
- DMARC-tools
- Certbot - Letsencrypt

This is how the aptification goes:

```
apt-get install postfix spamassassin clamav courier-imap courier-pop courier-authlib-pipe gamin sasl2-bin
libsasl2-2 libsasl2-modules spamc fetchmail procmail certbot
apt-get install opendkim opendkim-tools opendmarc
```

Separate installations

The above installation is far away from enough as we need to configure this too. Also note that many of the local install scripts I use is also splitted up like this:

1. **Primary mailserver, handles landing incoming e-mail. Also handles user accounts and outbound traffic. This requires for example courier or another inbound-mail daemon. However, as I'm used to use courier, this was the first choice.**
2. **Relay servers. They don't need user accounts, but they need to handle their own DKIM signings.**
3. **Some relay servers are dependent on the primary mailserver as they use dynamic ip-addresses. For example, my primary network are dependent on a DHCP link, which is normally not allowed to send mail (neither with relaying or as servers itself as dns blacklists blocks relaying dhcp-based servers). This requires the mailserver to be relaying over note 1. This may from time to time also requires smtpauth (see new posts about the SMTP-auth problems where no pre-imap/pop-authentication are made). The servers however still needs to sign their mails to not get blocked by the primary inbound.**

How about spamassassin checking?

The spamassassin server is separately installed. Actually, spamassassin are running through an AWS (amazon free tier to be exact). The installation looks like this:

```
apt-get -y install spamassassin spamc razor pyzor
# If you run on focal, you will get a problem with the spambayes candidate. So that will run separately (for
example for bionic).
# E: Package 'spambayes' has no installation candidate
apt-get -y install spambayes
apt-get -y install libcrypt-ssleay-perl libio-socket-ssl-perl razor libnet-ident-perl libdbi-perl pyzor libmail-
dkim-perl
apt-get -y install opendkim opendkim-tools opendmarc
```

As you can see above, the spamassassin server has pyzor and razor as options. This is quite weird part, that mostly actually works but needs initialization first. The packages I used for several years ago stinks, since they was manually installed. Gladly, those were apt-gettable. The initialization was quite simple. As long as your paths are correctly installed, as there are some places to keep in mind that may differ. But first, the initialization:

```
razor-admin -home=/etc/spamassassin/.razor -register
razor-admin -home=/etc/spamassassin/.razor -create
razor-admin -home=/etc/spamassassin/.razor -discover
```

To keep in mind

- /etc/mail
- /etc/spamassassin
- /etc/mail/spamassassin

During my years with qmail, I'm no longer sure about how those paths are linked together. I have several scripts that makes sure that everything else is set up properly, based on newer and older information (this is very unfortunate). For example /etc/mail/spamassassin is symlinked to /etc/spamassassin, just to be sure that applications finds everything in one place and not bombed all over the place.

Next?

The configuration may be very frustrating from time to time. While installing postfix, courier, etc, you may get warnings from the system about courier users and SSL certificates. When I started this project I chose to ignore it, until I got control over postfix. So how do my postfix configuration look? Because this is a quite important part of everything. There's a lot of things to think of to really make this work out on all levels. This is my level requirement list:

- Postfix should support virtual domains and users
- Postfix should support spamassassin, make use of standard sendmail and spamassassin without weird extra installations. People out there seem to be extremely complex (and stupid) from time to time.
- My networks should be trusted in a very tiny setup; only relays should now about each other.
- The entire world should, primarily, use authentication to be allowed to SEND mail from postfix.
- The rest of the world should be prohibited to relay e-mail via postfix, but STILL deliver mail to the virtual domains.

The above combination is quite interesting, because on the way through this configuration I found out that postfix is not allowing "relaying" unless it is properly configured. By means, doing all this setups should still allow any other useraccount-based email sending. As it seems, many people have huge problems with this combination. Besides, sendmail is - as I can see - the base core of postfix in some "cases", but people also seem to not use this solution. Don't ask me why. Probably it is something like "we cannot make everything work together by using default applications". As I want a standard installation to be extended, I went through with this. So here goes.

Some of the rows below are changed to fit my needs. For example, the standard smtpd_banner is ugly. And it reveals quite a lot of why I run. So this got replaced with my own half static banner, as I still wanted to identify the mailserver by hostname (so I could get use of more, know which I use and to be compliant with other spamcheckers online).

```
# myhostname is fetched from /etc/hostname if not set here. Together with mydomain in this configuration
# your servername will be quite accurate in it's greeting.
smtpd_banner = $myhostname ESMTP $mail_name
biff = no

# The basic stuff.
mydomain = #my-zone-name#
myorigin = /etc/mailname
mydestination = $myhostname, localhost

# appending .domain is the MUA's job.
append_dot_mydomain = no
readme_directory = no
compatibility_level = 2

# TLS parameters
smtpd_tls_cert_file=/etc/letsencrypt/live/#my-zone-name#/fullchain.pem
smtpd_tls_key_file=/etc/letsencrypt/live/#my-zone-name#/privkey.pem
smtpd_use_tls=yes
```

```
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

# Authentication
smtpd_sasl_auth_enable = yes
broken_sasl_auth_clients = yes
smtpd_sasl_security_options = noanonymous
smtpd_sasl_authenticated_header = yes
smtpd_recipient_restrictions = permit_mx_backup permit_auth_destination permit_sasl_authenticated
permit_mynetworks reject_unauth_destination
smtpd_sasl_path = smtpd

# SMTP Settings
smtpd_tls_loglevel = 3
smtpd_relay_restrictions = permit_mx_backup permit_auth_destination permit_mynetworks permit_sasl_authenticated
defer_unauth_destination

alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
relayhost = #my-real-relay-server#:#safe-smtp-port#

# NOTE: This should be replaced with trusted addresses
mynetworks = 127.0.0.255
 127.0.0.0/8
 ip/32
 ip-subnet-range/28
 [::ffff:127.0.0.0]/104
 [::1]/128
 [ip-subnet-range]/48

mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
# Warning: Run on DigitalOcean? Make sure ipv6 are tunnelled or disabled.
inet_protocols = all

fallback_relay = #my-safe-relay-backup-server#:#safe-smtp-port#

# Do not list major domain in BOTH virtual_alias_domains and relay_domains
# Virtual Aliases, Splitted by zone
virtual_alias_maps =
 regexp:/etc/postfix/virtual/virtual_users_regex
 hash:/etc/postfix/virtual/virtual_users_#zone-1#
 hash:/etc/postfix/virtual/virtual_users_#zone-2#
 hash:/etc/postfix/virtual/virtual_users_common

recipient_bcc_maps = hash:/etc/postfix/recipient_bcc_maps
virtual_alias_domains = hash:/etc/postfix/virtual_domains
virtual_mailbox_domains = /etc/postfix/virtual_domains
relay_domains = hash:/etc/postfix/relaydomains
header_checks = regexp:/etc/postfix/header_checks
#smtp_header_checks = regexp:/etc/postfix/smtp_header_checks
relay_recipient_maps = hash:/etc/postfix/relay_recipient_maps

transport_maps = hash:/etc/postfix/transport

# Maildir setup
virtual_mailbox_base = /var/spool/mail
home_mailbox = Maildir/

# Milter Configuration
milter_default_action = accept
milter_protocol = 6

# SMTPD MILTERS - RUNNING TCP INSTEAD OF SOCKETS ETC AS THIS IS WAY TO BUGGY. DO NOT TRUST ANY DOCUMENTS OUT
THERE OF HOW
# TO GET THIS FIXED!!
smtpd_milters = inet:#tcpport#
non_smtpd_milters = inet:#tcpport#

mailbox_command = /usr/bin/procmail -a "$EXTENSION" DEFAULT=$HOME/Maildir/ MAILDIR=$HOME/Maildir
```

```
spamassassin_destination_recipient_limit = 1
```

```
I presume there is risks with just cut-n-paste stuff into the main.cf - I configured the above file from scratch,
step by step to make sure everything really worked all the time. For example, I started with the authentication
parts - which first gave me problems with relaying rules, etc etc. So be careful.
```

Is this it?

Absolutely not!

master.cf must be updated too. For example, the smtp service that should also run itself through spamassassin:

```
smtp      inet  n       -       y       -       -       smtpd -o content_filter=spamassassin
 587       inet  n       -       y       -       -       smtpd -o content_filter=spamassassin
```

As you can see, I've also opened port 587 as an extra SMTP port. This is normally considered secure SMTP, but supports anything that falls into that port. To make sure spamassassin run it remote checks, this is also added in the master.cf:

```
spamassassin unix -     n       n       -       -       pipe
         user=debian-spamd argv=/usr/bin/spamc -H -d spamd.tornevall.net -f -e /usr/sbin/sendmail -oi -f
${sender} ${recipient}
```

You can of course open more ports to not handle data as strict as the primary ports are doing by for example adding

```
26 inet n - y - - smtpd
```

In this example, we're skipping the spamassassin usage. Just remember that this makes your system go "not as safe as the standards".

Now this is the only part of master.cf that is really edited. To not fall out of a configuration that I can't control, most of the configuration are centralized to main.cf - for example the procmail above. This is the only thing in postfix that I need to set there, to make procmail push spam into the special folder Spam.

```
mailbox_command = /usr/bin/procmail -a "$EXTENSION" DEFAULT=$HOME/Maildir/ MAILDIR=$HOME/Maildir
```

I tried to avoid as much as "extra configuration" as possible however. So most of the other applications is untouched if possible. For courier, I allow more connections at once as the "queue" may get full quite fast. I also have a script that does job for me, that I normally forget.

For example, Virtual users, domains, etc, is hashed. Each time I change the base raw text file, they need to be "recompiled". This is done with automation. I also have a script that fixes a lot of symlinks. For example, I use git to get track of my postfix configuration. The script used, makes sure that everything gets set up for me, besides of the main.cf and other postfix data.

A bash script is making sure that the following files exists, so that postfix can't complain.

```
/etc/postfix/virtual_users
/etc/postfix/virtual_domains
/etc/postfix/header_checks
/etc/postfix/relaydomains
/etc/postfix/relaydomains
/etc/postfix/relay_recipient_maps
/etc/postfix/transport
/etc/postfix/access-inbound
/etc/postfix/virtual_users_regex
```

Another script (Makefile) is making sure that the files are really hashed, and yet another script makes sure that all virtual users in the path /etc/postfix/virtual is up to date too. However, this is the main runner:

```
@postalias /etc/aliases
@postmap /etc/postfix/virtual/virtual_users
@postmap /etc/postfix/recipient_bcc_maps
@postmap /etc/postfix/virtual_domains
@postmap /etc/postfix/header_checks
@postmap /etc/postfix/relaydomains
@postmap /etc/postfix/relay_recipient_maps
@postmap /etc/postfix/transport
@postmap /etc/postfix/access-inbound
@newaliases
```

For courier, I sometimes got warned about a shared index. The above mentioned script makes sure that it does exist with...

```
echo "/etc/courier/shared/index missing?!"
touch /etc/courier/shared/index
chown daemon:daemon /etc/courier/shared
chmod 0755 /etc/courier/shared
service courier-authdaemon restart
service courier-imap restart
service courier-pop restart
service courier-imap-ssl restart
service courier-pop-ssl restart
```

Breaking in with complex structures

Having a lot of domains and services to take care of may make only one file with all virtual users a bit junky. For my own current solution, this has been splitted up in parts and the postmap scripts are checking via cronjobs if the files has been changed. This is being made so files can be automatically transferred (if necessary) to the server, and so the server then can reproduce new data tables for the virtual users.

```
virtual_alias_maps =
        hash:/etc/postfix/virtual/virtual_users_tornevall.net
        hash:/etc/postfix/virtual/virtual_users_tornevall.se
        hash:/etc/postfix/virtual/virtual_users_fraudbl.org
        hash:/etc/postfix/virtual/virtual_users_tornevall.org
        hash:/etc/postfix/virtual/virtual_users_small_domains
        hash:/etc/postfix/virtual/virtual_users
        regexp:/etc/postfix/virtual/virtual_users_regex
```

Let's continue...

I also make sure that procmail is set up properly. As I want spam from spamassassin moved to a specific mail folder /etc/procmail contains this:

```
:0
 ^X-Spam-Flag: YES
 $DEFAULT/.Spam/
```

I also had a lot of issues with the sasl-auth that really required extra work until I found out what the problems was (sometimes authentication fails with SASL and TLS). This is fixed with...

sasl.sh

```
#!/bin/bash
rl=$(readlink /var/run/saslauthd)
if [ "" != "$rl" ] ; then
                echo "/var/run/saslauthd is symlinked already."
                exit
fi

mkdir -p /var/spool/postfix/var/run/saslauthd
chgrp sasl /var/spool/postfix/var/run/saslauthd
adduser postfix sasl
# Problems with /var/run/saslauthd that should be fixed. If it exists, remove it for instance...
rmdir /var/run/saslauthd >/dev/null 2>&1
# Then symlink it away.
ln -sv /var/spool/postfix/var/run/saslauthd /var/run >/dev/null 2>&1
if [ "$?" != "" ] ; then
        echo "Obviously something is wrong here."
        theDate=$(date +'%Y%m%d%H%M')
        mv -v /var/run/saslauthd /var/run/saslauthd-${theDate}
        ln -sv /var/spool/postfix/var/run/saslauthd /var/run
fi

# By the way, if saslauthd is not autostarted, it should really be changed.
sed -i 's/START=no/START=yes/' /etc/default/saslauthd

# Do not forget to restart the daemons.
service postfix restart
service saslauthd restart
```

Edit /etc/postfix/sasl/smtpd.conf and add this row to it:

```
pwcheck_method: saslauthd
```

Do not forget the certificates for letsencrypt

This is made easy out from the basic courier configuration, I use the defaults and replacing them - WHEN LETSENCRYPT ARE RENEWING THE CERTIFICATES FOR ME - with this automation:

```
#!/bin/bash
cd /etc/postfix
copyfrom=/etc/letsencrypt/live/
domain=tornevall.net
extra=""
destination="/etc/courier/imapd.pem"

cat ${copyfrom}/${domain}${extra}/privkey.pem ${copyfrom}/${domain}${extra}/fullchain.pem >${destination}
chmod 600 ${destination}
```

And DKIM?

DKIM is very special as it needs keys to be added into the DNS. The DNS update is not automated in this document (yet). I just generate the keys and then paste them into the DNS.

However, it takes time doing this manually so this is the script that creates what I need – OBSERVE THAT THIS IS ONLY AN EXAMPLE ON HOW WE DO IT:

dkim.sh

```
#!/bin/bash

YYYYMM=$(date +%Y%m)
```

```
path=/etc/opendkim/keys
okey=$(which opendkim-genkey)

dkimu=$(grep dkim /etc/group|grep postfix)
if [ "" = "$dkimu" ] ; then
 echo "postfix user must be present in group for dkim"
 exit
fi

if [ ! -d /etc/opendkim ] ; then
 ln -sv /var/tornevall/system/etc/opendkim /etc/opendkim
fi

rl=$(readlink /etc/opendkim.conf)
if [ -f /etc/opendkim.conf ] ; then
 if [ "" = "$rl" ] ; then
 mv /etc/opendkim.conf /etc/opendkim.conf.old
 ln -sv /var/tornevall/system/etc/opendkim.conf /etc/opendkim.conf
 fi
 chmod u=rw,go=r /etc/opendkim.conf
 chmod u=rw,go=r /var/tornevall/system/etc/opendkim.conf
else
 if [ -f /var/tornevall/system/etc/opendkim.conf ] ; then
 ln -sv /var/tornevall/system/etc/opendkim.conf /etc/opendkim.conf
 fi
fi

if [ "" = "$okey" ] ; then
 echo "opendkim missing, trying to install..."
 apt-get -y -f install opendkim opendkim-tools >/dev/null 2>&1
 okey=$(which opendkim-genkey)
 if [ "" = "$okey" ] ; then
 echo "Could not install opendkim ..."
 exit
 fi
 echo "OK, ready!"
fi

if [ "" = "$2" ] ; then
 echo "Usage: $0 domain shortname"
 exit
fi

domain=$1
short=$2

if [ ! -f /etc/opendkim/signing.table ] ; then
 touch /etc/opendkim/signing.table
fi
if [ ! -f /etc/opendkim/key.table ] ; then
 touch /etc/opendkim/key.table
fi

echo "$domain - /etc/opendkim/signing.table"
dom=$(grep $domain /etc/opendkim/signing.table)
if [ "" = "$dom" ] ; then
 echo "*@${domain} ${short}" >>/etc/opendkim/signing.table
 echo "Added $domain (short=$short) to /etc/opendkim/signing.table"
else
 echo "Domain already exists in signing.table - please remove it from that table and the key.table to start
over."
 exit
fi

dom=$(grep $domain /etc/opendkim/key.table)
if [ "" = "$dom" ] ; then
 echo "${short} ${domain}:${YYYYMM}:/etc/opendkim/keys/${short}.private" >>/etc/opendkim/key.table
 echo "Added $domain (short=$short) to /etc/opendkim/keys/${short}.private"
fi

echo opendkim-genkey -b 2048 -h rsa-sha256 -r -s ${YYYYMM} -d $domain -v
```

```
opendkim-genkey -b 2048 -h rsa-sha256 -r -s ${YYYYMM} -d $domain -v

rm -vf keys/${short}.private
rm -vf ${short}.txt

mv -v ${YYYYMM}.private keys/${short}.private
mv -v ${YYYYMM}.txt ${short}.txt

chown -R opendkim:opendkim /etc/opendkim
chmod -R go-rwx /etc/opendkim/keys

echo "Fixing SHA-problem by sed ..."
sed -i 's/h=rsa-sha256/h=sha256/' ${short}.txt

chown opendkim:postfix -Rv /etc/opendkim/

echo "Restarting dkim ..."
service opendkim restart
```

The above script creates a proper setup for DKIM, that you later on can paste without problems into your DNS. This setup also supports a sha256-issues that is known to cause problems in the key validation. rsa-sha256 should actually be sha256 only. Which is also fixed here.

Do you remember the milter rows?

```
# SMTPD MILTERS - RUNNING TCP INSTEAD OF SOCKETS ETC AS THIS IS WAY TO BUGGY. DO NOT TRUST ANY DOCUMENTS OUT
THERE OF HOW
# TO GET THIS FIXED!!
smtpd_milters = inet:#tcpport#
non_smtpd_milters = inet:#tcpport#
```

This part is actually written for DKIM. I gave up with sockets in a very early phase of this project as I never got it to work. I instead, run opendkim by tcp. In **/etc/opendkim.conf** you'll find the data below (make sure you change #tcpport# to something you like and is not firewalled):

```
# Log to syslog
Syslog yes
# Required to use local socket with MTAs that access the socket as a non-
# privileged user (e.g. Postfix)
UMask 007

# OpenDKIM user. Remember to add postfix to group opendkim.
UserID                opendkim

KeyTable        /etc/opendkim/key.table
SigningTable        refile:/etc/opendkim/signing.table

# Hosts to ignore when verifying signatures
ExternalIgnoreList  /etc/opendkim/trusted.hosts
InternalHosts       /etc/opendkim/trusted.hosts

# Commonly-used options; the commented-out versions show the defaults.
Canonicalization relaxed/relaxed
Mode sv
SubDomains no
AutoRestart     yes
AutoRestartRate     10/1M
Background      yes
DNSTimeout      5
SignatureAlgorithm  rsa-sha256

Socket inet:#tcpport#@localhost
PidFile               /var/run/opendkim/opendkim.pid
OversignHeaders From
TrustAnchorFile       /usr/share/dns/root.key
```

SpamAssassin

SpamAssassin is very specific configuration. I will not cover the basic setup for SpamAssassin here as it usually works as is. Besides, I run SpamAssassin via mysql-setups and rules, which is a very personal setup. Normally, you won't need think of it here. The important thing with this post is the actual postfix setup.

Clients that does not support SMTP authentication via imap or pop

This text is written in october 2020 after ripping my hair of my head off for a while. What I did not think of, during the first round of installation, was that there will be non standard clients that won't do a pop/smtp-auth before entering the SMTP out. For example, Postfix, straight out of the box - where you want to relay from postfix to postfix via an authenticated user. With the solution above, there might happen things that you do not want. The error message below for example, is quite common but very much unanswered in different kinds of forums. Most of the posts are relating their problems to dovecot, cyrus and different kind of solutions that in the end seem to be database driven. This is not bad, it's just a little bit stupid since you suddenly rely your systems on yet another point of failure: The database. And the more crap you implement, the harder it will be to find the failing point.

```
warning: SASL authentication failure: unable to canonify user and get auxprops
```

However, the solution may be much simpler than you think in this case. It was first, when I stumbled over theURLs below URLs, I realized that some settings are just misconfigured with the sasl authentication daemon.

As the first step after finding the entire solution, I tried to change the auth mechanism to a shadow based solution as I don't like extra databases to just make authentication work. This however failed, and since the server itself is in a production state I have to stick with the db solution for a while (*because I actually don't know if this have effect on other working systems currently in operation)*.

The solution?

Well, since at least one of the sites are mentioning chrooted files, saslauthd won't read the "real" /etc/sasldb2, since it's not really in /etc - the real file resides in /var/spool/postfix/etc, and requires only one thing - that you create it and put it there and making the sasl user the group owner of the -file. This is how it should look.

```
# ls -l /var/spool/postfix/etc/sasldb2
-rw-r----- 1 root sasl 12288 Oct 4 12:00 sasldb2
```

This file contains the users that can authenticate without the pre-auth system.

The only backside of this is that you may have another load of users in another location /etc/passwd, that still won't be able to authenticate as long as they are not added to sasldb2.

Example of how to create it "properly" ***first time***:

```
#> saslpasswd2 -c nisse
Password:
Again (for verification):
#> mv /etc/sasldb2 /var/spool/postfix/etc
#> ln -s /var/spool/postfix/etc/sasldb2 /etc/sasldb2
```

The documented problem URLs

https://serverfault.com/questions/409828/cant-get-sasl-auxprop-sasldb-working-with-postfix-ubuntu-12-04
https://annvix.com/enabling_sasl_in_postfix

Greylisting

Preventing spam can be very exhausting. There is a suggested method on stopping spam by greylisting. You can read a longer description of how to set up greylisting for your system on the link below.

https://www.howtoforge.com/greylisting_postfix_postgrey

Installing

Run: apt install postgrey and service postgrey start

```
apt install postgrey
service postgrey start
```

If you have been following the above instructions, your main.cf probably looks like this:

```
smtpd_recipient_restrictions = permit_mx_backup, permit_auth_destination, permit_sasl_authenticated,
permit_mynetworks, reject_unauth_destination, check_policy_service unix:private/policyd-spf
```

Update this row to:

```
smtpd_recipient_restrictions = permit_mx_backup, permit_auth_destination, permit_sasl_authenticated,
permit_mynetworks, reject_unauth_destination, check_policy_service unix:private/policyd-spf,
check_policy_service inet:127.0.0.1:10023
```

Then, reload postfix.

Are we done?

I'm not entirely sure actually. I may have missed something. If someone is actually interested in this post and see that something won't work it may be cause by the fact that I missed that part. I also have a separate configuration (based on this big one) to make sure that relaying actually works out of the box. If you find anything, feel free to contact me and notify me about it...