GenPDSDM manual

Generate a system with **P**ostfix and **D**ovecot, with the available security features, like virus scanning, **S**PF, **D**KIM and D**M**ARC on a Raspberry Pi with Raspberry Pi OS

Introduction

This manual is about a bash script which describes a procedure to configure a system with **Raspberry Pi OS lite** on a Raspberry Pi to act as an email server for a specific domain. The domain name used in this document is example.com, but should obviously be replaced by the domain name you want to serve with this system.

This system is also configured to be an IMAP server. Both servers support encrypted access and the email server encrypted sending of messages.

SPF is used to protect e-mail with your domain name in the domain part of your email address to be send by other servers than your own server and to check incoming e-mail if the incoming connection comes from a trusted server. However only servers that do SPF checking will reject a message with your domain name in the domain part of your email address.

DKIM is used to sign your email message, so the receiver can check if it is coming from you. It is up to this server what to do when this check fails. Your server will also do this checking.

With DMARC you can indicate to the receiving server what should be done when a SPF and/or DKIM check fails.

The author uses quite a number of different email addresses in different roles. Most of the time this is supported by several email clients, like Thunderbird and KMail, where these roles are named identities. For these different identities an e-mail needs to be send to the e-mail servers (relay hosts) associated with these identities. However sometimes processes running in your systems need to send email as these identities. This script also handles this type of processing.

Preliminary actions

Prepairing the boot device of the server

Please follow the getting started guide on
https://www.raspberrypi.com/documentation/computers/getting-started.html

to prepare an SD card or, if possible, a USB device for your system. It is assumed that you did choose the lite version of Raspberry Pi OS, either the 32 or the 64 version. This manual has been

written using the 32 bit version. Furthermore it is assumed that you did enable ssh access to this system, because this system is supposed to be a headless server.

Connect to the server

Now you need to find the IP address of the system. It will get an IP address via DHCP, so your router might provide this address. Because the system is assumed to be headless you connect to the system via ssh with:

```
ssh <configured_username>@<IP address>
```

This will ask you to accept the key from this system and asks you for the password you configured in preparing the boot device.

I prefer to do the rest of the setup as root, so I do not need to use `sudo` in front of every command that needs root access. To prepare that do the following:

```
sudo passwd
```

You will be prompted twice for the password of root. You will need is once in the following command:

```
su -
```

You will be prompted for the password of root after which you are root.

For convenience to have access to and from your server directly via ssh you copy the ssh keys from your workstation using:

```
rsync -av <your_username>@<your_workstation>:.ssh ./
```

Obviously the have to accept the key of your workstation and enter the password.

```
chown root:root -R .ssh
```

Leaving the server back to your workstation you can go to the server as root using:

```
ssh root@<IP_address>
```

Assign a fixed IP address to the server

This might be the time to give your system a fixed IP address, but you can accept the given address by DHCP as well. In that case it is best to assign in your DHCP server/router a fixed assignment of the MAC address of your server to the given IP address. Anyway it is assumed that from now on the IP address of the server is fixed.

Configuration with raspi-config

This may be the time to configure your system with raspi-config using the command:

```
rapspi-config
```

In *Interface Options* you may want to enable the Serial Port.

In *Localisation Options* you may want to change the Locale, the Timezone and the Keyboard.

In *Advanced Options* Expand Filesystem is not needed, but maybe you want predictable network i/f names.

After this you need to reboot, otherwise locale parameters give trouble.

Downloading the script

You may already have the script downloaded together with this manual, but the easiest way is to perform the following commands:

```
apt install git
```

```
git clone https://github.com/freekdk/GenPDSDM.git
```

This will load the script in the folder GenPDSDM.

When you inspect this folder you will see a number of files. Some of them are files used to generate a server like this on an openSUSE system. When you inspect the script, we will be using here, you will see that it also can be used on openSUSE Leap 15.5, 15.6 and Tumbleweed. Right at the beginning this script will see on what system it runs and will act accordingly; in this case on Rasberry Pi OS (bookworm).

Using previously generated files

You may already have the two files used by DKIM. This script assumes that their names start with the domain name of this server. The format of the name is: `<domain_name>.dkim<date>`, where date is `yyyymmdd`. The `.pem` file contains the private key used for signing header and body of an email message. The `.txtrecord` file contains the data that is needed in the TXT record in the DNS for this domain. It contains the public key.

A third file, `dh.pem`, might already be available (it is in Rasberry Pi OS), if not, it will be generated, which will take a long time (might be more than an hour on the Raspberry Pi).

Put at least the two files in the folder GenPDSDM; the scripts expects these there, and dh.pem if you try to use this script on another operating system.

Certificated signed by Let's Encrypt

You may already have certificates for `smtp.<your_domain_name>` and `imap.<your_domain_name>` or a certificate for `*.<your_domain_name>` with alias `<your_domain_name>` signed by Let's Encrypt. In that case these certificates are present in the folders `/etc/letsencrypt/live/smtp.<your_domain_name>/` and `/etc/letsencrypt/live/imap.<your_domaine_name>/` or `/etc/letsencrypt/live/<your_domain_name>/` with the names `fullchain.pem` and `privkey.pem`. The script searches for these folders and will ask you if you want the self signed certificates replaced by the ones from Let's Encrypt.

Further preliminary actions

The script expects that you already have the necessary data in the DNS. Obviously you have a DNS for your domain. In the DNS you must have an A record for your domain. You also need one of more MX records for this domain. The one with the highest priority pointing to smtp.<your_domain>. This name must have an A record or a CNAME to the A record of the domain. Also mail.<your_domain> and imap.<your_domain> as A or CNAME records are needed. In case the server is also an email server for another domain, that domain must have an MX record pointing to smtp.<your_domain>.

In case you want or have certificates for postfix and dovecot signed by Let's Encrypt these certificates will be or become present in the `/etc/letsencrypt/` folder. You need to have certificates with a Common Name (CN), which is for postfix smtp.<your_domain_name> and for dovecot imap.<your_domain_name>. Alternative you can have a CN *.<your_domain_name> and an alias <your_domain_name>. This means that you have either the folders `/etc/letsencrypt/smtp.<your_domain_name>` and `/etc/letsencrypt/imap.<your_domain_name>` or the folder `/etc/letsencrypt/<your_domain_name>`. The script searches for these folders and will use these certificates instead of the self-signed ones, although these will still be generated.

Executing the script

From now on you can start the script and it will install all necessary packages and ask you for the necessary information. When you start the script it will first report a file name which will contain a log about the execution of the script. On a second terminal session you can scan progress using:

```
tail -f <name_of_logfile>
```

Safe execution mode

When executing the script for the first time all files that get changed are saved in the folder `/var/adm/backup/genpdsdm/`. Parameters that are needed are all asked for during the initialization phase of the script and are stored in the file `/etc/genpdsdm/parameters`. They contain also passwords, so be carefull. After a successful execution of the script and inspection of the result, you may not be satisfied with the result. In that case you can execute the script with the parameter `--new` or `--old`, which respectively means that you start allover again, new parameters will be asked (`--new`), or you start using the saved parameters as defaults (`--old`), these parameters can be changed. In both cases the original files are first restored and will be configured by the new parameters.

Exceptions are the files /etc/aliases and /etc/postfix/canonical. This means that you can make changes in these files, which will be preserved when running the script again. The script will enter a few lines in these files, but when these lines are present they will be replaced if anything changed.

Executing the script a second time without a parameter is only meaningful when the script has been aborted. In that case parts that are successfully completed will be skipped.

During the first execution of the script the following systemd services are enabled and are activated:

- postfix.service
- dovecot.service
- firewalld.service
- clamav-freshclam.service
- clamav-daemon.service
- amavis.service
- opendmarc.service

These services are also active after a reboot. During a following execution of the script with one of the parameters on the command line, the configuration of theses services may have been changed, but in that case, when the reconfiguration is done, the services are restarted.

Some of the above services did not have changes in their configuration or the configuration will not change anymore. These services will remain active during the execution of the script with one of the parameters.

The only account on this headless system is the root account. The script will create a user account (administrator), to which email for root, postmaster, etc. will be directed. When you execute the script a second time with one of the parameters, and you create another user than the first time, the previous one will not be removed.

Two interfaces

Originally the script has been designed asking questions using the read command in bash. A second interface is available in which the command `dialog` is used, giving a more sophisticated interface. This interface will be used when the option --`dial[og]` is given on the command line eventually next to --`old` or –`new`. Another option is `--help`, to explain how to start the script and what the different option do; this parameter only prints the information and exits.

Questions asked

Below is an example of executing the script for the first time when all answers are correct and without using the script with dialog. Start the script with:

```
root@raspberrypi:~# GenPDSDM/genpdsdm.sh
```

You will see:

```
Log /var/log/genpdsdm-2024-05-06_153758.log started.
ATTENTION: the log file contains sensitive information (e.g. passwords)
Handle with care and sanitize before sharing.
Press Enter to continue:
```

So press Enter and the script will continue with the first action. It is running `apt-get -y update` and `apt-get -y upgrade` and installing the required packages. After quite some time the name of the server system will be asked for. In this document we use `rpi-server` . Also the domain name will be asked for and we use `example.com` here in the document, but you use your domain name.

```
=============================================
= Trying to find host name and domain name... =
=============================================
```

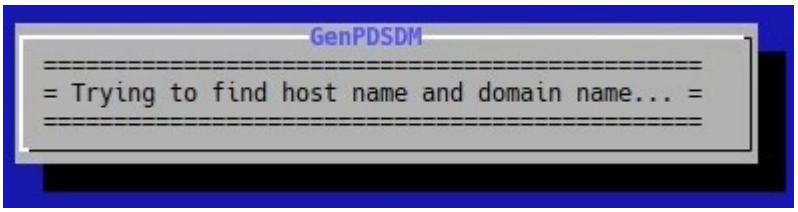

Figure 1: When using: --dialog

```
Questions about host name and domain name
The host name can be any name and consist of letters, digits, a "_"
and/or "-". This name should not be smtp or mail or imap, these names
will be used elsewhere in the server.
Enter the name of the system: rpi-server

An example of the domain name is: example.com; should at least contain
one dot. The script requires the existence of a DNS for this domain with
a MX record for the domain. The MX record should point to
smtp.<domain_name> or mail.<domain_name>; both should have an A record.
Also an imap.<domain_name> A record should exist, all with the same IP address.
Enter the domain name: example.com
```

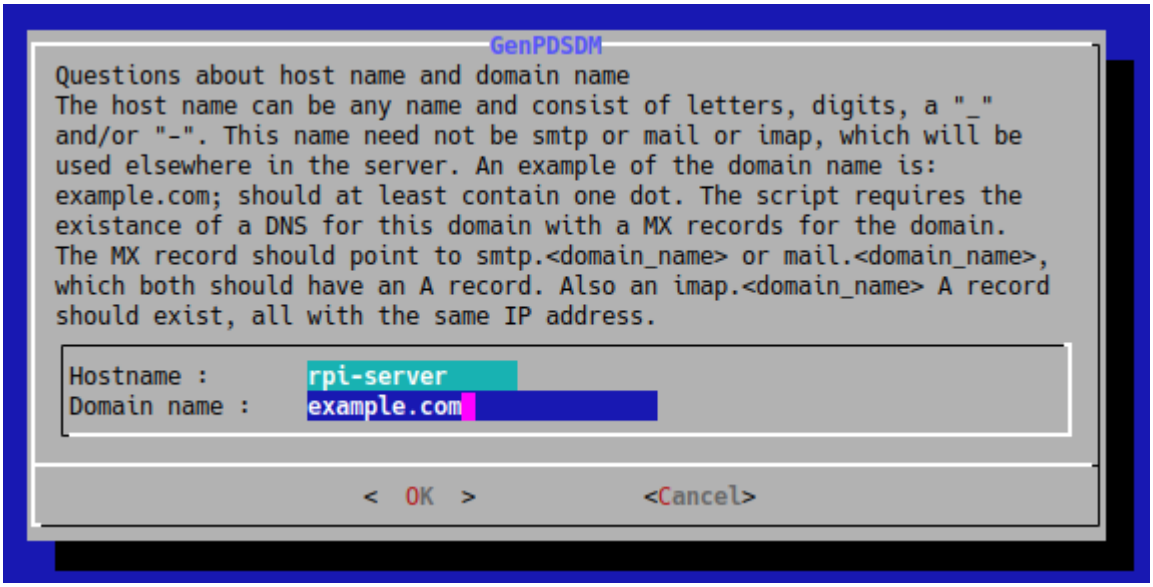

Figure 2: When using: --dialog

```
=============================================
= Checking for existing records in the DNS =
=============================================
```

Figure 3: When using --dialog

```
The domain name "example.com" will be used throughout this script
Is this OK?

Enter y or Y for OK, anything else is NO and the script will terminate : y
```

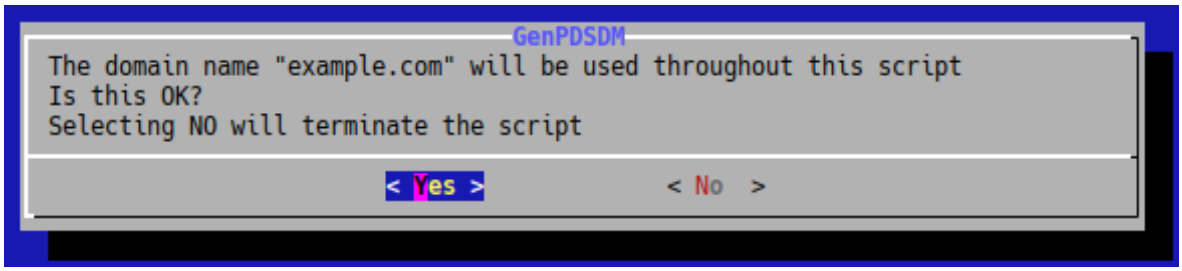

Figure 4: When using --dialog

The above is about establishing a host name and domain name. At first the host name is `raspberrypi` or the name you entered in the Imager, which is in the file `/etc/hostname` and there is no proper domain name. The script enters the entered host name in `/etc/hostname` and the domain name in `/etc/hosts` with the IP address of the interface. As is mentioned above the script will do some checking and looks for entries in the DNS for A records for the domain name, {smtp,mail,imap}<domain_name>. If these are not found the script will give an error message and asks to start the script again when this is fixed. The last question gives you the opportunity to start allover again with an empty host name and domain name.

Next the following will appear:

```
The command (hostname --fqdn) does NOT provide rpi4rasppdsdm.beelaertsict.nl.
This means the system needs to reboot to establish that.
Press Enter to reboot or Ctrl+C to abort :
```

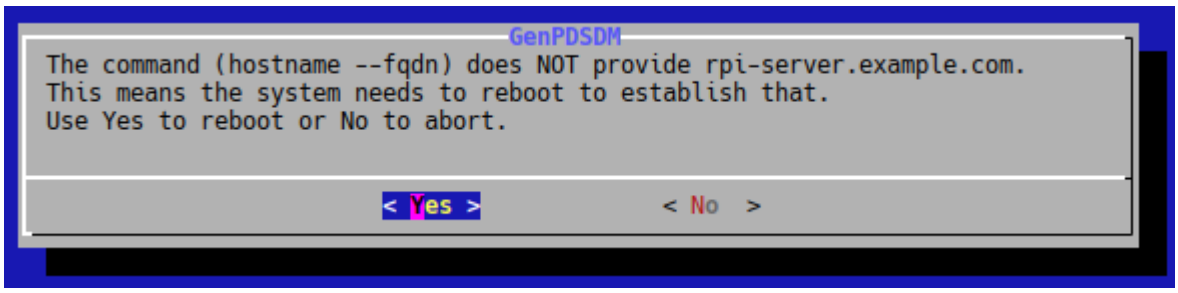

Figure 5: When using: --dialog

After reconnecting, using ssh `root@<IP-address>` , after the prompt the script needs to be restarted.

```
root@rpi-server:~# GenPDSDM/genpdsdm.sh
```

The following will appear:

```
Log /var/log/genpdsdm-2024-05-14_130637.log started.
ATTENTION: the log file contains sensitive information (e.g. passwords)
Handle with care and sanitize before sharing.
Press Enter to continue:
```

This is to get the name of the log file. So press Enter.

```
================================================
= Trying to find host name and domain name... =
================================================
```

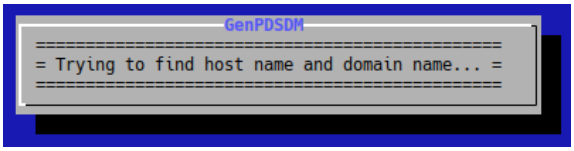

Figure 6: When using: --dialog

```
Found host name is : rpi-server
Domain name is     : example.com
```

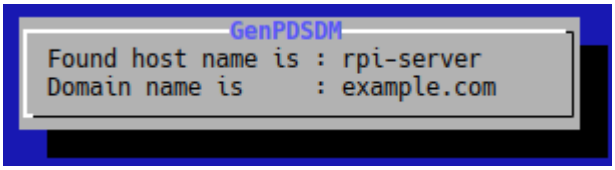

Figure 7: When using: --dialog

```
============================
Establishing needed parameters
============================
```

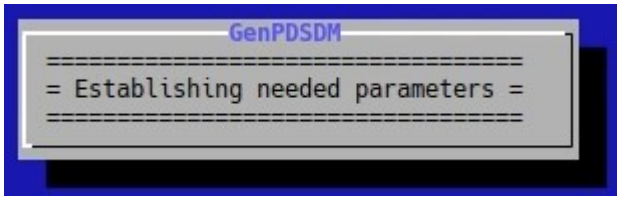

Figure 8: When using: --dialog

```
Do you want a limit on the message size? The standard limit is 10240000.
A recommended value is 30M, equivalent to 30000000.
A value less than the default will not be accepted.
Enter the limit on the message size : 30000000
```

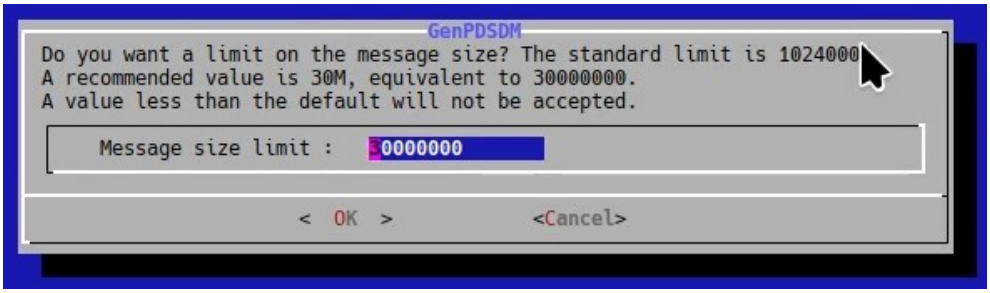

Figure 9: When using: --dialog

Enter the limit on the message size. Gmail accepts messages less than 25MB.

```
The original port for clients to submit messages to the email server via
an encrypted connection is 465. This one is now obsolete. But you may have
clients that still use this port. Messages submitted via this port will not
be DKIM signed. Do you want this port to be enabled?
Answer y or Y, anything else means No : n
```

No is chosen.

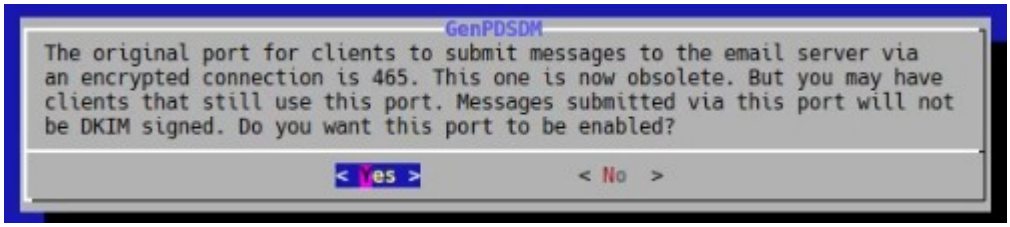

Figure 10: When using: --dialog

```
Questions about the relay host of your provider
We assume the relay host is accessible via port 587
(submission) and requires a user name and password.
An MX record for this name will not be used.

Please enter the name of the relayhost: mail.provider.com
```

```
Please enter your user name on the relay host, might be an e-mail address:
user@provider.com

Please enter the password of your account on the relay host: wert@yu!iop
```

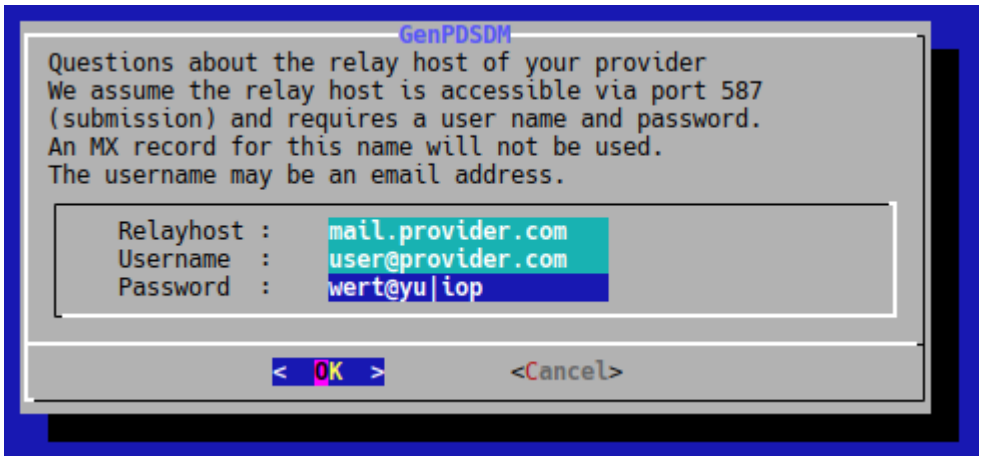

Figure 11: When using: --dialog

Again a check will be performed on the A record of the relay host. The user name and password will not be checked. The script keeps asking for the relay host until it finds the A record.

```
====================

Questions about username and name administrator.

The account name of the administrator to be created or
already present in this server. In case it is created, the
password for this account will be 'genpdsdm', but as root you
can easily change it.


Please enter the account name : john

Please enter the name of the administrator
for this account, like 'John P. Doe' : John P. Doe
```

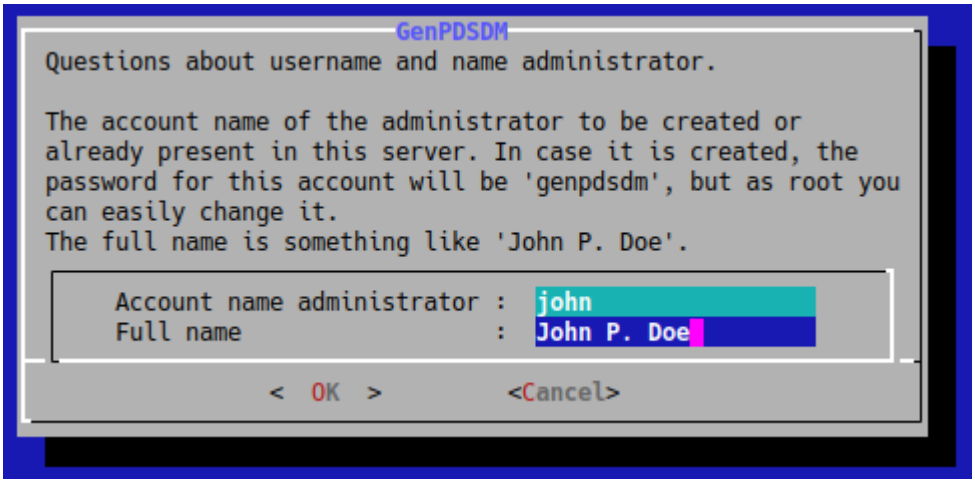

Figure 12: When using: --dialog

The following will appear if the account does not exists.

```
The user john has been created with password "genpdsdm".
```

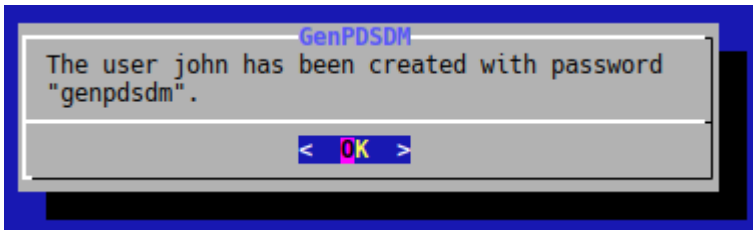

Figure 13: When using: --dialog

Otherwise…

```
The user "john" already exists. The name as comment
may have changed and will be replaced.
The password will remain the same as it is.
```

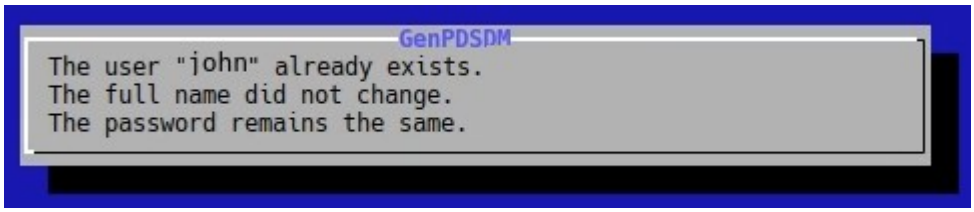

Figure 14: When using: --dialog

Next...

```
When sending an email as this user the sender address
will be "john@example.com"
You will have a canonical name like "john.p.doe@example.com".
Enter the part you want before the @ : john.p.doe
```

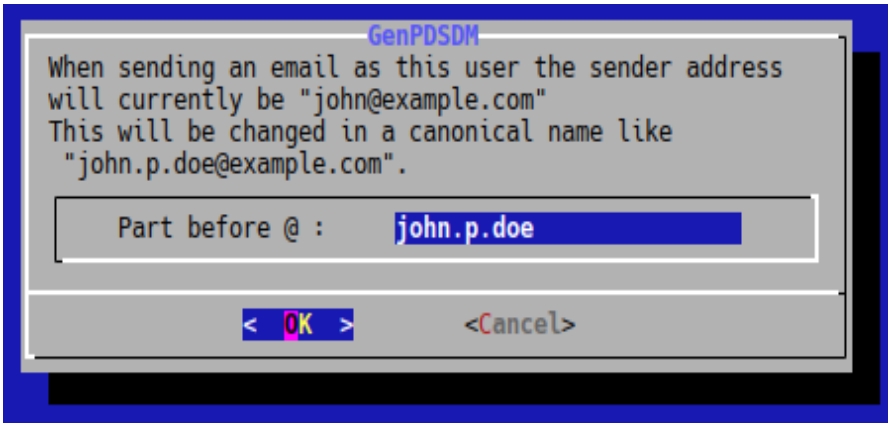

Figure 15: When using: --dialog

Sometimes a server like this does want to support email to one or more other domains. The following question gives you the possibility to enter one or more names of these domains. So if you want to receive, on this system, email with addresses like *user*@example.org, you can answer yes and add example.org. Note that *user* must be an account name on this system or have an entry in the file `/etc/aliases` .

```
Currently the server will consider email messages to the following domains as local:
smtp.example.com, example.com, localhost, localhost.example.com , rpi-server.example.com
```

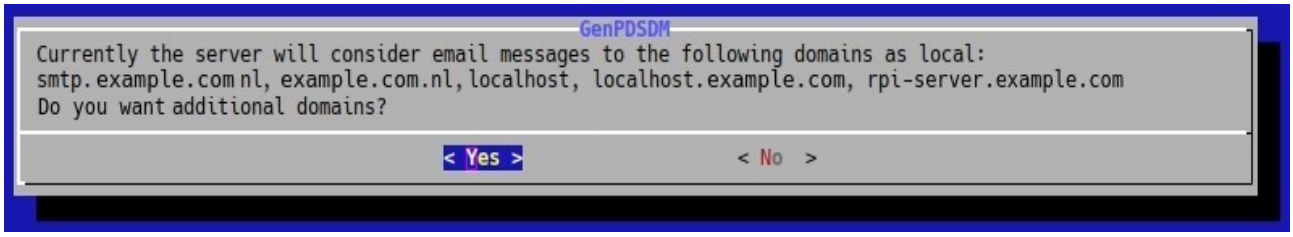

Figure 16: When using: --dialog

```
Do you want additional domains?
Y and y means OK, anything else means No : y

Enter the additional domain name; DNS entry will be checked.
Leave empty to finish asking.
Domain name: example.org
```

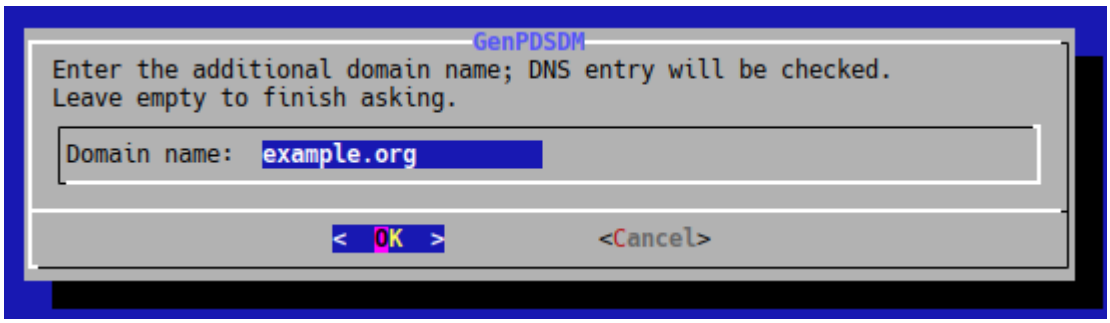

Figure 17: When using: --dialog

Simply press Enter to finish adding additional domains.

```
Enter the additional domain name; DNS entry will be checked.
Leave empty to finish asking.
```

```
Domain name:
```

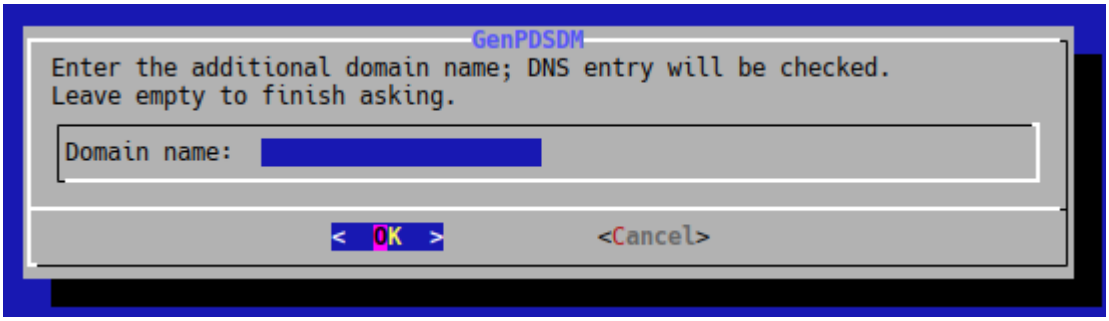

Figure 18: When using: --dialog

In order to protect your system from receiving spam, you can use so called black lists to refuse any connection to your e-mail server when the connecting IP address is listed on such a black list. Your server will, in that case, give a proper message to the sending server with a link to explain why the connection was refused. The script offers three servers to choose from.

```
Do you want any server with blacklisted hosts?
Answer y or Y for yes, no is anything else : y

Enter a combination of 1,2, and 3, which belong to the following options:
1: server bl.spamcop.net
2: server cbl.abuseat.org
3: server zen.spamhaus.org
: 123
```

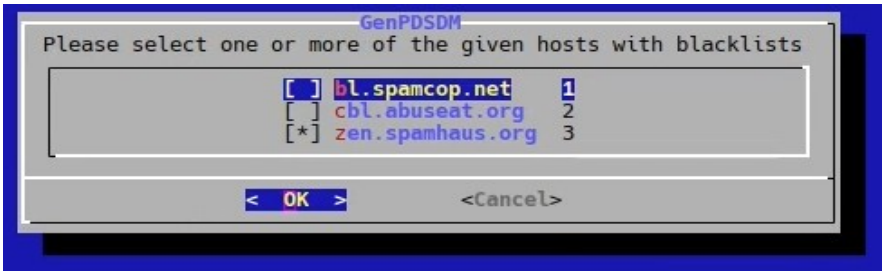

Figure 19: When using: --dialog

```
Questions about self signed certificates

In certificates usually parameters like Country, State, Locality/City, Organization
and Organizational Unit are present.
The script will use "Certificate Authority" as the Organizational Unit
for the signing certificate and "IMAP server" and "Email server"
respectively for Dovecot and Postfix certificates.
Common Names (CN) will be imap.example.com and smtp.example.com.

Enter the two character country code: XX
Enter the name of your STATE or PROVINCE: Some State
Enter the name of your LOCALITY/CITY: SomeCity
Enter the name of your ORGANIZATION: Some Organization Name
```

Figure 20: When using: --dialog

In case you or an other account on your system has more than one identity, different e-mail addresses with another domain name after @, like a gmail.com address, and you or another account wants to send messages with that address as sender address, it is quite possible that you need to send such a message via a server for that domain. In that case the following feature makes that possible.

```
Apart from sending all your messages with from addresses ...@example.com,
or all the other addresses, to the relay host mail.provider.com, you may want to
have additional email addresses in the from address, like some_user@gmail.com,
which you want to send to an additional relay host, i.e. smtp.gmail.com, based
on the from address. Do you want one or more of these additional relay hosts?

Answer y or Y, anything else will be No : y
```

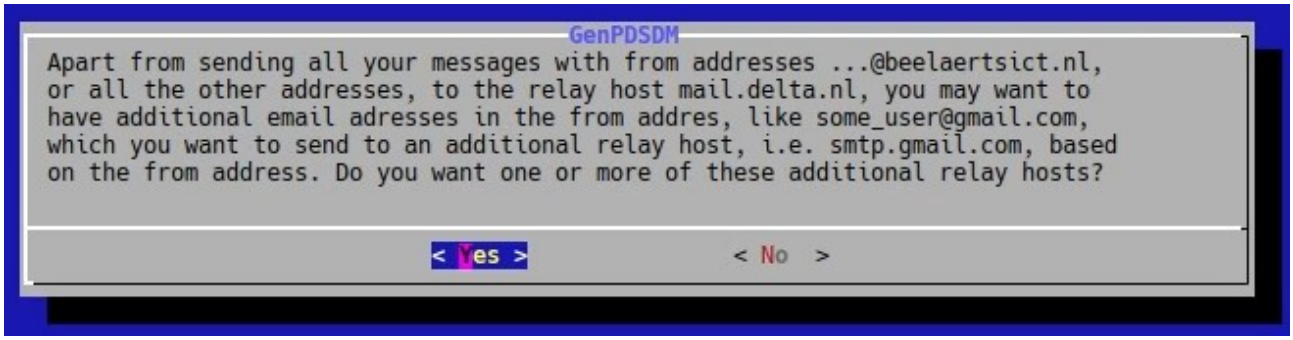

Figure 21: When using: --dialog

```
Email address to be send to additional relay host,
press Enter to end asking for these email addresses :  some_user@gmail.com

Enter the relay host, like smtp.gmail.com, the connection will be made
to the IP address in the A record of this name in the DNS. A check
will be performed on the existence of such a record.
Additional relay host : smtp.gmail.com

Enter the port for access to the relay host; choices are 465 or 587.
```

```
Port : 587

Enter the username for access to the relay host; might be the same as
the email address entered before. If so enter = , else the username
Username : =
Enter the password for access to the relay host.
Password : somepassw
```

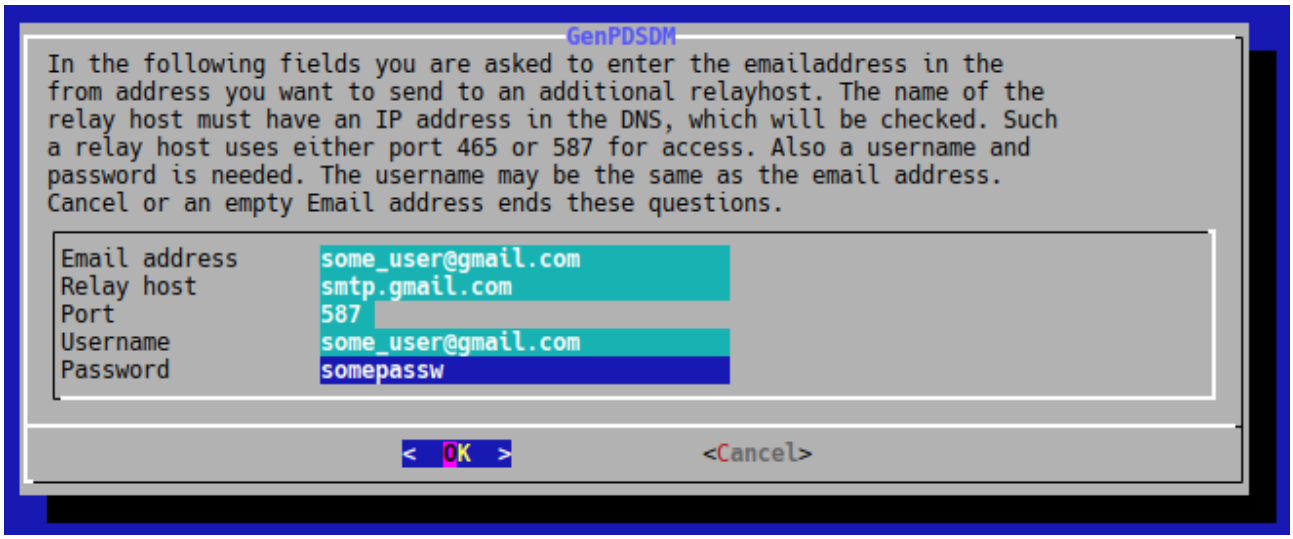

Figure 22: When using: --dialog

```
Email address to be send to relay host,
press Enter to end asking for these email addresses :
```

Press Enter to finish asking for this data.

When port 465 is needed special entries will be present in the files `/etc/postfix/master.cf` and `/etc/postfix/sender_dependent_default_transport` .

When you are using Let's Encrypt to sign your certificates for postfix and dovecot and you use the recommended method, you will have these certificates in a folder `/etc/letsencrypt/` . In that case the script will ask:

```
You do have certificates signed by Let's Encrypt,
do you want to use these certificates for postfix and dovecot?
Answer y or Y, or anything else if not : y
```

Figure 23: When using: --dialog

This is the last question asked. After this the script will show you what it is doing.

```
============================
= Configuring firewalld... =
============================
```

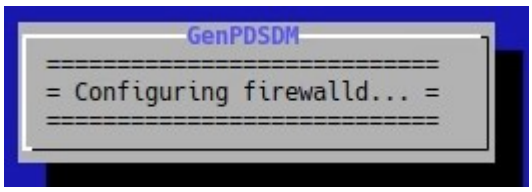

Figure 24: When using: --dialog

```
====================================
= Configuring /etc/postfix/main.cf =
= and referenced files...          =
====================================
```

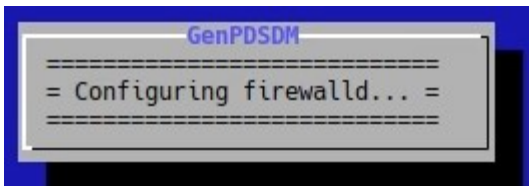

Figure 25 When using: --dialog

```
====================================
= Configuring /etc/postfix/master.cf =
= and referenced files...            =
====================================
```

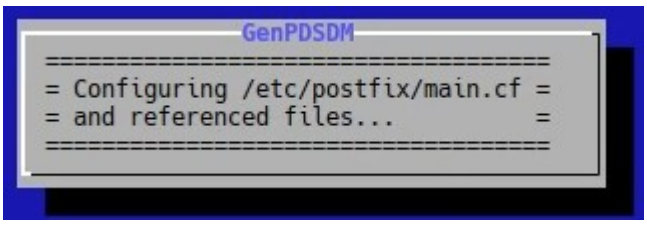

Figure 26: When using: --dialog

```
====================================
= Generating Certificates for CA... =
====================================
```

Figure 27: When using: --dialog

```
========================================
= Generating Certificates for postfix... =
========================================
```

Figure 28: When using: --dialog

```
========================================
= Generating Certificates for dovecot... =
========================================
```

Figure 29: When using: --dialog

```
==========================
= Configuring dovecot... =
==========================
```

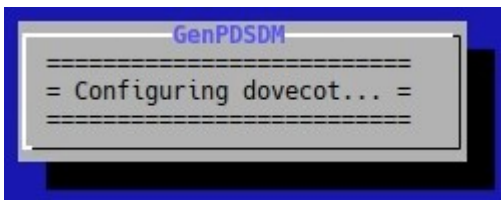

Figure 30: When using: --dialog

```
==================================
= Starting freshclam and clamd... =
==================================
```

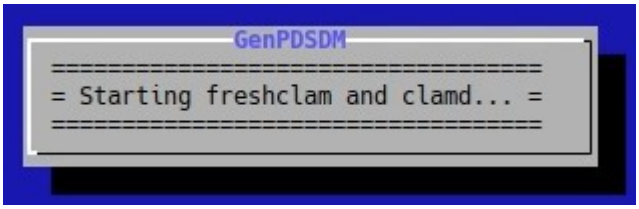

Figure 31: When using: --dialog

```
=========================
= Configuring amavis... =
=========================
```

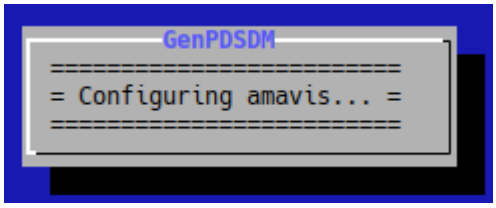

Figure 32: When using: --dialog

Part of configuring amavis is copying or generating the DKIM private and public keys. The location of the public key to be entered (maybe already present in the DNS) wil be shown.

```
The DKIM public key to be entered in the DNS is present in the file
/var/db/dkim/example.com.dkim20231111.txtrecord
```

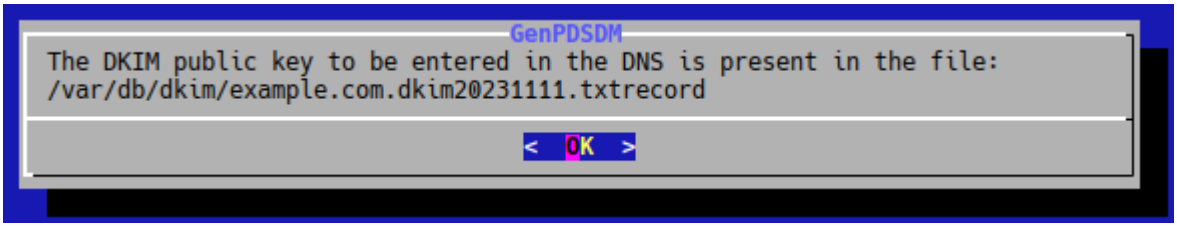

Figure 33: When using: --dialog

```
=========================
= Configuring DMARC...  =
=========================
```

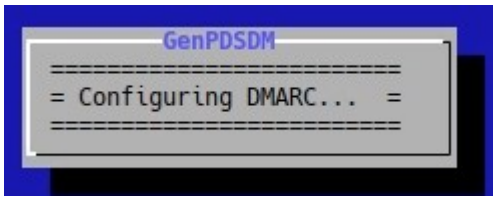

Figure 34: When using: --dialog

In case certificates signed by Let's Encrypt are present, these will replace the self-signed ones.

```
====================================
= Replacing self-signed certificates =
= by those signed by Let's Encrypt   =
====================================
```

Figure 35: When using: --dialog

```
=====================================================
= Restarting postfix, dovecot, amavis and opendmarc =
=====================================================
```

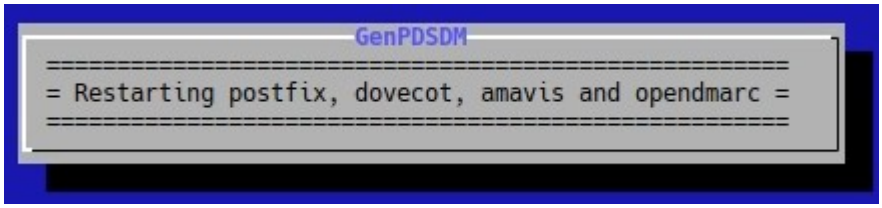

Figure 36: When using: --dialog

```
root@rpi-server:~#
```

Here the script will end and you have a configured email and imap server.

SPF TXT record explained

The script does **not** check on the presence of a SPF TXT record in the DNS of your domain.

In case you want to protect others from using email addresses of your domain, you are depending on mailers in the rest of the world. These mailers need to use SPF validation and you only need to have a SPF record in the DNS of your domain. Remember this check is done using the domain part of the address in the MAIL FROM and the IP-address that makes the connection. Such a record is a TXT record which contains the following:

```
@ IN TXT "v=spf1 mx a ip4:192.0.7.25 ip6:2001:1234:4567::32 a:example.com ~all"
```

- @ means it is a record in your top domain: example.com
- v=spf1 indicates the TXT record is a SPF record
- mx means that the IP address associated with your MX server is allowed to send email with your domain addresses
- a means that IP-addresses in your A and AAAA record are allowed to send email with your email addresses
- ip4:192.0.7.25 means that a system with this IP address is allowed ...
- ipv6:2001:1234:4567::32 means that this IPv6 address is allowed ...
- a:example.com means that IP-addresses mentioned in A and AAAA records of example.com are allowed ...
- ~all should always be the last element in your record, where ~ means: when there is no IP address allowed it is a soft failure, which adds up to the spam count. When this character is -, it means deny access.

When MAIL FROM contains an empty address, in case of a.o. a non-delivery message, there is no domain to check for an SPF record. In that case the receiving server will use the name in the HELO/EHLO command. So if your server uses smtp.example.com as the name in this command, you should also have the following record in your DNS:

```
smtp IN TXT "v=spf1 a -all"
```

Obviously you need an A and/or AAAA record for smtp.example.com and any server trying to mimic your server using this name in the HELO/EHLO command will be denied access.

More information about SPF records can be found in RFC7208.

DKIM explained

DKIM, short for Domain Keys Identified Mail, gives the body and header of an outgoing email a digital signature. The public key is published via DNS, so a receiving server can verify the digital signature of incoming messages.

The package amavisd-new contains the necessary support for both validating incoming messages and providing the signatures for header and body of outgoing messages. There is also the package *opendkim*, which is meant for large servers, but this is left out of this script.

Note that there might not be a relation between the domain of the sender in a message and the domain which provides the signature. It is up to the receiving server what to do with the result of a positive result. The hope is that this server is trusting the message more than non-signed messages.

The private key used for the signature is in the folder `/var/db/dkim/` and so is the public key to be entered in the DNS.

Maintenance of DKIM keys

It is good practice to renew DKIM keys occasionally. Depends on how many times you did send a signed message. The best way is to use a new signing key like the one in the above mentioned folder. The name can be any name that can be used in the DNS. After having implemented a new signing key (see below) the previous one can be removed from the DNS after the ttl time (10 days in the example). To generate a new key the following script can be used:

```
#!/bin/bash
date=$(date --date=now +%Y%m%d)
amavisd -c /etc/amavisd.conf genrsa /etc/amavisd/example.com.dkim${date}.pem 2048
```

After this you need to change the reference to the original file in a reference to this new file at the end of `/etc/amavis.conf` . After that you have to generate the public key:

```
amavisd -c /etc/amavisd.conf showkeys > /etc/amavisd/example.com.dkim${date}.txtrecord
```

Use this to add the new TXT record in the DNS.

DMARC explained

DMARC, short for Domain based Message Authentication, Reporting and Conformance, is an addition to the other 2 security standards for email, SPF and DKIM. DMARC gives receiving servers an indication on how to process incoming email messages that do not have signatures that provide valid SPF and DKIM results. These can be discarded or quarantined.

DMARC can check if the sender domain in the ‘From’ header matches with the sender domain in the envelope (SPF) and with the signing domain in the DKIM header. This means that DMARC forces a relation between the validated SPF and DKIM sender domains and the sender domain in the 'From'. This means that a message that gives a positive result on the validation of SPF and DKIM, still can be rejected by DMARC.

The DMARC policy is published via a record in the DNS. This record can also contain email addresses which can be used by mail systems to report accepted/refused messages. This way the manager of the mail domain will get some insight in the delivery of both real and falsified messages.

The DMARC TXT record in the DNS

To enable DMARC you need a DMARC TXT record in your DNS. An example of such a record is:

```
_dmarc    IN    TXT    "v=DMARC1; p=none; sp=none; adkim=s; aspf=s;
rua=mailto:dmarc-reports@example.com; ruf=mailto:dmarc-reports@example.com
fo=1;"
```

The meaning of the elements in this record is as follows:

- v=DMARC1 indicates together with the name of the record _dmarc that the TXT record is a DMARC record
- p=none indicates the policy for the domain
- sp=none indicates the policy for subdomains, when these subdomains do not have a DMARC TXT record
- adkim=s indicated the strength with which the receiving server should test the alignment between the domain name in the FROM address and the DKIM key inserted in the message
- aspf=s indicates the strength of with which the receiving server should test the alignment between the sending IP address and the allowed IP addresses in the SPF record
- rua=mailto:dmarc-reports@example.com indicates the email address where aggregated reports should be send to, by default once a day (another parameter, ri, may change this value, parameter pct, default 100, indicates that only a percentage of the reports need to be send)
- ruf=mailto:dmarc-reports@example.com indicates the email address where a reports should be send to in case the server does not accept the message
- fo=1 indicates that reports about not accepted messages should be send (see rfc7489 section 6.3 for additional values)

The possible policies (p= or sp=) are *none*, *quarantine* and *reject*. One should start with *none* which means that tests at the receiving end should be performed and reports should be send, but the message should be accepted. It is meant for situations where there are several systems being able to send message with your domain name in the FROM address and you expect that your SPF record may not be correct or not all sending servers have implemented the DKIM signing.

The strength in testing in *aspf* and *adkim* can be *s* for strict and *r* for relaxed. With adkim=s the domain part of the from address in the header **must** match the domain name in the DKIM signature. For adkim=r the domain part of from address in the header is allowed to be a subdomain. For aspf=s

the domain part of from address **must** exactly match the domain part in the smtp protocol command MAIL FROM, also called the envelop sender. For aspf=r this match is allowed to be partly.

Script to test the generation of a DKIM element in the header

The file `testdkim.sh` can be used to test the setup of your server. It asks for the domain name of your server, the user name of a user on your system and its password. After that a very small email message will be send via the port submission (587). This should enable the insertion of the DKIM item in the header of the message. The message is send to root, which means that it will be redirected to the user where messages for root are directed. In the folder `~/Maildir/new/` of that user you will find the message and you can inspect it for the presence of the DKIM item in the header.

This script uses the utility **expect**.