# GenPDSDM manual

**Gen**erate a system with **P**ostfix and **D**ovecot, with all the available security features, like virus scanning, **S**PF, **D**KIM and D**M**ARC



## Introduction

This manual is about a bash script which is a follow up on the wiki page https://en.opensuse.org/Mail_server_HOWTO. This page describes a procedure to configure an openSUSE system with Leap 15.5 on a Raspberry Pi 4B to act as an email server for a specific domain. The domain name used in this page 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 Kmail or Thunderbird, where these roles are named as 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

The script assumes that you start with a fresh installed system. As an example, we use a Raspberry Pi 4B image of Leap 15.5, meant for a headless system. Using the following commands on another Linux system will download and check the image.

```
repository="http://download.opensuse.org/distribution/leap/15.5/appliances"
image="openSUSE-Leap-15.5-ARM-JeOS-raspberrypi.aarch64.raw.xz"
mkdir -p image
cd image
curl -OL $repository/${image}.sha256
```

```
curl -OL $repository/$image
cat ${image}.sha256 | sha256sum -c -
```

After this you insert the boot device for the Raspberry Pi in a USB slot or SD slot of this computer. This device can be a microSD card in a SD envelop or a USB disk.

Now you have to find on what device the boot device is present. You can use the following command to find out. You can give this command before inserting the boot device and after inserting and deduce from the difference the device.

```
ls -l /dev/sd*
```

Most likely it is the last device in the list. In my case is was /dev/sdd ; not the one with a digit at the end. Now you give the following commands, the first ones are only necessary when your device has been used earlier for other purposes. You will be asked for the root password. ***Obviously you need to replace sdd in the following command lines by your device name.***

```
ssize=$(sudo /usr/sbin/blockdev --getss /dev/sdd)
echo Sector size = $ssize
```

The sector size needs to be 512, otherwise you need to investigate what the next commands need to be in that case. Next we clear the last blocks of the device and write the image on the boot device.

```
size=$(sudo /usr/sbin/blockdev --getsz /dev/sdd)
sudo dd if=/dev/zero of=/dev/sdd obs=1 seek=$(($size - 2)) count=2
sudo xzcat -v $image | sudo dd bs=4M of=/dev/sdd iflag=fullblock oflag=direct
sudo sync
```

The next step is to insert the microSD card or the USB device in the proper slot of the Raspberry Pi and power the device on. The system is meant to be a server, so you better connect the system via the Ethernet interface with your network.

## 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 root@<IP address>
```

This will ask you to accept the key from this system and ask you for the username, *root*, and the password, *linux* . The first thing to do is to change the password of root with the command `passwd`.

## 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.

## Downloading the script

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

```
zypper in git
```

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

This will load the script in the folder GenPDSDM.

From now on you can start the script and it will ask you for the necessary information. At a certain point it needs to reboot the system, so you need to login again and you just need to start the script again. It will continue where it left of. Starting the script is done by:

```
GenPDSDM/genpdsdm.sh
```

# Executing the script

## Safe execution mode

When executing the script for the first time all changed files are saved in the folder "/etc/genpdsdm/". Parameters that are needed are all asked for during the initialization phase of the script and are stored in that folder in the file "genpdsdm.history". 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, or you start using the saved parameters as defaults, which can be changed. In both cases the original files are first restored and will be configured by the new parameters. 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
- firewlld.service
- freshclam.service
- clamd.service
- amavis.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 service may be 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, 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.

## Questions asked

Below is an example of executing the script for the first time when all answers are correct. Start the script with:

```
localhost:~# GenPDSDM/genpdsdm.sh
```

The first action of the script is running `zypper up` and installing the required packages. You have to answer yes to the questions zypper asks. Finally you get:

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

Questions about host name and domain name

The host name can be any name and consist of letters, digits, a "_" and/or "-"
This name need not be smtp or mail or imap, which will be used elsewhere in the server
Enter the name of the system: somename
An example of the domain name is: example.com; should at least contain one dot
The script requires the existance of a DNS for this domain with A MX records for the domain
The MX record should point to smtp.<domain_name> or mail.<domain_name>, which 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
Checking for existing records in the DNS
The system will reboot now and please run this script again
```

The above is about establishing a host name and domain name. At first the host name is `localhost` and there is no proper domain name. The script enters the entered host name in `/etc/hostname` and the domain name in `/etc/hosts`. To make this live in the system a reboot is necessary, but is also necessary when updates on the software have been done.

When the reboot finishes, you can login again and will be greeted by another prompt, after which you start the script again. That line will look like:

```
somehost:~ # GenPDSDM/genpdsdm.sh
```

The following output will be seen:

```
Trying to find host name and domain name...
The domain name "example.com" will be used throughout this script
Is this OK enter y, Y or nothing and press Enter:
==============================
Establishing needed parameters
==============================

Questions about the relay host of your provider

We assume the relay host is accessable via port 587 (submission) and
requires a user name and password
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: wertyuiop
Please enter the account name to be created in this server: john
The password for this account will be 'genpdsdm', but as root you can easily change it
Please enter your name to be used with this account, like 'John P. Doe': John Doe
```

```
When sending an email as this user the sender address will be "john@domain.com"
You may want to have a canonical sender name like "John.P.Doe@beelaertsict.nl"
Enter the part you want before the @ : john.doe

Questions about self signed certificates

In certificates usually parameters like Country, State, Locality/City, Organization
and Organizational Unit are present. These are not really necessary, but at least a
two character country code is required
The script will use it own names for Organizational Unit
Enter your two character country code: XX
Enter the name of your STATE or PROVINCE
Enter means leave empty, anything else means the name : Some-State
Enter the name of your LOCALITY/CITY
Enter means leave empty, anything else means the name : SomeCity
Enter the name of your ORGANIZATION
Enter means leave empty, anything else means the name : Some Organization Name

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

==========================
Configurating firewalld...
==========================
Created symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service →
/usr/lib/systemd/system/firewalld.service.
Created symlink /etc/systemd/system/multi-user.target.wants/firewalld.service →
/usr/lib/systemd/system/firewalld.service.
success
success
success
success
success
success
==================================
Configuring /etc/postfix/main.cf...
==================================
====================================
Configuring /etc/postfix/master.cf...
====================================
====================================
Generating Certificates for postfix...
====================================
Generating a RSA private key
```

After this you will be asked, four times, for the same password for generating the self signed certificates for both Dovecot and Postfix. The password can be anything. The chosen lifetime of the certificates is 10 years, so you may forget this password unless you want to use it elsewhere.

```
======================
Configuring dovecot...
======================
====================================
Generating certificate for dovecot...
====================================
Generating a RSA private key
.............+++++
.....................+++++
writing new private key to '/etc/ssl/private/dovecot.pem'
-----

subject=C = XX, ST = SomeState, L = SomeCity, O = SomeOrganzation, OU = IMAP server, CN =
imap.domain.com, emailAddress = postmaster@domain.cm
SHA1 Fingerprint=C5:F3:19:7E:7E:76:2B:9C:47:82:C2:5A:4F:53:6F:B5:51:27:89:7E
```

The first time you run the script something random in `/etc/dovecot/dh.pem` will be generated for Dovecot, which will take quite some time. Om my Raspberry Pi 4B it took once **160 minutes,** other times less.

```
==============================
Starting freshclam and clamd...
==============================
====================
Configuring amavis...
====================

The DKIM public key to be entered in the DNS is present in the file
/etc/amavisd/domain.com.dkim20230630.txtrecord

====================
Configuring DMARC...
====================
====================================
Restarting postfix, dovecot and amavis
====================================
```

## 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 DomainKeys 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 `/etc/amavisd/` 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=$(date --date=now +%Y%m%d)
amavisd -c /etc/amavisd.conf genrsa /etc/amavisd/domain.com.dkim${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/$domain.com.dkim${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 as 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 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.