

Postfix

- [Configuration](#)
- [Logging](#)
 - [Searching Postfix logs](#)
- [OpenDKIM](#)

Configuration

Logging

Searching Postfix logs

Status messages

```
# find all log entries where the status is other than sent
grep status= /var/log/mail.log | grep -v status=sent

# make sure there aren't any other status messages
grep status= /var/log/mail.log | grep -v -e status=sent -e status=bounced -e status=deferred -
e status=expired

# find individual instances
grep status=bounced /var/log/mail.log
grep status=deferred /var/log/mail.log
grep status=expired /var/log/mail.log
```

#end

OpenDKIM

```
sudo apt install opendkim
```

```
mkdir -p /etc/opendkim/keys
cd /etc/opendkim
touch key.table signing.table trusted.hosts

sudo chown -R opendkim:adm /etc/opendkim
sudo find /etc/opendkim -type d -exec chmod 770 {} \;
sudo find /etc/opendkim -type f -exec chmod 660 {} \;

# socket and permissions
sudo mkdir /var/spool/postfix/opendkim
sudo chown opendkim:postfix /var/spool/postfix/opendkim
sudo chmod 755 /var/spool/postfix/opendkim
```

Example configuration files

The example below shows a configuration for domain.com and domain.net.

Only one selector is in production at a time. This facilitates easy key rotation.

FILE: key.table

```
selector1._domainkey.domain.com
domain.com:selector1:/etc/opendkim/keys/domain.com/selector1.private
selector1._domainkey.domain.net
domain.net:selector1:/etc/opendkim/keys/domain.net/selector1.private
```

```
selector2._domainkey.domain.com
domain.com:selector2:/etc/opendkim/keys/domain.com/selector2.private
selector2._domainkey.domain.net
domain.net:selector2:/etc/opendkim/keys/domain.net/selector2.private
```

FILE: signing.table

```
*@domain.com selector1._domainkey.domain.com
*@*.domain.com selector1._domainkey.domain.com
*@domain.net selector1._domainkey.domain.net
*@*.domain.net selector1._domainkey.domain.net
```

```
*@domain.com selector2._domainkey.domain.com
*@*.domain.com selector2._domainkey.domain.com
*@domain.net selector2._domainkey.domain.net
*@*.domain.net selector2._domainkey.domain.net
```

FILE: trusted.hosts

```
127.0.0.1
localhost

.domain.com
.domain.net
```

FILE: /etc/postfix/main.cf

Add the following to your postfix configuration:

```
# DKIM support - Militer configuration
milter_default_action = accept
milter_protocol = 6
smtpd_milters = local:opendkim/opendkim.sock
non_smtpd_milters = $smtpd_milters
```

FILE: /etc/opendkim.conf

```
# This is a basic configuration for signing and verifying. It can easily be
# adapted to suit a basic installation. See opendkim.conf(5) and
# /usr/share/doc/opendkim/examples/opendkim.conf.sample for complete
# documentation of available configuration parameters.

Syslog yes
SyslogSuccess yes
```

LogWhy yes

Common signing and verification parameters. In Debian, the "From" header is
oversigned, because it is often the identity key used by reputation systems
and thus somewhat security sensitive.

Canonicalization relaxed/simple

Mode sv

SubDomains no

OversignHeaders From

AutoRestart yes

AutoRestartRate 10/1M

Background yes

DNSTimeout 5

SignatureAlgorithm rsa-sha256

In Debian, opendkim runs as user "opendkim". A umask of 007 is required when
using a local socket with MTAs that access the socket as a non-privileged
user (for example, Postfix). You may need to add user "postfix" to group
"opendkim" in that case.

UserID opendkim

UMask 007

Socket for the MTA connection (required). If the MTA is inside a chroot jail,
it must be ensured that the socket is accessible. In Debian, Postfix runs in
a chroot in /var/spool/postfix, therefore a Unix socket would have to be
configured as shown on the last line below.

#Socket local:/run/opendkim/opendkim.sock

#Socket inet:8891@localhost

#Socket inet:8891

Socket local:/var/spool/postfix/opendkim/opendkim.sock

PidFile /run/opendkim/opendkim.pid

Hosts for which to sign rather than verify, default is 127.0.0.1. See the
OPERATION section of opendkim(8) for more information.

#InternalHosts 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12

The trust anchor enables DNSSEC. In Debian, the trust anchor file is provided
by the package dns-root-data.

TrustAnchorFile /usr/share/dns/root.key

```
#Nameservers          127.0.0.1

# Map domains in From addresses to keys used to sign messages
KeyTable              refile:/etc/openssl/key.table
SigningTable          refile:/etc/openssl/signing.table

# If you ever want to use SQLite3 this is what a dsn would look like
# This being untested, questions remain about the number of slashes needed
# OpenDKIM uses OpenDBX to access alternate database styles
#
# SigningTable
dsn:sqlite3:///etc/mail/config.sqlite/table=dkim_keys?keycol=domain?datacol=domain
# KeyTable
dsn:sqlite3:///etc/mail/config.sqlite/table=dkim_keys?keycol=domain?datacol=domain,selector,
private_key

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

# A set of internal hosts whose mail should be signed
InternalHosts         /etc/openssl/trusted.hosts

# 20250512
RequireSafeKeys False
```