

LinuxDays Luxembourg Tutorial Script

Secure Linux Communication Server

Christian Mock, based on work by Eric Dondelinger

Luxembourg 1 February 2007

Abstract

This tutorial consists in configuring a mail server based on Postfix. In the first, more theoretical part, the architecture and the different configuration files of the Postfix system will be explained. During the practical part, the attendees will setup a mail server that handles different domains. Then, the server will be extended in a way to be able to scan for viruses. In the second part of the tutorial, we will go into details how to protect your mail server and mails from spam.

Contents

1	Introduction	1
2	Postfix	2
2.1	About this tutorial	3
2.2	Introduction to Postfix	3
2.2.1	What is Postfix?	3
2.2.2	Where to get Postfix?	4
2.2.3	Why Postfix	5
2.3	Architecture of the Postfix system	5
2.4	Installing Postfix	8
2.4.1	Installation	8
2.4.2	Where are the files?	9
2.5	Basic configuration	11
2.5.1	Postfix configuration files	12
2.5.2	My own hostname	14
2.5.3	My own domain name	14
2.5.4	My own network addresses	15
2.5.5	What domain name to use in outbound mail	15
2.5.6	What domains to receive mail for	16
2.5.7	What clients to relay mail from	17

Contents

2.5.8	What destinations to relay mail to	18
2.5.9	What delivery method: direct or indirect	19
2.5.10	What trouble to report to the postmaster	20
2.5.11	What you need to know about Postfix logging	21
2.5.12	Running Postfix daemon processes chrooted	21
2.6	The basic configuration files for our example	22
2.7	Additional configuration	23
2.7.1	The <code>virtual</code> table	23
2.7.2	The <code>aliases</code> table	25
3	Virus scanner	27
3.1	AMaViS - A Mail Virus Scanner	27
3.1.1	Configuring Postfix to use Amavis-new	28
3.1.2	Configuring amavisd-new	29
3.1.3	Which Virus-scanner to use?	30
3.1.4	ClamAV	30
3.1.5	Installing and configuring ClamAV	31
3.1.6	Integrate ClamAV with AMaViS	33
4	Anti-Spam Configuration	35
4.1	Short intro on spam	35
4.1.1	Risk assessment	35
4.1.2	Where does spam come from?	36
4.1.3	What characteristics can we use to detect spam?	36
4.2	Blacklists/RBLs/DNSBLs	37
4.2.1	Using DNSBLs in Postfix	38
4.3	SpamAssassin	43
4.3.1	How does it work?	43

Contents

4.3.2	Installation	44
4.3.3	Configuration	44
4.3.4	Feeding the Bayes DB	47
4.3.5	Testing SpamAssassin	49
4.4	Configuring Amavis for SpamAssassin	51
4.4.1	Testing	53
4.5	Advanced issues	53
4.5.1	Whitelisting/Blacklisting in SpamAssassin	54
4.5.2	Performance Issues	54
4.5.3	Getting Feedback to SpamAssassin	55

List of Figures

2.1	Postfix Architecture	5
3.1	Postfix in combination with AMaViS	28

1 Introduction

This tutorial is based on 2005's LinuxDays' Server Tutorial's chapter on a mail server, and is only slightly modified to reflect changes in the LinuxDays' network environment set up for 2006. The DNS for instance is pre-configured.

The script supposes that attendees already know at least some theory on how email works, and does not go into details of the SMTP protocol unless needed for the understanding of the present setup of a mail server.

2 Postfix

This tutorial will show how to setup a Postfix based mail server to use in a small enterprise network. To clarify the system, we will exemplify the mail server in a so called virtual enterprise “Tux-Industries”.

Tux-Industries is a company with different offices located in different countries. The top-level domain name is tux-industries.com, and each country has a sub-domain called <country>.tux-industries.com. The setup of the different domains and related MX-records were shown in 2005’s server tutorial’s chapter on DNS and BIND and are not treated here.

Imaging now that you are responsible of the Tux-Industries office located in Luxembourg. So you have set up your DNS for the sub-domain luxembourg.tux-industries.lu. The next step will be to install and setup the mail server for your office. Additionally you want to handle two other sub-domains: sales.luxembourg.tux-industries.lu and support.luxembourg.tux-industries.lu. The employees’ mail-addresses have the following format: `firstname.name@luxembourg.tux-industries.lu`.

To achieve a higher security level, your incoming mails should be scanned by a virus scanner. Furthermore Spam mail should be blocked. This anti-spam part will be handled in the next chapter.

2.1 About this tutorial

This Postfix introduction is mainly a collection of existing documents. There are many parts copied from the official Postfix documentation. The original documentation can be found at <http://www.postfix.org/documentation.html>. It was tried to make a selection of interesting parts out of this very complete documentation, so that to setup a basic Mail Transport Agent (MTA) you can start reading this synthesis.

2.2 Introduction to Postfix

2.2.1 What is Postfix?

The goal of the Postfix project is to implement a viable alternative to the UNIX Sendmail program. Specific goals, and the ways that Postfix attempts to achieve them are:

- ▶ Wide dissemination. Postfix must be adopted by lots of people in order to make a significant impact on Internet mail performance and security. Therefore the software is given away for free, with no strings attached to it.
- ▶ Performance. Postfix is up to three times as fast as its nearest competitor. A desktop PC running Postfix can receive and deliver a million different messages per day. Postfix uses web server tricks to reduce process creation overhead and uses other tricks to reduce file system overhead, without compromising reliability.
- ▶ Compatibility. Postfix is designed to be sendmail-compatible to make migration easy. Postfix supports `/var[/spool]/mail`, `/etc/aliases`, NIS, and `~/.forward` files. However, Postfix also attempts to be easy to administer, and therefore it does not use `sendmail.cf`.

- ▶ **Safety and robustness.** Postfix is designed to behave rationally under stress. When the local system runs out of disk space or memory, the Postfix software backs off, instead of making the problem worse. By design, no Postfix program keeps growing as the number of messages etc. increases. Postfix is designed to stay in control.
- ▶ **Flexibility.** Postfix is built from over a dozen little programs that each perform only one specific task: receive a message via SMTP, deliver a message via SMTP, deliver a message locally, rewrite an address, and so on. Sites with specific requirements can replace one or more little programs by alternative versions or use one of the many extension interfaces. And it is easy to disable functionality, too: firewalls and client workstations don't need local delivery at all.
- ▶ **Security.** Postfix uses multiple layers of defence to protect the local system against intruders. Almost every Postfix daemon can run in a chroot jail with fixed low privileges. There is no direct path from the network to the security-sensitive local delivery programs - an intruder has to break through several other programs first. Postfix does not even trust the contents of its own queue files, or the contents of its own IPC messages. Postfix filters sender-provided information before exporting it via environment variables. Last but not least, no Postfix program is set-uid.

2.2.2 Where to get Postfix?

Postfix can be downloaded from <http://www.postfix.org/>. If you are using a Linux distribution precompiled Postfix packages can preferably be installed. Postfix packages exist for nearly all Linux distributions.

2.2.3 Why Postfix

The Postfix software package provides higher performance, better security, and simpler configuration than other standard compliant Mail Transport Agents (MTA). Postfix uses a modular design rather than the monolithic sendmail design. Due to this modularity, Postfix is a very robust and efficient MTA. Furthermore the configuration of Postfix is much easier than a sendmail configuration.

The combination and integration of Postfix, virus scanner and SPAM blocker is simple.

2.3 Architecture of the Postfix system

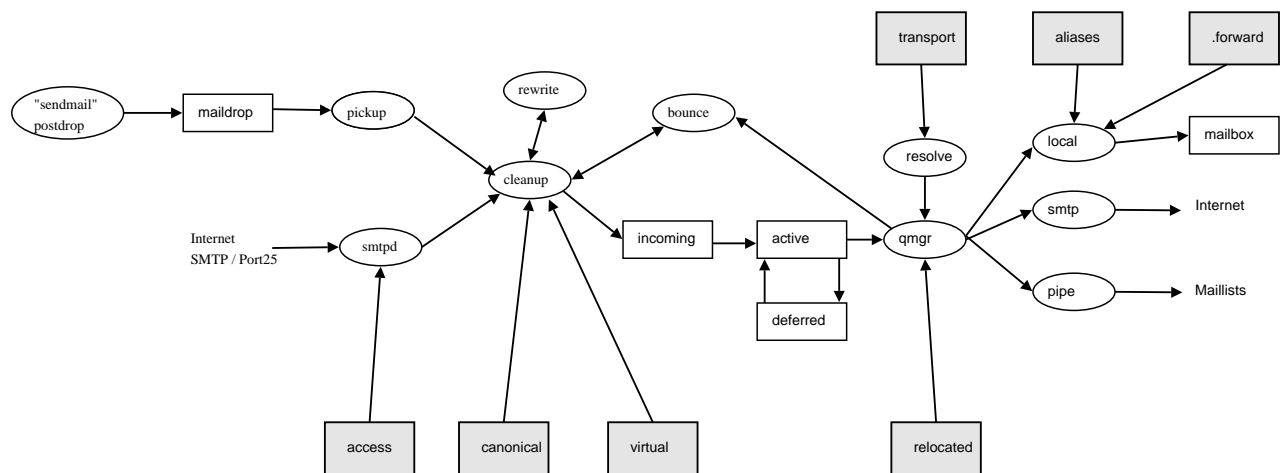


Figure 2.1: Postfix Architecture

To handle the complexity of the MTA, Postfix has been divided into different modules. Each module is responsible for a specific task. Once this task has been done, the module gives the mail to another module. The figure shows also the configuration files for the different modules.

When a message enters the Postfix mail system, the first stop on the inside is the incoming queue.

2 Postfix

Network mail enters Postfix via the `smtpd` server. This server removes the SMTP protocol encapsulation, enforces some sanity checks to protect Postfix, and gives the sender, recipients and message content to the cleanup module. The `smtpd` module can be configured to block unwanted mail according to many parameters.

Local submissions are received with the Postfix `sendmail` compatibility command, and are queued in the maildrop queue by the privileged `postdrop` command. This arrangement even works while the Postfix mail system is not running. The local pickup module picks up local submissions, enforces some sanity checks to protect Postfix, and gives the sender, recipients and message content to the `cleanup` module.

Mail from sources internal to Postfix (e.g. bounces) is given directly to the cleanup module.

The `cleanup` module implements the final processing stage before mail is queued. It adds missing `From:` and other message headers to the message. Furthermore, it gives the mail to the `rewrite` module, which rewrites addresses to the standard "user@fully.qualified.domain". Postfix currently does not implement a rewriting language, but a lot can be done via table lookups and, if need be, regular expressions. Optionally, the `cleanup` module can be configured to do light-weight content inspection with regular expressions. The `cleanup` module places the result as a single file into the incoming queue, and notifies the queue manager of the arrival of new mail.

The queue manager (the `qmgr` server process in the figure) is the heart of Postfix mail delivery. It contacts the `smtp`, `local`, `virtual`, `pipe`, `discard` or `error` delivery agents, and sends a delivery request for one or more recipient addresses. The `discard` and `error` delivery agents are special: they discard or bounce all mail, they are not shown in the figure above.

The queue manager maintains a small active queue with the messages that it has opened for delivery. The active queue acts as a limited window on potentially large incoming or

2 Postfix

deferred queues. The limited active queue prevents the queue manager from running out of memory under heavy load.

The queue manager maintains a separate deferred queue for mail that cannot be delivered, so that a large mail backlog will not slow down normal queue accesses.

The `trivial-rewrite` server resolves each recipient address according to its local and remote address class. Additional routing information can be specified with the optional transport table. The `trivial-rewrite` server optionally queries the relocated table for recipients whose address has changed; mail for such recipients is returned to the sender with an explanation.

The `smtp` client looks up a list of mail exchangers for the destination host, sorts the list by preference, and tries each server in turn until it finds a server that responds. It then encapsulates the sender, recipient and message content as required by the SMTP protocol; this includes conversion of 8-bit MIME to 7-bit encoding.

The local delivery agent understands UNIX-style mailboxes, `qmail`-compatible `maildir` directories, `Sendmail`-style system-wide aliases databases, and `Sendmail`-style per-user `.forward` files. Multiple local delivery agents can be run in parallel, but parallel delivery to the same user is usually limited.

The local delivery agent has hooks for alternative forms of local delivery: you can configure it to deliver to mailbox files in user home directories, you can configure it to delegate mailbox delivery to an external command such as `procmail`, or you can delegate delivery to a different Postfix delivery agent.

The virtual delivery agent is a bare-bones delivery agent that delivers to UNIX-style mailbox or `qmail`-style `maildir` files only. This delivery agent can deliver mail for multiple domains, which makes it especially suitable for hosting lots of small domains on a single machine.

The pipe mailer is the outbound interface to other mail processing systems (the Postfix

sendmail command being the inbound interface). The interface is UNIX compatible: it provides information on the command line and on the standard input stream.

2.4 Installing Postfix

2.4.1 Installation

During the tutorial, Postfix will be installed on a Debian Sarge system. The quickest way to install it is to use the Debian package management system, also used by Ubuntu. As root enter the following command to auto install postfix:

```
# apt-get install postfix
```

You need to have a network connection or Ubuntu or Debian CDs to use the prior command. The package management of Debian automatically checks for dependencies and installs the missing packages on which Postfix relies. On a Debian system, exim is installed as default MTA. The package management disables this MTA and configures Postfix as the new default MTA. Ubuntu does not by default install any MTA.

Using the Debian package management is easy, but has some disadvantages. You cannot specify all the options and features that Postfix offers. You have to live with the Postfix that Debian (or Ubuntu) people compiled for you. In practice, this is seldomly a problem – I'm using the Debian Postfix package on servers ranging from a handful to tens of thousands of users.

For this tutorial we will use the precompiled Ubuntu/Debian package.

2.4.2 Where are the files?

After installing the package, you want to know where the different Postfix files are stored.

There are 2 groups of files : binaries and configuration files.

The postfix binaries:

Path	Tool	Function
/usr/sbin	postfix	The postfix command controls the operation of the mail system. It is the interface for starting, stopping, and restarting the mail system, as well as for some other administrative operations. This command is reserved to the super-user.
/usr/sbin	postalias	The postalias command maintains Postfix aliases type databases. This is the program that does the work for the newaliases command.
/usr/sbin	postcat	The postcat command displays the contents of Postfix queue files.
/usr/sbin	postconf	The postconf command displays or updates Postfix main.cf parameters and displays system dependent information about the supported file locking methods, and the supported types of lookup tables.
/usr/sbin	postdrop	The postdrop command is can be used to take mails from command line and put them into the queues
/usr/sbin	postkick	With the postkick command you can send commands to the postfix modules
/usr/sbin	postlock	The postlock command provides Postfix-compatible mailbox locking for use in, for example, shell scripts.
/usr/sbin	postlog	The postlog command provides Postfix-compatible logging for shell scripts.

2 Postfix

Path	Tool	Function
/usr/sbin	postmap	The postmap command maintains Postfix lookup tables such as canonical, virtual and others. It is a cousin of the sendmail makemap command.
/usr/sbin	postqueue	The postqueue command is the privileged command that is run by Postfix sendmail and mailq in order to flush or list the mail queue.
/usr/sbin	postsuper	The postsuper command maintains the Postfix queue. It removes old temporary files, and moves queue files into the right directory after a change in the hashing depth of queue directories. This command is run at mail system startup time and when Postfix is restarted.
/usr/bin	mailq	The Postfix mailq command lists the content of the mail queues. It is a replacement of sendmail's mailq
/usr/bin	newaliases	The Postfix newaliases command is the same as postalias and replaces sendmail's newaliases.
/usr/sbin	sendmail	The Postfix sendmail command is a sendmail compatible application and can be used to send mail from local applications.

The postfix configuration files:

Path	Filename	Function
/etc/postfix	main.cf	Contains operational parameters used by the Postfix modules when processing messages
/etc/postfix	master.cf	Contains parameters used by the Postfix master program when running core programs
/etc/postfix	access	Maps remote SMTP hosts to an accept/deny table for security

Path	Filename	Function
/etc	aliases	Maps alternative recipients to local mailboxes
/etc/postfix	canonical	Maps alternative mailbox names to real mailboxes for message headers
/etc/postfix	relocated	Maps an old user mailbox name to a new mailbox name
/etc/postfix	transport	Maps domain names to delivery methods for remote host connectivity and delivery
/etc/postfix	virtual	Maps recipients and domains to local mailboxes for delivery

2.5 Basic configuration

The first step after the installation is to set up a basic configuration to receive and to send mails for the Tux-Industries domain. Remember that we want to set up a mail server for the luxembourg.tux-industries.com domain. We assume that DNS has the corresponding MX records. Our mail-server will be a member of our private network 192.168.1.0/24, and has the name mailserver.luxembourg.tux-industries.com with the corresponding IP-Address 192.168.1.5.

Postfix has several hundred configuration parameters that are controlled via the main.cf file. Fortunately, all parameters have sensible default values. In many cases, you need to configure only two or three parameters before you can start to play with the mail system.

The first parameters of interest specify the machine's identity and role in the network.

- ▶ What domain name to use in outbound mail
- ▶ What domains to receive mail for
- ▶ What clients to relay mail from

- ▶ What destinations to relay mail to
- ▶ What delivery method: direct or indirect

The default values for many other configuration parameters are derived from just these.

The next parameter of interest controls the amount of mail sent to the local postmaster:

- ▶ What trouble to report to the postmaster
- ▶ What you need to know about Postfix logging

If your machine has high security requirements you may want to run Postfix daemon processes inside a chroot environment.

- ▶ Running Postfix daemon processes chrooted

If you run Postfix on a virtual network interface, or if your machine runs other mailers on virtual interfaces, you'll have to look at the other parameters listed here as well:

- ▶ My own hostname
- ▶ My own domain name
- ▶ My own network addresses

2.5.1 Postfix configuration files

By default, Postfix configuration files are in `/etc/postfix`. The two most important files are `main.cf` and `master.cf`; these files must be owned by root. Giving someone else write permission to `main.cf` or `master.cf` (or to their parent directories) means giving root privileges to that person.

2 Postfix

In `/etc/postfix/main.cf` you will have to set up a minimal number of configuration parameters. Postfix configuration parameters resemble shell variables, with two important differences: the first one is that Postfix does not know about quotes like the UNIX shell does.

You specify a configuration parameter as:

```
/etc/postfix/main.cf: parameter = value
```

and you use it by putting a "\$" character in front of its name:

```
/etc/postfix/main.cf: other_parameter = $parameter
```

You can use `$parameter` before it is given a value (that is the second main difference with UNIX shell variables). The Postfix configuration language uses lazy evaluation, and does not look at a parameter value until it is needed at runtime.

Postfix uses database files for access control, address rewriting and other purposes. Here is a common example of how Postfix invokes a database:

```
/etc/postfix/main.cf: virtual_alias_maps = hash:/etc/postfix/virtual
```

Whenever you make a change to the `main.cf` or `master.cf` file, execute the following command as root in order to refresh a running mail system:

```
# postfix reload
```

2.5.2 My own hostname

The `myhostname` parameter specifies the fully-qualified domain name of the machine running the Postfix system. `$myhostname` appears as the default value in many other Postfix configuration parameters.

By default, `myhostname` is set to the local machine name. If your local machine name is not in fully-qualified domain name form, or if you run Postfix on a virtual interface, you will have to specify the fully-qualified domain name that the mail system should use.

Alternatively, if you specify `mydomain` in `main.cf`, then Postfix will use its value to generate a fully-qualified default value for the `myhostname` parameter.

You can check what the value of a variable is with the `postconf` command:

```
# postconf myhostname
myhostname = mailserver.luxembourg.tux-industries.com
```

Example

```
/etc/postfix/main.cf:
myhostname = mailserver.luxembourg.tux-industries.com
```

2.5.3 My own domain name

The `mydomain` parameter specifies the parent domain of `$myhostname`. By default, it is derived from `$myhostname` by stripping off the first part (unless the result would be a top-level domain).

Conversely, if you specify `mydomain` in `main.cf`, then Postfix will use its value to generate a fully-qualified default value for the `myhostname` parameter.

Example:

```
/etc/postfix/main.cf:  
mydomain = luxemburg.tux-industries.com
```

2.5.4 My own network addresses

The `inet_interfaces` parameter specifies all network interface addresses that the Postfix system should listen on; mail addressed to "user@[network address]" will be delivered locally, as if it is addressed to a domain listed in `$mydestination`.

You can override the `inet_interfaces` setting in the Postfix `master.cf` file by prepending an IP address to a server name.

The default is to listen on all active interfaces. If you run mailers on virtual interfaces, you will have to specify what interfaces to listen on.

IMPORTANT: If you run MTA's on virtual interfaces you must specify explicit `inet_interfaces` values for the MTA that receives mail for the machine itself: this MTA should never listen on the virtual interfaces or you would have a mailer loop when a virtual MTA is down.

Example:

```
/etc/postfix/main.cf:  
inet_interfaces = $myhostname localhost
```

Note: you need to stop and start Postfix after changing this parameter.

2.5.5 What domain name to use in outbound mail

The `myorigin` parameter specifies the domain that appears in mail that is posted on this machine. The default is to use the local machine name, `$myhostname`, which defaults to the name of the machine. Unless you are running a really small site, you probably

2 Postfix

want to change that into `$mydomain`, which defaults to the parent domain of the machine name.

For the sake of consistency between sender and recipient addresses, `myorigin` also specifies the domain name that is appended to an unqualified recipient address.

Examples :

```
/etc/postfix/main.cf:  
myorigin = $myhostname (default: send mail as "user@$myhostname")  
myorigin = $mydomain (probably desirable: "user@$mydomain")
```

For our example we prefer `user@luxembourg.tux-industries.com`:

```
myorigin = $mydomain
```

2.5.6 What domains to receive mail for

The `mydestination` parameter specifies what domains this machine will deliver locally, instead of forwarding to another machine. The default is to receive mail for the machine itself. To handle additional domains or sub-domains, you have to specify this using the virtual table. This will be explained later in this document.

IMPORTANT: If your machine is a mail server for its entire domain, you must list `$mydomain` as well.

Example 1: default setting.

```
/etc/postfix/main.cf:  
mydestination = $myhostname localhost.$mydomain localhost
```

Example 2: `luxembourg.tux-industries.com` example

```
/etc/postfix/main.cf:  
myhostname=mailserver.luxembourg.tux-industries.com  
mydomain=luxembourg.tux-industries.com  
mydestination = $myhostname localhost.$mydomain localhost $mydomain
```

Caution: in order to avoid mail delivery loops, you must list all hostnames of the machine, including `$myhostname`, and `localhost.$mydomain`.

2.5.7 What clients to relay mail from

By default, Postfix will forward mail from clients in authorised network blocks to any destination. Authorised networks are defined with the `mynetworks` configuration parameter. The default is to authorise all clients in the IP subnetworks that the local machine is attached to.

IMPORTANT: If your machine is connected to a wide area network then your default `mynetworks` setting may be too friendly and allow strangers to use your machine as a spam relay.

The `mynetworks_style` parameter can take one of the three parameters: `host`, `subnet` (default), `class`.

Specify "`mynetworks_style = host`" when Postfix should forward mail from only the local machine.

Specify "`mynetworks_style = subnet`" (the default) when Postfix should forward mail from SMTP clients in the same IP subnetworks as the local machine. On Linux, this works correctly only with interfaces specified with the `ifconfig` command.

Specify "`mynetworks_style = class`" when Postfix should forward mail from SMTP clients in the same IP class A/B/C networks as the local machine. Don't do this with a dial-up

2 Postfix

site - it would cause Postfix to "trust" your entire provider's network. Instead, specify an explicit `mynetworks` list by hand, as described below.

Alternatively, you can specify the `mynetworks` list by hand, in which case Postfix ignores the `mynetworks_style` setting. To specify the list of trusted networks by hand, specify network blocks in CIDR (`network/mask`) notation, for example:

```
/etc/postfix/main.cf:  
mynetworks = 168.100.189.0/28, 127.0.0.0/8
```

For our example we only allow to relay mail from our subnet:

You have 2 possibilities to specify this :

1. We specify to use subnet in `mynetwork_style`:

```
/etc/postfix/main.cf:  
mynetwork_style = subnet
```

2. We ignore `mynetwork_style` and we specify directly our subnet:

```
/etc/postfix/main.cf:  
mynetworks = 127.0.0.0/8 192.168.1.0/24
```

2.5.8 What destinations to relay mail to

By default, Postfix will forward mail from strangers (clients outside authorised networks) to authorised remote destinations only. Authorised remote destinations are defined with the `relay_domains` configuration parameter. The default is to authorise all domains (and subdomains) of the domains listed with the `mydestination` parameter.

The `relay_domains` parameter restricts what destinations this system will relay mail to. By default, Postfix relays mail

2 Postfix

- ▶ from "trusted" clients (IP address matches `$mynetworks`) to any destination
- ▶ from "untrusted" clients to destinations that match `$relay_domains` and subdomains thereof, except addresses with sender-specified routing.

The default `relay_domains` value is `$mydestination`.

In addition to the above, the Postfix SMTP server by default accepts mail that Postfix is final destination for:

- ▶ destinations that match `$inet_interfaces` or `$proxy_interfaces`,
- ▶ destinations that match `$mydestination`
- ▶ destinations that match `$virtual_alias_domains`,
- ▶ destinations that match `$virtual_mailbox_domains`.

These destinations do not need to be listed in `$relay_domains`.

For the Tux-Industries example we should disable this parameter as we only want to accept mails for our domain, resp. sub-domain.

2.5.9 What delivery method: direct or indirect

By default, Postfix tries to deliver mail directly to the Internet. Depending on your local conditions this may not be possible or desirable. For example, your system may be turned off outside of office hours, it may be behind a firewall, or it may be connected via a provider who does not allow direct mail to the Internet. In those cases you need to configure Postfix to deliver mail indirectly via a relay host.

Example:

In our example we want to deliver the mails directly to the Internet, so we specify the following:

```
relayhost =
```

To make things more complicated, we could imagine to relay our mails first to our top level mail-server (mailserver.tux-industries.lu) which then sends the mails to the Internet. In this case we have to specify the parameter:

```
relayhost = [mailserver.tux-industries.lu]
```

2.5.10 What trouble to report to the postmaster

You should set up a postmaster alias in the aliases table that directs mail to a human person. The postmaster address is required to exist, so that people can report mail delivery problems. While you're updating the aliases table, be sure to direct mail for the super-user to a human person too.

```
/etc/aliases:  
postmaster: root, cm
```

Execute the command "newaliases" after changing the aliases file.

The Postfix system reports problems to the postmaster alias. You may not be interested in all types of trouble reports, so this reporting mechanism is configurable. The default is to report only serious problems (resource, software) to postmaster:

Default setting:

```
/etc/postfix/main.cf:  
notify_classes = resource, software
```

If you are experimenting with complicated setups, it can be helpful to add "policy" and "protocol" to the list; this will give you detailed information in case of problems.

2.5.11 What you need to know about Postfix logging

Postfix daemon processes run in the background, and log problems and normal activity to the syslog daemon. The syslogd process sorts events by class and severity, and appends them to log-files. The logging classes, levels and log-file names are usually specified in `/etc/syslog.conf`. At the very least you need something like:

```
/etc/syslog.conf:
mail.err /dev/console
mail.debug /var/log/maillog
```

After changing the `syslog.conf` file, send a "HUP" signal to the `syslogd` process.

2.5.12 Running Postfix daemon processes chrooted

Postfix daemon processes can be configured (via the `master.cf` file) to run in a chroot jail. The processes run at a fixed low privilege and with file system access limited to the Postfix queue directories (`/var/spool/postfix`). This provides a significant barrier against intrusion. The barrier is not impenetrable (chroot limits file system access only), but every little bit helps.

With the exception of Postfix daemons that deliver mail locally and/or that execute non-Postfix commands, every Postfix daemon can run chrooted.

Sites with high security requirements should consider to chroot all daemons that talk to the network: the `smtp` and `smtpd` processes, and perhaps also the `lmtp` client.

The default `/etc/postfix/master.cf` file specifies that no Postfix daemon runs chrooted. In order to enable chroot operation, edit the file `/etc/postfix/master.cf`, and follow instructions in the file. When you're finished, execute "`postfix reload`" to make the change effective.

During this Tux-Industries tutorial, we will not handle the chroot jail, but for mail-servers that resides directly on the Internet it is highly recommended to run postfix in a chroot environment.

2.6 The basic configuration files for our example

`/etc/postfix/main.cf`

```
myhostname = mailserver.luxembourg.tux-industries.lu
mydomain = luxembourg.tux-industries.lu
inet_interfaces = $myhostname localhost
myorigin = $mydomain
mydestination = $myhostname localhost.$mydomain localhost $mydomain
mynetwork_style = subnet
# relay_domains =
relayhost =
notify_classes = resource, software
# Some installation specific parameters
command_directory = /usr/sbin
daemon_directory = /usr/lib/postfix
mail_owner = postfix
alias_maps = hash:/etc/aliases
smtp_banner = $myhostname ESMTP $mail_name
```

`/etc/aliases`

```
postmaster: root, cm
christian.mock: cm
```

```
test.testperson: test
```

```
...
```

Using the above configuration files, a Postfix system will be able to handle mails from and to `luxembourg.tux-industries.com` domain.

2.7 Additional configuration

2.7.1 The virtual table

For our domain (`luxembourg.tux-industries.com`) we want to create two new subdomains: `sales.luxembourg.tux-industries.com` and `support.luxembourg.tux-industries.com`. After configuring the name-servers to handle these new domains, Postfix has to accept mails for this domains. Postfix can handle additional domains by using the virtual table. The virtual table can be seen as an extension to the old sendmail alias table. The difference of these two tables is that the alias table can only rewrite local mail addresses, and is used for all domains in `$mydestination`. The virtual table at the other side can be used to rewrite whole domains, and supports the same username under different domains.

There are four different record types in the virtual table. Each one defines a different type of a virtual domain:

2 Postfix

<code>domain.name anything</code>	Allows Postfix to receive messages for <code>domain.name</code> and place them in the message queue. This must be used in addition to one of the forms below.
<code>user@domain address1, address2, ...</code>	Receives messages for <code>user@domain</code> and forwards them to addresses listed. This redirection has the highest priority.
<code>user address1, address2, ...</code>	Receives messages for <code>user</code> on the local host and forwards them to the addresses listed.
<code>@domain address1, address2, ...</code>	Receives messages for all users in <code>domain</code> and forwards them to the addresses listed.

Example

```
linuxdays.lu          anything
info@linuxdays.lu    patrick.harpes@tudor.lu, eric.dondelinger@tudor.lu
support.linux.lu      anything
@support.linux.lu     eric.dondelinger@tudor.lu
```

This configuration defines the following:

Incoming mails for `info@linuxdays.lu` should be directed to `patrick.harpes@tudor.lu` and `eric.dondelinger@tudor.lu`. Every message for the `support.linux.lu` domain should be forwarded to `eric.dondelinger@tudor.lu`.

After editing the virtual table, you have to tell Postfix to use this table. In the `main.cf` configuration file you have to specify the following line:

```
virtual_maps = hash:/etc/postfix/virtual
```

2 Postfix

This line says that postfix should look in `/etc/postfix` to find the virtual table. The `hash` keyword tells Postfix that this table uses the DBM-hash format. Because the file we created is in normal ASCII format, we have to convert the file into the DBM-hash format. This can be done with the `postmap` command:

```
# cd /etc/postfix
# postmap virtual
```

2.7.2 The aliases table

You can use the aliases table to map “fake” mail addresses to actual system users. This is directly compatible with the sendmail program’s system of aliases table.

The aliases table differs from most of Postfix’s other lookup tables in that it has a slightly different form:

```
pattern: result
```

The colon is added to that pattern to make the Postfix alias file compatible with sendmail alias file. This simplifies converting sendmail mail servers to Postfix.

Example

```
postmaster: root
christian.mock: cm
```

This configuration defines the following:

Mails to `postmaster@localdomain.com` are forwarded to the `root` account on this local machine. Mails for `christian.mock@localdomain` are forwarded to the `cm` user account on this local machine.

2 Postfix

After editing the aliases table, you have to tell Postfix to use this table. In the `main.cf` configuration file you have to specify the following lines:

```
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
```

The first line says that postfix should look in `/etc` to find the aliases table. The `hash` keyword tells Postfix that this table uses the DBM-hash format. The second line specifies which files are to be rebuilt by the “`newaliases`” command. Don’t forget to run it after changing the aliases file.

3 Virus scanner

To achieve a more secure mail server, we want to implement a virus scanner to scan for viruses on incoming mails.

3.1 AMaViS - A Mail Virus Scanner

AMaViS is a script that interfaces a mail transport agent (MTA) with virus scanners. It is a high-performance interface between mailer (MTA) and content checkers: virus scanners, and/or SpamAssassin. It communicates to MTA via (E)SMTP or LMTP, or by using wrapper software. There are different versions of AMaViS out: amavis, amavis-perl, amavisd, amavis-ng and amavisd-new. This tutorial will only handle the amavisd-new version.

To filter the incoming mails, Postfix must send the message to AMaViS. AMaViS then unpacks the message and related attachments and checks the contents with one or more virus scanners. Additionally the message can also be checked by SpamAssassin.

To do this work, there are different steps to do:

- ▶ amavisd-new opens a port on localhost:10024 with an SMTP server
- ▶ Postfix relays the message to localhost:10024. That means that the message leaves the Postfix system

3 Virus scanner

- ▶ amavisd-new filters the incoming message on its localhost:10024 port. If the message contains no virus, amavisd-new gives the message back to Postfix using the localhost:10025 port. To avoid mail loops, the messages that enter on localhost:10025 port will not be forwarded to amavisd-new
- ▶ If a mail includes a virus, the message will not be sent back to Postfix, but amavisd-new will store it in a special directory, and optionally send notification emails to the sender, recipient or administrator.

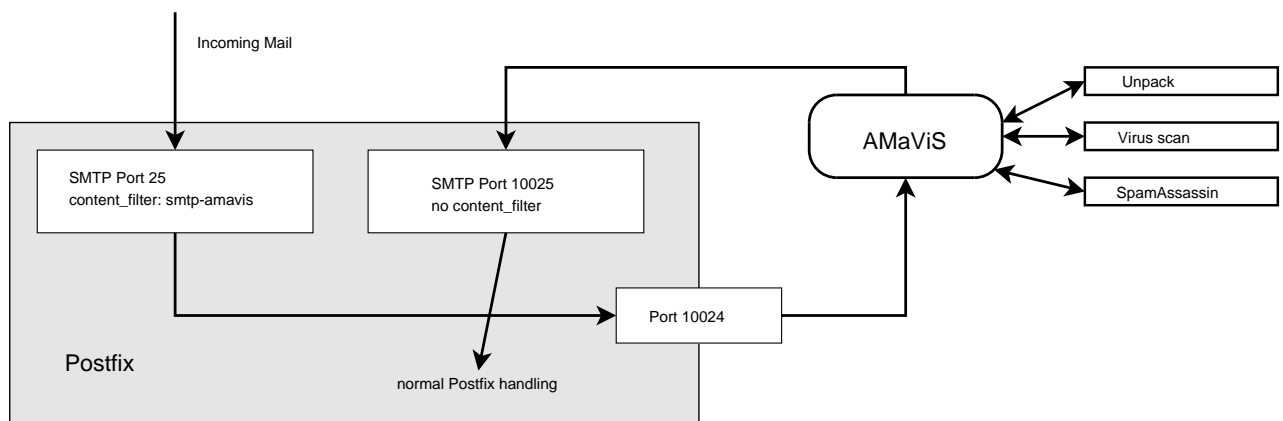


Figure 3.1: Postfix in combination with AMaViS

3.1.1 Configuring Postfix to use Amavis-new

The first step is to install amavisd-new. As we are using Ubuntu/Debian, the package management can install it:

```
# apt-get install amavisd-new
```

After installation, we have to tell Postfix to send the mails out to amavisd-new. We have to define a new transport method by adding the following into the `/etc/postfix/master.cf` file:

3 Virus scanner

```
smtp-amavis unix - - n - 2 smtp -o smtp_data_done_timeout=1800
-o disable_dns_lookups=yes
```

Furthermore we have to define the smtp server without `content_filter` running on the `localhost:10025` port. We have to add this line to `/etc/postfix/master.cf`

```
localhost:10025 inet n - n - - smtpd -o content_filter=
```

After restarting Postfix, the new smtp server should answer on port 10025. We can test it using the following command:

```
# telnet localhost 10025
```

Now we have to tell Postfix to relay all incoming mails to `amavisd-new` which listens on port `localhost:10024`. This parameter is specified in the `/etc/postfix/main.cf` file:

```
content_filter = smtp-amavis:[127.0.0.1]:10024
```

3.1.2 Configuring `amavisd-new`

The `/etc/amavis/amavis.conf` file is a large Perl script to define the parameters. For the moment we only define a few essential parameters, to tell `amavisd-new` to establish a smtp server on port 10024:

```
$mydomain = 'tux-industries.lu';
$myhostname = 'mailserver.tux-industries.lu';
$forward_method = 'smtp:127.0.0.1:10025'; # where to forward checked mail
$inet_socket_port = 10024;
```

3 Virus scanner

On a Debian configuration, you should now be able to start `amavisd-new` and test it using the following command:

```
# telnet localhost 10024
```

`amavisd-new` should now answer on port 10024.

3.1.3 Which Virus-scanner to use?

AMaViS itself doesn't contain a virus-scanner. So we need a virus-scanner to scan and to find viruses in our messages. There are many products on the market, but for this tutorial we want to use a free virus scanner, so the decision to use ClamAV has been taken. ClamAV is maybe not the best anti-virus, but it shows how to integrate a virus checker, and it might be sufficient. AMaViS also allows to use more than one anti virus product. That means that you can install more than one virus-scanner, and configure AMaViS to use all scanners on your system. This is even recommended for mission-critical systems. An important point for your choice of a virus-scanner is that the virus-definition files are up to date, and that these files can be installed easily using a script, so that the system administrator does not have to pay attention to these updates.

3.1.4 ClamAV

Clam AntiVirus is a GPL anti-virus toolkit for UNIX. The main purpose of this software is the anti-virus integration in mail servers (attachment scanning). The package provides a flexible and scalable multi-threaded daemon, a command line scanner, and a tool for automatic updating via Internet. The programs are based on a shared library, distributed with the Clam AntiVirus package, which you can use with your own software. Most importantly, the virus database is kept up to date .

3 Virus scanner

Main features are:

- ▶ command-line scanner
- ▶ fast, multi-threaded daemon
- ▶ milter interface for sendmail
- ▶ database updater with support for digital signatures
- ▶ virus scanner C library
- ▶ on-access scanning (Linux and FreeBSD)
- ▶ detection of over 87000 viruses, worms and trojans
- ▶ built-in support for RAR (2.0), Zip, Gzip, Bzip2, Tar, MS OLE2, MS Cabinet files, MS CHM (Compressed HTML), MS SZDD
- ▶ built-in support for mbox, Maildir and raw mail files
- ▶ built-in support for Portable Executable files compressed with UPX, FSG, and Petite

3.1.5 Installing and configuring ClamAV

ClamAV can be installed as a Debian package. The following command will install the software:

```
# apt-get install clamav-daemon
```

Note that since a virus scanner must be kept current, you should use the appropriate repository for your distribution (e.g. volatile.debian.org or Ubuntu backports).

3 Virus scanner

Because a virus-scanner needs up to date virus definition files, the first step after installation is to download the latest virus definition file. This can be done using `freshclam`. `freshclam` updates the virus database. It's a part of the Clam AntiVirus package. It requires an Internet connection. `freshclam` can be run in background to check regularly for new virus definition files. For the first time, we will start it in foreground:

```
# freshclam
Downloading main.cvd [*]
main.cvd updated (version: 42, sigs: 83951, f-level: 10, builder: tkojm)
Downloading daily.cvd [*]
daily.cvd updated (version: 2496, sigs: 3425, f-level: 9, builder: diego)
Database updated (87376 signatures) from db.local.clamav.net
```

ClamAV comes with two binaries for scanning files:

- ▶ `clamscan` is a stand-alone binary to scan a file, a directory
- ▶ `clamscan` works together with the `clamd` daemon. With this version you can have a gain of performance when scanning a huge amount of files.

To test a virus-scanner we need of course a virus to check if the virus-scanner filters it. Using “real” viruses is dangerous. There are test viruses out, and these viruses can be used without compunction. They can be downloaded at <http://www.eicar.org>.

- ▶ Download the virus sample files from www.eicar.org.
- ▶ Copy them into a directory `/tmp/virus-test`
- ▶ Copy other files to this directory
- ▶ `$ cd /tmp/virus-test`
- ▶ `$ clamscan`
- ▶ To test `clamavd` run `clamscan`

3.1.6 Integrate ClamAV with AMaViS

Once the virus-scanner and AMaViS are working, we have to configure AMaViS to use ClamAV for checking viruses.

Usually, AMaViS is configured in `/etc/amavisd.conf` or `/etc/amavis/amavisd.conf`; with Debian and Ubuntu, however, the configuration file is split into files in `/etc/amavis/conf.d`, and you should edit the file `50-user` there, which overrides all other files.

The virus-scanner settings defined in the Perl list `@av_scanner`. This list contains many virus-scanners. To configure AMaViS to use ClamAV please locate first this list, and then look if there is an entry for ClamAV. This entry should look like this:

```
### http://www.clamav.net/
['Clam Antivirus-clamd',
 \&ask_daemon, ["CONTSCAN {}\n", "/var/run/clamav/clamdctl"],
 qr/\bOK$/, qr/\bFOUND$/,
 qr/^.*?: (?!Infected Archive)(.*) FOUND$/ ],
# NOTE: run clamd under the same user as amavisd; match the socket
# name (LocalSocket) in clamav.conf to the socket name in this entry
# When running chrooted one may prefer: ["CONTSCAN {}\n", "$MYHOME/clamd"],
```

The expression `/var/run/clamav/clamdctl` is the path to the localsocket to connect to clamavd. To be sure that AMaViS and ClamAV use the same socket, you have to check in the ClamAV configuration file the socket settings. This configuration file can be found in `/etc/clamav/clamd.conf`.

```
LocalSocket /var/run/clamav/clamdctl
```

The remaining problem is that ClamAV and AMaViS run as different users, so they can't read the other daemon's files. To work around this, add to `/etc/clamav/clamd.conf`:

3 Virus scanner

`AllowSupplementaryGroups`

Add the ClamAV user to the AMaViS group in `/etc/group`:

```
amavis:x:112:clamav
```

Then you need to re-enable virus scanning by copying and uncommenting `bypass_virus_checks_maps` from `15-content_filter_mode`, and restart the daemons:

```
# /etc/init.d/clamav-daemon restart
# /etc/init.d/amavis restart
```

In `/var/log/mail.log`, AMaViS logs which components are enabled, search for these lines:

```
ANTI-VIRUS code    loaded
Using internal av scanner code for (primary) ClamAV-clamd
```

After restarting Postfix and AMaViS your system should now be able to filter viruses. You can test it by sending the sample virus eicar.

4 Anti-Spam Configuration

4.1 Short intro on spam

4.1.1 Risk assessment

Automatically classifying messages as either spam or ham (non-spam) has the risk of mis-classification. A piece of spam classified as ham is a False Negative, a piece of ham classified as spam is a False Positive. In an ideal world, you would be able to reduce both counts to zero¹.

Since we don't live in an ideal world, you need to think about the issue: what level of FP and FN can you and your organization accept? Would you rather have your users see no spam at all and risk losing legitimate mail, or can they live with a few spams per day and person and virtually never lose legitimate mail?

The next question, then, is what to do with the spam. The choices are:

- ▶ Rejecting during the SMTP session

This means the sender will get a bounce message in the case of an FP, but the faked sender of a spam will not get one (because spam-sending SMTP engines don't generate bounces).

¹Actually, in an ideal world you wouldn't need spam filters at all

4 Anti-Spam Configuration

► Bouncing

This means the sender of an FP will get a bounce, but also that the faked sender of a spam will be spammed by you with a bounce for a message he didn't send.

► Dropping

This means you leave the faked spam sender alone, but the sender of an FP gets no notice of his mail disappearing, either.

► Tagging

You pass along the message to the recipient, but tag it in the subject and/or header. The recipient can automatically filter it to his "junk" folder, which also means that it most probably has the same effect as dropping, except you (as the mail admin) aren't the one dropping the important message from the important customer (which will inevitably come up in the discussion), but the end-user will.

4.1.2 Where does spam come from?

Spam is either sent from the spammer's machine (hosted in a datacenter of an anti-social ISP somewhere), or from a machine the spammer abuses for his purposes. This can be an open SMTP relay, an open proxy, or a "zombie" machine infected with malware that is used to spam and DDoS.

4.1.3 What characteristics can we use to detect spam?

- Where does the message come from? (IP address, hostname, ISP, country)
- How is it sent? (Characteristics of the SMTP session)
- Technical features (Headers, MIME structure, Text and HTML parts)
- Content (Easy for humans, hard for computers)

4 Anti-Spam Configuration

More material is in my talk held at the 2003 linuxdays.lu: <http://www.tahina.priv.at/talks/spamblocking.pdf>

4.2 Blacklists/RBLs/DNSBLs

"Realtime Blackhole Lists"/"DNS based Blacklists"

DNSBLs are lists of IP addresses or host/domain names that match certain criteria, like "open relay", "open proxy", "has sent mail to a spamtrap address recently", "assigned to a well known spammer", "located in country X".

DNSBLs are queried via DNS; for IP address lists, the address is reversed like for a PTR query; if we were to look for 209.88.103.4:

```
$ host -a 4.103.88.209.proxies.relays.monkeys.com
4.103.88.209.proxies.relays.monkeys.com IN A 127.0.0.2
4.103.88.209.proxies.relays.monkeys.com IN TXT "BLOCKED: See http://www.monkeys.com/upl/listed-ip-0.cgi?ip=209.88.103.4"
```

For lists of host/domain names ("RHSBLs" for "Right Hand Side"), the hostname is prepended to the list's domain:

```
# host -t txt dvd-porno-shop.com.multi.surbl.org
dvd-porno-shop.com.multi.surbl.org text "Blocked, dvd-porno-shop.com on lists [ws][ob][ab][jp], See: http://www.surbl.org"
```

IP DNSBLs can be used to refuse mail from the listed machines; RHSBLs can be used to check URLs in the messages against.

You rely on an external entity, so chose a trustworthy source; DNSBL operators have listed ISPs they didn't like in the past, and one has closed down his list by listing ALL IP addresses.

My recommendations:

4 Anti-Spam Configuration

- ▶ cbl.abuseat.org - based on spamtraps, lists proxies/zombies, no FP seen; <http://cbl.abuseat.org/>
- ▶ list.dsbl.org - open relays/proxies/etc; no FP seen; <http://dsbl.org/>
- ▶ [sbl.spamhaus.org](http://www.spamhaus.org/) - lists well-known spammer's hosting IP addresses; virtual no chance of FP, operators highly regarded in the community; <http://www.spamhaus.org/>

More:

<http://shopping.declude.com/Articles.asp?ID=97>

<http://www.moensted.dk/spam/>

<http://openrbl.org/>

<http://www.sconconsult.com/bill/dnsblhelp.html>

4.2.1 Using DNSBLs in Postfix

Using a DNSBL in Postfix means mail from IP addresses on that DNSBL will be REJECTED! Does your policy allow that?

Using IP-based DNSBLs in Postfix is very simple, you just add the list to `smtpd_recipient_restrictions` in `/etc/postfix/main.cf`.

Postfix also has “`smtpd_client_restrictions`” (for the connecting host), “`smtpd_helo_restrictions`” (to check the SMTP HELO/ESMTP EHLO parameter given by the client), “`smtpd_sender_restrictions`” (for the envelope sender address); these react directly to the respective SMTP command by giving an error code if the result is "REJECT", but some SMTP implementations ignore that, compatibility is biggest if the error is returned only when the client issues the RCPT TO command, even if it is the result of an earlier command. Plus we get a single configuration value to implement our SMTP-level policy...

“`smtpd_recipient_restrictions`” contains a list of tests which can return “PERMIT”, “REJECT” or nothing; the first match is used, so the general layout should be:

4 Anti-Spam Configuration

```
smtpd_recipient_restrictions =  
    <permit LAN users to send mail>,  
    <deny mail to non-local destinations>,  
    <whitelist for "spam lovers">,  
    <whitelist for spam FP>,  
    <block spam>,  
    permit
```

DNSBLs can be queried via “`reject_rbl_client dnsbl.domain.name`”.

Example:

```
smtpd_recipient_restrictions =  
    permit_mynetworks,  
    reject_unauth_destination,  
    check_recipient_access hash:/etc/postfix/spamlovers,  
    check_sender_access hash:/etc/postfix/sender_whitelist,  
    check_client_access hash:/etc/postfix/client_whitelist,  
    check_helo_access hash:/etc/postfix/helo_restrictions,  
    reject_non_fqdn_sender,  
    reject_unknown_sender_domain,  
    reject_rbl_client sbl.spamhaus.org,  
    reject_rbl_client cbl.abuseat.org,  
    permit
```

In Detail:

```
    permit_mynetworks,
```

Permit LAN users to send mail (LAN addresses defined in `mynetworks` in `main.cf`)

4 Anti-Spam Configuration

```
reject_unauth_destination,
```

Deny mail for non-local destinations (i.e. not `mydestination`, `virtual*_domains`, `relay_domains`, ...); prevent unauthorized relaying. Everything which passes this line is therefore for a local recipient.

```
check_recipient_access hash:/etc/postfix/spamlovers,
```

Allow certain addresses to receive unfiltered mail, e.g. `<postmaster@luxembourg.tux-industries.com>`

```
check_sender_access hash:/etc/postfix/sender_whitelist,
```

```
check_client_access hash:/etc/postfix/client_whitelist,
```

Whitelists for sender addresses (SMTP envelope!) and client IP addresses/hostnames, so you can whitelist people whose servers/ISPs happen to be on DNSBLs.

```
check_helo_access hash:/etc/postfix/helo_restrictions,
```

Reject HELO/EHLO with my hostname, domain name or IP address

```
reject_non_fqdn_sender,
```

Reject sender addresses without fully qualified domain part

```
reject_unknown_sender_domain,
```

Reject sender addresses with non-existing domain (returns temporary 4xx error code, can take 5 days for mail to bounce if sender has DNS problems)

4 Anti-Spam Configuration

```
reject_rbl_client sbl.spamhaus.org,  
reject_rbl_client cbl.abuseat.org,
```

Reject IP addresses on one of these lists

```
permit
```

Anything else will be accepted

Example map files:

spamlovers:

```
postmaster@luxembourg.tux-industries.com PERMIT  
abuse@luxembourg.tux-industries.com PERMIT
```

sender_whitelist:

```
user@example.com PERMIT  
some-domain.com PERMIT
```

client_whitelist:

```
10.2.3.4 PERMIT  
mail.example.com PERMIT
```

helo_restrictions:

```
mailserver.luxembourg.tux-industries.com 554 That's me, liar!  
tux-industries.com 554 That's me, liar!  
10.4.5.6 554 That's my address, spammer!  
my-other-domain.lu 554 That's also me, liar!
```

Don't forget to run "postmap filename" after changing a map file.

4.2.1.1 Testing the setup

If you know what you want, to avoid configuration errors (typos, ...):

```
soft_bounce = yes
```

Turns all 5xx SMTP errors (hard errors) into 4xx (soft errors), which causes the sender to retry, so mail won't get lost.

If you want to know what a certain configuration would achieve without interfering with mail delivery, use `warn_if_reject`, e.g.

```
smtpd_recipient_restrictions =  
[...]  
warn_if_reject reject_rbl_client new-dnsbl.org,  
[...]
```

"`reject_rbl_client new-dnsbl.org`" is evaluated, but the results are only logged ("`reject_warning`" not used to block. Analyze the log, and if you're happy, remove the "`warn_if_reject`" from the line.

After a while, the log file `/var/log/mail.log` should contain lines like these:

```
Jan 23 06:21:21 trumm postfix/smtpd[25929]: NOQUEUE: reject: RCPT from  
DSL01.212.114.234.165.NEFkom.net[212.114.234.165]: 571 Service  
unavailable; Client host [212.114.234.165] blocked using  
list.dsbl.org; http://dsbl.org/listing?212.114.234.165;  
from=<savspkseakkoqf@dcemail.com> to=<user@tahina.priv.at>  
proto=SMTP helo=<DSL01.212.114.234.165.NEFkom.net>
```

4 Anti-Spam Configuration

```
Jan 23 06:22:09 trumm postfix/smtpd[25929]: NOQUEUE: reject: RCPT from
unknown[211.187.64.233]: 554 <194.152.163.253>: Helo command
rejected: Don't lie to me.; from=<savspkseakkoqf@dcemail.com>
to=<user@tahina.priv.at> proto=SMTP helo=<194.152.163.253>
```

The first was rejected because the sending machine's IP address is listed in the dsbl.org DNSBL, the second because the HELO command contained my IP address.

4.3 SpamAssassin

SpamAssassin (<http://www.spamassassin.org/>) is *the* Open Source anti-spam filter, continually improved and adapted to spammer's new techniques.

4.3.1 How does it work?

SpamAssassin analyzes structure and content of a message with about 700 tests. Each test has a score associated with it; SA sums up the scores of the tests that matched, and if the sum is bigger than the (configurable) limit then the message is considered spam and can have "*****SPAM*****" added to the subject, X-Spam-Status headers, etc. The default limit is 5.0 points.

Tests include: MIME structure, keywords and key-phrases in headers and body, header analysis, DNSBL and RHSBL lookups, and Bayesian filtering.

Bayesian filtering is a technique where a database is trained on existing data sets; it works on single words (or "tokens"), where the training phase supplies the probability that a token occurs in spam and ham; then, you can look up the tokens from a new message in the DB and calculate the probability that it is spam or ham.

4 Anti-Spam Configuration

There's an "auto-whitelisting" feature which adapts the score of the current message based on the past scores messages from that sender have got; so if somebody who sent you low-scoring mail in the past and sends a message which triggers a few tests in SA now, the score will be adjusted downwards. The AWL database works on a combination of sender address and sending IP address, so a spam faking your friend's email address will not have its score adjusted since it will not come from your friend's mail server.

4.3.2 Installation

Use a current version of SpamAssassin! If you're using Debian Woody (3.0), put

```
deb http://www.backports.org/debian woody spamassassin
```

in your `/etc/apt/sources.list`, or get the source from <http://www.spamassassin.org/> and install it yourself. Ubuntu has an up-to-date version in it's backports repository – enable the respective line in `/etc/apt/sources.list`.

Take care to install all the recommended modules, too - especially `libnet-dns-perl` and `libmail-spf-query-perl` on Debian, or you'll lose all DNS and SPF-related functionality.

Do not start `spamd` - we'll integrate SA into Amavis in a few moments.

Debian's SpamAssassin installs into

- ▶ `/etc/spamassassin` - Site-Wide configuration
- ▶ `/usr/share/spamassassin` - Rules

4.3.3 Configuration

The system-wide SpamAssassin config file is in `/etc/spamassassin/local.cf`, per-user config is in `~/.spamassassin/user_prefs`. For reasons of simplicity, we'll only use the site-wide config here.

4 Anti-Spam Configuration

We're using SA 3.1.3 in the examples; make sure to read the `Mail::SpamAssassin::Conf` documentation either via `'man'` or `'perldoc'` to see which options other versions support.

What do we want to achieve?

- ▶ Messages classified as spam are marked with `"*****SPAM*****"` in the subject
- ▶ A system-wide bayesian database with a reasonable size
- ▶ Use network tests (DNSBLs etc)
- ▶ Use the auto-whitelisting feature

Ubuntu's `local.cf` is empty. We add the following statements:

```
auto_whitelist_path /var/lib/amavis/.spamassassin/whitelist
bayes_path /var/lib/amavis/.spamassassin/bayes
bayes_expiry_max_db_size 1500000
trusted_networks 127/8 192.168.1/24
skip_rbl_checks 0
ok_locales en
rewrite_header subject *****SPAM*****
report_safe 0
report_contact postmaster@tux-industries.com
lock_method flock
```

Explanation:

```
auto_whitelist_path /var/lib/amavis/.spamassassin/whitelist
bayes_path /var/lib/amavis/.spamassassin/bayes
```

4 Anti-Spam Configuration

These are in `~/spamassassin/` (per-user) normally, we put them in a central location to make them system-wide.

```
bayes_expiry_max_db_size 1500000
```

The default is much too low; 1 Mio uses about 32MB of disk space, so see how much "cached" RAM there is in the "free" output and size accordingly if you run a busy installation - you want the bayes DB to be cached all the time, and you want a large DB so the spams with lists of random words in them don't force the interesting spam keywords out of the DB.

```
trusted_networks 127/8 192.168.1/24
```

This should be automatically detected, but the algorithm isn't very reliable; give all IP addresses/networks of machines you trust, i.e. your LAN, your secondary MX etc.

```
skip_rbl_checks 0
```

Use the DNSBLs.

```
ok_locales en
```

Which character sets do you expect mail to be in? "en" includes all western character sets, so e.g. spam from Korea is scored.

```
rewrite_header subject *****SPAM*****
```

How to mark the messages classified as spam (this setting has no effect in conjunction with Amavis).

4 Anti-Spam Configuration

```
report_safe 0
```

If set to "1", spam is attached to a new message, or "wrapped", so MUAs don't automatically display the HTML (which could verify the email address via a web bug or use an IE exploit to install malicious code); set this if you use Outlook. Has no effect when used with Amavis.

```
report_contact helpdesk@tux-industries.com
```

What address to use in the various text templates. Has no effect when used with Amavis.

```
lock_method flock
```

Set to "flock" for performance if you don't use NFS, RTFM otherwise.

4.3.4 Feeding the Bayes DB

Copy a few hundred pieces of ham and spam to two mbox files for training the bayes DB; it will train itself during operation, but that can take a few days, we'll give it better training (since these are human-selected messages) and get better results faster.

SpamAssassin needs at least 200 pieces of spam and ham before it starts using the Bayes DB.

There's a few such files on <http://cms-machine/>, use ham-small.mbox and spam-small.mbox for reasons of speed. When you set up a real server, try to get a good sample of real mail traffic for training, since this will heavily influence the quality of the result (and therefore, the acceptance from the users).

Run

4 Anti-Spam Configuration

```
su amavis -c 'sa-learn --mbox --spam --showdots' < spam.mbox
su amavis -c 'sa-learn --mbox --ham --showdots' < ham.mbox
```

The "su" command is so the files get created with the right owner.

```
# su amavis -c 'sa-learn --ham --mbox --showdots' < ham-small.mbox
.....
Learned from 297 message(s) (302 message(s) examined).
```

The difference (302 vs 297) is because SpamAssassin doesn't learn from all messages; there may be duplicates (by Message-ID), or messages may be too old or too small.

In the end, it should look like this:

```
# ls -l /var/lib/amavis/.spamassassin/
total 1116
-rw----- 1 amavis amavis 1520 Jan 23 03:15 bayes.mutex
-rw----- 1 amavis amavis 86016 Jan 23 03:15 bayes_seen
-rw----- 1 amavis amavis 1314816 Jan 23 03:15 bayes_toks
# sa-learn --dump magic
0.000  0          3  0 non-token data: bayes db version
0.000  0        299  0 non-token data: nspam
0.000  0        297  0 non-token data: nham
0.000  0       47549  0 non-token data: ntokens
0.000  0   1094658931  0 non-token data: oldest atime
0.000  0   1106314819  0 non-token data: newest atime
0.000  0          0  0 non-token data: last journal sync atime
0.000  0          0  0 non-token data: last expiry atime
```

4 Anti-Spam Configuration

```
0.000 0 0 0 non-token data: last expire atime delta
0.000 0 0 0 non-token data: last expire reduction count
```

"`bayes.mutex`" is a lock file; "`bayes_seen`" contains the Message-IDs of the messages trained (and auto-learned), and "`bayes_toks`" has all the token data.

"`nspam`" and "`nham`" are the number of learned spams and hams, respectively; "`ntokens`" is the number of tokens (words) in the DB, and the "`atime`" values are related to the expiry of the DB, which is the process to keep the size below the limit (`bayes_expiry_max_db_size`).

4.3.5 Testing SpamAssassin

Save a single piece of spam to a file, in mbox format (e.g. "save" from mutt); run "`spamassassin -t < file | less`" and look at the output. Try this with a few messages and make sure there's `RCVD_IN_DNSBL` and `BAYES_nn` in at least some of the X-Spam-Status headers, i.e. that DNS lookups and the bayes DB work.

At the end of the output, it shows a summary:

```
Content analysis details: (3.4 points, 5.0 required)
pts rule name description
-----
0.7 DATE_IN_PAST_12_24 Date: is 12 to 24 hours before Received: date
0.7 SUBJ_ALL_CAPS Subject is all capitals
1.6 PORN_URL_MISC URI: URL uses words/phrases which indicate porn (misc)
0.4 BAYES_60 BODY: Bayesian spam probability is 60 to 80%
[score: 0.7436]
```

There's something wrong here - no DNS tests seem to have happened. Let's look at SA's debug output (the "-D" flag):

4 Anti-Spam Configuration

```
# spamassassin -D -t < file >/dev/null
```

```
[...]
```

```
debug: failed to load Net::DNS::Resolver: Can't locate Net/DNS.pm in @INC
(@INC contains: /usr/share/perl5 /etc/perl /usr/local/lib/perl/5.8.4 /usr/local/share
/usr/lib/perl5 /usr/lib/perl/5.8 /usr/share/perl/5.8 /usr/local/lib/site_perl)
at /usr/share/perl5/Mail/SpamAssassin/Plugin/URIDNSBL.pm line 113.
```

```
[...]
```

Seems I didn't follow my own instructions and forgot the Net::DNS perl module; so
"apt-get install libnet-dns-perl libmail-spf-query-perl" and the result looks
much better:

```
Content analysis details: (18.0 points, 5.0 required)
```

```
pts rule name description
```

```
-----
0.7 DATE_IN_PAST_12_24 Date: is 12 to 24 hours before Received: date
0.7 SUBJ_ALL_CAPS Subject is all capitals
1.6 PORN_URL_MISC URI: URL uses words/phrases which indicate porn (misc)
0.4 BAYES_60 BODY: Bayesian spam probability is 60 to 80%
[score: 0.7436]
3.8 RCVD_IN_DSBL RBL: Received via a relay in list.dsbl.org
[<http://dsbl.org/listing?61.143.199.38>]
1.2 RCVD_IN_BL_SPAMCOP_NET RBL: Received via a relay in bl.spamcop.net
[Blocked - see <http://www.spamcop.net/bl.shtml?61.143.199.38>]
0.4 RCVD_IN_NJABL_PROXY RBL: NJABL: sender is an open proxy
[61.143.199.38 listed in combined.njabl.org]
3.1 RCVD_IN_XBL RBL: Received via a relay in Spamhaus XBL
[61.143.199.38 listed in sbl-xbl.spamhaus.org]
```

4 Anti-Spam Configuration

1.0 URIBL_SBL Contains an URL listed in the SBL blocklist

[URIs: dvd-porno-shop.com]

0.4 URIBL_AB_SURBL Contains an URL listed in the AB SURBL blocklist

[URIs: dvd-porno-shop.com]

1.5 URIBL_WS_SURBL Contains an URL listed in the WS SURBL blocklist

[URIs: dvd-porno-shop.com]

3.2 URIBL_OB_SURBL Contains an URL listed in the OB SURBL blocklist

[URIs: dvd-porno-shop.com]

We went from 3.4 points to 18.0, i.e. from "not detected" to "very high probability of spam".

4.4 Configuring Amavis for SpamAssassin

Copy and uncomment the `bypass_spam_checks_maps` variable from `/etc/amavis/conf.d/15-content` to `50-user`.

Insert the following in `/etc/amavis/conf.d/50-user`:

```
$sa_local_tests_only = 0;
$sa_mail_body_size_limit = 200*1024;
$sa_tag_level_deflt = -1000;
$sa_tag2_level_deflt = 5.0;
$sa_kill_level_deflt = 5.0;
$final_spam_destiny = D_PASS;
```

Explanation:

```
$sa_local_tests_only = 0;
```

4 Anti-Spam Configuration

Enable the DNS-based tests.

```
$sa_mail_body_size_limit = 200*1024;
```

Spam is usually below 200kB, and SA can take up a lot of RAM for big mails.

```
$sa_tag_level_deflt = -1000;
```

```
$sa_tag2_level_deflt = 5.0;
```

```
$sa_kill_level_deflt = 5.0;
```

We want X-Spam-headers in all messages (first line), "***SPAM***" in the subject from 5 points on.

```
$final_spam_destiny = D_PASS;
```

D_PASS means tagged spam goes to the recipient, who can filter it into a folder. D_DISCARD drops the message (meaning nobody will detect false positives!), and D_BOUNCE would harass the innocent victims whose mail addresses were stolen for this spam run.

Finally, add your domain to `/etc/amavisd/conf.d/50-user`:

```
@local_domains_acl = ( "tux-industries.lu" );
```

There's many more Amavis features, but these are the basics to get going, and the Amavis config file is 90% comments explaining what all the options mean.

Run `/etc/init.d/amavis reload` to apply the new settings.

4 Anti-Spam Configuration

4.4.1 Testing

Since all mail through our server will be scanned, we just need to send ourselves a piece of mail; start with a simple test to see if there's any signs of SpamAssassin in the headers:

```
date | mail -s test user
```

and look at it with "mutt" - the "h" keys shows all headers and should reveal something like this:

```
X-Virus-Scanned: Debian amavisd-new at ubuntu.tahina.priv.at
X-Spam-Status: No, score=3.716 required=5 tests=[AWL=1.114, NO_DNS_FOR_FROM=2.603, NO_RELAYS=-0.001]
X-Spam-Score: 3.716
X-Spam-Level: ***
```

This means SpamAssassin is being used by Amavis. So let's try with our previously used sample:

```
/usr/sbin/sendmail user < file
```

and look at the result in mutt, again:

```
X-Virus-Scanned: Debian amavisd-new at ubuntu.tahina.priv.at
X-Spam-Status: Yes, score=21.1 required=5 tests=BAYES_99, DATE_IN_PAST_12_24, NO_DNS_FOR_FROM, PORN_URL_MISC, RCVD_IN_RBL, RCVD_IN_XBL, SUBJ_ALL_CAPS, URIBL_AB_SURBL, URIBL_OB_SURBL, URIBL_SBL, URIBL_WS_SURBL
X-Spam-Level: *****
X-Spam-Flag: YES
```

4.5 Advanced issues

Or: "Issues I will only mention in passing because time will run out well before the basics are done"

4.5.1 Whitelisting/Blacklisting in SpamAssassin

You may be getting newsletters which you actually subscribed to tagged as spam; or you may get spammed from a constant From: address and SA doesn't detect it. The solution is `"whitelist_from add@ress"` or `"blacklist_from add@ress"` in `/etc/spamassassin/local.cf`; don't forget `"/etc/init.d/amavis restart"` afterwards.

If you are subscribed to a mailinglist which gets messages tagged as spam (e.g. a ML for discussing spam issues), use Amavis' `"whitelist_sender"`; there's already a long list of addresses in the default `amavisd.conf` (search for `"nobody@cert.org"`), make sure you use the envelope sender address.

4.5.2 Performance Issues

The maximum amount of messages/time SpamAssassin can handle is limited by the scan time per message; this is usually around 5-10 seconds for SpamAssassin, plus the virus scanning overhead. Our settings limit amavis to 2 concurrent connections, both in `/etc/postfix/master.cf`

```
smtp-amavis unix - - - - 2 smtp
```

and in the `amavisd` defaults

```
$max_servers = 2;
```

If that is too low (i.e. your mail queue gets big during high activity periods), you need more concurrent connections. For SA processing the limiting factor is RAM (assuming you're running on a CPU dating from this millenium), roughly 20-30MB per concurrent process, and the on-disk size of the databases, so those can stay cached in RAM (`"du -sh /var/lib/amavis/.spamassassin"` gives an estimate for this value).

4 Anti-Spam Configuration

Increase both of these values depending on the amount of RAM in your machine; I run 5 concurrent processes on 256 MB mail gateways for a customer which do nothing but spam checking.

4.5.3 Getting Feedback to SpamAssassin

The efficiency of the Bayesian Filter depends heavily on the feedback your give - the auto-learning feature is nice, but can only detect extreme cases, so the fine-tuning is left to the user.

Basically, what you need to do is to feed back at false positives and false negatives using "sa-learn --ham" or "sa-learn --spam". Doing this the manual way can get boring quite fast, so set up a pair of aliases in `/etc/aliases`:

```
secret+spam: amavis+spam
secret+ham: amavis+ham
```

(The "secret" part is so that spammers can't easily guess the address and send their spam to the ham training address).

Create `.forward` files in `/var/lib/spamassassin`

```
# ls -la
[...]
-rw-rw-r-- 1 amavis amavis 40 Mar 2 2004 .forward+ham
-rw-rw-r-- 1 amavis amavis 41 Mar 2 2004 .forward+spam
[...]
# cat .forward+ham
"|/usr/bin/sa-learn --ham -p /dev/null"
# cat .forward+spam
"|/usr/bin/sa-learn --spam -p /dev/null"
```

4 *Anti-Spam Configuration*

What this does is to use Postfix' `"recipient_delimiter"` feature to create two "sub-aliases" for the amavis account, which then get handled by Postfix according to the `".forward+extension"` files; those feed the messages to sa-learn with the correct parameters.

You then can use the "bounce" functionality (and only that!) of your MUA to send a 1:1 copy of the interesting messages to these accounts; a pair of keybindings (I use "H" for "learn as ham" and "S" for "learn as spam") makes this very easy to use.

If you happen to use a MUA which doesn't allow bouncing, you can forward the messages as MIME attachments and need to write a little script which extracts the attachment and feeds that to sa-learn... Or a script which extracts multiple of those attachments, sanitizes them, and feeds them to sa-learn, so Outlook users can drag-and-drop their false positives/false negatives into a new message (as attachments) and send this to the learning aliases; you get the idea.