

Postfix in Practical Scenarios

Flexible Postman

The Postfix mail server continues to gain popularity, above all as it is simple to modify and configure. No matter whether you use Postfix as a dialup host, in a DMZ, or as a simple relay, we will be looking at typical scenarios in production environments. **BY PEER HEINLEIN**



www.photocase.de

An email server is a standard piece of equipment for enterprises. Email communication is rapidly becoming the preferred method of exchanging messages. Even government offices are catching on, with mail servers rapidly gaining a foothold.

However, mail servers are a complex subject and beginners often have no real understanding of a mail server's potential. Reason enough to look at the Postfix Mail Transfer Agent (MTA) [1] in a few application scenarios, especially as it is known for its simplicity and security.

After completing an installation to put a Linux distribution on the designated Postfix machine, the steps for installing

Postfix using your distro's package manager should not prove too difficult. Suse users can call `yast -i postfix`, and Debian users `apt-get install postfix`. If this works as expected, you should now have a `/etc/postfix/main.cf` file. This is the major config file for your postmaster.

It is worthwhile taking a look at this file before we start modifying settings. The `main.cf` file is well-documented by most distributions. It is full of sample configurations, which are tagged as comments (`#`). If a parameter is disabled, Postfix will always have a useful default handy. Some distributions add values of their own at the end of `main.cf`, allowing admins to locate the most important parameters at a glance. The downside is that your modifications might be overwritten by entries further down in the file. The last value applied wins. The `postconf` program provides an alternative to manual editing (see the "Overview of Postfix Tools" box) which helps with the entering of new parameters.

The first time you run Postfix on a machine, you need to define a few basic settings for the hostname and network interface (see the "Postfix Basic Configu-

ration" box). After completing this step, and assuming that your tests have been successful, we can now turn to a few typical applications.

Scenario 1: Plain Vanilla

Our simplest case assumes that Postfix is running in a data center or on a corporation network with a static, routable IP address. In this case, the mail server should work after completing the basic installation steps, accepting email messages for its domain, and allowing clients on the local network to send messages to other mail addresses. A server like this is extensible. You can add spam and antivirus protection (see Box "Virus and Spam Protection with Amavis") or provide a mail relay for customers.

The most important question with a mail server is whether the server accepts mail or not. A normal mail server accepts email from anywhere in a network, assuming that the messages are destined for a mail address for which the server is responsible. Only users are allowed to contact the server to send mail to the Internet, and the server will deliver mail all over the world.

THE AUTHOR

Peer Heinlein has been an Internet Service Provider since 1992. Besides his book on Postfix, Peer has published two other books on "LPIC-1" and the "Snort" Intrusion Detection System with Open Source Press. Peer's company, www.heinlein-support.de, educates and trains administrators, and provides consulting and support services all over Europe.



It is not the server's responsibility to deliver mail from arbitrary addresses and users to arbitrary destinations.

If we apply this to a configuration, these distinctions are made by setting the `smtpd_recipient_restrictions` parameter. We can simply use the default:

```
smtpd_recipient_restrictions =
  permit_mynetworks,
  reject_unauth_destination
```

Relaying is permitted for any IP addresses in `$mynetworks`, but otherwise Postfix will refuse any mail not destined for it; in other words, the server needs to be the final destination.

Scenario 2: As a Backup MX

In contrast to Web servers, it is quite easy to designate a backup mail server that accepts any messages posted to your main mail server while it is down, and puts the messages into temporary storage. In smaller networks it is often not worth setting up a Backup MX; other mail servers along the path to your main mail server will also keep mail for delivery later, when your Postfix server is back up and running. However, there are a few situations where a Backup MX can still prove useful. Knowing where messages will be stored in case of down time gives admins more leeway; at least you

have some control over the backup server, and can patch up a workaround that will keep your business running.

There are some environments where another type of mail server (such as Exchange, Scenario 3) stores messages temporarily, allowing the Backup MX to deliver incoming messages without delay, despite any problems with the main server. Here, the two servers are peers, and fail over transparently.

Backup for Reliability

It is easy to set up a backup mail server. All you need is a second Linux machine with Postfix installed and the minimal configuration shown in the "Postfix Basic Configuration" box. The Backup MX needs a suitable MX entry in your DNS. Select a higher MX value to reflect a lower priority and allow other mail servers to contact your main mail server (MX 10) first. If this fails, they will choose the backout machine (MX 20).

The machine needs to know that it is a backup machine, and expected to store messages. To configure this, look for the line with `smtpd_recipient_restrictions` in `main.cf`, and add an entry for `permit_`

Listing 1: `ip-up.local`

```
01 # Permit DNS queries:
02 /usr/sbin/postconf -e
   "disable_dns_lookups = no"
03 # Allow immediate mail
   delivery:
04 /usr/sbin/postconf -e
   "defer_transports = "
05 # Reload configuration:
06 /usr/sbin/postfix reload
07 # Process mail queue:
08 /usr/sbin/postfix flush
```

`mx_backup`, paying attention to the order of the entries:

```
smtpd_recipient_restrictions =
  permit_mynetworks,
  permit_mx_backup,
  reject_unauth_destination
```

You can use any other mail server as a backup machine. Two active servers could back each other up if an error condition occurs. Smaller companies should look to use the provider's mail server or contact a partner to organize a mutual backup solution.

Overview of Postfix Tools

postfix flush: Immediately attempts to deliver any mail waiting in the mail queue.

mailq: Lists all the messages in the mail queue with their status.

postconf parameter: Shows the current settings for this parameter.

postconf -e "parameter=value": Sets a parameter in `main.cf` to the given value. This allows for convenient editing of `main.cf` in shell scripts, and other applications.

postconf -n: Displays all the parameters in `main.cf` which are not default values. This is very useful for troubleshooting.

postsuper -r: Initiates a "requeue" for all messages; that is it puts the messages back in the queue, resolves all DNS data, and checks any new address aliases (such as the ones in the `virtual` table).

postmap file: Converts a lookup table from Postfix (for example `virtual` or `transport` to a Postfix database format.

Postfix Basic Configuration

A few simple steps are all it takes to configure a newly installed Postfix machine to allow it to assume the role of a normal mail server on a LAN. You will typically need to modify a few parameters in `main.cf`. Ensure that the setup program provided by your distribution has entered useful values into `/etc/postfix/main.cf`. The first place to look is the end of the file. This is where setup programs tend to add critical parameters, which would overwrite any changes you made further up in the file.

Postfix can use variables. If you define `$myhostname` somewhere in the file, you can use the variable at other positions.

`$myhostname` has to be a hostname that DNS can fully resolve to your server name, including the domain (FQDN).

`$mydomain` is not normally set. Postfix checks `$myhostname` for the domain in this case.

`$mydestination` tells Postfix the mail addresses for which it is the final destination, that is, the domains for which it should accept messages for delivery to local accounts – this is a critical parameter. Besides `localhost` and `$myhostname`, `$mydo-`

`main` is typically added here, as messages are normally addressed to `user@example.com` rather than to `user@mail.example.com`.

\$mynetworks: Includes the trusted IP addresses that are permitted to send messages to external addresses (relaying). This is typically `127.0.0.0/8` and the local subnet served by the mail server.

\$mynetworks_style: Alternatively, Postfix allows you to use this parameter to define the network segments allowed to relay. The server refers to its own IP address to discover the networks, allowing for a flexible configuration that can be transferred to other servers.

```
$mynetworks_style = class
```

The value for `class` reflects the network class, A/B/C, of the network on which the server resides. The other alternatives are `subnet` (refers to the server's subnet and is normally the best choice), and `host` (for the server IP address).

Postfix has to reload the configuration from `main.cf` when you change it. Depending on the distribution you use, type either `rcpostfix reload` or `/etc/init.d/postfix reload` to do this.

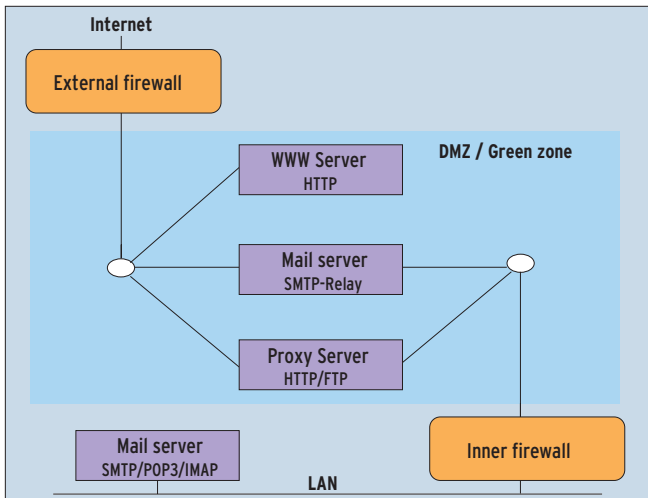


Figure 1: A relay server in the DMZ protects the mail server proper where the mailboxes reside.

After rebooting, Postfix will accept messages although they are not addressed to its *final_destination* domain. The server recognizes that there is another server with a lower MX value (a higher priority) and not try to deliver any messages to local accounts. The server will continually attempt to contact the

main mail server, and deliver the messages to that server when it comes back up, as if nothing had happened. To support critical Groupware functions on a LAN are just one example; Exchange is not renowned for its prowess in the battle against spam, viruses, and hackers. To mitigate the risk, some admins set up a Postfix server at the border to their networks, allowing Postfix to handle Internet-based mail exchanges, and pass-

ing incoming mail to the Exchange server on the local network.

This removes the need to expose the Exchange server on the Internet. At the same time, Postfix can show its strength in the fight against the issue of malevolent messages.

Scenario 3: Relay in the DMZ

There are good reasons for using Postfix as a mail relay, and storing messages on another mail server. Environments that use MS Exchange

Do-It-Yourself Routing

You can use the same setup if you need to deploy a mail server in your DMZ to relay incoming mail to another mail server in your secure LAN. It does not make any difference if this is an Exchange server or another Postfix server. Figure 1 shows this scenario.

Listing 2: ip-down.local

```
01 # Disable DNS queries:
02 /usr/sbin/postconf -e
    "disable_dns_lookups = yes"
03 # Stop SMTP transmissions:
04 /usr/sbin/postconf -e
    "defer_transports = smtp"
05 # Reload configuration.
06 /usr/sbin/postfix reload
```

Sharp Zaurus SL-C750/C860

Now Available In The UK

"The keyboard is surprisingly good for such a little device."



"The MP3 sound quality is excellent though the headphones."

Amazing 640 x 480 Screen, Full Keyboard, Word/Excel Compatible, Presentation, PIM, Web Browser, Video, MP3, PDF, Konsole, SSH, Java ...

Linux In Your ShirtPocket !

"Wonderful, bright, clear, sharp, and colorful display. The resolution is extremely good and looking at photos on the C760 is a real joy."

*Quotes From "The Gadgeteer", "BargainPDA" and "What Mobile" Reviews. See Website For Further Details & Links.

"...if you are a power user, who isn't afraid of being a little adventurous, this is the ultimate PDA to own."

w@mob
EDITORS CHOICE







ShirtPocket

L I M I T E D

Tel: 0116 2201272
 Email: salesdept@shirtpocket.co.uk
<http://www.shirtpocket.co.uk>

Just a few steps are required to modify the Postfix configuration. Several options are open to the admin, but the easiest way to go is to use a so-called transport table. The `/etc/postfix/transport` file tells Postfix to deliver mail based on a pre-configured IP address, rather than using the MX records in the DNS. If the local mail server on your LAN has an IP of 192.168.1.99, the following is fine:

```
example.com smtp:[192.168.1.99]
```

The next thing you need to do is call `postmap /etc/postfix/transport` to tell Postfix to convert the file to its native database format.

The rest of the Internet does not know about this setup, of course. Other mail servers will use the MX record to identify the DMZ mail server as their target, and send mail to that server. Postfix accepts this incoming mail, as the mail address includes the MX record for the domain and `permit_mx_backup` is specified in the `smtpd_recipient_restrictions`:

```
smtpd_recipient_restrictions 2
= permit_mynetworks,
  permit_mx_backup,
  reject_unauth_destination
```

The server will not look for local user accounts, but instead honor the `transport` record, sending the messages for this domain to the 192.168.1.99 IP.

Instead of using an IP, you could use a DNS hostname, but the square brackets are mandatory. The brackets have a special meaning and tell the server to relay the messages to the named host (identified by querying the DNS A record) and not to the authoritative mail server for the domain (identified by querying the DNS MX record – as this could be the relay itself).

Admins have to allow mail to travel in the opposite direction too, allowing messages to travel from the LAN-based mail server at 192.168.1.99 to the Postfix relay in the DMZ. The settings will depend on the LAN-based mail server's software. You will typically find a parameter called

“Relayhost” (or sometimes Smarthost) that handles this. Any messages with external addresses are sent to this host for forwarding; this is the counterpart of the `transport` table if you like. If the LAN-based mail server also runs Postfix, you can add the following entry to the `main.cf` file:

```
relayhost=[mail.example.com]
```

Scenario 4: Without a Permanent Connection

The scenarios we have looked at so far, assumed a leased line with static IPs to facilitate routing to the servers. Not everyone has a Class C network with a 2Mbit or even 34Mbit leased line, or a server of their own at the data center. Users in home or small business networks, or in areas without broadband, may still need to resort to a good old ISDN card and dialup access to an Internet-by-Call provider. With this kind of connection you can expect to be billed for access and connection time.

Virus and Spam Protection with Amavis

Effective spam protection is often a complex matter, and a whole armory of defensive methods may be required to provide it. The Amavis Project [8], which started off life as an antivirus gateway for mail servers, allows you to integrate Spam Assassin – an extremely complex, and intelligent software that can filter your mail on spam criteria.

Although you can install Spam Assassin as a stand-alone software without Amavis, the dual approach is preferable. Amavis is quick, and easy to install; it performs well, and handles the details of Postfix and Spam Assassin integration really neatly. Also, if you install the antivirus program, you get virus checking into the bargain.

Both programs are included by most of today's modern distros. As there are many versions around, you should always check `amavisd-new` for the latest version.

After installing the packages, you should have a `amavisd-new` daemon running in the background. It opens a lightweight SMTP daemon on port 10024, and uses a virus scanner, and Spam Assassin to check emails on this port. Assuming it does not find anything, the daemon uses SMTP to forward the messages to Postfix via port 10025. Figure 7 shows the setup.

After calling `/etc/init.d/amavis` to launch Amavis, it makes sense to check the logfile to see if everything is running as it should be.

Depending on your distribution, Amavis will log the results in `/var/log/mail` or `/var/spool/amavis`.

There are just two steps left with Postfix. An entry of

```
content_filter=[127.0.0.1]:10024
```

in `main.cf` tells the server to send new email messages to `amavis-new`.

At the same time, Postfix has to open the local port 10025 to pick up checked messages. The `master.cf` file handles this. A suitable entry may already be pre-configured:

```
localhost:10025 inet n - n2
- - smtpd -o content_filter=
```

The `-o content_filter=` parameter tells Postfix not to forward checked mail to `amavisd-new`, and prevents an infinite loop. Again, you need to restart Postfix to apply the changes.

Some distributions configure Spam Assassin to allow all mail to pass, even messages that have been identified as spam. Check out the documentation and readmes with your distribution for more details (look in `/usr/share/doc/packages/amavisd-new` for Suse). You can also use the test messages in this directory to ensure that spam and virus detection is working as designed. `/etc/amavisd.conf` is the place for any fine tuning you might want to perform.

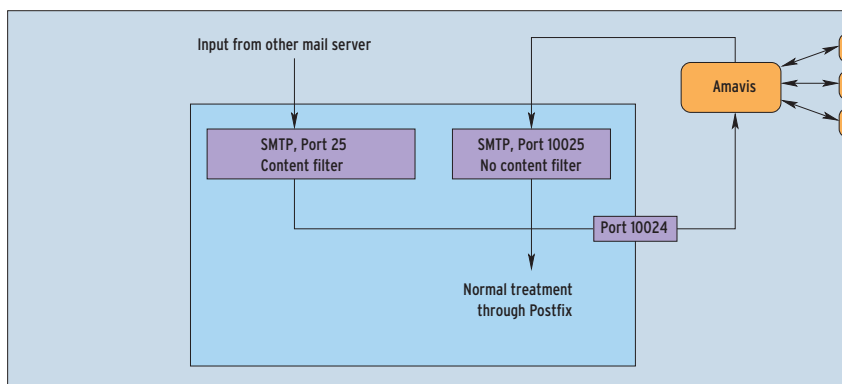


Figure 7: Every message goes through an additional check performed by Amavis.

Saving Money

Postfix does not know if the Internet connection is up or down. It simply attempts to query DNS for the target server, and deliver any messages it has in its queues. If your ISDN connection is configured to dial on demand, every email one of your users transmits would cause Postfix to dial up an Internet connection. You can imagine what this would cost in a busy office.

The idea is to configure Postfix to queue mail locally while the connection is down, and start handling messages as soon as the connection goes up (possibly because an Internet user has opened a dial-on-demand connection).

If the parameter `defer_transports = smtp` is set in `main.cf`, Postfix will not immediately attempt to deliver messages to external addresses (which would mean dialing up the provider), but first queue them locally. After dialing up the provider, you can use a script to set this parameter to zero, reload the configuration and call `postfix flush` to flush the messages waiting in the queue. `postconf -e` provides a convenient way of modifying `main.cf` from the console (see Box “Overview of Postfix Tools”).

The `/etc/ppp/ip-up` and `/etc/ppp/ip-down` that most distributions run after establishing a connection and disconnecting can be used to automate this process. For some Linux variants, you

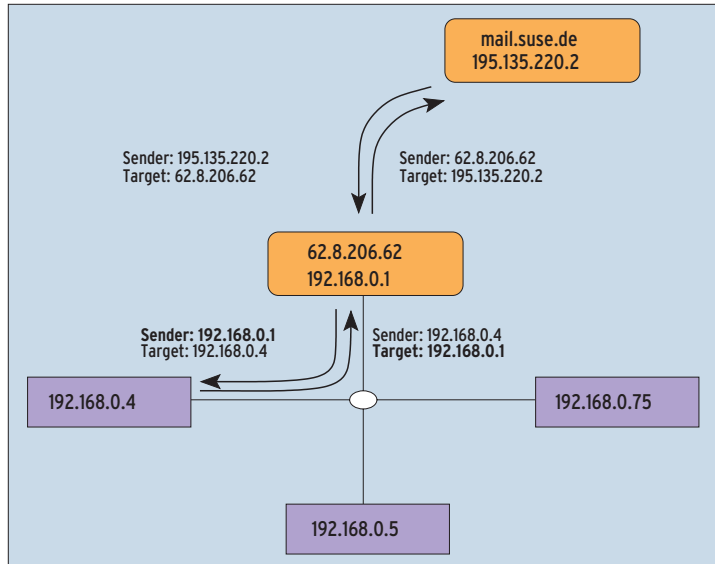


Figure 2: If port forwarding is used, Postfix has to know that the local router address is not trustworthy.

will need to add the changes in Listings 1 and 2 to these files; recent versions provide the `ip-up.local` and `ip-down.local` files for local modifications, however.

In contrast to sendmail, Postfix cannot check whether the queued messages have been delivered successfully as the connection is closed by a dial-on-demand timeout if not in use.

Modem, ISDN, and DSL connections in multi-user environments can all benefit from traffic shaping to prevent parallel mail transmissions from impacting the Web surfer's experience [2].

Scenario 5: Postfix and Port Forwarding/NAT

It is common practice to use an IP such as 192.168.0.0/24 for local networks and Network Address Translation (NAT) or

masquerading to map this private address to the routable IP assigned to the dialup host or router. Postfix has no trouble transmitting messages in this kind of environment, but incoming mail is another matter. Today's routers typically have a port forwarding feature, which is often referred to as “Destination NAT”. The router opens up a pre-configured port and forwards any incoming connections for this port to the servers running on the local 192.168.*.* network.

Although this is okay for many services and will work reliably with your Web server at home, for example, it can cause a mail server quite serious trouble. Here, the router opens up a TCP connection to the mail server, but from Postfix's point of view, the connection is from the host at 192.168.0.1 and not from a host out on the Internet.

The IP assigned to the router is typically within the subnet defined in `$mynetworks`, and thus a trusted IP which is allowed to relay to external addresses. Thanks to port forwarding, a home mail server can quickly become an open relay for spammers and cause the owner no end of surprises. An entry for the router IP in the `proxy_interfaces` variable will prevent this:

```
proxy_interfaces=[ip.of.router]
```



Figure 3: Services like DynDNS are popular with users of low-budget DSL accounts. They cause Postfix a few problems.

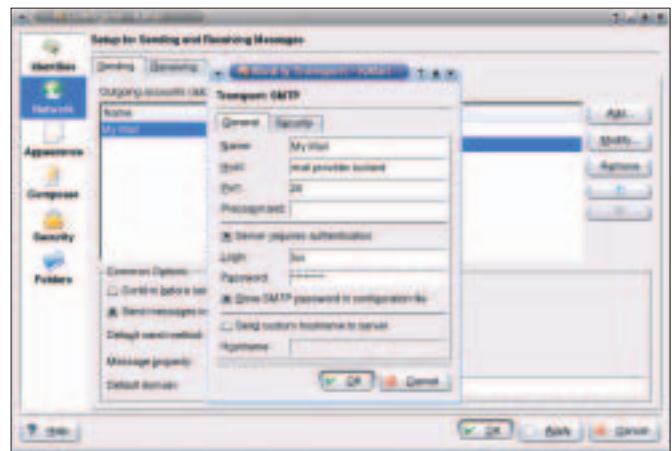


Figure 4: Two additional options in the software configuration, tell KMail to use SMTP-Auth to authenticate.

This tells Postfix to handle any connections to these IPs as not belonging to *\$mynetworks*. Also, the server knows that it has to treat these IPs as its own for DNS queries, to avoid relaying messages to itself, from Postfix to the router, and via port forwarding back to Postfix.

Scenario 6: Using DynDNS

DynDNS and similar services offer free subdomains for your hosts, and tools that flexibly change the IP address registered with DNS. Whenever a client dials up the Internet, it can automatically set its own DNS entry to the dynamic IP address assigned by the provider. Of course, this is a fantastic solution for home PCs or scratch servers, as you can use the same name for access despite IP address changes.

Using a DynDNS address to handle your mail might sound like an attractive proposition, but you need to be aware of how mail servers work. An MTA looking to deliver a message will first query DNS for the MX or A records in the domain to discover the host it needs to send the message to.

Of course, this would work fine with DynDNS. The going starts to get more difficult if your computer happens to be down for a few hours. Other MTAs will store your mail as usual, but will send it to the former IP when it becomes available again, instead of querying DNS before delivery. Even if the target mail server had a new IP, and the DynDNS

entry pointed at the right address, messages stored on other servers while the server was down would never reach it.

In an ideal case, these messages would be returned to the sender as undeliverable a few hours, or days, later. If you are unlucky, the IP might be assigned to another MTA, which might accept the delivery, putting your secrets in the hands of a total stranger.

DynDNS-based mail collection has to be regarded as a compromise that you should definitely not rely on.

Scenario 7: DynDNS as a Relay

There is an answer to the DynDNS issue, assuming you have another mail server with a static IP. If you assign highest priority (the lowest MX value) to the MX record in your DNS, MTAs will deliver messages to the relay first. An entry in the *transport* table tells Postfix to deliver messages to the DynDNS host. This scenario is more or less the same as Scenario 3 so far. Of course, you need square brackets for the */etc/postfix/transport* entry again:

```
test.dyndns.org smtp:➤
[test.dyndns.org]
```

Postfix will relay any messages for *...@test.dyndns.org* to the DynDNS host.

If this address is unavailable for a longer period of time, the relay would queue the messages and not resolve the IP number of the target host after doing so. As a workaround, you could call

```
postsuper -r ALL
```

at regular intervals – a cronjob every hour should do the trick. This tells Postfix to requeue any outstanding messages. At the same time, the mail server would resolve the DNS data, thus honoring any changes to the DynDNS address that have occurred in the meantime, and allowing it to deliver the messages to the right address.

This setup works with one or two exceptions, but it is far from perfect. The fact that requeuing only occurs once an hour could delay delivery by up to an hour. In this time, a new MTA might become available at the old IP, and your mail would end up in some stranger's mailbox. This kind of issue assumes that the DynDNS host is down for a longer period of time and would only affect users with modems or ISDN, rather than DSL, which is available 24x7 apart from a few seconds per day.

Of course, there is always the question as to the usefulness of a DynDNS solution that needs a genuine mail server as a relay to queue its messages. On the other hand, in an environment with multiple mailboxes the DynDNS setup shown here is less trouble than a complex fetchmail configuration on the DynDNS host that picks up the messages from the relay's mailboxes, if you need to have your mail stored in mailboxes on the DynDNS host.

Scenario 8: Postfix and SMTP Auth

The previous scenarios allow users to send mail via a Postfix mail server, if they are in *\$mynetworks*. Things are



Figure 5: Select MD5 encryption to secure password transmissions.

Critical SASL Parameters

```
smtpd_sasl_auth_enable = yes
```

enables or disables SMTP-Auth *no*.

```
smtpd_sasl_security_options =
noanonymous, noplain
```

noanonymous prevents anonymous logins, *noplain* prevents clients from transmitting the SMTP-Auth password in the clear, which is what the *PLAIN* and *LOGIN* authentication methods do. This parameter forces the client to encrypt the password to prevent it from being sniffed. Users need to select secure settings in their software (for example *CRAM-MD5* or *DIGEST-MD5*, Figure 5). This is not required if logins will be taking place over secure SSL/TLS connections.

```
smtpd_sasl_local_domain = mail
```

This parameter stores the value used as a *realm* by *sasl*. Realms are basically used to authenticate users from multiple (virtual) server domains, however, both Postfix and many clients can only handle a single SASL domain.

```
broken_sasl_auth_clients = yes
```

Some older clients, for example Microsoft Outlook Express 4.x, expect the mail server authentication in the following format, *AUTH=LOGIN...*, although this is more typically *AUTH LOGIN...*. Setting this value to *yes* tells Postfix to output the *AUTH* banner twice using each of these formats.

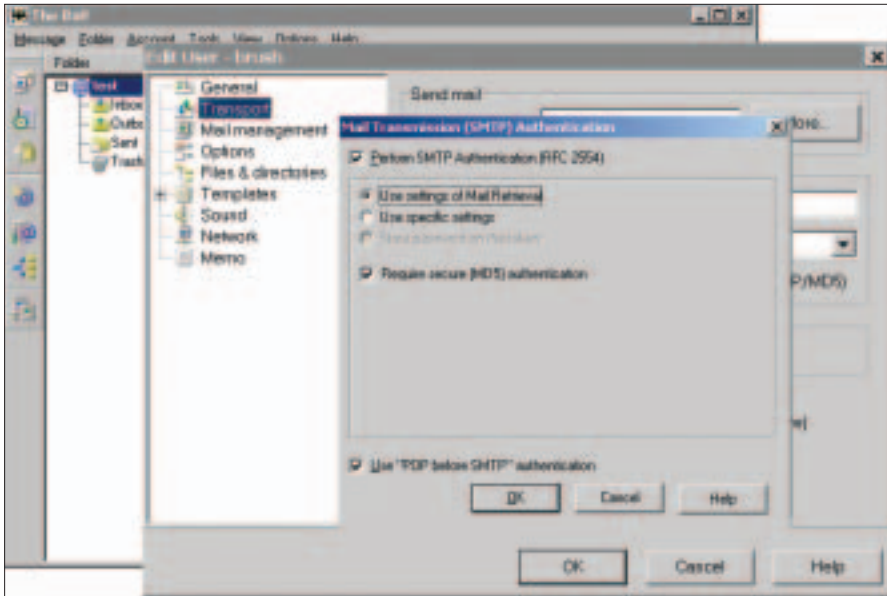


Figure 6: No matter whether you have SMTP-Auth with MD5, or SMTP-After-POP the Windows Client, “The Bat”, can handle both.

more complicated for users from outside of the local IP range, attempting to contact the server from an arbitrary host on the Internet in order to send email messages. In other words, we need to think about authentication.

A Mail Address is Not Proof of ID

The idea of using mail addresses for authentication is a bad one. Mail addresses can be chosen arbitrarily, and spammers often do just that. Some spammers use the mail addresses of the mail servers they exploit to convince these servers to relay the unsolicited mail.

If you are unable to identify clients by static IPs or cryptographic techniques, your only option is to use password protection. In contrast to the POP3 or IMAP protocols, which are used to collect email messages, SMTP did not originally specify a mechanism for checking user names or passwords for identification purposes. This gap was closed some time later, and modern MTAs and mail clients all support SMTP authentication (SMTP-Auth). A client needs to identify itself when it requests a mail transfer to be regarded as trustworthy.

User Management with Cyrus SASL

The Cyrus SASL package [3] brings this functionality to Postfix. The package

manages a small login database in `/etc/sasl/db2`, which Postfix can access to validate users.

You can add user entries with the `saslpasswd2` tool, and `sasldblistusers2` lists the existing entries.

Cyrus-SASL can manage multiple hostnames and domains in a single database, allowing multiple user accounts with the same name but for different domains to co-exist peacefully. Cyrus-SASL uses a *realm* concept to distinguish between accounts. You can either use the `-u` domainname parameter to define a specific *realm*, or `saslpasswd2` will simply use the hostnames, `mail` in this case (see Box “Critical SASL Parameters”) to define `smtpd_sasl_local_domain`.

To tell Postfix to allow users who have authenticated via SMTP-Auth to relay, admins need to add an option for `permit_sasl_authenticated` to the `smtpd_recipient_restrictions` list:

```
smtpd_recipient_restrictions= \
  permit_mynetworks,
  permit_sasl_authenticated,
  permit_mx_backup,
  reject_unauth_destination
```

You also need the following parameters in `main.cf`.

```
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = \
  noanonymous, noplaintext
smtpd_sasl_local_domain = mail
broken_sasl_auth_clients = yes
```

Refer to the “Critical SASL Parameters” box for more details on the individual options.

A small script will take care of everything else; transferring the user data to the SASL database. The `-p` parameter tells `saslpasswd2` to read the password from standard input:

```
echo secret | \
  saslpasswd2 -p -c tux
```

Although SMTP-Auth is quite simple to set up, it does have some drawbacks: The postmaster has to get users to enter the required credentials into their client configurations (Figures 4 and 6).

POP before SMTP

The popular POP-before-SMTP method (aka SMTP-after-POP) is an alternative, and it does work with IMAP servers despite the name. The idea is trivial, but it is still quite difficult to configure. After a successful POP3/IMAP login has occurred from a specific machine, you can assume that it is a trustworthy user. The mail server then accepts mail from authenticated email clients for a specific interval (typically 15 minutes).

Listing 3: pop-before-smtp

```
01 ~: # /etc/init.d/pop-before-smtp start
02 ~: # ps ax | grep pop-before-smtp
03 5022 ?      S    0:07 /usr/bin/perl -wT /usr/sbin/pop-before-
  smtp
  --watchlog=/var/log/mail --logto=/var/log/pop-before-smtp
  --daemon=/var/run/pop-before-smtp.pid
04 9367 pts/1  S    0:00 grep pop-before
05 ~: # ls -al /etc/postfix/pop*
06 -rw-r--r--  1 root  root 12288 Oct 8 11:18 /etc/postfix/pop-before-
  smtp.db
```

After configuring the mail software to check the mailbox first, before attempting to send messages, there are no other steps required on the part of the user. Some mail clients use this order by default (KMail for example); others allow you to change the configuration. Older versions of Outlook (Express) are basically incapable of POP-before-SMTP and you will need a few tricks to get them to play ball [4].

This approach has a few drawbacks for postmasters, and it is slightly controversial. For one thing, there is no real way of validating the sender's credentials. A computer that has been validated might be used by multiple users.

For another, there is always a danger of a dynamic IP being assigned to another user within the time slot, and this user might happen to access the mail server that authenticated the previous user (or an attacker might do this on purpose). Still, POP-before-SMTP is useful in many cases, and can be implemented in addition to SMTP-Auth.

The *pop-before-smtp* script [5], which many current distributions include as a ready-to-run package, uses a daemon to check the logfiles of a POP3/IMAP server. When new entries or successful logins occur, the script extracts the user IP and enters the IP into the */etc/postfix/pop-before-smtp* database.

If Postfix decides to accept mail from the authenticated IPs, there is nothing to stop authenticated users from sending mail. When the time slot has elapsed, *pop-before-smtp* removes the IP from the database.

Listing 4: Checking the logfile

```
01 /var/log: # tail pop-before-
02 smtp
03 [...]
04 read ts=Mar 22 11:03:29
05 ip=62.8.206.156
06 read ts=Mar 22 11:17:58
07 ip=217.235.18.66
08 accepted --- not in
09 mynetworks
10 written ok
11 read ts=Mar 22 11:17:59
12 ip=217.235.18.66
13 purging ts=Fri Mar 22 10:21:50
14 2002 ip=217.224.233.74
```

The */etc/pop-before-smtp-conf.pl* config file uses regular expressions to identify and extract the IP from the logfile entries created by POP/IMAP mail servers. The script has entries for many common mail server types.

The init script supplied with the package loads the daemon when you boot your machine; follow the steps required by your distro to enable the script. After restarting, the script should disappear into the background and create the required database (see Listing 3).

In *debug* mode the daemon logs any IPs it has recognized in */var/log/pop-before-smtp* (see Listing 4). It will not create entries for IP numbers that Postfix has added to *\$mynetworks*, or that are already in the database, but simply note that the IP was known. Debug mode is enabled when the *\$debug* variable in the configuration file is set to 1.

If you see an entry for *written ok* in the logfile, you can assume that the script has stored the IP in the database. If this entry is missing, *pop-before-smtp* is trying to tell you that the IP is known from a previous login and authenticated. An

entry for *purging* means that the time slot for this IP has elapsed, and that the IP has been removed from the database.

A quick look at the logfile after the install tells you if login detection is working okay. If *pop-before-smtp* does not recognize any IPs, you should check the regular expression in */etc/pop-before-smtp-conf.pl* for errors.

Now, Postfix needs to know how to evaluate the database to answer the question as to whether a user is allowed to relay. A *smtpd_recipient_restrictions* parameter called *check_client_access* is used to do this. The parameter expects the database for the *pop-before-smtp* script as an argument:

```
smtpd_recipient_restrictions = 2
permit_mynetworks,
permit_mx_backup,
check_sasl_authenticated,
check_client_access hash:2
/etc/postfix/pop-before-smtp,
reject_unauth_destination
```

Before you send your server off into the Wild, you should definitely check the configuration [6]. Databases such as ORDB [7] register open relays, and many servers refuse to accept mail from hosts known to be on these lists. If you get blacklisted in an open relay database, you might have a hard time getting off of the list again. Changing the server IP is often the only option as running an open relay means helping the spammers. ■

Mbox and Maildir Formats

There are two standard formats that mail servers use to store messages. The Mbox format stores all the messages for a specific user in a single large Mbox file—for example in */var/mail/username*. Shell mail programs like *mutt* or *elm* access the Mbox file directly. The Mbox format is simple and quick, but it can be time-consuming to extract and delete individual messages from a large mail file.

Maildir stores each message in a single file and creates a directory for each user, for example in */var/mail/username/...* The advantage that this format provides is its flexible structure, which also supports the use of IMAP servers, which create additional IMAP folders below the Maildir directory, providing the ability to sort messages.

Postfix uses the *mail_spool_directory* parameter to distinguish between Mbox and Maildir. If the pathname stored in this variable ends in a slash, (“/”) Postfix will store the messages in Maildir format:

```
mail_spool_directory = /var/mail/
If the path name ends in a folder name,
without a terminating slash, Postfix will use
the Mbox format instead:
mail_spool_directory = /var/mail
```

INFO

- [1] Postfix: <http://www.postfix.org/>
- [2] Traffic shaping: <http://lartc.org/>
- [3] SASL info and howtos: <http://www.thecabal.org/~devin/postfix/smtp-auth.txt>
- [4] POP-before-SMTP with Outlook: http://www.spinnet.jp/man/pbsmtp/wine/pbs_we_0198.html
- [5] Script and howto for SMTP-after-POP: <http://popbsmtp.sourceforge.net/>
- [6] Very useful relay test: <http://www.abuse.net/relay.html>
- [7] Open Relay Database: <http://www.ordb.org/>
- [8] Amavis: <http://www.amavis.org>

Pictures courtesy of Open Source Press, "Das Postfix-Buch", ISBN 3-937514-04-X, www.opensourcepress.de