

Configuring Postfix to Prevent Exploitation

Self Defense

Spam, spam, spam – the bane of most email inboxes. Whether you are inundated by porn ads, ominous looking gaming offers, premium rate numbers, or purportedly lucrative investment opportunities abroad, the perpetrator normally remains hidden, and attempts to follow pesky bulk mail back to its source often leads to a deadend, or the tracks lead to an innocent mail server, but no further. The problem with this server is that has been configured as an open relay, so it accepts mail from anywhere and relays it to the target mail server (or onwards to an intermediate server).

Although this does not sound particularly evil, and was even a desirable feature during the infancy of the Internet, it makes it easy for spammers to conceal their true identity. At the same time the abundance of open relays ensures that spammers do not need to invest in technology on a scale capable of transmitting thousands or even millions of email messages, or even pay for the traffic this creates. Instead they simply transfer their junk mail, and a long list of targets, to a

Wouldn't it be nice to have an answer to spamming? But at least it's no big deal, to configure your mail server to prevent it from being misused as a relay station by spammers. **BY PEER HEINLEIN**

third-party open relay. A mail server exploited in this way then happily spends hours, or even days and weeks, transmitting gigabytes of spam to the list of targets. The spammer has no transmitting traffic worthy of note, and thus virtually no costs. And risks are low, provided the spammer is smart enough not to leave tracks.

Stop the Accomplices!

The real solution to the spam issue cannot be to install filters for the victims. That would be removing the symptoms but ignoring the cause. It makes more sense to start with the mail servers and first ensure that they are incapable of acting as relays, and thus as the spammers' accomplices. The reasoning behind this is quite simply:

- A normal mail server accepts email from any point in an internet, provided it is addressed to a mail address that the server is responsible for.
- On the other hand, users are allowed to contact the mail server when they want to transmit mail

across the Internet, and the mail server will deliver outgoing mail world wide.

- However, it can never be the mail server's job to accept mail from anywhere and anyone, and to deliver it to an arbitrary target. The mail server is not responsible for mail from arbitrary senders destined to arbitrary targets.

Unfortunately, not all Mail Transport Agents (MTAs) reflect this in their basic configuration – as the administrator you may need to check this, and possibly modify your mail server's settings manually, before installing it onto the live network.

Alternative Postfix

Even discovering how your mail server is set up might entail a detailed investigation of the mail server configuration. And that is one of the reasons that System Administrators are turning to Postfix [1], an increasingly popular MTA which is simple to set up while remaining flexible and secure. Although we will be using

Postfix to demonstrate the configuration steps required to set up a secure mail server, users of other programs, such as Sendmail or Qmail, should be able to derive the configu-



ration steps for their individual programs from the steps shown here.

Step one restricts the server's capability to receive mail to the two case mentioned previously and thus closes a potentially open relay. Following this, we will be looking into exceptions for specific accounts or authentications mechanisms that may be in place. But an open relay should never be allowed onto the network. In other words, before you even start setting up the mail server, you must discover the IP addresses that belong to your own network. The server will be allowed to relay mail to the Internet only for these addresses.

The addresses will usually include the IP address range in your LAN or IP address ranges of any Internet access points if you are a provider. These addresses must now be added to the main configuration file for Postfix, `/etc/postfix/main.cf`:

```
mynetworks = 127.0.0.0/8, 168.100.0.0/24, 127.0.0.0/8
```

Alternatively, Postfix permits more flexible use of the `mynetworks_style` parameter to define the addresses where access is permitted; the server then uses its own IP address to discover the address ranges. This allows for a flexible configuration – easily ported to other servers:

```
mynetworks_style = class
```

`class` corresponds to the class A/B/C network in which the server resides.

You can also use the `subnet` keyword (which corresponds to your server's subnet, and is normally your best bet), and the `host` keyword for your server's IP address. These parameters will have no immediate effect on Postfix. They will be significant later, though, when the server needs to decide whether the MTA is allowed to accept an email message, or required to reject it.

The server makes this decision when the client supplies the target address, as the server refers to this address to decide whether or not the message is addressed to one of its own users. If this is so, the MTA will accept the message independently of its origin.

Restrictive

The `smtpd_recipient_restrictions` keyword is used to apply conditions and policies. The basic configuration is as follows:

```
smtpd_recipient_restrictions=
    permit_mynetworks,
    check_relay_domains,
    reject
```

The keywords `permit_mynetworks` and `check_relay_domains` allow Postfix to

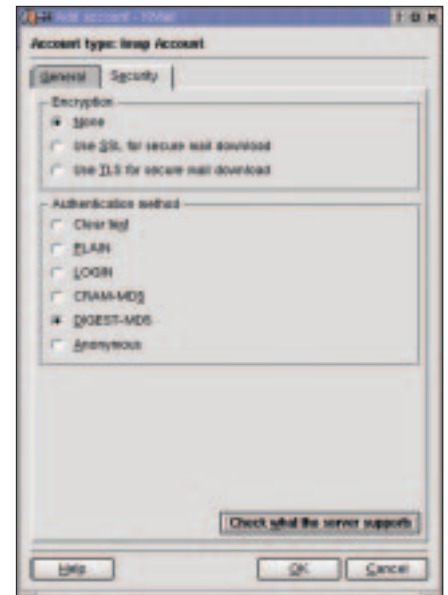


Figure 2: MD5 encryption is the preferred method for transferring passwords securely

accept email messages either when they are addressed to a local user (incoming), or if they originate from a known IP address, that is from a client on the local network (outgoing). If neither case applies, Postfix will *reject* the message.

So far, so good – our server is secure and relay proof. However, practical applications are normally not that simple, and may require your server to relay email messages for your own users, although their origin lies outside of your IP address range (a home office, for example). It looks like we will need to consider an authentication mechanism, and allow the server to perform selective relaying for users who can prove their identity to the server.

Mail Addresses Do Not Prove Identity

Using the sender's email address to authenticate your users is definitely not a good idea. Email addresses can be chosen arbitrarily and are easily spoofed by spammers. Some spammers deliberately use an email address belonging to the mail server they are exploiting, to persuade the mail server to relay their messages.

As simple as it might seem, this method was often successful: Up to a few weeks ago, a major domain hoster in Europe allowed arbitrary mail relays provided the source address belonged (or was spoofed to appear to belong) to

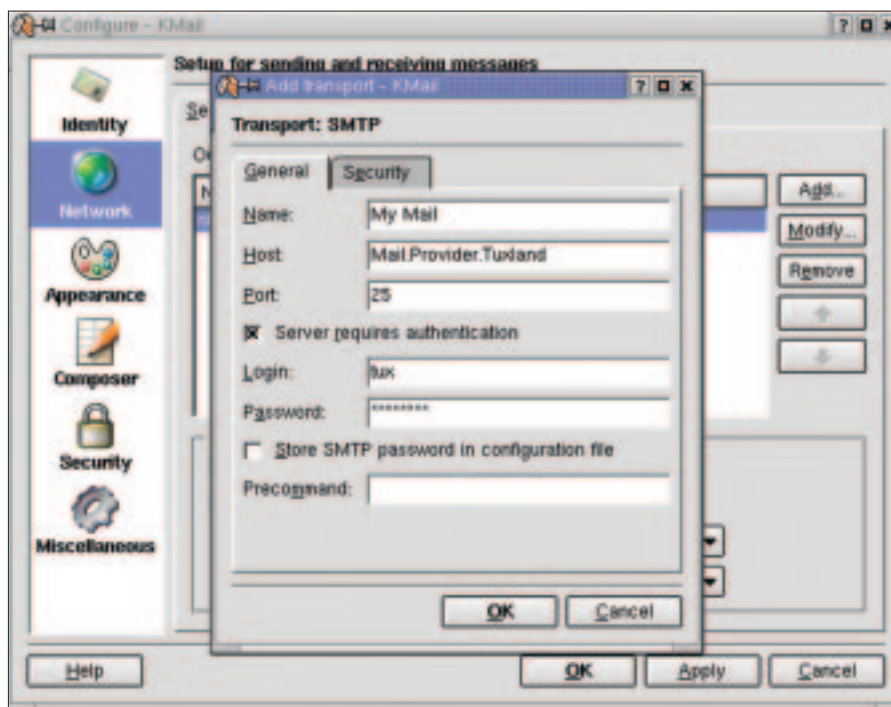


Figure 1: Two software configuration settings allow KMail to use SMTP-Auth for logging on

one of their customers' domains. And this despite the fact that clean and secure solutions to the authentication issue abound.

If it is not feasible to identify clients directly using the IP address assigned to them or by cryptographic methods, you can still resort to password protection. In contrast to the POP3 or IMAP protocols, which are responsible for retrieving email messages from mailboxes, the SMTP protocol originally did not provide for proof of identity based on user names or passwords. This weakness was resolved later, and all modern MTAs and mail clients support SMTP authentication (SMTP-Auth).

If a client program can correctly log on to the server while presenting mail for processing, the client will usually be considered trustworthy.

The Cyrus SASL package, [2], [3], can help Postfix out. You can use the package to set up a small login database `/etc/sasldb` that Postfix will use for authentication. The `saslpasswd` tool is used to add users to the database and `sasldblistusers` lists the current entries (see Listing 1).

Cyrus SASL can manage multiple host names and domains in the database allowing user accounts with the same name and different domains to coexist. Cyrus-SASL uses the concept of *realms*. If you do not use the `-u domainname` parameter to specify a *realm*, `saslpasswd` will assume the host name (`mailserver` in this case). For Postfix you can enter the name referred to as `smtpd_sasl_ local_domain` in the "Critical SASL Parameters ..." boxout!

To allow Postfix to relay messages from users authenticated by SMTP-Auth, the admin user now needs to add the `permit_sasl_authenticated` to the `smtpd_recipient_restrictions` list:

Listing 1: Database for SMTP-Auth

```
root: # saslpasswd -c tux
Password: secret
Again (for verification): secret
root: # sasldblistusers
user: tux realm: mailserver mech: DIGEST-MD5
user: tux realm: mailserver mech: PLAIN
user: tux realm: mailserver mech: CRAM-MD5
```

Critical SASL Parameters for Postfix

Use the following configuration parameters to teach Postfix SASL (Simple Authentication and Security Layer):

smtpd_sasl_auth_enable = yes

This parameter enables or disables

SMTP-Auth (no). smtpd_sasl_security_options = noanonymous, noplaintext noanonymous

prevents anonymous logins (the whole effort would be senseless otherwise). *noplaintext* prevents clients from transmitting SMTP authentication passwords in the clear, in contrast to the *PLAIN* and *LOGIN* authentication methods. This parameter forces the client to apply an encryption algorithm to encode its password and prevent it from being snarfed.

This makes sense from a security point of view: Users are now forced to use a secure configuration (such as *CRAM-MD5* or *DIGEST-MD5*, Figure 2). If they log on via a connection protected by SSL/TLS (and clear text passwords are protected by this encryption scheme), there is no reason to ban *PLAIN* and *LOGIN*, of course.

smtpd_sasl_local_domain = postfixbuch

This parameter must contain the value defined as your *realm* in *sasldb*. Realms are basically used to authenticate users from multiple (virtual) server domains, however, both Postfix and other SASL clients can normally handle only one SASL domain.

broken_sasl_auth_clients = yes

Some older clients, for example Microsoft Outlook Express 4.x, expect an answer from the mail server in *AUTH=LOGIN...* format, although *AUTH LOGIN...* is standard. If you set this parameter to *yes*, Postfix will issue the *AUTH* banner twice, using both formats.

```
smtpd_recipient_restrictions=
  permit_mynetworks,
  permit_sasl_authenticated,
  check_relay_domains,
  reject
```

also allows `saslpasswd` to read a password from standard input:

```
echo secret | saslpasswd >
-p -c tux
```

The following parameters are additionally added to `main.cf`:

```
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = >
noanonymous, noplaintext
smtpd_sasl_local_domain = >
mailserver
broken_sasl_auth_clients = yes
```

The "Critical SASL Parameters ..." boxout explains the individual options. The procedure is completed by running a small script to transfer your user data to the SASL database. The parameter `-p`

Although SMTP-Auth is simple to set up, there is one drawback: The postmaster will still need to teach users to add the required data to their client configurations (Figures 1 and 3).

POP before SMTP

POP before SMTP, also known as SMTP after POP, a commonly used method that works on IMAP servers, is a possible alternative.

Again, the idea is simple, although configuration may be more complex. When a client successfully completes the POP3/IMAP login procedure on a server,

Listing 2: Checking the Logfile

```
/var/log: # tail pop-before-smtp
[...]
read ts=Mar 22 11:03:29 ip=62.8.206.156
read ts=Mar 22 11:17:58 ip=217.235.18.66
    accepted --- not in mynetworks
    written ok
read ts=Mar 22 11:17:59 ip=217.235.18.66
purging ts=Fri Mar 22 10:21:50 2002 ip=217.224.233.74
```

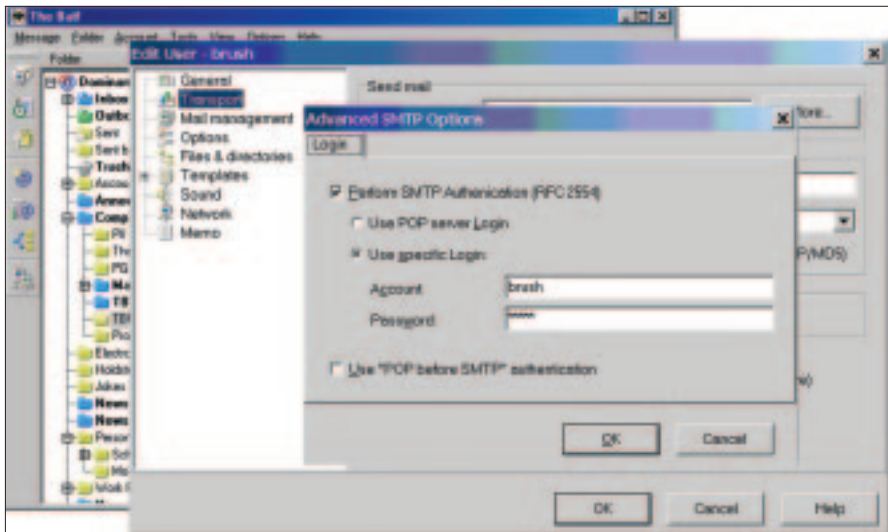


Figure 3: No matter whether you require MD5 based SMTP Authentication, or SMTP after POP, the Windows client, The Bat, is prepared

the server ascertains that it is one of your users. The mail server will now accept email messages from the client and relay them for a pre-defined period (normally 15 minutes). If a user has configured their mail software to query their inbox

before sending outgoing email messages, you do not need to perform any additional configuration steps on the client.

Some clients use this order by default today and others are easily re-configured. Outlook (Express), which still

ignores POP before SMTP is an exception and needs to be fooled into co-operating [4].

This approach creates a few problems for postmasters, and is controversial. For one thing, the identity of the sender cannot be proved, as access is permitted to a computer and it is impossible to check for multiple parallel users.

There is some risk of a dynamically assigned IP address being reassigned to another user within the time slot, and that this user coincidentally or specifically accesses the mail server their predecessor was permitted to access. But POP before SMTP still makes sense in many scenarios and can be implemented along with SMTP-Auth.

The *pop-before-smtp* [5] script (see the “Setting up *pop-before-smtp*” boxout for a description) runs as a daemon and monitors the logfiles on the POP3/IMAP server.

If new entries are added for successful login events, the script reads the IP address of the user from the logfile and

pop-before-smtp

If you have not yet installed the required Perl modules, you can perform this step automatically. If you have not used Perl/CPAN previously, you will need to navigate a few configuration prompts:

```
linux: # perl -MCPAN -e 'install Time::HiRes'
linux: # perl -MCPAN -e 'install File::Tail'
linux: # perl -MCPAN -e 'install Date::Parse'
linux: # perl -MCPAN -e 'install Net::Netmask'
```

Use the following init script to launch *pop-before-smtp* automatically at the appropriate runlevels:

```
/usr/local/src/pop-before-smtp-1.30: # cp pop-before-smtp.init /etc/init.d/pop-before-smtp
/usr/local/src/pop-before-smtp-1.30: # cp pop-before-smtp-config.pl /etc
/usr/local/src/pop-before-smtp-1.30: # cp pop-before-smtp /usr/sbin

/etc/init.d: # ln -s pop-before-smtp rc3.d/S11pop-before-smtp
/etc/init.d: # ln -s pop-before-smtp rc3.d/K11pop-before-smtp
/etc/init.d: # ln -s pop-before-smtp rc5.d/S11pop-before-smtp
/etc/init.d: # ln -s pop-before-smtp rc5.d/K11pop-before-smtp
/usr/sbin: # ln -s pop-before-smtp /usr/sbin/rcpop-before-smtp
```

Now edit the configuration file, */etc/pop-before-smtp-conf.pl*, and launch the daemon manually. You should check whether the script runs correctly when you boot your computer at a later stage:

```
/etc/init.d: # rcpop-before-smtp start
/etc/init.d: # ps ax | grep pop-before-smtp
5022 ?      S      0:07 /usr/bin/perl -wT /usr/sbin/pop-before-smtp --watchlog=/var/log/mail 2
|--logto=/var/log/pop-before-smtp --daemon=/var/run/pop-before-smtp.pid
9367 pts/1 S      0:00 grep pop-before
/etc/init.d: # ls -al /etc/postfix/pop*
-rw-r--r-- 1 root  root 12288 0kt 8 11:18 /etc/postfix/pop-before-smtp.db
```

adds it to the `/etc/postfix/pop-before-smtp.db` database.

Now that Postfix will accept and relay mail from the allowed IP addresses, there is nothing to stop authenticated users from sending messages. The script also removes the IP address from the database after the time slot has expired.

To allow the program to accurately identify and extract the logfile entries for your POP/IMAP servers, you will need to use regular expressions in the `/etc/pop-before-smtp-conf.pl` configuration file. The script contains entries for major mail servers – use an editor to enable the lines, as required.

Make sure that you check the paths in the configuration file, paying particular attention to the logfile entries:

```
# Set the log file we will watch
# for pop3d/imapd records.
$file_tail{'name'} = >
'/var/log/maillog';
```

You can use the init script in the “Setting up `pop-before-smtp`” box to run the daemon when you boot your system.

```
rcpop-before-smtp start
```

should launch the script, retire to the background and create the database. In `debug` mode the daemon will log each IP address it recognizes in `/var/log/pop-before-smtp` (Listing 2). IP addresses that Postfix already recognizes as `mynet-`

`works`, or allowed addresses are not transferred to the database, but simply logged. Use the configuration file to enable debug mode; you will need to set the `$debug` variable to `1` to do so:

```
# Set $debug to output some
# extra log messages
# (if logging is enabled).
$debug = 1;
```

If the logfile contains the *written ok* string, you can assume that the script has added the IP address to the database. If the string is missing, `pop-before-smtp` is simply letting you know that the IP address was noted during an earlier login and is not allowed currently. *purging* means that the time slot has expired for this IP address, and the address has been deleted from the database.

Monitoring Logfiles

Monitor the logfile after installation to ensure that logins are being correctly identified. If `pop-before-smtp` refuses to recognize any IP addresses, this indicates a faulty regular expression in `/etc/pop-before-smtp-conf.pl`.

Postfix still needs to understand how to evaluate the database, in order to answer the question as to whether a user should be permitted to relay. The `check_client_access` parameter in `smtpd_recipient_restrictions` is used for this purpose. The database from the `pop-before-smtp` script is also required:

```
smtpd_recipient_restrictions=
  permit_mynetworks,
  check_sasl_authenticated,
  check_client_access hash:>
/etc/<$> postfix/>
pop-before-smtp,
  check_relay_domains,
[... other restrictions as >
applicable ...]
  reject
```

Before you let your server loose on the Internet, you might like to recheck carefully the configuration [6]: Open relays crop up in databases such as ORDB [7], and many servers refuse to accept mail from computers listed there and at similar locations.

Once you have been black-listed in an open relay database, it is quite hard to get your name off the list. Your last resort may be changing your server’s IP address. It may also be a fitting punishment, because running a mail server as an open relay really makes you a spammer’s accomplice. ■

INFO

- [1] Postfix project site: <http://www.postfix.org/>
- [2] Cyrus-SASL Sources: <ftp://ftp.andrew.cmu.edu/pub/cyrus-mail/>
- [3] SASL info and howto: <http://www.thecabal.org/~devin/postfix/smtp-auth.txt>
- [4] POP before SMTP with Outlook: http://people.freenet.de/ingos_pages/english.htm
- [5] Script and howto for SMTP after POP: <http://popbsmtp.sourceforge.net/>
- [6] Very good relay test: <http://www.abuse.net/relay.html>
- [7] The Open Relay Database: <http://www.ordb.org/>

For Experts: POP before SMTP without root privileges

Apart from the fact that normal users do not have write access to `/etc`, that is the default path for the database, there is no reason why `pop-before-smtp` needs to run with root privileges. So you might like to create a special account for the daemon with a user ID of less than 500, setting `*` as (an invalid) password in `/etc/shadow`, and `/bin/false` as the login shell, and `/var/popbsmtp` as the home directory for the daemon.

You can then assign appropriate access privileges for just this directory, and change the script and Postfix database path configuration to `/var/popbsmtp/pop-before-smtp`. Change the call in the init script to use `startproc -u user` to run the script (insert the user name you assigned to replace `user`). Look for the following lines:

```
start)
  echo -n "Starting $progname: "
  $pgm $dbfile $watchlog $logto --daemon=$pid
```

and change them to:

```
start)
  echo -n "Starting $progname: "
  startproc -u user $pgm $dbfile $watchlog $logto --daemon=$pid
```

THE AUTHOR

Peer Heinlein has been teaching Linux for many years now, and is an Internet Service Provider based in Berlin.

He has recently published a PostFix book for SuSE Press, on the MTA and running secure mail servers on Linux.

