workaround.org

literacy and subtleties for open-source bigots and other weirdos

ISP-style Email Server with Debian "Sarge" and Postfix 2.1

Submitted by Christoph Haas on Sun, 12/27/2009 - 00:29

*Note: Debian Sarge is not the stable version of Debian any more. Consider reading the **newest tutorial** (http://workaround.org/ispmail) .*

Abstract

You have probably already seen web hosters who allow you to rent domains and receive email on these domains. Have you ever wondered how they actually handle these thousands of domains? There is surely nobody entering all these domains and aliases into a '*main.cf*' configuration file manually. Postfix offers two nice features that simplify such tasks:

- Virtual domains
  In addition to your local domain (which is probably the domain that is configured in */etc/defaultdomain*) you may receive email for other domains that are called virtual domains. There is no limitation on the number of domains you can receive email for.
- Database lookups
  You do not need to store all the information about your users and valid email addresses in text files. Postfix supports database lookups to common DBMSs like MySQL or PostgreSQL. This approach is especially charming as you may write a web administration GUI to manage the database. You may even allow your users to take care of their email accounts themselves.

This tutorial will introduce you to the basics of this kind of configuration. If you carefully follow all the steps in this document you will end up with a mail server that can handle thousands of domains and user accounts. These are some features you will get:

- POP3/IMAP access for your users
- Webmail access
- Virus scanning
- Spam prevention
- Secure mail relay access for road-warriors
- Easy domain administration

Although I will try to get you going quickly you will need to know a few things already:

- MySQL (creating a database, granting access for users and how SQL queries look)
- SMTP, POP3, IMAP (I assume you have a basic knowledge of these protocols)
- Basic Postfix configuration (you should be familiar with the 'main.cf' configuration file)
- Debian/Linux (you should know basic system administration tasks like installing software or editing text files)

Table of Contents

The components
What are mappings?
How virtual domains work

Step 1: Install the needed Debian packages
Step 2: Create the database
Step 3: Create the tables

- domains
- forwardings
- users

Step 4: Create the database mapping definitions

- mysql-virtual_domains.cf
- mysql-virtual_forwardings.cf
- mysql-virtual_mailboxes.cf
- mysql-virtual_email2email.cf

Step 5: Create a vmail user
Step 6: Edit the main.cf

- A quick test

Step 7: Make Postfix understand authenticated SMTP (Auth-SMTP)

- Tell Postfix to use SASL/MySQL
- Use TLS to encrypt SMTP traffic

Step 8: Configure the POP3 / IMAP service

- A quick test

Step 9: Test your setup

- domains
- users

Step 10: Populate your database

- For every new domain...
- For every new user...
- For every new forwarding...

Scanning incoming email for viruses and spam (optional)

- Introduction to AMaViS
- Configure AMaViS
- Tell Postfix to use AMaViS
- How does the content filtering work?
- Train it with ham and spam

Offering webmail access (optional)
Mailing lists with mailman

- Forwarding to local pipes
- Generic approach using regular expressions
- Using mailman as a transport service

Troubleshooting

- Error messages
- MySQL debugging
- Getting help on IRC

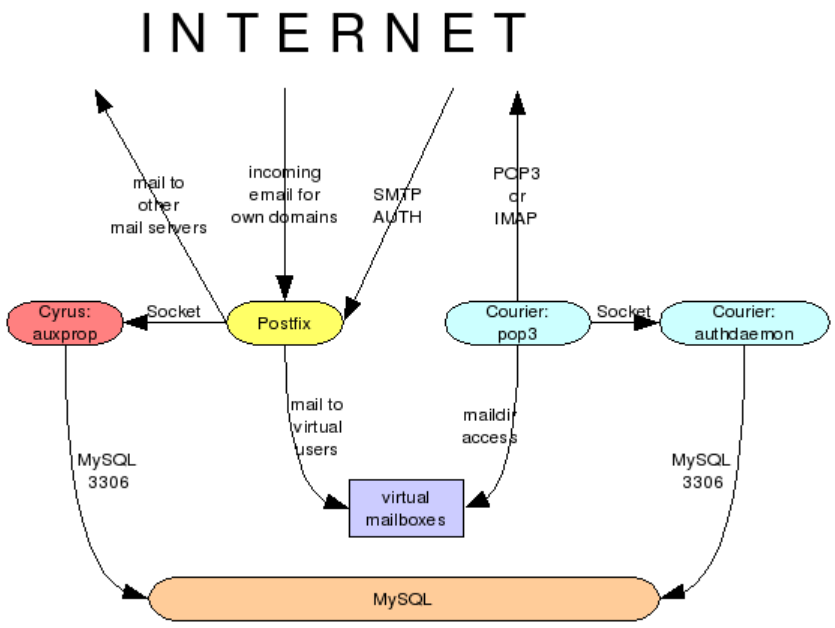
Migrating from the previous version of the tutorial
Server-side mail filtering using maildrop
Thanks
Planned for the Etch tutorial
Contributions from readers

The components

The whole setup depends on different software components that play together nicely. Let me clarify what each of them does:

- Postfix: Your MTA (Mail Transfer Agent) that receives emails via the SMTP (simple mail transfer protocol) and delivers them to different places on your hard disk.
- MySQL: The database server that stores the information to control the behaviour of postfix. It knows about users, domains, email forwardings and passwords.
- Courier: Courier is a standalone mail server just like Postfix. I will however just use its POP3/IMAP server component to let users access the mailboxes.
- SASL (the Cyrus library): If your users are dialed in at another ISP (Internet Service Provider) while they are on the road they get an IP address outside of your network. Your mail server however only trusts local IP addresses. The SASL (Simple Authentication and Security Layer) adds authentication to SMTP and makes your mail server trust them.
- AMaViS: A mail virus scanner that works as a content filter in Postfix. It scans incoming mail for spam pattern (using the well-known spamassassin) or viruses.
- phpmyadmin: A web interface to manage your local MySQL databases. It's far more comfortable than using the 'mysql' command from the command-line.

The big picture looks something like this:



What are mappings?

In short a mapping assigns one value to another. You probably know the file */etc/aliases* where you define forwardings for your local domain. A line there looks like this:

```
postmaster: root
```

This makes all mail to postmaster@your-domain be redirected to root@your-domain. The left side (here: "postmaster") is commonly called LHS (left-hand side) and the right side (here: "root") is called RHS (right-hand side) accordingly. You will find that these are common abbreviations when talking about mappings.

Hint: Usually map files do not have colons (':') on the left side. This is special to the aliases table for historical and compatibility reasons. Also the local aliases file is special in that it is not compiled with **postmap** but the **newaliases** command. This is just a (bad) example. :)

If you are setting up Postfix the quick and dirty way you will typically start with text files like the one above. You just write the mappings into it and run **postmap filename** on it to convert the text file into a hash file called "filename.db". Then you can access this mapping using *"hash:filename"* in your Postfix configuration. You may notice that the default alias maps configuration looks like *"alias_maps = hash:/etc/aliases"* - just as an example. The *"hash:"* is called the lookup method.

In my setup I replace the text files by MySQL tables. This makes data handling a lot more flexible. But as database tables usually contain more than just two columns you will need to tell Postfix which column is meant to be the LHS and which is the RHS. This definition is stored in a text file like this:

```
user = provider_admin
password = DomAKg07
```

```
dbname = provider
table = virtual_mailboxes
select_field = mailbox
where_field = email
hosts = 127.0.0.1
```

If a file like this would be saved as */etc/postfix/mysql_virtual_mailboxes.cf* you could use a mapping of *"virtual_mailbox_maps=mysql:/etc/postfix/mysql_virtual_mailboxes.cf"*. The LHS of the mapping is defined as *'where_field'* and the RHS is defined as *'select_field'*. In this example it would be a mapping of the *`email`* column to the *`mailbox`* column. The other fields in this definition file are *user* (the username that connects to the MySQL database), *password* (the password of that user), *dbname* (the name of the database), *table* (the name of the table in that database) and *hosts* (the name of the server that MySQL runs on).

How virtual domains work

Let me begin with a brief introduction to virtual domains in Postfix 2.x (the deprecated Postfix 1.x handles them differently) because misconfiguration will cost you hair and time. There are two types of domains:

- Local Domains
  All domains listed as *mydestination* in your main.cf are treated as *local domains*. Your default domain (*/etc/defaultdomain*) is usually configured as a local domain. Emails for local domains are delivered to system users (those you configure in */etc/passwd*). The mails will be delivered to /var/mail.
- Virtual Domains
  Your mail server can receive emails for additional domains called *virtual domains*. Virtual domains are very flexible. You do not need system accounts for every mail user (that means users do not need to be configured in the */etc/passwd*). So your system can handle thousands of email users easily. A mapping (see above) is used to save the information about the users. In my example I use MySQL for that reason.

To make matters a little more complicated there are two different kinds of virtual domains:

- Virtual Alias Domains
  A *virtual alias domain* can be used for forwarding ("aliasing") email from an email address to another email address. Such a domain can in turn not be used to receive email for (that gets delivered to a mailbox on your hard disk). I won't use this type of virtual domains for my setup because I can still use aliasing using the *virtual_alias_maps* mapping even if the domains are not listed as virtual alias domains. (The *virtual_alias_maps* is a general-purpose redirection mapping that works for everything that passes your system - even local domains.)
- Virtual Mailbox Domains
  A *virtual mailbox domain* lets you receive email for users of that domain to mailboxes on your hard disk. You can still use the *virtual_alias_maps* mapping to forward email to other mailboxes or external email addresses so not every user on that domain must actually have a mailbox but can also just have the email forwarded somewhere else. This is merely a parameter that tells Postfix what you mainly intend to do with the domain. By the way - the *virtual_mailbox_maps* mapping is used to determine the location of the mailbox on your hard disk.

It is important to understand that a domain is either a *virtual alias domain* or a *virtual mailbox domain* or a *local domain*. If you make a domain a virtual alias domain you will not be able to receive email for that domain on your server. On the contrary you can use the *virtual_alias_maps* to forward/alias email for both kinds of domain. So the virtual mailbox domains are generally the more flexible choice.

> A domain can either be virtual or local - never both! So if you decide you want your default domain be a virtual domain then remove it from the *mydestination* definition. Just leave it blank or set it to *"mydestination=localhost"*. Email addresses like *root@localhost* would then be delivered to the local 'root' user.

I recommend you also betimes read the original documentation about virtual domains in the VIRTUAL_README that shipped with the postfix-doc package and is found in */usr/share/doc/postfix/VIRTUAL_README.gz*.

Step 1: Install the needed Debian packages

Packages you will absolutely need:

- postfix (Choose: "Local only")
- postfix-mysql
- postfix-doc

If you intend to run the MySQL server on the same machine:

- mysql-server (for MySQL 3.x/4.0) or mysql-server-4.1 (for MySQL4.1)

If you want to offer mail access using POP3/IMAP you need:

- courier-authdaemon
- courier-authmysql
- courier-pop (for unencrypted POP3 access)
- courier-pop-ssl (for SSL-encrypted POP3 access)
- courier-imap (for unencrypted IMAP access)
- courier-imap-ssl (for SSL-encrypted IMAP access)

If you want to allow road-warriors to send email through your server using authenticated SMTP you also need:

- postfix-tls (for encrypted authenticated SMTP)
- libsasl2 (the Cyrus SASL library)
- libsasl2-modules (the mechanisms for the SASL library)
- libsasl2-modules-sql
- openssl (to create the certificate)

If you want to scan incoming email for viruses and spam:

- amavisd-new
- spamassassin
- clamav
- clamav-daemon
- zoo
- unzip
- unarj
- lha (in non-free!)

If you want to offer webmail for your users:

- squirrelmail

Optional but useful packages:

- phpmyadmin (PHP interface for easy administration of MySQL databases)

Step 2: Create the database

You need to create a database first which holds the tables. If you are experienced in using MySQL you can of course do this work on the command line. However I would rather use *phpmyadmin* for MySQL database management. It is up to you.

Hint: when the mysql-server is first run you can access the database as user *'root'* with no password. You should first set a password for that account:

```
mysqladmin -u root password your-mysql-password
```

First create a database. I call it *'provider'* because I intend to do more than just email (which is not within the scope of this document at the moment). Either do this in phpmyadmin or run this shell command:

```
mysqladmin -u root -p create provider
```

Next you need a database user who has the sufficient permissions to access your database. Open a connection to the database using this shell command:

```
mysql -u root -p
```

...and enter this SQL command when you see the *mysql>* prompt:

```
grant select on provider.* to provider_admin@localhost identified by 'your-password';
```

This creates a new database user called *provider_admin* who just has 'SELECT' privileges on your database for security reasons. Please replace the 'your-password' with a password of your taste. I usually run *'pwgen -s 16* to automatically create passwords. (You will need to **apt-get install pwgen** to make this command available. Finally you need to reload the permissions information in MySQL by running this SQL statement:

```
flush privileges;
```

Step 3: Create the tables

After you have created the database you will need to create the database tables that will contain the control information

for Postfix. We will create the tables now. If you are using the command line program **mysql** then send the command

```
USE provider;
```

first to select the right database where to create the tables.

domains
The first table will contain just one boring column containing the virtual domain name. This table will need to have a row for each virtual domain. Just run this SQL statement to create it:

```
CREATE TABLE domains (
domain varchar(50) NOT NULL,
PRIMARY KEY (domain) )
TYPE=MyISAM;
```

forwardings
The table *'forwardings'* will be used to alias one email address to another. You can use this table for general redirections. (Hint: this even works for your local domain.) This is the SQL statement to create the table:

```
CREATE TABLE forwardings (
source varchar(80) NOT NULL,
destination TEXT NOT NULL,
PRIMARY KEY (source) )
TYPE=MyISAM;
```

users
Finally the *'users'* table contains information about your user accounts. Every user has a username and password for accessing the mailbox by POP3 or IMAP. As users tend to forget things (just look under the keyboard of your boss for his password) I decided to use the *'email'* address also as a login username. The email address is also used for the directory name where emails for this users will be stored on the hard disk. So just these two fields are sufficient here. Just another SQL query to copy and paste:

```
CREATE TABLE users (
email varchar(80) NOT NULL,
password varchar(20) NOT NULL,
PRIMARY KEY (email)
) TYPE=MyISAM;
```

Step 4: Create the database mapping definitions

As specified earlier in this document you need to tell Postfix where the control information is stored in the database. You need to create the following four text files in */etc/postfix* for that reason.

Postfix runs in a chroot directory (*/var/spool/postfix*) and cannot access any files outside that directory. Usually you talk to the MySQL database via its socket file */var/run/mysqld/mysqld.sock*. As you can see this file is out of reach for Postfix. So you can either try to move the socket file (which may lead to other problems) or use TCP networking. The latter means you are talking to MySQL through your network stack. Advantage: you do not need to care about the chroot restrictions. Slight disadvantage: the MySQL server is accessible through the *lo* interface which could in theory be a security problem.

mysql-virtual_domains.cf
This is a simple mapping of your virtual domain name (LHS) to the string *'virtual'* (RHS). This mapping is used for the *virtual_mailbox_domains* definition where just the LHS matters. (The string could be effectively anything - set it to "banana daiquiri" if you like.)

Create the file and replace the '...' by your database access password. Do not use "localhost" as the server name because Postfix would try to use the MySQL socket for communication. If the database server runs on your server then 127.0.0.1 means to use TCP/IP communication.

```
user = provider_admin
password = ...
dbname = provider
table = domains
select_field = 'virtual'
where_field = domain
hosts = 127.0.0.1
```

mysql-virtual_forwardings.cf
This mapping reads the 'forwardings' table to provide a way to redirect email addresses. It simply maps the 'source' column to the 'destination' column. We will use it for the virtual_alias_maps mapping.

```
user = provider_admin
```

```
password = ...
dbname = provider
table = forwardings
select_field = destination
where_field = source
hosts = 127.0.0.1
```

mysql-virtual_mailboxes.cf

The next definition file deals with user mailboxes. It tells Postfix where to store email for a specific email address. We will use it later in the *virtual_mailbox_maps* mapping. Postfix will search the *'mailbox'* column (RHS) for a given *'email'* address (LHS). The purpose of the weird *select_field* line is to convert an email like 'user@domain.net' to a path like 'domain.net/user/' which is the place where the user's mailbox is found. It contains a trailing slash so Postfix will create a maildir structure instead of a mailbox file. This is needed for the Courier POP3 and IMAP services later.

```
user = provider_admin
password = ...
dbname = provider
table = users
select_field = CONCAT(SUBSTRING_INDEX(email,'@',-1),'/',SUBSTRING_INDEX(email,'@',1),'/')
where_field = email
hosts = 127.0.0.1
```

mysql-virtual_email2email.cf

Another quirk of virtual domains is the precedence of the *virtual_alias_maps* mapping when using catch-all addresses. ("Catch-all" means you are using a forwarding like *"@domain.com"->"service@domain.com"* to catch all email that is sent to any user name in that domain.) If you were to map *@domain.com -> service@domain.com* in *virtual_alias_maps* but not any of the other addresses like *specific.user@domain.com*, all email would be delivered to service@domain.com even if specific.user@domain.com is listed in *virtual_mailbox_maps*.

This is a simple mapping for all email addresses in the *users* table to themselves to work around the above problem.

```
user = provider_admin
password = ...
dbname = provider
table = users
select_field = email
where_field = email
hosts = 127.0.0.1
```

> Make sure nobody but root can read these files. Otherwise everybody on your system could read your database access password in plain text. You need to set the group of these files to postfix by running **"chgrp postfix /etc/postfix/mysql-virtual_*.cf"**. And make the files readable by the group: **"chmod u=rw,g=r,o= /etc/postfix/mysql-virtual_*.cf"**.

Step 5: Create a vmail user

Your system can hold mailboxes for thousands of users. You probably do not want to assign a unique UID (user ID) to every user. So I recommend you create a pseudo-user who will become the owner of all mailboxes.

Just enter these lines in a root shell:

```
groupadd -g 5000 vmail
useradd -g vmail -u 5000 vmail -d /home/vmail -m
```

Step 6: Edit the main.cf

The */etc/postfix/main.cf* is the main configuration file for Postfix. I will describe the basic settings needed for virtual domains. Do not wipe away your main.cf and paste these lines into it. You will probably want to customise the configuration in other ways so I cannot be complete here.

Setting	Meaning
inet_interfaces = all	Debian sets this to *loopback-only* during the installation for security reasons. If you want Postfix to listen to incoming SMTP requests on all interfaces you may want to set it to 'all' or comment out the line because 'all' is the default for Postfix anyway. See also its definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#inet_interfaces) .
myhostname = ...	Make sure this is set to your fully qualified domain name. See also its definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#myhostname) .
mydestination = ...	List your local domains here seperated by commas. Do not list any virtual domain here. See also its definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#mydestination) .

mynetworks = ...	List the IP ranges here that are allowed to send email through your mail server. This is most likely your local network. See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#mynetworks)** .
virtual_alias_domains =	This parameter is supposed to be unset. We will not use virtual alias domains. If you left it to the default you would have it set to *virtual_alias_maps* for backwards-compatability to older releases. See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#virtual_alias_domains)** .
virtual_alias_maps = mysql:/etc/postfix/mysql-virtual_forwardings.cf mysql:/etc/postfix/mysql-virtual_email2email.cf	This is a general purpose redirection table. You can redirect one email address to another or even catch all mail for a domain and redirect it to one specific email address. The information is stored in the *'forwardings'* table. Every LHS email address is rewritten to the addresses on the RHS. If you have multiple destination addresses you can comma-seperate them in one entry. I also use the *mysql-virtual_email2email.cf* mapping to point the email to itself. That may sound stupid but is badly needed if you use catchall addresses. See **the mysql-virtual_email2email mapping file (http://workaround.org/articles/ispmail-sarge/index.shtml.en#mysql-virtual_email2email.cf)** to see what's in that mapping file. See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#virtual_alias_maps)** .
virtual_mailbox_domains = mysql:/etc/postfix/mysql-virtual_domains.cf	This is the list of virtual mailbox domains from the *`domains`* table. See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#virtual_mailbox_domains)** .
virtual_mailbox_maps = mysql:/etc/postfix/mysql-virtual_mailboxes.cf	Another important mapping is the virtual mailbox maps. It maps email addresses (LHS) to the location of the mailbox on your hard disk (RHS) relative to the *virtual_mailbox_base*. See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#virtual_mailbox_maps)** .
virtual_mailbox_base = /home/vmail	This is the base path where the mailboxes of the users will be stored on your hard disk (see above). See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#virtual_mailbox_base)** .
virtual_uid_maps = static:5000	You should tell Postfix who shall be the owner of the mailboxes. This is the UID (user ID) of the owner (taken from */etc/passwd*). Set it to the UID of the *"vmail"* user you just created. See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#virtual_uid_maps)** .
virtual_gid_maps = static:5000	The same for the GID (group ID). See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#virtual_gid_maps)** .
smtpd_sasl_auth_enable = yes	Enable the authenticated SMTP feature. See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#smtpd_sasl_auth_enable)** .
broken_sasl_auth_clients = yes	Some broken mail clients like Microsoft Outlook use a deprecated way to detect if a mail server speaks authenticated SMTP. Make them happy. Who said something about Outlook? :) See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#broken_sasl_auth_clients)** .
smtpd_recipient_restrictions = permit_mynetworks, permit_sasl_authenticated, reject_unauth_destination	These restrictions are checked whenever a new email arrives at your mail server to check who is allowed to relay email. *permit_mynetworks* will allow everybody you configured in mynetworks. *permit_sasl_authenticated* will allow everybody from everywhere as long as they use authenticated SMTP. And finally relaying is allowed if the user wants to send email to one of the domains listed in *relay_domains* or to local domains. See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#smtpd_recipient_restrictions)** .
smtpd_use_tls = yes	Encrypt the authenticated SMTP session using SSL See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#smtpd_use_tls)** .
smtpd_tls_cert_file = /etc/postfix/smtpd.cert	The location of the SSL certificate for TLS (we will create it later) See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#smtpd_tls_cert_file)** .
smtpd_tls_key_file = /etc/postfix/smtpd.key	The location of the SSL private key for TLS See also its **definition in the Postfix documentation (http://www.postfix.org/postconf.5.html#smtpd_tls_key_file)** .

A quick test

Restart Postfix (**/etc/init.d/postfix restart**) and run **postfix check**. If you do not get any warnings this part is complete.

Step 7: Make Postfix understand authenticated SMTP (Auth-SMTP)

Imagine your users fetch their email using POP3. Now they need a way to send mails back through your mail server. For security reasons Postfix allows users defined in *mynetworks* to send emails. Usually your mail server will only accept mails for its own domains. If you allowed everybody to send email to every other domain you would provide a so called *open relay* that spammers abuse to send out their digital trash. So the logical way is to make remote users trusted by letting them provide their username and password. If the credentials are correct the user will be trusted like he has an IP in *mynetworks*. Most email cilents have this feature included.

Setting up authenticated SMTP is quite easy. The only pitfall is that Debian runs Postfix in a chroot'ed environment in */var/spool/postfix* by default.

Tell Postfix to use SASL/MySQL

As written earlier Postfix uses the Cyrus SASL library for authenticated SMTP. So you need to tell Postfix how to access the data storage that keeps the usernames and passwords. This is easy. You just need to create a file */etc/postfix/sasl/smtpd.conf* like this:

```
pwcheck_method: auxprop
auxprop_plugin: sql
mech_list: plain login cram-md5 digest-md5
sql_engine: mysql
sql_hostnames: 127.0.0.1
sql_user: provider_admin
sql_passwd: ...
sql_database: provider
sql_select: select password from users where email='%u@%r'
```

> You know the drill. Use proper permissions on this file: **chown root:postfix /etc/postfix/sasl/smtpd.conf** and **chmod u=rw,g=r,o= /etc/postfix/sasl/smtpd.conf**.

(If you have trouble with SASL you may consider inserting a line like "log_level: 7" here. It will write more verbose information to the log files and perhaps help to find the cause.)

Use TLS to encrypt SMTP traffic

An important step is to encrypt the SMTP session. Otherwise the username and password could be transmitted in a very insecure way if the mail client chose to use one of plaintext authentication methods). So I encourage you to encrypt that communication using TLS. TLS is short for Transport Layer Security (RFC2246) and in short terms uses SSL (Secure Socket Layer) which encrypts the mail connection between the road-warrior and the mail server.

First you will need an SSL certificate. If you don't want to pay for one from your favorite trustcenter you can well use a self-signed one. (Personal note: I wonder how paying for something makes it more trusted.) The only drawback: the mail client does not know about your CA (certificate authority) and will spit out a warning to the user. Either tell the users to ignore the warning or let them install the certificate on their computers. (Outlook does not allow the unknown CA to be ignored. You need to install the certificate to make it work.)

For a certificate that is valid for ten years for the hostname smtp.domain.tld you would type this:

```
openssl req -new -outform PEM -out /etc/postfix/smtpd.cert -newkey rsa:2048 \
-nodes -keyout /etc/postfix/smtpd.key -keyform PEM -days 3650 -x509
```

You will then be asked a few question about the fields of the certificate. It does not matter what you enter. Just fill the fields. One exception though - the "Common Name" must be the hostname of your mail server. Example session:

```
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Hamburg
Locality Name (eg, city) []:Hamburg
Organization Name (eg, company) [Internet Widgits Pty Ltd]:workaround.org email services
Organizational Unit Name (eg, section) []:Master of Disaster
Common Name (eg, YOUR name) []:smtp.domain.tld
Email Address []:postmaster@domain.tld
```

After a short moment you will get two files: "smtpd.key" (the private key file) and "smtpd.cert" (the certificate).

> Make sure at least the key file is not readable for the whole wide world: **chmod u=rw,g=r,o= /etc/postfix/smtpd.key** and **chown root:postfix /etc/postfix/smtpd.key**

Step 8: Configure the POP3 / IMAP service

You already set up a large part of the configuration. However your users will still be unhappy as they cannot reach their mailboxes. So it is time to configure the POP3 or IMAP service. First you need to edit the file */etc/courier/authdaemonrc* and set the directive authmodulelist to *"authmysql"* like this:

```
authmodulelist="authmysql"
```

Then you need to define the fields of the MySQL database table in */etc/courier/authmysqlrc* like this:

```
MYSQL_SERVER localhost
MYSQL_USERNAME provider_admin
```

```
MYSQL_PASSWORD ...
MYSQL_PORT 0
MYSQL_DATABASE provider
MYSQL_USER_TABLE users
#MYSQL_CRYPT_PWFIELD (comment this out)
MYSQL_CLEAR_PWFIELD password
MYSQL_UID_FIELD 5000
MYSQL_GID_FIELD 5000
MYSQL_LOGIN_FIELD email
MYSQL_HOME_FIELD "/home/vmail"
#MYSQL_NAME_FIELD (comment this out)
MYSQL_MAILDIR_FIELD CONCAT(SUBSTRING_INDEX(email,'@',-1),'/',SUBSTRING_INDEX(email,'@',1),'/')
```

Careful - the authmysqlrc is a bit picky. Make sure you have not accidentally inserted trailing spaces on the lines. (If you are editing the file in 'vim' this command should take care of that: **:%s/\s\+$//g**) Do not forget to restart the authdaemon process using **/etc/init.d/courier-authdaemon restart**.

A quick test

Try to reach the POP3 service by running **telnet localhost pop3**. You should get a *"+OK Hello there."*. Voila - your users should be happy now. :)

> You cannot fetch emails from a mailbox unless at least one mail has been sent there. Users would get cryptic error messages. So I recommend sending a welcome email to new users.

Step 9: Test your setup

Congratulations. The configuration part is done. Now comes the more practical part. We will now create the database entries for your first domain so you can test to receive an email for the virtual domain. Please create these rows in the appropriate database tables:

domains

Column	Value
domain	virtual.test

users

Column	Value
email	user@virtual.test
password	secret

In MySQL speak this means entering the following two statements:

```
INSERT INTO `domains` (`domain`) VALUES ('virtual.test');
INSERT INTO `users` (`email`,`password`) VALUES ('user@virtual.test','secret');
```

This means we have one domain called "virtual.test" and one user whose email address (and also the username) is "user@virtual.test". The password for this user is "secret". As you do not have an MX entry (mail exchanger - part of the DNS zone) you need to 'deliver' the email manually. Establish an SMTP connection to your mail server (telnet servername 25) and enter the SMTP commands written on the right side:

Server	You
`220 myserver ESMTP Postfix (Debian/GNU)`	
	`ehlo workaround.org`

`250-mailtest`
`250-PIPELINING`
`250-SIZE 10240000`
`250-VRFY`
`250-ETRN`
`250-STARTTLS`
`250-AUTH LOGIN PLAIN DIGEST-MD5 CRAM-MD5`
`250-AUTH=LOGIN PLAIN DIGEST-MD5 CRAM-MD5`
`250 8BITMIME`	

	`mail from:<test@workaround.org>`

```
250 Ok
```

```
                                                    rcpt to:<user@virtual.test>
```

```
250 Ok
```

```
                                                    data
```

```
354 End data with <CR><LF>.<CR><LF></LF></CR></LF></CR>
```

```
                                                    This is a test email.
                                                    .
```

```
250 Ok: queued as ABC1D1C123
```

```
                                                    quit
```

```
221 BYE
```

If the server responded like in the example dialog above then the email was at least received. In the log file */var/log/mail.log* you should find a section like this:

```
Jul 24 21:48:28 myserver postfix/smtpd[9119]: connect from myserver[127.0.0.1]
Jul 24 21:48:48 myserver postfix/smtpd[9119]: F2C1B47BD: client=myserver[127.0.0.1]
Jul 24 21:48:52 myserver postfix/cleanup[9144]: F2C1B47BD: message-id=<20040724194842.F2C1B47BD@myserver>

Jul 24 21:48:52 myserver postfix/qmgr[9117]: F2C1B47BD: from=<test@workaround.org>, size=313, nrcpt=1 (qu
eue active)
Jul 24 21:48:52 myserver postfix/virtual[9148]: F2C1B47BD: to=<user@virtual.test>, relay=virtual, delay=1
0, status=sent (delivered to maildir)
```

If you read "status=sent (delivered to maildir)" then the email has been successfully delivered. Run the command **find /home/vmail** to see all directories and files there. It should look like this:

```
/home/vmail/virtual.test
/home/vmail/virtual.test/user
/home/vmail/virtual.test/user/tmp
/home/vmail/virtual.test/user/cur
/home/vmail/virtual.test/user/new
/home/vmail/virtual.test/user/new/1114511715.V801I7400b.my.server
```

Everyting worked like I described? Great. Then as a last test you may try to fetch the email from your mail client via POP3 or IMAP (depending on what service you installed). The username for fetching email is always the email address ("user@virtual.test") and the password is "secret" in this test case.

Step 10: Populate your database

Now that the first test has succeeded you will want to configure the datase for your own domains and users. Let me explain what you need to insert into the database:

For every new domain...
Insert the domain into the *'domains'* table.

For every new user...
Insert a new row into the *'users'* table containing the email address and the password (in plain text).

For every new forwarding...
Insert a new row into the *'forwardings'* table containing the source (the address you send mail to) and destination email address (the address the mail gets forwarded to). If you have multiple destinations (like a poor-man's mailing list) you may list all email addresses in one row just separated by commas. Hint: this table is used on every email that passes your system. So you can even redirect local mail addresses.

Examples:

source	destination	Effect
postmaster@my.domain	philip@my.domain	Redirect emails for postmaster to philip.
@my.domain	john@my.domain	Forward all email to email addresses in the domain my.domain to john. This does not apply to users in the *`email`* table though (like tina@my.domain). So more specific users of a domain always have a higher precedence than these so called "catch-all" accounts.
@my.domain	@another.domain	This is a whole domain redirection. Every email address in the my.domain domain will be directed to the same user at another.domain. So julie@my.domain will be redirected to julie@another.domain.
jesper@my.domain	dilbert@my.domain,dilbert@gmail.com	Forward email that is sent to jesper@my.domain to the two email addresses listed on the right side. Both users get a copy.

Scanning incoming email for viruses and spam (optional)

Introduction to AMaViS

The two most annoying things when using email are spam and viruses. Fortunately you can fight both using a software called AMaViS (A Mail Virus Scanner). AMaViS is an interface between Postfix, spamassassin (famous for its bayesian spam filtering capabilities) and any virus scanner (like ClamAV which is freely available). Do not get confused: AMaViS contains the spam filtering part but has no virus scanner built in. I suggest you install ClamAV, too. It is a free virus scanner that gets updated frequently.

You should already have the *amavisd-new* and other extractors installed.

Configure AMaViS

Please take a look at the */etc/amavis/amavisd.conf* file. These settings need to be taken care of:

Setting	Meaning
$mydomain = 'yourdomain.org';	You will find a reference to $mydomain quite often in the configuration file. So you know where you find it. If you do not use local domains I recommend you set it to 'localhost'.
@bypass_virus_checks_acl = qw(.);	If this line is commented out (has a '#' at the beginning of the line) then virus checks are enabled.
@bypass_spam_checks_acl = qw(.);	If this line is commented out (has a '#' at the beginning of the line) then spam checks are enabled.
@lookup_sql_dsn = (['DBI:mysql:provider', 'provider_admin', '...']);	This information tells AMaViS how to access your database. It needs to query the domains table. You need to replace the '...' with your database access password again. See also the next entry.
$sql_select_policy = 'SELECT "Y" as local FROM domains WHERE CONCAT("@",domain) IN (%k)';	AMaViS needs to know which domains are hosted on your server. When an email passes the mail server this setting is used to determine if the email is outgoing (sent by your system) or incoming (sent by someone on the internet). Outgoing email will not be scanned for viruses or spam.
$final_virus_destiny = D_DISCARD;	The default is to bounce infected mails. That is mostly useless because the mails are already received (Postfix has put them into the mail queue) and bouncing would mean to send back an error to whoever appears to be the sender of the email. Trust me - the real sender of a virus email is never the person who appears in the mail headers. Bouncing would mean you harass someone who is innocent. Don't.
$final_banned_destiny = D_REJECT;	This defines how to handle emails with banned attachment types. Many MIME types like PIF, EXE, COM or DOC files are infected. You may choose later which types you do not like.
$final_spam_destiny = D_PASS;	This defines what to do with emails that are classified as spam. You may wonder why I suggest you let these emails pass. My users would get upset if I just threw away emails that a machine has found to be spam. In any case a line *"X-Spam-Status"* is added to the mail header. So users can configure their email clients to handle mails differently depending on the spam status.
$sa_tag_level_deflt = -1000;	AMaViS rates every mail with a "spam score". Usually spam has levels of 5-10. You will get *"X-Spam-Status"* header lines added to the email if the spam score is above this value. I set it to -1000 because I want all mail to get the header lines.
	If the spam score is higher than this value then the mail will be flagged as spam. This is

$sa_tag2_level_deflt = 5.0;	done by adding a header line *"X-Spam-Status: Yes"*.
$sa_kill_level_deflt = 10;	If the spam score is higher than this value AMaViS starts to take action on the mail. The action is defined by $final_spam_destiny.
$sa_spam_subject_tag = '***SPAM*** ';	If you like to have the email subject altered to make spam mails more visible then you can have it rewritten. This string will be added if spam is detected.
@av_scanners = (...	This is a section where you can configure one or more virus scanner. Many common scanners are already preconfigured and just need to have the comment sign ('#') removed from the lines. You are also free to add your own command-line virus scanner here. For the beginning I suggest you just enable ClamAV and comment out all other entries. You may wonder why clamav is mentioned twice. Well, the @av_scanners' entry is about the daemonized version (clamd) and the @av_scanners_backup's entry runs the command-line version (clamscan).
$sa_local_tests_only = 0;	Unless your email gateway is already completely overloaded I strongly recommend you enable *remote tests*. Postfix can only check if the IP address of the system it just received an email from is in a blacklist. SpamAssassin also does this for every host that the email went through. This increases your chance of finding out if the email went through a known spam relay. It also checks if URLs that are mentioned in the email are blacklisted. All this is done automatically if you do not restrict your server to local tests only.

Run **adduser clamav amavis** if you intend to use the *clamd* (the daemon variant of the ClamAV scanner). In any case remember to restart the amavis service to activate your changes (**/etc/init.d/amavis restart**).

Tell Postfix to use AMaViS

Now that the amavis daemon is hopefully running in the background you still need to tell Postfix that you want to have all your emails scanned. This is done by setting the global content filter in your */etc/postfix/main.cf* like this:

```
content_filter = amavis:[127.0.0.1]:10024
receive_override_options = no_address_mappings
```

The *content_filter* entry makes all email be sent to a service called *amavis* which you will define in the next step. Next edit the */etc/postfix/master.cf* and add these two services:

```
amavis unix - - - - 2 smtp
    -o smtp_data_done_timeout=1200
    -o smtp_send_xforward_command=yes

127.0.0.1:10025 inet n - - - - smtpd
    -o content_filter=
    -o local_recipient_maps=
    -o relay_recipient_maps=
    -o smtpd_restriction_classes=
    -o smtpd_client_restrictions=
    -o smtpd_helo_restrictions=
    -o smtpd_sender_restrictions=
    -o smtpd_recipient_restrictions=permit_mynetworks,reject
    -o mynetworks=127.0.0.0/8
    -o strict_rfc821_envelopes=yes
    -o receive_override_options=no_unknown_recipient_checks,no_header_body_checks
```

In the past you may have set up a **pre-cleanup** service. And perhaps the AMaVis documentation still tells you to set up such a service. This is deprecated.

Finally run a **postfix reload** and **postfix check** to make sure you have still no errors in your Postfix configuration. All incoming emails should now be tested for viruses and spam. Look at your */var/log/mail.log* file for details. A great service for checking AMaViS is provided by **webmail.us (http://www.rackspace.com/apps/email_hosting/)** . They are sending you harmless test mails with the EICAR virus test signature.

How does the content filtering work?

Let me explain how Postfix and AMaViS work together. Postfix receives your email and sends it to AMaViS as defined by your *content_filter*. As the *no_address_mappings* option is set during this first stage, aliases are ignored. AMaViS will then use the configured virus scanner to scan all attachments and spamassassin to check the email for certain spam criteria. If the attachments are archives (like .zip or .tar.gz) it will use the archive tools to unpack them. After scanning the email it will be sent back to Postfix on localhost's port 10025. But this time it sets the *content_filter* option to nothing thus bypassing further content scanning. And it clears the *no_address_mappings* option so aliases are now applied. If you would not disable the aliases during the first stage the aliases were applied twice which leads to getting emails twice. So we generally ignore the aliases at first and only use the aliases in the Postfix service that gets the emails back from AMaViS.

You can see these two steps in your */var/log/mail.log* file. First you will notice a "relay=amavis" and then a "relay=virtual". If everything worked you will see a line like this in the */var/log/mail.log:*

```
amavis[677]: (00677-02) Passed, <my@test.address> -> <user@virtual.test>, ...
Message-ID: <20040716231708.8717C1C21E@myserver>, Hits: -
```

If you catch viruses (see **eicar.org (http://eicar.org/)** for a harmless test signature) you will see lines like this in the */var/log/mail.log:*

```
amavis[11843]: (11843-01) INFECTED (Eicar-Test-Signature), <user@domain1> -> <user@domain2>...
```

Train it with ham and spam

To make AMaViS even more effective you should tell it what you think is spam. Since SpamAssassin is used for the bayesian filtering (it applies probabilities to every word to find out if the email is likely to be spam) it needs some training. You need to show it a large amount (more than a few hundred) mails of *both* categories. You cannot just show it spam mails and expect it to work.

Just a note: I know that it's possible to store these settings individually for each virtual user. Honestly I have not yet tried that since I found the documentation of AMaViS a bit confusing. Contributions on storing the learned information in the database per-user are welcome.

The tricky part is that you need sufficient access permissions to read the users' emails while at the same time storing the learned information in the home directory of the 'amavis' user. The permissions only allow the user 'vmail' to access emails. The group (g) and others (o) do not have any permissions. So I decided to run the learning task as 'root' from cron.

Just create a batch script that reads certain folders consisting of spam emails and have it called by crontab ('crontab -e' as root). I have hacked a script together and put it at /usr/local/bin/spamlearner:

```
#!/bin/bash -e

SADIR=/var/lib/amavis/.spamassassin
DBPATH=/var/lib/amavis/.spamassassin/bayes
SPAMFOLDERS="\
         /home/vmail/well-known-customer.com/fred/.spam/cur \
         /home/vmail/spamtrap.org/jeanne/.spam/cur \
         "
HAMFOLDERS="\
         /home/vmail/spamtrap.org/jeanne/.trash/cur \
         "

for spamfolder in $SPAMFOLDERS ; do \
         echo Learning spam from $spamfolder ; \
         nice sa-learn --spam --showdots --dbpath $DBPATH $spamfolder
done

for hamfolder in $HAMFOLDERS ; do \
         echo Learning ham from $hamfolder ; \
         nice sa-learn --ham --showdots --dbpath $DBPATH $hamfolder
done

chown -R amavis:amavis $SADIR
```

You can see what AMaViS has recorded into its database by running **sa-learn --dbpath /var/lib/amavis/.spamassassin/bayes --dump magic** .

A note: this way you are training AMaViS' *global* bayesian database. So it applies to all emails that enter your system. Make sure you don't learn spam mails from unreliable or untrusted users. They can ruin your spam detection ratio by putting messages in their 'spam' folder that are not really spam. Just because a user never wants to get email from his ex-girlfriend again does not make it spam for everybody. So be careful about the mails you feed to the global bayes database.

Another note: Do not make SpamAssassin learn emails that users just 'forwarded' to you. The header information would be gone which contains much of the information that helps SpamAssassin recognise spam. Make sure they use the 'bounce' feature. Your best choice is to take the email messages directly from the user's maildir directory.

Offering webmail access (optional)

Now that you know how your users can get and send their emails via POP3, IMAP and SMTP you may wonder if there is a comfy way to give your users access to their mailbox using webmail. Luckily this is easy. The package you need is called "squirrelmail" - a web mail system that can use any IMAP server - just like the Courier IMAP server we use here.

Install it using **apt-get install squirrelmail**. To set it up you first need to add its configuration to your Apache configuration:

```
ln -s /etc/squirrelmail/apache.conf /etc/apache2/conf.d/squirrelmail.conf
```

Then run **squirrelmail-configure** to configure the basic settings. The default settings should suffice for a test. You can reconfigure it later at any point if you like.

Now access your web server like http://hostname/squirrelmail and you should get a login dialog. As usual use the email address as the username and the appropriate password from the *`users`* table. Squirrelmail will then contact the IMAP server on 'localhost' and show your emails.

Mailing lists with mailman

A frequently asked question is how to run mailing lists with virtual domains using the famous 'mailman' software. Actually this is not very hard. There are three ways to accomplish that.

Forwarding to local pipes

This first approach will use the aliases that you usually insert into **/etc/aliases**. You just have to understand that virtual users cannot have piped aliases. Piped... what? Well, once you have created a mailing list with the **newlist** command you are requested to add certain aliases to your /etc/aliases file like chitchat-admin, chitchat-bounces, chitchat-join and so on that point to destinations that look like **"|/var/lib/mailman/mail/mailman post powerdns-debian"**. This means that the email is piped into the **mailman** program. Mailman is called and gets the email as input.

This works well for local domains but will fail if you put a piped alias like this into a virtual alias. That is because you do not have a system user whose user-id you could use to run this pipe. So you will have to make a circuit and forward your virtual email address to a local email address where you can use piped aliases.

I recommend that you use 'localhost' as your local domain (**mydestination = localhost**) and forward each of the mailman aliases to the @localhost equivalent. So if you have a chitchat-subscribe@virtual.domain address you just forward it to chitchat-subscribe@localhost and use the /etc/aliases as suggested by 'mailman'. Perhaps someone comes up with a nifty solution to put the /etc/aliases into the MySQL database, too. Let me know if you have invented something. :)

Generic approach using regular expressions

There is a simple way to use mailman-based mailing lists when you are ready to sacrifice a hostname for mailinglists. Assume your domain is *domain.com*. You would then use *lists.domain.com* as the **dedicated server** (http://www.webhostingsearch.com/dedicated-server.php) name for mailing lists. Please see: **http://listes.rezo.net/how.php** (http://listes.rezo.net/how.php) .

Using mailman as a transport service

The third way to use mailman is to create a transport (one of those services in **/etc/postfix/master.cf** for mailman. Please see the file **/etc/mailman/postfix-to-mailman.py** and read the *INSTALLATION* section of it.

Troubleshooting

Error messages

Log file	Error message	Meaning
/var/log/mail.log	postfix/smtpd[11960]: NOQUEUE: reject: RCPT from myserver[10.20.30.40]: 550 <nonexisting@virtual.test>: Recipient address rejected: User unknown in virtual mailbox table; from=<user@domain> to= <nonexisting@virtual.test> ...to= <user2@my.testdomain>, relay=none, delay=0, status=bounced (unknown user: "user2@my.testdomain")	Postfix knows that my.testdomain is a virtual domain. But the user account "user2@my.testdomain" was not found in the 'users' table.
/var/log/mail.log	...warning: connect to mysql server mailtest: Access denied for user: 'service@myserver' (Using password: YES)	Postfix tried to read from the MySQL database using the user service@myserver. However the combination of username and password have been rejected by MySQL.
/var/log/mail.log	...warning: connect to mysql server mailtest: Access denied for user 'provider_admin'@'localhost.localdomain' (using password: YES)	You seem to have an entry **localhost.localdomain** in your **/etc/hosts** file. Remove it. (Note: this has been discussed on the debian-devel mailing list. It was said that **localhost.localdomain** is correct. So you can as well grant MySQL privileges on **localhost.localdomain** instead of **localhost**. MySQL

		5.0 will understand that these two host entries are identical.)
/var/log/mail.log	Clam Antivirus-clamd FAILED - unknown status: /var/lib/amavis/amavis-20050925T193330-15533/parts: Access denied. ERROR	You need to add the user *clamav* to the group *amavis* if you want to use the **clamd** (ClamAV daemon). Use this command: **adduser clamav amavis**
Output from **postfix check**	postfix/postfix-script: warning: /var/spool/postfix/etc/hosts and /etc/hosts differ	A file in the chroot jail in */var/spool/postfix/etc* differs from the files in */etc*. Just restart the postfix service and these files will be copied into the jail.

MySQL debugging

In some cases it may happen that Postfix cannot even read from the MySQL database. If you suspect this you may want to take a look at the */var/log/mysql/mysql.log* file. It contains all SQL queries to the database.

Getting help on IRC

A lot of smart users can be found on the IRC channels *#postfix* and *#postfix-de* (German speaking) in the freenode.net network. You are invited to join us. I usually attend there as "Signum". Please do not message me privately. Just join the channel and ask your question as precisely as possible. And though I am flattered that so many people use my tutorial I cannot guarantee any reaction times if you send me an email.

Migrating from the previous version of the tutorial

If you have already used an earlier version of this tutorial and want to adopt the changes then move your mail directories so that /home/vmail/user@domain becomes /home/vmail/domain/user.

Server-side mail filtering using maildrop

A frequently asked question from our readers deals with server-side filtering. Many mail services offer to automatically sort spam emails into a seperate mail folder. You may even be used to 'procmail' which is a handy mail filter for local accounts on a system. Unfortunately it cannot work with virtual mailboxes. Most administrators appear to use 'maildrop' for this purpose. 'maildrop' is part of the Courier mail server and a Debian package exists already. It will not work as expected though because the 'maildrop' version in Debian/Sarge does not have MySQL support compiled in. As 'maildrop' would replace Postfix' built-in 'virtual' delivery agent it needed to know about the location of the virtual users' mailboxes. And it needed to do MySQL lookups for that purpose. If you really need this feature then consider reading **Jasper Slits' page (http://www.xs4all.nl/%7Ejaspersl/quota/)** on this topic. He gives instructions on how to compile a 'maildrop' binary that is MySQL-enabled. It will also help you configure mail quotas.

My personal opinion on 'maildrop' is: forget about it if you can. It's nearly impossible to debug and more a dirty hack than a reasonable delivery agent. You may see warnings in your logs that are internally a 'segmentation fault' of the program itself that has just been hidden from you. Unfortunately there aren't any other delivery agents as far as I know.

If you are still happy with 'maildrop' then you are probably happy to hear that the next stable Debian release - Codename 'Etch' - will contain the new version of 'maildrop' that is using Courier's 'authlib' authentication library. You won't need to compile anything then.

Thanks

A lot of people have mailed me for suggestions. I would like to thank the following people for providing valuable additions to this tutorial.

- Alexander Stielau
- Christian Garling
- Christian Kurz
- Christof Gruber
- Daniel Hackenberg
- Daniel Wladow
- Eicke Kemm
- Jesper Krogh
- Jon Cox
- Kay-Michael Voit
- Lucas Baltes
- Mario Duve
- Mikael Tokarev
- Patrick Blitz
- Patrick Jäger
- Pete Boyd
- Ricardo Arguello

- Simon Kammerer
- Thomas "Balu" Walter
- Tim Weippert

Greetings to the fine people on #postfix at freenode.net and of course thanks to my wife who showed a lot of patience with me.

Planned for the Etch tutorial

- Using the MySQL connection socket using a nifty /etc/fstab entry with a binding mount.
- Switch (back) to encrypted passwords for security reasons. (courier-authdaemon with "-r imap" option)
- Web interface to manage the accounts.
- A proper vacation/autoresponder solution.

Contributions from readers

A lot of users have sent feedback that constantly helped improving this document. Also thanks to all readers who sent feedback on the tutorial - both technical and emotional.
(http://workaround.org/moin/PostfixTutorialFeedback)

127354 reads

7 Comments

Used this doc in Ubuntu Lucid LTS

Submitted by Anonymous on Wed, 08/18/2010 - 22:12

I used this document and the problem I ran into after installing on a fresh ubuntu system Lucid LTS was the following:

needed to install package

apt-get install courier-authlib-mysql

Things still not doing well following this document is SPAM Messages are still get through to the inboxes.

and AmAvis paramenters are to be applied to the middle of */etc/amavis/conf.d/50-user and not file /etc/amavis/amavisd.conf*

Otherwise this ia great documentation for a home mailserver.

HELP! HELP! HELP!

Submitted by Russell Hunt (http://www.russ-web.co.uk) on Tue, 09/28/2010 - 23:44

I am so stuck, and I have searched the internet for hours.

Last night I could send emails to hotmail, gmail and yahoo no problem from my e-mail server. Today they are being ditched into the junk mail folders of these ISPs
, I can send logs and config files, or if someone with the know how is really willing to help I am willing to let you have at my server to have a look at whats wrong. I have tried everything to get this to work.

I am not on any blacklists, or greylists and I have a valid SPF file. Is it an absolute MUST to have reverse DNS for an IP address in order to get hotmail to not junk my emails?

Thanks in advance

Are the other ISPs rejecting

Submitted by Christoph Haas on Tue, 09/28/2010 - 23:51

Are the other ISPs rejecting your email or just putting your mails into the junk folders?

Things I'd check if I were you:

- several RBLs (http://rbl-check.org/)
- your SPF entry if you have one (http://www.kitterman.com/spf/validate.html)
- proper DKIM signature if you use DKIM (http://www.myiptest.com/staticpages/index.php/DomainKeys-DKIM-SPF-Validator-test)
- run an example email through 'spamassassin' (best done with the email when received on the remote side so that the Received: lines can be checked)

Help Required Urgently

Submitted by Rubal on Tue, 09/27/2011 - 11:22

Hi

Followed your tutorial completely step by step. At the end i am able to recive mails from anywhere but i cannot send mail to any other domain outside the list of virtual domains o my system. When i tried to send the mail to other domain using telnet i recived error as:

telnet localhost 25
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 test1.example.com ESMTP Postfix (Microsoft Windows Server 2008)
ehlo testubuntu.com
250-test1.example.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH CRAM-MD5 PLAIN DIGEST-MD5 LOGIN
250-AUTH=CRAM-MD5 PLAIN DIGEST-MD5 LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
mail from:<user@testubuntu.com>
250 2.1.0 Ok
rcpt to:<rubal033@yahoo.com>
554 5.7.1 <rubal033@yahoo.com>: Relay access denied

How can i send mail to outside domains???
Please help. Its urgent.

Thanks in advance
Rubal

Sarge? Relaying.

Submitted by Christoph Haas on Tue, 09/27/2011 - 15:38

1. I'm almost sure that "Sarge" is not your distribution any more.

-ERR [IN-USE] Temporary authentication failure.

Submitted by **Rhyan (http://malek.co)** on **Tue, 01/24/2012 - 06:05**

good day,

when i trying to connect pop3 in terminal. i get this error.

root@malek:/home/rhyan# telnet localhost pop3
Trying 127.0.0.1...
Connected to malek.co.
Escape character is '^]'.
+OK Dovecot ready.
user rhyan@malek.co
+OK
pass diansay
-ERR [IN-USE] Temporary authentication failure.

im using debian 6

any one can help me?

Please check your

Submitted by **Christoph Haas** on **Tue, 01/24/2012 - 12:19**

Please check your /var/log/mail.log to find the cause.

About the author

I am a system administrator and programmer. In my nerdy spare time I work on web applications, Python and Ruby programs, write articles or learn new technology. On workaround.org you can find problems, solutions and hints on my findings and get help. Of course your feedback is as welcome as any donation. :) *<Christoph Haas>*