Mail servers: resolving the identity crisis

How to get Dspam, Postfix, and Procmail to play well together

John Locke

Dspam (http://nuclearelephant.com/projects/dspam) filters spam with the best. In my installation, it stops over 98% of all spam: I've only had one false positive in the last year, and that was a message to the Dspam list that contained a real spam!

Administering Dspam is a breeze. No rules to configure, new users can automatically benefit from a global dictionary and quarantine management is simple. But getting a Dspam quarantine set up the first time, without losing any email, can challenge the most seasoned mail administrators.

In this article, I'm going to trace email through the various parts of a Postfix mail server running Dspam and Procmail or Maildrop, focusing on getting the CGI quarantine working correctly. I'll provide the configuration steps for local users. I'll also explain where the configuration differs for virtual users, and the basic changes or decisions that you need to consider, but I won't provide the actual configuration for virtual users in this article.

The ingredients

There are many different recipes for putting together a mail server on Unix. I'm going to use these programs because they're as good, or better than, anything else available.

First, install all of this software according to your distribution. In most cases, you should be able to install Postfix, Apache, mod_suexec, and Procmail through your distribution's package management system. Dspam and Maildrop require custom configuration at compile time, so they usually need to be compiled from source code.

Tab. 1: Programs discussed in this article

Program	Role	User IDs
Postfix	Mail Transfer Agent (MTA)	Root Postfix
Dspam	Spam filter	Actual User ID mail dspamcgi
Apache with mod_suexec	Quarantine	Dspamcgi apache/nobody
Procmail Maildrop	Local Delivery Agent (LDA)	Root Actual User ID Virtual User ID

Getting Postfix running

Administering Postfix, or any other MTA, is the subject of many a book and it's hard to do justice to the topic in a few hundred words. Before even attempting to get Dspam working, you should have Postfix up and running and successfully delivering to your mailboxes. You should have Postfix rejecting mail for invalid email accounts — the days when a "catchall" account caught anything other than spam are long gone. Besides, it's much better to have real email

users learn immediately when they've sent something to the wrong address, than for their message to get through and sit in quarantine with a few thousand spams.

Postfix is actually a collection of smaller programs, each dedicated to a particular task. One program talks to the outside world. Another cleans up message headers and determines the next place to direct a message. Others check that a message is valid.

Postfix comes with two different Local Delivery Agents (LDAs). These are the programs that actually deliver mail to a mailbox. The two LDAs are called Local and Virtual(8).

Local delivers mail to real Unix user accounts. When a remote mail server connects and attempts to send an email, Local can verify whether the corresponding local user account exists, allowing Postfix to accept or reject the message before even receiving it.

Virtual(8) can deliver mail to a set of virtual users (users who don't actually have a local Unix account); in the Postfix main configuration file, Virtual(8) uses directives starting with `virtual_mailbox`. There is also a Virtual(5), which is used to rewrite the destination address of an email before attempting delivery — this is entirely different, using `virtual_alias` directives in the Postfix configuration files. The numbers refer to the section of the man pages containing the documentation for the program. Virtual(8) uses a specified user id to deliver mail to any virtual user, and is limited to a single directory structure that you specify. Both Virtual(5) and Virtual(8) can be used to reject unknown users, exactly the same as Local.

Read the man pages to learn about these programs:

```
# man 8 local
# man 8 virtual
# man 5 virtual
```

Before adding any other programs to your mail server, get Postfix delivering mail to the correct place using one of its built-in LDAs. There are many How-Tos on the web to help you get this done — start with the links available at the Postfix (`http://www.postfix.org`) web site.

There are many different recipes for putting together a mail server on Unix. I'm going to use these programs because they're as good, or better than, anything else available

What happens when an email arrives

Postfix runs a process called Master, to start other processes as necessary. Master runs as root, and changes user ID, as appropriate, when starting its other processes. A process called Smtpd listens to port 25, running as the postfix user with very few permissions. Smtpd handles all communication with the remote mail software.

Before accepting any mail, Postfix runs through a series of checks. You can have it check the remote address, the validity of the server greeting, the sender address, and the recipient address for all kinds of different criteria (see the article by Kirk Strauser (`http://www.freesoftwaremagazine.com/free_issues/issue_02/focus_spam_postfix/`) in this magazine for more detailed information on Postfix). At a minimum, use the recipient checks to only allow mail going to or from known users — you don't want your server to become an *Open Relay*, a mail server that allows unknown people to send mail to anyone. Here's a reasonable set of recipient checks:

```
smtpd_recipient_restrictions =
  reject_non_fqdn_sender,
  reject_unknown_recipient_domain,
  reject_unauth_pipelining,
  reject_non_fqdn_recipient,
  permit_mynetworks,
  permit_sasl_authenticated,
  reject_unauth_destination,
  permit
```

In this example, the `permit_mynetworks` and `permit_sasl_authenticated` statements allow known users to send valid email anywhere. (Setting up SASL authentication is a topic for another article...) The key blocking directive here is `reject_unauth_destination`. This directive tells Postfix to see if it knows exactly where to deliver the message. It will check Local, Virtual(8), Virtual(5), its list

of relay domains, and a few other places to see if it can deliver the mail to a known location. If not, Smtpd returns a REJECT code, without ever accepting the message to be delivered.

Otherwise, Smtpd receives the message and hands it off to another internal Postfix process. After going through a number of processes, which you can configure for content filtering and other tasks, the incoming mail eventually reaches the point where it's going to be delivered to a mailbox. The recipient address is rewritten according to alias and virtual alias rules, and matched to a transport. The transport is used to determine which LDA to use: Local, Virtual(8), or another transport defined in the `/etc/postfix/master.cf` file. For a local Unix user, the mail is handed to the Local transport, started as the actual Unix user id. Local then uses the command specified in the `mailbox_command` directive to deliver the message to a mailbox.

Inserting Dspam or another LDA

There are basically three different places where you can insert Dspam into the Postfix mail delivery sequence:

1. As a content filter, before delivery.
2. As a transport.
3. As the mailbox command for the Local LDA.

It's theoretically possible to use the first option and insert Dspam as a content filter before even resolving the destination address. If you're deploying it on a relay server, this is a good option, but this goes beyond the scope of the article. Setting up Dspam this way gets pretty tricky, but it's possible to integrate with Amavisd-new along with an antivirus engine, configuring all users to share a single token dictionary.

The main issue with the other two options is that once Postfix delivers the message to Dspam, Dspam can't get the message back into Postfix for actual delivery. You can't use either of Postfix's native LDAs, Local or Virtual(8). This means that before deploying Dspam, you must get another LDA installed and working.

The same thing is true when you configure the Dspam quarantine — Dspam needs an LDA to deliver false positive messages (messages flagged as spam that are really innocent mail). If you use Dspam as a content filter, it's difficult to set up the quarantine — much easier to configure Dspam to add its header, deliver spam and innocent mail, and separate the spam at a later stage.

Fortunately, Procmail is a perfectly fine LDA included with most Linux distributions, and Maildrop is pretty easy to obtain and install. Which should you use? If you only have local users, Procmail is probably already on your machine and probably needs no further configuration. If you're trying to support virtual users, Maildrop is much easier to set up because it can look up virtual users from a number of different sources.

LDA user ids and permissions

If you understand Unix permissions, you probably know that only programs running as the root user can change user ids. When you start a program, it uses your user id to execute — unless it has a special bit set, the SetUID or SetGID bit. Programs that are SetUID and owned by the root user will run as the root user, regardless of which user account was used to start the program. Obviously, this can be a huge security risk, because if the program isn't carefully written to protect exploits, any user on the system could potentially gain root access.

Procmail is a perfectly fine LDA included with most Linux distributions, and Maildrop is pretty easy to obtain and install

When you use the Postfix Local LDA to deliver to a command, the command is started with the user id of the recipient. This works great for delivering mail, and adding Dspam to the mix doesn't cause any further problems with initial delivery. You can set Procmail as the delivery command by adding this directive to `/etc/postfix/main.cf`:

```
mailbox_command = /usr/bin/procmail
```

Naturally, if Procmail is in a different location, change the path as necessary.

The problem comes when you add delivery from the quarantine, or if you use the transport method to deliver to Dspam.

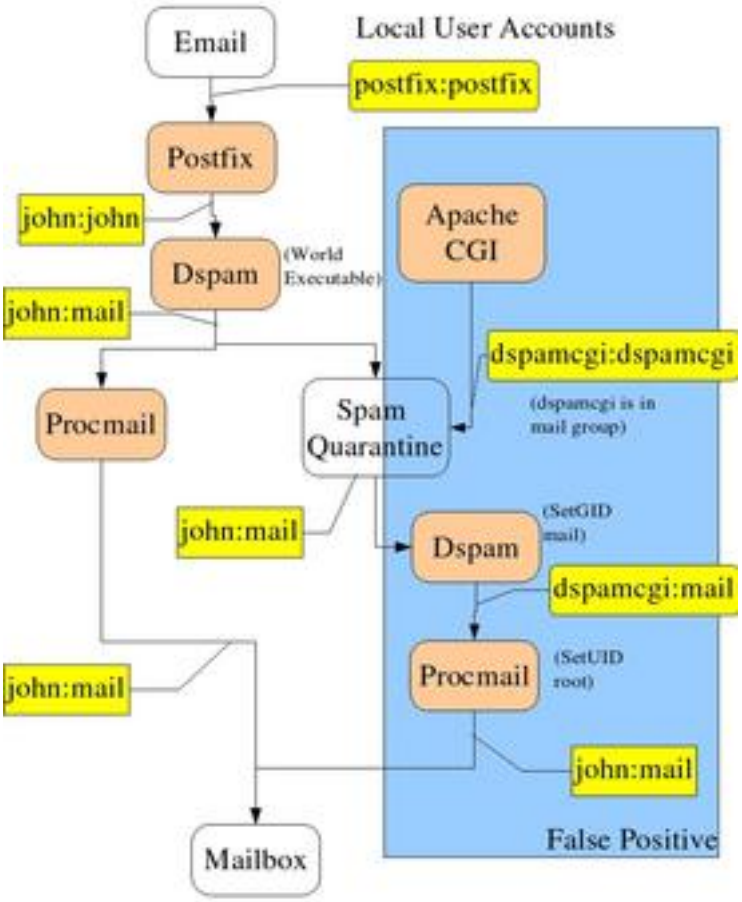


Fig. 1: To get Dspam and Procmail working correctly together, trace the execution of the various programs handling each email through the entire delivery process. Pay special attention to the effective user and group ids of each program.

In either case, Dspam will run under a different user account, *not* the user account of the user receiving the mail. At some point, the LDA needs to change to the correct user account to be able to deliver the mail. For it to be able to do this, it needs to be SetUID root.

Procmail delivery modes

Because Procmail often needs to be SetUID root, it limits the amount of time it spends as root, changing to the receiving user id as soon as it can. Postfix can call Procmail in either of two of its delivery modes: *Deliver* mode, or *Normal* mode.

In Normal mode, Procmail is called without any arguments, and delivers to the mailbox for the user id which is used to call Procmail. As shown, this works for normal delivery, even with Dspam. However, you cannot easily get Dspam to change to the real user account, when it's called by the quarantine CGI, so if you're going to use the quarantine, this mode won't meet your needs.

In Delivery mode, Procmail must be launched with the `-d` switch and a real user ID. For example:

```
/usr/bin/procmail -d john
```

For this to work, Procmail must either be SetUID root, or started by the root user. You can find out if Procmail is SetUID root by listing the Procmail binary, which should look like one of these:

```
# ls -l /usr/bin/procmail
-rwsr-sr-x 1 root mail /usr/bin/procmail
```

or

```
-rwsr-xr-x 1 root mail /usr/bin/procmail
```

To set Procmail to be SetUID root, do these commands:

```
# chown root /usr/bin/procmail
# chmod 4755 /usr/bin/procmail
```

Procmail allows any user account to use delivery mode, but requires that the destination user specified be a valid Unix user account.

Maildrop Trusted Users

If you choose to use Maildrop instead of Procmail, you still need to set it to SetUID root. You also need to specify at compile time which users are allowed to use its equivalent of delivery mode — Maildrop returns an error if an untrusted user tries to deliver to a different user account. The advantage of Maildrop is that you can specify a virtual user account to deliver to, and Maildrop will look up, in any one of several different locations, to determine where to deliver the message.

You can download Maildrop from here (`http://www.courier-mta.org/maildrop`).

Here's what my configure line for Maildrop looks like:

```
$ ./configure --enable-syslog=1 --enable-maildropmysql \
  --enable-trusted-users="root dspamcgi mail"
```

The crucial step here is the `--enable-trusted-users` flag. In my case, I've allowed three users to utilize Maildrop to deliver to other users: root, dspamcgi, and mail. I've specified the root user to make troubleshooting easier. Dspamcgi is a user account I created for the CGI quarantine — you'll need to create this user account before specifying it in the Maildrop configure line. Mail, in my case, is the Unix user account that I use to have Postfix deliver as a transport. I'll get to this in a little bit.

After the usual `./configure`, just run `make` and `make install`, and Maildrop should be ready to go. It should look like this:

```
$ ls -l /usr/local/bin/maildrop
-rwsr-xr-x 1 root mail 165616 Oct 4 07:45 /usr/local/bin/ma
```

You can configure Postfix to call Maildrop as a transport, or use the same `mailbox_command` directive as I used for Procmail:

```
mailbox_command = /usr/local/bin/maildrop
```

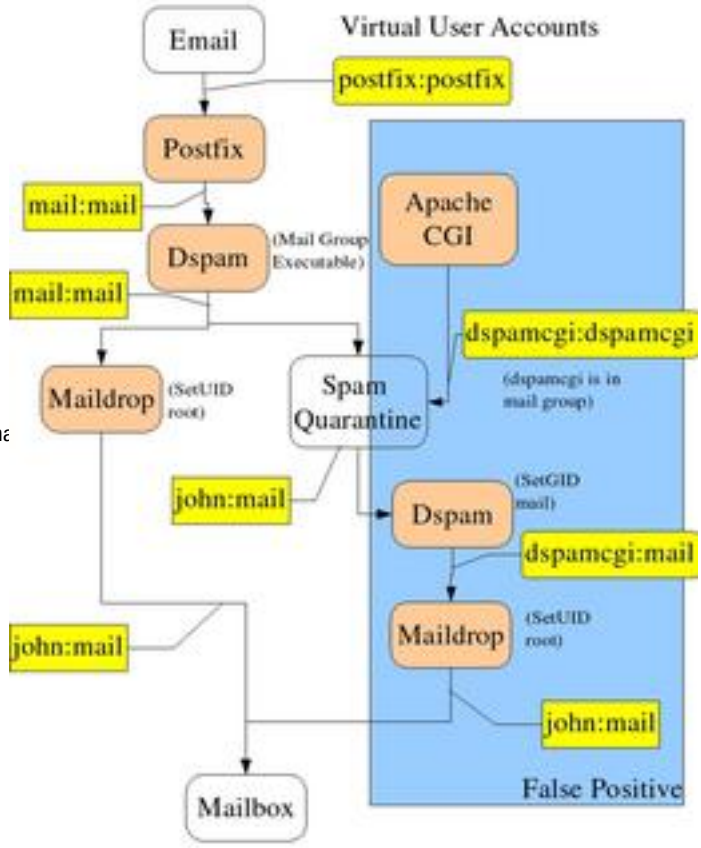
Adding Dspam to the chain

Before adding Dspam, get either Maildrop or Procmail working correctly. Verify that mail gets delivered to the right mailbox, and that invalid email addresses are rejected rather than bounced. Once you have an external LDA working correctly, adding Dspam is pretty simple. Download the most recent stable version from here (`http://nuclearelephant.com/projects/dspam`), and unpack the tarball.

You used to need to specify a delivery agent on the configure line, and make assorted setting in different files, but now most options are in the `dspam.conf` file.

Compile and install Dspam using the instructions in the README. The main decision you'll have to make is the storage driver. The MySQL driver is the most stable and best performing driver — if you have installed MySQL from your distribution sources, you'll also need to install the development packages.

Fig. 2: Maildrop can be configured exactly like Procmail. However, Maildrop can also deliver to virtual user accounts, mail accounts that do not exist as local Unix users. This diagram traces the flow of mail and ownership of processes when delivering to a virtual email account.



The README contains instructions for configuring Dspam as a transport in Postfix. To configure as a mailbox command for the Postfix Local LDA, read on.

When Postfix delivers to Dspam as a transport, you need to specify a particular Unix user account that Postfix can use to start Dspam. In my case, I use the Mail user. Whatever user you list here must be added to `dspam.conf` as a trusted user.

On the other hand, if Dspam is called from the Local LDA, it runs as a real Unix user, the user to be used for delivery. In this case, do not add individual users as trusted users `in dspam.conf` — they are not trusted users. Dspam will run under the specified user account, and use the `UntrustedDeliveryAgent` specified in

`dspam.conf` to deliver the message.

Unlike the LDA, Dspam does not need to be SetUID when used with Postfix. But if you're using the CGI quarantine, it does need to be SetGID.

> The advantage of Maildrop is that you can specify a virtual user account to deliver to, and Maildrop will look up, in any one of several different locations, to determine where to deliver the message

To get Dspam to process mail and hand off to Procmail as a mailbox_command (option #3), you need to make three changes:

- Change the permissions on the Dspam binary to world executable:

```
# chmod o+x /usr/local/bin/dspam
```

- Change the delivery agents in `Dspam.conf`:

```
UntrustedDeliveryAgent "/usr/bin/procmail"
TrustedDeliveryAgent "/usr/bin/procmail -d %u"
```

- Change the Postfix `mailbox_command` in `/etc/postfix/main.cf` to deliver to Dspam:

```
mailbox_command = /usr/local/bin/dspam
   --deliver=innocent --user $USER
```

To get Dspam to process mail as a transport (option #2) is a bit trickier. It involves setting up a Dspam transport in `/etc/postfix/master.cf`, adding a transport map that tells Postfix which addresses to deliver to the Dspam transport, and setting the `destination_recipient_limit` for the Dspam transport to 1.

Dspam Binary Permissions

The CGI quarantine for Dspam runs under the web server. Dspam creates a bunch of data files under its `$DSPAM_HOME`, by default `/var/lib/dspam`. Because Dspam can run under several different user accounts, and Dspam needs to write to many of these files as different users, Dspam sets all of these files as group writeable. To make this work, Dspam must always run under the same group ID, regardless of its user id. By default, Dspam installs itself SetGID to the mail group, so it should work for you correctly.

As I noted above, if you use the `mailbox_command` method of launching Dspam, untrusted users will be calling Dspam, so Dspam needs to be world executable. If you use transport mode, Dspam will always be called by the user you specify in the transport, in `/etc/postfix/master.cf`, so Dspam doesn't need to be world executable. But in this case, you need to add all of the user accounts that call Dspam to whatever group Dspam is set to use.

Finally, if you use Procmail and the CGI quarantine, either Dspam or Procmail needs to be SetUID root.

Training Aliases

Dspam comes with a handy tool to generate spam training aliases, called `dspam_genaliases`. It parses `/etc/passwd`, finds all valid users, and outputs a list of users prefixed with "spam-". You can write this output straight to a file and set Postfix to use it. To get it to work for all users, do something like this:

```
# dspam_genaliases --minuid 500 >
    /etc/postfix/dspam.aliases
# postalias /etc/postfix/dspam.aliases
```

Then add the file to the Postfix `alias_maps` directive in `/etc/postfix/main.cf`:

```
alias_maps = hash:/etc/aliases hash:
             /etc/postfix/dspam.aliases
```

Remember, you'll need to run this whenever you add or delete a user from the system.

This method becomes much more cumbersome if you support virtual users. You can use a regular expression transport map and a training transport in Postfix to avoid having to generate an alias file for virtual users, but you can't easily have the "spam" part added to the front of the address. Search the Dspam mailing list (via Google) to find out how.

Tab. 2: Dspam Binary Permissions

Postfix configuration	Local Delivery Agent	Dspam permissions	Command to set permissions
mailbox_command	Procmail, SetUID Root Maildrop, SetUID Root	-r-x--s--x root mail	# chmod 2511
mailbox_command	Procmail, not SetUID Root	-r-s--s--x root mail	# chmod 6511
Transport	Procmail, SetUID Root Maildrop, SetUID Root	-r-x--s--- root mail (all delivery users must be added to Unix mail group and Dspam Trusted Users)	# chmod 2510

Apache User and Group id

When you set up the Dspam CGI using the Apache web server, the CGI will always execute under a single user id. In order to deliver false positives to the correct user account, the LDA needs to change to the correct user id. Either Dspam or the LDA must be SetUID root to allow this context switch.

Dspam, Procmail, and Maildrop all have schemes to limit the exposure of running SetUID root, and SetGID in the case of Dspam. Because the CGI needs to be trusted to deliver mail to other user ids on the computer, you should do what you can to lock down Apache.

If the only reason you're running Apache is to provide access to the CGI quarantine, you can probably set it to a fairly standard configuration and have it be reasonably secure. If the CGI shares Apache with other web applications, you should install the mod_suexec Apache module and run it under its own user id. Before I get into actual Apache configuration, a quick note about user accounts.

Depending on your distribution and how Apache was installed, it may run by default as the nobody user, a user called apache, or perhaps something else like www or web. Exactly what user account Apache utilizes doesn't matter that much, but the effective user id can — on many systems, accounts with a user id greater than 65000 will not switch user context, even when executing a SetUID or SetGID program. The nobody account is often set to a number above this, limiting the damage that can be done if Apache or another process running as nobody is compromised.

Set up a real Unix user account with limited permissions for the Dspam CGI. On my system, I created a user account called dspamcgi. If the CGI is the only use of Apache on your system, you can simply set Apache to your user account with these two directives in `httpd.conf`:

```
User dspamcgi
Group dspamcgi
```

Whatever user you list here needs to be added to these places:

- `dspam.conf` Trust directive
- Unix group that Dspam is SetGID to, in `/etc/group` (the mail group in the default install)
- Maildrop's `trusted-users` configure switch.

Editing an htpasswd file every time you add a user to the system can quickly become a major hassle. There are several approaches to managing Apache authentication

CGI Setup

Besides creating a user account for the CGI script, you'll need to add directives to set Apache to execute the actual scripts. Copy the contents of the cgi directory from the Dspam source directory to a permanent place in the file system. I use `/var/www/dspam`. Then set up the appropriate alias and directory directives. For example:

```
Alias /dspam/base.css /var/www/dspam/base.css
Alias /dspam/logo.gif /var/www/dspam/logo.gif
Alias /dspam/dspam-logo-small.gif
   /var/www/dspam/dspam-logo-small.gif
ScriptAlias /dspam/ /var/www/dspam/
<Directory "/var/www/dspam">
Options +ExecCGI
  AuthName "DSPAM Quarantine Area";
  AuthType Basic
  Require valid-user
  Order Deny,allow
  Allow from all
</Directory>
```

The above settings make Apache find the scripts, stylesheet, and logos, and execute them using CGI. The `Order` and `Allow` directives relax the default IP address access controls to allow people to connect from any location. The `Auth*` directives set up authentication — but I haven't chosen an authentication method yet. The simplest way to get started is use the mod_auth module, included with the base Apache configuration. Create a password file using the htpasswd command:

```
# htpasswd -c /var/www/conf/htpasswd john
# htpasswd /var/www/conf/htpasswd seconduser
```

The first command, with the `-c` switch, creates the password file. The next line adds a user to an existing file. Then add the `AuthUserFile` directive to the `<Directory>` block above, near the other `Auth*` files:

```
AuthUserFile /var/www/conf/htpasswd
```

Once Apache is configured, check the configuration of the actual CGI. Add your user account to the admins file in the CGI directory, and edit the `configure.pl` file for the domain name of the web server and anything else custom for your configuration. Start Apache, and you should be in business!

Automatic CGI authentication

Editing an htpasswd file every time you add a user to the system can quickly become a major hassle. There are several approaches to managing Apache authentication. I recommend using mod_auth_shadow for local users, and mod_auth_mysql for virtual users. Both are available at SourceForge:

- mod_auth_shadow (`http://sourceforge.net/projects/mod-auth-shadow/`)
- mod_auth_mysql (`http://modauthmysql.sourceforge.net/`)

Both of these are Apache authentication modules, which are not included in the base Apache release. They can be easily added to an existing Apache installation. If you've compiled Apache from source code, you'll have everything you need. If you've installed from your distribution's package file, you may need to install the apache-devel or apache2-devel packages. Both of these modules use a script called apxs or apxs2 in the development packages or full source package to compile and load correctly.

mod_auth_shadow provides a way to authenticate against the built-in Unix user and group files. Modify the Makefile in the source distribution to point to the correct apxs script, and do:

```
# make all
# make install
```

You may need to add the module in the Apache configuration. To use mod_auth_shadow, simply change the `AuthUserFile` directive to:

```
AuthShadow on
```

mod_auth_mysql takes a bit more configuration, specifying the database, user, password, and several other settings. Read the documentation for more help.

Using a dedicated Virtual Host with suexec

The Apache suexec module makes it possible to specify a different Unix user id and group for different virtual hosts. If you run several web applications on the same server, you can limit access to the SetUID programs involved in Dspam by configuring a special virtual host to run as the trusted user.

Other virtual hosts on the same computer may still be able to access the programs, but if they're not trusted users, each program has a way to limit the risk associated with running SetUID root.

Dspam comes with a handy tool to generate spam training aliases, called dspam_genaliases. It parses /etc/passwd, finds all valid users, and outputs a list of users prefixed with "spam-"

To make your Dspam CGI run under Apache's mod_suexec, you'll need to install mod_suexec, specify the user id and group id to run as, and follow all the rules that suexec enforces to get your script to run.

Before you roll out mod_suexec, let me warn you about something. I personally make the virtual host for the Dspam CGI run under SSL encryption. The problem is, I have several web applications all running under SSL, but only one IP address assigned to the box. Because you can only have one SSL-encrypted virtual host per IP address, setting up Dspam using suexec on this virtual host means all the other web applications on that site must be set up under the same user accounts, and following all the suexec rules. The main one that has affected me is that suexec does not allow symbolic links — if you visit a URL that is actually a symbolic link to a script, suexec will not allow the script to execute and you'll get an error page.

That said, here's what needs to change to get mod_suexec to work:

1. Install mod_suexec. If you've installed Apache from your distribution's package system, you can probably install the appropriate RPM. If you've installed from source, try this configure line for Apache (make sure you have an Apache user, or change to reflect the main user account to run Apache):

```
./configure --prefix=/usr/local/apache2 \
   --enable-mods-shared=most \
   --enable-suexec \
   --with-suexec-caller=apache \
   --with-suexec-docroot=/var/www \
```

1. Change the main user and group in `httpd.conf` to apache or appropriate.
2. Put the remaining configuration directives (`Alias`, `ScriptAlias`, <`Directory`> block) listed in the section above inside a <`VirtualHost`> block.
3. Add this to the <`VirtualHost`> block:

```
SuexecUserGroup dspamcgi dspamcgi
```

1. Change ownership of the cgi directory and all the scripts in it to match the SuexecUserGroup:

```
# chown dspamcgi:dspamcgi -R /var/www/dspam
```

1. Restart Apache, and you should be up and running!

Conclusion

I've covered a lot of ground in this article. Running a mail server is no easy feat, and getting Dspam and its quarantine working correctly without losing mail can challenge the most diligent administrator.

If you take the time to trace the user id and group id at each step of delivery, you can sort out the issues and get it all to work.

Good luck!

Copyright information

About the author

John Locke is the author of the book *Open Source Solutions for Small Business Problems.* He provides technology strategy and free software implementations for small and growing businesses in the Pacific Northwest through his business, Freelock LLC (`http://www.freelock.com/`).