

Serveur Mail Postfix - Dovecot - LDAP - MDS

Debian GNU/Linux



Matthieu Vogelweith
15 septembre 2009

Résumé

L'objectif de ce document est de détailler l'installation d'un serveur mail complet basé sur des briques libres dont la réputation n'est plus à faire : Postfix [1], Dovecot [2] et OpenLDAP [3]. Un paragraphe détaillera l'utilisation du module de messagerie du MDS (**M**andrive **D**irectory **S**erver [4]), qui simplifie grandement la gestion des utilisateurs virtuels LDAP. Enfin ce document présente de nombreux outils et astuces permettant de tester et évaluer le serveur mis en place.

Ci-dessous, la liste des fonctionnalités qui seront détaillées dans cette page :

- Gestion des services SMTP, SMTPS, IMAP et IMAPS ;
- Sécurisation de tous les protocoles de communication utilisés ;
- Gestion des utilisateurs virtuels dans un annuaire LDAP ;
- Gestion des domaines virtuels dans LDAP ;
- Gestion des quotas dans LDAP ;
- Gestion des alias et des groupes dans LDAP ;
- Gestion des filtres SIEVE ;
- Interface de gestion en AJAX ;
- Filtrage anti-Virus et anti-SPAM ;
- Accès distant par webmail.

Ce document a été rédigé en LaTeX en utilisant l'excellent Vim sous Debian GNU/Linux. Il est disponible aux formats XHTML et PDF. Les sources LaTeX sont disponibles ici : [L^AT_EX](#)

Licence

Copyright ©2009 Matthieu VOGELWEITH <matthieu@vogelweith.com>.

Vous avez le droit de copier, distribuer et/ou modifier ce document selon les termes de la GNU Free Documentation License, Version 1.3 ou ultérieure publiée par la Free Software Foundation ; avec aucune section inaltérable, aucun texte de première page de couverture, et aucun texte de dernière page de couverture. Une copie de la licence est disponible dans la page [GNU Free Documentation License](#).

Historique

- **07-09-2009** : Utilisation des backports pour Roundcube
- **07-08-2009** : Mise à jour pour Debian Lenny
- **12-01-2009** : Utilisation de postfix-add-filter pour gérer les outils de filtrage de contenu dans master.cf

Table des matières

Table des matières	4
1 Préparation	6
1.1 Pré requis	6
1.2 Configuration du gestionnaire de paquets	6
1.3 Annuaire OpenLDAP & MDS	7
1.4 Plugin mail du MDS	7
1.5 Création d'un domaine et d'un utilisateur	8
2 Serveur IMAP : Dovecot	9
2.1 Installation	9
2.2 Binding LDAP	10
2.3 Premier test	11
3 Serveur SMTP : Postfix	12
3.1 Installation	12
3.2 Configuration de base	12
3.3 Binding LDAP	13
3.4 Dovecot LDA	14
3.5 Premier test	15
4 Elements de sécurisation	16
4.1 Protection contre l'Open-Relay et le SPAM	16
4.2 Authentification SMTP : SASL	17
4.3 Certificats SSL	19
4.4 IMAP : Chiffrement avec OpenSSL	20
4.5 SMTP : Chiffrement avec OpenSSL	20
4.6 Configuration du firewall	21
5 Analyse des messages	22
5.1 Filtrage de contenu : Amavis	22
5.2 Filtrage des messages entrant uniquement	23
5.3 Intégration d'un Antivirus : ClamAv	24
5.4 Intégration d'un AntiSpam : Spamassassin	25
6 Gestion des quotas	28
6.1 Configuration du plugin	28
6.2 Définition des quotas	29
6.3 Quelques tests	29

7	Tri des messages coté serveur : SIEVE	30
7.1	Configuration du plugin	30
7.2	Filtres globaux	31
7.3	Édition des scripts : MANAGESIEVE	32
8	Apprentissage des SPAMS	33
8.1	Introduction	33
8.2	Cron d'apprentissage	33
8.3	Ajustements manuels	34
9	Support TLS pour les clients SMTP	35
9.1	Introduction	35
9.2	Configuration du client	35
9.3	Configuration du serveur	35
10	Envoi de mails depuis le réseau local	37
10.1	Présentation	37
10.2	Mise en place	37
11	Webmail : RoundCube	38
11.1	Installation	38
11.2	Configuration de RoundCube	38
11.3	Configuration d'apache	39
12	Références	41

Chapitre 1

Préparation

1.1 Pré requis

Ce document suppose que le serveur est déjà installé avec une Debian Lenny [5] propre. L'installation et la configuration du système de base sont présentées en détail dans un document dédié à cet effet : [6].

Par ailleurs, pour des raisons évidentes de sécurité, les différents services du serveur de messagerie utiliseront un utilisateur dédié pour les différentes opérations sur le système, notamment le stockage des messages. La création de cet utilisateur système, se fait tout simplement avec la commande suivante :

```
# adduser --system --ingroup mail --uid 500 vmail
```

1.2 Configuration du gestionnaire de paquets

Le serveur de mail qui va être installé fera du filtrage de contenu afin de détecter un maximum de messages indésirables ou infectés par des virus. Pour que ce filtrage soit le plus efficace possible, il est indispensable que les outils d'analyse de contenu soient très récents et mis à jour régulièrement. Pour cela il existe un miroir Debian spécifique qui fournit des versions récentes pour des paquets comme Amavis ou Spamassassin : Debian Volatile [7]. Pour obtenir les paquets disponibles dans ce miroir, ajouter simplement les lignes suivantes dans le fichier `/etc/apt/sources.list` :

```
# Debian Volatile
deb http://volatile.debian.org/debian-volatile lenny/volatile main contrib non-free
```

Comme indiqué précédemment, l'administration courante du serveur se fera avec le MDS. Pour obtenir les paquets, il faut également ajouter un dépôt externe dans le fichier `/etc/apt/sources.list` :

```
# Mandriva Directory Server
deb http://mds.mandriva.org/pub/mds/debian lenny main
```

Reste maintenant à mettre à jour la liste des paquets disponibles avec la commande suivante :

```
# aptitude update
```

1.3 Annuaire OpenLDAP & MDS

L'installation du serveur LDAP et la configuration de base du MDS faisant l'objet d'un document dédié [8], elles ne seront pas détaillées ici. Le plugin de messagerie du MDS sera cependant l'objet du paragraphe suivant.

1.4 Plugin mail du MDS

Le MDS possède un plugin permettant de gérer les champs LDAP relatifs à la messagerie. Notons que cette page propose d'utiliser le MDS pour la gestion de l'annuaire mais il est tout à fait possible de suivre cette documentation en éditant l'annuaire en utilisant d'autres outils : Le MDS apporte simplement un confort d'administration, il n'engendre aucune dépendance vis-à-vis d'une brique logicielle.

```
# aptitude install mmc-agent mmc-web-base mmc-web-mail python-mmc-mail slapd ldap-utils
```

Lors de l'installation des paquets, il suffit simplement de renseigner le mot de passe admin de l'annuaire LDAP. Ce mot de passe sera utilisé plus tard pour toutes les opérations d'écriture dans l'annuaire.

Les schémas proposés de base avec OpenLDAP ne permettent pas de gérer un serveur de mail complet. Il n'existe pour l'instant aucun schéma officiel pour la gestion avancée de la messagerie. Un schéma spécifique à la messagerie est donc fourni avec les paquets du MDS. Pour utiliser ce schéma, il faut dans un premier temps le copier dans le répertoire dédié à cet effet (ainsi que le schéma de base MMC si cela n'a pas été fait) :

```
# cp /usr/share/doc/python-mmc-base/contrib/ldap/mmc.schema /etc/ldap/schema/  
# cp /usr/share/doc/python-mmc-base/contrib/ldap/mail.schema /etc/ldap/schema/
```

Il faut maintenant configurer slapd pour qu'il utilise ces nouveaux schémas. Ajouter donc les lignes suivantes dans `/etc/ldap/slapd.conf` :

```
include /etc/ldap/schema/mmc.schema  
include /etc/ldap/schema/mail.schema
```

Et enfin redémarrer le service pour que les modifications soient prises en compte :

```
# /etc/init.d/slapd restart
```

Le serveur LDAP est maintenant prêt pour fournir les différents champs nécessaires aux serveurs de mails, il reste maintenant à configurer le plugin mail du MDS. Il faut notamment indiquer que l'on désire être en mode "domaines virtuels", mettre les différents domaines virtuels dans une OU spécifique et remplir le champ mailbox automatiquement. Cette configuration se fait dans le fichier `/etc/mmc/plugins/mail.ini` :

```
[main]  
vDomainSupport = 1  
vDomainDN = ou=mailDomains, dc=example, dc=org
```

```
[userdefault]
mailbox = /home/vmail/%uid%/
mailuserquota = 204800
```

Attention, dans l'exemple ci-dessus, les domaines virtuels sont stockés dans l'OU "ou=mailDomains, dc=example, dc=org". Cette OU est bien évidemment à adapter pour correspondre à la racine de l'annuaire LDAP présent sur la machine. De la même manière, il peut être nécessaire de modifier les variables "baseDN" et "password" du fichier /etc/mmc/plugins/base.ini pour correspondre à l'annuaire LDAP utilisé.

Lorsque ceci est réalisé, il reste à redémarrer l'agent MMC pour appliquer les modifications et vérifier que la structure de l'annuaire est correcte :

```
# /etc/init.d/mmc-agent restart
```

Toutes les branches LDAP nécessaires au fonctionnement du MDS sont automatiquement créées lors du redémarrage de l'agent, notamment la branche ou=mailDomains qui n'existait sûrement pas auparavant.

1.5 Création d'un domaine et d'un utilisateur

- Création du groupe d'utilisateur s'il n'existe pas
- Création d'un domaine de messagerie dans le MDS
- Création d'un premier utilisateur

L'annuaire LDAP et sa console de gestion sont maintenant opérationnels, le domaine virtuel de messagerie est créé ainsi qu'un utilisateur virtuel, reste à configurer le serveur de messagerie proprement dit ...

Chapitre 2

Serveur IMAP : Dovecot

Il existe de nombreux serveurs IMAP, mais celui qui sort du lot est à mon avis Dovecot [2]. Il est conçu en favorisant toujours les aspects de sécurité, est activement développé et implémente de nombreuses fonctionnalités comme :

- POP, POPS, IMAP et IMAPS ;
- Un démon d'authentification SASL commun pour tous les services ET utilisable par Postfix pour réaliser l'authentification SMTP ;
- Le support LDAP pour l'authentification ;
- Le support de nombreux formats de stockage des boîtes aux lettres, notamment Maildir ;
- Un Agent de delivery (MDA) supportant les filtres sieves et les quotas ;
- Un module permettant de faire du proxying IMAP.
- ...

2.1 Installation

La version packagée dans Lenny est encore dans la branche 1.0.x mais elle est stable, fonctionnelle et suffit largement pour fournir les fonctions de base du serveur de messagerie. Si besoin, une version plus récente est présente dans les backports. L'installation se fait simplement en installant le paquet `dovecot-imapd` :

```
# aptitude install dovecot-imapd
```

Le fichier de configuration principal de Dovecot est `/etc/dovecot/dovecot.conf`. Quasiment toute la configuration se fait dans ce fichier, mis à part pour l'authentification LDAP. Le fichier fourni avec les paquets Debian est très documenté. Dans un premier temps, il suffit de modifier quelques paramètres pour que le serveur soit fonctionnel :

```
protocols = imap
listen = 0.0.0.0
log_timestamp = "%Y-%m-%d %H:%M:%S "
login_greeting = Dovecot IMAP Server ready.
mail_location = maildir:~/Maildir

protocol imap {
    imap_client_workarounds = outlook-idle
}

auth default {
    mechanisms = plain login
```

```

#passdb pam {
# ...
#}
passdb ldap {
    args = /etc/dovecot/dovecot-ldap.conf
}
#userdb passwd {
# ...
#}
userdb ldap {
    args = /etc/dovecot/dovecot-ldap.conf
}
socket listen {
    master {
        path = /var/run/dovecot/auth-master
        mode = 0660
        user = vmail
        group = mail
    }
}
}

```

Avec cette configuration, le serveur ne proposera que le protocole IMAP à ses clients. On remarque également que la base d'authentification est configurée pour utiliser un annuaire LDAP.

Note : Pour que l'authentification SASL fonctionne correctement avec les clients Outlook, il faut impérativement ajouter "login" dans la liste des méthodes d'authentification supportées. Ci-dessus l'option "mechanisms" a été modifiée en conséquence.

2.2 Binding LDAP

La configuration de l'annuaire se fait tout simplement dans le fichier **/etc/dovecot/dovecot-ldap.conf**. Comme souvent, il faut indiquer le serveur à joindre, la base de l'annuaire et un mapping d'attribut pour que Dovecot puisse savoir quel est l'attribut qui correspond à l'adresse mail, au login, au mot de passe, ...

```

hosts = 127.0.0.1
auth_bind = yes
ldap_version = 3
base = ou=Users,dc=example,dc=org
user_attrs = mailbox=home
user_filter = (&(objectClass=mailAccount)(mail=%u)(mailenable=OK))
pass_attrs = mail=user,userPassword=password
pass_filter = (&(objectClass=mailAccount)(mail=%u)(mailenable=OK))
user_global_uid = vmail
user_global_gid = mail

```

La directive **auth_bind = yes** permet de faire le binding LDAP en utilisant le login et le mot de passe fournis par l'utilisateur. On n'est donc pas obligé de mettre le mot de passe admin en clair dans le fichier de configuration.

La configuration présentée ci-dessus utilise un uid commun (l'uid de l'utilisateur vmail) pour tous les utilisateurs. Ce mode de fonctionnement a deux avantages :

- L'agent de livraison (MDA) n'aura pas besoin des droits root pour écrire les mails sur le système de fichiers. (Pas de SetUID)
- On limite le nombre de requêtes vers l'annuaire LDAP.

Pour que Dovecot relise son fichier de configuration, il suffit d'exécuter la commande suivante avant de faire le premier test :

```
# /etc/init.d/dovecot restart
```

2.3 Premier test

Le serveur IMAP est maintenant fonctionnel, on peut par exemple tester l'authentification en faisant une connexion IMAP en telnet :

```
# aptitude install telnet
# telnet 127.0.0.1 143
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
* OK Dovecot ready.
. login user@example.org password
. OK Logged in.
. logout
* BYE Logging out
. OK Logout completed.
```

La séquence de commandes ci-dessus présente simplement une authentification réussie sur le serveur IMAP et une déconnexion. A ce stade, le serveur IMAP est donc fonctionnel mais encore loin d'être sécurisé.

Chapitre 3

Serveur SMTP : Postfix

3.1 Installation

L'installation du serveur SMTP proprement dite est très simple. Il suffit d'installer les paquets postfix et postfix-ldap. Pour faciliter les tests en ligne de commande, on peut également installer le paquet **mailx** qui fournira le binaire `/usr/bin/mail`.

```
# aptitude install postfix postfix-ldap mailx
```

3.2 Configuration de base

```
# Identification banner
smtpd_banner = $myhostname ESMTPEX $mail_name (Debian/GNU)
biff = no

# Basic settings
mail_name = Example Mail Server
myhostname = smtp.example.org
mydomain = example.org
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mydestination = smtp.example.org,smtp,localhost.localdomain,localhost
relayhost =
mynetworks = 127.0.0.0/8
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
luser_relay =

# Appending .domain is the MUA's job.
append_dot_mydomain = no
append_at_myorigin = yes

delay_warning_time = 4h
maximal_queue_lifetime = 10d
mailbox_size_limit = 0
message_size_limit = 15728640
```

3.3 Binding LDAP

Tous les fichiers de configuration nécessaires au binding LDAP sont fournis avec les paquets du MDS. Il suffit de les copier dans le répertoire de Postfix :

```
# cp /usr/share/doc/python-mmc-base/contrib/postfix/with-virtual-domains/ldap-* /etc/postfix/
```

Dans ces fichiers de configuration, le chemin de base LDAP utilisé est `dc=mandriva,dc=com`. Pour modifier tous les fichiers en une seule fois, vous pouvez utiliser la commande suivante en remplaçant `dc=example,dc=org` par le chemin de base utilisé dans votre annuaire LDAP.

```
# for i in `ls /etc/postfix/ldap-*.cf`; do sed -i s/dc=mandriva,dc=com/dc=example,dc=org/$i; done
```

Les fichiers qui viennent d'être copiés contiennent toutes les informations nécessaires pour que Postfix puisse interroger l'annuaire LDAP. Avec cette configuration, il est possible de stocker dans l'annuaire les noms de domaines virtuels, les comptes de messagerie, les alias, les quotas, les chemins des boîtes aux lettres, ...

Le fichier qui définit les domaines virtuels pris en charge est le fichier `ldap-domains.cf`. Il contient les informations suivantes :

```
server_host = 127.0.0.1
version = 3
search_base = ou=mailDomains,dc=example,dc=org
query_filter = (&(objectClass=mailDomain)(virtualdomain=%s))
result_attribute = virtualdomain
```

+ Dans `/etc/postfix/ldap-accounts.cf`

```
server_host = 127.0.0.1
version = 3
search_base = ou=Users,dc=example,dc=org
query_filter = (&(objectClass=mailAccount)(mailenable=OK)(mail=%s))
result_attribute = mailbox
```

+ Dans `/etc/postfix/ldap-maildrop.cf`

```
server_host = 127.0.0.1
version = 3
search_base = ou=Users,dc=example,dc=org
query_filter = (&(objectClass=mailAccount)(mailenable=OK)(mail=%s))
result_attribute = maildrop
```

+ Dans `/etc/postfix/ldap-aliases.cf`

```
server_host = 127.0.0.1
version = 3
search_base = ou=Users,dc=example,dc=org
query_filter = (&(objectClass=mailAccount)(mailalias=%s)(mailenable=OK))
result_attribute = mail
```

+ Dans /etc/postfix/ldap-transport.cf

```
server_host = 127.0.0.1
version = 3
search_base = ou=Users,dc=example,dc=org
query_filter = (&(objectClass=mailAccount)(mailalias=%s)(mailenable=OK))
result_attribute = mailhost
expansion_limit = 1
result_format = smtp:[%s]
```

+ Dans /etc/postfix/main.cf

```
# LDAP Transport
transport_map = ldap:/etc/postfix/ldap-transport.cf

# Virtual Domains Control
virtual_mailbox_domains = ldap:/etc/postfix/ldap-domains.cf
virtual_mailbox_maps = ldap:/etc/postfix/ldap-accounts.cf
virtual_mailbox_base =
virtual_alias_maps = ldap:/etc/postfix/ldap-aliases.cf, ldap:/etc/postfix/ldap-maildrop.
cf
virtual_alias_domains =
virtual_minimum_uid = 100
virtual_uid_maps = static:vmail
virtual_gid_maps = static:mail
```

3.4 Dovecot LDA

Postfix possède son propre agent de livraison (LDA ou MDA pour **L**ocal ou **M**ail **D**elivery **A**gent) mais cette documentation propose d'utiliser le LDA de Dovecot qui est beaucoup plus riche fonctionnellement : il permet notamment de gérer les quotas sur les Maildir et apporte le support des filtres Sieve.

Pour indiquer à Postfix que le nouvel agent de livraison pour les utilisateurs virtuels est Dovecot, il suffit d'ajouter les lignes suivantes dans /etc/postfix/main.cf :

```
# Dovecot LDA
virtual_transport = dovecot
dovecot_destination_recipient_limit = 1
```

Il faut maintenant définir le transport utilisé ci-dessus dans le fichier /etc/postfix/master.cf :

```
# Dovecot LDA
dovecot    unix    -    n    n    -    -    pipe
           flags=DRhu user=vmail:mail argv=/usr/lib/dovecot/deliver -d $recipient
```

Dovecot requiert également quelques modifications dans /etc/dovecot/dovecot.conf pour activer le LDA :

```
# MDA Configuration
protocol lda {
    postmaster_address = postmaster
    auth_socket_path = /var/run/dovecot/auth-master
}
```

Après un redémarrage de Dovecot et de Postfix, les mails seront donc transmis à **deliver** qui se chargera de les enregistrer dans la boîte de l'utilisateur.

```
# /etc/init.d/dovecot restart
# /etc/init.d/postfix restart
```

3.5 Premier test

Le serveur de mail est en principe complètement fonctionnel, et on peut tester l'ensemble de la chaîne en envoyant un mail en ligne de commande comme indiqué ci-dessous :

```
# echo test | mail -s "Premier test SMTP" user@example.org
```

Notons que si le Maildir de l'utilisateur n'existe pas, deliver va le créer automatiquement.

Chapitre 4

Elements de sécurisation

La sécurité du serveur de mails se situe à plusieurs niveaux : sécurisation des communications, authentification des utilisateurs et intégrité des données. Les communications sont généralement sécurisées en utilisant un chiffrement avec OpenSSL, l'authentification est assurée à l'aide de SASL et les données sont analysées par un antivirus et un anti-spam.

4.1 Protection contre l'Open-Relay et le SPAM

Pour augmenter la sécurité de notre serveur de mail, il est important de contrôler les échanges avec les serveurs SMTP ouverts (Open Relay). Ces serveurs sont utilisés pour envoyer des messages électroniques non sollicités, connus sous le nom de spam. Pour cela, il existe des listes d'adresses IP correspondant aux machines qui fonctionnent comme des serveurs SMTP ouverts. On peut donc demander à Postfix de consulter ces listes avant d'accepter un message entrant. Ceci est réalisé grâce à la directive **smtpd_client_restrictions** dans le fichier `/etc/postfix/main.cf`.

Enfin pour éviter que notre serveur devienne lui-même ouvert, il faut limiter les accès de façon très stricte. Par exemple, on peut rejeter la mail si les en-têtes sont incomplètes, si le message est mal formé, ... Par contre, mes messages provenant des réseaux connus (`my_networks`) ou des personnes authentifiées (`sasl_authenticated`) seront toujours acceptés.

```
# Wait until the RCPT TO command before evaluating restrictions
smtpd_delay_reject = yes

# Basics Restrictions
smtpd_helo_required = yes
strict_rfc821_envelopes = yes

# Requirements for the connecting server
smtpd_client_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_rbl_client bl.spamcop.net,
    reject_rbl_client dnsbl.njabl.org,
    reject_rbl_client cbl.abuseat.org,
    reject_rbl_client sbl-xbl.spamhaus.org,
    reject_rbl_client list.dsbl.org,
    permit

# Requirements for the HELO statement
smtpd_helo_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_non_fqdn_hostname,
    reject_invalid_hostname,
```

```

    permit

# Requirements for the sender address
smtpd_sender_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_non_fqdn_sender,
    reject_unknown_sender_domain,
    permit

# Requirement for the recipient address
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_non_fqdn_recipient,
    reject_unknown_recipient_domain,
    reject_unauth_destination,
    permit

```

Notons que ce type de règles permettent de diminuer sensiblement la charge de la machine dans le sens où le mail est rejeté avant d'être analysé par les outils de filtrage de contenu (Amavis, ...) qui sont souvent très consommateurs de ressources. On pourra également observer un léger gain de bande passante puisque seules les en-têtes des messages indésirables seront échangés sur le réseaux, en aucun cas les corps de message ou les pièces jointes.

La configuration ci-dessus autorise toujours le relay pour les personnes authentifiées grâce à la directive **permit_sasl_authenticated**. La configuration de l'authentification SASL, fait l'objet de la section suivante.

4.2 Authentification SMTP : SASL

SASL (Simple Authentication and Security Layer) est un protocole d'authentification décrit dans la RFC 2222. Il est conçu pour permettre une authentification en utilisant un des nombreux mécanismes disponibles (PAM, LDAP, shadow, ...). Si l'on utilise Dovecot comme serveur IMAP, il est maintenant possible d'utiliser son démon d'authentification pour réaliser l'authentification SMTP via SASL.

La première étape dans la configuration de SASL est l'édition de `/etc/postfix/main.cf`. A travers ce fichier on indique à Postfix qu'il faut utiliser l'authentification SMTP pour tous les utilisateurs qui désirent envoyer un mail, sauf s'ils font partie de `mynetworks`. Avec la configuration suivante, on interdit également les connexions anonymes :

```

# Enable SASL authentication for the smtpd daemon
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
# Fix some outlook's bugs
broken_sasl_auth_clients = yes
# Reject anonymous connections
smtpd_sasl_security_options = noanonymous
smtpd_sasl_local_domain =

```

Il faut maintenant modifier le fichier `/etc/postfix/master.cf` en modifiant l'option **smtpd_sasl_auth_enable** pour activer l'authentification avec SASL pour Postfix.

```

smtps      inet  n       -       -       -       -       smtpd
-o smtpd_tls_wrappermode=yes

```

```
-o smtpd_sasl_auth_enable=yes
```

Il reste à configurer Dovecot pour qu'il réponde aux requêtes d'authentification de Postfix. Ceci se fait toujours dans le fichier `/etc/dovecot/dovecot.conf`, il suffit de rajouter les lignes suivantes dans la section **socket listen** :

```
auth default {
  [...]
  socket listen {
    [...]
    # Client for postfix SASL
    client {
      path = /var/spool/postfix/private/auth
      mode = 0660
      user = postfix
      group = postfix
    }
  }
  [...]
}
```

Bien entendu, pour que les modifications soient prises en compte, il faut redémarrer les services avec la commandes suivantes :

```
# /etc/init.d/dovecot restart
# /etc/init.d/postfix restart
```

Comme pour le serveur IMAP, il est maintenant possible de tester l'authentification en faisant un connexion SMTP en telnet. Il faut préalablement préparer la chaîne d'authentification encodé en base64 avec la commande suivante :

```
# aptitude install metamail
# printf "\0user@example.org\0password" | mimencode
AHVzZXJAZXhhbXBsZS5vcmcAcGFzc3dvcmQ=
```

Cette chaîne sera utilisée avec la commande SMTP **AUTH PLAIN** pour envoyer le couple login/password au serveur. On peut maintenant commencer la session telnet comme indiqué ci-dessous :

```
# telnet 127.0.0.1 25
220 smtp.example.org ESMTP Postfix (Debian/GNU)
EHLO example.org
250-smtp.example.org
250-PIPELINING
250-SIZE 15728640
250-VERFY
250-ETRN
250-STARTTLS
250-AUTH PLAIN
250-AUTH=PLAIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
AUTH PLAIN AHVzZXJAZXhhbXBsZS5vcmcAcGFzc3dvcmQ=
235 2.0.0 Authentication successful
quit
221 2.0.0 Bye
```

La session ci-dessus présente une authentification réussie sur le serveur smtp.example.org avec le couple user@example.org / password pour l'authentification. On remarque dans cette session que le mot de passe est encodé en base64, il n'est donc pas lisible directement, il est par contre très facilement déchiffrable (avec mimencode -u). Cette méthode n'offre donc aucune sécurité, il est très facile de voir le mot de passe d'une personne en analysant les trames réseaux. La solution la plus répandue pour contourner ce problème est de chiffrer toutes les communications avec SSL, c'est l'objet du paragraphe suivant.

4.3 Certificats SSL

Les services de messagerie (SMTP et IMAP) demandent souvent une authentification de la part des utilisateurs. Dans ce cas, ce dernier va fournir un couple login/mot de passe qui devra transiter sur le réseau entre le client et le serveur. Pour que ces identifiants ne circulent pas en clair, la plupart des services offrent la possibilité de chiffrer les échanges à l'aide d'OpenSSL.

Pour cela, chaque service doit disposer d'un certificat SSL qui permettra de s'assurer de son authenticité et de chiffrer/déchiffrer les communications. Le fichier de configuration ci-dessous permettra de générer le certificat du serveur SMTPS. On peut par exemple l'enregistrer dans /etc/ssl/smtpd.cnf

```
[ req ]
default_bits           = 2048
default_keyfile        = privkey.pem
distinguished_name     = req_distinguished_name
prompt                = no
string_mask            = nombstr
x509_extensions       = server_cert

[ req_distinguished_name ]
countryName            = FR
stateOrProvinceName   = France
localityName          = Strasbourg
organizationName      = Example
organizationalUnitName = SMTP Server
commonName             = smtp.example.org
emailAddress           = postmaster@example.org

[ server_cert ]
basicConstraints       = critical, CA:FALSE
subjectKeyIdentifier   = hash
keyUsage               = digitalSignature, keyEncipherment
extendedKeyUsage       = serverAuth, clientAuth
nsCertType             = server
nsComment              = "SMTP Server"
```

La génération du certificat se fait ensuite avec la commande suivante :

```
# openssl req -x509 -new \
  -config /etc/ssl/smtpd.cnf \
  -out /etc/ssl/certs/smtpd.pem \
  -keyout /etc/ssl/private/smtpd.key \
  -days 730 -nodes -batch
# chmod 600 /etc/ssl/private/smtpd.key
```

Attention, lors de la configuration du certificat, le champ "commonName" est très important. Il doit toujours correspondre au FQDN qui sera utilisé pour accéder au service. Dans le cas contraire, la plupart des clients vont générer une alerte à chaque négociation TLS avec le serveur.

La génération du certificat du serveur IMAP est tout à fait similaire, il suffit de renommer et de modifier légèrement le fichier de configuration en remplaçant "smtp" par "imap".

4.4 IMAP : Chiffrement avec OpenSSL

Dovecot offre la possibilité de sécuriser les communications IMAP avec OpenSSL : IMAPS. Il est souvent raisonnable de n'utiliser que cette déclinaison du protocole afin qu'aucun mot de passe ne circule en clair sur le réseau.

Pour activer le support de l'IMAPS dans Dovecot, il faut dans un premier temps disposer d'un certificat pour ce service, ce qui a été fait dans le paragraphe précédent. Il suffit ensuite d'éditer le fichier `/etc/dovecot/dovecot.conf` pour modifier les paramètres suivants :

```
protocols = imaps
ssl_cert_file = /etc/ssl/certs/imapd.pem
ssl_key_file = /etc/ssl/private/imapd.key
```

Notons que le paramètre **protocols** ne contient que la valeur **imaps**. Avec cette configuration, les clients peuvent maintenant se connecter au serveur IMAP uniquement sur le port 993 et toutes les communications seront chiffrées.

Il reste simplement à redémarrer le serveur avant de faire un premier test en ligne de commande :

```
# /etc/init.d/dovecot restart
# openssl s_client -connect 127.0.0.1:993
---
* OK Dovecot ready.
. login user@example.org password
. OK Logged in.
. logout
* BYE Logging out
. OK Logout completed.
read:errno=0
```

4.5 SMTP : Chiffrement avec OpenSSL

Évidemment, Postfix permet également le chiffrement des communications avec OpenSSL, que ce soit pour le client ou le serveur SMTP. Maintenant que le certificat est prêt, il faut paramétrer les options de TLS/SSL pour Postfix dans le fichier de configuration principal `/etc/postfix/main.cf` comme indiqué ci-dessous. Pour information, le TLS (Transport Layer Security) est le nom donné à SSL (Secure Socket Layer) dans ces dernières versions. (Voir www.commentcamarche.net/crypto/ssl.php3 par exemple).

```
# TLS/SSL
smtpd_tls_security_level = may
smtpd_tls_loglevel = 1
smtpd_tls_cert_file = /etc/ssl/certs/smtpd.pem
smtpd_tls_key_file = /etc/ssl/private/smtpd.key
```

L'activation du support SSL pour le serveur SMTP se fait enfin en ajoutant la ligne suivante dans `/etc/postfix/master.cf`. Il est possible que cette ligne soit déjà présente mais commentée. Dans ce

cas il faut la dé-commentée et faire attention que les différents paramètres correspondent bien à l'exemple ci-dessous :

```
smtps      inet n       -       -       -       -       smtpd
-o smtpd_tls_wrappermode=yes
-o smtpd_sasl_auth_enable=yes
```

Bien sûr, un redémarrage du service est nécessaire pour que les modifications soient prises en compte :

```
# /etc/init.d/postfix restart
```

Les clients peuvent maintenant se connecter au serveur SMTP sur le port 465 pour envoyer leurs messages. Il est possible de faire un rapide test en telnet afin de valider le fonctionnement du serveur SMTPS, par exemple :

```
# openssl s_client -connect 127.0.0.1:465
[...]
---
220 smtp.example.org ESMTP Postfix (Debian/GNU)
helo example.org
250 smtp.example.org
mail from: <test@example.org>
250 2.1.0 Ok
rcpt to: <mat@example.org>
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Hello !!
.
250 2.0.0 Ok: queued as 8EF8730102C3
quit
221 2.0.0 Bye
read:errno=0
```

4.6 Configuration du firewall

La dernière étape de sécurisation du serveur de mail concerne le firewall. Dans la configuration retenue ici, il faut ouvrir uniquement les ports suivant en entrée :

- 25 (SMTP) : pour que le serveur puissent recevoir les mails provenant des autres serveurs de mails ;
- 465 (SMTPS) : pour que les utilisateurs authentifiés puissent envoyer des mails en mode crypté (SSL) ;
- 993 (IMAPS) : pour que les utilisateurs authentifiés puissent recevoir des mails en mode crypté (SSL) en utilisant IMAP.
- 2000 (MANAGESIEVE) : pour que les utilisateurs authentifiés puissent éditer les règles de filtrage.

Pour plus d'informations sur la configuration du firewall, voir ma page dédiée à ce sujet.

Chapitre 5

Analyse des messages

Le deuxième volet de la sécurité du serveur mail concerne le contenu des messages. En effet maintenant que les communications sont chiffrées et que les utilisateurs sont authentifiés avant toute opération, il reste à s'assurer que le contenu des messages n'est pas malveillant.

5.1 Filtrage de contenu : Amavis

Afin de centraliser toutes les opérations de filtrage de contenu, il faut mettre en place un démon SMTP qui sera capable de recevoir un mail, d'en extraire le contenu pour analyse puis de le renvoyer sous certaines conditions : c'est le rôle d'Amavis [9]. Le dépôt Debian Volatile étant maintenant un dépôt officiel de Debian, Amavis est disponible dans une version très récente et son cycle de mise à jour est beaucoup plus court.

```
# aptitude install amavisd-new libdbd-ldap-perl
```

Postfix prévoit une directive spécifique pour le filtrage de contenu : **content_filter**. Pour analyser le contenu avant de délivrer le message, il faut donc simplement indiquer à Postfix le transport à utiliser pour faire cette analyse dans `/etc/postfix/main.cf` :

```
# Use AMaVis
content_filter = amavis:[127.0.0.1]:10024
```

Ainsi, Postfix transmet le message au transport nommé **amavis** sur le port 10024 de la machine locale. Il faut maintenant configurer ce transport dans le fichier `/etc/postfix/master.cf`. Cette modification peut-être simplement réalisée à l'aide de la commande suivante :

```
# postfix-add-filter amavis 10025
```

Cette commande a permis de modifier le fichier `master.cf` afin de modifier le routage des messages. Le message sera maintenant analysé par Amavis puis, si le contenu n'est pas indésirable, il sera retransmis à Postfix sur le port 10025.

Il maintenant indiquer à Amavis quel sont les utilisateurs considérés comme "locaux", c'est à dire les utilisateurs pour lesquels il faudra ajouter les entêtes de SPAM par exemple. Pour cela, on interroge à nouveau notre base LDAP en éditant le fichier `/etc/amavis/conf.d/50-user` comme indiqué ci-dessous :

```
# Enable LDAP support
$enable_ldap = 1;

# Default LDAP settings
$default_ldap = {
  hostname => "127.0.0.1",
  tls => 0,
  version => 3,
  base => "ou=Users,dc=example,dc=org",
  scope => "sub",
  query_filter => "(&(objectClass=mailAccount)(|(mail=%m)(maildrop=%m))(mailenable=OK))"
};
```

Le routage permettant à Amavis de fonctionner est maintenant en place, il suffit de redémarrer les services pour que les modifications soient prises en compte :

```
# /etc/init.d/amavis restart
# /etc/init.d/postfix restart
```

5.2 Filtrage des messages entrant uniquement

La configuration décrite ci-dessus analyse les mails reçus ou envoyés par les utilisateurs stockés dans l'annuaire LDAP. Afin d'économiser des ressources non négligeables sur la machine, il peut être intéressant de ne filtrer que les mails provenant de sources non sûres. Les mails suivants ne seront donc pas analysés par l'antivirus :

- Mails provenant des machines présentes dans \$mynetworks
- Mails envoyés par des utilisateurs authentifiés.

Il est possible de configurer Amavis pour n'analyser qu'une partie des messages mais il est beaucoup plus intéressant de le faire au niveau de Postfix. L'idée est d'éviter de transmettre le message à Amavis si on ne désire pas l'analyser. On évite ainsi de remettre le message en queue et on accélère nettement le temps de traitement de chaque message.

Pour mettre en place cette configuration, il faut dans un premier temps s'assurer que la directive `content_filter` n'est plus définie dans le fichier `/etc/postfix/main.cf`. On désactive ainsi le routage systématique vers Amavis :

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

Il faut ensuite éditer les restrictions d'accès mises en place précédemment avec la directive **`smtpd_sender_restrictions`**. Avec la configuration ci-dessus, les mails provenant de \$mynetworks et les mails envoyés par des utilisateurs authentifiés sont acceptés directement. Si les messages ne sont pas rejetés par les directives `reject_non_fqdn_sender` et `reject_unknown_sender_domain`, ils sont ensuite envoyés à Amavis en utilisant la directive **`check_sender_access`** :

```
smtpd_sender_restrictions =
  permit_mynetworks,
  permit_sasl_authenticated,
  reject_non_fqdn_sender,
  reject_unknown_sender_domain,
  check_sender_access regexp:/etc/postfix/sender_access_catchall,
  permit
```

Reste enfin à définir la règle qui enverra tous les messages vers Amavis dans le fichier `/etc/postfix/sender_access_catchall` :

```
/^/ FILTER amavis:[127.0.0.1]:10024
```

Pour que ces modifications soient effectives, il faut bien entendu redémarrer postfix :

```
# /etc/init.d/postfix restart
```

5.3 Intégration d'un Antivirus : ClamAv

Il existe de nombreux antivirus mais beaucoup sont distribués sous des licences propriétaires. Dans le monde libre, il existe ClamAv qui fait très bien son boulot. Pour que Amavis puisse faire du filtrage antivirus, il faut donc installer ClamAv ainsi que quelques bibliothèques de compression (pour analyser les archives) :

```
# aptitude install clamav clamav-daemon gzip bzip2 unzip unrar zoo arj
```

Pour que tout ce passe correctement, il faut maintenant s'assurer que l'utilisateur utilisé par "clamav" fasse partie du groupe "amavis".

```
# adduser clamav amavis
```

- Activation du support antivirus pour Amavis dans `/etc/amavis/conf.d/15-content_filter_mode` :

```
@bypass_virus_checks_maps = (  
  \bypass_virus_checks, \bypass_virus_checks_acl, \bypass_virus_checks_re);
```

- Configuration de Amavis dans `/etc/amavis/conf.d/20-debian_defaults`

```
$final_virus_destiny = D_BOUNCE;
```

- Redémarrage des services

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

Pour vérifier le bon fonctionnement de l'antivirus, on peut utiliser des sites prévus à cet effet comme **GFI Email Security Test** [10]. Ces sites permettent d'envoyer des mails infectés avec un virus de test (non dangereux) vers votre serveur. En fonction des paramètres de notification qui ont été choisis, la réception de ce mail devrait, ou non, déclencher une alerte à l'administrateur du serveur. On peut par ailleurs consulter le contenu détaillé des headers des mails reçus pour s'assurer qu'ils contiennent une ligne du type :

```
X-Virus-Scanned: Debian amavisd-new at example.org
```

5.4 Intégration d'un AntiSpam : Spamassassin

- Installation de Spamassassin et des outils anti-spam

```
# aptitude install spamassassin libnet-dns-perl razor pyzor
```

- Activation de la cron Spamassassin qui met à jour les règles chaque nuit. Dans `/etc/default/spamassassin` :

```
CRON=1
```

- MAJ des règles Spamassassin

```
# sa-update
```

- Configuration de Spamassassin dans `/etc/spamassassin/local.cf`

```
report_safe 0
lock_method flock

# Bayes-related operations
use_bayes 1
use_bayes_rules 1
bayes_auto_learn 1
bayes_auto_expire 1
bayes_path /var/lib/amavis/.spamassassin/bayes
bayes_file_mode 0777

# External network tests
dns_available yes
skip_rbl_checks 0
use_razor2 1
use_pyzor 1

# Use URIBL (http://www.uribl.com/about.shtml)
urirhssub    URIBL_BLACK  multi.uribl.com.      A    2
body         URIBL_BLACK  eval:check_uridnsbl('URIBL_BLACK')
describe     URIBL_BLACK  Contains an URL listed in the URIBL blacklist
tflags       URIBL_BLACK  net
score        URIBL_BLACK  3.0

urirhssub    URIBL_GREY   multi.uribl.com.      A    4
body         URIBL_GREY   eval:check_uridnsbl('URIBL_GREY')
describe     URIBL_GREY   Contains an URL listed in the URIBL greylist
tflags       URIBL_GREY   net
score        URIBL_GREY   0.25

# Use SURBL (http://www.surbl.org/)
urirhssub    URIBL_JP_SURBL  multi.surbl.org.      A    64
body         URIBL_JP_SURBL  eval:check_uridnsbl('URIBL_JP_SURBL')
describe     URIBL_JP_SURBL  Has URI in JP at http://www.surbl.org/lists.html
tflags       URIBL_JP_SURBL  net
score        URIBL_JP_SURBL  3.0
```

- Configuration de razor en utilisant l'utilisateur **amavis**

```
# su - amavis
$ razor-admin -d --create
```

```
$ razor-admin -register
$ razor-admin -discover
```

- Configuration de pyzor en utilisant l'utilisateur **amavis**

```
# su - amavis
$ pyzor discover
```

- Activation du support anti-spam pour amavis dans `/etc/amavis/conf.d/15-content_filter_mode` :

```
@bypass_spam_checks_maps = (
  \bypass_spam_checks, \bypass_spam_checks_acl, \bypass_spam_checks_re);
```

Il faut maintenant définir le comportement de Amavis lorsque Spamassassin a analysé le message. Comme indiqué précédemment, Spamassassin affecte un score au message. Amavis va maintenant comparé ce score au seuil défini par la variable `FIXME`. Si le score est supérieur à cette valeur, Amavis va appliquer les règles destinées aux messages indésirables. Le score affecté par Spamassassin n'étant jamais fiable à cent pour cent, il faut prendre beaucoup de précautions lors de la définition des règles d'Amavis. Par exemple :

- il ne vaut mieux pas modifier le sujet du message en ajoutant `****SPAM****` : si le message est marqué par erreur, l'utilisateur final aura un message désirable dont le sujet commencera par `***SPAM***` ;
- il est fortement recommandé de NE PAS SUPPRIMER directement les messages marqués comme SPAM ;
- la méthode la plus sûre, est sans doute d'inscrire le score dans les en-têtes et de ne pas toucher au reste du message. Il suffit ensuite d'utiliser les différentes méthodes de filtrage pour que les SPAMs soient délivrés dans un dossier spécifique.

Toute cette configuration se fait dans le fichier `/etc/amavis/conf.d/20-debian_defaults` :

```
# $sa_spam_subject_tag      = '****SPAM**** ' ;
$sa_tag_level_deflt        = undef;
$sa_tag2_level_deflt       = 6.00;
$sa_kill_level_deflt       = 6.00;
$final_banned_destiny      = D_BOUNCE;
$final_spam_destiny        = D_PASS;
$final_bad_header_destiny  = D_PASS;
```

Notons qu'avec la configuration ci-dessus, les en-têtes de spam seront ajoutées sur tous les messages, quelque soit le score obtenu. Par ailleurs, tous les messages atteignant un score supérieur à 6.0 seront taggés comme Spam.

Comme toujours, les modifications ne seront prises en compte qu'après un redémarrage d'Amavis :

```
# /etc/init.d/amavis restart
```

Pour vérifier le bon fonctionnement de l'anti-SPAM, on peut par s'envoyer un mail (depuis l'extérieur) contenant la chaîne de test suivante (GTUBE) :

```
XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X
```

Cette chaine de test est reconnue par la plupart des anti-SPAM et doit forcément être reconnue par SpamAssassin. Si le mail contenant cette chaine n'est pas identifiée comme SPAM, il faut revoir la configuration de Amavis et de SpamAssassin : elle n'est surement pas correcte.

Chapitre 6

Gestion des quotas

6.1 Configuration du plugin

Dovecot offre la possibilité de gérer les quotas au niveau du MDA, donc lors de la réception des messages, mais également au niveau IMAP. Cela permet de s'assurer que les utilisateurs ne vont pas dépasser le quota en faisant un glisser-déposer entre deux comptes par exemple.

Le support des quotas se fait simplement en modifiant le fichier `/etc/dovecot/dovecot.conf` comme indiqué ci-dessous :

```
protocol imap {
    mail_plugins = quota imap_quota
}
protocol lda {
    [...]
    mail_plugins = [...] quota
    [...]
}
```

Afin que Dovecot puisse lire les quotas dans LDAP, il faut ensuite modifier le fichier `/etc/dovecot/dovecot-ldap.conf` comme indiqué ci-dessous :

```
user_attrs = mailbox=home,mailuserquota=quota=maildir:ignore=Trash:storage
```

La configuration décrite ci-dessous utilisera le champ LDAP `mailuserquota` de chaque utilisateur pour définir la taille maximum de sa boîte aux lettres.

Notons que l'option **ignore=Trash** permet de ne pas appliquer de quotas sur la poubelle. (Afin de permettre aux utilisateurs de supprimer un mail même s'ils ont atteint la limite autorisée). Cette solution montre cependant vite ses limites puisqu'elle ne permet pas de limiter la taille de la corbeille. Une alternative plus élégante et plus flexible est d'ores et déjà développée et sera intégrée dans Dovecot v1.1.

Pour activer la gestion des quotas, il ne reste plus qu'à redémarrer le service :

```
# /etc/init.d/dovecot restart
```

6.2 Définition des quotas

Pour définir les quotas, il faut maintenant éditer le champ mailuserquota dans l'annuaire LDAP. Avec le MDS, il est possible de définir ce quota de manière globale ou par utilisateur.

- Définition des quotas par défaut au niveau du domaine
- Définition des quotas par utilisateur

6.3 Quelques tests

Il est maintenant possible de faire un premier test en telnet pour vérifier que les quotas définis dans l'annuaire LDAP ont bien été pris en compte :

```
# telnet 127.0.0.1 143
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
* OK Dovecot IMAP Server ready.
. login user@example.org password
. OK Logged in.
. getquotaroot inbox
* QUOTAROOT "inbox" ""
* QUOTA "" (STORAGE 244982 500000)
. OK Getquotaroot completed.
. getquota ""
* QUOTA "" (STORAGE 244979 500000)
. OK Getquota completed.
. logout
* BYE Logging out
. OK Logout completed.
```

Coté client, Thunderbird permet de visualiser l'état des quota lors de l'affichage des propriétés d'un dossier. Avec Thunderbird > 2.0, il est possible de visualiser les quotas dans barre de progression en bas de la fenêtre. Pour cela, il suffit de modifier la variable suivante dans l'éditeur de configuration :

```
mail.quota.mainwindow_threshold.show = 0
```

Coté webmail, RoundCube [11] affiche également l'état des quotas dans une barre de progression en bas de la fenêtre.

Chapitre 7

Tri des messages coté serveur : SIEVE

Après la gestion des quotas, un autre atout intéressant de Dovecot est la gestion des filtres SIEVE. SIEVE est un langage permettant de traiter les messages à leur arrivée sur le serveur : il est par exemple possible de rediriger un message vers un dossier particulier ou de mettre en place un répondeur automatique de vacance.

SIEVE agit donc de manière comparable à Procmail mais apporte une fonctionnalité très intéressante : MANAGESIEVE. MANAGESIEVE est un protocole permettant de gérer les scripts de filtrage depuis l'application de messagerie cliente : plus besoin de se connecter en SSH pour éditer son script de filtrage !

Bien entendu, le support SIEVE n'est disponible uniquement si la livraison des messages est réalisée par le LDA de Dovecot.

7.1 Configuration du plugin

- Tri des mails avec sieve, détection du Spam ...

```
protocol lda {
  [...]
  # sieve_global_path =
  mail_plugins = [...] cmusieve
  [...]
}
```

Pour activer la gestion des filtres SIEVE, il ne reste plus qu'à redémarrer le service :

```
# /etc/init.d/dovecot restart
```

Édition d'un script de base dans /home/<user_name>/.dovecot.sieve :

```
require "fileinto";
if address :is "from" "matthieu@example.org" {
  fileinto "Matthieu";
}
```

Exemple de script pour mettre en place un répondeur automatique de vacance :

```

require ["fileinto", "vacation"];
if header :contains "X-Spam-Flag" "YES" {
    fileinto "Junk";
} else {
    vacation
    # Repondre uniquement 1 fois par jour au meme expéditeur
    :days 1
    # Sujet de la reponse automatique
    :subject "Message automatique de vacance"
    # Liste des adresses de destination pour lesquelles il faut envoyer le
    message
    # de vacance automatique
    :addresses ["matthieu@example.org", "mat@example.org"]
    "Inserez ici le message de vacance.

    Cordialement,
    Matthieu";
}

```

7.2 Filtres globaux

Dovecot supporte les filtres globaux pour permettre à tous les utilisateurs de partager un script de filtrage. Attention cependant, le script global défini par la variable **sieve_global_path** ne sera exécuté uniquement si il n'y a aucun filtre personnel pour l'utilisateur.

Dans `/etc/dovecot/dovecot.conf` :

```

protocol lda {
    [...]
    sieve_global_path = /etc/dovecot/global_script/dovecot.sieve
    mail_plugins = [...] cmusieve
    [...]
}

```

Création du répertoire :

```
# mkdir /etc/dovecot/global_script
```

Exemple de script pour trier les mails marqués par Spamassassin :

```

require "fileinto";
if header :contains "X-Spam-Flag" "YES" {
    fileinto "Junk";
}

```

Modification des droits :

```
chown -R vmail:mail /etc/dovecot/global_script/
chmod -R 770 /etc/dovecot/global_script/
```

Pour appliquer la configuration, il ne reste plus qu'à redémarrer le service :

```
# /etc/init.d/dovecot restart
```

7.3 Édition des scripts : MANAGESIEVE

Comme indiqué précédemment, Dovecot gère maintenant le protocole MANAGESIEVE. Ce support n'est pas encore officiel mais est disponible sous forme de patch. Côté Debian, le patch est inclus dans les paquets depuis la version 1.0.12.

Pour activer le démon managesieve, il suffit simplement de l'ajouter dans la directive "protocols" du fichier `/etc/dovecot/dovecot.conf` :

```
protocols = imaps managesieve

[...]

protocol managesieve {
    sieve=~/.dovecot.sieve
    sieve_storage=~/.sieve
}
```

Malheureusement, les clients capables de gérer correctement le protocole MANAGESIEVE sont très rares :

- Il existe un plugin pour roundcube mais qui ne fonctionne pas sur la version stable actuelle.
- Le plugin Thunderbird fonctionne maintenant correctement depuis sa version 0.1.5. Ce plugin permet de gérer le protocole managesieve mais malheureusement l'édition des scripts doit encore se faire manuellement.
- ...

Chapitre 8

Apprentissage des SPAMS

8.1 Introduction

- apprentissage des SPAMS dans les dossiers Junk de tous les utilisateurs
- apprentissage des HAMS (Non-SPAMS) dans tous les autres dossiers de tous les utilisateurs
- ménage dans les dossiers Junk des utilisateurs

8.2 Cron d'apprentissage

Dans /usr/local/sbin/spam_learn.sh

```
#!/bin/bash

MAILDIRS="/home/vmail/"
DB_PATH="/var/lib/amavis/.spamassassin/"
DB_USER="amavis"
DB_GROUP="amavis"
SA_LEARN="'which sa-learn '"
LEARN_SPAM_CMD="$SA_LEARN --spam --no-sync --dbpath $DB_PATH"
LEARN_HAM_CMD="$SA_LEARN --ham --no-sync --dbpath $DB_PATH"

# Learn SPAM
find $MAILDIRS -iregex '.*\/.Junk\/(.*)?\/\.(cur|new)\.*/.*' -type f -exec $LEARN_SPAM_CMD {} \;
chown -R $DB_USER:$DB_GROUP $DB_PATH

# Learn HAM
find $MAILDIRS -iregex '.*\/cur\/.*' -not -regex '.*\/.Junk\/(.*)?\/\.(cur|new)\.*/.*' -type f -exec $LEARN_HAM_CMD {} \;
chown -R $DB_USER:$DB_GROUP $DB_PATH

# Clean SPAM folders
find $MAILDIRS -iregex '.*\/.Junk\/(.*)?\/\.(cur|new)\.*/.*' -type f -ctime +30 -delete
```

Ajout des droits d'exécution :

```
# chmod u+x /usr/local/sbin/spam_learn.sh
```

Dans /etc/cron.d/amavisd-new

```
#  
# SpamAssassin maintenance for amavisd-new  
#  
# m h dom mon dow user  command  
00 20 * * 6 root /usr/local/sbin/spam_learn.sh > /var/log/spam_learn.log 2> /var/  
log/spam_learn.err  
18 */3 * * * amavis test -e /usr/sbin/amavisd-new-cronjob && /usr/sbin/amavisd-new-  
cronjob sa-sync
```

8.3 Ajustements manuels

Si la configuration de spamassassin doit être ajustée manuellement, pour ajouter des adresses blacklistées par exemple, il est toujours possible d'ajouter des règles personnalisées dans le fichier de configuration. Par exemple, pour blacklister l'adresse spam@spammer.org, il suffit d'ajouter la ligne suivante dans le fichier /etc/spamassassin/local.cf :

```
blacklist_from spam@spammer.org
```

Notons qu'en réalité, tous les mails provenant de cette adresse auront un score augmenté de 100 points.

Chapitre 9

Support TLS pour les clients SMTP

9.1 Introduction

- Roadwarriors
- Relay selectif

9.2 Configuration du client

Coté client :

```
# TLS client parameters
smtp_tls_cert_file=/etc/ssl/certs/smtpd.pem
smtp_tls_key_file=/etc/ssl/private/smtpd.key
smtp_tls_security_level = may
```

- Génération de l'empreinte du certificat.

```
# openssl x509 -md5 -fingerprint -in /etc/ssl/certs/smtpd.pem | grep -i fingerprint
```

9.3 Configuration du serveur

Coté serveur :

```
# TLS based relaying
relay_clientcerts = hash:/etc/postfix/relay_clientcerts
smtpd_tls_ask_ccert = yes
smtpd_use_tls = yes
```

Dans chaque restriction, ajouter :

```
[...]
permit_tls_clientcerts,
[...]
```

- Définition des clients autorisés dans `/etc/postfix/relay_clientcerts` :

```
XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX roadwarrior.example.org
```

- Génération du hash et redémarrage du service :

```
# postmap /etc/postfix/relay_clientcerts  
# /etc/init.d/postfix restart
```

Chapitre 10

Envoit de mails depuis le réseau local

10.1 Présentation

La messagerie est souvent utilisée pour remonter des alertes systèmes ou des alertes de sécurité. Lorsque les serveurs sont dans un réseau local et ne sont pas associés à un DNS publique, il devient difficile d'envoyer des mails vers l'extérieur : le sender du message (par exemple `root@serveur.domainelocal`) est rejeté par les serveurs de mails distant qui considèrent l'émetteur comme invalide.

Pour contourner ce problème il est possible de modifier l'adresse de l'émetteur à la volée en utilisant une adresse contenant un domaine valide sur le net. Notons que la méthode présentée ci-dessous ne modifie l'adresse de l'émetteur que dans l'enveloppe du message : l'en-tête "From" restera donc inchangée. Ainsi le mail sera correctement routé, mais le destinataire pourra toujours identifier l'émetteur original.

10.2 Mise en place

Postfix prévoit une directive permettant la réécriture de l'adresse de l'émetteur à la volée : **sender_canonical_maps**. Il suffit d'indiquer le type de table utilisé et le fichier contenant les informations pour la réécriture :

```
sender_canonical_maps = regexp:/etc/postfix/sender_rewrite
```

Il faut ensuite définir les réécritures à l'aide de regexp dans le fichier `/etc/postfix/sender_rewrite` :

```
/(.*)@(.*).\domainelocal/ ${1}-${2}.domainelocal@example.org
```

Avec la configuration décrite ci-dessous, tous les mails arrivant sur cette machine en provenance de `root@serveur.mon-lan` seront expédiés en utilisant l'adresse `root.serveur.mon-lan@example.org` dans l'enveloppe du message.

Chapitre 11

Webmail : RoundCube

Pour permettre l'accès aux boîtes IMAP depuis n'importe où, l'installation d'un webmail s'impose. De nombreux webmail Open Source sont disponibles mais celui qui sort du lot en ce moment (à mon goût) c'est RoundCube [11]. Il est encore très jeune mais offre de gros atouts côté ergonomie, notamment grâce à l'utilisation d'AJAX. L'interface est très soignée, claire et simple.

Techniquement, ça reste très classique, c'est du PHP et ça s'installe très facilement. Une dernière remarque sur le carnet d'adresse qui peut maintenant être couplé à un annuaire LDAP.

11.1 Installation

Pour que RoundCube fonctionne correctement, il faut bien entendu que la machine héberge un serveur Apache et un serveur MySQL fonctionnels. Pour plus d'informations sur la configuration du serveur Web, vous pouvez consulter ma page dédiée à ce sujet. De façon plus rapide, voilà la liste des paquets nécessaires pour le fonctionnement de RoundCube :

```
# aptitude install apache2 php5 php5-ldap
```

RoundCube est maintenant dans les backports il suffit donc de l'installer avec aptitude :

```
# aptitude install roundcube
```

Notons que roundcube peut être utilisé avec plusieurs types de bases de données : dans Debian, il utilise SQLite par défaut. Le backend SQLite fonctionne très bien, il est cependant préférable d'utiliser MySQL ou PostgreSQL pour les grosses installations.

Côté base de données en elle-même, il n'y a rien à faire, Debconf se charge de tout.

11.2 Configuration de RoundCube

La configuration de l'accès à la base de données se fait dans `/etc/roundcube/dd.inc.php`. En principe, il ne faut rien modifier, Debconf a généré les fichiers correctement.

Tout le reste de la configuration se fait dans le fichier `/etc/roundcube/main.inc.php`. La configuration décrite ci-dessous permet d'utiliser l'annuaire LDAP dans le carnet d'adresse et de définir quelques options par défaut (comme la langue, l'affichage du panneau de pré visualisation des messages, ...)

```

$rcmail_config['default_host'] = 'ssl://127.0.0.1:993';
$rcmail_config['smtp_server'] = '127.0.0.1';
$rcmail_config['smtp_user'] = '%u';
$rcmail_config['smtp_pass'] = '%p';
$rcmail_config['create_default_folders'] = TRUE;

$rcmail_config['ldap_public']['example.org'] = array(
    'hosts' => array('127.0.0.1'),
    'port' => 389,
    'base_dn' => 'ou=Users,dc=example,dc=org',
    'search_fields' => array('cn', 'sn', 'givenname', 'mail'),
    'name_field' => 'cn',
    'firstname_field' => 'givenName',
    'surname_field' => 'sn',
    'email_field' => 'mail',
    'scope' => 'sub',
    'filter' => 'mailenable=OK',
    'fuzzy_search' => true);

$rcmail_config['locale_string'] = 'fr';
$rcmail_config['preview_pane'] = TRUE;
$rcmail_config['check_all_folders'] = TRUE;

```

11.3 Configuration d'apache

Il est possible d'utiliser la configuration d'apache fournie dans les paquets Debian (/etc/roundcube/apache.conf) ou de dédier un VirtualHost au webmail. Ce paragraphe présente la création d'un virtualhost dédié.

Notons que la configuration décrite ci-dessous re-dirige tous les visiteurs vers le vhost en HTTPS afin de ne laisser aucune information importante circuler en clair sur le réseau. Par exemple, créez un fichier /etc/apache2/sites-available/webmail contenant les lignes suivantes :

```

<VirtualHost *:80>

    ServerName webmail.example.org
    ServerAlias webmail
    ServerAdmin webmaster@example.org

    RewriteEngine on
    RewriteCond %{SERVER_PORT} !^443$
    RewriteRule ^/(.*)$ https://%{SERVER_NAME}%{REQUEST_URI} [R=301,L]

</VirtualHost>

<VirtualHost *:443>

    ServerName webmail.example.org
    ServerAlias webmail
    ServerAdmin webmaster@example.org

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/apache.pem

    DocumentRoot /var/lib/roundcube

    <Directory /var/lib/roundcube/>
        Options +FollowSymLinks
        AllowOverride All
        order allow,deny
        allow from all
    </Directory>

```

```

</Directory>

# Protecting basic directories:
<Directory /var/lib/roundcube/config>
    Options -FollowSymLinks
    AllowOverride None
</Directory>

<Directory /var/lib/roundcube/temp>
    Options -FollowSymLinks
    AllowOverride None
    Order allow,deny
    Deny from all
</Directory>

<Directory /var/lib/roundcube/logs>
    Options -FollowSymLinks
    AllowOverride None
    Order allow,deny
    Deny from all
</Directory>

CustomLog /var/log/apache2/webmail_access.log combined
ErrorLog /var/log/apache2/webmail_error.log

</VirtualHost>

```

Pour terminer avec apache, il faut simplement modifier une option de PHP dans le fichier `/etc/php5/apache2/php.ini` :

```
magic_quotes_gpc = Off
```

Reste à activer le vhost et à redémarrer apache pour que les modifications soient prise en compte :

```

# a2enmod rewrite
# a2enmod ssl
# a2ensite webmail
# /etc/init.d/apache2 restart

```

Le webmail est maintenant accessible avec un navigateur Web à l'adresse `http://webmail.example.org`.

Chapitre 12

Références

- [1] Site officiel de postfix. www.postfix.org.
- [2] Site officiel de dovecot. www.dovecot.org.
- [3] Site officiel de openldap. www.openldap.org.
- [4] Site du mandriva directory server. mds.mandriva.org.
- [5] Site officiel du projet debian. www.debian.org.
- [6] Installation et configuration de debian lenny. www.vogelweith.com/debian_server/00_installation.php.
- [7] Site officiel du projet debian volatile. volatile.debian.org.
- [8] Document ldap - mds. www.vogelweith.com/debian_server/050_openldap.php.
- [9] Site officiel de amavis. www.amavis.org.
- [10] Test de sécurité. gfi.com/emailsecuritytest/.
- [11] Site officiel de roundcube. www.roundcube.net.
- [12] Documentation de postfix en français. x.guimard.free.fr/postfix/.