21 juin 2008

Configuration avancée de Postfix

Catégorie : Administration système Tags : GLMF

~~Retrouvez cet article dans : Linux Magazine 85~~

Postfix est l'un des « Mail Transfert Agent » les plus utilisés sur Internet. Destiné à remplacer Sendmail dont les problèmes de sécurité étaient nombreux, Postfix a tout de suite été conçu autour de trois objectifs : un système de configuration simple, une forte compatibilité avec les commandes de Sendmail et enfin une conception hautement sécurisée. Ainsi, dans beaucoup de petits réseaux, seules quelques directives suffisent à paramétrer convenablement le serveur de messagerie et en toute sécurité. Au travers de la résolution de quelques problèmes de configuration avancée, ce cycle d'articles vous propose d'aller plus loin avec Postfix.

Ce premier article se concentre sur la sécurité interne d'un réseau de messagerie. Le réseau imaginaire qui va nous permettre d'étudier Postfix est composé d'un réseau d'utilisateurs sur lequel certains utilisateurs peuvent bénéficier de droits d'administration de leurs machines et de réseaux de serveurs. (Voir Fig. 1)

1. Restriction de la fonction relais

Dans sa configuration par défaut, Postfix gère simplement les autorisations de relais : tous les clients des réseaux répertoriés dans la directive ~~mynetworks~~ sont autorisés à envoyer du courrier à l'extérieur tandis que les autres ne sont autorisés à envoyer du courrier qu'à destination des domaines gérés par le serveur. Cette configuration se traduit dans Postfix par la valeur par défaut suivante du fichier de configuration ~~main.cf~~ :

```
smtpd_recipient_restrictions =
          permit_mynetworks,
          reject_unauth_destination
```

Comme les restrictions sont lues de gauche à droite jusqu'à la première règle qui correspond, un client interne aura le droit d'envoyer du courrier à tous les destinataires (~~permit_mynetworks~~), tandis qu'un client externe se verra refuser les destinations non traitées par le serveur (~~reject_unauth_destination~~). Si le client externe envoie un courrier vers une destination interne, aucune règle ne correspond et la règle par défaut s'applique (~~permit~~).
On peut toutefois rendre ce mécanisme plus fin. Dans le réseau que nous construisons ici, on distingue 4 types de clients : les extérieurs, les serveurs autorisés à envoyer du courrier sans restrictions, les serveurs autorisés à envoyer du courrier en interne uniquement et enfin les utilisateurs autorisés à envoyer du courrier partout à condition d'être authentifié comme nous le verrons plus loin. Le mécanisme des tables de Postfix nous permet aisément de répondre à cette problématique. Les éléments placés avant la directive ~~reject_unauth_destination~~ identifient les clients autorisés à communiquer avec l'extérieur, tandis que ceux placés après ne peuvent communiquer qu'en interne. La configuration devient alors :

```
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination,
    check_client_access hash:fichier,
    reject
```

Quelques explications :

- ~~permit_mynetworks~~ : la variable mynetworks ne référence alors que les réseaux des serveurs privilégiés. En particulier, on y inclut tous les relais de messagerie ;
- ~~permit_sasl_authenticated~~ : les utilisateurs sont autorisés à envoyer du courrier depuis n'importe quel point du réseau pourvu qu'ils soient authentifiés (voir ci-après) ;
- ~~check_client_access hash:fichier~~ : le fichier en paramètre référence les serveurs autorisés à communiquer en interne. Exemple d'utilisation, les serveurs relais de messagerie situés en DMZ doivent être inscrits ici, car ils ne sont pas censés amener à l'intérieur du réseau des messages destinés à l'extérieur ;
- ~~reject~~ : les clients ne répondant pas aux conditions précédentes ne sont pas autorisés à envoyer de courrier même en interne. Si notre serveur est directement accessible depuis Internet, il ne faut bien entendu pas mettre cette réserve, car le serveur doit rester accessible pour la réception du courrier.

Le fichier passé en paramètre au quatrième point ne contient pas des réseaux dans leur notation traditionnelle, mais est examiné par Postfix avec les 4 octets de l'adresse IP, puis trois, deux et un. On ne peut donc y déclarer que des classes d'adresses. Exemple de contenu du fichier [1] :

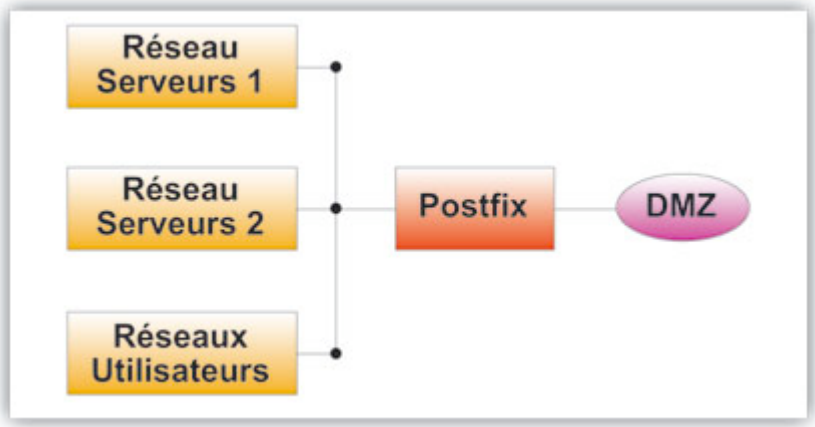

Fig. 1

```
192.168.4    OK
172.168      OK
10.12.13.14 OK
```

Autre avantage de cette configuration : seuls les réseaux utilisateurs sont soumis à l'authentification. Les serveurs sont simplement répartis dans ~~mynetworks~~ pour ceux qui génèrent du courrier à destination de l'extérieur et dans le fichier ci-dessus pour les serveurs à vocation interne. On peut bien entendu imposer l'authentification pour tous en ne conservant dans ~~smtpd_recipient_restrictions~~ que :

```
smtpd_recipient_restrictions =
    permit_sasl_authenticated,
    reject
```

Note :

Les fichiers passés en paramètre dans les restrictions concernent par défaut le champ inspecté par la restriction : par exemple, ~~smtpd_recipient_restrictions~~ consulte les tables en utilisant l'adresse mail du destinataire. Pour inspecter un autre champ, il faut précéder les tables par les mots clefs ~~check_client_access~~, ~~check_sender_access~~ ou ~~check_recipient_access~~. Dans l'exemple ci-dessus, s'agissant d'une restriction sur les destinataires, on utilise ~~check_client_access~~ pour examiner les adresses IP des clients [2].

Note :

Le test des restrictions basées sur l'adresse IP peut se faire à partir de n'importe quelle machine en inscrivant son adresse IP dans ~~smtpd_authorized_xclient_hosts~~. Dès lors, elle est autorisée à utiliser la commande SMTP ~~XCLIENT~~ pour simuler une connexion depuis une autre IP [4] : ~~XCLIENT ADDR=10.1.1.2~~

2. Contrôle de la validité des expéditeurs internes

Tout comme le courrier postal qui est son modèle, le protocole SMTP ne prévoit pas de contrôle du nom de l'expéditeur. L'adresse d'expéditeur exigée dans l'enveloppe n'est utilisée que pour les

notifications en cas de problème de remise et n'a aucun lien avec l'adresse communiquée au logiciel de messagerie du client. Il en est de même pour les adresses de destination. L'exemple ci-dessous montre une transaction valide au sens du protocole SMTP :

```
220 mondomaine.fr ESMTP Postfix
HELO bidon.org
250 mondomaine.fr
MAIL FROM: inconnu@bidon.org
250 Ok
RCPT TO: quelquun@mondomaine.fr
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: "Toto" <toto@nexistepas.org>
To: "Titi" <titi@nexistepas.fr>
Subject: Test

Ceci est un test
.
250 Ok: queued as 8DDC4471
QUIT
```

Ce message arrivera bien au destinataire (quelquun@mondomaine.fr), mais ce dernier ne verra que les adresses mentionnées dans les champs From et To, car la consultation du courrier ne se fait plus par SMTP, mais par le protocole de relevé de la boîte aux lettres, soit généralement POP ou IMAP. Le serveur SMTP en revanche n'utilise que les champs RCPT TO et MAIL FROM – l'enveloppe – pour assurer la livraison ou notifier les erreurs. La solution souvent proposée est d'obliger les utilisateurs à s'authentifier, mais là encore, rien n'oblige a priori un utilisateur disposant d'un compte valide à utiliser son adresse dans le champ MAIL FROM. Postfix nous offre toutefois des mécanismes pour fiabiliser les adresses sources en les mettant en relation avec le nom utilisé pour l'authentification.

2.1 Mécanisme d'authentification

Comme d'autres logiciels tels Cyrus-Imap, Postfix n'effectue pas lui-même l'authentification, mais utilise les mécanismes SASL de la librairie Cyrus-SASL. Sur Debian, la version par défaut de Postfix n'est pas compilée avec le support SASL. Il suffit de remplacer Postfix et d'installer le démon SASL :

```
apt-get install postfix-tls sasl2-bin
```

Sur cette distribution, le démon saslauthd ne démarre pas par défaut : il faut ajouter « start = yes » dans le fichier de configuration /etc/default/saslauthd. Nous ne détaillerons pas ici la configuration du démon SASL. Pour Postfix, le choix des mécanismes d'authentification à utiliser se fait dans le fichier /etc/postfix/sasl/smtpd.conf (fichier à créer) :

L'authentification SMTP est l'une des méthodes de contrôle du relaying avec des clients sur des IP attribuées dynamiquement. Une autre solution peut prendre la forme de tunnels SSL ou d'un VPN.

```
pwcheck_method: saslauthd
mech_list: plain login
```

L'usage du mécanisme « plain » n'est pas sécurisant hors d'une connexion TLS/SSL. Nous verrons plus bas comment activer le chiffrement en cas d'authentification. Le paramétrage du

serveur saslauthd s'effectue lui dans le fichier /etc/saslauthd.conf. Un exemple utilisant un annuaire LDAP [3] :

```
ldap_servers: ldap://ldap.domaine.fr/
ldap_search_base: dc=domaine,dc=fr
ldap_bind_dn: cn=postfix,dc=domaine,dc=fr
ldap_password: mot-de-passe
ldap_filter: (uid=%u)
```

L'activation du support SASL pour la réception des messages s'effectue alors très simplement dans le fichier main.cf :

```
smtpd_sasl_auth_enable = yes
```

Dès lors, l'authentification est proposée au client lors de l'appel ESMTP :

```
220 server.host.tld ESMTP Postfix
EHLO client.host.tld
250-server.host.tld
250-PIPELINING
250-ETRN
250-AUTH DIGEST-MD5 PLAIN CRAM-MD5
250 8BITMIME
```

Pour tester la bonne configuration du serveur, on peut se connecter avec netcat ou à défaut telnet sur le port 25 et composer la séquence suivante :

```
$ nc 127.0.0.1 25
220 server.host.tld ESMTP Postfix
EHLO client.host.tld
250-server.host.tld
...
250-AUTH DIGEST-MD5 PLAIN CRAM-MD5
AUTH PLAIN <code en base 64>
235 Authentification successful
```

SMTP est l'un des protocole « en clair » les plus faciles à « écouter ». Jetez un œil du côté des outils Dsniff pour vous en convaincre.

Le code en base 64 contient la chaîne « user\0user\0password ». Pour constituer ce code, utilisez la commande suivante en remplaçant les valeurs user et pass par des valeurs adéquates :

```
perl -MMIME::Base64 -e 'print
encode_base64(«user\0user\0pass»);'
```

Pour imposer à certains clients de s'authentifier, il suffit d'insérer permit_sasl_authenticated dans une restriction comme nous l'avons vu dans le paragraphe précédent.

2.2 Correspondances entre utilisateur et adresse source

Une fois l'utilisateur authentifié, la correspondance entre le nom d'utilisateur et l'adresse source ne s'établit pas directement, mais en passant par des tables de correspondances entre adresse mail et utilisateurs autorisés à l'employer. Exemple :

```
smtpd_sender_login_maps =
```

hash:/etc/postfix/fichier

où fichier contient toutes les adresses et les utilisateurs autorisés à les utiliser :

```
root@exemple.fr user1,user2
```

Dans le cas où l'adresse peut être calculée simplement à partir du nom, on peut utiliser des expressions rationnelles :

```
smtpd_sender_login_maps =
    pcre:/etc/postfix/fichier
```

où fichier contient :

```
/^([^@]+)@exemple\.fr$/ $1
```

Dans cet exemple, les utilisateurs s'authentifient avec la partie gauche de leur adresse mail. On peut également combiner ces deux solutions en indiquant plusieurs tables de correspondances dans le champ ~~smtpd_sender_login_maps~~ ou encore utiliser un annuaire LDAP ou une base SQL. La mise en correspondance des adresses mail et noms de login n'impose rien à Postfix. Pour rendre effective l'interdiction d'envoyer du courrier si la correspondance n'est pas établie, il faut créer une restriction sur les adresses d'expédition. Par exemple :

```
smtpd_sender_restrictions =
    reject_authenticated_sender_login_mismatch
```

Dans ce cas, seuls les utilisateurs authentifiés sont filtrés. Pour les autres, la règle par défaut s'applique (~~permit~~).

3. Confidentialité

Mis à part le cas où un administrateur peut garantir que sur aucun de ses réseaux un utilisateur n'est en mesure d'espionner les communications des autres, la protection des communications passe par le chiffrement. Nous n'aborderons pas ici la possibilité d'employer le chiffrement par S/MIME qui ne permet de toute façon pas de protéger la phase d'authentification. Avant la version 2.3 de Postfix, le support du chiffrement n'est pas compilé par défaut. Vous devez donc recompiler Postfix avec le support TLS. Avec une distribution Debian, si vous ne l'avez pas déjà installé avec le support SASL, ce changement s'opère simplement :

```
apt-get install postfix-tls
```

La mise en place d'un certificat dans Postfix est également très simple :

```
smtpd_tls_cert_file =
     /etc/postfix/server.pem
smtpd_tls_key_file =
     $smtpd_tls_cert_file
smtpd_use_tls = yes
```

Dans notre exemple, on utilise le même fichier pour le certificat et la clef privée, mais ces fichiers peuvent être distincts. La dernière ligne active le chiffrement. A partir de là, les clients qui le souhaitent peuvent chiffrer leurs communications. Il nous reste alors à imposer l'emploi du chiffrement aux clients. Deux solutions se présentent à nous : forcer le chiffrement pour tous les

clients ou ne chiffrer que certaines communications. Dans le premier cas, il suffit simplement d'ajouter :

```
smtpd_enforce_tls = yes
```

Tous les clients devront alors utiliser le chiffrement, ce qui nous interdit d'utiliser notre serveur directement sur Internet, car le courrier y circule en clair (RFC 2487). Cette configuration est rarement utilisée. Dans notre exemple, seuls les réseaux utilisateurs nécessitent une protection de leurs communications :

```
smtpd_enforce_tls = no
smtpd_tls_auth_only = yes
```

Cette directive n'impose le chiffrement que lorsque la communication comporte une authentification. Comme nous avons imposé l'authentification aux utilisateurs, on obtient le résultat voulu : toutes les communications sont établies en clair sauf sur les réseaux des utilisateurs [5].

4. Contrôle différencié par boîte aux lettres

Dans ce paragraphe, nous étudions la possibilité de restreindre l'autorisation d'envoi de courrier vers certaines boîtes aux lettres à certains expéditeurs seulement. Au travers de cet exemple, nous verrons comment introduire un système de décision basé sur des tests.
La logique du système que nous voulons mettre en place est résumée ci-dessous :

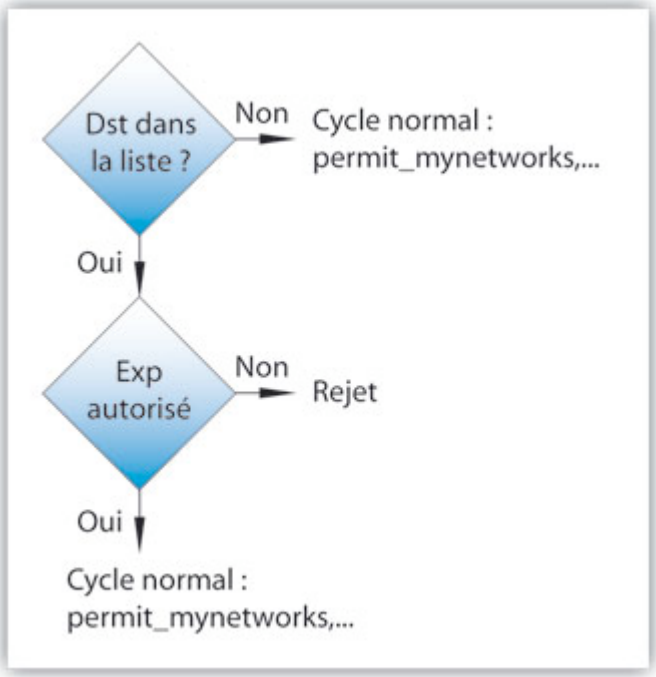

Fig. 2

Le « cycle normal » est appelé afin de ne pas créer de trous de sécurité en acceptant aveuglément un message, pourvu que l'expéditeur et le destinataire se trouvent dans nos listes ! Dans Postfix, les restrictions sont examinées de gauche à droite jusqu'à trouver la première qui correspond. L'action trouvée, généralement ~~OK~~, ~~DROP~~ ou ~~REJECT~~, est ensuite exécutée. Or, Postfix nous

permet également d'insérer dans ces tables d'autres instructions. C'est précisément ce que nous allons faire : tout d'abord, décrivons la première ligne de notre système logique :

```
smtpd_recipient_restrictions =
    hash:/mon/fichier.dst,
    permit_mynetworks,
    ...
```

Si le destinataire ne se trouve pas dans le fichier, on a bien application du cycle normal des restrictions. Dans le cas contraire, Postfix va exécuter l'action indiquée :

```
#/mon/fichier.dst
user1@domaine.fr  check_sender_access \
    hash:/fichier/user1.exp,reject
```

Si l'expéditeur ne se trouve pas dans le fichier ~~/fichier/user1.exp~~ correspondant aux expéditeurs autorisés à envoyer du courrier à ~~user1@domaine.fr~~, le message est rejeté. On obtient bien le fonctionnement correspondant à la deuxième ligne du schéma de principe.

Il ne nous reste plus qu'à implémenter le « cycle normal » :

```
#/fichier1/user1.exp
dst@autorisé    permit_mynetworks,...
```

Conclusion

La grande facilité avec laquelle on peut paramétrer Postfix dans le cas général peut parfois laisser penser qu'il n'est pas adapté à des situations plus complexes. Ces petites astuces issues de mon expérience personnelle vous auront, je l'espère, montré le contraire. Dans les prochains articles, nous verrons comment gérer le routage des messages dans un réseau complexe et comment insérer des traitements particuliers dans les flots de messages.

Références

- [1] Le format des fichiers de restriction est détaillé dans la page de manuel ~~access(5)~~ accessible également sur : http://www.postfix.org/access.5.html
- [2] L'usage d'une restriction sur les destinataires dans une restriction sur les expéditeurs ne fonctionne que si le paramètre ~~smtpd_delay_reject~~ est maintenu à sa valeur par défaut (~~yes~~). En effet, dans ce cas Postfix attend les deux champs ~~MAIL FROM~~ et ~~RCPT TO~~ avant d'examiner les restrictions. Dans le cas contraire, la restriction sur les expéditeurs est évaluée avant que les destinataires ne soient connus.
- [3] ~~http://www.linuxorable.net/7-Cyrus-IMAP-58-LDAP.html~~ : une excellente documentation sur la configuration de saslauthd pour utiliser ldap. L'article concerne Cyrus-Imap, mais peut parfaitement s'appliquer à Postfix.
- [4] L'emploi de la commande ~~XCLIENT~~ est détaillée dans la documentation française en ligne (http://postfix.traduc.org/).
- [5] Les clients Microsoft ne sont pas toujours capables de déclencher le chiffrement de la connexion en lançant la commande ~~SMTP STARTTLS~~. Pour les accueillir, il faut ajouter un serveur smtpd en écoute sur le port 465 (~~SMTP~~ sur ~~SSL~~) avec l'option

~~-o smtpd_tls_wrappermode = yes~~. Ceci se fait très simplement en dupliquant la ligne lançant le serveur smtpd dans le fichier ~~master.cf~~ :

```
465 inet  n  -  n  -  -  smtpd
    -o smtpd_tls_wrappermode=yes
```

- Sur Debian, la ligne est prête, il reste à la décommenter. Si après tout ça, vous utilisez toujours une autre distribution, vous êtes irrécupérable ;-).
- ☐Documentation de Postfix : http://www.postfix.org/documentation.html
- ☐Traduction de la documentation par l'auteur : http://postfix.traduc.org/ ou http://x.guimard.free.fr/postfix/

~~Retrouvez cet article dans : Linux Magazine 85~~

Posté par (La rédaction) | Signature : Xavier Guimard | Article paru dans 

Laissez une réponse

Vous devez avoir ouvert une session pour écrire un commentaire.

« Précédent Aller au contenu »

Identifiez-vous
Inscription
S'abonner à UNIX Garden

Articles de 1ère page

- Yafray, le moteur de rendu photoréaliste libre, une première approche
- Ajouter des logiciels : la gestion des paquets
- Web : récupérez vos marque-pages
- Une nouvelle disposition de clavier français pour Xorg
- Installation et configuration d'E17
- Migration des données
- Ubuntu, un peu d'histoire...
- H.P.Anvin : M. ISOLinux / SYSLinux
- Les marques déposées et les Logiciels libres
- Créez un forum de discussion avec phpBB2

- **Il y a actuellement**
- 638 articles/billets en ligne.

Recherche

- **Catégories**

- Administration réseau
- Administration système
- Agenda-Interview
- Audio-vidéo
- Bureautique
- Comprendre
- Distribution
- Embarqué
- Environnement de bureau
- Graphisme
- Jeux
- Matériel
- News
- Programmation
- Réfléchir
- Sécurité
- Utilitaires
- Web

- **Archives**

- juillet 2008
- juin 2008

- mai 2008
- avril 2008
- mars 2008
- février 2008
- janvier 2008
- décembre 2007
- novembre 2007
- février 2007

GNU/Linux Magazine

- GNU/Linux Magazine 107 - Juillet/Août 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine 107
- GNU/Linux Magazine HS 37 - Juillet/Août 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine HS 37
- GNU/Linux Magazine 106 - Juin 2008 - Chez votre marchand de journaux !

GNU/Linux Pratique

- Linux Pratique N°48 -Juillet/Août 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique N°48
- Linux Pratique Essentiel N°2 - Juin/Juillet 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique Essentiel N°2
- Linux Pratique Hors-Série N°15 - Juin / Juillet 2008 - chez votre marchand de journaux.

MISC Magazine

- Références de l'article « Détection de malware par analyse système » d'Arnaud Pilon paru dans MISC 38
- Références de l'article « La sécurité des communications vocales (3) : techniques numériques » d'Éric Filiol paru dans MISC 38
- Misc 38 : Codes Malicieux, quoi de neuf ? - Juillet/Août 2008 - Chez votre marchand de journaux
- Edito : Misc 38
- Misc 37 : Déni de service - Mai/Juin 2008 - Chez votre marchand de journaux