

Sécurité du service de courrier électronique : amavisd-new

Philippe Latu

philippe.latu(at)inetdoc.net

<http://www.inetdoc.net>

Depuis plusieurs années, on assiste à une globalisation des menaces véhiculées par le service de courrier électronique. Il est donc essentiel de traiter la problématique de sécurisation de ce service avec une approche globale permettant d'adapter les relations entre les différentes fonctions de sécurité. Amavisd-new est un logiciel libre dont le rôle est de mettre en relation différents outils de sécurité. Ce document s'appuie sur ce logiciel pour illustrer l'organisation des de fonctions classiques de sécurité du service de courrier électronique.

Table des matières

1. Copyright et Licence	1
1.1. Meta-information	1
1.2. Conventions typographiques	2
2. Le contexte sécurité du service de courrier électronique	2
2.1. L'ingénierie sociale ou les limites de la technologie	2
2.2. Le placement des fonctions de sécurisation du courrier électronique	3
2.3. L'architecture système	5
2.4. L'architecture réseau	5
2.5. L'architecture «domestique»	6
3. Le service amavisd-new	8
3.1. L'installation	8
3.1.1. Le téléchargement des sources et les dépendances	8
3.2. La gestion de l'arborescence de quarantaine	8
3.3. La libération d'un message en quarantaine	9
3.4. La reconnaissance passive d'empreinte de système d'exploitation	10
4. La lutte contre les pourriels	11
4.1. La configuration de spamassassin	11
4.1.1. Le fichier de configuration principal	11
4.2. La mise à jour des jeux de règles de spamassassin	12
4.3. Les calculs distribués avec Pyzor et Razor	13
4.3.1. Pyzor	13
4.3.2. Razor	14
4.3.3. Tests de fonctionnement	14
5. Clam AntiVirus	15
5.1. Paquets Debian à installer	15
5.2. Interaction avec amavisd-new	16
5.3. Configuration du service amavisd-new	17
6. Les échantillons relevés	17
6.1. Module MIME::Tools	17
7. Documents de référence	17

1. Copyright et Licence

Copyright (c) 2000,2012 Philippe Latu.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Copyright (c) 2000,2012 Philippe Latu.

Permission est accordée de copier, distribuer et/ou modifier ce document selon les termes de la Licence de Documentation Libre GNU (GNU Free Documentation License), version 1.3 ou toute version ultérieure publiée par la Free Software Foundation ; sans Sections Invariables ; sans Texte de Première de Couverture, et sans Texte de Quatrième de Couverture. Une copie de la présente Licence est incluse dans la section intitulée « Licence de Documentation Libre GNU ».

1.1. Meta-information

Cet article est écrit avec *DocBook*¹ XML sur un système *Debian GNU/Linux*². Il est disponible en version imprimable aux format PDF : [amavisd-new.pdf](#)³.

¹ <http://www.docbook.org>

1.2. Conventions typographiques

Tous les exemples d'exécution des commandes sont précédés d'une invite utilisateur ou *prompt* spécifique au niveau des droits utilisateurs nécessaires sur le système.

- Toute commande précédée de l'invite \$ ne nécessite aucun privilège particulier et peut être utilisée au niveau utilisateur simple.
- Toute commande précédée de l'invite # nécessite les privilèges du super-utilisateur.

2. Le contexte sécurité du service de courrier électronique

«La sécurité est un processus et non un produit. Et comme dans tout processus, certaines de ces composantes sont plus solides, plus fiables, mieux huilées et plus sûres que d'autres. De plus, ces composantes doivent s'emboîter les unes dans les autres. Mieux elles s'emboîtent, mieux le processus fonctionne. Souvent, ce sont les interfaces entre les composantes qui sont les éléments les moins sûrs.»

—Secrets et mensonges - Sécurité numérique dans un monde en réseau - Bruce Schneier

2.1. L'ingénierie sociale ou les limites de la technologie

L'humain est-il la composante la plus faible du processus de sécurisation du courrier électronique ? Répondre oui à cette question c'est choisir un alibi facile. L'ingénierie sociale (ou «esbroufe» dans un français plus académique), montre simplement les limites de la technologie. Il n'existe pas (encore ?) de système technique capable de détecter toutes les formes d'usurpation d'identité.

- Ouvrir la pièce jointe chiffrée (contenant le virus !) dont la clé de déchiffrement est fournie dans le corps du message est un piège facile à éviter tant que l'on est pas «sensible» au contenu du message en question. Seule l'interprétation humaine provoquera le déclenchement du code viral.
- Compléter un formulaire demandant toutes les coordonnées bancaires (n° de carte de crédit, etc.) transmis par courrier est un piège facile à éviter tant que l'on «détecte une différence» entre le message et le formulaire Web que l'on a l'habitude de saisir pour consulter l'état de son compte en banque. Là encore, tout n'est qu'une question d'interprétation qui échappe totalement aux technologies de sécurité.

Face aux évolutions constantes des méthodes d'usurpation d'identité, seule une information constante et complète des utilisateurs sur le «bon usage» du service de courrier électronique est efficace. Cette sensibilisation sur les usages, aussi nécessaire soit-elle, est parfois difficile. Essayez de convaincre la totalité des utilisateurs d'un service de ne plus composer leurs courriers en HTML !. Gros challenge en perspective ;).

Même si la technologie n'est pas une arme absolue, ses capacités de traitements automatisés sur des volumes de messages importants rendent de grands services.

L'objet de ce document est justement de présenter un service *interface* entre les fonctions classiques de filtrage de contenus de courrier électronique.

² <http://www.debian.org>

³ <http://www.inetdoc.net/pdf/amavisd-new.pdf>

2.2. Le placement des fonctions de sécurisation du courrier électronique

L'acheminement du courrier passe par des (étapes|points) bien particuliers. Tout commence par le service de noms de domaines (DNS) qui désigne les adresses IP responsables du traitement du courrier électronique d'une zone donnée. Les hôtes responsables de ces traitements (*Mail Transfer Agent*) sont repérés par le champ *Mail eXchanger* (MX) dans le fichier de zone DNS.

Voici un exemple simple permettant d'obtenir la liste des hôtes responsables du courrier électronique pour la zone nic.fr. :

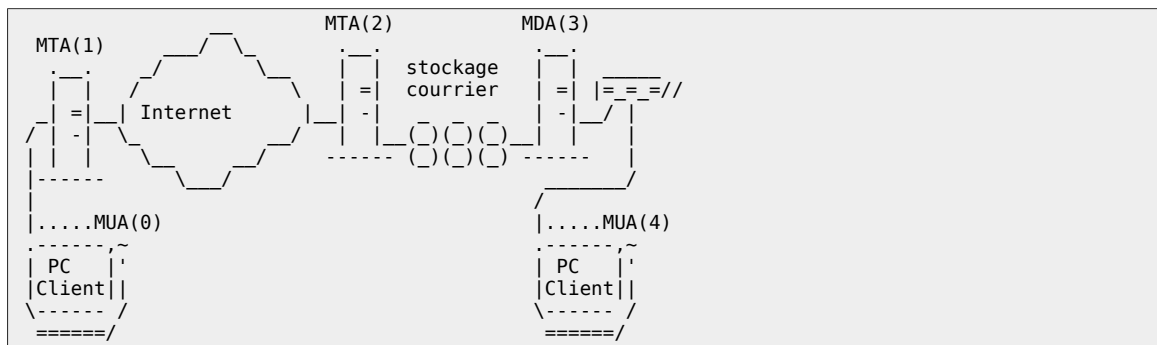
```
$ dig nic.fr MX
; <<> DiG 9.7.3 <<> nic.fr MX
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 39798
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;nic.fr.                IN      MX

;; ANSWER SECTION:
nic.fr.                156070 IN     MX     30 mx3.nic.fr.
nic.fr.                156070 IN     MX     10 mx1.nic.fr.
nic.fr.                156070 IN     MX     20 mx2.nic.fr.

;; Query time: 56 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Sep 29 16:39:07 2011
;; MSG SIZE rcvd: 84
```

Passons maintenant au courrier électronique proprement dit et son acheminement entre zones de l'Internet.



MUA(0), Mail User Agent, Émission du courrier

L'utilisateur émet un courrier à l'aide d'une application appelée *Mail User Agent* vers le **MTA(1)** de son fournisseur d'accès Internet. La seule précaution possible à ce niveau consiste à utiliser un antivirus «à jour». Concrètement, cette condition est rarement respectée sur les postes domestiques. Pire encore, la vitesse de propagation des vers est maintenant supérieure à la fréquence des mises à jour de signatures de virus. Le niveau de sécurité d'un poste émettant du courrier est donc *nécessairement* faible.



Avertissement

Il est possible, via une configuration particulière, d'émettre directement du courrier vers n'importe quel MTA sur l'Internet à partir du poste client.

Ce type de fonctionnement correspond presque systématiquement à une infection virale. C'est pour cette raison que dès que ces émissions sont détectées, l'adresse IP du poste est classée en liste noire. Il est donc vivement déconseillé d'émettre directement du courrier vers d'autres MTA que celui de son fournisseur d'accès.

MTA(1), Mail Transfer Agent (-TX), Transfert du courrier en émission

Le rôle du *Mail Transfer Agent* est de transférer le courrier sur l'Internet vers le *Mail Transfer Agent* correspondant à l'adresse du destinataire.

La sécurisation des communications entre les MTAs est à la charge des opérateurs qui acheminent les données. Le rôle des fournisseurs d'accès dans la propagation des vers n'est pas neutre. Les clients sont facturés deux fois pour un service qui leur est dû. La surconsommation de bande passante due aux «pourriels» et aux vers est facturée à travers les forfaits et les fonctions antivirus et antispam sont facturées à travers des prestations supplémentaires. Si le problème était traité à la source, la surconsommation de bande passante serait sensiblement diminuée et la propagation des vers limitée.

Outre la logique commerciale de la démarche, l'argumentaire sur l'approche globale de la sécurisation s'applique parfaitement ici. Tant qu'un opérateur «respecte» la segmentation du marché qui lui impose d'affecter un groupe d'unités centrales à une fonction unique (antivirus, antispam, listes noires, etc.) sans (*organiser|contrôler*) les relations entre ces fonctions, il n'a d'autre choix que d'investir sans fin dans de

nouveaux châssis dédiés. Cette course est asphyxiante aussi bien sur le plan financier que sur le plan des ressources humaines disponibles pour administrer le service.

Dans ce contexte, il faut *réagir* à son niveau et prendre un maximum de précautions lors de l'émission de courrier électronique depuis sa propre infrastructure.

Les fonctions de sécurité sont identiques à celles mises en œuvre au niveau d'un **MTA(2)** de réception du courrier électronique. Seules les relations entre ces fonctions diffèrent. C'est au service de réception d'assumer la protection des périmètres placés sous son contrôle.

MTA(2), *Mail Transfer Agent* (-RX), Transfert du courrier en réception

Relativement au **MTA(1)**, le *Mail Transfer Agent* de réception joue le rôle le plus important au niveau sécurité. C'est à ce point de passage que la charge de sécurisation est la plus critique. Il n'est pas rare de devoir «jeter» plus de 70% des courriers présentés au *Mail Transfer Agent*. Les événements des années 2003 et 2004 ont montré combien il est important que les relations entre les fonctions de sécurité soient bien contrôlées. Cette maîtrise de la chaîne de sécurisation permet «d'encaisser» les chocs lors des propagations massives de nouveaux vers.

MDA(3), *Mail Delivery Agent*, Délivrance du courrier

Le rôle du *Mail Delivery Agent* est de prélever le courrier dans les files d'attente et de le déposer dans le répertoire de boîte aux lettres de l'utilisateur. Procmail est l'outil MDA le plus utilisé dans l'univers GNU/Linux. Il est possible de placer des fonctions de sécurité à ce niveau : appels antivirus (et/ou) antispam. Ce type d'appel est très pénalisant en charge CPU et surtout en accès au système de fichiers sur la machine qui exécute le programme. Il est donc déconseillé de traiter de gros volumes de courrier de cette façon.

Malgré ce défaut très pénalisant, le *Mail Delivery Agent* est l'outil de personnalisation des fonctions de sécurité. Si l'utilisateur souhaite régler lui-même les paramètres de fonctionnement des outils de sécurité, c'est là que l'opération doit se faire. Ces réglages individuels ne sont pas incompatibles avec les fonctions mises en œuvre au niveau *Mail Transfer Agent*.

MUA(4), *Mail User Agent*, Réception du courrier

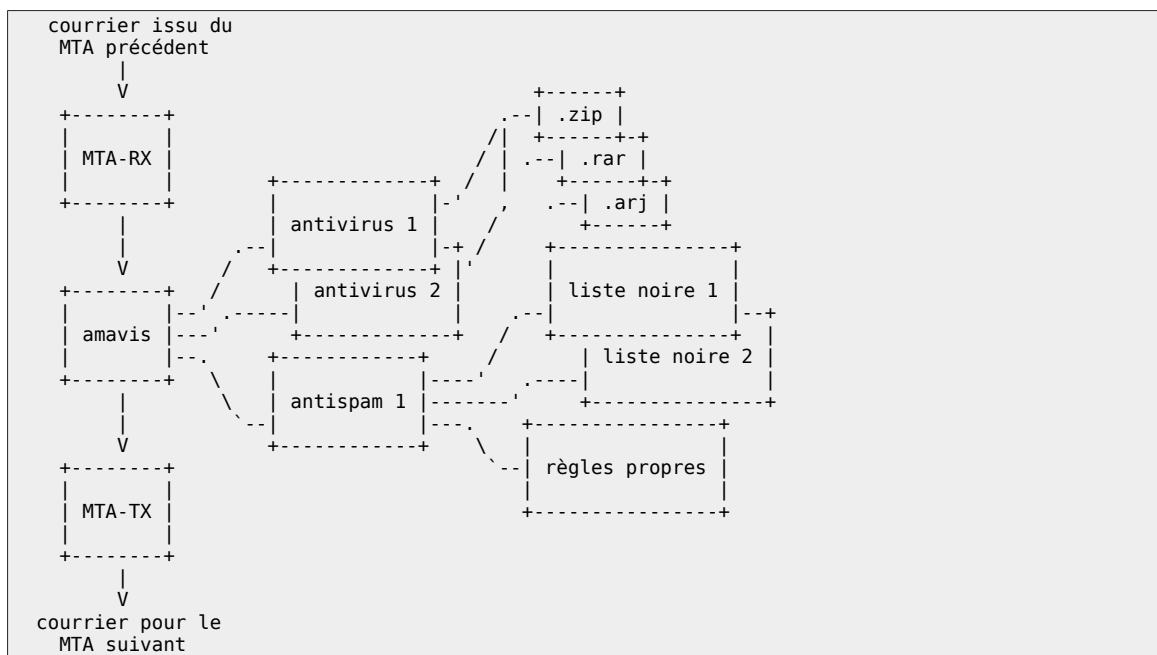
Pour faire simple, si un courrier infecté arrive à ce niveau : c'est foutu !. La population des administrateurs de parc informatique ayant constaté que les antivirus et antispam clients ont une efficacité plus que limitée ne cesse de croître. Si tous les ténors des solutions propriétaires cherchent encore à faire illusion, on (entend|lit) de plus en plus que la vitesse de propagation des vers et des pourriels est beaucoup trop rapide pour que ces solutions propriétaires soutiennent le rythme.

Dans la suite de ce document, nous allons nous concentrer sur les fonctions de sécurité appliquées aux *Mail Transfer Agents* assurant l'émission (-TX) et la réception (-RX) du courrier à la frontière du périmètre administré. Comme indiqué ci-avant (**MDA(3)**), il est toujours possible d'affiner les paramètres d'utilisation des outils de sécurité. C'est le moyen de personnaliser au niveau utilisateur la chaîne de sécurité. Il faut rappeler que cette individualisation a un coût d'exécution important. Par exemple, les appels à spamassassin à travers procmail sont si longs à exécuter qu'ils gênent la consultation des courriers électroniques.

2.3. L'architecture système

En considérant les points de passage obligatoires de l'acheminement du courrier électronique on retient que les *Mail Transfer Agents* situés à la frontière du réseau sont les outils critiques sur lesquels l'approche globale de sécurité du service de courrier électronique doit s'appliquer.

Pour appliquer cette approche globale on utilise un service dédié aux relations entre les fonctions de sécurité : **amavisd-new**.



Dans l'exemple ci-dessus, on a représenté 3 fonctions de sécurité à titre indicatif : 2 antivirus et 1 antisipam. Il existe de très nombreuses combinaisons possibles. Chaque fonction de sécurité peut elle même faire appel à plusieurs autres services : bases de données de signatures de virus, bases de données de listes noires, algorithmes de calcul de notes, algorithmes de (dé)compression, etc. Une représentation exhaustive de toutes les relations possibles serait illisible.

Les en-têtes de courrier servent à (véhiculer|échanger) des paramètres entre les services exécutés par les différents *Mail Transfer Agents*. Ce sont ces échanges de paramètres (X-Virus-*, X-Spam-*, etc.) qui régissent les relations entre MTA. Le service **amavisd-new** doit donc nécessairement prendre en compte ces champs d'en-têtes pour prétendre à l'approche globale de sécurité.

Voici un «bel exemple» d'utilisation des champs d'en-tête indiquant le résultat d'un calcul de pondération antisipam. Ici, le message a obtenu un score final de 46.551 alors que le seuil de classement en quarantaine est de 6.31. Aucun doute n'est permis sur la nature de ce courrier électronique. Il s'agit d'un pourriel.

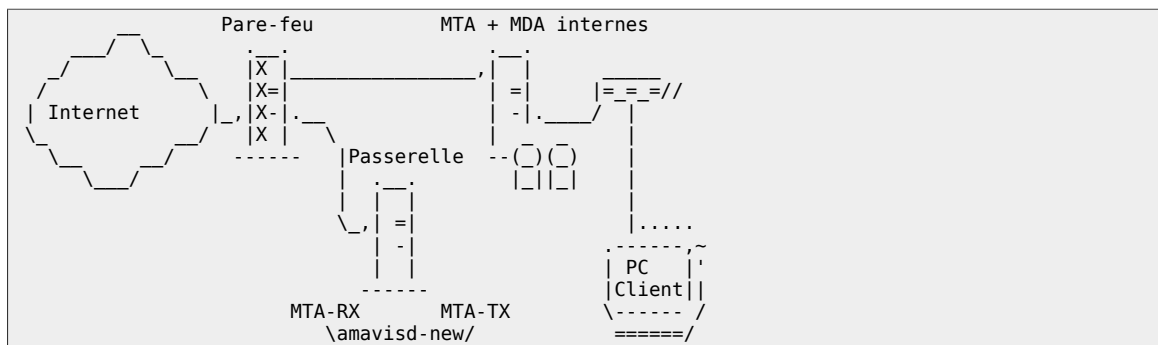
```
X-Spam-Level: *****
X-Spam-Status: Yes, score=46.551 tag=-999 tag2=6.31 kill=6.31
tests=[ADVANCE_FEE_2_NEW_FORM=0.001, ADVANCE_FEE_2_NEW_FRM_MNY=0.001,
ADVANCE_FEE_2_NEW_MONEY=0.661, ADVANCE_FEE_3_NEW=0.835,
ADVANCE_FEE_3_NEW_FORM=0.902, ADVANCE_FEE_3_NEW_FRM_MNY=2.969,
ADVANCE_FEE_3_NEW_MONEY=0.001, BAYES_99=5.8, DATE_IN_PAST_96_XX=3.405,
DEAR_BENEFICIARY=2.999, DKIM_SIGNED=0.1, FILL_THIS_FORM=0.001,
FILL_THIS_FORM_LOAN=2.88, FILL_THIS_FORM_LONG=3.404,
FORGED_MUA_OUTLOOK=1.927, FORM_FRAUD_3=0.001,
FREEMAIL_FORGED_REPLYTO=2.095, FROM_MISSPACED=1.064,
FROM_MISSP_DKIM=0.001, FROM_MISSP_MSFT=2.312,
FROM_MISSP_REPLYTO=0.001, FROM_MISSP_TO_UNDISC=1.262,
FROM_MISSP_URI=0.001, FROM_MISSP_USER=2.401, FSL_CTYPE_WIN1251=0.001,
FSL_UA=0.782, FSL_XM_419=0.02, LOTS_OF_MONEY=0.001,
MONEY_FRAUD_3=0.179, MONEY_FROM_MISSP=0.786,
MSOE_MID_WRONG_CASE=2.584, NSL_RCVD_FROM_USER=0.001,
RCVD_IN_BRBL_LASTEXT=1.449, RCVD_IN_PSBL=2.7, RCVD_IN_RP_RNBL=1.31,
RCVD_IN_SBL=0.141, RCVD_IN_SORBS_WEB=0.77, RDNS_NONE=0.793,
T_DKIM_INVALID=0.01] autoLearn=spam
```

2.4. L'architecture réseau

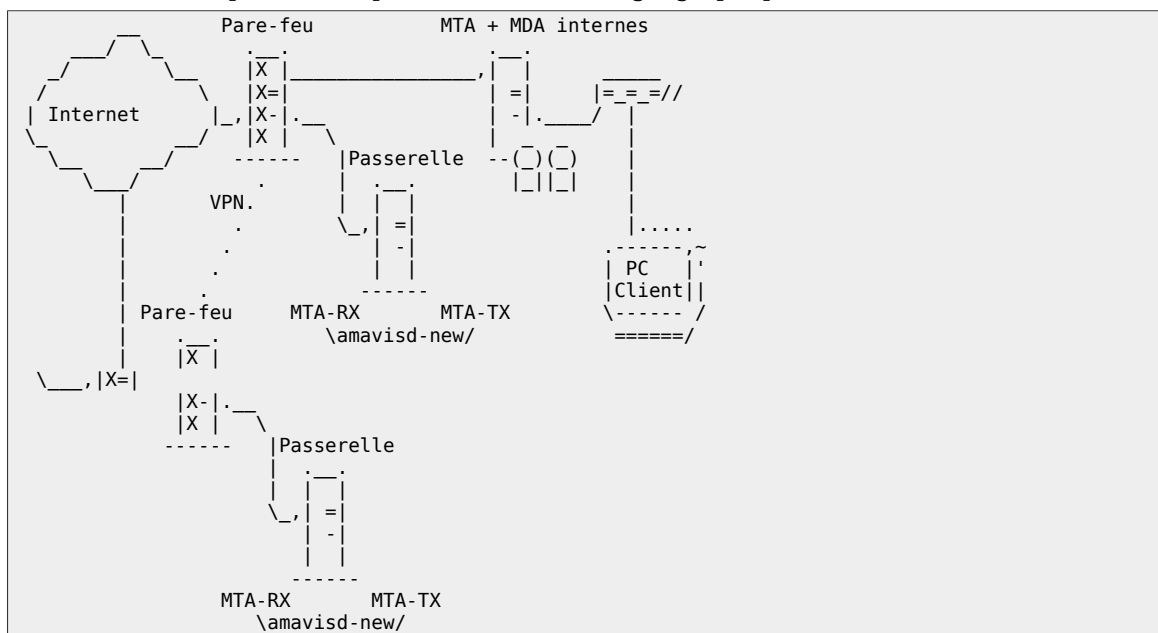
Voici une architecture réseau «simple» avec laquelle l'acheminement du courrier suit les processus suivants :

1. Le *Mail Transfer Agent* de réception de la *passerelle* est désigné comme service prioritaire dans la configuration du service de noms de domaines (champ MX du fichier de configuration de zone).
2. Les règles du *Pare-feu* sont rédigées de façon à ce que les courriers électroniques issus de l'Internet arrivent à la passerelle et nulle part ailleurs.

3. Le service **amavisd-new** appelle tous les traitements voulus et prend une décision sur chaque courrier : passage au MTA suivant, mise en quarantaine ou destruction.
4. Dans le cas où le courrier doit passer au MTA suivant, les règles du *Pare-feu* sont rédigées de façon à ce que les courriers électroniques arrivent au MTA interne depuis la *passerelle* pour être délivrés dans les boîtes aux lettres des utilisateurs.



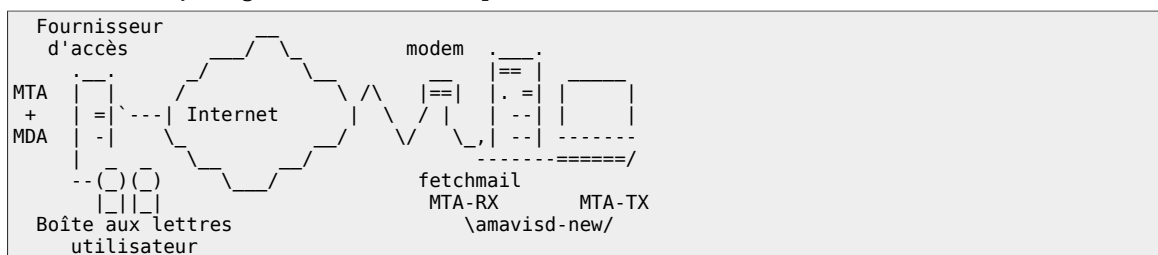
Voici une architecture réseau «plus réaliste» qui intègre une redondance de la sécurisation du service ; on parle aujourd'hui de haute disponibilité. L'acheminement du courrier doit toujours disposer de 2 voies de communication distinctes de façon à tolérer une panne quelconque sur une ou plusieurs fonctions de sécurité. On utilise alors 2 passerelles placées sur des sites géographiques différents.



Le fonctionnement des passerelles est identique à celui décrit ci-dessus. Ce sont les champs MX du service de noms de domaines qui régissent les priorités entre les deux passerelles. Un réseau privé virtuel (VPN) assure les communications entre les deux pare-feux.

2.5. L'architecture «domestique»

Comment appliquer la stratégie de sécurité présentée lorsque l'on ne gère pas un domaine réseau en propre ? Il est tout à fait possible de «réintroduire» le courrier électronique déposé chez un fournisseur d'accès vers un *Mail Transfer Agent*. Voici un exemple d'architecture :

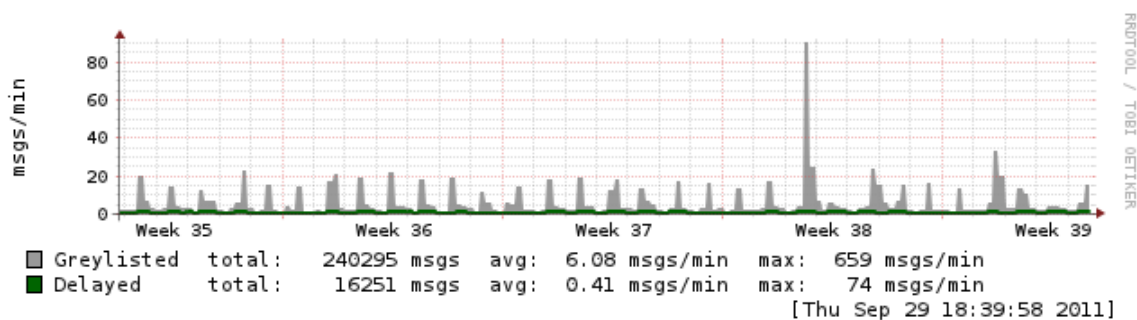
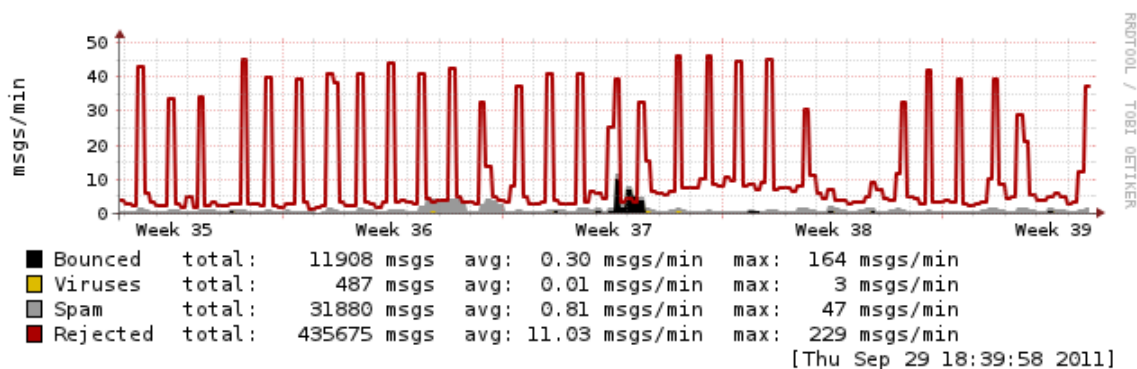
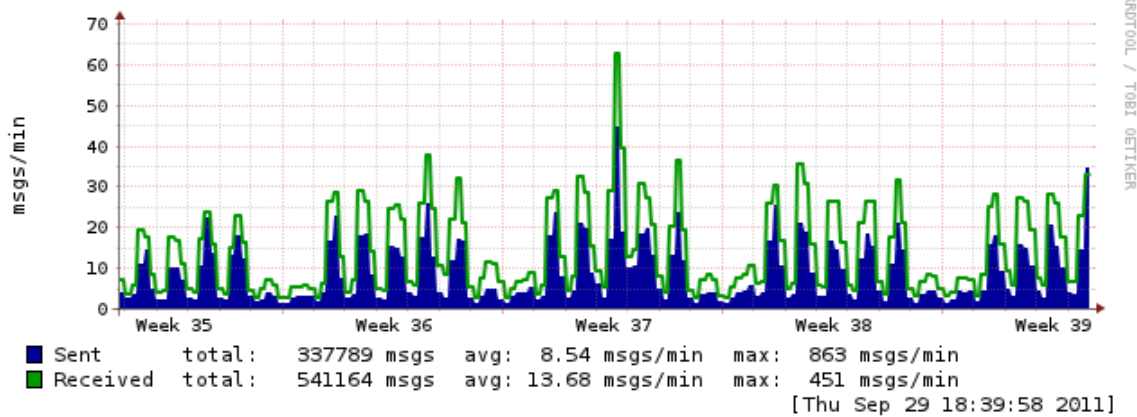


On utilise un outil particulier : **fetchmail**, dont la fonction de base est la collecte à distance et la retransmission du courrier électronique. L'outil fetchmail peut être utilisé comme passerelle POP ou IMAP pour un domaine DNS entier.

Cette architecture «domestique» peut paraître d'une complexité trop importante relativement à la charge utile de traitement du courrier électronique. Il faut cependant prendre en compte les arguments suivants :

- La première approche de sécurisation consiste à faire appels aux fonctions antispam et antivirus à partir du *Mail delivery Agent*. Comme indiqué au point **MDA(3)**, le coût d'exécution des fonctions de sécurité gêne considérablement l'utilisateur dans la consultation du courrier électronique. Cette gêne est d'autant plus importante qu'il n'existe pas de relations entre les fonctions de sécurité. Un même courrier doit passer par toutes les fonctions avant qu'une décision soit prise.
- Il ne faut pas oublier que si les «pourriels» (et/ou) les virus arrivent jusqu'à l'interface réseau de votre poste, c'est que les opérateurs n'ont pas joué leur rôle. Il n'est pas rare aujourd'hui, de devoir rejeter plus de 80% du courrier arrivant sur l'interface externe d'une passerelle.

Last Month



Échantillon de statistiques amavisd-new sur un mois

D'après les données ci-dessus, le total des messages bloqués est : $435675 + 31880 + 487 = 468042$

Le total des messages reçus est : 541164

Le pourcentage de messages rejeté est : $468042 / 541164 * 100 = 86,5\%$

Dans ces conditions, la mise en place de cette configuration de traitement du courrier électronique, même à titre domestique, se justifie pleinement.

3. Le service amavisd-new

Le service **amavisd-new** n'est pas unique en son genre, mais il n'existe pas beaucoup d'équivalents. Citons juste **MIMEDefang**⁴ qui s'appuie sur **milter**⁵ pour composer un système de filtrage centralisé du courrier électronique.

Le service **amavisd-new** est une branche de la famille **Amavis**. Parmi les autres membres on trouve : **amavis-perl**, **amavisd** et **amavis-ng**. Aujourd'hui, le code du service **amavisd-new** est très éloigné des versions initiales d'**amavis-perl**.

Le code du service **amavisd-new** a été complètement revu. Il se distingue par l'utilisation de démons pour lesquels toutes les phases d'initialisations sont effectuées avant de commencer les traitements sur le courrier électronique. C'est ce «pré-chargement» de démon pour chaque fonction de sécurité qui assure une tenue en charge très supérieure à celle obtenue avec les outils classiques qui recourent massivement au système de fichiers.

3.1. L'installation

Bien qu'il existe un paquet Debian dont l'évolution est présentée à la page : **amavisd-new source package**⁶, celui-ci n'est pas mis à jour assez fréquemment pour pouvoir coller à l'actualité de la sécurité du courrier électronique.

Le présent document décrit donc l'installation du service **amavisd-new** à partir de ses sources disponibles à la page : **amavisd-new**⁷. Comme la distribution utilisée est **Debian GNU/Linux**, l'organisation des fichiers et répertoires du service essaie de se conformer aux recommandations de la **Charte Debian**.

Enfin, ce qui suit ne peut se substituer à la documentation officielle sur l'installation du service : le fichier **INSTALL**⁸.

3.1.1. Le téléchargement des sources et les dépendances

Rien de bien original pour ce qui concerne l'obtention des sources :

```
$ wget http://www.ijs.si/software/amavisd/amavisd-new.tar.gz
$ su
# mv amavisd-new.tar.gz /usr/local/src ; cd /usr/local/src
# tar xf amavisd-new.tar.gz
# chown -R root.src amavisd-new-2.7.x
# cd amavisd-new-2.7.x
# cp amavisd amavisd-agent amavisd-nanny amavisd-release p0f-analyzer.pl \
  /usr/local/sbin
```

Pour ce qui est des autres logiciels nécessaires au fonctionnement du service, il faut comparer la liste fournie dans le fichier **INSTALL**⁹ et les dépendances du paquet Debian.

L'exécution de la commande `$ apt-cache depends amavisd-new` donne la liste des paquets nécessaires et recommandés pour le fonctionnement du service.

On affiche les versions de la liste des paquets ci-dessus pour faire la correspondance avec le fichier **INSTALL**¹⁰ à l'aide de la commande suivante.

```
$ dpkg -l file libconvert-tnef-perl \
  libconvert-uulib-perl libcompress-zlib-perl libarchive-zip-perl \
  libmailtools-perl libmime-perl libunix-syslog-perl libnet-server-perl \
  perl spamassassin clamav clamav-daemon lha arj rar zoo nomarch cpio lzop pax
```

Comme les numéros de version évoluent régulièrement, donner le résultat des deux commandes précédentes ne présente pas un grand intérêt. Il faut simplement suivre régulièrement l'évolution des paquets en question de façon à ce que la suite des outils soit pleinement efficace.

Une fois les paquets correctement installés, on contrôle la disponibilité des fonctions en lançant le service en mode *debug*. Dans ce mode, l'exécution du démon lance un processus non détaché du *shell* courant et tous les messages sont directement envoyés dans le terminal. On lance la commande `# amavisd debug`.

3.2. La gestion de l'arborescence de quarantaine

Une politique minimum de gestion des messages mis en quarantaine est nécessaire pour ne pas saturer l'espace de stockage d'une passerelle de courrier électronique. Ici, on se fixe comme règle de conserver les spams pendant 90 jours et les messages infectés pendant 6 jours. Ainsi, si un utilisateur du service de messagerie émet une réclamation sur un message à l'administrateur de la passerelle, il est toujours possible de retrouver le message en question et d'extraire manuellement une partie non infectée. Il s'agit plus d'une précaution d'usage que d'une pratique d'exploitation sachant qu'un message reconnu comme porteur de virus est rarement réclamé par son émetteur ou son destinataire.

⁴ <http://www.mimedefang.org/>

⁵ http://www.milter.org/milter_api/

⁶ <http://packages.qa.debian.org/a/amavisd-new.html>

⁷ <http://www.ijs.si/software/amavisd/>

⁸ <http://www.ijs.si/software/amavisd/INSTALL>

⁹ <http://www.ijs.si/software/amavisd/INSTALL>

¹⁰ <http://www.ijs.si/software/amavisd/INSTALL>

Par configuration du service amavisd-new, le répertoire de quarantaine est généralement : `/var/lib/amavis/virusmails/`. Il contient l'ensemble des messages interceptés.

L'application des règles énoncées ci-avant, se fait à l'aide d'un simple script shell placé dans le répertoire `/etc/cron.daily`. Voici le code du script `quarantine-cleanup` :

```
#!/bin/bash

if [ -d /var/lib/amavis/virusmails/ ]; then
  find /var/lib/amavis/virusmails/ -mtime +90 -exec rm -rf {} \;
fi

for file in `find /var/lib/amavis/virusmails/ -type f -name "virus*" -mtime +6 \
  -exec grep -l '^X-Amavis-Alert: INFECTED' {} \;`; do
  rm -rf $file
done

exit 0
```

En ayant placé le script dans le répertoire indiqué, le service de planification `cron` lance son exécution tous les jours à 6h25.

3.3. La libération d'un message en quarantaine

Le service amavisd-new prévoit que l'on puisse libérer un message placé en quarantaine par erreur. Même si le nombre de faux positifs est extrêmement faible, il est important de pouvoir effectuer des recherches pour identifier et forcer la transmission d'un message placé en quarantaine.

L'archive de distribution du service contient un utilitaire dédié appelé `amavisd-release` que l'on place habituellement dans le même répertoire que le démon du service : `/usr/local/sbin`.

Il faut intervenir sur la configuration du service en éditant le fichier `amavisd.conf` pour autoriser le fonctionnement de l'utilitaire.

```
$policy_bank{'AM.PDP-SOCK'} = {
  protocol => 'AM.PDP', # Amavis policy delegation protocol
  auth_required_release => 0, # don't require secret_id for amavisd-release
};
<snip/>
$interface_policy{'SOCK'} = 'AM.PDP-SOCK';
```

Ces deux modifications du fichier de configuration principal du service ont pour but d'autoriser l'accès au `socket` UNIX du démon `amavisd` via l'utilitaire `amavisd-release`.

Sachant que par configuration, on a choisi de placer les `sockets` dans le répertoire `/var/run/amavis`, il faut aussi éditer le fichier source de l'utilitaire pour qu'il communique avec le bon fichier de `socket`.

```
$ ll /var/run/amavis/amavisd.sock
srwxr-x--- 1 amavis amavis 0 2007-03-21 17:26 /var/run/amavis/amavisd.sock
```

La modification correspondante dans le code de l'utilitaire `amavisd-release` donne :

```
$ diff -uBb /usr/local/src/amavisd-new-2.4.5/amavisd-release amavisd-release
--- /usr/local/src/amavisd-new-2.4.5/amavisd-release      2006-08-08 20:14:47.000000000 +0200
+++ amavisd-release      2007-03-21 17:24:23.000000000 +0100
@@ -76,7 +76,7 @@

  $log_level = 1;
  # $socketname = '127.0.0.1:9998';
- $socketname = '/var/amavis/amavisd.sock';
+ $socketname = '/var/run/amavis/amavisd.sock';

  sub sanitize_str {
    my($str, $keep_eol) = @_;
```

Maintenant que l'utilitaire est prêt à fonctionner, il faut trouver un faux positif en quarantaine à libérer. Nous allons prendre un cas classique de service publicitaire sensible pour les utilisateurs et la réputation de l'administrateur du service de courrier électronique : une agence de voyage !.

Comme on a pris soin de journaliser toutes les transactions du service amavisd-new, on commence par rechercher les références d'un message mis en quarantaine avec l'adresse de notre fameuse agence de voyage.

```
$ for (( i=2 ; i < 50 ; i++ ))❶ ; do \
zcat /var/log/amavis.mail.info.$i.gz |grep "fram\.fr"❷ |grep "Blocked" ; \
done

Feb  9 08:59:23 MailGw amavis[30460]: (30460-14) Blocked SPAM,❸ [aaa.bbb.ccc.ddd] \
<xxxxxxxx@fram.fr> -> <DIRECTION@xxxxx.FR>, quarantine: l/spam-l1bnIZN88okd.gz❹, \
Message-ID: <45CC2AC0.2090800@fram.fr>, mail_id: l1bnIZN88okd, Hits: 8.43,❺ 4730 ms
```

- ❶ Avec une rotation quotidienne des journaux du service de courrier électronique, la recherche est effectuée sur les 50 derniers jours moins les deux les plus récents pour lesquels les journaux ne sont pas compressés.
- ❷ La scrutation recherche une ligne comprenant le domaine de l'adresse de courrier électronique suspecte ainsi que le mot clé `Blocked` indiquant que le courrier n'a pas été transmis à son destinataire.

- 35 Le résultat de la recherche montre qu'un message émis depuis le domaine suspect a bien été considéré comme un *spam* et qu'il a obtenu un score supérieur au seuil de mise en quarantaine mais relativement faible comparé au flot de *spams* habituel.

Dans cet exemple, le seuil de mise en quarantaine est de 6.31 et le score atteint est de 8.43. Les scores classiques obtenus par les *spams* dépassent facilement les 20 points.

- 4 Le même résultat de recherche désigne la référence du fichier de quarantaine. C'est cette référence qui doit être utilisée par l'utilitaire `amavisd-release` pour extraire le message et le transmettre au MTA.

```
# amavisd-release l/spam-11bnIZN88okd.gz
250 2.6.0 Ok, xml:id=rel-11bnIZN88okd, from MTA([127.0.0.1]:10025): 250 2.0.0 Ok: queued as AF6DF4E8004
```

L'appel à l'utilitaire `amavisd-release` indique que le message a été transmis au gestionnaire de mise en file d'attente du service de courrier électronique (MTA).

Dernière remarque très importante ; toutes ces opérations ont été effectuées sans *jama*s consulter le contenu du moindre message. On respecte ainsi caractère privé du courrier électronique.

Comme les journaux peuvent être déposés sur un serveur dédié et que l'utilitaire `amavisd-release` peut communiquer via un *socket inet*, il est possible d'organiser la délégation d'administration de façon à ce qu'il soit totalement impossible à l'administrateur responsable de la «libération» des messages d'en connaître le contenu.

3.4. La reconnaissance passive d'empreinte de système d'exploitation

Depuis la version 2.4.0, le service `amavisd-new` propose de compléter la pondération des scores calculés par `spamassassin` en exploitant les informations fournies par le scanner passif `p0f`¹¹.

Il existe une condition importante pour l'exploitation de cet outil. L'adresse IP de l'hôte pair du dialogue SMTP doit être transmise au service `amavisd-new` par l'agent de transport de courrier électronique (*Mail Transfer Agent* ou MTA). Dans le cas de Postfix, l'extension `XFORWARD` doit être activée. Voici un extrait du fichier de configuration `master.cf` de Postfix.

```
smtp-amavis unix - - y - 2 smtp
-o smtp_data_done_timeout=1200
-o smtp_send_xforward_command=yes
-o max_use=20
```

Le fichier `RELEASE_NOTES`¹² contient les instructions de configuration de transfert des résultats de `p0f` vers `amavisd-new` puis `spamassassin`. Le principe de fonctionnement est le suivant :

1. Le scanner `p0f` est exécuté de façon à identifier le système d'exploitation de l'hôte pair lors du dialogue SMTP. Il intercepte donc les requêtes sur le port destination `tcp/25` et génère des rapports sur la console courante.
2. Le programme `p0f-analyzer.pl`, fourni avec la distribution du service `amavisd-new`, lit le rapport du scanner `p0f` sur la console courante (flux `stdin`). Ce programme conserve en cache pendant 10 minutes les rapports générés par `p0f`. Ce même programme reste en écoute sur le port `udp/2345` de l'interface de boucle locale. Tous les paramètres indiqués ici sont soit des valeurs par défaut soit donnés dans la ligne de commande.
3. Le service `amavisd-new` interroge le programme `p0f-analyzer.pl` qui maintient les rapports du scanner en cache. Les informations collectées sont transmises à `spamassassin` via un champ d'en-tête inséré spécifiquement. Il faut éditer le fichier de configuration du service pour activer cette fonctionnalité.

Pour mettre en place cet outil, on effectue les opérations suivantes :

- Installation du paquet `p0f`.

```
# dpkg -l p0f
Souhait=inconnU/Installé/supprimé/Purgé/H=à garder
| État=Non/Installé/fichier-Config/dépaqueté/échec-Config/H=semi-installé
|/ Err?=(aucune)/H=à garder/besoin Réinstallation/X=Les deux (État,Err: maj=mauvais)
||/ Nom Version Description
+++-----
ii p0f 2.0.8-2 Passive OS fingerprinting tool
```

- Lancement du scanner et du gestionnaire de cache de rapports d'identification.

```
# p0f -l -i eth0 'dst host <MailGw_IP_address> and tcp dst port 25' 2>&1 \
| p0f-analyzer.pl 2345 &
```

- Édition du fichier `amavisd.conf` pour décommenter la ligne d'appel au programme `p0f-analyzer.pl`.

```
# grep p0f /etc/amavisd.conf
$os_fingerprint_method = 'p0f:127.0.0.1:2345'; # query p0f-analyzer.pl
```

- Insertion des scores à intégrer dans les calculs de `spamassassin` dans le fichier de configuration global.

```
# echo # p0f \
```

¹¹ <http://lcantuf.coredump.cx/p0f.shtml>

¹² <http://www.ijs.si/software/amavisd/release-notes.txt>

```
header L_P0F_WXP X-Amavis-05-Fingerprint =~ /^Windows XP/ \
score L_P0F_WXP 3.5 \
header L_P0F_W X-Amavis-05-Fingerprint =~ /^Windows(?! XP)/ \
score L_P0F_W 1.7 \
header L_P0F_UNKN X-Amavis-05-Fingerprint =~ /^UNKNOWN/ \
score L_P0F_UNKN 0.8 \
header L_P0F_Unix X-Amavis-05-Fingerprint =~ /^(Free|Open|Net)BSD|Solaris|HP-UX|Tru64)/ \
score L_P0F_Unix -1.0 \
>> /etc/spamassassin/local.cf
```

- Après redémarrage du service amavisd-new, on peut valider le fonctionnement de l'identification passive de système d'exploitation en consultant les journaux système. Dans l'exemple ci-dessous, on constate que le score a été augmenté à partir de la correspondance : L_P0F_W=1.7.

```
MailGw amavis[23777]: (23777-08) OS_fingerprint: 88.242.7.27 \
46.023 Windows 2000 SP4, XP SP1, (distance 19, link: unknown-1460)
MailGw amavis[23777]: (23777-08) SPAM, <xxxx@xxx-xx.com> -> <xxxxx@xxx-xxx3.fr>, \
Yes, score=46.023 tag=-999 tag2=6.31 kill=6.31 tests=[BAYES_95=5, DCC_CHECK=4.5, \
DIGEST_MULTIPLE=0.765, HTML_MESSAGE=0.001, L_P0F_W=1.7, MIME_HTML_ONLY=0.001, \
NO_RECEIVED=-0.001, NO_RELAYS=-0.001, RAZOR2_CF_RANGE_51_100=3.5, \
RAZOR2_CF_RANGE_E4_51_100=1.5, RAZOR2_CF_RANGE_E8_51_100=1.5, \
RAZOR2_CHECK=2.5, SARE_SPEC_REPLICA_OBFU=1.812, SARE_SPEC_ROLEX_NOV5A=1.062, \
URIBL_AB_SURBL=3.812, URIBL_BLACK=3, URIBL_JP_SURBL=4.087, URIBL_OB_SURBL=3.008, \
URIBL_SBL=1.639, URIBL_SC_SURBL=4.498, URIBL_WS_SURBL=2.14], \
autolearn=spam, quarantine ZLIibG52Rt-A (spam-quarantine)
```

4. La lutte contre les pourriels

L'outil de lutte contre les pourriels (*spams*) le plus répandu dans le monde du logiciel libre est **spamassassin**¹³. Son mode de fonctionnement se rapproche de celui du service **amavisd-new**. Il collecte les résultats obtenus par une collection d'autres outils et les transmet au service **amavisd-new** qui prend la décision sur le message en fonction du score total obtenu par celui-ci.

Du point de vue système, la mise en oeuvre de **spamassassin**¹⁴ est très simple. Il suffit de s'appuyer sur le paquet Debian/testing qui est très bien maintenu. Cette situation est d'autant plus intéressante que la collection des bibliothèques et des outils dépendants de spamassassin est très volumineuse. La commande `$ apt-cache show spamassassin` liste ces dépendances.

4.1. La configuration de spamassassin

Il faut distinguer 2 niveaux pour la configuration de cet outil.

- À l'échelle du système, le répertoire `/etc/spamassassin` contient les fichiers de configuration communs à l'ensemble des utilisateurs.
- Pour l'utilisateur `amavis`, le répertoire `.spamassassin` situé sous le répertoire personnel de l'utilisateur contient les fichiers de configuration dédiés ainsi que les bases de données constituées en cours d'exécution : liste blanche automatique, jetons de calcul, etc..

Comme dans les scénarios décrits dans ce document, le système sur lequel sont installés les outils est un serveur passerelle dédié au traitement du courrier électronique, tous les paramètres de configuration seront dans le répertoire général `/etc/spamassassin` et toutes les bases de données seront dans le répertoire `/var/lib/amavis/.spamassassin`.

4.1.1. Le fichier de configuration principal

Le fichier `local.cf` regroupe les options principales du service. Voici un exemple de paramètres adaptés à l'exploitation d'une passerelle de courrier électronique. En plus des paramètres propres à spamassassin, on y trouve des éléments de sélection des outils complémentaires.

```
# cat /etc/spamassassin/local.cf
# This is the right place to customize your installation of SpamAssassin.
#
# See 'perldoc Mail::SpamAssassin::Conf' for details of what can be
# tweaked.
#
# Only a small subset of options are listed below
#
#####
dns_available yes

whitelist_from root@MailGw
whitelist_from amavis@MailGw
whitelist_from logcheck@MailGw

# Enable the Bayes system
use_bayes 1
bayes_auto_expire 0
```

¹³ <http://spamassassin.apache.org/>

¹⁴ <http://spamassassin.apache.org/>

```

bayes_learn_to_journal 1
bayes_journal_max_size 0
bayes_path /var/lib/amavis/.spamassassin/bayes

use_auto_whitelist 1

# Text to prepend to subject if rewrite_subject is used
add_header all Report _REPORT_

# DCC
use_dcc 1
dcc_timeout 8

# Razor
use_razor2 1
razor_timeout 8

# Pyzor
use_pyzor 1

## Optional Score Increases
score DCC_CHECK 4.500
score SPF_FAIL 10.000
score SPF_HELO_FAIL 10.000
score RAZOR2_CHECK 2.500
score RAZOR2_CF_RANGE_51_100 3.500
score BAYES_99 5.300
score BAYES_95 4.500
score BAYES_80 3.500
score BAYES_60 2.500
score BAYES_50 2.000

# p0f
header L_P0F_WXP X-Amavis-0S-Fingerprint =~ /(^Windows XP/
score L_P0F_WXP 3.5
header L_P0F_W X-Amavis-0S-Fingerprint =~ /(^Windows(?: XP)/
score L_P0F_W 1.7
header L_P0F_UNKN X-Amavis-0S-Fingerprint =~ /(^UNKNOWN/
score L_P0F_UNKN 0.8
header L_P0F_Unix X-Amavis-0S-Fingerprint =~ /((Free|Open|Net)BSD|Solaris|HP-UX|Tru64)/
score L_P0F_Unix -1.0

```

4.2. La mise à jour des jeux de règles de spamassassin

La pondération des champs d'un message électronique est basée sur des jeux de règles prédéfinies. Comme de nouveaux styles de pourriels apparaissent constamment, il est nécessaire de mettre à jour ces règles périodiquement.

Le paquet spamassassin fournit un outil dédié à cette tâche : **sa-update**. L'option **-D** de *debugging* affiche les informations intermédiaires du traitement.

```

# sa-update -D
dbg: logger: adding facilities: all
dbg: logger: logging level is DBG
dbg: generic: SpamAssassin version 3.3.0
dbg: generic: Perl 5.010001, PREFIX=/usr, \
DEF_RULES_DIR=/usr/share/spamassassin, \
LOCAL_RULES_DIR=/etc/spamassassin, \
LOCAL_STATE_DIR=/var/lib/spamassassin
dbg: config: timing enabled
dbg: config: score set 0 chosen.
dbg: dns: is Net::DNS::Resolver available? yes
dbg: dns: Net::DNS version: 0.65
dbg: generic: sa-update version svn897929
dbg: generic: using update directory: /var/lib/spamassassin/3.003000
dbg: diag: perl platform: 5.010001 linux
dbg: diag: [...] module installed: Digest::SHA1, version 2.12
dbg: diag: [...] module installed: HTML::Parser, version 3.64
dbg: diag: [...] module installed: Net::DNS, version 0.65
dbg: diag: [...] module installed: NetAddr::IP, version 4.024
dbg: diag: [...] module installed: Time::HiRes, version 1.9719
dbg: diag: [...] module installed: Archive::Tar, version 1.52
dbg: diag: [...] module installed: IO::Zlib, version 1.09
dbg: diag: [...] module installed: Digest::SHA1, version 2.12
dbg: diag: [...] module installed: MIME::Base64, version 3.08
dbg: diag: [...] module installed: DB_File, version 1.82
dbg: diag: [...] module installed: Net::SMTP, version 2.31
dbg: diag: [...] module installed: Mail::SPF, version v2.007
dbg: diag: [...] module installed: IP::Country::Fast, version 604.001
dbg: diag: [...] module installed: Razor2::Client::Agent, version 2.84
dbg: diag: [...] module installed: Net::Ident, version 1.20
dbg: diag: [...] module installed: IO::Socket::INET6, version 2.54
dbg: diag: [...] module installed: IO::Socket::SSL, version 1.31
dbg: diag: [...] module installed: Compress::Zlib, version 2.022
dbg: diag: [...] module installed: Mail::DKIM, version 0.37
dbg: diag: [...] module installed: DBI, version 1.609
dbg: diag: [...] module installed: Getopt::Long, version 2.38
dbg: diag: [...] module installed: LWP::UserAgent, version 5.834
dbg: diag: [...] module installed: HTTP::Date, version 5.831

```

```

dbg: diag: [...] module installed: Encode::Detect, version 1.01
dbg: gpg: Searching for 'gpg'
dbg: util: current PATH is: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
dbg: util: executable for gpg was found at /usr/bin/gpg
dbg: gpg: found /usr/bin/gpg②
dbg: gpg: release trusted key id list: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
dbg: channel: attempting channel updates.spamassassin.org
dbg: channel: update directory /var/lib/spamassassin/3.003000/updates_spamassassin_org
dbg: channel: channel cf file /var/lib/spamassassin/3.003000/updates_spamassassin_org.cf
dbg: channel: channel pre file /var/lib/spamassassin/3.003000/updates_spamassassin_org.pre
dbg: channel: metadata version = 903765
dbg: dns: 0.3.3.updates.spamassassin.org => 903765, parsed as 903765③
dbg: channel: current version is 903765, new version is 903765, skipping channel
dbg: diag: updates complete, exiting with code 1

```

- ① Le traitement débute par une vérification de la liste des bibliothèques et outils nécessaires à l'exploitation des jeux de règles de spamassassin.
- ② Les transferts de jeux de règles sont signés pour éviter les falsifications.
- ③ Les champs TXT du serveur de noms de domaines DNS sont utilisés pour véhiculer les informations de version des jeux de règles. Dans l'exemple, la version sur le serveur distant est identique à celle présente localement 473378. Il n'est donc pas nécessaire de télécharger les fichiers de règles de pondération.

4.3. Les calculs distribués avec Pyzor et Razor



Note

Si les outils présentés dans cette section n'occupent pas le «devant de la scène» en matière de lutte contre les pourriels, ils n'en sont pas moins essentiels au bon fonctionnement d'une passerelle de courrier électronique digne de ce nom. Après plus de cinq ans d'exploitation, on peut affirmer qu'ils contribuent efficacement aux calculs délivrés par spamassassin et donc aux décisions prises par le service amavisd-new.

Les deux outils Pyzor et Razor appartiennent à la même catégorie. Ce sont des réseaux collaboratifs de partage de signatures de spams en cours de propagation. Les signatures peuvent être assimilées à des sommes de contrôle (*checksums*) calculées à partir d'éléments clés du message de courrier électronique. Les utilisateurs du réseau contribuent au catalogue en transmettant les sommes de contrôles pour les messages qu'ils émettent et exploitent les calculs stockés dans le catalogue partagé lorsqu'ils reçoivent les messages.

Ces deux outils peuvent être utilisés de façon autonome mais ce n'est pas la solution retenue dans notre cas. Avec spamassassin, les résultats obtenus à partir des sommes de contrôle sont intégrés dans le calcul global qui sert à déterminer si un courrier est un pourriel ou non. Pyzor et Razor constituent donc des sources d'alimentation supplémentaires pour spamassassin.

Voici une présentation de l'installation et de la configuration de ces trois outils dans le contexte du service amavisd-new.

4.3.1. Pyzor

On débute par l'installation du paquet et le contrôle de sa version.

```

$ aptitude search 'pyzor'
i pyzor - spam-catcher using a collaborative filtering network

```

Une fois le paquet installé, la configuration doit être appliquée à l'utilisateur amavis.

```

# su amavis -c 'pyzor discover'
downloading servers from http://pyzor.sourceforge.net/cgi-bin/inform-servers-0-3-x
# cat /var/lib/amavis/.pyzor/servers
public.pyzor.org:24441

```

Le service étant configuré, on peut contrôler la disponibilité du serveur de distribution du catalogue des sommes de contrôle.

```

# su amavis -c 'pyzor ping'
public.pyzor.org:24441 (200, 'OK')

```

Avec la version courante du paquet Debian, un message d'alerte apparaît à chaque exécution. Ce message rend l'outil inopérant.

```

pyzor: failure to parse response
"/usr/lib/pymodules/python2.6/pyzor/__init__.py:11: DeprecationWarning: the sha
module is deprecated; use the hashlib module instead"

```

Pour contourner le problème, il faut neutraliser les messages d'alerte lors de l'exécution de l'outil. Voici une copie du fichier `/usr/bin/pyzor`.

```

#!/usr/bin/python -Wignore::DeprecationWarning

import os
# set umask
os.umask(0077)

import pyzor.client

```

```
pyzor.client.run()
```

4.3.2. Razor

On débute par l'installation du paquet et le contrôle de sa version.

```
$ aptitude search razor
v  librazor2-perl -
p  posterazor    - splits an image into multiple sheets for assembly into a poster
i  razor         - spam-catcher using a collaborative filtering network
```

Une fois le paquet installé, il faut distinguer deux niveaux de configuration : système et utilisateur. Dans le cas présent, on doit créer la configuration pour l'utilisateur amavis. Cette étape est plus complexe que pour l'outil précédent.

Au niveau global ou système, on efface la configuration existante de façon à mettre en place une nouvelle configuration.

```
# rm /etc/razor/razor-agent.conf
# razor-admin -create
# cp ~/.razor/razor-agent.conf /etc/razor/
```

Le fichier de configuration système est modifié pour que les messages émis par l'outil soient journalisés dans un répertoire spécifique dans lequel le service amavisd-new dispose d'un accès en écriture.

```
# mkdir /var/log/razor
# chown amavis.adm /var/log/razor
# diff -uBb /etc/razor/razor-agent.conf.orig /etc/razor/razor-agent.conf
--- /etc/razor/razor-agent.conf.orig 2011-10-10 22:12:28.000000000 +0200
+++ /etc/razor/razor-agent.conf 2011-10-10 22:10:47.000000000 +0200
@@ -14,7 +14,7 @@
 listfile_catalogue    = servers.catalogue.lst
 listfile_discovery    = servers.discovery.lst
 listfile_nomination   = servers.nomination.lst
 -logfile              = razor-agent.log
 +logfile              = /var/log/razor/razor-agent.log
 logic_method          = 4
 min_cf                = ac
 razordiscovery        = discovery.razor.cloudmark.com
```

Pour éviter une saturation de l'espace de stockage du volume dédié à la journalisation système, on configure la rotation du fichier de journalisation `/var/log/razor/razor-agent.log`.

```
# cat /etc/logrotate.d/razor
/var/log/razor/razor-agent.log {
    weekly
    rotate 52
    compress
    nomail
    notifempty
    missingok
    create 640 amavis adm
}
```

Au niveau utilisateur, on enregistre le service.

```
# mkdir /var/lib/amavis/.razor
# chown amavis.amavis /var/lib/amavis/.razor
# su amavis -c 'razor-admin -home /var/lib/amavis/.razor -register'
Register successful. Identity stored in /var/lib/amavis/.razor/identity-rumyOpNfhL
```

Le répertoire `/var/lib/amavis/.razor/` contient l'ensemble des éléments de la configuration utilisateur.

4.3.3. Tests de fonctionnement

Pour qualifier l'utilisation des deux outils, il faut étudier les rapports délivrés par spamassassin.

La première méthode consiste à tester un échantillon de pourriel fourni avec les sources du service amavisd-new.

```
# su amavis -c 'spamassassin -D </usr/local/src/amavisd-new-2.7.0/test-messages/sample-spam-GTUBE-junk.txt >test-report.txt 2>
```

Une fois cette commande exécutée, le rapport peut être consulté avec **less** par exemple. On peut aussi effectuer des recherches de chaînes de caractères avec **egrep**. Voici un extrait qui montre que les deux outils ont bien été appelés. Seul Pyzor a détecté un *spam*.

```
# egrep -i '(pyzor|razor)' test-report.txt
dbg: plugin: loading Mail::SpamAssassin::Plugin::Pyzor from @INC
dbg: pyzor: network tests on, attempting Pyzor
dbg: plugin: loading Mail::SpamAssassin::Plugin::Razor2 from @INC
dbg: razor2: razor2 is available, version 2.84
dbg: config: fixed relative path: /var/lib/spamassassin/3.003001/updates_spamassassin_org/25_pyzor.cf
dbg: config: using "/var/lib/spamassassin/3.003001/updates_spamassassin_org/25_pyzor.cf" for included file
dbg: config: read file /var/lib/spamassassin/3.003001/updates_spamassassin_org/25_pyzor.cf
dbg: config: fixed relative path: /var/lib/spamassassin/3.003001/updates_spamassassin_org/25_razor2.cf
dbg: config: using "/var/lib/spamassassin/3.003001/updates_spamassassin_org/25_razor2.cf" for included file
dbg: config: read file /var/lib/spamassassin/3.003001/updates_spamassassin_org/25_razor2.cf
dbg: razor2: part=0 noresponse
dbg: razor2: results: spam? 0
```

```

dbg: razor2: results: engine 8, highest cf score: 0
dbg: razor2: results: engine 4, highest cf score: 0
dbg: util: executable for pyzor was found at /usr/bin/pyzor
dbg: pyzor: pyzor is available: /usr/bin/pyzor
dbg: pyzor: opening pipe: /usr/bin/pyzor check < /tmp/.spamassassin17676Ek7f7Utmp
dbg: pyzor: [17677] finished successfully
dbg: pyzor: got response: public.pyzor.org:24441 (200, 'OK') 304 0
dbg: pyzor: listed: COUNT=304/5 WHITELIST=0
dbg: rules: ran eval rule PYZOR_CHECK =====> got hit (1)
dbg: check: tests=GTUBE,NO_RECEIVED,NO_RELAYS,PYZOR_CHECK
dbg: timing: total 2001 ms - init: 1136 (56.8%), parse: 1.21 (0.1%),
  extract_message_metadata: 21 (1.1%), get_uri_detail_list: 0.49 (0.0%),
  tests_pri_-1000: 7 (0.4%), compile_gen: 107 (5.3%), compile_eval: 21 (1.1%),
  tests_pri_-950: 6 (0.3%), tests_pri_-900: 6 (0.3%), tests_pri_-400: 5 (0.2%),
  tests_pri_0: 743 (37.1%), dkim_load_modules: 23 (1.2%), check_dkim_signature:
  0.45 (0.0%), check_dkim_adsp: 180 (9.0%), check_spf: 30 (1.5%), check_dcc:
  0.21(0.0%), check_razor2: 209 (10.4%), check_pyzor: 80 (4.0%), tests_pri_500:
  56(2.8%), tests_pri_1000: 11 (0.5%), total_awl: 6 (0.3%), check_awl: 0.56
  (0.0%), update_awl: 1.70 (0.1%)
dbg: markup: [...] * 2.0 PYZOR_CHECK Message list=e9 par Pyzor, voir http://pyzor.sf.net/
  NO_RELAYS,PYZOR_CHECK autolearn=no version=3.3.1
  * 2.0 PYZOR_CHECK Message listé par Pyzor, voir http://pyzor.sf.net/

```

La seconde méthode consiste à consulter les journaux du service amavisd-new en cours de fonctionnement. Voici un exemple de résultat de calcul sur un message dont la nature ne fait aucun doute. On y voit clairement apparaître les coefficients attribués par Pyzor et Razor.

```

header_edits_for_quar: <xxxxxx@xxxxxxxxxxxx.fr> -> <xxxxxxxx@xxxxxxxxxxxx.fr>,
Yes, score=48.957 tag=-.999 tag2=6.31 kill=6.31
tests=[AV:Sanesecurity.Junk.38228.UNOFFICIAL=0.1, DIGEST_MULTIPLE=0.001,
FROM_IN_TO_AND_SUBJ=3.399, HELO_DYNAMIC_IPADDR2=3.888,
HELO_DYNAMIC_SPLIT_IP=2.893, HTML_IMAGE_ONLY_28=0.726, HTML_MESSAGE=0.001,
INVALID_MSGID=1.167, L_P0F_UNKN=0.8, MIME_HTML_ONLY=1.105,
MIME_QP_LONG_LINE=0.001, MISSING_DATE=1.396,
MSGID_SHORT=0.337, PYZOR_CHECK=1.985, RAZOR2_CF_RANGE_51_100=0.365,
RAZOR2_CF_RANGE_E8_51_100=2.43, RAZOR2_CHECK=1.729, RCVD_IN_BRBL_LASTEXT=1.644,
RCVD_IN_PBL=3.558, RCVD_IN_SORBS_DUL=0.001, RCVD_IN_XBL=0.724, RDNS_NONE=1.274,
SPF_FAIL=0.919, TO_EQ_FM_DIRECT_MX=0.001, TO_EQ_FM_DOM_SPF_FAIL=0.001,
TO_EQ_FM_HTML_DIRECT=1.753, TO_EQ_FM_HTML_ONLY=0.001, TO_EQ_FM_SPF_FAIL=0.973,
TO_IN_SUBJ=3.899, TVD_RCVD_IP=0.001, URIBL_AB_SURBL=4.499, URIBL_BLACK=1.775,
URIBL_JP_SURBL=1.948, URIBL_WS_SURBL=1.659, X_MAILER_CME_6543_MSÑ=2.004]

```

5. Clam AntiVirus

En quelques années, le projet ClamAV™ est devenu la référence en matière de logiciel libre antivirus. La réactivité de l'équipe de développement est excellente qu'il s'agisse de la vitesse de publication des signatures de nouveaux virus ou des publications de correctifs de sécurité du logiciel lui-même.

L'installation et la configuration de cet antivirus est assez immédiate dans la mesure où le paquet Debian est parfaitement maintenu et son adaptation au service amavisd-new ne demande pas beaucoup d'efforts.

5.1. Paquets Debian à installer

Les paquets Debian à installer pour une utilisation avec le service amavisd-new sont donnés dans la liste suivante :

```

$ dpkg -l *clam* |grep ^ii
ii clamav 0.90.1-2 antivirus scanner for Unix
ii clamav-base 0.90.1-2 base package for clamav, an anti-virus utility
ii clamav-daemon 0.90.1-2 antivirus scanner daemon
ii clamav-freshclam 0.90.1-2 downloads clamav virus databases from the Internet
ii libclamav2 0.90.1-2 virus scanner library

```

Cette liste comprend trois fonctions distinctes.

clamav

Ce paquet contient le *scanner* clamscan que l'on utilise au niveau système de fichiers. Ce mode de traitement est excessivement coûteux en temps d'accès disque et dégrade très sensiblement les performances du traitement du courrier électronique. Le service amavisd-new ne fait appel à ce *scanner* qu'en secours ; lorsque les antivirus primaires travaillant en RAM ne sont plus disponibles.

Vis-à-vis du service amavisd-new, le *scanner* est un antivirus secondaire. On obtient les informations suivantes lors du lancement du service.

```

amavis[3000]: Found secondary av scanner ClamAV-clamscan at /usr/bin/clamscan

```

clamav-daemon

Ce paquet contient le démon clamd qui réside en mémoire de façon permanente et qui communique avec le service amavisd-new via un *socket UNIX*. C'est ce mode de traitement qui donne les meilleures performances. Une fois que le service a extrait tous les objets contenu dans les champs d'un message, il transmet ces objets aux antivirus pour examen. Il est possible de stocker les objets à traiter dans un système de fichiers «monté» en RAM pour accroître encore les performances.

Vis-à-vis du service amavisd-new, le démon est un antivirus primaire. On obtient les informations suivantes lors du lancement du service.

```
amavis[3000]: Using internal av scanner code for (primary) ClamAV-clamd
```

freshclam

Ce paquet contient le démon freshclam qui réside en mémoire de façon permanente et qui collecte périodiquement les nouvelles signatures de virus. Ce démon est aussi chargé de notifier le démon clamd pour qu'il recompose son cache en fonction des nouvelles signatures.

Ce démon n'a aucune interaction avec le service amavisd-new.

5.2. Interaction avec amavisd-new

Comme les communications entre les démons freshclam, clamd et le service amavisd-new sont basées sur un *socket UNIX*, les principales opérations de configuration consistent à organiser l'arborescence et les permissions sur le système de fichiers.

1. On choisit de stocker les informations sur l'état courant des démons dans le répertoire `/var/run/amavis`.
2. Ce répertoire et tous ses objets doivent avoir pour propriétaire l'utilisateur et le groupe `amavis`.

En cours d'exécution, on obtient les informations suivantes.

```
$ ls -lA /var/run/amavis/
total 8
-rw-r----- 1 amavis amavis 0 2006-12-30 19:21 amavisd.lock
-rw-r----- 1 amavis amavis 5 2006-12-30 09:35 amavisd.pid
srwxr-x--- 1 amavis amavis 0 2006-12-30 09:35 amavisd.sock
-rw-rw---- 1 amavis amavis 4 2006-12-30 09:34 clamd.pid
srwxrwxrwx 1 amavis amavis 0 2006-12-30 09:34 clamd.sock
```

Configuration de freshclam

La configuration de ce démon suit les directives du paquet Debian.

```
$ cat /etc/clamav/freshclam.conf
# Automatically created by the clamav-freshclam postinst
# Comments will get lost when you reconfigure the clamav-freshclam package

DatabaseOwner clamav
UpdateLogFile /var/log/clamav/freshclam.log
LogFileMaxSize 0
MaxAttempts 5
# Check for new database 8 times a day
Checks 8
DatabaseMirror db.local.clamav.net
DatabaseMirror database.clamav.net
DatabaseDirectory /var/lib/clamav/
NotifyClamd ❶
DNSDatabaseInfo current.cvd.clamav.net
```

- ❶ La notification du démon clamd est l'élément essentiel de ce fichier de configuration.

Configuration de clamd

La configuration de ce démon suit aussi les directives du paquet Debian.

```
$ cat /etc/clamav/clamd.conf
#Automatically Generated by clamav-base postinst
#To reconfigure clamd run #dpkg-reconfigure clamav-base
#Please read /usr/share/doc/clamav-base/README.Debian.gz for details
LocalSocket /var/run/amavis/clamd.sock ❶
FixStaleSocket
User amavis ❷
AllowSupplementaryGroups
ScanMail
ScanArchive
ArchiveMaxRecursion 5
ArchiveMaxFiles 1000
ArchiveMaxFileSize 10M
ArchiveMaxCompressionRatio 250
ScanRAR
MaxDirectoryRecursion 15
ReadTimeout 180
MaxThreads 12
MaxConnectionQueueLength 15
LogSyslog
LogFile /var/log/clamav/clamav.log
LogTime
LogFileMaxSize 0
PidFile /var/run/amavis/clamd.pid
DatabaseDirectory /var/lib/clamav/
SelfCheck 3600
```

- ❶ Conformément au choix énoncé ci-dessus, on utilise le répertoire `/var/run/amavis/` pour stocker le fichier *socket*.
- ❷ Le propriétaire du processus clamd est l'utilisateur `amavis`.

Utilisateur clamav

L'utilisateur clamav doit faire partie du groupe amavis.

```
$ grep clam /etc/group
amavis:!:109:clamav
clamav:!:112:
```

5.3. Configuration du service amavisd-new

C'est dans la *Section VII - External programs, virus scanners* du fichier `amavisd.conf` que l'on trouve les paramètres d'appel à l'antivirus `clamd`. Il faut désigner le fichier *socket* tel qu'il a été défini ci-avant : `/var/run/amavis/clamd.sock`

```
# ### http://www.clamav.net/
['ClamAV-clamd',
 \&ask_daemon, ["CONTSCAN {}\n", "/var/run/amavis/clamd.sock"],
 qr/\bOK$/, qr/\bFOUND$/,
 qr/^.*?: (?!Infected Archive)(.*) FOUND$/ ],
## NOTE: run clamd under the same user as amavisd, or run it under its own
## uid such as clamav, add user clamav to the amavis group, and then add
## AllowSupplementaryGroups to clamd.conf;
## NOTE: match socket name (LocalSocket) in clamav.conf to the socket name in
## this entry; when running chrooted one may prefer socket "$MYHOME/clamd".
```

6. Les échantillons relevés

Voici quelques échantillons relevés lors de l'utilisation du service `amavisd-new`. Certaines informations telles que les adresses de courrier électronique ont été masquées par des caractères `x`.

6.1. Module MIME::Tools

Ces 2 exemples sont issus de deux installations du type : `sendmail - amavis-milter - amavisd-new`.

Il est important de disposer d'un module de désassemblage MIME de bonne qualité pour pouvoir traiter les cas anormaux. Il faut absolument éviter qu'un message à la composition MIME volontairement erronée suffise à sauter les appels aux antivirus. Dans le cas ci-dessous, un `'W32/Dumaru-Y'` est «accroché» à un message avec un en-tête MIME erroné.

```
mai 5 10:38:11 MailGw amavis[8378]: (i458c9dV005475) warning - MIME::Parser
error: BadHead: couldn't parse header; ProblemNear:; name="accounts.zip";
Content-Transfer-Encoding: base64; Content-Disposition: attachment;;
filename="myphoto.zip"; error: UnexpectedBound: part didn't end with expected
boundary [in multipart messag...

May 5 10:38:11 MailGw sophie[831]: WARNING : Scan result =>
'/var/lib/amavis/amavis-milter-i458c9dV005475/parts/part-00004' infected with
virus 'W32/Dumaru-Y'
```

Dans ce second exemple, un autre message avec une mauvaise construction MIME contient un fichier joint exécutable. La configuration du service interdit la transmission de fichiers `.exe` par courrier électronique.

```
mai 5 10:29:54 MailGw amavis[22332]: (i458Trqb012919) warning - MIME::Parser
error: UnexpectedBound: part didn't end with expected boundary [in multipart
message]; EOSToken: CLOSE 3692B18230B.1083745793/XXXXXXXXX.XXXXX.XXX; EOSType:
EXT; error: SeveredParts: unexpected end of parts before epilogue [in multipart
message]; Virtu...

mai 5 10:29:54 MailGw amavis[22332]: (i458Trqb012919) NOTICE: DSN contains
BANNED NAME; bounce is not bouncable, mail intentionally dropped

mai 5 10:29:54 MailGw amavis[22332]: (i458Trqb012919) BANNED name/type (.exe),
<> -> <XXXX@XXXX.XXXXXXXXXX.XX>, quarantine
virus-20040505-102954-i458Trqb012919, Message-ID:
<20040505082953.5687E182367@XXXXXXXXX.XXXXX.XXX>, Hits: -
```

7. Documents de référence

`amavisd-new`

`amavisd-new`¹⁵ : site principal de l'outil de filtrage de contenu de courrier électronique. L'interface `amavisd-new` se place entre les gestionnaires de mise en file d'attente de réception et d'émission de l'agent de transfert e courrier électronique (MTA).

Debian Anti-Spam Anti-Virus Gateway Email Server using Postfix, Amavisd-new, SpamAssassin, Razor, DCC, Pyzor and ClamAV HOWTO

Support de référence en anglais : [Debian Anti-Spam Anti-Virus Gateway Email Server](#)¹⁶.

Amavis

Page principale du projet d'origine : [amavis](#)¹⁷. Cet outil n'est plus supporté|maintenu.

¹⁵ <http://www.ijs.si/software/amavisd/>

¹⁶ <http://www200.pair.com/mecham/spam/>

¹⁷ <http://www.amavis.org/>

fetchmail

*The fetchmail Home Page*¹⁸ : fetchmail est un utilitaire de récupération et de retransmission de courrier électronique prévu pour fonctionner avec des connexions à la demande de type PPP.

Debian GNU/Linux

*Debian GNU/Linux*¹⁹ : page principale de la distribution.

Charte Debian

*La Charte Debian*²⁰ décrit l'organisation des ressources du système d'exploitation.

¹⁸ <http://www.catb.org/~esr/fetchmail/>

¹⁹ <http://www.debian.org/>

²⁰ <http://www.debian.org/doc/devel-manuals#policy>