24 mai 2008

SMTP reject 554 : Eliminez le spam à la racine

Catégorie : Administration système Tags : GLMF

~~Retrouvez cet article dans : Linux Magazine 83~~

Les initiatives visant à réduire le volume de pourriels ne manquent pas. Certains gouvernements ont tenté de légiférer sans beaucoup de succès. On peut citer le CAN-SPAM Act américain, qui génère à l'occasion des procès médiatisés, mais ne fait finalement que déplacer le problème. De telles approches législatives sont louables, mais sont loin de résoudre le cœur du problème.
À défaut de mieux pour l'instant, il reste des solutions techniques pour tenter de réduire cette nuisance. Nous allons nous intéresser en particulier à leur mise en œuvre sur le même serveur communautaire qui avait servi de base à l'article « Allez au-delà de LAMP », du même auteur, publié dans Linux Magazine 72.

Jusqu'à il y a de cela quelques années, compter principalement sur un filtrage de courriers électroniques mis en œuvre sur le logiciel client était parfaitement acceptable. Les premières méthodes étaient majoritairement heuristiques, fondées sur la quantité de HTML dans le texte, les balises utilisées, les en-têtes, les encodages MIME...

Les filtres bayésiens ont alors fait leur apparition pour évaluer le caractère pourriel d'un mél. Dis-moi quels mots tu utilises et selon quelle loi statistique, et je te dirai si tu es un pourriel. Des contre-attaques sont rapidement apparues. On peut citer les méls dont le contenu « pourri » est inclus dans des images représentant du texte, ou encore ceux dont les mots suspects sont noyés autour de mots anodins rendus invisibles par des balises HTML judicieusement choisies.

Dans l'arsenal contre les pourriels, ont aussi été développées des approches communautaires. Le principe est de partager la connaissance du caractère pourriel d'un mél de manière largement automatisée.

Les pourriels sont référencés par des méthodes plus ou moins savantes d'empreintes numériques, et des bases de données de ces empreintes sont mises à jour par la communauté. Les logiciels anti-pourriel consultent alors ces bases avant de prendre leurs décisions.

Dans toutes ces solutions mises en œuvre du côté client, l'intervention de l'utilisateur est requise à un moment ou à un autre. Ce qui est problématique à plus d'un titre, à commencer par le temps que l'utilisateur perd à entraîner son filtre bayésien ou ajuster les paramètres de ses filtres heuristiques. Dans ce modèle de protection, l'utilisateur est confronté de manière quotidienne à des pourriels. On doit alors s'attendre à ce que régulièrement un utilisateur réponde par inadvertance à un pourriel, ce qui est particulièrement hasardeux. D'abord, parce que l'envoyeur indélicat peut facilement avoir construit son adresse de toute pièce – le pourriel ne fait plus une victime, mais deux : le pseudo-envoyeur et le destinataire. Ou bien, dans le cas contraire, la victime du pourriel indique par sa réponse qu'il y a bien une personne derrière cette adresse mél. Et c'est bien cette information qui intéresse les diffuseurs de pourriels.

Il est bien devenu temps d'alléger la charge qui incombe à l'utilisateur final, en traitant le problème de manière plus agressive en amont.

La lutte contre les pourriels est passée à la vitesse supérieure quand les serveurs de courrier ont commencé à s'y mêler. Signe du changement progressif des mentalités, avec sa version 8.9.0 en 1998 le logiciel serveur de méls Sendmail cessait par défaut de faire suivre les messages en relais ouvert. À partir de ce moment-là, si Sendmail relaie un mél, c'est qu'il est explicitement responsable.

- ou bien du réseau IP de l'envoyeur ;
- ou bien du domaine du destinataire final.

Cela a rendu la diffusion des pourriels anonymes un peu plus compliquée, au moins pour un temps.
D'autres systèmes anti-pourriels qu'on avait commencé à voir mis en œuvre au niveau des clients se sont fait adapter d'une manière ou d'une autre aux serveurs chargés du relais mél.

Dans le monde de Sendmail, la meilleure manière de rajouter des fonctionnalités se fait via des milters. Ces ajouts peuvent modifier le comportement de Sendmail alors même que le mél est transmis au serveur, au sein de la connexion SMTP (protocole de transfert de méls). Une mimique de la conversation pendant laquelle ~~serveur1~~ envoie un pourriel à ~~serveur2~~ serait :

- ~~[serveur1]~~ Bonjour, moi c'est ~~serveur1~~ (~~HELO serveur1.example1.com~~)
- ~~[serveur2]~~ Bonjour ~~serveur1~~
- ~~[serveur1]~~ J'ai un mél en provenance de ~~utilisateur1@example1.com~~ (~~MAIL FROM: utilisateur1@example1.com~~)
- ~~[serveur2]~~ ~~utilisateur1@example1.com~~... L'adresse a l'air valide, continue
- ~~[serveur1]~~ C'est pour ~~utilisateur2@example2.com~~ (~~RCPT TO: utilisateur2@example2.com~~)
- ~~[serveur2]~~ Oui, je le connais, lui
- ~~[serveur1]~~ Je vais t'envoyer les données du mél (~~DATA~~)
- ~~[serveur2]~~ Vas-y, et termine par un point tout seul sur une ligne
- ~~[serveur1]~~ Blah, blah, blah...
- ~~[serveur2]~~ Mais, c'est un pourriel ! Fin de la conversation, erreur **SMTP 554** « la transaction a échoué » (~~reject=554 5.7.1 Pourriels dehors~~)

Avec un tel système, le pourriel n'est plus seulement le problème de l'utilisateur. Il attire l'attention de l'administrateur système (de ~~serveur1~~ dans notre cas), qui est notifié qu'un mél venant de son serveur a été refusé. La notification de refus remonte ainsi le chemin de distribution du mél, serveur par serveur. Et ne se fonde pas uniquement sur les champs ~~From~~ ou ~~Reply-To~~, auxquels un client mél se limite.

Parmi les multiples milters de Sendmail disponibles, MIMEDefang est des plus utiles pour la lutte contre les pourriels. Il est écrit en Perl, et bénéficie donc de la souplesse et de la multitude des modules qui vont avec. Pour garantir de bonnes performances, MIMEDefang se compile au démarrage de son démon, reste résident en mémoire, tandis qu'un multiplexeur répartit la charge entre plusieurs fils. On conserve la flexibilité de Perl, mais sans le coût d'une compilation à chaque exécution.

MIMEDefang se configure majoritairement à partir d'un fichier Perl, ~~/etc/mail/mimedefang-filter~~. C'est à partir de ce fichier qu'une bonne partie d'une politique de sécurité peut être mise en œuvre : mise en quarantaine de fichiers attachés quand ils ont des extensions dévastatrices pour les systèmes Windows (y compris à l'intérieur des fichiers ZIP), interdiction des fichiers ZIP chiffrés, rejet des connexions quand le serveur distant ne se présente pas avec sa véritable adresse IP, réactions aux réponses des modules anti-pourriels et antivirus...

MIMEDefang peut aussi être étendu avec toute une batterie de modules externes. On citera HTMLCleaner, pour nettoyer les messages HTML d'incongruités comme les web bugs qui fournissent des renseignements précieux à l'envoyeur de pourriels – sans même que la victime n'ait à répondre au message. SpamAssassin le bien nommé est un logiciel de détection et de gestion des pourriels qui peut être utilisé en module de MIMEDefang. Il combine à lui seul des approches heuristiques et bayésiennes, et peut interroger d'autres modules qui utilisent une

approche communautaire (~~pyzor~~, ~~razor~~, ~~dcc~~...). Il suffit de regarder la liste des paquetages suggérés par le paquetage Debian ~~spamassassin~~ pour avoir une idée du niveau d'intégration entre tous ces modules.

L'activation des modules SpamAssassin depuis MIMEDefang se fait dans ~~/etc/mail/sa-mimedefang.cf~~. On rajoutera par exemple :

```
# Utilisation du module razor2 (type communautaire)
use_razor2 1
# Utilisation du module dcc (type communautaire)
use_dcc 1
# Utilisation du filtre bayésien
use_bayes 1
bayes_auto_learn 1
bayes_journal_max_size 5120000
```

Pour conserver dans les journaux d'activité la caractérisation des méls par SpamAssassin, il convient de modifier son traitement dans ~~/etc/mail/mimedefang-filter~~. La boucle

```
if ($Features{«SpamAssassin»}) {...
```

peut ainsi devenir :

```
   if ($Features{«SpamAssassin»}) {
        if (-s «./INPUTMSG» < 100*1024) {
            # On ne s'intéresse qu'aux messages de plus de 100 ko.
                # Les méls plus gros sont rarement des pourriels, et ralentissement
            # énormément SpamAssassin
            my($hits, $req, $names, $report) = spam_assassin_check();
            my $names2 = $names ; $names2 =~ tr/,/:/;

            # Si le score SpamAssassin est supérieur à 7,
                # le mél est mis en quarantaine (conservé sur le disque),
                # ses caractéristiques mises en journal
                # et le serveur envoyant le mél reçoit une notification de rejet
                # [léger mensonge, donc]
            if ($hits >= 7) {
              action_quarantine_entire_message();
              md_graphdefang_log('spam', $hits, $RelayAddr.»:».$names2);
              return action_bounce(«Pourriels dehors»);
            }

            # Si le score SpamAssassin est supérieur à 5
                # (valeur stockée dans $req),
                # les caractérisations du mél sont mises en journal,
                # le rapport SpamAssassin est incorporé au mél,
                # le sujet du mél est modifié pour commencer par [POURRIEL]
                # [et le mél poursuit son chemin normal]
            if ($hits >= $req) {
                md_graphdefang_log('pourriel', $hits, $RelayAddr.»:».$names2);
                action_add_part($entity, «text/plain», «-suggest»,
                                «$report\n»,
                                «SpamAssassinReport.txt», «inline»);
                action_change_header(«Subject», «[POURRIEL] $Subject»);
            }
        }
    }
```

Une erreur de syntaxe dans ce fichier de configuration aurait des conséquences bien désagréables. Il convient donc de le valider avant de redémarrer MIMEDefang :

```
sudo mimedefang.pl -f /etc/mail/mimedefang-filter -test
```

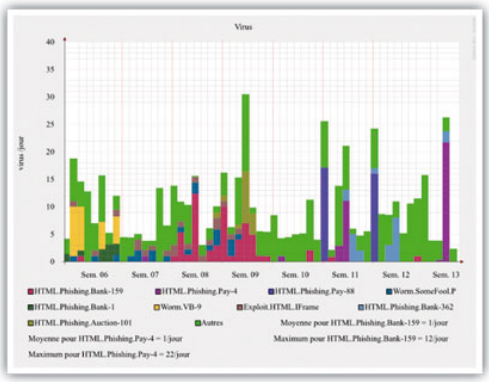
puis alors seulement

```
sudo /etc/init.d/mimedefang restart
```

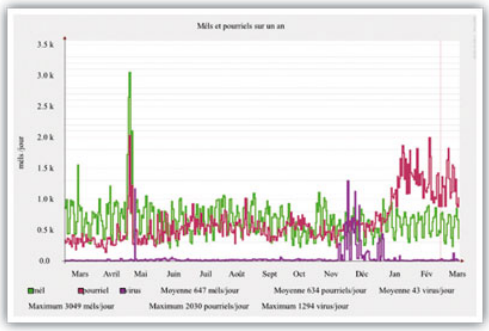
Après quelque temps, on peut s'intéresser aux caractéristiques du pire pourriel jamais reçu dans le temps de vie des journaux encore en ligne (celui avec le plus grand score) :

```
$ sudo gzip -fdc /var/log/mail.log* | \
> grep «,pourriel,» | \
> sort -k4 -t ,," -n | \
> tail -1
Mar 22 11:33:28 hôte mimedefang.pl[26796]: MDLOG,k2MBXIYY027790,spam,61.553,a.b.c.d:BAYES_99:DCC_CHECK:DIGEST_MULTIPLE:
DNS_FROM_RFC_WHOIS:FORGED_MUA_OUTLOOK:FORGED_OUTLOOK_HTML:FORGED_YAHOO_RCVD:
FRONTPAGE:HEAD_ILLEGAL_CHARS:HTML_80_90:HTML_CHARSET_FARAWAY:HTML_FONT_BIG:
HTML_IMAGE_ONLY_12:HTML_IMAGE_RATIO_02:HTML_MESSAGE:MIME_BOUND_DD_DIGITS:
MIME_HTML_ONLY:MIME_HTML_ONLY_MULTI:MIME_QP_LONG_LINE:MISSING_MIMEOLE:
MPART_ALT_DIFF:MSGID_SPAM_CAPS:PYZOR_CHECK:RAZOR2_CF_RANGE_51_100:RAZOR2_CHECK:
RCVD_BY_IP:RCVD_DOUBLE_IP_SPAM:RCVD_HELO_IP_MISMATCH:RCVD_ILLEGAL_IP:
RCVD_IN_BL_SPAMCOP_NET:RCVD_IN_DSBL:RCVD_IN_NJABL_PROXY:RCVD_IN_XBL:
RCVD_NUMERIC_HELO:SUBJ_ILLEGAL_CHARS:URIBL_OB_SURBL:URIBL_WS_SURBL,
<npcukrymxxgicd@yahoo.com>,[…]
```

Le filtre bayésien considère ce mél comme un pourriel avec une certitude de 99%, le système communautaire DCC a reconnu son empreinte dans sa base de données de pourriels, au moins un autre système communautaire est du même avis, le serveur de l'envoyeur a été catalogué comme mauvais citoyen du net par www.rfc-ignorant.com, le pourriel prétend venir d'Outlook mais visiblement ce n'est pas le cas...
On peut aussi utiliser ces journaux d'activité pour garder un œil sur les différentes composantes de l'évaluation des méls. Quelles composantes se retrouvent le plus souvent, lesquelles ont cessé de fonctionner pour quelque raison que ce soit... ?
Enfin, ClamAV, l'antivirus libre, est aussi accessible sous la forme d'un module MIMEDefang.
Il est utile, un petit peu pour le nettoyage des virus, mais surtout pour le nettoyage des méls d'hameçonnage (phishing). En illustration, les statistiques de ClamAV pour le serveur à la base de cet article :



Parmi les 9 virus les plus filtrés par cette installation de ClamAV, 6 sont des méls d'hameçonnage.
Un tel système de filtrage est en tout état de cause invasif, et inclut des analyses statistiques poussées de chaque mél qui passe par le serveur. Ledit serveur est situé aux États-Unis, ce qui évite d'avoir à considérer dans cet article le conendrum des lois européennes et françaises sur la protection de la vie privée. Google et son service de méls Gmail en savent quelque chose : analyser d'un peu trop près le contenu des méls laisse une odeur persistante de souffre, sur le vieux continent...
L'approche décrite jusqu'à présent a donné toutes satisfactions jusqu'au début 2006. À ce moment-là, le nombre de pourriels a commencé à excéder de manière constante et significative le nombre de méls légitimes, comme illustré sur ce graphique :



Les comptes système (~~root~~, ~~postmaster~~ et autres) ont alors commencé à recevoir des pourriels. Pas nécessairement en gros volume, mais suffisamment pour annoncer une tendance inquiétante et motiver un renforcement des moyens de prévention. Retour à la planche à dessin.

L'amplitude du phénomène de pourriel peut s'expliquer par le fait que les méls sont excessivement bon marché à envoyer. Vous pouvez envoyer un million de méls, cela ne vous coûtera pas grand-chose.

Même si un dix millième des destinataires reçoivent un exemplaire et mordent à l'hameçon, vous avez 100 victimes potentielles, pour bien moins cher qu'une seconde de publicité pendant le super bowl (seconde qui se facture tout de même à 80 000 $).

Certains fournisseurs d'accès se sont généreusement proposés de facturer les méls. D'autres prétendent facturer l'option de se faire inscrire sur leur liste blanche. L'intérêt pour une entreprise qui bénéficie de ce service est que ses méls ne seront pas filtrés et accéderont directement aux abonnés du fournisseur d'accès en question.

Une autre approche consiste à améliorer l'authentification des envoyeurs de méls. L'idée sous-jacente a été exprimée sur un sujet connexe par John Madelin (employé de RSA), quand il identifie l'identité comme principe fondateur d'organisation des réseaux de demain (http://www.rsasecurity.com/blog/entry.asp?id=1070).

C'est précisément ce que vise SPF pour l'infrastructure mél (Sender Policy Framework, http://www.openspf.org), en publiant via le service DNS la liste des serveurs qui sont autorisés à envoyer des méls en provenance d'un certain nom de domaine. Il existe un outil qui répond à ces deux critères – rendre l'envoi de pourriels plus cher, tout en favorisant le déploiement de SPF.

La méthode greylist (liste grise) a été développée à partir de l'observation que les logiciels spécialisés d'envoi de pourriels ne prennent pas la peine de mettre les messages en queue.

Si le serveur auquel ils prétendent envoyer leur pourriel ne l'accepte pas immédiatement (i. e. renvoie un message d'indisponibilité temporaire), ils renoncent.

Mettre des messages en queue consomme en effet des ressources (en termes d'espace disque, RAM, CPU et bande passante), rallonge la probabilité de se faire prendre la main dans le sac et ralentit l'envoi massif de pourriels.
Autrement dit, cela coûte trop cher dans l'optique d'inonder l'Internet de pourriels à moindre coût

La conversation entre serveur1, qui prétend envoyer pour la première fois un mél à serveur2, qui met en œuvre greylist, ressemble à ceci :

- ~~[serveur1]~~ Bonjour, moi c'est ~~serveur1~~
- ~~[serveur2]~~ Bonjour ~~serveur1~~
- ~~[serveur1]~~ J'ai un mél en provenance de ~~utilisateur1@example1.com~~
- ~~[serveur2]~~ ~~utilisateur1@example1.com~~... L'adresse a l'air valide, continue
- ~~[serveur1]~~ C'est pour ~~utilisateur2@example2.com~~
- ~~[serveur2]~~ Repasse dans 30 minutes, erreur SMTP 451, action requise abandonnée : erreur locale

Si serveur1 essaie une demi-heure plus tard (pour un mél en provenance de ~~utilisateur1@example1.com~~ et à destination de ~~utilisateur2@example2.com~~), le mél sera accepté. Ce sera aussi le cas pour tous les méls qu'il enverra pendant 24 heures – ~~serveur1~~ restera dans la liste blanche de ~~serveur2~~ pendant 24 heures.

Il faut noter que le renseignement « ce mél sera accepté dans 30 minutes » n'est pas standard, il faut donc s'attendre à ce que ~~serveur1~~ fasse une tentative potentiellement avant le délai imparti.

Le standard lui dicte de refaire une tentative, mais ne précise pas quand. En temps normal, quand un serveur a un problème temporaire, il ne sait pas vraiment quand il sera de nouveau fonctionnel.

La mise en œuvre de greylist pour Sendmail, milter-greylist (http://hcpnet.free.fr/milter-greylist/), a la bonne idée de ne pas appliquer de délai aux mails qui suivent le cadre SPF (i. e. qui ont été envoyés par un serveur de mél « habilité » à le faire dans le cadre de SPF).

Pour savoir si un domaine en particulier a déployé SPF, une requête DNS suffit :

```
$ host -t txt gmail.com
gmail.com text «v=spf1 ip4:216.239.56.0/23 ip4:64.233.160.0/19 ip4:66.249.80.0/20 ip4:72.14.192.0/18 ?all»
```

Il est entendu qu'un utilisateur qui enverrait des méls en provenance de son adresse en gmail.com, mais à partir du serveur SMTP de son fournisseur d'accès à Internet, ne bénéficiera pas du blanchiment SPF automatique.

Tourné autrement : si un mél est envoyé depuis un serveur « homologué » pour le domaine en question, il n'est pas retardé, sinon il doit patienter un peu.

milter-greylist n'existe pas en paquet Debian, il faut le compiler depuis les sources. La compilation marche en utilisant les paquetages flex et bison. libspf0 et libspf-dev sont chaudement recommandés, pour bénéficier du support de SPF.

Dans le fichier de démarrage /etc/init.d/milter-greylist copié depuis à la main depuis le répertoire source, il est souhaitable de remplacer USER=root par USER=smmsp, en application du principe du moindre privilège. update-rc.d permet de créer les liens qui vont bien dans les /etc/rc?.d. Les répertoires /var/run/milter-greylist et /var/milter-greylist doivent être créés, et appartenir à l'utilisateur smmsp. Il faut aussi noter que la base de données des serveurs en attente, /var/milter-greylist/greylist.db, n'apparaît pas dès le lancement du démon, mais quelques temps plus tard. La configuration du fichier /etc/mail/sendmail.mc est documentée. Il suffit d'y inclure :

```
INPUT_MAIL_FILTER(`greylist', `S=local:/var/run/milter-greylist/greylist.sock')
define(`confMILTER_MACROS_CONNECT', `j, {if_addr}')
define(`confMILTER_MACROS_HELO', `{verify}, {cert_subject}')
define(`confMILTER_MACROS_ENVFROM', `i, {auth_authen}')
INPUT_MAIL_FILTER(`mimedefang', `S=unix:/var/spool/MIMEDefang/mimedefang.sock, F=T, T=S:1m;R:1m')dnl
```

pour que milter-greylist et MIMEDefang cohabitent intelligemment.

Le fichier de configuration de milter-greylist, /etc/mail/greylist.conf, ne nécessite pas beaucoup de changements. Le « subnetmatch /24 » permet de considérer toutes les machines venant d'un même réseau /24 (notation CIDR pour 255.255.255.0) comme un seul serveur de mél. Si une de ces machines frappe à la porte une première fois pour un certain mél (ou en tous cas un certain couple (envoyeur, destinataire)), n'importe quelle autre machine de ce réseau envoyant le même mél après les 30 minutes réglementaires le verra accepté et fera entrer les serveurs dans la liste blanche pour la durée standard de 24 heures.

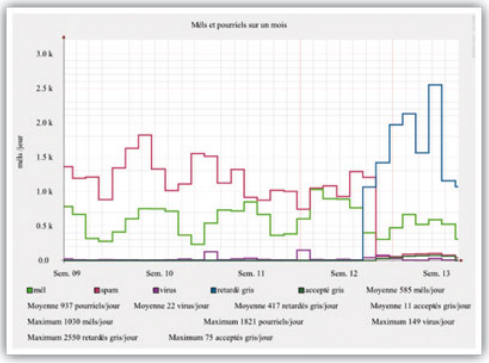
La liste d'accès acl whitelist addr a.b.c.d/nn devrait contenir les partenaires récurrents, respectueux de la netiquette. Ils sont ainsi inscrits de manière permanente dans la liste blanche. Cette liste statique peut être mise à jour à partir des serveurs de mél qui entrent régulièrement dans la liste blanche. Pour obtenir cette liste, assortie du nombre de fois que chaque serveur est entré dans la liste blanche :

```
$ sudo gzip -fdc /var/log/mail.log* | \
> grep "autowhitelisted for" | \
> cut -f8 -d" " | \
> sort | uniq -c | sort -nr | head -10
nn a.b.c.d
[...]
```

Il est aussi intéressant d'avoir une idée de la répartition des tentatives infructueuses:

```
$ sudo gzip -fdc /var/log/mail.log* | \
> awk ' (/Greylisting in action, please come back in /) { delayed+=1 }
> (/autowhitelisted/) { autowhitelisted+=1 }
> (/Greylisting in action, please come back in / &&
> !/Greylisting in action, please come back in 00:30:00/) { redelayed+=1 }
> END { print «Premiere tentative : \t»delayed»\nTentatives suivantes : \t»redelayed»\nFinalement accepte : \t»autowhitelisted }
Premiere tentative :    11786
Tentatives suivantes :  1294
Finalement accepte :    1269
```

Sur les 11 786 tentatives initiales de réception de mél depuis l'installation de milter-greylist la semaine dernière, le serveur n'a revu que 1 294 tentatives ultérieures – celles-ci ont été repoussées parce que les 30 minutes n'étaient pas écoulées. 1 269 tentatives ont eu l'honneur et l'avantage de passer à l'étape suivante du filtrage, MIMEDefang.
Pour ce qui est d'une vision plus globale de l'impact de milter-greylist sur les pourriels reçus par le serveur :



Les « retardés gris » comptent le nombre de fois que le serveur a demandé à l'envoyeur de retenter. Étant entendu que l'envoyeur peut retenter infructueusement dans l'intervalle de 30 minutes imparti. Les « méls » sont les méls légitimes qui ne sont pas passés à travers milter-greylist, parce que les serveurs correspondants sont sur la liste blanche statique.

Milter-greylist a été déployé en fin de semaine 12, s'il était nécessaire de le préciser. Enfin, il convient de noter que la configuration se complique légèrement si plusieurs échangeurs de méls (MX) servent le domaine à protéger. Les milter-greylists se communiquent alors, via une connexion TCP, le contenu de leurs listes blanches et tables d'états respectifs, dans le but de maintenir un front cohérent. Ajuster le pare-feu pour n'autoriser ce trafic qu'en provenance et à destination des MX paraît un minimum, si ce n'est utiliser Stunnel pour chiffrer les échanges dans un tunnel SSL.

Les Logiciels libres offrent donc une panoplie intéressante d'outils pour lutter contre les pourriels, directement au niveau des serveurs de méls. Les distributions Linux modernes fournissent la majeure partie de ces outils sous forme de paquetages, ce qui réduit grandement les efforts nécessaires pour les déployer. Peu ou pas de maintenance est nécessaire, cependant un bon système de surveillance est souhaitable : ces filtres, aussi évolués soient-ils, agissent au cœur même d'un service de communication critique.

Retrouvez cet article dans : Linux Magazine 83

Posté par admin-web (fabrice) | Signature : Guillaume Tamboise | Article paru dans LINUX

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

Il y a actuellement

- 497 articles/billets en ligne.

Recherche

Catégories

Archives

- février 2008
- janvier 2008
- décembre 2007
- novembre 2007
- février 2007

GNU/Linux Magazine

- GNU/Linux Magazine 105 - mai 2008 - Chez votre marchand de journaux !
- Edito : GNU/Linux Magazine 105
- GNU/Linux Magazine Hors-série 36 - mai/juin 2008 - Chez votre marchand de journaux !
- Edito : GNU/Linux Magazine Hors-série 36
- GNU/Linux Magazine 104 - Avril 2008 - Chez votre marchand de journaux !

GNU/Linux Pratique

- Linux Pratique Hors-Série N°15 - Juin / Juillet 2008 - chez votre marchand de journaux.

MISC Magazine

- Misc 37 : Déni de service - Mai/Juin 2008 - Chez votre marchand de journaux
- Edito : Misc 37
- Misc 36 : Lutte informatique offensive, les attaques ciblées - Mars/Avril 2008 - Chez votre marchand de journaux
- Edito : Misc 36
- MISC N°35 : Autopsie & Forensic comment réagir après un incident ?