Support de cours

Durcissement des MTA
Sendmail et *Postfix*

Introduction

Les systèmes de messagerie représentent l'une des applications les plus difficiles à sécuriser.

Non pas que les MTA soient peu sécurisés mais parce que le protocole SMTP souffre de lacunes congénitales :

- il a été créé lorsque l'Internet était encore un espace paisible ;
- il ne propose aucun mécanisme qui permette de vérifier l'origine d'un message (la forge est facile) ;
- il ne permet pas de garantir qu'un message arrivera à destination, ni même, si un message est perdu, que l'expéditeur sera averti ;
- etc.

Par ailleurs, la remise d'un message ou l'utilisation de services supplémentaires (filtres, robots, listes de diffusion, etc.) ont banalisés l'exécution de programmes tiers par le MTA. Par exemple : procmail, vacation, majordomo, sympa, etc. Le niveau de sécurité offert par un ensemble de composants étant celui qui offre le niveau de sécurité le plus faible...

Pré-requis

Cette formation présente des concepts de sécurité associés au protocole SMTP et aux MTA Sendmail et Postfix.

En particulier, on attend des stagiaires qu'ils :

- connaissent les principes de base du protocole SMTP ;
- sachent installer et configurer Sendmail avec des macros *m4*.

Par ailleurs, il s'agit d'une formation interactive, il est donc tout à fait indiqué d'interrompre le formateur pour lui poser des questions, lui faire préciser certains points, demander l'étude d'un cas particulier, etc.

Architecture relais/serveur

Comme on le verra par la suite, il est difficile de supprimer tout risque d'attaque par dénis de service quelque soit le MTA.

C'est pourquoi, un relais situé entre le serveur final et le réseau publique (Internet) est souvent utilisé. Ce relais est traditionnellement installé sur une DMZ. Il s'agit ni plus ni moins d'un *proxy* applicatif tel qu'il en existe pour le protocole HTTP.

Avantages :

- le serveur de messagerie interne n'est pas joignable depuis le réseau publique ;
- la messagerie interne peut continuer à fonctionner même si le relais est « cassé » ;
- possibilité de configuration différente pour les messages internes et externes.

Inconvénients :

- sensibilité forte aux problème de la « patate chaude ».

Conclusion : Toujours d'actualité mais avec un mécanisme qui permette de vérifier la validité des adresses des destinataires sur le relais lui-même.

Sécuriser un serveur SMTP ?

Qu'entend t'on par « sécuriser » un serveur SMTP ?

Nous allons traiter ce sujet autour de 6 axes :

- le MTA ne devrait pas pouvoir être utiliser par un utilisateur local ou distant pour gagner des privilèges sur le système ;
- les attaques par dénis de service devraient être difficiles ;
- des messages ne devraient pas être perdus (fiabilité) ;
- le service ne devrait pas pouvoir être usurpé (relais non autorisé) ;
- il ne devrait pas être possible d'obtenir des données autre que celles strictement nécessaires au transfert des messages ;
- la forge devrait être difficile ou, tout au moins, décelable.

Nous verrons que les MTA n'apporteront que peu à pas de solution à certains de ces problèmes. Il sera alors nécessaire de se rabattre sur d'autres logiciels (au niveau du MUA, par exemple) si cela est possible.

Sendmail

Ce support de cours est basé sur une version 8.12 de Sendmail qui est la version stable à ce jour. Il utilise les fichiers de configuration générés par le langage de macros ***m4***. Le fichier modifié sera donc toujours `sendmail.mc`.

Voici un exemple typique de ce fichier :

```
define(`_USE_ETC_MAIL_')dnl
include(`/usr/share/sendmail/cf/m4/cf.m4')dnl
OSTYPE(`debian')dnl
DOMAIN(`debian-mta')dnl
```

Partie spécifique à chaque distribution

```
LOCAL_CONFIG
Cwgaia.anet.fr
FEATURE(`use_cw_file')dnl
FEATURE(`use_ct_file')dnl
FEATURE(`mailertable')dnl
define(`confME_TRUE', `True')dnl
```

Définition des paramètres de la configuration

```
MAILER_DEFINITIONS
MAILER(local)dnl
MAILER(smtp)dnl
```

Déclaration des *mailers*

Nous partirons de cet exemple basique que nous allons enrichir au fur et à mesure.

Sendmail – privilèges

Sendmail s'exécute en tant que root car il a besoin de certains privilèges :

- accès à certain fichiers
(fichiers `:include:` et `.forward`, bases de données d'authentification, etc.) ;
- ouverture des ports TCP privilégiés 25 et 587 ;
- nécessité de prendre l'identité d'autres utilisateurs
(pour la distribution des messages ou l'exécution de programmes).

Cependant, depuis la version 8.12, Sendmail n'est plus SUID root comme il l'était auparavant afin de prendre en charge les messages envoyés localement. Désormais deux modes sont utilisés bien qu'un seul binaire existe, chaque mode utilise un fichier de configuration distinct :

- Mail Transfert Agent (MTA) `sendmail.cf` ;
- Mail Submission Program (MSP) `submit.cf`.

Le second mode utilise l'identité d'un utilisateur spécifique : `smmsp:smmsp`.
Le fichier binaire de Sendmail est SGID `smmsp`.

Sendmail – privilèges

Le MTA est exécuté en tant que démon au démarrage du système. C'est lui qui écoute sur les ports TCP réservés à SMTP (25 et parfois 587).

Le MSP est SGID smmsp afin de pouvoir insérer les messages envoyés en local dans une file distincte de celle du MTA (`/var/spool/client/clientmqueue` ou `/var/spool/mqueue-client`).

Cette file doit être vidée régulièrement :

- soit avec un démon `sendmail -L sm-msp-queue -Ac -q15m`
- soit avec via cron `sendmail -L sm-msp-queue -Ac -q`

L'utilisateur utilisé par le MSP est smmsp :

```
gaia:~# ps aux |grep sendmail |grep -v grep
root  30473  0.0  0.5  5004 2060 ?  S  17:50  0:00 sendmail: MTA: accepting connections
smmsp 30488  0.0  0.4  4916 1908 ?  S  17:50  0:00 sendmail: MSP: Queue runner@00:10:00 ...
```

La configuration du mode MSP de Sendmail est générée par m4 lorsque la ligne suivante apparaît dans le fichier de macros (dans ce cas, /etc/mail/submit.mc) :

```
FEATURE( `msp', `[127.0.0.1]', `MSA' )dnl
```

Sendmail – privilèges

Les utilisateurs spécifiques à Sendmail :

- `DefaultUser` : identité prise par Sendmail pour exécuter certains programmes
  valeur par défaut : `mailnull` ou `daemon` ou `[1:1]`

  ```
  define confDEF_USER_ID(`daemon:daemon')dnl
  ```

- `TrustedUser` : propriétaire des fichiers *maps*, des fichiers d'alias et autres
  valeur par défaut : `root`

  ```
  define confTRUSTED_USER(`mailadmin:staff')dnl
  ```

- `RunAsUser` : utilisateur utilisé par Sendmail dès que possible
  (c'est-à-dire, juste après avoir lu le fichier de configuration ou après avoir accepté une connexion en mode démon)
  valeur par défaut : `root`

  ```
  define confRUN_AS_USER(`mail:mail')dnl
  ```

Sendmail - smrsh

L'utilitaire smrsh (*Sendmail Restricted Shell*) peut être utilisé pour limiter la liste des programmes autorisés à l'exécution pour Sendmail. Les programmes autorisés doivent être la cible d'un lien symbolique dans le répertoire /etc/smrsh.

Il effectue également des vérifications sur le contenu des lignes de commande :

- caractères « `<>;$() » et le retour chariot interdits ;
- limitation des commandes intégrées au *Shell* ;
- transformation des chemins d'accès aux programmes (/usr/bin/vacation devient /etc/smrsh/vaction).

La mise en oeuvre est simple, utiliser la directive suivante dans sendmail.mc :

```
FEATURE(`smrsh')dnl
```

Puis faire les liens dans le répertoire /etc/smrsh vers les programmes autorisés.

Ne pas autoriser procmail (permet l'exécution de commandes arbitraires). D'une manière générale, restez très suspicieux par rapport à la liste des programmes autorisés.

Sendmail – *mailer* `local`

Le *mailer* `local` est celui qui permet de délivrer les messages aux utilisateurs locaux ainsi qu'à des programmes.

Si le contexte le permet il est possible de le désactiver :

- si le MTA est un relais de messagerie ;
- si l'exécution d'aucun programme par le MTA n'est nécessaire (donc aucun services tels que `procmail`, `vacation`, gestionnaire de listes de diffusion, etc.) ;
- si les utilisateurs locaux n'ont pas besoin de recevoir de messages (prévoir de rediriger les adresses « spéciales » : `root`, `postmaster`, `MAILER-DAEMON`, etc.).

Cela se fait dans le fichier `sendmail.mc` en déclarant un *mailer* `local` illusoire (attention, technique expérimentale) :

```
MAILER_DEFINITIONS
dnl MAILER(`local')dnl
MAILER(`smtp')dnl

LOCAL_CONFIG
Mlocal, P=/bin/false, A=dummy, F=l
define(`_MAILER_local_', `dummy')dnl
```

Sendmail – exécuter dans un *chroot*

L'option `SafeFileEnvironment` permet d'obliger Sendmail à exécuter un *chroot* avant toute écriture dans un fichier. La directive de configuration *m4* associée est :

```
define(`confSAFE_FILE_ENV', `/safe')dnl
```

Cela signifie que tous les fichiers dans lesquels Sendmail a besoin d'écrire doivent se trouver dans cette arborescence (boîtes aux lettres, fichiers trouvés dans les tables d'alias, etc.). Cela ne concerne pas les fichiers écrits par les *mailers*.
Cette option désactive également l'écriture dans les fichier spéciaux et les liens symboliques.

Depuis Sendmail 8.10 il est possible de demander l'exécution de chaque *mailer* dans un environnement *chroot*. Il faut pour cela utiliser le paramètre `/=` de la définition du *mailer* concerné.

Enfin, il est aussi possible d'exécuter Sendmail lui-même dans un environnement *chroot*. Cela requiert la construction complète de la structure des répertoires et fichiers nécessaires à Sendmail (fichiers spéciaux, librairies, fichiers de configuration, etc.).

Sendmail – protection contre les DoS

Il est très difficile de se prémunir de manière efficace contre les attaques par dénis de service. Ce qu'il est possible de faire pour limiter le risque se décline en deux catégories :

- modifier la valeur de certains paramètre pour contrôler et protéger nos ressources (espace disque, connexion entrantes simultanées, nombre de processus, etc.) ;

- exiger le respect de certaines règles de la part des MTA qui nous envoient des messages.

Sendmail – protéger nos ressources

Il est souhaitable de fixer une valeur raisonnable aux paramètres suivants :

- `MaxDaemonChildren` nombre maximal de connexions entrantes
- `MaxMessageSize` taille maximale des messages
- `ConnectionRateThrottle` nombre de connexions entrantes par seconde et par démon à partir duquel Sendmail introduit un délais pour toute nouvelle connexion
- `BadRcptThrottle` nombre de destinataires rejetés lors d'une connexion à partir duquel un délais d'une seconde est inséré après chaque nouveau mauvais destinataire
- `DelayLA` charge moyenne (*load average*) à partir de laquelle un délais d'une seconde est introduit après chaque commande SMTP et pour toute nouvelle connexion
- `MinFreeBlocks` nombre minimal de blocs libres sur le système de fichiers qui contient les files pour continuer à accepter les connexions entrants
- `RefuseLA` charge moyenne à partir de laquelle les nouvelles connexion sont refusées (par défaut 12 fois le nombre de processeurs)

Sendmail – protéger nos ressources

Voici un exemple de ce qui pourraient être configuré dans `sendmail.mc` :

```
dnl  Nombre maximal de connexions, pas trop, ni trop peu
define(`confMAX_DAEMON_CHILDREN', `100')dnl

dnl  Taille maximal de message, entre 5 et 10 Mo, ici 8 Mo
define(`confMAX_MESSAGE_SIZE', `8388608')dnl

dnl  Nombre maximal de connexion par seconde
define(`confCONNECTION_RATE_THROTTLE', `5')dnl

dnl  Nombre maximal de "mauvais" destinataire
define(`confBAD_RCPT_THROTTLE', `5')dnl

dnl  Charge a partir de laquelle on ralentit
define(`confDELAY_LA', `6')dnl

dnl  Place minimale pour ne pas refuser les connexions
define(`confMIN_FREE_BLOCKS', `500')dnl

dnl  Nombre maximal de destinataires par enveloppe
define(`confMAX_RCPTS_PER_MESSAGE', `50')dnl
```

Sendmail – protéger nos ressources

Les valeurs des paramètres donnés comme exemple ne conviendront pas forcément à tous les en environnements. Il est important d'observer le comportement du MTA afin de vérifier s'ils conviennent.

Dans tous les cas il est important de superviser les ressources du système.

Par ailleurs, il devient de plus en plus critique de connaître dès le relais de messagerie la liste des utilisateurs autorisés pour le domaine. Par exemple, *via* la *virtusertable* de Sendmail dont un exemple suit (de plus cela permet d'interdire l'utilisation de listes de diffusion internes) :

```
jo.dalton@luke.net          jo
jack.dalton@luke.net        jack
william.dalton@luke.net     william
averell.dalton@luke.net     averell
postmaster@luke.net         postmaster
MAILER-DAEMON@luke.net      postmaster
@luke.net                   error:5.5.1:550 User unknown
```

Ne pas oublier les adresses `postmaster` et `MAILER-DAEMON` !

Sendmail – se protéger des MTA indélicats

Nous nous évertuons à mettre en oeuvre un MTA qui respecte les règles. Il est bon et sain que les MTA qui entrent en communication avec nous respectent également les règles définies par les standards. Cela pour deux raisons :

1) afin d'améliorer le service de messagerie SMTP d'un point de vue global ;
2) si un MTA ne respecte pas les formats standard, c'est qu'il cherche peut-être à faire passer un message suspect.

Par conséquent, le laxisme qui était de mise depuis la création du service de messagerie sur Internet (« je fais tout ce que je peux pour essayer de transmettre un message ») a tendance à disparaître au profit d'une attitude de plus en plus rigoureuse.

Cependant, nous pourrons accepter ou refuser les messages selon certaines règles plus en moins en fonction de l'environnement. En effet, des critères trop « durs » peuvent nuire à la perception de la fiabilité qu'auront les utilisateurs du service.

Il existe deux catégories de règles :

- celles qui sont liées au caractéristiques du message transmis ;
- celles qui dépendent du serveur qui entre en communication avec nous.

Sendmail – se protéger des MTA indélicats

Refuser certains messages en fonction de leur caractéristiques :

- si vous ne relayez pas les messages UUCP, rejetez-les :

```
FEATURE(`nouucp', `reject')dnl
```

- nombre maximal de destinataire par enveloppe :

```
define(`confMAX_RCPTS_PER_MESSAGE', `50')dnl
```

- utilisation de la base de données d'accès :

```
FEATURE(`access_db')dnl
```

Exemple de contenu du fichier `access` :

```
spammer@aol.com          REJECT
cyberspammer.com         DISCARD
support@microsoft.com    ERROR:550 "Go away, you're probably a virus"
192.168.200              RELAY
```

Sendmail – se protéger des MTA indélicats

Il devient quasi-indispensable de refuser des messages provenant de relais SMTP mal configurés. Ceux-ci sont recensés dans des bases de données en ligne et accessibles *via* le protocole DNS.

Ces bases de données sont appelées RBL (*Real-time Block Lists*) et recensent différents types de problème de configuration. En voici quelques uns :

- relais ouverts ;
- utilisateurs requis non présents (`postmaster`, `abuse`, etc.) ;
- refus des *bounces* (adresse d'expédition étant « <> ») ;
- relais réputés héberger des *spammers* notoires ;
- DUL (*Dialup Users Lists*) ;
- etc.

Il convient de les utiliser à bon escient, certaines étant un peu trop « radicales ». Les relais ouverts sont à proscrire dans tous les cas. Il est aussi généralement admis que les domaines n'acceptant pas les *bounces* ne se comportent pas de manière correcte.

Je recommande de n'utiliser les autres types de RBL qu'avec des systèmes dits de *scoring* afin d'atténuer leur « agressivité ».

Sendmail – RBLs

Il existe deux manières d'interroger une RBL : avec une adresse IP ou en utilisant un nom de domaine.

L'interrogation d'une RBL avec l'adresse IP du relais qui nous contacte est simple. Par exemple, pour utiliser les services de DSBL et d'ORDB :

```
dnl  RBL DSBL (http://dsbl.org)
FEATURE(`dnsbl', `list.dsbl.org', `"550 Email rejected due to sending       \
  server misconfiguration - see http://dsbl.org"')dnl

dnl  RBL ORDB (http://ordb.org)
FEATURE(`dnsbl', `relays.ordb.org', `"550 Email rejected due to sending     \
  server misconfiguration – see http://www.ordb.org/faq/\#why_rejected"')dnl
```

Note : L'utilisation des RBLs se généralise, c'est pourquoi il est important de bien configurer son relais de messagerie afin de ne pas être listé dans l'une d'entre elles. Chaque service propose sur son site Web une procédure qui permet d'enlever une adresse IP de la RBL mais seulement une vérification sera réalisée auparavant.

Sendmail – RBLs

Sendmail ne supporte pas nativement l'interrogation des RBL avec le nom de domaine. Mais il est très d'ajouter ce support. Pour cela, il suffit de télécharger le fichier suivant et de le copier :

- pour RedHat : dans le répertoire `/usr/share/sendmail-cf/feature` ;
- pour Debian : dans le répertoire `/usr/share/sendmail/cf/feature`.

`http://www.megacity.org/software_downloads/rhsbl.m4`

Voila, maintenant il est possible d'utiliser ce type de RBL. L'exemple suivant utilise le service RFC Ignorant pour interdire les messages provenant de domaines qui refusent les *bounces* :

```
FEATURE(rhsbl,`dsn.rfc-ignorant.org',`"550 Mail from domain " $`'&{RHS} "    \
  refused. MX of domain do not accept bounces. This violates RFC             \
  821/2505/2821 - see http://www.rfc-ignorant.org/"')dnl
```

Il est cohérent de refuser les messages qui proviennent de ces domaines car, en cas d'erreur (destinataire inconnu, par exemple), nous n'auront pas moyen de renvoyer un message de notification (*bounce*) à l'expéditeur.

Sendmail – fiabilité

Comme nous l'avons vu précédemment, SMTP signifie *Simple Mail Transfert Protocole*. En particulier, rien ne garantit qu'un message sera bien délivré à tous ses destinataires. Et il est tout à fait possible qu'un message soit perdu sans que personne n'en soit avertit. Personne n'est à l'abri d'un disque dur qui flanche.

Cela dit les MTA modernes font tout pour que cette situation n'existe pas. La règle d'or en la matière étant qu'il vaut mieux refuser un de prendre en charge un message (avec en retour un code d'erreur qui peut être temporaire ou permanent) si nous sommes dans une situation critique (problème de ressources, typiquement).

Deux règles de base pour assurer un service de qualité :

- la redondance de stockage (RAID 1ou 5) ;
- la supervision des ressources.

Les MUA n'offrent pas non plus de technique universellement reconnues afin de palier à ce problème. Il y a bien les DSN (*Delivery Status Notification*) mais, comme nous le verrons plus tard, la tendance est plutôt de les désactiver.

Sendmail – contrôle du relais

Il est très important de contrôler qui a le droit d'utiliser le relais de messagerie. Deux éléments sont utilisés afin de permettre de déterminer si un message peut être le relayé ou non :

- le machine d'où provient le message (son adresse IP ou son nom) ;
- le destinataire du message.

Le premier élément permet d'autoriser le relais des messages qui sont envoyés depuis des machines internes, le second contrôle les messages entrants.

Par défaut, Sendmail interdit tout relais. Pour l'autoriser, il existe plusieurs mécanismes. Le plus simple étant d'utiliser la base de données d'accès vue précédemment.

À noter que les domaines dont la destination est locale n'ont pas besoin d'être listés dans la base de données d'accès (ils le sont déjà soit dans `sendmail.mc`, soit dans `local-host-names`).

Enfin, n'oublions pas de nous prémunir contre les relais ouverts en utilisant une ou plusieurs RBL comme vu précédemment.

Sendmail – authentification SMTP

L'interdiction de relais est saine mais pose un problème pour les nomades. Pour remédier à cela, la solution la plus simple reste l'authentification SMTP. L'idée est d'autoriser le relais à partir du moment où la connexion est authentifiée.

L'authentification SMTP utilise SASL (*Simple Authentification and Security Layer*). Il sera donc nécessaire de mettre en oeuvre une base de données de comptes utilisateurs qui soit supportées par les librairies SASL utilisées par Sendmail (Cyrus SASL). Il peut s'agir de LDAP, Kerberos, etc.

Ici, nous utiliserons tout simplement `sasldb` qui est une base de données simple stockée dans le fichier `/etc/sasldb`. Cela nous permettra de ne pas avoir à créer des utilisateurs systèmes pour l'authentification SMTP.

Il faut donc commencer par modifier le fichier `/usr/lib/sasl/Sendmail.conf` qui doit ne contenir que la ligne suivante :

```
pwcheck_method: sasldb
```

Sendmail – authentification SMTP

Ensuite, créer le fichier `/etc/sasldb` et y insérer le premier utilisateur :

```
# touch /etc/sasldb
# saslpasswd -c snameche
Password:
Again (for verification):
# sasldblistusers
user: snameche realm: lucky.luke.net mech: PLAIN
user: snameche realm: lucky.luke.net mech: DIGEST-MD5
user: snameche realm: lucky.luke.net mech: CRAM-MD5
```

Ajouter les directives suivantes au fichier `sendmail.mc` :

```
define(`confAUTH_MECHANISMS', `LOGIN PLAIN DIGEST-MD5 CRAM-MD5')dnl
TRUST_AUTH_MECH(`LOGIN PLAIN DIGEST-MD5 CRAM-MD5')dnl
```

Remarque : Si possible, ne pas inclure les mécanismes `LOGIN` et `PLAIN` dans la liste car ils laissent passer le mot de passe en claire contrairement aux deux autres.

Sendmail – confidentialité

Le premier point à vérifier pour assurer la confidentialité des données manipulées par Sendmail est, bien évidemment, les droits d'accès sur le système de fichiers.

La commande suivante liste les droits sur les répertoires et fichiers critiques, les droits qu'ils devraient normalement posséder sont indiqués en italique :

```
# ls -ld / /etc /etc/mail /etc/mail/* /var /var/spool /var/spool/*mqueue
drwxr-xr-x  21 root   root    4096 avr 15 11:36 /                          0755 root
drwxr-xr-x  93 root   root    8192 avr 15 19:57 /etc                       0755 root
drwxr-xr-x   3 root   root    4096 avr 15 21:45 /etc/mail                  0755 TrustedUser
-rw-r--r--   1 root   root     331 sep 17  2003 /etc/mail/*                0644 TrustedUser
drwxr-xr-x  28 root   root    4096 avr  9 17:13 /var                       0755 root
drwxr-xr-x  26 root   root    4096 avr 11 20:56 /var/spool                 0755 root
drwxrwx---   2 smmsp  smmsp  98304 avr 15 21:50 /var/spool/clientmqueue    0770 smmsp
drwx------   2 root   mail    4096 avr 15 21:50 /var/spool/mqueue          0700 RunAsUser
```

Sendmail vérifie les droits sur ces fichiers et répertoires, le paramètre de configuration `DontBlameSendmail` (`confDONT_BLAME_SENDMAIL` en *m4*) permet d'ajuster sa paranoïa.

Sendmail – confidentialité

On peut aussi utiliser le paramètre `PrivacyOptions` afin de limiter l'exécution de certaine commandes aux utilisateurs non privilégiés. Typiquement, leur interdire de visualiser ou forcer la file des messages ou bien la vérification des membres des alias.

Dans le fichier `sendmail.mc`, ajouter cette ligne :

```
define(`confPRIVACY_FLAGS',`restrictmailq,restrictqrun,restrictexpand')dnl
```

Remarque : Si le MSP est utilisé, la plupart des informations fournis par les commandes `mailq`, `runq` et `sendmail -bv` sont soit inaccessibles, soit limitées.

Sendmail – confidentialité

L'exemple qui suit montre qu'un MTA peut se montrer fort indiscret :

```
puck:~# telnet rantanplan.luke.net smtp
Trying 192.168.200.52...
Connected to rantanplan.luke.net.
Escape character is '^]'.
220 rantanplan.luke.net ESMTP Sendmail 8.12.8/8.12.8; Thu, 15 Apr 2004 22:52:31 +0200
HELO you
250 rantanplan.luke.net Hello puck.anet.fr [192.168.200.1], pleased to meet you
VRFY root
250 2.1.5 <root@rantanplan.luke.net>
EXPN root
250 2.1.5 <sebastien@nameche.fr>
EXPN daltons@luke.net
250-2.1.5 <averell@daltoncity.com>
250-2.1.5 <william@daltoncity.com>
250-2.1.5 <jack@daltoncity.com>
250 2.1.5 <jo@daltoncity.com>
QUIT
221 2.0.0 rantanplan.luke.net closing connection
Connection closed by foreign host.
```

À partir d'un simple *telnet* il est possible de connaître le nom et la version du MTA et, pire, de connaître les membres d'un alias.

Sendmail – confidentialité

Voici les paramètres qui vont permettre de museler notre Sendmail :

```
define(`confSMTP_LOGIN_MSG',`$j MTA; $b')dnl
define(`confPRIVACY_FLAGS',`novrfy,noexpn,noetrn')dnl
```

La liste des options du second paramètre *m4* est a ajouter à celle qui pourrait déjà exister (cf. `restrictqrun`, etc.). Voici le résultat vu de l'extérieur :

```
puck:~# telnet rantanplan.luke.net smtp
Trying 192.168.200.52...
Connected to rantanplan.luke.net.
Escape character is '^]'.
220 rantanplan.luke.net ESMTP MTA; Thu, 15 Apr 2004 23:07:08 +0200
HELO you
250 rantanplan.luke.net Hello puck.anet.fr [192.168.200.1], pleased to meet you
VRFY root
252 2.5.2 Cannot VRFY user; try RCPT to attempt delivery (or try finger)
EXPN daltons@luke.net
502 5.7.0 Sorry, we do not allow this operation
QUIT
221 2.0.0 rantanplan.luke.net closing connection
Connection closed by foreign host.
```

Sendmail – confidentialité

L'option `noetrn` du paramètre *m4* `confPRIVACY_FLAGS` permet de désactiver l'opération SMTP `ETRN` qui peut constituer une vulnérabilité sévère dans certains cas (lors de l'utilisation d'Amavis ou MailScanner notamment).

Une dernière option chaudement recommandée (liée à la situation actuelle relative au SPAM) est `noreceipts` qui désactive l'envoi des DSN (*Delivery Status Notification*).

Pour synthétiser, voici la liste complète des options que je recommande pour le paramètre *m4* `confPRIVACY_FLAGS` (nous reparlerons de l'option `authwarnings` un peu plus tard) :

```
define(`confPRIVACY_FLAGS',`novrfy,noexpn,noetrn,noreceipts,   \
  restrictqrun,restrictmailq,restrictexpand,authwarnings')dnl
```

Sendmail – TLS

Le protocole TLS permet d'assurer le chiffrement des données qui sont transmises entre deux MTA (ou d'un MUA vers un MTA) ainsi que l'authentification des deux parties. Nous ne nous pencherons que sur l'aspect chiffrement.
SMTP supporte TLS via l'opération `StartTLS`.

Le comportement par défaut est d'initier une demande de chiffrement si le MTA auquel se connecte Sendmail présente la capacité `STARTTLS` après l 'échange des salutations :

```
puck:~# telnet gaia.anet.fr smtp
Trying 80.118.2.10...
Connected to gaia.anet.fr.
Escape character is '^]'.
220 gaia.anet.fr ...
EHLO you
250-gaia.anet.fr ...
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-AUTH DIGEST-MD5 CRAM-MD5
250-STARTTLS                                    <--- ici
250 HELP
quit
```

Sendmail – TLS

Il faut pour cela que Sendmail soit en possession de certificats X.509 signés. Les directives *m4* suivantes permettent de spécifier où les trouver :

```
define(`confCACERT_PATH', `/etc/ssl/certs')dnl
define(`confCACERT',      `/etc/ssl/certs/ca.crt')dnl
define(`confSERVER_CERT', `/etc/ssl/certs/gaia.anet.fr-mta.crt')dnl
define(`confSERVER_KEY',  `/etc/mail/private/gaia.anet.fr-mta.key')dnl
define(`confCLIENT_CERT', `/etc/ssl/certs/gaia.anet.fr-mta.crt')dnl
define(`confCLIENT_KEY',  `/etc/mail/private/gaia.anet.fr-mta.key')dnl
```

Il est possible d'utiliser ou non le même certificat lorsque Sendmail est client ou serveur. La génération et la signature des certificats se fait généralement avec OpenSSL.

Attention, malgrès cette configuration, cela ne signifie pas du tout que tous les messages seront chiffrés lors de leur transmission. Certains le seront si les MTA impliqués supportent tous les deux TLS.
De plus, les messages sont toujours stockés en clair dans les files d'attente. Si la confidentialité est un enjeu important, il faudra faire appel à des techniques implémentées sur les MUA (PGP, GPG ou S/MIME).

Sendmail – forge

Ici encore, il s'agit d'un problème qui va être très délicat à traiter au niveau du MTA. Cela est dû à la nature intrinsèque du protocole SMTP.

Quelques techniques existent pour vérifier *a posteriori* si un message a été forgé, mais rien qui ne soit vraiment fiable (en-têtes `Received:`, informations `IDENT` si elles sont présentes, avertissements insérés par Sendmail si l'option `authwarnings` du paramètre `PrivacyOptions` est activée, etc.).

Pour résumer :

- ne jamais prendre pour argent comptant ce que vous recevez par émail ;
- ou bien utiliser PGP, GPG ou S/MIME.

Sendmail – programmes externes

Sendmail propose plusieurs mécanismes qui peuvent être utilisés pour faire appel à des programmes externes. Ceux-ci permettent de vérifier le contenu des messages qui transitent parle MTA et, éventuellement, d'en rejeter certains.

Pour faire passer les messages via un programme externe, l'une des techniques suivantes est utilisée :

- utilisation d'un MDA élaboré (par exemple procmail) ;
- déclaration d'un *mailer* spécifique ;
- utilisation d'une double file, chacune étant gérée par un processus Sendmail dédié (approche retenue par MailScanner ou Amavis) ;
- et, surtout, l'API Milter.

En général chaque programme supporte une approche et il faut se référer à sa documentation d'installation.

Petit lexique

MTA *Mail Transfert Agent*
Logiciel qui a pour rôle le transfert des messages d'un serveur à l'autre (Sendmail, Postfix, Exchange, Notes, etc.).

MUA *Mail User Agent*
Logiciel utilisé par les utisateurs pour lire leurs messages (Outlook, Mozilla, Eudora, Mutt, etc.).

MSP *Mail Submission Program*
Programme permettant d'insérer un message dans la file d'un MTA local.

MSA *Mail Submission Agent*
Démon (écoutant génralement sur le port TCP 587) dédié à l'envoi des messages par les MUA *via* SMTP

MDA *Mail Delivery Agent*
Programme appelé par le MTA afin de prendre en charge

DSN *Delivery Status Notification*
Message envoyé en retour à l'expéditeur lorsque le message initial a été délivré ou lu

RBL *Real-time Block List*
Service base sur le protocole DNS et qui permet d'interroger en temps réel des bases de données en ligne

SMTP *Simple Mail Transfert Protocol*
Protocole de communication utilisé pour transmettre un message d'un MTA à l'autre ou d'un MUA vers un MTA

SSL *Socket Secure Layer*
Protocole permettant d'ajouter des fonctionnalités de cryptographie et d'authentification à la couche TCP d'IP

TLS *Transport Layer Security*
Successeur de SSL

Petit lexique

SASL *Simple Authentification and Security Layer*
Standard qui permet d'augmenter le vocabulaire d'un protocole (par exemple SMTP) dans le but de lui ajouter des fonctionnalités d'authentification et/ou de chiffrement

PAM *Pluggable Authentication Modules*
Ensembles de librairies qui permettent de contrôler finement l'authentification des services

LDAP *Lightweight Directory Access Protocol*
Protocole dont le but est de normaliser l'accès à des serveurs d'annuaire

Pour aller plus loin...

Voici une liste (bien entendu non exhaustive) de logiciels ou technologies à même de fournir une aide précieuse afin de fournir un service SMTP de qualité :

- MIMEDefang environnement de filtrage des flux SMTP
- Nagios supervision active des services et ressources (sondes spécialisées)
- SNMP protocole de supervision actif ou passif
- ClamAV anti-virus
- OpenSSL infrastructure pour l'utilisation de SSL/TLS

Références

Livres

« Sendmail », 3ème édition
Bryan Costales – Éditions O'Reilly

« Sendmail Cookbook »
Craig Hunt – Éditions O'Reilly

« Sendmail : Installer, administrer et optimiser un serveur de messagerie »
Thibaut Maquet – Éditions Eyrolles

Sur le Web

http://www.sendmail.org

Autres

`/usr/share/doc/sendmail/README.cf` (RedHat)
`/usr/share/doc/sendmail/cf.README.gz` (Debian, paquet `sendmail-doc`)

Pour en savoir plus...

Sébastien Namèche

Conseil

- Architecture des systèmes et réseaux
- Sécurité transverse

Formation

- Logiciels Libres
- Systèmes Unix
- Réseaux et protocoles IP
- Oracle
- annuaires
- Messageries
- Supervision

Tél. +33 2 9907 2144
Mob. +33 6 0373 1442
http://sebastien.nameche.fr
sebastien@nameche.fr