TP ISRI MESSAGERIE noté
27 Mars 2025

Nom :
Prénom :
Numéro Étudiant :

Le but de ce TP est de comprendre pourquoi il est important de maîtriser le contenu et le fonctionnement des protocoles SMTP, IMAP, POP. Dans cet exercice vous allez mettre en place une petite architecture de messagerie.

- Mettre en place un serveur SMTP postfix
- Mettre en place un serveur pop/imap
- Mettre en place quelques directives postfix pour agir sur le filtrage des messages
- Mettre en place une solution anti-virale de messagerie basée sur Clamav

Travail à réaliser :

- Installer votre serveur SMTP postfix et placer les directives minimales dans le fichier main.cf. Créer 2 comptes systèmes et tester le fonctionnement en envoyant un message d'un compte vers l'autre comme vous l'avez fait en TD avec Telnet. Vérifiez au niveau système via la commande mail (en étant connecté via le compte destinataire du message) ou directement en affichant le contenu du fichier mails (/var/spool/mail/Compte). Vérifier les traces du fichier de log (/var/log…).

root@debian:~/f# telnet 172.30.177.86 25
Trying 172.30.177.86...
Connected to 172.30.177.86.
Escape character is '^]'.
220 smtp.beauvelo.fr ESMTP Postfix (Debian/GNU) TP ISRI
…

- Vous Installez votre serveur pop/imap uw-imap (voir la suite). Vous envoyez des messages comme précédemment et vous les récupérez via vos serveur pop et imap via un simple Telnet comme pour le TD.

- Vous allez un peu plus loin avec postfix en utilisant des directives pour limiter le contenu des commandes MAIL FROM: (directive *smtpd_sender_restrictions)* et RCPT To: (directive smtpd_recipient_restrictions).
Vous interdisez également l'envoi de fichiers attachés avec une extension particulière (ex : .exe).

- Vous installer une solution de messagerie anti-virale basée sur Clamav.

Présentation de l'infrastructure Postfix

Cette présentation ne reprend pas les concepts des protocoles de messagerie.(voir le cours et le TD pour cela). Pour utiliser POSTFIX il faut un minimum de connaissance sur ce serveur SMTP. C'est le but de cette présentation. Les informations de ce document ne sont pas nécessairement utilisées dans le cadre de ce TP mais reste un minimum pour la compréhension de son fonctionnement.

Vous pouvez consulter la traduction de la documentation postfix sur :
http://postfix.traduc.org/

Le processus principal est le programme « master » qui est lancé par le script de démarrage du démon. Il est résident et se charge d'appeler les différents modules en fonction des traitements et des besoins :

- Réception des messages,
- Analyse des messages,
- Gestion des files d'attente,
- Routage des courriers,
- Distribution des courriers,
- …

* un schéma représente ces processus dans la suite de ce document.

Il existe 2 fichiers principaux pour la configuration de Postfix qui sont « *main.cf* » et « *master.cf* ».

L'ensemble des processus composant postfix sont définis dans master.cf. Le fichier main.cf permet de configurer l'ensemble de l'infrastructure à travers les variables de configuration.
* Certaines variables peuvent être surchargées en fonction des besoins dans le fichier master.cf.

main.cf :
Le fichier de configuration typiquement Linux. Toute la configuration générale du serveur de messagerie. Les entrées sont du type VarX= Valeur ou VarY=$VarX ou multi-valeurs var= val1,val2,val3. Certaines variables peuvent aussi faire référence au contenu d'un fichier. La variable permet alors de définir le type et le nom de fichier de définition associé. Pour générer certains fichiers avec des formats particuliers on utilisera la commande « postmap ».

Exemple :
« Transport_maps = hash:/etc/postfix/Nom_Fic » permet de définir un fichier contenant les valeurs des transports dans un format « hash » (Postfix fourni la commande **Postmap** pour générer des fichiers dans un format particulier à partir d'un fichier texte).

master.cf :
Les lignes de ce fichier sont utilisées pour initialiser les différents modules (processus) composants postfix. Pour différentes instances de programmes de même nature (smtpd généralement) on peut faire varier la configuration en redéfinissant la valeur au niveau de la définition du processus par surcharge des valeurs définies dans main.cf

Les exercices de la suite du cours vous permettront de mieux saisir cette particularité de la configuration.

Chaque ligne de ce fichier est composée de colonnes permettant de préciser le comportement du processus (instance(s) de MTA, client(s) smtp, agent(s) de distribution…).

Exemple:

```
clientsmtp      unix  -       -       n       -       16      smtp

127.0.0.1:2626 inet  n   -  n       -       16      smtpd
    -o content_filter=spamassassin
    -o smtp_send_xforward_command=yes
```

Les différents « processus de traitement » (fils du processus master):

master est le processus père de tous les autres. Il supervise l'ensemble du système. Il est lancé par le script de démarrage de postfix. master lance ensuite :

- pickup : lit les messages dans la file d'attente locale (file maildrop)
- cleanup : vérifie la cohérence des messages locaux et réseaux
- trivial-rewrite : Traitement des messages (vérifie, modifie, ajoute les entêtes manquantes, convertie les adresses, choisit le bon transport…)
- bouce deferred : Créer un rapport de non-délivrance (notification)
- qmgr est le gestionnaire des files d'attentes

Les agents d'entrés et de distributions principaux :

Entrée :

- smtpd : un serveur smtp (réception des messages du réseau)
- postdrop : processus qui réceptionne les messages locaux (comptes systèmes…)

Distribution :

- smtp, lmtp : se comporte comme un client de messagerie smtp
- - pipe : processus qui attend des infos en entrée pour le faire passer à un script, driver…
- - virtual : se charge des messages pour les boites dîtes « virtual » (compte non système)

Les files d'attente :

- maildrop : les messages locaux tombent dans cette file d'attente (provenance postdrop)
- incoming : les messages conformes (après analyse par cleanup et trivial-rewrite) transitent dans cette file. Ils sont passés à la file active.
- active : file des messages prêt à être traité par un transport (agent de distribution).
- deffered : file des messages ne pouvant être distribué
- corrupt : file des messages corrompus

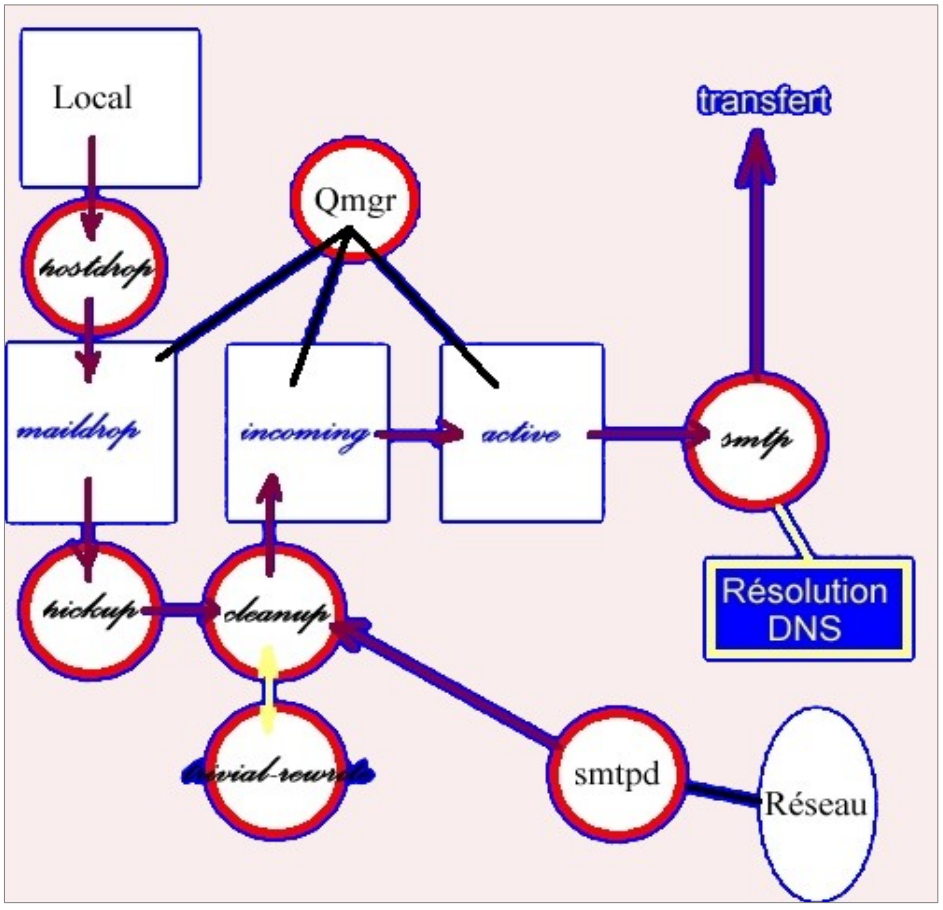
- hold : file des messages conservés indéfiniment.

Exemple de la liste des processus (utilisation pstree)

```
├─master(2139)─┬─cleanup(6663)
│              ├─local(6665)
│              ├─pickup(6302)
│              ├─qmgr(4455)
│              ├─trivial-rewrite(6664)
│              └─smtpd(6735)
```

Bien entendu, l'affichage peut varié en fonction du moment où nous tapons la commande. En effet, certains processus sont exécutés à intervalles réguliers ou à l'arrivée d'événements particuliers.

Synoptique simplifié (cas local et réseau):



Les notions essentielles :

La notion de domaine :
- Domaine Canonique : domaine local, domaine du serveur
- Domaine relayé : les domaines a relayer (on ne stocke pas dans une boite, on ne sert que d'intermédiaire (MX de faible valeur pour un domaine ami). Les messages sont en file d'attente.
- Domaine virtuel : domaine hébergé mais pas canonique. On distingue les domaines d'alias virtuel (re-routage vers comptes unix) et les boites aux lettres virtuelle (boites ne correspondant pas à un compte système : les messages sont stockés dans des fichiers ou répertoires) ou utilisent un transport particulier pour router les messages (lmtp, pipe...).

Les alias :
Les alias permettent de rediriger une adresse vers une autre, un domaine par un autre. Les alias sont configurés à travers une table des alias. La variable « virtual_alias_maps » est utilisée pour indiquer le format et le nom du fichier de stockage des alias.

Exemple :
Virtual_alias_maps = hash:/etc/postfix/virtual.alias

Dans le fichier alias on a:

Adresse1@doamine adresse@domaine...

Les fichiers « listes » après une modification se reconstruisent via la commande « postmap Nom_Fichier ». La commande produit un ficher au format indiqué et utilisé par le système.
La commande « postmap /etc/postfix/virtual.alias » va produire un fichier .db au format indiqué

Routage par les transports :

Il existe une table de transport pour diriger les messages en fonction de critères définis. La variable transport_maps permet d'indiquer le format et le « fichier liste » qui permet ce routage.
« Transport_maps = hash:/etc/postfix/transport »

Dans le fichier transport on a :
adresse@domaine Nom_du_transport

Nom_du_transport correspond à une entrée du fichier master.cf (un pipe, un script...)
Attention! Avant de faire un transport, le fichier des alias est consulté.

Dans tous les cas, on passe d'abord par les alias (virtual_alias_maps) donc si on veut qu'une adresse particulière d'un domaine virtuelle par exemple soit redirigée vers une adresse locale on peut le faire en plaçant une entrée dans cette table. On peut également créer un compte fourre tout en plaçant le nom de domaine virtuel sur boite unique (ex: @etab.ac-amiens.fr service). Par contre, si on place cette entrée dans le fichier des alias plus rien n'arrive au domaine virtuel car les alias sont prioritaires sur les domaines virtuels (voir la suite de la présentation).

Réécrire des émetteurs et destinataires d'un mail :

Cette fonction utilise les canonical maps. Celles-ci peuvent être appliqués en émission, en réception ou les deux.

Voyons comment opérer au moyen de 2 exemples simples.

1- Changement d'émetteur :

Admettons que vous avez un utilisateur root@monserveur.lan, nous allons réécrire l'adresse avec une canonical map afin de la faire apparaître en tant que root-serveur@example.org

Ajoutez la ligne suivante dans votre fichier /etc/postfix/main.cf

sender_canonical_maps = hash :/etc/postfix/sender_canonical

Insérez la ligne suivante dans le fichier /etc/postfix/sender_canonical.

root@monserveur.lan root-serveur@example.org

Enfin tapez :

postmap /etc/postfix/sender_canonical

service postfix reload

2- Changement de destinataire

Vous souhaitez que les messages adressés à hostmaster@dns1.example.org soient adressés à dns-master@example.org. Nous allons créer une canonical map pour cela.

Ajoutez la ligne suivante dans /etc/postfix/main.cf

recipient_canonical_maps = hash :/etc/postfix/recipient_canonical

Insérez la ligne suivante dans le fichier /etc/postfix/recipient_canonical :

hostmaster@dns1.example.org dns-master@example.org

Enfin tapez :

postmap /etc/postfix/recipient_canonical

service postfix reload

Changer un nom de domaine

Installation du serveur SMTP : Postfix

Par les paquets binaires ou sources.

Sources :

Assurez-vous d'avoir une machine parfaitement configurée: c'est important pour le fonctionnement de postfix ! (réseau, résolution DNS, Domaine, nom de la machine, fichier hosts…),
Vérifier la version correspondant à votre distribution,
Vérifier et Installer les dépendances,
Télécharger et installer cette version à partir des sources (téléchargement des sources par par les dépôts et installation par make, make install sans le .configure),
Le service va fonctionner via UID et GID postfix/postdrop, créer cet utilisateur et ce groupe (adduser, groupadd, addgroup).
Vérifier que postfix appartient au groupe postdrop,
Installer si ce n'est pas le cas le MDA procmail (dans le cadre du TP, c'est lui qui va stocker le message dans le fichier mail de l'utilisateur du système dans /var/spool/mail/user).

Configuration du serveur SMTP : Postfix

Le fichier principal est « main.cf » dans le répertoire de configuration. La syntaxe est du type paramètre=val (comme pour de nombreux services linux classiquement). Pour donner à val la valeur d'un autre paramètre, il suffit de placer un « $ » devant le non du paramètre dans la zone valeur (comme pour les variables linux).

Ex :
Mon_domaine = dom.com
Mon_origine = $Mon_domaine

Les principaux paramètres du fichier main.cf :
Les chemins d'accès sont renseignés automatiquement après l'installation :
queue_directory : spécifie le répertoire des files d'attente
command_directory : spécifie le répertoire des commandes administratives
daemon_directory : spécifie le répertoire des démons de postfix
mail_spool_directory : spécifie le répertoire des boites mail
mail_owner : spécifie le compte utilisé par le démon et les queues
myhostname : le non du serveur pleinement qualifié
mydomain : le domaine du serveur
myorigin : utilisé pour renseigner la partie droite d'une adresse mail non renseignée ($myhostname ou $mydomain)
mydestination : permet d'indiquer pour qui on accepte les mails localement ($myorigin, localhost, une liste dans une base…)
alias_maps : indique l'ensemble des bases d'alias d'adresses (correspondances d'adresse)
alias_database : idem pour ce qui se rapporte au monde unix
mailbox_command : permet d'indiquer l'agent de délivrance à utiliser (voir le cours)
relayhost :Pour une délivrance des messages vers un autre MTA.

La conf minimale :

En plus des variables pré-renseignées (les chemins de configurations) il faut paramétrer un minimums de variables:
 # Le réseau autorisé à se connecter
mynetworks = l172.30.177.0/24
 # Le nom du domaine
mydomain = le domaine reconnu ex : in.ac-amiens.fr
 # Le nom de la machine pleinement qualifié
myhostname = ex : linux.in.ac-amiens.fr
 # Pour des adresses non pleinement qualifiées ajouter myorigin à droite du @
myorigin = $mydomain
 # Pour identifier les domaines locaux (à delivrer sur cette machine)
mydestination = $myhostname, localhost, $mydomain
 # identifier le MDA pour délivrer un message local
mailbox_command = /usr/bin/procmail

Il faut adapter cette configuration minimale à votre contexte.

L'agent de délivrance des messages sera **/usr/bin/procmail** car en effet, dans un premier temps, nous utilisons les comptes systèmes (linux) dans le cadre du TP.

Procmail est un MDA. Le fonctionnement de procmail est très simple : il est appelé avec un mail en entrée standard, et traite le mail en fonction des filtres contenus dans le fichier *.procmailrc* (si il existe) du répertoire home de l'utilisateur. Si ce fichier n'existe pas il le dépose simplement dans la boite locale de l'utilisateur /var/mail/compte. Vous pouvez également utiliser le MDA « virtual » pour gérer des comptes sans avoir à créer un compte sur le système (faire un « man virtual » pour obtenir plus d'infos).
Voir également l'excellent article sur :
http://www.linux-france.org/article/mail/procmail/procmail.html

alias_maps = hash:/chemin/aliases permet d'indiquer le fichier des alias. Il faut construire cette BDD (ici un simple format hash) via la commande postmap (même si le fichier est vide au début de l'installation) et vérifier la création du fichier .db sinon le service ne fonctionne pas correctement.

Pour une config mydestination=myhostname,localhost et mydomain, les comptes valides seront les adresses du type nom@NomMachine, nom@Localhost, nom@domaine . Les logs du démon sont stockés dans /var/log/maillog (Ils sont écrit par le démon syslog).

Le test de la configuration se fait via la commande « postfix check » pour lancer arrêter le service utiliser « start » et « stop » à la place de « check ». Pour avoir la configuration de toutes les variables même celles par défaut la commande est « postconf » et pour avoir les valeurs par défauts il faut utiliser l'option -d.

Installation du serveur pop/imap de l'université de Washington:

Les sources sont accessibles via le site http://www.washington.edu/imap/
(imap-2007f pour la version disponible)

Téléchargement et installation:
wget https://www.mirrorservice.org/sites/ftp.cac.washington.edu/imap/imap-2007f.tar.gz
apt-get build-dep uw-imap
vi Makefile → modifier les variables
make slx (slx pour une compilation pour un système Linux).
Vous obtenez 2 binaires ipop3d et imapd utilisables sous inetd ou xinetd

Installation par le paquet source de votre distribution (si 22.04 alors changer xenial par jammy)

https://packages.ubuntu.com/fr/source/xenial/uw-imap pour ubuntu xenial

add-apt-repository "deb http://archive.canonical.com/ubuntu $(lsb_release -sc) partner"
apt-get update
apt source –download-only uw-imap

Dans le répertoire des sources :

root@PV-Linux:/usr/src/imap-2007f# make slx

```
make[1]: entrant dans le répertoire « /usr/src/imap-2007f »
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Building in NON-COMPLIANCE with RFC 3501 security requirements:
+ Non-compliant:
++ TLS/SSL encryption is NOT supported
++ Unencrypted plaintext passwords are permitted
+
+ In order to rectify this problem, you MUST build with:
++ SSLTYPE=nopwd
+ You must also have OpenSSL or equivalent installed.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Do you want to continue this build anyway? Type y or n please:

Y
…
`cat ../c-client/CCTYPE` -I../c-client `cat ../c-client/CFLAGS`  -c -o tquota.o tquota.c
`cat ../c-client/CCTYPE` -I../c-client `cat ../c-client/CFLAGS` -o tmail tmail.o tquota.o ../c-client/c-client.a `cat ../c-client/LDFLAGS`
```

make[2]: quittant le répertoire « /usr/src/imap-2007f/tmail »

make[1]: quittant le répertoire « /usr/src/imap-2007f »

root@PV-Linux:/usr/src/imap-2007f#

Les binaires ipopd et imapd sont bien compilés :

$ file ipop3d
ipop3d: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=6104446a50de557803fce8d270cc21e3480e32c7, with debug_info, not stripped

root@PV-Linux:/usr/src/imap-2007f# ls -l ipopd

total 6028

-rwxr-xr-x 1 root dip 2942668 déc. 19 09:57 ipop2d

lrwxrwxrwx 1 root dip   38 déc. 19 09:57 ipop2d.c ->

-rw-r--r-- 1 root dip 100872 déc. 19 09:57 ipop2d.o

Copier les exécutables dans *usr*/bin :

root@PV-Linux:/usr/src/imap-2007f# cp ./ipopd/ipop3d /usr/bin

root@PV-Linux:/usr/src/imap-2007f# cp ./imapd/imapd /usr/bin

Ce paquetage renferme un serveur pop(s) et un serveur imap(s). La configuration par défaut compile l'application avec la couche SSL. Nous simplifions l'installation, en supprimant la couche SSL, utilisation en ipv4 et sans chiffrement des mots de passe. il est donc nécessaire de modifier les directives de compilation du fichier make pour ça (fichier Makefile) option SSLTYPE, IP, et PASSWDTYPE.

Ces services fonctionnent au dessous du super démon inetd (inetd, ou xinetd ou openbsd-inetd en fonction des distributions. Faire la compilation en fonction de votre système (linux : make slx).

Copier le démon dans le répertoire /usr/sbin (imapd et ipopd), vérifier le bit x, ajouter les lignes nécessaires dans /etc/inetd.conf ainsi que dans le fichier /etc/services si besoin. (PASSWDTYPE=std, SSLTYPE=none, IP=4)

N'oublier pas de relancer le service xinetd (ou la variante de votre distribution).

Exemple de configuration :

Pour inetd:

```
pop3    stream    tcp    nowait    root    /usr/sbin/ipop3d    ipop3d
```

Pour xinetd :

```
service pop3
{
        socket_type        = stream
        wait               = no
        user               = root
        server             = /usr/bin/ipop3d
        log_on_success  += HOST DURATION
        log_on_failure   += HOST
        instances          = 150
        cps                = 70 30
        disable            = no
}
```

Relancer le service xinetd et tester la connexion :

```
root@PV-Linux:/etc/xinetd.d# telnet 10.1.16.50 110

Trying 10.1.16.50...
Connected to 10.1.16.50.
Escape character is '^]'.
+OK POP3 PV-Linux 2007f.104 server ready

QUIT

+OK Sayonara

Connection closed by foreign host.
```

Test de la configuration (voir TD):

Connectez vous sur le serveur pop et imap avec l'un des deux comptes et vérifiez la connexion. La connexion se fait en mode commande via un telnet sur le port 110 (143 pour imap).

Nous avons donc maintenant une architecture de messagerie opérationnelle avec un MTA serveur (SMTP) gérant le domaine de l'entreprise et les MAA serveurs (IMAP, POP) pour permettre aux usagers de retirer leurs messages. Il faut maintenant consolider notre infrastructure (anti-virus, anti-spam...).

Un peu plus loin avec Postfix...

Postfix permet de nombreux contrôles paramétrables : sur le contexte de la communication (réseaux, ip, résolution DNS…), au niveau des commandes : Enveloppe : HELO, MAIL FROM…, au niveau des variables d'en-têtes, du corps du message...

voir http://postfix.traduc.org et http://www.postfix.org/uce.html

C'est ce que nous allons voir maintenant...

Mise en place des contrôles d'accès au serveur de mail : Un début d'anti-spam !

Exemples de paramètres de contrôles :
au niveau de la connexion SMTP (directive *smtpd_client_restrictions)*
au niveau de l'adresse donnée dans la phase HELO/EHLO (directive *smtpd_helo_restrictions)*
au niveau de l'adresse donnée dans la phase MAIL FROM: (directive *smtpd_sender_restrictions)*
au niveau du RCPT TP: (directives smtpd_recipient_restrictions= reject_unauth_destinations)

Exemple:

Par défaut Postfix accepte tous les mails de n'importe quel site émetteur. Pour constituer une liste noire , on peut créer un fichier */etc/postfix/access* dont la syntaxe est la suivante (c'est un moyen rapide pour créer une règle d'anti-spam) :

dom1.com REJECT
dom*2.com REJECT*
Sdom3.dom..com OK

Comme expliqué précédemment, ce fichier sera indexé via la commande « postmap » dbm:*/etc/postfix/access* ou hash:*/etc/postfix/access* selon les types d'indexation supportés par votre compilation» (nous utiliserons le format hash). Cette indexation va permettre d'accélérer la recherche du contenu.

Ce fichier de liste-noire peut être utilisé dans les variables "*smtpd_sender_restrictions"* et "*smtp_client_restrictions"* pour restreindre les adresses émettrices dont on accepte les mails

smtpd_sender_restrictions = check_sender_access hash:*/etc/postfix/access*

Mettre en oeuvre pour interdir l'envoi d'un message par un utilisateur particulier.

Interdire les fichiers attaché dangereux :

Il est également possible de sécuriser la boite aux lettres de vos utilisateurs en testant les pièces jointes transitant par le serveur.

Ex : refuser les fichiers portant une extension « dangereuse » exe com bat…

Ligne à rajouter dans /etc/postfix/main.cf :

mime_header_checks = regexp:/etc/postfix/mime_header_checks

Cela permet d'indiquer de tester les formats mimes via un fichier de règles construites via des expressions régulières.

Le contenu du fichier avec la règle :

```
/filename="?.*\.(bat|com|pif|vb|exe|lnk|scr|reg|chm|wsh|js|inf|shs|job|ini|
shb|scp|scf|wsc|sct|dll|msc|msi|ppt|pps|mdb|rar|bmp|wav|mpg|mpeg|wma|wmv)/
REJECT
/^Content-(Disposition|Type):.*(file)?name="?.*\.(bat|com|pif|vb|exe|lnk|
scr|reg|chm|wsh|js|inf| shs|job|ini|shb|scp|scf|wsc|sct|dll|msc|msi|ppt|
pps|mdb|rar|bmp|wav|wma|wmv)/ REJECT Extension de fichier joint refusée.
Desole!
```

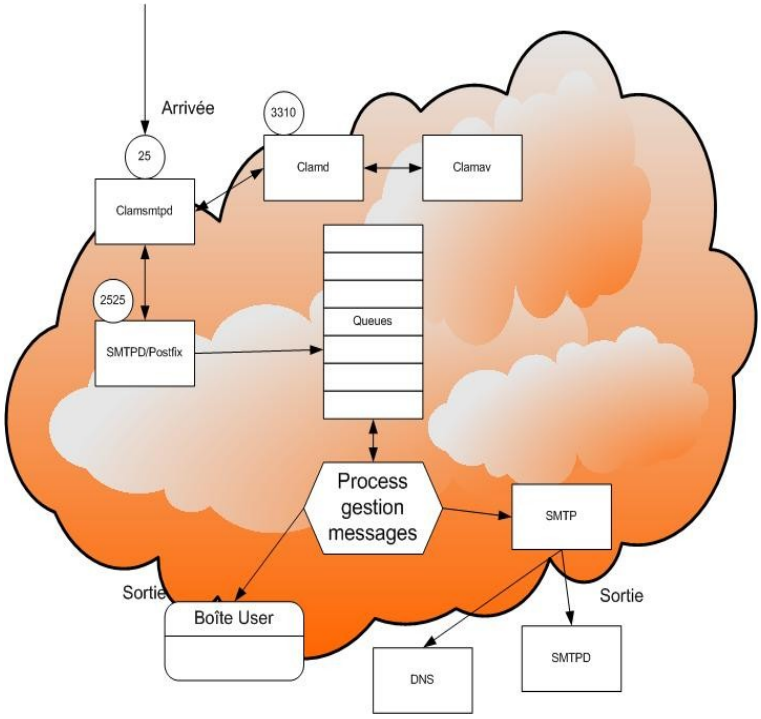
Mettre en oeuvre pour interdire l'envoi d'un fichier avec une extension particulière.

Installation d'un Antivirus en Amont de votre serveur

- Vous changez le port d'écoute de votre serveur de messagerie configuré précédemment pour écouter sur le port 2525 et sur l'ip 127.0.0.1. Vous vérifiez le bon fonctionnement par un test d'envoi d'un message par telnet sur l'IP 127.0.0.1 port 2525.

- Vous installez la chaîne Clamsmtp Clamd/clamav pour tester les virus dans les messages. Clamsmtp tourne sur le port 10025 et clamd sur le port 3310. Clamsmtp reçoit les connexions SMTP (les messages) les forward à Clamd qui forward le message à Clamav. Le flux scanné par clamav fait le chemin inverse. Le flux scanné est ensuite forwardé au serveur Postfix sur le port 2525.

Installation d'un anti-virus pour les messages



Attention ! Erreur sur le graphique clamsmtp tourne sur le port 10025 !

Clamav est une application qui permet de scanner un ensemble de fichiers directement via la commande clamscan. La commande Freshclam permet de télécharger les signatures. Clamd est une brique suplémentaire qui va recevoir le flux SMTP pour le passer à clamav.

Clamd est un socket qui reçoit des commandes spécifiques pour agir sur clamav avec le service clamav. A travers un jeu de commande clamd passe le message à clamav qui l'analyse.Clamd récupère le message et le passe à Postfix (127.0.0.1 port 2525).

Après l’installation clamd/clamav, on teste le fonctionnement:

à travers la commande PING :

root@tech-Msg:/home/tech# telnet 127.0.0.1 3310
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
PING
PONG

à travers la commande SCAN Fichier
root@tech-Msg:/home/tech# telnet 127.0.0.1 3310
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
SCAN /home/tech
/home/tech: File path check failure: Permission denied. ERROR
/home/tech: OK
Connection closed by foreign host.

Dans notre cas ces un flux de données que l’on va tester donc pas de problèmes de droits de fichiers.

Avec un fichier appartenant à clamav :

tech@tech-Msg:~$ telnet 127.0.0.1 3310
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
SCAN /tmp/test
/tmp/test: OK

Avec le test « virus » eicar :

Dans /tmp
wget https://secure.eicar.org/eicar.com
sudo chown clamav eicar.com

tech@tech-Msg:~$ telnet 127.0.0.1 3310
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
SCAN /tmp/eicar.com
/tmp/eicar.com: Win.Test.EICAR_HDB-1 FOUND
Connection closed by foreign host.

Installation via les paquets binaires clamd/clamav:

apt-get install clamav
apt-get install clamav-daemon

Via les sources du projet:
wget http://downloads.sourceforge.net/clamav/clamav-X.tar.gz
tar -zxvf clamav-X.tar.gz (ou via un apt-get source)
cd clamav-X

Via les sources de la ditribution Ubuntu:

apt-get build-dep clamav
apt-get –download-only clamav
vi INSTALL
groupadd clamav
useradd -g clamav ... avec les options qui vont bien !
./configure --prefix=/usr/local/clamav –disable-zlib-vcheck --disable-unrar
make
make install
cd /usr/local/clamav/
cd etc
./clamconf
cp clamd.conf clamd.conf.ori
vi clamd.conf
Voir les signatures
ll /usr/share/share/clamav/
vi /usr/local/clamav/etc/freshclam.conf
../bin/clamconf
#Dieser Exemple
chown et chgrp clamav clamd.conf
lire la conf par clamconf (attention au chemin)
#Mise a jour des signatures
../bin/freshclam
chown -R clamav /usr/local/clamav
#Lancer un scan d'un repertoire
../bin/clamscan -r /home
créer un fichier avec la signature Eicar:
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!
$H+H*
#Lancer en mode demon
./sbin/clamd
touch /var/log/clamd.log
chown clamav /var/log/clamd.log
./sbin/clamd
netstat -lnt
vi clamd.log

touch /var/run/clamd.pid
chown clamav /var/run/clamd.pid

Vérifier le fonctionnement (ps, netstat...)

Le fichier clamd.conf pour une écoute sur TCP/IP:

LocalSocket /tmp/clamd.socket

LogFile /usr/local/clamav/var/log/clamd.log

LogFileUnlock yes

LogFileMaxSize 200M

LogTime yes

LogClean yes

LogVerbose yes

PidFile /var/run/clamd.pid

TemporaryDirectory /var/tmp

DatabaseDirectory /usr/local/clamav/share/clamav

TCPSocket 3310

TCPAddr 127.0.0.1

User clamav

ScanMail yes

Clamav tourne sur une socket Unix et sur une socket inet (réseau) sur TCP port 3310.

Fonctionnement :

clamd reconnaît quelques commandes (voir man clamd), on peut lui passer un flux de données à analyser. Ce n'est pas un MTA, il ne peut donc pas dialoguer directement en SMTP comme un MTA. C'est pourquoi, nous allons utiliser un MTA spécialement créer pour dialoguer avec les serveurs ou les clients SMTP et passer le flux à clamd.

Installation du service clamsmtp:
Comme d'habitude il existe différentes possibilités :

- apt-get install clamsmtp
- wget http://ftp.de.debian.org/debian/pool/main/c/clamsmtp/clamsmtp_1.10.orig.tar.gz
- apt-get –download-only clamsmtp (avec un apt-get build-dep clamsmtp)

Dans le cadre d'une installation via les sources:

```
 tar -zxvf clamsmtp-X.tar.gz
 cd clamsmtp-X
./configure --prefix=/usr/local/clamsmtp
make
make install
cd /usr/local/clamsmtp/
mkdir etc
cp /chemin/clamsmtp-1.8/doc/clamsmtpd.conf ./etc
cd etc
# Modifier le fichier de conf
vi clamsmtpd.conf

#création d'un fichier pid vide
 cat > /var/run/clamsmtp.pid
chown clamav.clamav /var/run/clamsmtp.pid
```

Le fichier de clamsmtpd.conf:

Attention clamsmtp tourne sous le compte précisé par la directive User. Il faut donc utiliser les ports hauts si le daemon tourne sous un autre compte que root. Il est préférable d'utiliser le même compte pour clamsmtpd et clamav car les processus vont s'échanger les "fichiers mails".
Pour ne pas compliquer le TP nous allons faire tourner clamsmtp sur le port 25 sous le compte root qui va faire passer à clamd qui va également tourner sur le compte root sous le port 3310 qui va repasser à clamsmtp qui va à son tour passer le mail à une instance de postfix configuré sur le port 2525 sur l'ip loopback (127.0.0.1). Nous n'avons plus besoin d'ouvrir postfix sur le réseau car c'est clamsmtpd qui va se charger de la réception des messages.

Tester la chaîne en envoyant un mail puis tester de nouveau en envoyant un fichier avec une signature virale (Eicar).

Le contenu de clamsmtp.conf:

cat /etc/clamsmtpd.conf

MaxConnections: 16

TimeOut: 30

Send XCLIENT commands to receiving server
#XClient: off

Address to listen on (defaults to all local addresses on port 10025)
clamsmtp écoute ici...
Listen: 127.0.0.1:10025

The address clamd is listening on
la conf de clamd est juste à la suite...
ClamAddress: 127.0.0.1:3310

 # A header to add to all scanned email
 # un entête est ajouté au message à la sortie...
 # On pourra vérifier dans le code source du message reçu
Header: X-Virus-Scanned: ClamAV using ClamSMTP Mis en œuvre par P.VANIET

Directory for temporary files
#TempDirectory: /tmp

What to do when we see a virus (use 'bounce' or 'pass' or 'drop'
Action: drop

Whether or not to keep virus files
#Quarantine: off

Enable transparent proxy support
#TransparentProxy: off

User to switch to
User: clamav

Virus actions: There's an option to run a script every time a virus is found.
!IMPORTANT! This can open a hole in your server's security big enough to drive
farm vehicles through. Be sure you know what you're doing. !IMPORTANT!
#VirusAction: /path/to/some/script.sh

PidFile: /var/run/clamsmtp.pid

Faire suivre le message scanné vers le serveur de stockage des messages ou spamassassin ou ...
OutAddress: 127.0.0.1:2525

La conf du postfix pour ça :

Modifier votre serveur postfix pour écouter sur 127.0.0.1 port 2525

Que faut il faire maintenant pour avoir une infrastructure complètement opérationnelle ?

Ajouter un serveur postfix en entrée et en écoute sur le port 25 qui transfert les messages reçu sans les traiter à clamsmtp qui écoute cette fois sur 127.0.0.1 port 10025.

Comment faire ?

A vous de jouer...