Logiciel

Vsftpd : cinq exemples de configuration

Une configuration classique

Supposons que vous souhaitez mettre en place un serveur de fichiers, accessible à tous (par exemple par Internet). Votre fichier vsftpd.conf pourrait alors ressembler à ceci :

/etc/vsftpd/vsftpd.conf

```
listen=YES
max_clients=200
max_per_ip=4
anonymous_enable=YES
local_enable=NO
write_enable=NO
anon_upload_enable=NO
anon_mkdir_write_enable=NO
anon_other_write_enable=NO
anon_world_readable_only=YES
connect_from_port_20=YES
hide_ids=YES
pasv_min_port=50000
pasv_max_port=60000
xferlog_enable=YES
ls_recurse_enable=NO
ascii_download_enable=NO
async_abor_enable=YES
one_process_model=YES
idle_session_timeout=120
data_connection_timeout=300
accept_timeout=60
connect_timeout=60anon_max_rate=50000
Examinons-le en détail :
listen=YES
```

Le serveur est positionné en mode standalone, ce qui est normal puisqu'il sera a priori sollicité fréquemment.

max_clients=200
: 200 clients simultanés sont acceptés. Les autres recevront un message d'erreur jusqu'à ce qu'une place se libère. De la sorte, le serveur ne sera pas saturé par les requêtes.

max_per_ip=4
: Chaque client ne pourra ouvrir que 4 connexions simultanées. Il n'y aura donc pas de client qui saturera la bande passante.

anonymous_enable=YES

Nous autorisons les connexions anonymes, ce qui est normal puisque le serveur est public.

local_enable=NO

Nous refusons les connexions d'utilisateurs dotés de compte sur la machine. Seules les connexions anonymes seront donc acceptées. Soyons clair : l'idée ici n'est pas d'empêcher vos utilisateurs locaux chéris de se connecter, mais de prévenir des hacks faciles utilisant les comptes standards présents sur toute machine digne de ce nom, comme Apache, gdm ou autre lp.

write_enable=NO

Personne ne modifie de fichier ni n'écrit sur notre serveur.

anon_upload_enable=NO

Les utilisateurs anonymes (donc les utilisateurs du serveur) ne peuvent pas uploader de fichier.

anon_mkdir_write_enable=NO

Les utilisateurs (anonymes) ne peuvent pas non plus créer de répertoire.

anon_other_write_enable=NO

Les utilisateurs (anonymes) ne peuvent pas non plus renommer ni supprimer ni fichier ni répertoire.

anon_world_readable_only=YES

Les utilisateurs (anonymes) ne peuvent télécharger que les fichiers accessibles en lecture à tous.

connect_from_port_20=YES

Nous acceptons les connexions depuis le port 20 (notre éventuel firewall laisse ce port passer).

hide_ids=YES

Tous les fichiers et répertoires du serveur sont montrés comme appartenant à FTP.

pasv_min_port=50000, pasv_max_port=60000

En mode passif, les ports acceptés sont dans la tranche 50000 à 60000. Tous les autres sont refusés sur le serveur et sans doute aussi sur le firewall. Qu'est-ce que cette chose-là ? Le mode passif est celui dans lequel le client, au lieu d'envoyer au serveur le numéro du port à partir duquel il souhaite que le serveur envoie les données, envoie PASV, laissant passivement le serveur décider quel port utiliser pour l'envoi de données, habituellement le port 20.

Cette directive est par défaut sur Yes, le mode passif est possible (ce qui est souvent nécessaire lorsqu'un firewall est présent avant le serveur. Le mode actif permet en revanche au client d'avoir la certitude, puisque c'est lui qui décide du port, que les données qu'il reçoit correspondent bien à sa demande initiale).

xferlog_enable=YES

Nous logons les transferts.

ls_recurse_enable=NO

Nous interdisons l'option -R de "ls -R", qui consomme trop de ressources machine.

ascii_download_enable=NO

Nous n'envoyons pas les fichiers en mode ASCII.

async_abor_enable=YES

Nous acceptons la commande async ABOR, pour certains de nos clients qui en ont besoin

(qui ? Nous ne savons pas, mais le serveur est public donc nous ne savons pas quel client FTP utilisera notre visiteur).
A quoi sert cette directive étrange ? Elle permet d'activer la commande FTP async ABOR, qui permet de stopper un téléchargement asynchrone en cours. Elle est considérée comme complexe et inélégante, parce qu'async ABOR produit des effets différents en fonction des clients (fermeture de session, déconnexion, etc.).
C'est pourquoi cette directive est par défaut sur No, désactivant le support de cette commande. Certains clients FTP ont cependant besoin d'async ABOR pour pouvoir annuler un téléchargement dans de bonnes conditions. Vous aurez donc peut-être besoin de positionner cette directive sur Yes dans certains cas, et c'est ce que nous choisissons de faire ici, puisque nous ne savons pas quel type de client est susceptible de se connecter chez nous.

one_process_model=YES
Si vous disposez d'un noyau 2.4 sous Linux, vous pouvez en activant cette fonction générer un processus par connexion. Il s'agit là d'une option moins " pure " en termes de sécurité, mais qui permet une performance parfois meilleure.
N'activez cette fonction, par défaut sur No, que si vous savez ce que vous faites et ce que ce changement implique et par ailleurs seulement si votre site reçoit un grand nombre de connexions simultanées.
C'est peut-être le cas pour ce serveur, nous l'activons et monitorerons naturellement la charge qu'elle provoquera.

idle_session_timeout=120
Nous acceptons 120 secondes d'inactivité de la part du client, au-delà il est rejeté pour laisser la place à un autre.

data_connection_timeout=300
Si un transfert est gelé pendant plus de 300 secondes, nous considérons que le client est déconnecté ou en échec et clôturons la session pour laisser la place à un autre.

accept_timeout=60
Nous donnons 60 secondes à un client en mode passif pour établir sa connexion. Au delà, nous clôturons pour laisser la place à un autre (60 secondes est largement assez, sauf si le client rencontre un problème important, qui handicapera de toutes les façons son transfert).

connect_timeout=60
Nous donnons le même délai au client qui se connecte en mode PORT.

anon_max_rate=50000
Nous autorisons un débit maximal de 50000 octets par seconde (49 ko/s) pour les clients.

Deuxième configuration : filtrage par adresses IP

Dans ce scénario, votre serveur héberge des documents que vous voulez mettre à la disposition d'une partie de votre réseau (par exemple le département comptabilité), mais vous voulez que d'autres utilisateurs, pourtant eux aussi sur le LAN, ne puissent pas les lire (par exemple le département marketing). Sachant que vous avez " ces facilités avec les serveurs ".

Votre patron vous demande en plus " quelques petits ajustements ", qui consistent à lui donner, à lui tout seul, des possibilités de download spéciales et privilégiées et de réduire les possibilités des stagiaires, qui ont tendance, c'est bien connu, à abuser du système… Vsftpd sait s'appuyer sur tcp_wrappers pour autoriser ou refuser une ou plusieurs adresses IP et appliquer une configuration

spécifique.

Support de tcp_wrappers

Cette opération suppose que vsftpd ait été construit avec le support de tcp_wrapper, ce qui est le cas des versions rpm et des installations standards. Si la manipulation que nous indiquons ici ne fonctionne pas dans votre configuration, alors c'est peut être que tcp_wrapper n'est pas activé dans votre version de vsftpd. Il vous suffit alors d'éditer le fichier builddefs.h et de remplacer #undef VSF_BUILD_TCPWRAPPERS par #define VSF_BUILD_TCPWRAPPERS. Recompilez alors vsftpd et le support sera activé.

Tcp_wrapper étant activé, il faut indiquer à vsftpd que nous souhaitons nous appuyer sur lui pour le filtrage. Ceci se fait en ajoutant dans /etc/vsftpd/vsftpd.conf la ligne :

```
tcp_wrappers=YES
```

Définition d'une politique de filtrage

Reste alors l'étape la plus importante, la création de la politique de filtrage dans tcp_wrapper. Elle se fait au travers de deux fichiers /etc/hosts.allow et /etc/hosts.deny, mais un seul fichier est suffisant la plupart du temps (puisqu'il est possible de préciser deny dans hosts.allow).

Éditons donc /etc/hosts.allow et modifions-le pour nos paramètres en profitant de l'occasion pour examiner les possibilités de ce filtre :

```
USER@MACHINE:~$ cat /etc/hosts.allow
vsftpd: 10.0.0.99: setenv VSFTPD_LOAD_CONF /etc/vsftpd/vsftpd_patron.conf
vsftpd: 10.0.1.: DENY
vsftpd: 10.0.72.0/255.255.255.0: setenv VSFTPD_LOAD8CONF
/etc/vsftpd/vsftpd_marketing.conf
vsftpd: .compta.maboite.com: setenv VSFTPD_LOAD8CONF
/etc/vsftpd/vsftpd_compta.conf
```

Vous voyez d'un coup d'œil le principe. Chaque ligne définit une règle, qui précise une cible (une adresse, un domaine), pour laquelle on utilisera un fichier de configuration spécifique. Brillant de simplicité, non ?

Ainsi, la première ligne décide que, pour l'adresse IP 10.0.0.99 (celle du patron), nous utiliserons pour vsftpd le fichier de configuration /etc/vsftpd/vsftpd_patron.conf. Ce fichier de configuration sera spécifique, et pourra contenir toutes les fleurs que vous voudrez faire, comme :

- max_clients= 0 (le patron ouvre autant de connexions simultanées qu'il veut) ;
- max_per_ip= 0 (dans ce cas, puisqu'il n'y a qu'une IP, c'est la même chose) ;
- local_enable=Yes (le patron se connectera avec son compte local) ;
- chroot_local_user= No (le patron n'est pas chrooté, c'est le comportement par défaut de cette directive, mais pourquoi ne pas le rappeler) ;

- write_enable=Yes (le patron peut écrire) ;
- local_max_rate=0 (débit illimité pour le patron) ;
- ftpd_banner=Prenez tout ce qu'il vous faut et n'oubliez pas mon augmentation.

D'autres aménagements sont bien sûr possibles, relisez la liste des options étendues que nous citons en référence et l'inspiration viendra sûrement. La première ligne indiquait une adresse IP entière, le filtrage s'appliquait donc à elle et elle seule. La seconde ligne indique 10.0.1. (notez le point), les trois premiers octets. Dans ce cas, le filtrage s'appliquera à l'ensemble du réseau 10.0.1.0 (donc de 10.0.1.1 à 10.0.1.255).

Tcp_wrappers suppose que la partie manquante prenne toutes les valeurs possibles (ainsi, 10.5. comprendrait les adresses commençant par 10.5, soit de 10.5.0.0 à 10.5.255.255). Ce mode de filtrage est très puissant. Dans cet exemple, nous supposons qu'il s'agit des adresses des stagiaires. Ils sont tout simplement refusés sur le serveur (DENY), bien que nous soyons dans hosts.allow…

La troisième ligne offre une logique similaire, mais avec un adressage classique : 10.0.72.0/255.255.255.0 pourrait être écrit 10.0.72. dans le paragraphe précédent. L'intérêt de cette façon de faire est de filtrer des adresses pour lesquelles le masque est non standard (par exemple 10.0.72.0/255.255.252.0 ou 192.168.0.0/255.255.255.192) : dans ces cas en effet, le filtrage précédent est inadapté. Nous utilisons alors un autre fichier de configuration, vsftpd_marketing.conf, qui pourra être très restrictif, jusqu'à :

```
anonymous_enable=No
local_enable=No
```

Ni les utilisateurs dotés de compte ni les anonymes ne peuvent se connecter, bref, c'est l'équivalent d'un deny… Ce serait peut-être un peu excessif dans la réalité…

La quatrième ligne fonctionne sur une logique différente. Il est en effet possible, et même fréquent, que les machines de votre réseau soient en DHCP, auquel cas le filtrage par IP n'est pas idéal. Tcp_wrapper offre aussi la possibilité de filtrer par le nom. Ainsi, .compta.maboite.com désigne tous les noms finissant par .compta.maboite.com, c'est à dire ordi1.compta.maboite.com, serveur2.compta.maboite.com, etc.

Nous pourrions là aussi indiquer une machine précise (expert.compta.maboite.com), mais nous choisissons de prendre en compte tout le segment (notez le point en début de nom dans .compta.maboite.com).

Pour tous les ordinateurs du département comptabilité, nous utilisons un fichier de configuration nommé vsftpd_compta.conf, et qui contiendra bien sûr les paramètres que vous voulez.

Et les autres ? Ceux qui ne sont pas cités dans hosts.allow se verront appliquer le fichier de configuration standard, /etc/vsftpd/vsftpd.conf, qui pourra à votre choix être très ouvert ou très restrictif.

Il ne vous reste plus qu'à redémarrer les services réseau et vsftpd et l'ensemble est fonctionnel.

Bien sûr, les mauvaises langues diront qu'il serait possible de contourner cette sécurité en spoofant l'adresse ou le nom d'un hôte autorisé : c'est vrai et cette évidence est valable pour tous les filtres basés sur l'adresse ou le nom (un filtrage sur l'adresse Mac pourrait être contourné de la même façon). Ce filtrage n'empêche pas l'existence d'un mot de passe, qui pourra être très restrictif

(comme nous l'avons vu plus haut avec les directives secure_email_list_enable et email_password_file=/etc/vsftpd.email_passwords). Là encore, il est possible de contourner cette protection. Mais un utilisateur non autorisé qui possède à la fois la liste des adresses autorisées et des mots de passe vous conduira sans aucun doute à vous interroger sur la fiabilité globale de votre politique de sécurité…

Troisième configuration : les utilisateurs virtuels

Justement, les utilisateurs sont capables de tout… une protection supplémentaire de vsftpd consiste à faire en sorte que le client qui accède ainsi à votre serveur ait des pouvoirs très limités… l'idée consiste à créer un utilisateur très particulier, un utilisateur virtuel, qui n'aura donc de droits que dans le cadre de vsftpd.

Puisqu'il n'existe pas vraiment sur le système (il n'a pas de mot de passe sur la machine elle-même), il ne pourra faire autre chose que… ce que vous voudrez bien qu'il fasse en tant qu'utilisateur de votre serveur FTP : lire, bien sûr, mais pourquoi pas aussi écrire ou créer des fichiers… dès qu'il sortira des répertoires que vous aurez autorisés pour lui (en tentant par exemple d'écrire ailleurs, d'installer un programme ou de lire autre chose), donc dès qu'il tentera d'échapper au cadre du serveur vsftpd, le système le rejettera en tant qu'utilisateur inconnu !

Il s'agit là d'une protection encore plus efficace…

Elle s'opère en quatre étapes : la création d'une micro base de données qui contient les utilisateurs que vous autoriserez, la liaison de PAM avec cette base, la création d'un utilisateur virtuel, vers lequel sera mappé tout utilisateur autorisé et le paramétrage de vsftpd pour lui donner les droits que vous voudrez.

Création de la base de données

Il s'agit là d'une étape délicate si vous ne maîtrisez pas PAM. Pour faire simple, PAM est le module d'authentification le plus efficace et le plus développé à ce jour sur les systèmes Linux.

Il utilise une base de données qui contient la liste des utilisateurs. Bien sûr, nous pourrions créer un utilisateur et PAM l'intégrerait directement dans sa base de données. Mais justement, nous ne voulons pas que l'utilisateur existe vraiment sur le système !

Nous ne le créons donc que pour PAM et en dehors de la machine elle-même et de ses commandes useradd et autres…

Pour ce faire, créons un fichier, qui contiendra la liste des utilisateurs virtuels que nous voulons ajouter. La règle est : sur la première ligne un login, sur la seconde le mot de passe correspondant, sur la troisième un autre login, sur la quatrième son mot de passe, etc. autant de fois que vous voudrez ajouter d'utilisateurs. Par exemple, créons le fichier virtuels.txt, qui contiendra :

/etc/vsftpd/virtuels.txt

```
Pierre
```

```
Password
Toto
1234
```

Le premier utilisateur est pierre, son mot de passe password. Le second toto, son mot de passe 1234 (je sais, c'est une idée exécrable d'utiliser toto et 1234 : c'est justement pour vous décourager d'utiliser ce fichier tel quel !).

Créons à partir de ce fichier un morceau de base de données utilisable par PAM :

```
USER@MACHINE:~$ cd /etc/vsftpd
USER@MACHINE:~$ sudo db_load -T -t hash -f virtuels.txt
/etc/vsftpd/vsftpd_virtuels.db
```

Cette commande doit être lancée en tant que root, depuis le répertoire dans lequel se trouve votre fichier virtuels.txt. Elle suppose que vous ayez le programme Berkeley db installé.

Sur certains systèmes, comme Debian, il arrive que cette commande échoue, parce que plusieurs versions de Berkeley db sont installées (ne me demandez pas pourquoi... peut-être avez-vous recompilé plusieurs fois et les différentes éditions de ce programme, sans lequel aucune authentification n'est raisonnablement possible, se sont ajoutées sans s'écraser). Il faut lancer la version du programme qu'utilise le PAM installé, en remplaçant db_load par db3_load ou db4_load... Pour plus d'informations sur ce programme, allez sur : http://www.sleepycat.com/docs/utility/index.html.

Dans tous les cas, cette commande crée le fichier /etc/vsftpd_virtuels.db, qui contient la même chose que notre fichier virtuels.txt, mais formaté d'une façon utilisable par PAM.

Une bonne idée est de rendre ce fichier accessible par root et lui seul :

```
USER@MACHINE:~$ sudo chmod 600 /etc/vsftpd_virtuels.db
```

Effacez parallèlement virtuels.txt. Si vous voulez le garder pour conserver la mémoire de ce que vous avez fait, donnez-lui les mêmes droits restrictifs que pour /etc/vsftpd_virtuels.db.

Création du fichier PAM à partir de cette base de données

L'objectif de cette étape consiste à informer PAM qu'il faut utiliser notre base de données avec vsftpd... Créez un fichier vsftpd.pam, que vous positionnerez dans le répertoire /etc/pam.d. Ce fichier contiendra deux lignes, du type :

```
auth required /lib/security/pam_userdb.so db=/etc/vsftpd_virtuels
account required /lib/security/pam_userdb.so db=/etc/vsftpd_virtuels
```

Elles précisent que tant le mot de passe que le nom d'utilisateur doivent être recherchés par PAM dans le fichier, de type base de données, /etc/vsftpd_virtuels (l'extension .db est donc sous-entendue). Puisque ce fichier s'appelle vsftpd.pam et qu'il est positionné dans /etc/pam.d, il sera pris en compte par PAM à chaque gestion des autorisations pour le démon vsftpd.

Création d'un utilisateur virtuel

Jusqu'ici, pierre et toto sont deux utilisateurs réels. Mais nous ne voulons pas qu'ils aient de compte en dehors de vsftpd. Ils n'existeront donc pas sur le système et c'est pourquoi nous avons contourné le processus de création normal pour ces utilisateurs… mais ils ne peuvent pas faire grand-chose à ce stade !

PAM les reconnaît dans vsftp, mais ils n'ont aucun droit sur aucun fichier ni répertoire et ne peuvent même pas interagir avec le système ! Ils pourront cependant se connecter et rien d'autre.

A quoi servent-ils ? En fait, nous allons faire en sorte qu'ils puissent faire quelque chose, parce que nous allons les mapper avec un utilisateur du système créé spécialement pour l'occasion. Nommons-le virtuelftp.

Ce dernier n'aura, lui, aucun mot de passe sur le système et dépendra donc complètement de vsftpd… Vous voyez la boucle ? pierre et toto ont un mot de passe mais n'existent pas sur le système ; virtuelftp existe sur le système, mais, n'ayant pas de mot de passe, ne peut pas se connecter… De la sorte, pierre et Toto peuvent se connecter, mais dès qu'ils dépassent le cadre limité de ce que vous autorisez pour eux en les soudant à virtuelftp, ils se retrouvent soit pierre ou Toto, donc n'existant pas sur le système, soit virtuelftp, donc n'ayant pas de droit de connexion non plus… Un peu compliqué mais diablement sécurisé, n'est-ce pas ?

Créons donc cet utilisateur virtuelftp et précisons que les fichiers du serveur FTP seront dans son répertoire personnel, que nous nommerons siteftp :

```
USER@MACHINE:~$ sudo useradd -d /home/siteftp virtuelftp
USER@MACHINE:~$ ls -ld /home/siteftp
drwx------    3 virtuelftp  virtuelftp      4096 Feb 29 10:34 /home/siteftp
```

Vous pouvez à présent positionner des fichiers dans ce répertoire /home/siteftp, avec par exemple

```
USER@MACHINE:~$ touch /home/siteftp/essai.txt
```

Paramétrage de vsftpd

Il ne reste plus qu'à demander à vsftpd d'utiliser tous ces paramétrages, au travers de /etc/vsftpd/vsftpd.conf. Le fichier pourrait être du type :

/etc/vsftpd/vsftpd.conf

```
anonymous_enable=NO
local_enable=YES
write_enable=NO
anon_upload_enable=NO
anon_mkdir_write_enable=NO
anon_other_write_enable=NO
chroot_local_user=YES
```

```
guest_enable=YES
guest_username=virtuelftp
listen=YES
listen_port=10021
pasv_min_port=30000
pasv_max_port=30999
```

Dans ce fichier, nous interdisons naturellement les utilisateurs anonymes et tous les droits qu'ils pourraient avoir (nous ne voulons que pierre ou toto) et autorisons les utilisateurs locaux (parmi lesquels virtuelftp).

Nous les enfermons dans une prison chroot (dans notre cas, /home/siteftp sera vu comme /) et autorisons les invités, ce qui est fondamental, parce que cette option permet d'activer les utilisateurs virtuels ! Le nom de cet utilisateur, qui figure ligne suivante, sera donc bien virtuelftp. Par ailleurs, mais ce n'est pas obligatoire, nous positionnons notre serveur en mode standalone (Listen=Yes), en écoute sur un port particulier, 10021, au lieu de 21. Il faudra en plus que nos utilisateurs appellent le bon port pour pouvoir se connecter. Enfin, les ports ouverts seront dans la tranche 30000 à 30999, ce qui est utile si vous paramétrez un firewall.

Notre configuration est prête ! Redémarrez le serveur vsftpd et essayez une connexion ! Vous allez voir comment tout ceci fonctionne :

```
USER@MACHINE:~$ ftp localhost 10021
Connected to
 localhost (127.0.0.1).
220 ready, dude (vsFTPd 1.1.0: beat me, break me)
USER@MACHINE:~$ Name (localhost:loulou): toto
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
USER@MACHINE:~$ ftp> pwd
257 „/“
ftp> ls
227 Entering Passive Mode (127,0,0,1,117,135)
150 Here comes the directory listing.
226 Transfer done (but failed to open directory).
ftp> size essai.txt
9 12
ftp>
```

Dans cet exemple, nous nous connectons avec l'utilisateur toto et son mot de passe, le serveur nous dit bien que nous sommes à la racine, alors que l'emplacement véritable est /home/siteftp.

Observez les commandes et le retour du serveur. Le message d'erreur lors du ls est bien normal (Transfer done but failed to open directory), puisque le répertoire n'est pas lisible par tous, ce qui est mieux pour la sécurité (vous pourriez changer cet état de chose, par exemple en ajoutant anon_world_readable_only=NO, mais il n'est pas sûr que ce confort supplémentaire soit vraiment indispensable. Si vos utilisateurs savent ce qu'ils cherchent, pas besoin de leur en donner plus…). En revanche, la commande size nous montre que nous avons bien accès à essai.txt.

Par la suite, chaque fois que vous voudrez rajouter un utilisateur comme pierre ou toto, il vous suffira de refaire les étapes 1 et 2. Si vous voulez assouplir les règles du répertoire d'accueil, vous pouvez modifier le fichier vsftpd.conf en relisant le début de cet article ou directement accroître ses droits locaux (par exemple drwx—x–x pour en autoriser le listing).

Quatrième configuration : utilisateurs virtuels, suite et encore plus de sécurité

La configuration qui précède est très sécurisée, mais elle présente évidemment une limitation : pierre et toto ont exactement les mêmes droits, ils sont rigoureusement identiques du point de vue du serveur.

Il est possible d'améliorer les choses pour rendre notre configuration ultime, en créant plusieurs niveaux de droits, avec par exemple un utilisateur pierre qui pourra lire le contenu d'un répertoire et en télécharger quelques fichiers, et un autre, toto, qui lui aura plus de pouvoir, avec par exemple le droit d'upload en plus...

Ces configurations différentes sont possibles parce que vsftpd dispose d'une option nommée " configurabilité utilisateur par utilisateur " (hou ! l'affreux barbarisme).

Cette troisième configuration consiste donc simplement dans un premier temps à activer cette option, puis, dans un deuxième et troisième temps, à définir les droits de pierre et de toto.

Activation de la configurabilité utilisateur par utilisateur

Pour ce faire, dans le fichier /etc/vsftpd/vsftpd.conf de la configuration précédente, ajoutez la ligne :

```
user_config_dir=/etc/vsftpd_user_conf
```

De la sorte, vous indiquez à vsftpd qu'il existe un répertoire (un répertoire, hein, pas un fichier) **/etc/vsftpd_user_conf** (vous pouvez le nommer différemment si vous le souhaitez), qui contiendra les fichiers de configuration pour vsftpd de chaque utilisateur. Chaque fichier portera le nom de l'utilisateur, et contiendra ses paramètres particuliers, qui seront prioritaires sur les paramètres du fichier de configuration général /etc/vsftpd/vsftpd.conf.

S'il n'existe pas de fichier spécifique à l'utilisateur, seul le fichier général sera utilisé.

Créez le répertoire en question :

```
USER@MACHINE:~$ mkdir /etc/vsftpd_user_conf
```

Nous utilisons ici la configuration précédente pour l'enrichir, mais il va de soi que vous pourriez utiliser cette configurabilité utilisateur par utilisateur avec des utilisateurs " normaux ", connus sur le système, comme dans nos exemples du début de l'article.

Paramétrages pour pierre

Nous voulons que pierre ait des droits en lecture sur tous les répertoires. Dans la configuration précédente, ls produisait un échec.

Corrigeons ce problème pour pierre. Créons donc un fichier /etc/vsftpd_user_conf/pierre, qui contiendra :

/etc/vsftpd_user_conf/pierre

```
anon_world_readable_only=NO
```

De la sorte, nous autorisons le ls même si le répertoire n'est pas 'r' pour les autres (il ne contient pas r dans la position ——r–).

Redémarrez le serveur et connectez-vous en tant que pierre, vous verrez que ls fonctionne, tandis qu'il produit un échec à ce stade si vous êtes toto !

Affinage des droits de toto

Nous voulons que toto puisse lui aussi avoir ls, mais aussi créer ou uploader, sans bien sûr toucher aux fichiers déjà existants. Créez pour ce faire un fichier /etc/vsftpd_user_conf/toto qui contiendra :

/etc/vsftpd_user_conf/toto

```
anon_world_readable_only=NO
anon_upload_enable=YES
write_enable=YES
```

La première directive lui permet le ls, la deuxième et le troisième l'upload vers le serveur. toto ne peut en revanche pas supprimer de fichier ni créer de répertoire.

Testez ces configurations, vous verrez combien elles sont efficaces !

Vous voyez qu'au fond, passer à ce stade ultime n'est pas beaucoup plus compliqué, puisqu'il s'agit au fond simplement, une fois la directive de configurabilité utilisateur par utilisateur activée, de créer un fichier de configuration cousu main pour chaque utilisateur particulier du système.

Cinquième configuration : les IP virtuelles

Mais bon, vous n'êtes peut-être pas paranoïaque à ce point et votre souci est peut-être plus de disposer d'un serveur à tout faire à peu près bien rangé, qui sépare bien le LAN de l'Internet par exemple, que de faire du tuning utilisateur par utilisateur. Imaginons donc un dernier cas.

Dans ce scénario, il ne s'agit donc plus de répondre à un besoin de filtrage serré, mais d'une configuration qui pose des problèmes différents.

Votre serveur FTP doit répondre à des demandes multiples : il sert de source pour les drivers que vous mettez à disposition du grand public, mais aussi de serveur pour que les employés puissent récupérer les formulaires communs à l'entreprise (demandes de congés et autres documents préformatés).

Il sert enfin à abriter les fichiers d'un projet de développement pour la cellule de veille…

Vous n'avez bien sûr pas envie que tout ce petit monde se mélange, que le grand public récupère vos formulaires ou que quiconque ait accès aux fichiers stratégiques de votre cellule de veille.

Votre consultant en sécurité vous dit que vous feriez mieux d'installer des serveurs différents et IL A RAISON.

Mais vous n'avez pas le budget, pas l'organisation qu'il faut, etc., bref, vous êtes cantonné à une seule machine.

Ce n'est pas grave (en fait si, mais disons plutôt que, dans cette situation désagréable, voyons comment faire en sorte de limiter les cauchemars de vos nuits trop courtes) !

Vsftpd offre une solution pour vous : les IP virtuelles.

L'idée de ce mécanisme consiste à attribuer plusieurs adresses IP à votre carte réseau (des alias) et de consacrer chaque IP à l'un des trois sites FTP que vous voulez maintenir.

Création d'alias réseau

Pour quoi faire ? Vous allez voir. Supposons que votre carte physique ait pour adresse 10.0.0.10 et qu'il s'agisse de la carte eth0.

La façon la plus simple de créer une adresse IP virtuelle consiste à copier le fichier /etc/sysconfig/network-scripts/ifcfg-eth0, qui est votre fichier de configuration de la carte eth0, vers /etc/sysconfig/network-scripts/ifcfg-eth0:1 (cp /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/sysconfig/network-scripts/ifcfg-eth0:1). Éditez ensuite ifcfg-eth0:1.

Il devrait contenir des lignes du type :

ifcfg-eth0:1

```
DEVICE=eth0
BOOTPROTO=none
BROADCAST=10.255.255.255
IPADDR=10.0.0.10
NETMASK=255.0.0.0
NETWORK=10.0.0.0
ONBOOT=yes
TYPE=Ethernet
USERCTL=yes
PEERDNS=yes
GATEWAY=10.199.0.254
```

```
IPV6INIT=no
```

Les adresses ne seront sans doute pas les mêmes, laissez-les telles qu'elles se présentent dans votre fichier.

L'opération ne consiste qu'à modifier les lignes DEVICE et IPADDR, qui deviendront :

```
DEVICE=eth0:1
```

et

```
IPADDR=10.0.0.11
```

Enregistrez vos modifications et redémarrez le réseau

```
USER@MACHINE:~$ sudo service network restart
```

Votre carte réseau dispose désormais de deux adresses, 10.0.0.10 et 10.0.0.11, ce que vous pouvez vérifier avec la commande ifconfig (si les deux cartes réseau n'apparaissent pas, vérifiez vos fichiers, une erreur s'est sûrement glissée lors de votre modification).

De la même façon, nous créons un deuxième alias eth0:2, qui aura l'adresse 10.0.0.12. Notre carte réseau a désormais trois adresses.

Écoute d'une adresse particulière

Occupons-nous à présent du serveur vsftpd et voyons ce que ce merveilleux outil permet de faire avec cette manipulation.

Il s'agit pour lui de créer trois fichiers de configuration différents, un pour chaque utilisation, et de lancer trois instances de vsftpd ! Lourd ?

Mais non… certes trois démons écoutent, mais la charge totale des utilisateurs reste la même qu'il y ait un processus ou trois et le surplus est faible en termes d'utilisation de la RAM et du processeur… essayez un top et vous verrez les démons alterner et un ps -aux vous montrera combien la différence est faible avec un seul démon gérant tous les utilisateurs.

En ce qui concerne les trois configurations, nous vous laissons décider, en fonction des paramètres qui précèdent, quels paramètres vous voudrez positionner dans chacun des fichiers.

Nous nommerons le fichier de configuration tout venant, à destination des utilisateurs qui téléchargent des drivers, /etc/vsftpd/vsftpd.conf.

De la sorte, en l'absence de directive précise, l'utilisateur lambda utilisera ce fichier. Il devra contenir en revanche obligatoirement les lignes suivantes :

```
listen_address = 10.0.0.10
```

De cette façon, les connexions sur cette IP (l'IP vers laquelle vous renvoyez les utilisateurs du grand

public) utiliseront ce fichier.

Les employés, eux, connaissent l'IP 10.0.0.11 et vous pouvez même prendre la précaution de leur dire que le port n'est pas 21, mais par exemple 10021…

Créez alors un second fichier, par exemple /etc/vsftpd/employes.conf, qui contiendra les paramètres que vous souhaitez pour eux, mais aussi les lignes :

/etc/vsftpd/employes.conf

```
listen_address = 10.0.0.11
listen_port = 10021
```

Enfin, le groupe d'étude qui partage des fichiers accédera à la machine par l'adresse 10.0.0.12, sur le port que vous voulez, par exemple 10021 également (mais tout autre port est possible).

Créez alors un fichier /etc/vsftpd/groupe.conf, qui contiendra les paramètres que vous souhaitez, mais également :

/etc/vsftpd/groupe.conf

```
listen_address = 10.0.0.12
listen_port = 10021
```

Il ne reste plus qu'à lancer trois instances de vsftpd, qui chacune écoutera une des adresses ! Encore une fois, ce processus alourdit un peu le serveur, mais pas d'une façon énorme, puisque chaque instance occupera surtout les ressources requises par chaque utilisateur connecté…

```
USER@MACHINE:~$ vsftpd /etc/vsftpd/vsftpd.conf &
vsftpd /etc/vsftpd/employes.conf &
vsftpd /etc/vsftpd/groupe.conf &
```

Bien sûr, vous aurez intérêt à positionner ces différents utilisateurs dans des répertoires différents, avec les directives local_root et anon_root que nous avons évoquées au début de l'article.

Par ailleurs, cette répartition par adresse IP virtuelle peut se combiner avec les autres configurations vues dans ces exemples.

Ce dernier scénario est loin des exemples extrêmes qui précèdent, puisqu'il donne l'impression d'être facile à contourner.

Un utilisateur disposant de la bonne adresse (10.0.0.12 par exemple) pourra tenter une connexion vers le répertoire du groupe… encore faut-il qu'il ait le bon login et mot de passe… comme plus haut, s'il a toutes les informations et qu'il n'est pas supposé les avoir, il faudrait sans doute remettre à plat la politique globale de sécurité.

Conclusion

Nous avons essayé dans cet article de vous proposer plusieurs configurations qui vous permettront, en les combinant, de créer votre propre serveur. Un bon moyen de les combiner consiste à les essayer sur un serveur de test.

Pendant quelques temps, chaque fois qu'un droit particulier doit être donné, cependant que vous agirez sur le vrai serveur au niveau utilisateur et répertoire, demandez-vous comment il serait possible, sur le serveur de test, d'aboutir au même résultat en modifiant le fichier de configuration ou en créant une configuration spécifique. Vous constaterez rapidement que la forêt d'options disponibles pour vsftpd et sa capacité à accepter plusieurs fichiers de configuration différents lui donnent une puissance et une souplesse qui lui font mériter sa place en tête du hit parade des serveurs double S (simple mais sûr)…

Voir aussi

- **(fr)** Article
- **(en)** Article

Basé sur « Article » par Auteur.

From:
http://doc.wikis.frapp.fr/ - **doc**

Permanent link:
http://doc.wikis.frapp.fr/doku.php?id=logiciel:reseau:ftp:serveur:vsftpd:exemples:start

Last update: **2024/09/21 13:41**