28 août 2008

Quatre serveurs FTP hyper sécurisés avec vsftpd

Catégorie : Sécurité Tags : lmhs

~~Retrouvez cet article dans : Linux Magazine Hors série 21~~

Vous utilisez un serveur FTP et craignez autant les failles des démons classiques que les abus des utilisateurs ? Découvrez vsftpd, un serveur dont la devise « beat me, break me » vous permettra enfin de dormir tranquille...

Introduction

vsftpd signifie Very Secure File Transfert Protocol Daemon. Il s'agit d'un serveur FTP, dont l'objectif est d'être à la fois simple et très sécurisé. Sans entrer dans les détails, sa structure même le rend très robuste, et peu vulnérable à la plupart des failles de configuration standards qui existent sur les serveurs dont l'écriture est plus « classique ».
Pourquoi ? Parce que son code limite considérablement tous les paramètres possibles, qu'il vous appartiendra d'ouvrir en fonction de vos besoins.
Il dispose par ailleurs d'une grande quantité d'options permettant un paramétrage très fin de ce que vous voulez autoriser et refuser sur votre serveur, jusqu'à un filtrage utilisateur par utilisateur si nécessaire.
Dans cet article, nous vous proposons, après avoir passé en revue les principes d'utilisation de ce programme, d'examiner quatre scénarii extrêmes qui vous permettront de comprendre jusqu'à quel point cet outil est capable de pousser la sécurisation des accès sur votre machine : jusqu'au point où les droits sont tellement contrôlés que c'est à peine si les utilisateurs existent encore...

Installation standalone ou xinetd

L'installation du programme ne pose pas de problème, elle se fait via rpm, tar.gz ou autre, en téléchargeant le programme depuis le site vsftpd lui-même (http://vsftpd.beasts.org), avec votre commande favorite de décompression/installation, comme ~~tar -zxvf, ./configure, make & make install~~.
À ce stade, le serveur est installé et prêt à fonctionner. Vous pourriez vous contenter de cette étape, mais un fonctionnement optimal suppose la compréhension de « ce » qui est installé et la mise en place de vos propres paramètres, c'est pourquoi cet article comporte encore quelques paragraphes !

Comme vous le savez peut-être, sous Linux, les services réseau standards peuvent soit être installés individuellement, c'est le mode « standalone », soit être regroupés sous un seul démon, xinetd. Traditionnellement, l'administrateur installe un serveur en mode standalone quand il s'agit d'un service à part entière (c'est le cas si, par exemple pour FTP, vous souhaitez mettre en place un serveur qui sera destiné à assurer la fonction de partage de fichiers de façon permanente ou fréquente : c'est-à-dire dans la plupart des cas) et ne positionne sous le contrôle de xinetd que les services dont l'utilisation est très occasionnelle (c'est le cas si votre serveur FTP n'est appelé à fonctionner que très occasionnellement et si vous ne voulez pas avoir à penser à le démarrer dans ces cas-là).
Dans tous les cas, le fichier de configuration du serveur est ~~/etc/vsftpd/vsftpd.conf~~.
Dans un premier temps, positionnons-le en mode standalone. Pour ce faire, vérifiez que dans vsftpd.conf la directive ~~listen~~ est sur :

```
listen = YES
```

Démarrez alors simplement le serveur avec :

```
/usr/sbin/vsftpd &
```

ou, sous Redhat/Fedora :

```
service vsftpd start
```

Pour le tester, connectez-vous en ligne de commande sur votre serveur, selon une procédure du type :

```
[root@localhost]ftp localhost
Connected to localhost.localdomain.
220 (vsFTPd 1.2.1)
Name (localhost:root): Test (évidemment, vous utiliserez ici un nom d'utilisateur disposant
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
150 Here comes the directory listing.
-rw-r--r-- 1 1001 1001 17421 Dec 19 14:02 fichier1.txt
drwx------ 2 1001 1001 4096 Dec 24 15:54 repertoire
226 Directory send OK.
ftp> quit
221 Goodbye.
[root@localhost]
```

Si le résultat est proche de celui qui est décrit ici, alors votre serveur fonctionne ! Si ce n'est pas le cas, lisez la suite de l'article pour comprendre où votre configuration pêche. Vous pouvez aussi recharger le programme avec un fichier ~~vsftpd.conf~~ par défaut, et recommencer la manipulation.
Si vous souhaitez utiliser xinetd, il faudra alors, dans le répertoire ~~/etc/xinetd.d/~~, qu'un fichier ~~vsftpd~~ existe, qui permettra à xinetd de gérer le lancement de vsftpd si des requêtes FTP se présentent.
Ce fichier devra contenir des lignes du type :

```
default: off
service ftp
{
disable = no
socket_type = stream
wait = no
user = root
server = /usr/sbin/vsftpd
nice = 10
}
```

Les paramètres importants ici sont ~~default~~ (sur ~~off~~, le serveur sera la plupart du temps arrêté, et ne sera lancé qu'en cas de besoin, sur on il sera lancé systématiquement, mais dans ce cas pourquoi ne pas utiliser le mode standalone ? Gardez ~~default~~ sur ~~off~~ en général et lancez manuellement le service) et ~~disable~~ (positionnez à

no, signifiant que le serveur peut fonctionner ; en mettant yes, vous désactiveriez la fonction ftpd de xinetd). Éditez ensuite le fichier /etc/vsftpd/vsftpd.conf et passez la directive listen sur :

```
listen = NO
```

Redémarrez alors xinetd :

```
service xinetd restart
```

Le serveur est normalement prêt à fonctionner. Testez ensuite en ligne de commande votre connexion serveur afin de vérifier son bon fonctionnement, comme décrit au paragraphe précédent.

Les paramètres les plus fréquents

Le fichier /etc/vsftpd/vsftpd.conf contient une liste de paramètres, qui vont modifier le comportement par défaut du serveur. Tout est là. En activant ou modifiant un paramètre de ce fichier, vous configurez vsftpd pour un comportement particulier. Après chaque modification, pensez à redémarrer le serveur, en tapant :

```
service vsftpd restart
```

ou

```
/usr/sbin/vsftpd &
```

Puis testez le bon fonctionnement du paramètre que vous avez modifié.
Commençons par examiner les plus fréquents d'entre eux, c'est-à-dire ceux que vous aurez sans aucun doute envie d'adapter à la configuration particulière de votre réseau :

```
anonymous_enable=NO/YES
```

Comme son nom le laisse entendre, cette fonction permet d'activer ou de désactiver le mode anonyme. S'il est activé (YES), les connexions anonymes sur le serveur sont autorisées. S'il est désactivé (NO), seuls les utilisateurs connus, c'est-à-dire disposant d'un compte utilisateur sur la machine, seront autorisés à accéder au serveur.

```
local_enable=NO/YES
```

Cette directive interdit ou autorise l'accès au serveur aux utilisateurs connus de la machine (c'est-à-dire disposant d'un compte, login/password). Dans la plupart des configurations, cette ligne sera sur YES, sauf si votre serveur est public et que vous ne voulez pas accepter les connexions d'utilisateurs dotés de compte.

```
write_enable=NO/YES
```

Permet d'écrire sur le serveur (upload). Il est là encore fréquent de mettre YES, mais en vérifiant bien sûr que les fichiers téléchargés vers votre serveur ne soient pas à terme trop volumineux et qu'ils correspondent à ce qui est supposé pouvoir s'y trouver.
Vous mettrez NO si vous ne voulez pas qu'un autre que vous-même dépose des fichiers sur le serveur.

```
local_umask=022
```

Il s'agit du masque par défaut lors du positionnement de fichiers sur le serveur, dans le cas où vous auriez mis la directive précédente sur YES. Rappelons que le masque est l'inverse des droits. Ainsi, un masque 022 conduira à écrire des fichiers avec un droit 755. Si vous ne précisez pas ce masque, celui de vsftpd sera utilisé par défaut, qui est 077, laissant donc un droit 700 aux fichiers écrits.

```
xferlog_enable=NO/YES
```

Cette option permet de loguer les uploads et downloads.

```
ftpd_banner=Bienvenue sur mon serveur
```

Cette directive est loin d'être essentielle, mais elle est confortable. Elle permet de générer un message d'accueil pour les clients qui se connectent sur votre serveur.

```
chroot_list_enable=NO/YES
chroot_list_file=/etc/vsftpd.chroot_list
```

Ces deux directives ne sont pas indispensables, mais elles sont très utiles. La première permet de demander au serveur de rechercher un fichier ~~vsftpd.chroot_list~~ dont le chemin est indiqué à la seconde ligne. Dans ce cas, tous les utilisateurs présents dans la liste que représente le fichier ~~vsftpd.chroot_list~~ seront automatiquement chrootés, c'est-à-dire cantonnés dans leur répertoire personnel. C'est une procédure très utile pour empêcher vos utilisateurs fouineurs de sortir de leur répertoire. Si la première ligne est sur ~~NO~~, cherchez si le fichier contient ~~chroot_local_user~~ (voir ci-dessous), auquel cas le sens donné à la combinaison est différent. Si la première ligne est sur No et que ~~vsftpd.conf~~ ne contient pas de directive ~~chroot_local_user~~ (ou si elle est sur ~~No~~), alors ~~chroot_file_list~~ ne sert à rien.

```
chroot_local_user=NO/YES
```

En activant cette directive, les utilisateurs qui possèdent un compte local seront cantonnés dans leur répertoire personnel après connexion. En revanche, combinée aux deux lignes chroot_list_enable et chroot_list_file, cette directive en modifie le sens. Avec ~~chroot_list_enable=YES~~, ~~chroot_list_file=/etc/vfstpd.chroot_list~~, ~~chroot_local_user=NO~~, le fichier ~~vsftpd.chroot_list~~ contient la liste des utilisateurs qui seront cantonnés dans leur répertoire personnel.
Mais avec ~~chroot_list_enable=YES~~, ~~chroot_list_file=/etc/vfstpd.chroot_list~~, ~~chroot_local_user=YES~~, le fichier ~~vsftpd.chroot_list~~ contient alors la liste des utilisateurs qui ne seront pas cantonnés à leur répertoire personnel, c'est-à-dire qui pourront aller dans d'autres répertoires, tandis que tous les autres utilisateurs (ceux qui ne sont pas cités dans la liste) seront à ce moment, eux, cantonnés à leur répertoire personnel.

```
userlist_enable=NO/YES
userlist_deny=NO/YES
```

Là encore, les deux directives ne sont pas indispensables, mais très utiles. En positionnant la première sur ~~YES~~, vsftpd cherchera un fichier nommé ~~vsftpd.ftpusers~~ ou ~~vsftpd.user_list~~ (vous pouvez donner l'un des deux noms au choix),et trouvera dans ce fichier la liste des utilisateurs qui sont autorisés sur le serveur.
En revanche, si à la première directive vous ajoutez la seconde (~~userlist_deny=YES~~), alors la liste citée ci-dessus devient celle des utilisateurs qui, au contraire, ne sont pas autorisés sur le serveur. Typiquement, ces deux directives sont couramment utilisées pour interdire les tentatives de connexions avec les comptes utilisateurs système de votre machine (ainsi, personne ne pourra se connecter en tant qu'utilisateur root, Apache, etc., si vous combinez les deux directives).
Il y a bien d'autres directives, mais notre objectif n'est pas de les examiner toutes.
Nous voulons vous donner les principales à titre d'outil pour vous permettre de comprendre le fonctionnement de l'ensemble.
Vous trouverez en référence un pointeur vers la liste de toutes les options possibles.

Une configuration classique de vsftpd

Voyons à présent ce qui fait le sel de cet outil, c'est-à-dire comment il est possible de combiner les directives et même les fichiers de configuration, au travers de quelques exemples de configuration simples ou avancés.
Supposons dans un premier temps que vous souhaitiez simplement mettre en place un serveur de fichiers, accessible à tous (par exemple à l'Internet).
Vous souhaiterez sans doute privilégier l'efficacité des transactions et la sécurité. Votre fichier ~~vsftpd.conf~~ pourrait alors ressembler à celui-ci :

```
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 (c'est le cas par exemple lors de l'utilisation d'un download accelerator). 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.

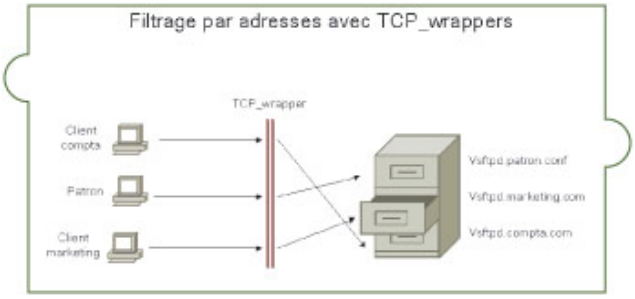
```
anon_max_rate=50000
```

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

Sécurité affinée, quatre scenarii

L'exemple qui précède donne une configuration fonctionnelle, même si bien d'autres paramètres pourraient être ajoutés afin d'accroître le confort que vous pouvez vouloir donner à votre serveur. Vsftpd est particulièrement construit pour permettre une grande finesse dans le filtrage des autorisations d'accès. Donnons-lui sa pleine mesure au travers de trois scenarii imaginés par l'équipe de développement et de sécurisation de vsftpd, qui, s'ils ne collent pas forcément précisément à votre besoin, vous donneront assez d'éléments pour vous permettre de bâtir la configuration qui vous correspondra le mieux, et qui donnera exactement les droits que vous voulez aux utilisateurs que vous voulez, avec un raffinement dans la précision de la gestion qui, nul doute, vous ravira !

Première 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.

1. 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
```

2. 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 :

```
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 :

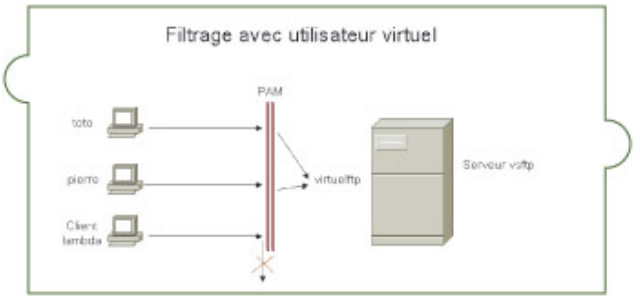
- max_clients= 0 (le patron ouvre autant de connexions simultanées qu'il veut) ;
- max_per_ip= 0 (dans ce cas, puisqu'il n'y a qu'une IP, c'est la même chose) ;
- local_enable=Yes (le patron se connectera avec son compte local) ;
- chroot_local_user= No (le patron n'est pas chrooté, c'est le comportement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

Deuxième 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.

1. 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 :

```
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 :

```
db_load -T -t hash -f virtuels.txt /etc/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 :

```
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.

2.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.

3. 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~~ :

```
useradd -d /home/siteftp virtuelftp
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 touch ~~/home/siteftp/essai.txt~~...

4. 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 :

```
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
```

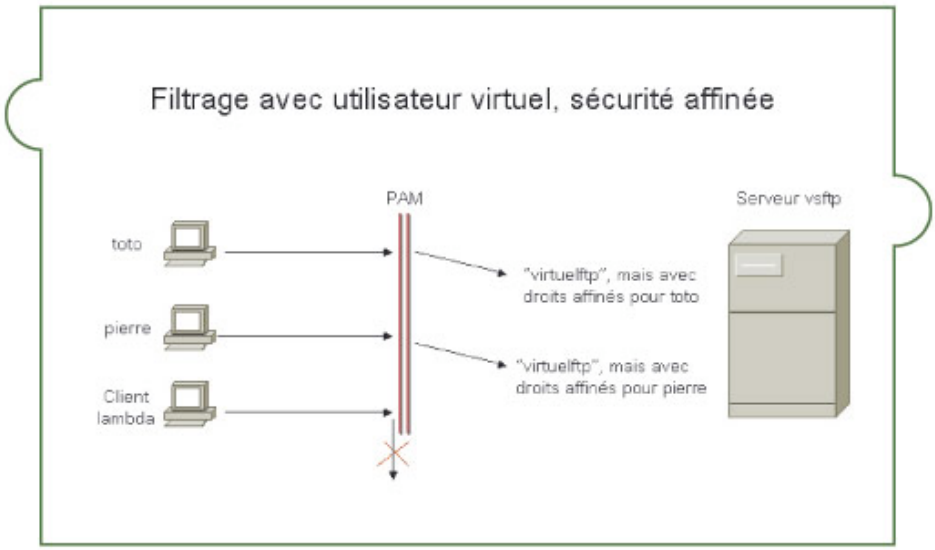
Dans ce fichier, nous interdisons naturellement les utilisateurs anonymes et tous les droits qu'ils pourraient avoir (nous ne voulons que ~~pierre~~ ou ~~toto~~) et autorisons les utilisateurs locaux (parmi lesquels virtuelftp). Nous les enfermons dans une prison ~~chroot~~ (dans notre cas, ~~/home/siteftp~~ 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 :

```
ftp localhost 10021
Connected to
 localhost (127.0.0.1).
220 ready, dude (vsFTPd 1.1.0: beat me, break me)
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.
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).

Troisième 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~~.

1. 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 :

```
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.

2. 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 :

```
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 !

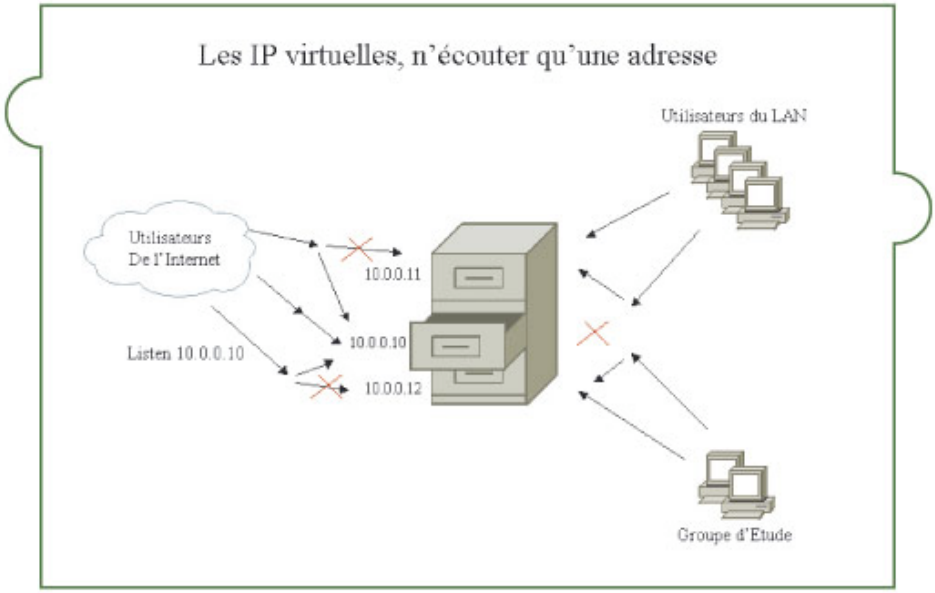
3. Affinage des droits de toto

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 :

```
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.

Quatrième 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.

1. 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 :

```
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 (~~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.

2. 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 :

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

Enfin, le groupe d'étude qui partage des fichiers accèdera 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 :

```
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...

```
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)...

Références

- Le site de référence : http://vsftpd.beasts.org/
- Un site d'astuces : http://www.vsftpdrocks.org/faq/
- Toutes les directives traduites en français : http://www.walkonthegrass.net/tutoriels/Linux/vsftpd
- Les mêmes dans la langue de Shakespeare : http://vsftpd.beasts.org/vsftpd_conf.html

~~Retrouvez cet article dans : Linux Magazine Hors série 21~~

Posté par (La rédaction) | Signature : Jérôme Henry | Article paru dans 

Laissez une réponse

Vous devez avoir ouvert une session pour écrire un commentaire.

« Précédent Aller au contenu »

Identifiez-vous
Inscription
S'abonner à UNIX Garden

- **Articles de 1ère page**

- La protection du secret : approche juridique
- Mieux connaître OOo Draw : les objets 3D et leur espace
- Calc et les variables
- Noël 94 : le cas Mitnick-Shimomura ou comment le cyber-criminel a souhaité joyeux Noël au samurai
- Mettre en place une politique SSI : des recettes pratiques
- Extensions de Firefox : notre sélection
- Konversation : pour discuter librement sur IRC
- BitTorrent : l'autre façon d'échanger des fichiers
- Au-delà de Diffie-Hellman … ?
- Envy : l'installation facile des drivers graphiques dernier cri pour ATI et Nvidia

Actuellement en kiosque :

Il y a actuellement

743 articles/billets en ligne.

Recherche

Catégories

- Administration réseau
- Administration système
- Agenda-Interview
- Audio-vidéo
- Bureautique
- Comprendre
- Distribution
- Embarqué

- Environnement de bureau
- Graphisme
- Jeux
- Matériel
- News
- Programmation
- Réfléchir
- Sécurité
- Utilitaires
- Web

Archives

- septembre 2008
- août 2008
- juillet 2008
- juin 2008
- mai 2008
- avril 2008
- mars 2008
- février 2008
- janvier 2008
- décembre 2007
- novembre 2007
- février 2007

GNU/Linux Magazine

- GNU/Linux Magazine 108 - Septembre 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine 108
- GNU/Linux Magazine HS 38 - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : GNU/Linux Magazine HS 38
- GNU/Linux Magazine 107 - Juillet/Août 2008 - Chez votre marchand de journaux

GNU/Linux Pratique

- Linux Pratique N°49 -Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique N°49
- À télécharger : Les fichiers du Cahier Web de Linux Pratique n°49
- Linux Pratique Essentiel N°3 - Août/Septembre 2008 - Chez votre marchand de journaux
- Edito : Linux Pratique Essentiel N°3

MISC Magazine

- Misc 39 : Fuzzing - Injectez des données et trouvez les failles cachées - Septembre/Octobre 2008 - Chez votre marchand de journaux
- Edito : Misc 39
- MISC 39 - Communiqué de presse
- Salon Infosecurity & Storage expo - 19 et 20 novembre 2008.
- Misc 38 : Codes Malicieux, quoi de neuf ? - Juillet/Août 2008 - Chez votre marchand de journaux