FTP

Pourquoi s'intéresser à FTP ?

Le protocole FTP est probablement le protocole applicatif le plus complexe, et aussi celui qui est le moins utilisé au maximum de ses capacités.

Il est intéressant d'étudier FTP au moins pour les raisons suivantes :

- C'est le protocole le plus sûr pour faire du téléchargement de fichiers, même si cette opération peut aussi être réalisée avec HTTP,
- ce protocole peut poser problème lorsqu'on l'utilise à travers un filtre de paquets (firewall) en qualité de client, et encore plus en qualité de serveur,
- si l'on souhaite placer un serveur FTP accessible à travers un firewall à filtre de paquets, en faisant du "port forwarding" sur le firewall, il devient impératif de bien comprendre le processus, faute de quoi, les transferts FTP resteront probablement bloqués à un moment où à un autre.

Que peut-on faire avec FTP ?

Parmi les applications les plus fréquentes :

- téléchargement de fichiers depuis un serveur vers le client (Download),
- téléchargement de fichiers depuis le client vers un serveur (Upload), par exemple pour la mise à jour des pages web personnelles.

Mais il est possible de faire d'autres choses encore :

- bien que cette méthode soit de moins en moins utilisée, FTP peut servir à envoyer un document à imprimer sur une imprimante distante, l'imprimante faisant alors office de serveur,
- un client FTP peut effectuer des transferts de fichiers entre deux serveurs distants. Bien que cette possibilité soit peu intéressante pour l'internaute moyen, ça reste une fonctionnalité importante pour les administrateurs de sites distants.

Plan du chapitre

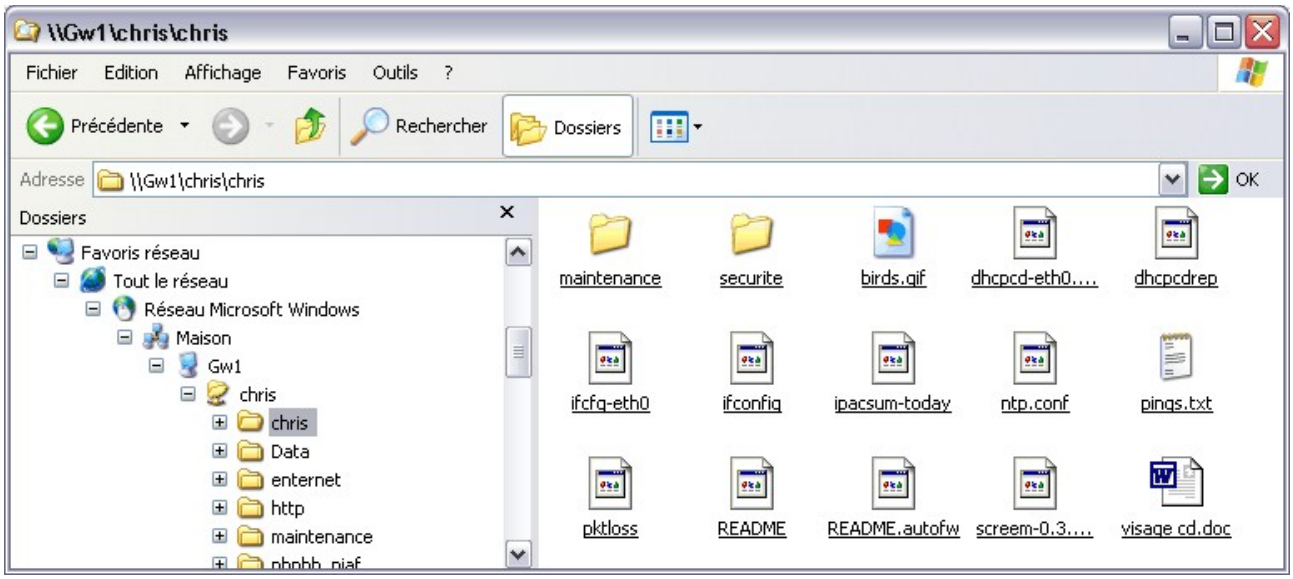
Pourquoi FTP ?

Transfert de fichiers sur un réseau

Il est peut-être bon de faire une mise au point préalable, à propos du transfert de fichiers sur un réseau. Ceux qui sont familiers avec les LAN le savent bien, rien de plus simple en effet que de déplacer des fichiers d'une machine à l'autre, tous les systèmes d'exploitation réseau proposent un moyen de le faire simplement.

Les réseaux Microsoft

Le voisinage réseau est fait pour ça. Il suffit, sur un poste du réseau, de définir le partage d'un répertoire, pour que ce dernier apparaisse dans le voisinage réseau. Si l'on possède les droits en lecture et écriture, il sera alors très simple d'utiliser ce répertoire distant, comme on le ferait avec un répertoire local.



Le chemin **\\gw1\chris\chris** est appelé un chemin UNC (Universal Naming Convention). Comprenons nous bien, cette convention n'a de réalité universelle que dans le monde Microsoft, c'est à dire qu'elle n'a absolument rien d'universel, chacun sait que le monde de l'informatique ne se réduit pas au monde Microsoft...

Pour fonctionner, le voisinage réseau utilise un protocole propriétaire, nommé SMB (Server Message Block), lui même s'appuyant sur NetBIOS (NETwork Basic Input-Output System).

Les réseaux Unix

Sun a développé NFS (Network File System), qui permet de monter un répertoire distant dans l'arborescence locale. Le résultat est sensiblement le même qu'avec le voisinage réseau de Microsoft, et tout aussi "propriétaire".

Et les autres...

Et chaque concepteur de NOS (Network Operating System) y est allé de sa recette "maison".

Tous ces systèmes propriétaires fonctionnent, mais ne savent pas communiquer entre eux. Sur un réseau hétérogène, c'est à dire un réseau où cohabitent des machines qui ne fonctionnent pas toutes avec le même genre de NOS, il devient difficile, voire impossible, de réaliser des transferts de fichiers.

Il est certes possible d'installer des "passerelles", comme par exemple SAMBA sur GNU/Linux, qui permet une interopérabilité entre ces systèmes et les réseaux Microsoft. De la même manière, il existe des services Unix qui peuvent s'installer sur Windows pour permettre de "voir" les partages Unix sur un réseau Microsoft. Ces solutions restent tout de même basées sur des protocoles propriétaires.

File Transfert Protocol

FTP, qui est un protocole ouvert, peut être exploité sur tout système disposant d'une pile IP. Il devient donc possible de réaliser des transferts, sans se préoccuper du NOS de chacune des machines.

Contrairement à ce qu'il peut paraître, FTP est un protocole très complexe, capable de beaucoup plus de choses qu'un simple téléchargement depuis un lien sur une page web. Les clients FTP fournis avec les navigateurs sont souvent minimalistes et n'exploitent qu'une infime partie des possibilités de FTP.

Les bases de FTP

Le cas le plus "classique"

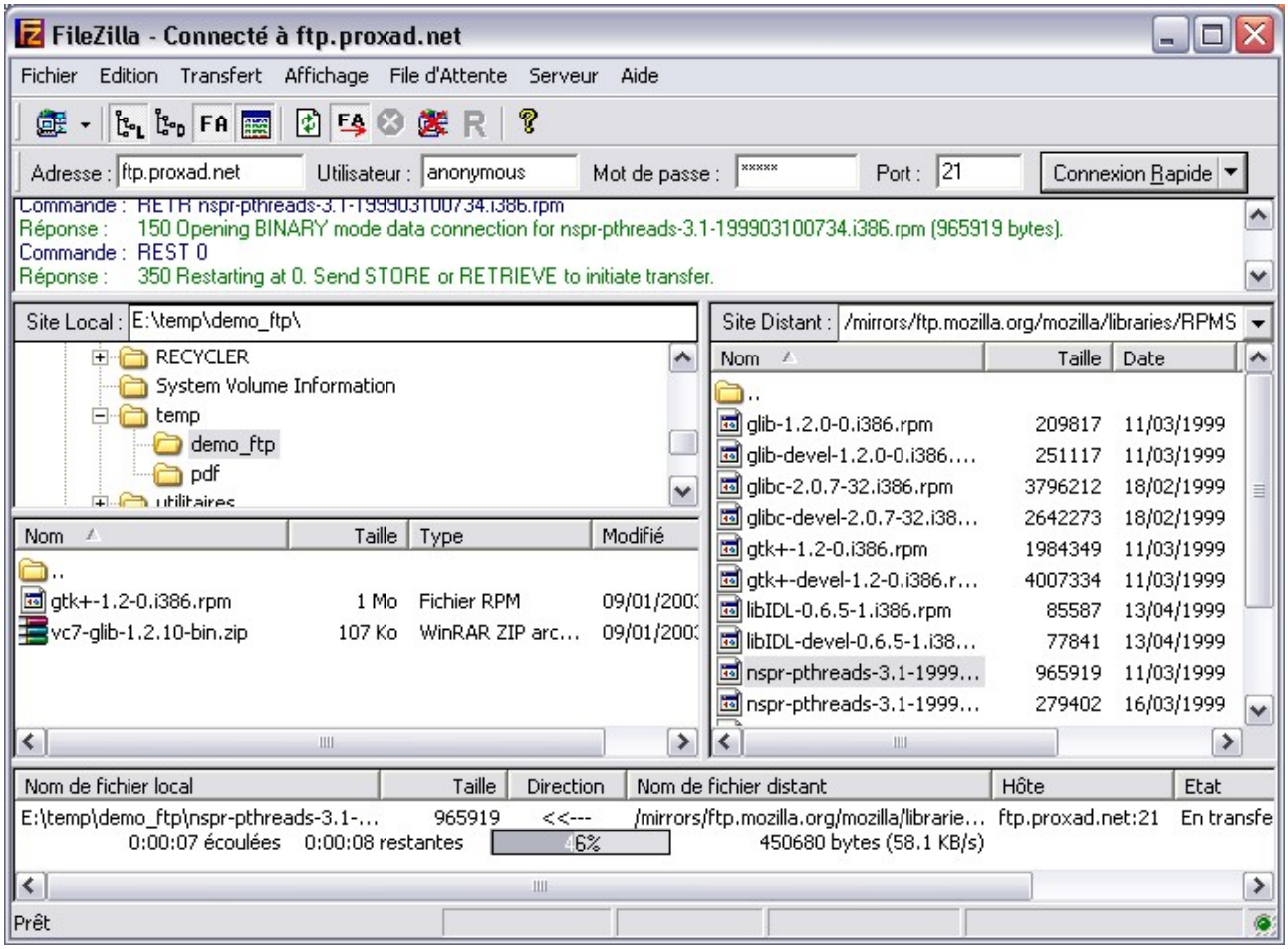
Un client (utilisateur FTP) va se servir de ce protocole, pour faire du transfert de fichiers (upload ou download) avec un serveur (serveur FTP).

Il existe une multitude de logiciels clients en mode graphique pour réaliser ces opérations. Passons sur les fonctions FTP implémentées dans les navigateurs web (Internet Explorer, Mozilla...), qui ne sont pas toujours très performantes.

Clients Windows

Vous serez probablement surpris de constater le nombre impressionnant de clients FTP disponibles, le plus souvent en "shareware" (Windows oblige). Il en existe cependant au moins un sous licence GPL : Filezilla[1].

En plus d'être sous licence GPL, ce logiciel est également localisé en français, ce qui ne gâte rien. Je vous le conseille absolument. Voici l'allure de sa fenêtre de travail dans sa version 2.1.3a :

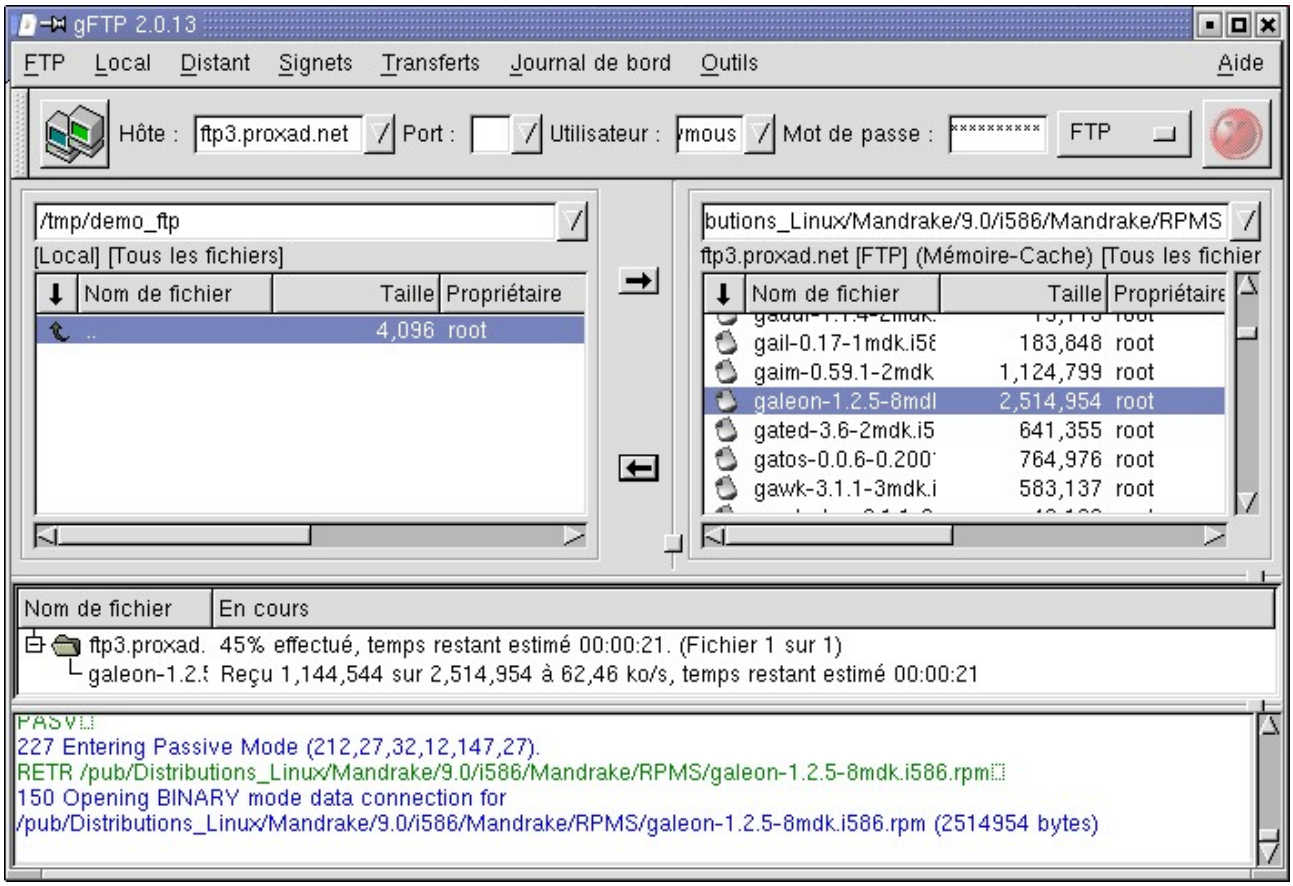


1 FileZilla : http://filezilla.sourceforge.net/

Dans la suite, nous aurons également l'occasion d'utiliser un "shareware" : FlashFXP qui sait faire des choses que FileZilla ne sait réaliser.

Clients GNU/Linux

Ici, le choix est peut-être plus restreint. Le plus abouti (parmi ceux que je connais) est probablement gftp[2], naturellement sous licence GPL :



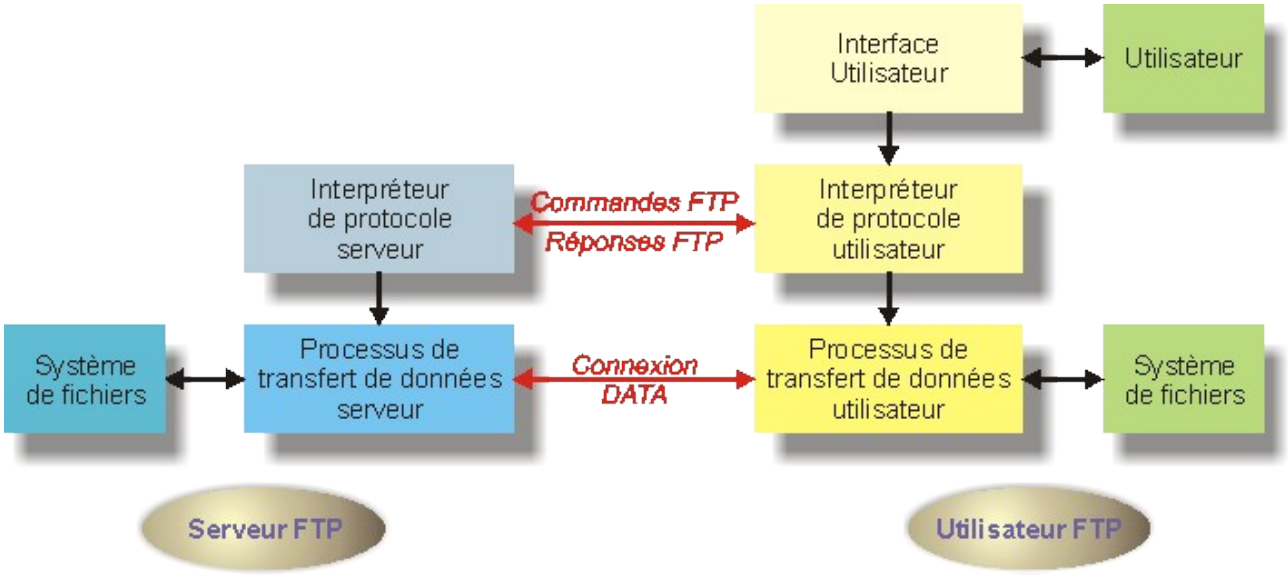
Autres possibilités

Il existe également des utilitaires FTP en ligne de commande, aussi bien sous Windows que sous Linux. Il sont certainement moins conviviaux, mais pas forcément moins puissants.

Le principe de base

Le client ouvre une session FTP sur un serveur. Il existe une grande quantité de serveurs FTP publics. Un serveur FTP requiert une identification du client. Il existe souvent un compte "anonyme", qui donne accès en lecture seule dans la partie publique du serveur, mais il existe également des parties privées où les clients disposant d'un compte peuvent accéder en écriture sur certains répertoires de l'arborescence. C'est le cas, par exemple, pour les mises à jour de pages web personnelles.

2 gftp : http://gftp.seul.org/
NTDL : Celui-ci est inclu dans la distribution GNU/Debian.

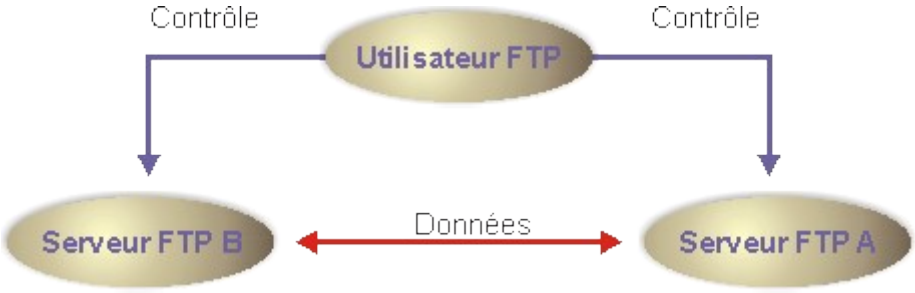


La première chose que l'on constate, c'est que, contrairement à d'autres protocoles comme HTTP, nous allons ici utiliser deux canaux distincts :

- L'un pour l'échange des commandes du protocole,
- l'autre pour le transfert des données elle-mêmes.

Le client FTP (partie de droite), par l'intermédiaire de l'interface utilisateur, va cacher les diverses commandes du protocole FTP par des manipulations plus conviviales, en proposant à l'utilisateur une vision des choses similaire à un gestionnaire de fichiers. Avec des clicks et des "glisser/déposer" l'utilisateur exploitera FTP sans en connaître la multitude de commandes. Mais sachez qu'avec beaucoup de courage et de connaissance du protocole, vous pourriez utiliser Telnet pour faire du FTP.

L'autre cas



Un utilisateur pourra exploiter FTP pour transférer depuis son poste de travail des fichiers d'un serveur distant à un autre serveur distant, sans que les données ne transitent par sa machine, ce qui est fort intéressant si l'on travaille depuis une connexion RTC pour passer des données d'une machine à une autre, ces dernières étant, elles, connectées par des liens à haut débit.

Cependant, cette opération ne sera possible que si les serveurs FTP l'acceptent, ce qui n'est pas souvent le cas, pour des raisons de sécurité.

Après avoir longuement tourné autour du pot (et avoir relu plusieurs fois la RFC 959[3]), j'ai finalement pensé que le meilleur moyen pour comprendre FTP n'était certainement pas la lecture de cette RFC mais plutôt l'expérimentation. Nous allons donc mettre en oeuvre FTP, voir comment ça se passe et vérifier seulement après que c'est bien conforme à ce qui est dit dans les Livres.

Les manipulations sont faites depuis un poste client Windows connecté à un LAN, lui-même connecté à l'Internet par une passerelle NAT GNU/Linux. Un sniffeur est placé sur le poste Windows lui-même, il aurait pu l'être sur la passerelle.

Modes Actif et Passif

Il faut bien prendre le problème par un bout pour le décortiquer, même si pour l'instant, nous ne savons rien, ou pas grand chose, de FTP. Nous sommes donc obligé de faire appel à un paramétrage du client, sans trop savoir pourquoi on va le faire comme ça. Rassurez-vous, nous y reviendrons par la suite.

Le protocole FTP supporte deux manières de fonctionner, à peine différentes, mais la différence est d'importance, surtout lorsque l'on a à traverser un firewall par filtrage de paquets. Ce sont :

- Le mode Actif,
- le mode Passif.

Pour l'instant, contentons-nous de dire que si l'on doit passer un firewall, il vaut mieux utiliser le

3 RFC 959 en français : http://abcdrfc.free.fr/rfc-vf/rfc959.html

mode passif, car le mode actif risque de se solder rapidement par un échec. Ceci dit, mon firewall à moi que j'ai fait moi-même avec IPtables, il sait parfaitement reconnaître du FTP actif. Nous allons donc commencer par ce mode là, qui est la configuration par défaut de Filezilla.

Nous allons nous connecter au serveur ftp.oleane.fr[4], parcourir son arborescence, et télécharger un fichier quelconque, par exemple /pub/doc/rfc/rfc765.txt, puis qu'on en parle. Cette RFC est rendue obsolète par la RFC 959, mais ça peut servir de début...

Ce que montre le sniffeur

Avertissement

L'étude qui va suivre est assez longue, voire laborieuse. Munissez-vous de temps, de friandises et de boissons, parce qu'on va rester coincé ici pendant un petit moment...

Pour vous éviter de faire plusieurs aller-retours sur la page, j'ai essayé d'organier cette étude de façon la plus linéaire possible, c'est ce qui rend ce paragraphe très long. Vous êtes prêts ? Allons-y.

Établissement de la connexion pour les commandes

```
No. Time     Source        Destination   Protocol Info
  1 0.000000 192.168.0.10  194.2.0.36    TCP      1175 > ftp [SYN]
  2 0.022327 194.2.0.36    192.168.0.10  TCP      ftp > 1175 [SYN, ACK]
  3 0.022356 192.168.0.10  194.2.0.36    TCP      1175 > ftp [ACK]
```

Établissement d'une connexion TCP entre le client (192.168.0.10:1175) et le serveur (192.2.0.36:21). Le port 21 est le port standard d'écoute des commandes FTP. Nous trouvons ici le classique dialogue [SYN], [SYN,ACK], [ACK]. Était-il nécessaire de le signaler ? FTP s'appuie bien entendu sur un mode connecté (TCP).

Pour savoir que le port nommé "ftp" est bien le port 21, il suffit d'aller regarder dans le détail de la trame 1 par exemple :

```
Frame 1 (62 bytes on wire, 62 bytes captured)
    ...
    Transmission Control Protocol, Src Port: 1175 (1175), Dst Port: ftp (21)
    Source port: 1175 (1175)
    Destination port: ftp (21)
    ...
```

Mais passons à la suite...

```
  4 0.055680 194.2.0.36    192.168.0.10 FTP      Response: 220 ProFTPD 1.2.0pre10 Server (ProFTPD)
                                                           [ftp.oleane.net]
```

Le serveur entame le dialogue propre au protocole FTP en se présentant. Chaque réponse commence par un nombre, optionnellement suivi d'un commentaire. La réponse 220 signifie : "Service disponible pour nouvel utilisateur".
Vous aurez l'occasion de constater dans la suite à quel point les systèmes informatiques savent être civilisés (souvent plus que les humains). Le serveur se présente, par la même occasion.

```
  5 0.057744 192.168.0.10  194.2.0.36    FTP      Request: USER anonymous
```

4 ftp.oleane.fr : ftp://ftp.oleane.fr/

Le client se présente aussi en indiquant son nom. Comme nous avons fait un accès anonyme, nous utilisons le nom conventionnel "anonymous". Nous n'aurons droit qu'à un accès en lecture. Si nous avions disposé d'un compte d'utilisateur, nous aurions un identifiant personnel (nom d'utilisateur et mot de passe) qui nous permettrait éventuellement de disposer d'un droit d'accès en écriture dans un répertoire de l'arborescence.

```
 6 0.078527 194.2.0.36    192.168.0.10 TCP     ftp > 1175 [ACK]
 7 0.081892 194.2.0.36    192.168.0.10 FTP     Response: 331 Anonymous login ok, send your complete
                                                 e-mail address as password.
```

Le serveur accepte les accès anonymes. Ce n'est pas une obligation, certains serveurs ne le font pas. En général, en accès anonyme, on envoie son adresse e-mail comme mot de passe, mais tout ce qui a vaguement une forme d'adresse e-mail est généralement accepté.

```
 8 0.084076 192.168.0.10  194.2.0.36   FTP      Request: PASS anon@
```

La preuve, Filezilla envoie un laconique " anon@ " et ça va fonctionner quand même...
PASS est une commande FTP, c'est l'abréviation de Password.

```
 9 0.108851 194.2.0.36    192.168.0.10 FTP  Response: 230-Welcome, archive user
                                                anonymous@ca-marseille-51-107.abo.wanadoo.fr!
10 0.109282 194.2.0.36    192.168.0.10 FTP  Response: 230-
```

La preuve... La réponse 230 veut dire : "Session ouverte".

Remarque.
J'ai travaillé comme un cochon, à savoir que dans mon client Filezilla, je n'ai pas renseigné les champs "Utilisateur" ni "Mot de passe". Connaissant les usages, j'aurais pu les remplir en mettant "anonymous" dans "Utilisateur" et "christian.caleca@free.fr" dans "Mot de passe". Ca aurait fonctionné aussi, mais l'expérience montre que dans la plupart des cas, le serveur se moque complètement du mot de passe envoyé pour une connexion anonyme.

```
11 0.109313 192.168.0.10  194.2.0.36   TCP      1175 > ftp [ACK]
12 0.109914 194.2.0.36    192.168.0.10 FTP      Response: 230-The local time is:
                                                          Sat Jan 11 10:32:57 2003
13 0.110341 194.2.0.36    192.168.0.10 FTP      Response: 230-
14 0.110365 192.168.0.10  194.2.0.36   TCP      1175 > ftp [ACK]
15 0.131452 194.2.0.36    192.168.0.10 FTP      Response: 230-For informations about
                                                          this archive service,
                                                          or to report problems,
```

Le serveur nous donne son heure locale, qui peut être utile si l'on devait signaler un problème à l'administrateur du service.

```
16 0.141903 192.168.0.10  194.2.0.36   FTP      Request: PWD
```

Le client envoie la commande PWD qui signifie : "Print Working Directory"

```
17 0.172747 194.2.0.36    192.168.0.10 FTP      Response: 257 "/" is current directory.
```

Nous sommes à la racine de l'arborescence du serveur FTP. Le code 257 signifie "Chemin créé".

Où en sommes-nous ?

Nous avons initié une connexion FTP avec le serveur. Nous nous sommes identifié comme un utilisateur anonyme et nous nous retrouvons dans la racine de l'arborescence du serveur FTP.
Nous avons vu quelques commandes FTP : USER, PASS, CWD et quelques codes de réponse.

Jusqu'ici, c'était relativement simple, nous n'avons transmis que des commandes et des réponses à ces commandes. Maintenant, ça va commencer à se compliquer, parce que nous allons faire aussi transiter des données.

```
18 0.176308 192.168.0.10  194.2.0.36   FTP      Request: PORT 192,168,0,10,4,152
```

Première commande curieuse : PORT 192.168.0.10,4,152, qui nécessite quelques explications.

- PORT, c'est une commande. Le client signale qu'il voudrait utiliser un port particulier,
- 192.168.0.10, on reconnaît, c'est l'adresse du client,
- 4,152, c'est la notation d'un numéro de port, écrit à la mode des adresses IP, c'est à dire sous forme de deux octets, dont chaque octet est exprimé en valeur décimale. Dans la pratique, ça veut dire que le port spécifié sera 4 x 256 + 152 = 1176.

La suite va nous indiquer plus clairement à qui va servir ce port.

```
19 0.198149 194.2.0.36    192.168.0.10 FTP      Response: 200 PORT command successful.
```

Le serveur répond qu'il est d'accord. 200 signifie : "Commande conclue".

```
20 0.200751 192.168.0.10  194.2.0.36   FTP      Request: TYPE A
```

La commande TYPE indique au serveur quel type de données sont attendues. Le type A signale que l'on attend du texte ASCII.

```
21 0.223387 194.2.0.36    192.168.0.10 FTP      Response: 200 Type set to A.
```

Le serveur est toujours d'accord.

```
22 0.225874 192.168.0.10  194.2.0.36   FTP      Request: LIST
```

La commande LIST qui signifie que le client attend la liste des objets présents dans le répertoire courant (l'équivalent de la commande DIR de MSDOS ou ls de UNIX).

Attention !!!
Ce qu'il va se passer maintenant réclame beaucoup d'attention. Rappelons-nous :

- **Que nous sommes en mode Actif,**
- **Que le client a demandé l'usage du port 1176**

```
23 0.247769 194.2.0.36    192.168.0.10 TCP      ftp-data > 1176 [SYN]
24 0.247826 192.168.0.10  194.2.0.36   TCP      1176 > ftp-data [SYN, ACK]
25 0.264940 194.2.0.36    192.168.0.10 TCP      ftp > 1175 [ACK]
26 0.267461 194.2.0.36    192.168.0.10 TCP      ftp-data > 1176 [ACK]
```

Le serveur FTP initie une nouvelle connexion FTP (trames 23,24 et 26). Mais vous avez bien vu, c'est le serveur qui initie la connexion, autrement dit, il agit comme un client TCP, et c'est le client FTP qui va agir comme un serveur TCP, c'est à dire qu'il va rester à l'écoute de son port 1176. Cette particularité est due au mode actif. Le client FTP est actif, parce qu'ici, ce sera lui le serveur (au sens TCP).

Résumons nous

A ce stade, nous avons deux connexions TCP ouvertes :

- 192.168.0.10:1175 -> 194.2.0.36:21. Cette connexion sert à faire passer les commandes et les réponses à ces commandes, 194.2.0.36 est le serveur, au sens TCP, c'est lui qui écoute sur son port 21 ce que le client FTP lui raconte, c'est le canal de commande
- 194.2.0.36:20 -> 192.168.0.10:1176. Cette connexion va servir à faire passer les données. Ici, la liste du répertoire racine du serveur FTP vers le client FTP, qui agit comme un serveur TCP. Ce sera le canal de données.

```
27 0.299444 194.2.0.36    192.168.0.10 FTP      Response: 150 Opening ASCII mode data
                                                  connection for file list.
```

Le serveur FTP répond 150, c'est à dire : "Statut de fichier vérifié, ouverture de canal de données en cours".

```
28 0.303502 194.2.0.36    192.168.0.10 FTP-DATA FTP Data: 75 bytes
29 0.305993 194.2.0.36    192.168.0.10 FTP-DATA FTP Data: 699 bytes
30 0.306052 192.168.0.10  194.2.0.36   TCP      1176 > ftp-data [ACK]
31 0.306101 194.2.0.36    192.168.0.10 FTP      Response: 226-Transfer complete.
```

Normalement, c'est bien le catalogue de la racine du serveur FTP qui a été envoyée vers le client FTP. Nous pouvons le vérifier en regardant par exemple les données contenues dans la trame 29. Ce n'est pas très lisible, mais c'est bien ça. Je vous demande de me croire sur parole, inutile de charger encore d'avantage cette page déjà lourde.

La réponse 226 signifie : "Fermeture du canal de données. Service terminé".

```
32 0.306117 192.168.0.10  194.2.0.36   TCP      1175 > ftp [ACK]
33 0.306470 194.2.0.36    192.168.0.10 FTP      Response: 226 Quotas off
34 0.307484 192.168.0.10  194.2.0.36   TCP      1176 > ftp-data [FIN, ACK]
35 0.338378 194.2.0.36    192.168.0.10 TCP      ftp-data > 1176 [ACK]
```

Comme c'était prévu, 192.168.0.10:1176 met fin à la connexion TCP qui a servi de support au canal de données. C'est le client FTP qui met fin à cette connexion.

```
36 0.457543 192.168.0.10  194.2.0.36   TCP      1175 > ftp [ACK]
37 2.447889 192.168.0.10  194.2.0.36   FTP      Request: CWD pub
```

Nous somme de nouveau sur le canal de commandes et le client FTP demande à changer de répertoire. CWD voulant dire : "Change Working Directory". Nous allons dans le répertoire "pub". Ce qui va maintenant suivre va ressembler à ce que nous venons de voir. Je vous laisse la totalité des trames pour deux raisons :

- Ca va vous permettre de vérifier que vous avez bien compris le mécanisme, parce que je ne vais pas tout répéter,
- Ca va aussi permettre d'observer un détail qui n'est pas sans importance...

```
38 2.471233 194.2.0.36    192.168.0.10 FTP      Response: 250 CWD command successful.
39 2.473782 192.168.0.10  194.2.0.36   FTP      Request: PWD
40 2.499085 194.2.0.36    192.168.0.10 FTP      Response: 257 "/pub" is current directory.
41 2.502415 192.168.0.10  194.2.0.36   FTP      Request: PORT 192,168,0,10,4,153
42 2.524624 194.2.0.36    192.168.0.10 FTP      Response: 200 PORT command successful.
43 2.527863 192.168.0.10  194.2.0.36   FTP      Request: TYPE A
44 2.549182 194.2.0.36    192.168.0.10 FTP      Response: 200 Type set to A.
45 2.551642 192.168.0.10  194.2.0.36   FTP      Request: LIST
46 2.572805 194.2.0.36    192.168.0.10 TCP      ftp-data > 1177 [SYN]
47 2.572856 192.168.0.10  194.2.0.36   TCP      1177 > ftp-data [SYN, ACK]
48 2.585185 194.2.0.36    192.168.0.10 TCP      ftp > 1175 [ACK]
49 2.593535 194.2.0.36    192.168.0.10 TCP      ftp-data > 1177 [ACK]
```

Selon le même mécanisme que celui vu plus haut, un nouveau canal de données est ouvert, mais le client FTP utilise un nouveau port : 1177, cette fois-ci. C'est ce détail qui a son importance...

En effet, dans le cas où nous avons beaucoup de fichiers à transférer, nous allons utiliser beaucoup de ports succesivement. Pour calculer des firewalls qui ne sont pas "statefull", ça ne simplifie pas les choses.

```
 50 2.595535 194.2.0.36    192.168.0.10 FTP      Response: 150 Opening ASCII mode data
                                                          connection for file list.
 51 2.625058 194.2.0.36    192.168.0.10 FTP-DATA FTP Data: 59 bytes
 52 2.627332 194.2.0.36    192.168.0.10 FTP-DATA FTP Data: 718 bytes
 53 2.627375 192.168.0.10  194.2.0.36   TCP      1177 > ftp-data [ACK]
 54 2.629615 194.2.0.36    192.168.0.10 FTP      Response: 226-Transfer complete.
 55 2.629654 192.168.0.10  194.2.0.36   TCP      1175 > ftp [ACK]
 56 2.630203 194.2.0.36    192.168.0.10 FTP      Response: 226 Quotas off
 57 2.652599 194.2.0.36    192.168.0.10 FTP-DATA FTP Data: 1229 bytes
 58 2.652668 192.168.0.10  194.2.0.36   TCP      1177 > ftp-data [ACK]
 59 2.654073 192.168.0.10  194.2.0.36   TCP      1177 > ftp-data [FIN, ACK]
 60 2.700163 194.2.0.36    192.168.0.10 TCP      ftp-data > 1177 [ACK]
```

Bien, nous n'allons pas poursuivre plus longtemps le cheminement dans les sous répertoires, d'autant qu'à chaque niveau, le catalogue devient de plus en plus volumineux.

```
 87 11.063406   192.168.0.10          194.2.0.36            FTP      Request: CWD rfc
...
 95 11.177496   192.168.0.10          194.2.0.36            FTP      Request: LIST
...
```

Et un grand saut plus loin :

```
413 29.719734 192.168.0.10  194.2.0.36   FTP      Request: PWD
414 29.741101 194.2.0.36    192.168.0.10 FTP      Response: 257 "/pub/doc/rfc"
                                                          is current directory.
```

Nous arrivons enfin dans le bon répertoire...

```
415 29.912570 192.168.0.10  194.2.0.36   TCP      1180 > ftp [ACK]
416 30.584219 192.168.0.10  194.2.0.36   FTP      Request: TYPE A
417 30.605939 194.2.0.36    192.168.0.10 FTP      Response: 200 Type set to A.
418 30.609380 192.168.0.10  194.2.0.36   FTP      Request: PORT 192,168,0,10,4,157
419 30.635498 194.2.0.36    192.168.0.10 FTP      Response: 200 PORT command successful.
420 30.639387 192.168.0.10  194.2.0.36   FTP      Request: RETR rfc765.txt
421 30.660814 194.2.0.36    192.168.0.10 TCP      ftp-data > 1181 [SYN]
422 30.660867 192.168.0.10  194.2.0.36   TCP      1181 > ftp-data [SYN, ACK]
423 30.676003 194.2.0.36    192.168.0.10 TCP      ftp > 1180 [ACK]
424 30.683263 194.2.0.36    192.168.0.10 TCP      ftp-data > 1181 [ACK]
425 30.684698 194.2.0.36    192.168.0.10 FTP      Response: 150 Opening ASCII mode data
                                                          connection for rfc765.txt (146641
bytes).
426 30.685450 194.2.0.36    192.168.0.10 FTP-DATA FTP Data: 2 bytes
427 30.687400 194.2.0.36    192.168.0.10 FTP-DATA FTP Data: 716 bytes
```

Nous ouvrons encore un nouveau canal de données, nous en sommes maintenant au port 1181, données de type A toujours (ASCII) et utilisons la commande RETR (Retrieve) pour transférer le fichier rfc765.txt depuis le serveur FTP vers le client FTP.

Certains lecteurs à l'oeil acéré auront remarqué que le port utilisé par le client FTP sur le canal de commande a changé depuis le début de la session. Pour une raison parasite, il y a eu une re connexion au serveur à la fin de la transmission du catalogue de /pub/doc/rfc, re connexion qui a entraîné l'ouverture d'un nouveau canal de commande, sans que le précédent ne soit fermé.

```
428 30.687446 192.168.0.10  194.2.0.36   TCP      1181 > ftp-data [ACK]
429 30.710881 194.2.0.36    192.168.0.10 FTP-DATA FTP Data: 1400 bytes
430 30.712372 194.2.0.36    192.168.0.10 FTP-DATA FTP Data: 1400 bytes
```

```
431 30.712414 192.168.0.10  194.2.0.36   TCP      1181 > ftp-data [ACK]
...
```

Les données commencent à venir, il y en a pour un moment. Nous nous retrouvons à la fin du fichier :

```
614 33.613493 192.168.0.10  194.2.0.36   TCP      1181 > ftp-data [FIN, ACK]
615 33.636771 194.2.0.36    192.168.0.10 TCP      ftp-data > 1181 [ACK]
```

Voilà, c'est fini. nous arrêtons la transaction avec le serveur FTP :

```
616 36.666211 192.168.0.10  194.2.0.36   TCP      1175 > ftp [FIN, ACK]
617 36.686973 194.2.0.36    192.168.0.10 TCP      ftp > 1175 [ACK]
618 36.689259 194.2.0.36    192.168.0.10 TCP      ftp > 1175 [FIN, ACK]
619 36.689275 192.168.0.10  194.2.0.36   TCP      1175 > ftp [ACK]
620 38.511841 192.168.0.10  194.2.0.36   TCP      1180 > ftp [FIN, ACK]
621 38.530529 194.2.0.36    192.168.0.10 TCP      ftp > 1180 [ACK]
622 38.533514 194.2.0.36    192.168.0.10 TCP      ftp > 1180 [FIN, ACK]
623 38.533552 192.168.0.10  194.2.0.36   TCP      1180 > ftp [ACK]
```

Toutes les connexions TCP encore ouvertes sont fermées, y compris le canal de commande initialement ouvert (port 1175).

Qu'avons-nous appris ?

Déjà beaucoup de choses :

- nous avons mis en évidence la présence du canal de commande et du canal de données, placés sur des connexions TCP différentes,
- nous avons, dans ce cas de FTP en mode actif, observé que, pour la création du canal de données, le client FTP :
  - indique au serveur un numéro de port,
  - se met à l'écoute sur ce port (fonction "serveur TCP"),
  - le serveur FTP va quant à lui, utiliser le port 20 pour ce canal de données et agir en client TCP.

  Ce détail est extrêmement important. Il explique la raison pour laquelle le FTP actif ne fonctionne pas correctement sur des filtres de paquets qui interdisent tout paquet SYN depuis le Net vers la zone protégée. Nous avons vu en effet qu'ici, le serveur FTP initie une nouvelle connexion TCP sur le client FTP (ne vous mélangez pas les pédales entre FTP et TCP svp...).
  De plus, si le routeur fait du NAT, comme c'est souvent le cas, ça ne va pas être simple de savoir à qui s'adresse cette nouvelle connexion. N'oublions pas que nous regardons ici ce qu'il se passe derrière le NAT. Le serveur FTP, qui est sur le Net, n'a aucune connaissance de l'IP réelle de son client. Pour lui, son interlocuteur, c'est le routeur NAT lui-même, vu du côté Internet. Je vous laisse réfléchir à cet aspect des choses...
  Je vous l'avais dit, Netfilter sait faire des miracles, si l'on utilise les bons modules,
- nous avons vu également quelques commandes FTP ainsi que quelques codes de réponses,
- enfin, nous avons vu un type de transfert, le type A, qui correspond à de l'ASCII. Mais tout fichier n'est pas forcément de l'ASCII. que va-t-il se passer si le fichier à transporter est, par

exemple, une image jpg ?

Bien qu'à ce niveau, si vous avez réussi à suivre ces explications, vous pouvez commencer à lire cette fameuse rfc 765 avec quelques chances d'y comprendre quelque chose, vous sentez bien, n'est-ce pas, qu'il y aurait encore quelques manipulations intéressantes à faire...

Rassurez-vous, nous allons les faire dans la suite.

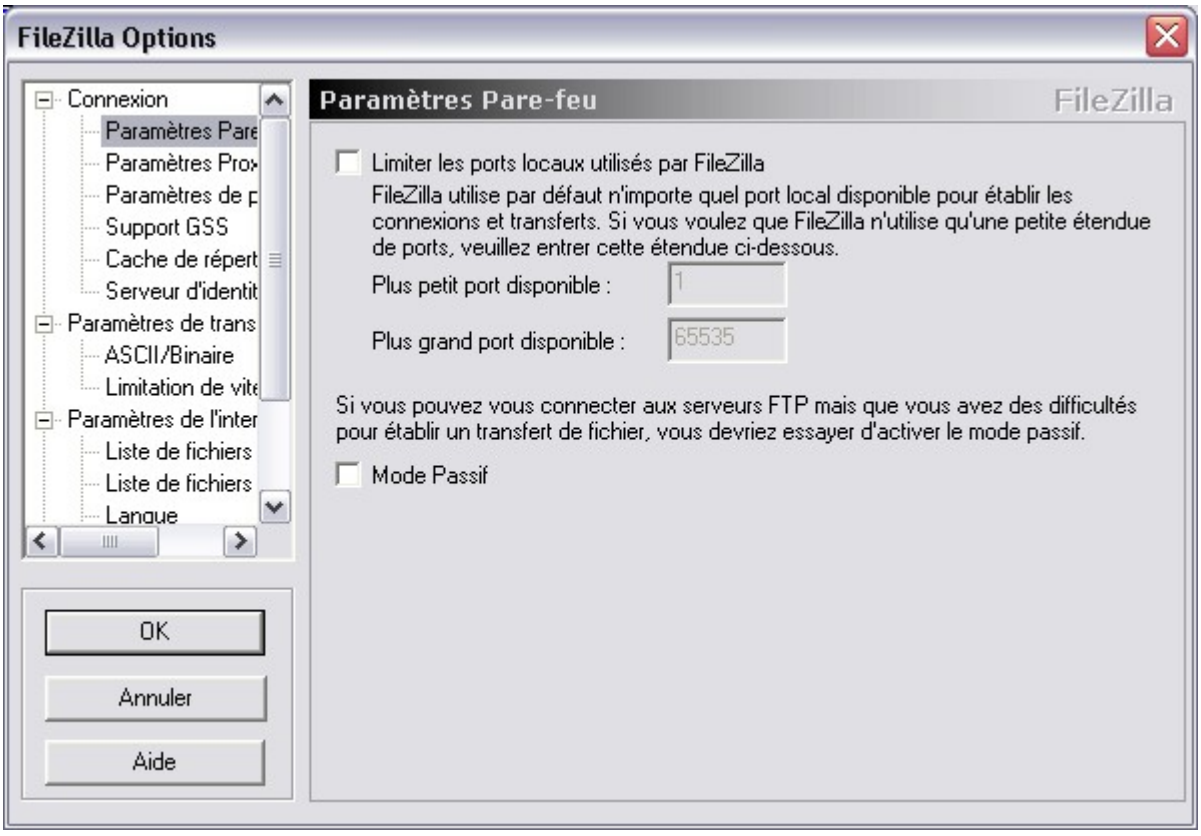
Plus avant

ci, nous allons voir quelques points importants :

- Le mode passif,
- le transfert de fichiers contenant autre chose que du texte,
- un transfert en upload.

Le mode Passif

La première chose à faire, lorsque l'on prend en main un client FTP, c'est de trouver à quel endroit on peut lui expliquer qu'il doit fonctionner en mode actif, ou passif. Continuons avec FileZilla. Dans le menu "Edition/paramètres" :



La rubrique "Connexion/Paramètres Pare feu" permet, entre autres, de sélectionner le mode passif ou non. Nous allons sélectionner le mode passif et voir ce que ça donne. Rassurez-vous, nous n'allons voir que ce qui change vraiment par rapport au mode actif :

```
No. Time     Source        Destination  Protocol Info
...
```

L'ouverture de session FTP est faite, la connexion de commandes est établie, le client envoie la première commande "PWD" :

```
20 10.588511 192.168.0.10  194.2.0.36   FTP      Request: PWD
```

```
21 10.609876 194.2.0.36    192.168.0.10 FTP      Response: 257 "/" is current directory.
```

Jusqu'ici, tout était pareil, mais voyons la suite :

```
22 10.613105 192.168.0.10  194.2.0.36   FTP      Request: PASV
```

Cette nouvelle commande indique au serveur FTP que nous souhaitons fonctionner en mode passif.

```
23 10.661077 194.2.0.36   192.168.0.10 FTP   Response: 227 Entering Passive Mode (194,2,0,36,17,77).
```

Le serveur accepte (227 veut dire : "Passage en mode passif"), et indique un numéro de port, ici 17 x 256 77 = 4429

```
24 10.663545 192.168.0.10  194.2.0.36   FTP      Request: TYPE A
25 10.693712 194.2.0.36    192.168.0.10 FTP      Response: 200 Type set to A.
26 10.697611 192.168.0.10  194.2.0.36   FTP      Request: LIST
27 10.698306 192.168.0.10  194.2.0.36   TCP      1870 > 4429 [SYN]
```

Voyez ici comment la connexion TCP destinée à supporter le canal de données est créée. Ce n'est plus le serveur FTP qui, depuis son port 20 initie une connexion vers le client FTP, mais le client FTP qui ouvre une connexion TCP sur le port que lui a indiqué le serveur, à la suite de la commande PASV.

```
28 10.731631 194.2.0.36    192.168.0.10 TCP      ftp > 1869 [ACK]
29 10.736716 194.2.0.36    192.168.0.10 TCP      4429 > 1870 [SYN, ACK]
30 10.736751 192.168.0.10  194.2.0.36   TCP      1870 > 4429 [ACK]
31 10.757437 194.2.0.36    192.168.0.10 FTP      Response: 150 Opening ASCII mode data
                                                          connection for file list
32 10.761018 194.2.0.36    192.168.0.10 FTP-DATA FTP Data: 75 bytes
...
```

Le reste ne présente pas de nouveautés particulières.

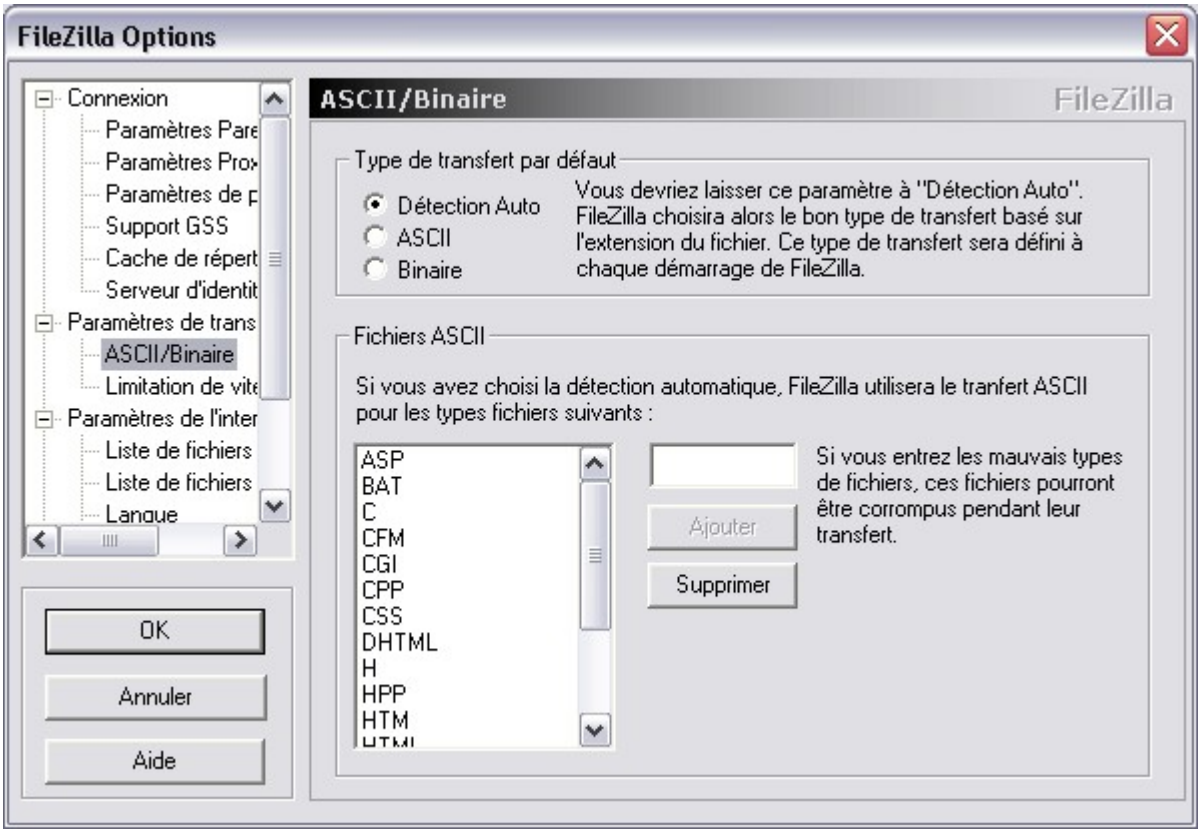
Que pouvons-nous en conclure ?

Dans le mode passif, le client FTP est toujours client TCP. A charge pour le serveur FTP d'ouvrir de nouveaux ports sur lesquels il sera encore serveur TCP, pour le support des canaux de données. Le client FTP n'est jamais serveur TCP. Ca simplifie la vie pour le passage des filtres de paquets, mais ça alourdit la charge des serveurs FTP.

Chaque fois que ce sera possible, surtout si vous travaillez sur des "petits" serveurs FTP , utilisez plutôt le mode actif.

Le transfert non ASCII

C'est la deuxième chose qu'il faut savoir paramétrer sur son client FTP, parce qu'en principe, c'est lui qui décide. Nous avons bien vu que c'était lui qui envoyait la commande TYPE. Toujours sur Filezilla :



Ce client FTP est vraiment très bien fait... Toujours dans le menu "Edition/paramètres", allez dans "Paramètres de transfert/ASCII/Binaire" et voyez. Par défaut, notre client est paramétré pour faire une détection automatique, en fonction de l'extension du fichier. Comme nous avons téléchargé un fichier txt, il a choisi automatiquement le mode ASCII. Vous pouvez forcer un mode. Bien entendu, si vous forcez le mode ASCII, les fichiers non ASCII arriveront corrompus. Il n'y a pas ici de mécanisme de type MIME pour faire de l'encodage base 64 par exemple.

Si vous devez forcer un mode, forcez le mode binaire, qui fonctionnera avec tout type de fichier, mais pas forcément de manière optimale.

Quelles différences ?

Juste un petit morceau de sniff :

```
No. Time       Source       Destination  Protocol Info
...
418 57.591410 192.168.0.10 194.2.0.36   FTP      Request: TYPE I
```

Le type demandé est maintenant I et non plus A

```
419 57.612530 194.2.0.36   192.168.0.10 FTP      Response: 200 Type set to I.
420 57.615147 192.168.0.10 194.2.0.36   FTP      Request: PASV
421 57.635511 194.2.0.36   192.168.0.10 FTP      Response: 227 Entering Passive Mode
(194,2,0,36,18,136).
422 57.640090 192.168.0.10 194.2.0.36   FTP      Request: RETR rfc765.txt
423 57.640783 192.168.0.10 194.2.0.36   TCP      1111 > 4744 [SYN]
424 57.679211 194.2.0.36   192.168.0.10 TCP      4744 > 1111 [SYN, ACK]
425 57.679269 192.168.0.10 194.2.0.36   TCP      1111 > 4744 [ACK]
426 57.684695 194.2.0.36   192.168.0.10 TCP      ftp > 1105 [ACK]
427 57.699374 194.2.0.36   192.168.0.10 FTP      Response: 150 Opening BINARY mode data
                                                  connection for rfc765.txt (146641 bytes).
```

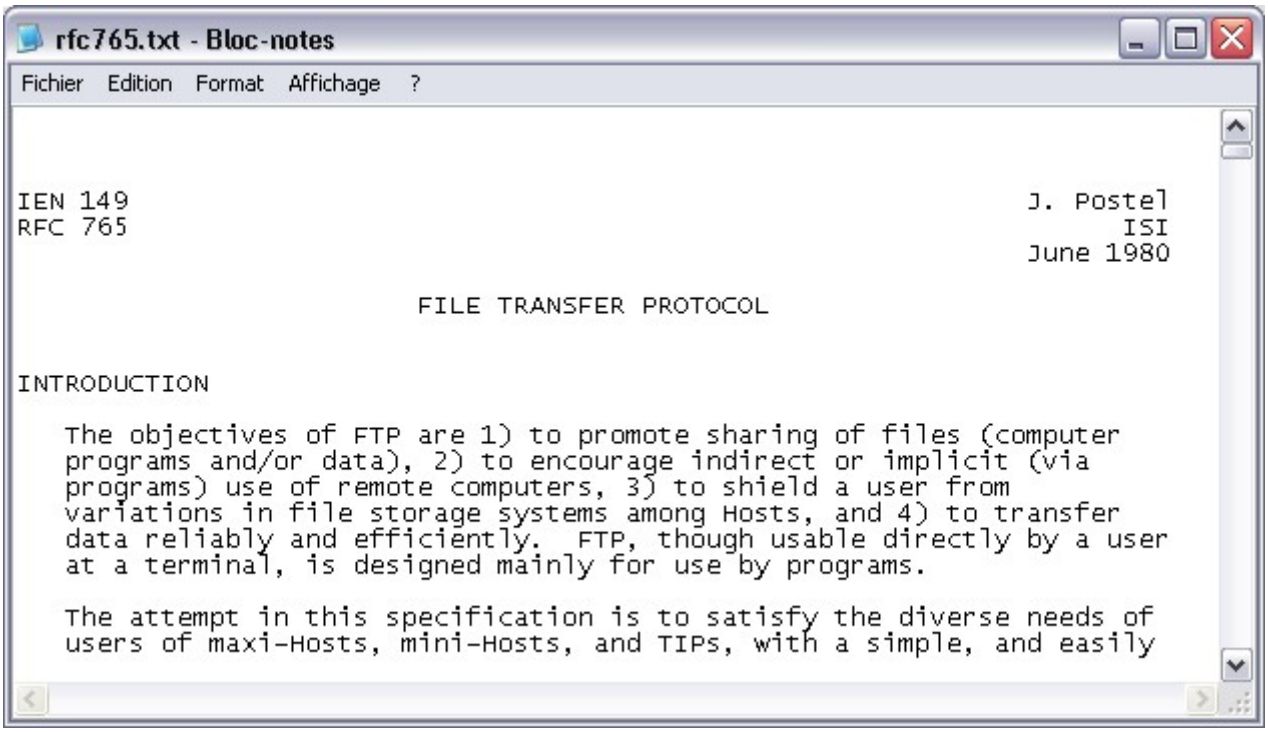
Le serveur répond qu'il ouvre le fichier en mode binaire et non plus ASCII :

```
428 57.722806 194.2.0.36   192.168.0.10 FTP-DATA FTP Data: 1412 bytes
429 57.724012 194.2.0.36   192.168.0.10 FTP-DATA FTP Data: 1412 bytes
...
```

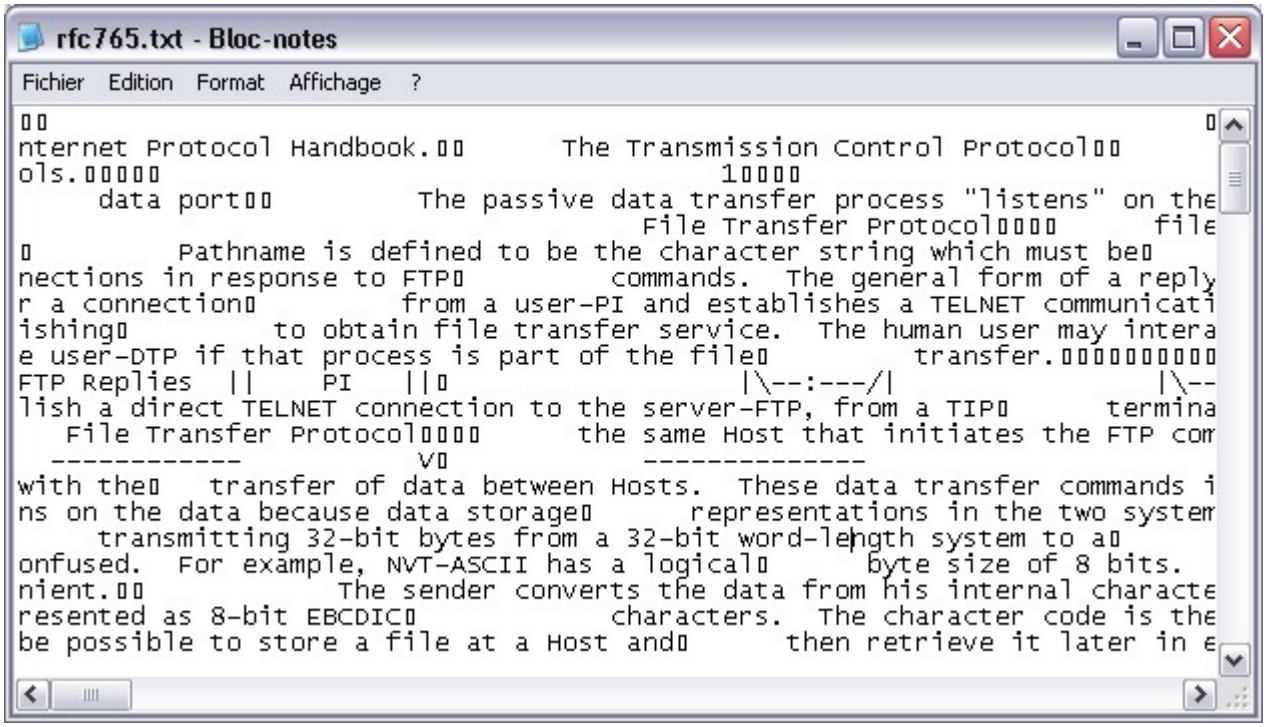
Quels résultats ?

Juste pour voir, ouvrons les deux copies téléchargées avec Wordpad.

La copie téléchargée au format ASCII :



Et celle téléchargée au format binaire :



C'est peut-être un peu moins lisible...

Mais que s'est-il donc passé ?

Rien de bien grave, rassurez-vous. Le document initial est au format UNIX et le document local est passé au format Microsoft. La différence est infime, mais perturbante.

En ASCII, il existe deux caractères non imprimables nommés CR (Carriage Return, retour chariot , de code 0D) et LF (Line Feed, saut de ligne, de code 0A).

Dans un fichier texte UNIX, les retours à la ligne sont signalés par un LF, alors que chez Microsoft, ils le sont par un couple CR LF.

Lors du transfert en mode ASCII, la conversion est effectuée, alors que lors du transfert binaire, rien n'est modifié dans le fichier, ce qui est heureux.

Rassurez-vous, si Notepad ne sait pas interpréter correctement le code LF tout seul, Wordpad, lui, sait le faire, et vous retrouverez votre fichier bien lisible, même s'il est téléchargé en mode binaire.

La RFC 959 explique d'ailleurs la chose au paragraphe 3.1.1.1

Où le serveur engueule le client

J'ai hésité à développer un transfert "upload", cette opération se passant de façon très similaire à un "download". Finalement, je me suis dit que ce serait l'occasion de voir quelques subtilités supplémentaires :

- En utilisant un autre client FTP, ici, ce sera le client gFTP sous GNU/Linux,
- en utilisant un autre serveur FTP, ici, le serveur perso-ftp.wanadoo.fr.

En effet, chaque client FTP a ses petites habitudes et chaque serveur aussi...

Upload en mode actif d'un fichier binaire depuis gFTP vers perso-ftp.wanadoo.fr

Une fois de plus, nous allons étudier dans le détail cet échange, ça ne fait pas de mal de répéter les choses.

```
 No. Time     Source          Destination     Protocol Info
  93 1.620013 192.168.0.254   193.252.18.14   TCP   32781 > ftp [SYN]
  94 1.650006 193.252.18.14   192.168.0.254   TCP   ftp > 32781 [SYN, ACK]
  95 1.650162 192.168.0.254   193.252.18.14   TCP   32781 > ftp [ACK]
```

Ouverture du canal de contrôle.

```
  96 2.228519 193.252.18.14   192.168.0.254   FTP   Response: 220 pwp-ftp2 FTP server
                                                          (@(#)Version : 3-2-2  2002/08/21) ready.
  97 2.228698 192.168.0.254   193.252.18.14   TCP   32781 > ftp [ACK]
  98 2.229021 192.168.0.254   193.252.18.14   FTP   Request: USER xxxxxxxx
  99 2.256639 193.252.18.14   192.168.0.254   TCP   ftp > 32781 [ACK]
 100 2.261983 193.252.18.14   192.168.0.254   FTP   Response: 331 PagePerso login ok, send your
passwd.
 101 2.262205 192.168.0.254   193.252.18.14   FTP   Request: PASS xxxxxx
 102 2.388503 193.252.18.14   192.168.0.254   TCP   ftp > 32781 [ACK]
 103 2.388952 193.252.18.14   192.168.0.254   FTP   Response: 230 User xxxxxx logged in.
                                                                    Access restrictions apply.
```

Identification du client.

```
 104 2.389150 192.168.0.254   193.252.18.14   FTP       Request: TYPE I
 105 2.414035 193.252.18.14   192.168.0.254   FTP       Response: 200 Type set to I.
```

gFTP, quand on lui dit : "Mode binaire", il fait "mode binaire", même pour la récupération de la liste du répertoire. Bête et discipliné, quoi...

```
 106 2.414241 192.168.0.254   193.252.18.14   FTP       Request: PWD
 107 2.433222 193.252.18.14   192.168.0.254   FTP       Response: 257 "/pub" is current directory.
 108 2.438940 192.168.0.254   193.252.18.14   FTP       Request: PORT 192,168,0,254,128,14
 109 2.462881 193.252.18.14   192.168.0.254   FTP       Response: 200 PORT command successful.
 110 2.463200 192.168.0.254   193.252.18.14   FTP       Request: LIST -aL
 111 2.582733 193.252.18.14   192.168.0.254   TCP       ftp > 32781 [ACK]
```

Mais il demande des détails. Les options "a" et "L" permettent d'obtenir les noms commençant par un point (entrées cachées) ainsi que la taille et les attributs de chaque entrée du répertoire.

```
 112 2.757014 193.252.18.14   192.168.0.254   TCP       ftp-data > 32782 [SYN]
 113 2.757157 192.168.0.254   193.252.18.14   TCP       32782 > ftp-data [SYN, ACK]
 114 2.774846 193.252.18.14   192.168.0.254   TCP       ftp-data > 32782 [ACK]
```

Ouverture du canal de données.

```
 115 2.775365 193.252.18.14   192.168.0.254   FTP     Response: 150 Opening BINARY mode data connection
                                                              for /bin/ls.
 116 2.808833 193.252.18.14   192.168.0.254   FTP-DATA FTP Data: 520 bytes
 117 2.808916 193.252.18.14   192.168.0.254   TCP       ftp-data > 32782 [FIN, ACK]
 118 2.808983 192.168.0.254   193.252.18.14   TCP       32782 > ftp-data [ACK]
 119 2.809821 192.168.0.254   193.252.18.14   TCP       32782 > ftp-data [FIN, ACK]
 120 2.822139 192.168.0.254   193.252.18.14   TCP       32781 > ftp [ACK]
 121 2.835875 193.252.18.14   192.168.0.254   TCP       ftp-data > 32782 [ACK]
```

Le canal de données est fermé d'un commun accord.

```
 122 2.844263 193.252.18.14   192.168.0.254   FTP       Response: 226-Transfer complete.
 123 2.844383 192.168.0.254   193.252.18.14   TCP       32781 > ftp [ACK]
 126 6.738405 192.168.0.254   193.252.18.14   FTP       Request: PORT 192,168,0,254,128,15
 127 6.760458 193.252.18.14   192.168.0.254   FTP       Response: 200 PORT command successful.
 128 6.760606 192.168.0.254   193.252.18.14   TCP       32781 > ftp [ACK]
```

```
129 6.760844 192.168.0.254  193.252.18.14  FTP      Request: STOR /pub/ie6_proxy_1.gif
```

Le client va envoyer son fichier.

```
130 6.886677 193.252.18.14  192.168.0.254  TCP      ftp > 32781 [ACK]
131 6.923111 193.252.18.14  192.168.0.254  TCP      ftp-data > 32783 [SYN]
132 6.923242 192.168.0.254  193.252.18.14  TCP      32783 > ftp-data [SYN, ACK]
133 6.947755 193.252.18.14  192.168.0.254  TCP      ftp-data > 32783 [ACK]
```

Un nouveau canal de données est ouvert.

```
134 6.948206 193.252.18.14  192.168.0.254  FTP    Response: 150 Opening BINARY mode data connection
                                                            for /pub/ie6_proxy_1.gif.
135 6.959283 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 1412 bytes
136 6.960470 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 1412 bytes
137 6.982307 192.168.0.254  193.252.18.14  TCP      32781 > ftp [ACK]
```

Et le transfert démarre.

```
138 6.986320 193.252.18.14  192.168.0.254  TCP      ftp-data > 32783 [ACK]
139 6.987578 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 1412 bytes
140 6.988767 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 1412 bytes
141 6.999923 193.252.18.14  192.168.0.254  TCP      ftp-data > 32783 [ACK]
142 7.001176 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 1412 bytes
143 7.002388 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 1412 bytes
144 7.123111 193.252.18.14  192.168.0.254  TCP      ftp-data > 32783 [ACK]
145 7.124365 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 1412 bytes
146 7.125555 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 1412 bytes
147 7.126746 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 1412 bytes
148 7.314251 193.252.18.14  192.168.0.254  TCP      ftp-data > 32783 [ACK]
149 7.315529 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 1412 bytes
150 7.315885 192.168.0.254  193.252.18.14  FTP-DATA FTP Data: 372 bytes
151 7.315950 192.168.0.254  193.252.18.14  TCP      32783 > ftp-data [FIN, ACK]
152 7.502913 193.252.18.14  192.168.0.254  TCP      ftp-data > 32783 [ACK]
153 7.680958 193.252.18.14  192.168.0.254  TCP      ftp-data > 32783 [ACK]
154 7.713910 193.252.18.14  192.168.0.254  TCP      ftp-data > 32783 [ACK]
155 7.714014 193.252.18.14  192.168.0.254  FTP      Response: 226 Transfer complete.
156 7.714093 193.252.18.14  192.168.0.254  TCP      ftp-data > 32783 [FIN, ACK]
157 7.714166 192.168.0.254  193.252.18.14  TCP      32781 > ftp [ACK]
158 7.714232 192.168.0.254  193.252.18.14  TCP      32783 > ftp-data [ACK]
```

Le transfert est terminé, le canal de données est fermé.

```
159 7.714542 192.168.0.254  193.252.18.14  FTP      Request: SITE CHMOD 744 /pub/ie6_proxy_1.gif
160 7.765540 193.252.18.14  192.168.0.254  FTP      Response: 200 CHMOD command successful.
161 7.802347 192.168.0.254  193.252.18.14  TCP      32781 > ftp [ACK]
```

Le monde Unix...
Le client gFTP demande au serveur de changer les attributs du fichier. 744, ça veut dire en gros que le propriétaire a tous les droits et que les autres ne peuvent que lire.

```
162 8.334702 192.168.0.254  193.252.18.14  FTP    Request: PORT 192,168,0,254,128,16
163 8.355047 193.252.18.14  192.168.0.254  FTP    Response: 200 PORT command successful.
164 8.355196 192.168.0.254  193.252.18.14  TCP    32781 > ftp [ACK]
165 8.355366 192.168.0.254  193.252.18.14  FTP    Request: LIST -aL
166 8.435637 193.252.18.14  192.168.0.254  TCP    ftp-data > 32784 [SYN]
167 8.435767 192.168.0.254  193.252.18.14  TCP    32784 > ftp-data [SYN, ACK]
168 8.457199 193.252.18.14  192.168.0.254  TCP    ftp-data > 32784 [ACK]
169 8.457363 193.252.18.14  192.168.0.254  FTP    Response: 150 Opening BINARY mode data connection
                                                            for /bin/ls.
170 8.464111 193.252.18.14  192.168.0.254  FTP-DATA FTP Data: 591 bytes
171 8.464194 193.252.18.14  192.168.0.254  TCP    ftp-data > 32784 [FIN, ACK]
172 8.464262 192.168.0.254  193.252.18.14  TCP    32784 > ftp-data [ACK]
173 8.465013 192.168.0.254  193.252.18.14  TCP    32784 > ftp-data [FIN, ACK]
174 8.492369 192.168.0.254  193.252.18.14  TCP    32781 > ftp [ACK]
175 8.494386 193.252.18.14  192.168.0.254  TCP    ftp-data > 32784 [ACK]
176 8.511638 193.252.18.14  192.168.0.254  FTP    Response: 226-Transfer complete.
```

Rien de bien nouveau, le client rafraîchit la liste des entrées du répertoire courant.

```
177  8.511763 192.168.0.254  193.252.18.14  TCP    32781 > ftp [ACK]
178 14.584417 192.168.0.254  193.252.18.14  TCP    32781 > ftp [FIN, ACK]
179 14.603193 193.252.18.14  192.168.0.254  TCP    ftp > 32781 [ACK]
```

Le client ferme alors le canal de contrôle, sans autre forme de procès...

```
180 14.603616 193.252.18.14  192.168.0.254  FTP      Response: 221 You could at least say goodbye.
```

Et se fait jeter par le serveur qui, bien poli, aurait aimé entendre une formule de politesse du genre "au revoir et merci".

```
181 14.603731 192.168.0.254  193.252.18.14  TCP      32781 > ftp [RST]
```

Mais le client répond en gros "va te faire f..." [RST] indiquant au serveur qu'il parle devant une porte fermée.

Finalement, le savoir vivre informatique est aussi mal distribué que le savoir vivre humain...

Ça y est ? Nous avons tout vu ?

Non. FTP sait encore faire d'autres choses que nous n'avons pas vues ici, mais nous avons pu observer l'essentiel :

- Le transfert binaire ou ASCII. C'est une notion qu'il faut connaître si l'on ne veut pas risquer de perdre beaucoup de temps. Si l'on peut transférer n'importe quoi en mode binaire sans trop de risques, il n'en va pas de même si l'on essaye de transférer en mode ASCII un fichier qui n'est pas du texte.
- Les modes Actif et Passif. La notion est elle aussi essentielle, surtout s'il y a un pare-feu à filtrage de paquets non configuré pour autoriser le mode actif, c'est à dire le mode où le client FTP se retrouve serveur TCP sur le canal de données.
- Nous avons vu avec suffisamment de précision le mécanisme des canaux de commandes et de données, en modes actif et passif, pour pouvoir configurer avec quelques espoir de réussite un firewall du genre GNU/Linux avec Netfilter.

Nous n'avons pas vu en revanche :

- Comment utiliser FTP pour faire de l'impression distante.
- Comment utiliser FTP à travers un serveur PROXY ou un proxy SOCKS, mais ces problèmes ne sont pas spécifiquement du ressort de FTP et ne peuvent se rencontrer qu'en entreprise.
- Enfin, nous n'avons pas vu certains modes de transfert que FTP sait faire, relatifs à la compression des données et à la reprise en cas d'incident, mais pour utiliser ces possibilités, encore faut-il disposer d'un client qui sache les gérer.
- Finalement, nous n'avons pas fait non plus de transfert de fichier de serveur à serveur avec un client distant. Pour mener à bien cette tâche, il vous faudra :
    - Un client FTP qui sache faire ce genre d'opérations, sous Windows, je n'en ai trouvé

qu'un : FlashFXP[5], qui est un shareware,

- des serveurs FTP qui acceptent ce genre d'opérations.

Comme la maison ne recule devant rien, vous trouverez tout de même en page suivante une illustration de ce type de transfert.

Si vous voulez en savoir d'avantage sur FTP, vous pourrez lire maintenant la RFC 959[6] (dont la traduction en français est disponible ici[7]) avec quelques chances d'y comprendre quelque chose.

5 FlashFXP : http://www.flashfxp.com/
6 RFC 959 : http://www.ietf.org/rfc/rfc0959.txt?number=959
7 RFC 959 en français : http://abcdrfc.free.fr/rfc-vf/rfc959.html

Cadeau bonus

Le transfert de serveur à serveur, via un client tiers

Je vous l'ai dit, ne reculant devant aucune difficulté, nous allons tout de même illustrer cet aspect du protocole FTP.

Ce qui est nécessaire

Pour que ça fonctionne, il nous faut disposer :

- de deux serveurs FTP, configurés pour accepter ce type d'opération, ce n'est pas toujours le cas, ça l'est même rarement. Faites quelques recherches sur ce sujet et vous verrez qui se sert le plus souvent de ce type de fonctionnalité...
- un client FTP qui sache le faire.

Pour illustrer l'opération, il faut pouvoir sniffer la totalité de la manipulation, c'est à dire que les trois protagonistes doivent être placés sur le même réseau IP, sans composants de type switch ou pont ; uniquement des hubs.

J'ai donc monté deux serveurs FTP sous GNU/Linux (ProFTPD), configurés pour "Allow foreign data transfers". Dans la pratique, il faut utiliser la directive "AllowForeignAddress on" dans /etc/proftpd.conf, pour la zone où l'on désire accepter ce type d'opération. Voici ce que dit la documentation de ProFTPD :

AllowForeignAddress

Syntax: *AllowForeignAddress on|off*
Default: `AllowForeignAddress off`
Context: *server config, <VirtualHost&, <Anonymous>, <Global>*
Compatibility: *1.1.7 and later*

Normally, proftpd disallows clients from using the ftp PORT command with anything other than their own address (the source address of the ftp control connection), as well as preventing the use of PORT to specify a low-numbered (< 1024) port. In either case, the client is sent an "Invalid port" error and a message is syslog'd indicating either "address mismatch" or "bounce attack". By enabling this directive, proftpd will allow clients to transmit foreign data connection addresses that do not match the client's address. This allows such tricks as permitting a client to transfer a file between two FTP servers without involving itself in the actual data connection. Generally it's considered a bad idea, security-wise, to permit this sort of thing.

AllowForeignAddress ***only*** *affects data connection addresses; not tcp ports. There is no way (and no valid reason) to allow a client to use a low-numbered port in it's PORT command.*

Pour ceux qui ne comprennent absolument pas l'anglais, disons qu'en gros, cette option est par défaut désactivée et que c'est une mauvaise idée que de l'activer, pour des raisons de sécurité.

Le client FTP est quant-à-lui installé sur une machine Windows, il s'agit du shareware FlashFXP, qui sait faire ça très bien, et beaucoup d'autres choses encore. C'est un très bon client FTP, mais

c'est un shareware.

Si vous faites des recherches sur le Net à propos de cette opération, vous verrez souvent apparaître l'acronyme FXP (File eXchange Protocol). FXP est souvent présenté comme un protocole à part entière, ce n'est pas vrai, ce n'est qu'une fonctionnalité décrite dans FTP.

Mais passons à l'acte...

Pour vous aider à vous y retrouver :

- Un serveur FTP s'appelle "gw1.maison.mrs" et dispose de l'adresse 192.168.0.250,
- l'autre serveur s'appelle pchris2.maison.mrs et dispose de l'adresse 192.168.0.254,
- le client Windows s'appelle pchris.maison.ms et dispose de l'adresse 192.168.0.10.

Il n'y a aucun filtre sur le sniffeur, nous verrons donc aussi ARP et les requêtes DNS, DNS également placé sur gw1.maison.mrs. Pour alléger la lecture, j'ai supprimé ces traces inutiles pour la compréhension de FXP.

De même, ce client FTP utilise quelques commandes FTP non vues plus haut, et qui ne sont pas fondamentales pour comprendre FXP. Je les ai également ignorées.

```
 No. Time       Source            Destination       Protocol Info
```

Ouverture du canal de contrôle sur le premier serveur :

```
   5 19.364106 192.168.0.10      192.168.0.250     TCP      1219 > ftp [SYN]
   6 19.364747 192.168.0.250     192.168.0.10      TCP      ftp > 1219 [SYN, ACK]
   7 19.364790 192.168.0.10      192.168.0.250     TCP      1219 > ftp [ACK]
  10 19.488677 192.168.0.250     192.168.0.10      FTP      Response: 220 ProFTPD 1.2.5 Server
                                                             (ProFTPD Default Installation)
                                                             [gw1.maison.mrs]
  11 19.495288 192.168.0.10      192.168.0.250     FTP      Request: USER chris
  12 19.495852 192.168.0.250     192.168.0.10      TCP      ftp > 1219 [ACK]
  13 19.543993 192.168.0.250     192.168.0.10      FTP      Response: 331 Password required for
chris.
  14 19.544575 192.168.0.10      192.168.0.250     FTP      Request: PASS xxxxxxx
  15 19.545099 192.168.0.250     192.168.0.10      TCP      ftp > 1219 [ACK]
  16 19.989752 192.168.0.250     192.168.0.10      FTP      Response: 230 User chris logged in.
```

Voilà qui est fait. Nous n'utilisons pas ici de connexion anonyme, mais un identifiant de connexion d'un utilisateur. C'est nécessaire pour disposer (éventuellement) d'un accès en écriture.

```
...
  24 20.078321 192.168.0.10      192.168.0.250     FTP      Request: PWD
  25 20.080811 192.168.0.250     192.168.0.10      FTP      Response: 257 "/" is current directory.
```

Récupération du listing du répertoire :

```
  26 20.090418 192.168.0.10      192.168.0.250     FTP      Request: TYPE A
  27 20.092416 192.168.0.250     192.168.0.10      FTP      Response: 200 Type set to A.
  28 20.099576 192.168.0.10      192.168.0.250     FTP      Request: PASV
  29 20.101994 192.168.0.250     192.168.0.10      FTP      Response: 227 Entering Passive Mode
                                                             (192,168,0,250,12,185).
```

Ouverture d'un canal de données pour la réception du listing :

```
  30 20.102720 192.168.0.10      192.168.0.250     TCP      1220 > 3257 [SYN]
  31 20.103284 192.168.0.250     192.168.0.10      TCP      3257 > 1220 [SYN, ACK]
  32 20.103327 192.168.0.10      192.168.0.250     TCP      1220 > 3257 [ACK]
  33 20.109338 192.168.0.10      192.168.0.250     FTP      Request: LIST
  34 20.112077 192.168.0.250     192.168.0.10      FTP      Response: 150 Opening ASCII
```

```
                                                                mode data connection for file list
 35 20.196803 192.168.0.250    192.168.0.10     FTP-DATA FTP Data: 83 bytes
 36 20.203187 192.168.0.250    192.168.0.10     FTP-DATA FTP Data: 1460 bytes
 37 20.203223 192.168.0.10     192.168.0.250    TCP      1220 > 3257 [ACK]
 38 20.204008 192.168.0.250    192.168.0.10     FTP-DATA FTP Data: 393 bytes
 39 20.204043 192.168.0.10     192.168.0.250    TCP      1220 > 3257 [ACK]
 40 20.205736 192.168.0.250    192.168.0.10     FTP      Response: 226-Transfer complete.
 41 20.205758 192.168.0.10     192.168.0.250    TCP      1219 > ftp [ACK]
 42 20.206247 192.168.0.250    192.168.0.10     FTP      Response: 226 Quotas off
 43 20.214397 192.168.0.10     192.168.0.250    TCP      1220 > 3257 [FIN, ACK]
 44 20.214872 192.168.0.250    192.168.0.10     TCP      3257 > 1220 [ACK]
 45 20.363982 192.168.0.10     192.168.0.250    TCP      1219 > ftp [ACK]
```

Le listing est transmis et le canal de données est fermé.

Nous allons faire maintenant exactement la même chose avec l'autre serveur FTP

```
 50 39.613933 192.168.0.10     192.168.0.254    TCP      1221 > ftp [SYN]
 51 39.614229 192.168.0.254    192.168.0.10     TCP      ftp > 1221 [SYN, ACK]
 52 39.614272 192.168.0.10     192.168.0.254    TCP      1221 > ftp [ACK]
 55 39.636550 192.168.0.254    192.168.0.10     FTP      Response: 220 ProFTPD 1.2.5 Server
                                                          (ProFTPD Default Installation)
                                                          [pchris2.maison.mrs]
 56 39.642774 192.168.0.10     192.168.0.254    FTP      Request: USER chris
 57 39.643003 192.168.0.254    192.168.0.10     TCP      ftp > 1221 [ACK]
 58 39.644690 192.168.0.254    192.168.0.10     FTP    Response: 331 Password required for chris.
 59 39.645224 192.168.0.10     192.168.0.254    FTP      Request: PASS xxxxxxx
 60 39.676795 192.168.0.254    192.168.0.10     TCP      ftp > 1221 [ACK]
 61 39.743052 192.168.0.254    192.168.0.10     FTP      Response: 230 User chris logged in.
 69 39.774644 192.168.0.10     192.168.0.254    FTP      Request: PWD
 70 39.775221 192.168.0.254    192.168.0.10     FTP      Response: 257 "/" is current directory.
 71 39.784746 192.168.0.10     192.168.0.254    FTP      Request: TYPE A
 72 39.785222 192.168.0.254    192.168.0.10     FTP      Response: 200 Type set to A.
 73 39.791956 192.168.0.10     192.168.0.254    FTP      Request: PASV
 74 39.792560 192.168.0.254    192.168.0.10     FTP      Response: 227 Entering Passive Mode
                                                          (192,168,0,254,128,12).
 75 39.793291 192.168.0.10     192.168.0.254    TCP      1222 > 32780 [SYN]
 76 39.793518 192.168.0.254    192.168.0.10     TCP      32780 > 1222 [SYN, ACK]
 77 39.793549 192.168.0.10     192.168.0.254    TCP      1222 > 32780 [ACK]
 78 39.800533 192.168.0.10     192.168.0.254    FTP      Request: LIST
 79 39.801169 192.168.0.254    192.168.0.10     FTP      Response: 150 Opening ASCII mode data
                                                          connection for file list
 80 39.978200 192.168.0.10     192.168.0.254    TCP      1221 > ftp [ACK]
 81 40.059200 192.168.0.254    192.168.0.10     FTP-DATA FTP Data: 83 bytes
 82 40.061438 192.168.0.254    192.168.0.10     FTP-DATA FTP Data: 1460 bytes
 83 40.061471 192.168.0.10     192.168.0.254    TCP      1222 > 32780 [ACK]
 84 40.062869 192.168.0.254    192.168.0.10     FTP-DATA FTP Data: 1460 bytes
 85 40.063016 192.168.0.254    192.168.0.10     FTP-DATA FTP Data: 110 bytes
 86 40.063049 192.168.0.10     192.168.0.254    TCP      1222 > 32780 [ACK]
 87 40.063095 192.168.0.254    192.168.0.10     FTP      Response: 226-Transfer complete.
 88 40.063169 192.168.0.254    192.168.0.10     FTP      Response: 226 Quotas off
 89 40.063182 192.168.0.10     192.168.0.254    TCP      1221 > ftp [ACK]
 90 40.076529 192.168.0.10     192.168.0.254    TCP      1222 > 32780 [FIN, ACK]
 91 40.076735 192.168.0.254    192.168.0.10     TCP      32780 > 1222 [ACK]
```

A ce niveau, le client FTP a fait les choses suivantes :

- ouverture d'un canal de commande sur le premier serveur,
- ouverture d'un canal de données sur le premier serveur,
- récupération du contenu du répertoire de base du premier serveur,
- fermeture du canal de données sur le premier serveur,
- ouverture d'un canal de commande sur le second serveur,
- ouverture d'un canal de données sur le second serveur,

- récupération du contenu du répertoire de base du second serveur,
- fermeture du canal de données sur le second serveur.

Rappelons pour la suite, que l'objectif est de transférer un fichier depuis 192.168.0.250 vers 192.168.0.254.

```
 92 49.440572 192.168.0.10      192.168.0.254     FTP      Request: SIZE UsersGuide.pdf
 93 49.441478 192.168.0.254     192.168.0.10      FTP      Response: 550 UsersGuide.pdf:
                                                                    No such file or directory
```

Un moyen de savoir si le fichier existe déjà sur la cible. Tout va bien, il n'y est pas.

```
 94 49.453420 192.168.0.10      192.168.0.250     FTP      Request: TYPE I
 95 49.455704 192.168.0.250     192.168.0.10      FTP      Response: 200 Type set to I.
 96 49.462042 192.168.0.10      192.168.0.254     FTP      Request: TYPE I
 97 49.462533 192.168.0.254     192.168.0.10      FTP      Response: 200 Type set to I.
```

Le client demande aux deux serveurs de faire le transfert en mode Image (binaire).

```
 98 49.474530 192.168.0.10      192.168.0.250     FTP      Request: PASV
 99 49.476988 192.168.0.250     192.168.0.10      FTP      Response: 227 Entering Passive Mode
                                                                    (192,168,0,250,12,186).
```

Le serveur source est placé en mode passif.

```
100 49.483559 192.168.0.10      192.168.0.254     FTP      Request: PORT 192,168,0,250,12,186
101 49.484082 192.168.0.254     192.168.0.10      FTP      Response: 200 PORT command successful.
```

Le serveur cible va utiliser le port 3258 de la source.
Mais avez-vous bien remarqué la commande PORT ? Le client (192.168.0.10) envoie au serveur cible (192.168.0.254) la commande PORT avec, comme adresse, celle du serveur source (192.168.0.250).
Dans la pratique, le serveur cible est donc invité à ouvrir un canal de données directement sur le serveur source, sans passer par le client, ce que nous allons vérifier très prochainement.

```
102 49.490373 192.168.0.10      192.168.0.254     FTP      Request: STOR UsersGuide.pdf
```

Le client demande à la cible de récupérer le fichier.

```
103 49.504530 192.168.0.254     192.168.0.250     TCP      ftp-data > 3258 [SYN]
104 49.505090 192.168.0.250     192.168.0.254     TCP      3258 > ftp-data [SYN, ACK]
105 49.505220 192.168.0.254     192.168.0.250     TCP      ftp-data > 3258 [ACK]
```

La cible (192.168.0.254) ouvre donc un canal de données depuis son port 20 (ftp-data) vers la source (192.168.0.250) sur le canal indiqué précédemment : 3258.

```
106 49.507460 192.168.0.254     192.168.0.10      FTP      Response: 150 Opening BINARY mode data
connection
                                                                    for UsersGuide.pdf.
```

La cible est maintenant prête à recevoir les données.

```
107 49.513809 192.168.0.10      192.168.0.250     FTP      Request: RETR UsersGuide.pdf
```

Le client demande maintenant à la source d'envoyer les données.

```
108 49.525033 192.168.0.250     192.168.0.10  FTP Response: 150 Opening BINARY mode data connection
                                                           for UsersGuide.pdf (511748 bytes).
```

La source est prête à envoyer les données.

```
109 49.541114 192.168.0.250     192.168.0.254     FTP-DATA FTP Data: 1448 bytes
110 49.542353 192.168.0.250     192.168.0.254     FTP-DATA FTP Data: 1448 bytes
111 49.542423 192.168.0.254     192.168.0.250     TCP      ftp-data > 3258 [ACK]
112 49.542494 192.168.0.254     192.168.0.250     TCP      ftp-data > 3258 [ACK]
```

Et le transfert commence...

Laissons le faire tranquillement jusqu'à la fin.

```
...
652 49.993346 192.168.0.254     192.168.0.250     TCP      ftp-data > 3258 [ACK]
653 49.993429 192.168.0.250     192.168.0.10      FTP      Response: 226 Transfer complete.
```

La source a fini son transfert, elle le signale au client (pas à la cible).

```
654 49.993629 192.168.0.254     192.168.0.250     TCP      ftp-data > 3258 [FIN, ACK]
655 49.994016 192.168.0.250     192.168.0.254     TCP      3258 > ftp-data [ACK]
```

La cible ferme le canal de données.
Comment fait donc la cible pour savoir que la totalité des données a été transférée, alors que la réponse 226 a été envoyée au client FTP, qui ne l'a pas répercuté à la cible ?
Vous le saurez en lisant la rfc :-)

```
656 49.998088 192.168.0.254     192.168.0.10      FTP      Response: 226 Transfer complete.
```

La cible signale également au client que le transfert est fini.

Et voilà. Pour le reste, il n'y a rien de bien nouveau, le client va rafraîchir l'affichage des contenus des dossiers des deux serveurs. Comme nous n'avons plus rien à faire, nous fermons les canaux de commandes sur les deux serveurs.

Je vous laisse la suite sniff, pour que vous puissiez vérifier par vous-même.

```
657 50.031721 192.168.0.10   192.168.0.250  FTP   Request: TYPE A
658 50.033904 192.168.0.250  192.168.0.10   FTP   Response: 200 Type set to A.
659 50.040995 192.168.0.10   192.168.0.250  FTP   Request: PASV
660 50.043551 192.168.0.250  192.168.0.10   FTP   Response: 227 Entering Passive Mode
                                                   (192,168,0,250,12,187).
661 50.044240 192.168.0.10   192.168.0.250  TCP   1223 > 3259 [SYN]
662 50.044848 192.168.0.250  192.168.0.10   TCP   3259 > 1223 [SYN,ACK]
663 50.044896 192.168.0.10   192.168.0.250  TCP   1223 > 3259 [ACK]
664 50.051654 192.168.0.10   192.168.0.250  FTP   Request: LIST
665 50.061709 192.168.0.250  192.168.0.10   FTP   Response: 150 Opening ASCII mode data connection
                                                    for file list
666 50.097847 192.168.0.250  192.168.0.10   FTP-DATA FTP Data: 83 bytes
667 50.104583 192.168.0.250  192.168.0.10   FTP-DATA FTP Data: 1460 bytes
668 50.104617 192.168.0.10   192.168.0.250  TCP   1223 > 3259 [ACK]
669 50.105386 192.168.0.250  192.168.0.10   FTP-DATA FTP Data: 393 bytes
670 50.105419 192.168.0.10   192.168.0.250  TCP   1223 > 3259 [ACK]
671 50.107014 192.168.0.250  192.168.0.10   FTP   Response: 226-Transfer complete.
672 50.107031 192.168.0.10   192.168.0.250  TCP   1219 > ftp [ACK]
673 50.107533 192.168.0.250  192.168.0.10   FTP   Response: 226 Quotas off
674 50.115291 192.168.0.10   192.168.0.250  TCP   1223 > 3259 [FIN, ACK]
675 50.115737 192.168.0.250  192.168.0.10   TCP   3259 > 1223 [ACK]
676 50.155919 192.168.0.10   192.168.0.254  TCP   1221 > ftp [ACK]
677 50.163703 192.168.0.10   192.168.0.254  FTP   Request: TYPE A
678 50.164199 192.168.0.254  192.168.0.10   FTP   Response: 200 Type set to A.
679 50.171900 192.168.0.10   192.168.0.254  FTP   Request: PASV
680 50.172516 192.168.0.254  192.168.0.10   FTP   Response: 227 Entering Passive Mode
                                                   (192,168,0,254,128,13).
681 50.173205 192.168.0.10   192.168.0.254  TCP   1224 > 32781 [SYN]
682 50.173433 192.168.0.254  192.168.0.10   TCP   32781 > 1224 [SYN, ACK]
683 50.173470 192.168.0.10   192.168.0.254  TCP   1224 > 32781 [ACK]
684 50.180336 192.168.0.10   192.168.0.254  FTP   Request: LIST
685 50.183052 192.168.0.254  192.168.0.10   FTP   Response: 150 Opening ASCII mode data connection
                                                    for file list
```

```
686 50.192934 192.168.0.254  192.168.0.10   FTP-DATA FTP Data: 83 bytes
687 50.195151 192.168.0.254  192.168.0.10   FTP-DATA FTP Data: 1460 bytes
688 50.195183 192.168.0.10   192.168.0.254  TCP   1224 > 32781 [ACK]
689 50.196585 192.168.0.254  192.168.0.10   FTP-DATA FTP Data: 1460 bytes
690 50.196787 192.168.0.254  192.168.0.10   FTP-DATA FTP Data: 181 bytes
691 50.196822 192.168.0.10   192.168.0.254  TCP   1224 > 32781 [ACK]
692 50.196866 192.168.0.254  192.168.0.10   FTP   Response: 226-Transfer complete.
693 50.196883 192.168.0.10   192.168.0.254  TCP   1221 > ftp [ACK]
694 50.197117 192.168.0.254  192.168.0.10   FTP   Response: 226 Quotas off
695 50.210196 192.168.0.10   192.168.0.254  TCP   1224 > 32781 [FIN, ACK]
696 50.210395 192.168.0.254  192.168.0.10   TCP   32781 > 1224 [ACK]
697 50.256503 192.168.0.10   192.168.0.250  TCP   1219 > ftp [ACK]
698 50.357064 192.168.0.10   192.168.0.254  TCP   1221 > ftp [ACK]
699 55.592433 192.168.0.10   192.168.0.254  FTP   Request: QUIT
700 55.592873 192.168.0.254  192.168.0.10   FTP   Response: 221 Goodbye.
701 55.593249 192.168.0.10   192.168.0.254  TCP   1221 > ftp [FIN, ACK]
702 55.594302 192.168.0.254  192.168.0.10   TCP   ftp > 1221 [FIN, ACK]
703 55.594334 192.168.0.10   192.168.0.254  TCP   1221 > ftp [ACK]
704 56.783780 192.168.0.10   192.168.0.250  FTP   Request: QUIT
705 56.785470 192.168.0.250  192.168.0.10   FTP   Response: 221 Goodbye.
706 56.785846 192.168.0.10   192.168.0.250  TCP   1219 > ftp [FIN, ACK]
707 56.789903 192.168.0.250  192.168.0.10   TCP   ftp > 1219 [FIN, ACK]
708 56.789942 192.168.0.10   192.168.0.250  TCP   1219 > ftp [ACK]
```