RFC 7151 : File Transfer Protocol HOST Command for Virtual Hosts

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 12 mars 2014

Date de publication du RFC : Mars 2014

https://www.bortzmeyer.org/7151.html

On ne peut pas dire que le protocole de transfert de fichiers FTP soit encore très utilisé, largement concurrencé qu'il est par HTTP ou BitTorrent. Mais il existe encore de nombreuses ressources accessibles en FTP et de nombreux serveurs FTP donc l'IETF continue à améliorer le protocole. Cette nouvelle extension résout un problème : lorsque plusieurs noms pointent vers la même adresse IP et donc le même serveur FTP, ce serveur ne savait pas sous quel nom il avait été désigné à l'origine, et ne pouvait donc pas ajuster son comportement à ce nom. L'extension `HOST` résout ce problème en permettant au client FTP d'indiquer le nom original.

Cela se fait depuis longtemps pour HTTP (RFC 2616[1] qui a créé HTTP 1.1, remplaçant le HTTP 1.0 du RFC 1945) avec l'en-tête `Host` : envoyé dans la requête, qui permet le mécanisme de "*virtual hosting*" (plusieurs serveurs, ayant des comportements différents, sur la même adresse IP). Mais FTP n'avait pas encore l'équivalent. FTP est normalisé dans le RFC 959. Si vous ne connaissez pas ce protocole ou son histoire, je recommande le texte de Jason Scott <http://ascii.textfiles.com/archives/4199>. Traditionnellement, on n'accédait à un serveur FTP que par un seul nom, et les ressources (fichiers) disponibles ensuite étaient rangés dans des répertoires différents. Si le même serveur FTP anonyme hébergeait un miroir de FreeBSD et un autre de NetBSD, on pouvait créer deux noms, mettons `ftp.freebsd.example` et `ftp.netbsd.example` pointant vers la même adresse IP mais, comme la connexion ne se fait qu'avec les adresses IP (après que le logiciel client ait résolu le nom de domaine en adresse IP, grâce au DNS), le logiciel serveur FTP ne pouvait pas savoir le nom utilisé par le client. Résultat, le serveur devait présenter le même contenu, et l'utilisateur, qu'il soit fan de FreeBSD ou de NetBSD voyait à la racine les répertoires de deux systèmes (et probablement de bien d'autres). Un exemple avec l'excellent client ncftp sur un site miroir de FreeBSD :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc2616.txt`

```
% ncftp ftp.fr.freebsd.org
NcFTP 3.2.5 (Feb 02, 2011) by Mike Gleason (http://www.NcFTP.com/contact/).
Connecting to 158.255.96.2...
Welcome free.org FTP service
Logging in...
Login successful.
Logged in to ftp.fr.freebsd.org.
ncftp / > ls
debian@   mirrors/  private/  pub/      stats/    ubuntu@

ncftp / > cd mirrors/
Directory successfully changed.
ncftp /mirrors > ls
archive.ubuntu.com/          ftp.freebsd.org/             releases.ubuntu-fr.org/
cdimage.debian.org/          ftp.ubuntu.com/              releases.ubuntu.com/
ftp.debian.org/              ftp.xubuntu.com/             videolan/
```

Il aurait quand même été plus pratique d'arriver directement dans `mirrors/ftp.freebsd.org`! Le fait que le serveur ne connaisse pas le nom original a d'autres conséquences, comme de ne pas pouvoir utiliser des systèmes d'authentification différents selon le nom (accès FTP anonyme sur certains et pas sur d'autres, par exemple).

La section 3 du RFC décrit cette extension `HOST`. Elle a la forme d'une commande qui doit être envoyée au serveur **avant** l'authentification (rappelez-vous qu'un des buts de cette commande est d'avoir la possibilité d'authentifications différentes selon le serveur virtuel). Le serveur doit répondre par un code 220 si tout s'est bien passé. Le RFC recommande que le serveur n'envoie le message de bienvenue qu'**après** la commande `HOST` puisque cela lui permettrait d'adapter le message de bienvenue au nom utilisé. Voici un exemple, où « C¿ » indique un message envoyé par le client FTP et « S¿ » un message envoyé par le serveur FTP :

```
C> HOST ftp.example.com
S> 220 Host accepted
```

Et, en cas d'authentification :

```
C> HOST ftp.example.com
S> 220 Host accepted
C> USER foo
S> 331 Password required
C> PASS bar
S> 230 User logged in ftp.example.com
```

La commande `HOST` prend comme paramètre le nom de domaine qui a été utilisé par le client, après d'éventuelles normalisations comme l'ajout d'un suffixe (par contre, si c'est un alias dans le DNS, on utilise l'alias, pas le nom canonique). Si le nom est un IDN, c'est la forme Punycode qui doit être utilisée dans la commande `HOST` :

```
% logiciel-client ftp.café-crème.example
C> HOST ftp.xn--caf-crme-60ag.example
S> 220 Host accepted
```

Si l'utilisateur humain avait utilisé directement une adresse IP, c'est elle qu'on passe à la commande `HOST` :

```
C> HOST [2001:db8::c000:201]
S> 220 Host accepted
```

Ou bien :

```
C> HOST 192.0.2.1
S> 220 Host accepted (but you should use IPv6)
```

Par contre, le client ne doit **pas** indiquer le numéro de port, même lorsque c'était un port non-standard. En effet, le serveur le connait déjà, puisqu'il a cette information dans les données de connexion.

Que doit faire le serveur en recevant un `HOST` ? Il doit normalement vérifier que ce nom correspond bien à un des serveurs virtuels déclarés dans sa configuration. Sinon :

```
C> HOST ftp.example.net
S> 504 Unknown virtual host
```

(Le serveur peut, à la place, décider d'envoyer le client vers le serveur virtuel par défaut.) En cas de succès, le serveur peut ensuite s'adapter à ce nom : changer de répertoire par défaut, activer un autre méthode d'authentification, etc.

Mais que se passe t-il si l'utilisateur a un vieux client FTP qui n'envoie pas de `HOST` ? Là aussi, c'est au serveur de décider quoi faire dans ce cas (utiliser un serveur virtuel par défaut, refuser la connexion, etc). Et si c'est le serveur qui est vieux et qui ne connait pas `HOST` ? C'est alors le cas normal d'une commande inconnue (RFC 959, section 4.2) :

```
C> HOST ftp.example.com
S> 502 Unimplemented
```

Un peu plus compliqué, le cas où la session est protégée par TLS. Si le client utilise l'indication TLS du nom, normalisée dans l'extension `server_name` (RFC 6066), le serveur doit vérifier que le nom donné en paramètre à `HOST` est le même que celui de la session TLS.

Des questions de sécurité ? Comme indiqué plus haut, l'utilisation de la commande `HOST` peut faire changer de système d'authentification. Il est donc crucial que le serveur FTP sépare bien les serveurs virtuels pour éviter, par exemple, qu'on puisse se connecter sur un serveur virtuel avec les lettres de créance d'un autre. Idem lorsque le client authentifie le serveur avec les certificats du RFC 4217. Notre RFC rappelle également l'utilité des extensions d'authentification du RFC 2228 et l'intérêt de la (re)lecture du RFC 2577 sur la sécurité des serveurs FTP.

La nouvelle commande `HOST` est désormais officiellement enregistrée dans le registre IANA <`https://www.iana.org/assignments/ftp-commands-extensions/ftp-commands-extensions.xhtml`>.

L'annexe A discute des autres choix qui auraient pu être faits mais qui ont finalement été rejetés. Une solution possible était d'étendre la commande déjà existante `CWD` ("*Change Working Directory*") en lui donnant comme argument le nom du serveur virtuel. Cela permettrait ainsi d'avoir des sous-répertoires

différents par serveur virtuel, comme c'est déjà souvent le cas (regardez l'exemple de `ftp.fr.freebsd.org` et de son répertoire `mirrors/`). Mais cela a des limites : `CWD` n'est accepté qu'après l'authentification et ne permet donc pas d'avoir des mécanismes d'authentification différents selon le serveur virtuel. Et ça manque de souplesse en forçant une certaines organisation des répertoires du serveur (ou bien en modifiant le code du serveur pour traiter cette première commande `CWD` différemment. Et ce n'est pas terrible du côté sécurité, un visiteur pouvant facilement voir tous les serveurs virtuels.

Une autre solution qui avait été suggérée était la commande `ACCT` qui permet de choisir un compte particulier, mais après l'authentification. Elle a donc un défaut en commun avec `CWD`, elle ne permet pas une authentification différente par serveur virtuel. (Ni d'utiliser des certificats différents, lorsqu'on utilise les extensions de sécurité des RFC 2228 et RFC 4217).

Et la commande `USER` ? Pourquoi ne pas l'étendre en ajoutant le serveur virtuel ?

```
C> USER foo@example.com
S> 331 Password required
C> PASS bar
S> 230 User logged in
```

Le problème est que certains environnements ont déjà des comptes utilisateurs comportant un `@domaine`, cas qu'on ne pourrait pas distinguer de celui où on indique un serveur virtuel. Et les extensions de sécurité de FTP mentionnées plus haut font la négociation de certificats avant le `USER` donc, là encore, on ne pourrait pas avoir des certificats différents selon le serveur virtuel.

Cette extension à FTP est ancienne (première proposition en 2007, apparemment), même si elle n'avait pas été formellement normalisée. Elle est incluse dans IIS comme documenté par un des auteurs du RFC <http://www.iis.net/learn/publish/using-the-ftp-service/using-ftp-virtual-host-n
dans le serveur ncftpd <http://www.ncftp.com/ncftpd/features.html#hosting>, dans le serveur proftpd <http://www.proftpd.org/docs/configs/virtual_authuserfile.conf>...
Mais je n'ai pas testé ce que cela donnait en vrai (il faudrait déjà trouver un client qui le gère).