File Transfer Protocol

File Transfer Protocol (**FTP**) is a standard network protocol used to transfer files from one host to another host over a TCP-based network, such as the Internet.

FTP is built on a client-server architecture and uses separate control and data connections between the client and the server.[1] FTP users may authenticate themselves using a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that hides (encrypts) the username and password, and encrypts the content, FTP is often secured with SSL/TLS ("FTPS"). SSH File Transfer Protocol ("SFTP") is sometimes also used instead, but is technologically different.

The first FTP client applications were command-line applications developed before operating systems had graphical user interfaces, and are still shipped with most Windows, Unix, and Linux operating systems.[2][3] Dozens of FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware, and FTP has been incorporated into hundreds of productivity applications, such as Web page editors.

History

The original specification for the File Transfer Protocol was written by Abhay Bhushan and published as RFC 114 on 16 April 1971. Until 1980, FTP ran on NCP, the predecessor of TCP/IP.[2] The protocol was later replaced by a TCP/IP version, RFC 765 (June 1980) and RFC 959 (October 1985), the current specification. Several proposed standards amend RFC 959, for example RFC 2228 (June 1997) proposes security extensions and RFC 2428 (September 1998) adds support for IPv6 and defines a new type of passive mode.[4]

Protocol overview

Communication and data transfer

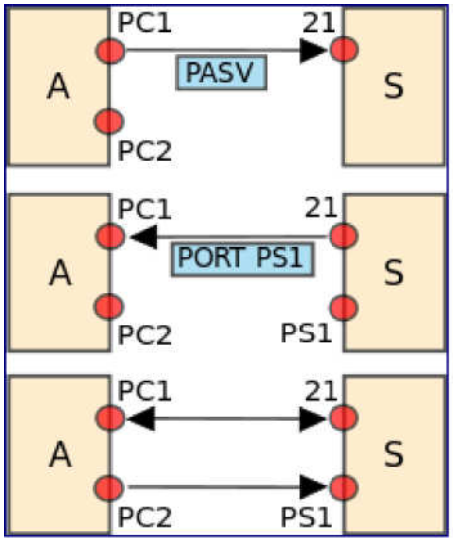

Illustration of starting a passive connection using port 21

FTP may run in *active* or *passive* mode, which determines how the data connection is

established.[5] In both cases, the client creates a TCP control connection from a random unprivileged port N to the FTP server command port 21. In active modes, the client starts listening for incoming data connections on port N+1 from the server (the client sends the FTP command PORT N+1 to inform the server on which port it is listening). In situations where the client is behind a firewall and unable to accept incoming TCP connections, *passive mode* may be used. In this mode, the client uses the control connection to send a PASV command to the server and then receives a server IP address and server port number from the server,[5][6] which the client then uses to open a data connection from an arbitrary client port to the server IP address and server port number received.[7] Both modes were updated in September 1998 to support IPv6. Further changes were introduced to the passive mode at that time, updating it to *extended passive mode*.[8]

The server responds over the control connection with three-digit status codes in ASCII with an optional text message. For example "200" (or "200 OK") means that the last command was successful. The numbers represent the code for the response and the optional text represents a human-readable explanation or request (e.g. <Need account for storing file>).[1] An ongoing transfer of file data over the data connection can be aborted using an interrupt message sent over the control connection.

While transferring data over the network, four data representations can be used:[2][3][4]

- ASCII mode: used for text. Data is converted, if needed, from the sending host's character representation to "8-bit ASCII" before transmission, and (again, if necessary) to the receiving host's character representation. As a consequence, this mode is inappropriate for files that contain data other than plain text.
- Image mode (commonly called Binary mode): the sending machine sends each file byte for byte, and the recipient stores the bytestream as it receives it. (Image mode support has been recommended for all implementations of FTP).
- EBCDIC mode: use for plain text between hosts using the EBCDIC character set. This mode is oth232
- Local mode: Allows two computers with identical setups to send data in a proprietary format without the need to convert it to ASCII

For text files, different format control and record structure options are provided. These features were designed to facilitate files containing Telnet or ASA

Data transfer can be done in any of three modes:[1][2]

- Stream mode: Data is sent as a continuous stream, relieving FTP from doing any processing. Rather, all processing is left up to TCP. No End-of-file indicator is needed, unless the data is divided into records.
- Block mode: FTP breaks the data into several blocks (block header, byte count, and data field) and then passes it on to TCP.[4]
- Compressed mode: Data is compressed using a single algorithm (usually run-length encoding).

Login

FTP login utilizes a normal username and password scheme for granting access.[2] The username is sent to the server using the USER command, and the password is sent using the PASS command.[2] If the information provided by the client is accepted by the server, the server will send a greeting to the client and the session will commence.[2] If the server supports it, users may log in without providing login credentials, but the same server may authorize only limited access for such sessions.[2]

Anonymous FTP

A host that provides an FTP service may provide anonymous FTP access.[2] Users typically log into the service with an 'anonymous' (lower-case and case-sensitive in some FTP servers) account when prompted for user name. Although users are commonly asked to send their email address instead of a password,[3] no verification is actually performed on the supplied data.[9] Many FTP hosts whose purpose is to provide software updates will allow anonymous logins.[3]

NAT and firewall traversal

FTP normally transfers data by having the server connect back to the client, after the PORT command is sent by the client. This is problematic for both NATs and firewalls, which do not allow connections from the Internet towards internal hosts.[10] For NATs, an additional complication is that the representation of the IP addresses and port number in the PORT command refer to the internal host's IP address and port, rather than the public IP address and port of the NAT.

There are two approaches to this problem. One is that the FTP client and FTP server use the PASV command, which causes the data connection to be established from the FTP client to the server.[10] This is widely used by modern FTP clients. Another approach is for the NAT to alter the values of the PORT command, using an application-level gateway for this purpose.[10]

Differences from HTTP

When operating in its modern passive mode, FTP uses a single socket for both signalling and for actual file data, just like the HTTP protocol. But when used in its original configuration, in “active mode” with a separate socket for the download, FTP exhibits true out-of-band control which is not an option with HTTP.[11]

Web browser support

Most common web browsers can retrieve files hosted on FTP servers, although they may not support protocol extensions such as FTPS.[3][12] When an FTP—rather than an HTTP—URL is supplied, the accessible contents on the remote server are presented in a manner that is similar to that used for other Web content. A full-featured FTP client can be run within Firefox in the form of an extension called FireFTP

Syntax

FTP URL syntax is described in RFC1738,[13] taking the form: ftp://[<user>[:<password>]@]<host>[:<port>]/<url-path>[13] (The bracketed parts are optional.) For example:

ftp://public.ftp-servers.example.com/mydirectory/myfile.txt

or:

ftp://user001:secretpassword@private.ftp-servers.example.com/mydirectory/myfile.txt

More details on specifying a username and password may be found in the browsers' documentation, such as, for example, Firefox [14] and Internet Explorer.[15] By default, most web browsers use passive (PASV) mode, which more easily traverses end-user firewalls.

Security

FTP was not designed to be a secure protocol—especially by today's standards—and has many security weaknesses.[16] In May 1999, the authors of RFC 2577 listed a vulnerability to the following problems:[17]

- Brute force attacks
- Bounce attacks
- Packet capture (sniffing)
- Port stealing
- Spoof attacks
- Username protection

FTP is not able to encrypt its traffic; all transmissions are in clear text, and usernames, passwords, commands and data can be easily read by anyone able to perform packet capture (sniffing) on the network.[2][16] This problem is common to many of the Internet Protocol specifications (such as SMTP, Telnet, POP and IMAP) that were designed prior to the creation of encryption mechanisms such as TLS or SSL.[4] A common solution to this problem is to use the "secure", TLS-protected versions of the insecure protocols (e.g. FTPS for FTP, TelnetS for Telnet, etc.) or a different, more secure protocol that can handle the job, such as the SFTP/SCP tools included with most implementations of the Secure Shell protocol.

Secure FTP

There are several methods of securely transferring files that have been called "Secure FTP" at one point or another.

FTPS

Explicit FTPS is an extension to the FTP standard that allows clients to request that the FTP session be encrypted. This is done by sending the "AUTH TLS" command. The server has the option of allowing or denying connections that do not request TLS. This protocol extension is defined in the proposed standard: RFC 4217. Implicit FTPS is a deprecated standard for FTP that required the use of a SSL or TLS connection. It was specified to use different ports than plain FTP.

SFTP

SFTP, the "SSH File Transfer Protocol", is not related to FTP except that it also transfers files and has a similar command set for users. SFTP, or secure FTP, is a program that uses Secure Shell (SSH) to transfer files. Unlike standard FTP, it encrypts both commands and data, preventing passwords and sensitive information from being transmitted openly over the network. It is functionally similar to FTP, but because it uses a different protocol, standard FTP clients cannot be used to talk to an SFTP server, nor can one connect to an FTP server with a client that supports only SFTP.

FTP over SSH (not SFTP)

FTP over SSH (not SFTP) refers to the practice of tunneling a normal FTP session over an SSH connection.[16] Because FTP uses multiple TCP connections (unusual for a TCP/IP protocol that is still in use), it is particularly difficult to tunnel over SSH. With many SSH clients, attempting to set up a tunnel for the *control channel* (the initial client-to-server connection on port 21) will protect only that channel; when data is transferred, the FTP software at either end will set up new TCP connections (*data channels*), which bypass the SSH connection and thus have no confidentiality or integrity protection, etc.

Otherwise, it is necessary for the SSH client software to have specific knowledge of the FTP protocol, to monitor and rewrite FTP control channel messages and autonomously open new packet forwardings for FTP data channels. Software packages that support this mode include:

- Tectia ConnectSecure (Win/Linux/Unix) of SSH Communications Security's software suite
- Tectia Server for IBM z/OS of SSH Communications Security's software suite
- FONC (the GPL licensed)
- Co:Z FTPSSH Proxy

Other methods of transferring files using SSH that are not related to FTP include SFTP and SCP; in each of these, the entire conversation (credentials and data) is always protected by the SSH protocol.

FTP reply codes

Below is a summary of the reply codes that may be returned by an FTP server. These codes have been standardized in RFC 959 by the IETF. As stated earlier in this article, the reply code is a three-digit value. The first digit is used to indicate one of three possible outcomes—success, failure or to indicate an error or incomplete reply:

- 2yz – Success reply
- 4yz or 5yz – Failure Reply
- 1yz or 3yz – Error or Incomplete reply

The second digit defines the kind of error:

- x0z – Syntax. These replies refer to syntax errors.
- x1z – Information. Replies to requests for information.
- x2z – Connections. Replies referring to the control and data connections.
- x3z – Authentication and accounting. Replies for the login process and accounting

procedures.
- x4z – Not defined.
- x5z – File system. These replies relay status codes from the server file system.

The third digit of the reply code is used to provide additional detail for each of the categories defined by the second digit.

References

1. Forouzan, B.A. (2000). TCP/IP: Protocol Suite. 1st ed. New Delhi, India: Tata McGraw-Hill Publishing Company Limited.
2. Kozierok, Charles M. (2005). "The TCP/IP Guide v3.0". Tcpiguide.com.
3. Dean, Tamara (2010). *Network+ Guide to Networks*. Delmar. pp. 168–171.
4. Clark, M.P. (2003). Data Networks IP and the Internet. 1st ed. West Sussex, England: John Wiley & Sons Ltd.
5. "Active FTP vs. Passive FTP, a Definitive Explanation". Slacksite.com. (see http://webcache.googleusercontent.com/search?q=cache:http://slacksite.com/other/ftp.html if the original web page is not available).
6. Parker, Don (September 2005). "Understanding the FTP Protocol". Windowsnetworking.com.
7. Postel, J., & Reynolds. J. (October 1985). . In The Internet Engineering Task Force..
8. Allman, M. & Metz, C. & Ostermann, S. (September 1998). RFC 2428. In The Internet Engineering Task Force.
9. P. & Emtage, A. & Marine, A. (May 1994). "RFC 1635". The Internet Engineering Task Force.
10. Gleason, Mike (2005). "The File Transfer Protocol and Your Firewall/NAT". Ncftp.com.
11. Kurose, J.F. & Ross, K.W. (2010). *Computer Networking*. 5th ed. Boston, MA: Pearson Education, Inc.
12. Matthews, J. (2005). Computer Networking: Internet Protocols in Action. 1st ed. Danvers, MA: John Wiley & Sons Inc.
13. Berners-Lee, T. & Masinter, L. & McCahill, M. (December 1994). "RFC 1738". The Internet Engineering Task Force.
14. "Accessing FTP servers | How to | Firefox Help". Support.mozilla.com. 2012-09-05. Retrieved 2013-01-16.
15. "How to Enter FTP Site Password in Internet Explorer". Support.microsoft.com. 2011-09-23. Retrieved 2013-01-16.
16. Securing FTP using SSH. Retrieved from http://www.nurdletech.com/linux-notes/ftp/ssh.html
17. Allman, M. & Ostermann, S. (May 1999). "RFC 2577". The Internet Engineering Task Force.

Further reading

- RFC 959 – (Standard) File Transfer Protocol (FTP). J. Postel, J. Reynolds. October 1985.
- RFC 1579 – (Informational) Firewall-Friendly FTP.
- RFC 2228 – (Proposed Standard) FTP Security Extensions.
- RFC 2389 – (Proposed Standard) Feature negotiation mechanism for the File Transfer

Protocol. August 1998.

- RFC 2428 – (Proposed Standard) Extensions for IPv6, NAT, and Extended passive mode. September 1998.
- RFC 2640 – (Proposed Standard) Internationalization of the File Transfer Protocol.
- RFC 3659 – (Proposed Standard) Extensions to FTP. P.Hethmon. March 2007.
- RFC 5797 – (Proposed Standard) FTP Command and Extension Registry. March 2010.
- RFC 697 - CWD Command of FTP
- RFC 1639 - FTP Operation Over Big Address Records (FOOBAR)
- RFC 5797 - FTP Command and Extension Registry