Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 3, 2013

A. Bryan
D. Stenberg
T. Tsujikawa

January 30, 2013

File Transfer Protocol LOCK Command for Using a Single Port
draft-bryan-ftp-lock-02

Abstract

One of the biggest hurdles for FTP in real life usage is its use of two connections. First, it uses a primary connection to send control commands on, and when it sends or receives data, it opens a second TCP stream for that purpose. This document specifies a new FTP LOCK command to be used by clients to request the server to use the control connection for data transfers, using a single port instead of two.

Editorial Note (To be removed by RFC Editor)

Discussion of this draft should take place on the FTPEXT2 working group mailing list (ftpext@ietf.org), although this draft is not a WG item. Related documents (including fancy diffs) can be found at <http://tools.ietf.org/wg/ftpext2/>.

The changes in this draft are summarized in Appendix C.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 3, 2013.

Copyright Notice

Table of Contents

1. Introduction 3
2. Example 3
3. Document Conventions 4
  3.1. Basic Tokens 4
  3.2. Server Replies 5
4. FTP Data Transfers Over Control Connection 5
5. The LOCK Command (LOCK) 5
  5.1. FEAT Command Response for LOCK Command 7
  5.2. User-PI usage of LOCK 7
  5.3. LOCK Command Errors 8
6. IANA Considerations 9
7. Security Considerations 9
8. References 9
  8.1. Normative References 9
  8.2. Informative References 9
Appendix A. Acknowledgements and Contributors 10
Appendix B. Two Port FTP 10
Appendix C. Document History 10

1. Introduction

One of the biggest hurdles for FTP in real life usage is its use of two connections. First, it uses a primary connection to send control commands on, and when it sends or receives data, it opens a second TCP stream for that purpose. This document specifies a new FTP LOCK command to be used by clients to request the server to use the control connection for data transfers, using a single port instead of two.

The use of two connections for FTP, where the second one uses dynamic port numbers and can go in either direction, has been known to give firewall administrators grief and firewalls really have to "understand" FTP at the application protocol layer to work really well. Since firewalls need to inspect and understand FTP to be able to open ports for the secondary connection etc, there's a huge problem with encrypted FTP (FTP-SSL or FTPS) since then the control connection is sent encrypted and the firewall(s) cannot interpret the commands that deal with creating the second connection. Things might seem fine over the control connection, but will start to fail when the data connection is attempted.

This also means that if both parties are behind NATs (Network Address Translation), you cannot use FTP.

Additionally, as NATs often are setup to kill idle connections and the nature of FTP makes the control channel remain quiet during long and slow FTP transfers, we often end up with the control channel getting cut off by the NAT due to idleness.

2. Example

Example of LOCK client request:

```
C> LOCK
S> 200 LOCK OK to current port
C> RETR filename.ext
S> 150 Opening BINARY mode
S> boundary=separator189dhde78b287734237842g3847g
S> --separator189dhde78b287734237842g3847g
   [Server sends raw binary data]
S> --separator189dhde78b287734237842g3847g--
S> 226 Transfer complete.
```

3. Document Conventions

This specification describes conformance of File Transfer Protocol Extension for LOCK, so data transfers occur over the control connection and not a separate data connection.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119], as scoped to those conformance targets.

This document also uses notation defined in STD 9, [RFC0959]. In particular, the terms or commands "reply", "user", "file", "FTP commands", "user-PI" (user protocol interpreter), "server-FTP process", "server-PI", "control connection", "data connection", "mode", "Image type", "Stream transfer mode", "type", "LIST", "NLST", "STOR", "RETR", "STOU", "APPE", and "ASCII", are all used here as defined there.

In the examples of FTP dialogs presented in this document, lines that begin "C> " were sent over the control connection from the user-PI to the server-PI, and lines that begin "S> " were sent over the control connection from the server-PI to the user-PI. In all cases, the prefixes shown above, including the one space, have been added for the purposes of this document, and are not a part of the data exchanged between client and server.

Syntax required is defined using the Augmented BNF defined in [RFC5234].

3.1. Basic Tokens

This document imports the core definitions given in Appendix B of [RFC5234]. There definitions will be found for basic ABNF elements like ALPHA, DIGIT, SP, etc. To that, the following term is added for use in this document.

```
   TCHAR = VCHAR / SP / HTAB    ; visible plus white space
```

The VCHAR (from [RFC5234]) and TCHAR rules give basic character types from varying sub-sets of the ASCII character set for use in various commands and responses.

Note that in ABNF, string literals are case insensitive. That convention is preserved in this document, and implies that FTP commands and parameters that are added by this specification have values that can be represented in any case. That is, "LOCK" is the

same as "lock", "Lock", "LoCk", etc., and "ftp.example.com" is the same as "Ftp.Example.Com", "fTp.eXample.cOm", etc.

3.2. Server Replies

Section 4.2 of [RFC0959] defines the format and meaning of replies by the server-PI to FTP commands from the user-PI. Those reply conventions are used here without change.

```
error-response = error-code SP *TCHAR CRLF
error-code     = ("4" / "5") 2DIGIT
```

Implementers should note that the ABNF syntax (which was not used in [RFC0959]) used in this document, and other FTP related documents, sometimes shows replies using the one line format. Unless otherwise explicitly stated, that is not intended to imply that multi-line responses are not permitted. Implementers should assume that, unless stated to the contrary, any reply to any FTP command (including QUIT) can be of the multi-line format described in [RFC0959].

Throughout this document, replies will be identified by the three digit code that is their first element. Thus the term "500 reply" means a reply from the server-PI using the three digit code "500".

4. FTP Data Transfers Over Control Connection

As mentioned, the use of two ports with FTP became problematic with the advent of firewalls and NATs.

By using a single port, like many other protocols, these problems are eliminated.

There are drawbacks, such as not being able to carry out control connection actions during transfer, but the benefits outweigh them. Many implementations do not really allow the client to do much on the control connection while the transfer is ongoing anyway.

The LOCK command causes anything that would normally open a data connection to be re-routed over the control connection and remain using a single connection.

5. The LOCK Command (LOCK)

A new command "LOCK" is added to the FTP command set to allow the client to request that data transfers occur over the current control connection without opening up another port for a data connection.

The syntax for the LOCK command when the current transfer mode is STREAM is:

```
lock-command = "LOCK" CRLF
lock-response = lock-ok / error-response
lock-ok       = "200" SP *TCHAR CRLF
```

Post-LOCK transfers use a MIME-style separator. The boundary style is a subset of MIME. The first boundary string is prefixed with two dashes "--" and a blank line, while the final boundary string is prefixed and suffixed with two dashes "--".

The sender MUST set the separator. A careful sender SHOULD check the file first so that there is no occurance of the separator within the file, but this method is used by browsers and HTTP clients today without checking the file since the risk that a very long string with lots of randomness would actually exist in the file is next to none.

When uploading files, the client MUST use end-of-marker solution.

```
boundary-announce = "boundary=" separator CRLF
boundary-start    = "--" separator CRLF CRLF
boundary-end      = "--" separator "--" CRLF
separator         = *TCHAR
```

The following commands would usually cause a data connection to be opened, but post-LOCK they will occur over the control connection: LIST, NLST, STOR, RETR, STOU, APPE.

The LOCK command will cause LIST and NLST directory listings to be sent over the control connection, instead of the data connection. Some clients issue one of these commands automatically at login, so if the server supports LOCK and the client prefers LOCK, then LOCK should be issued after login but before one of these commands.

If the LOCK command is issued after a data connection has already been opened, it will continue to completion uninterrupted and close once finished.

To turn off LOCK and return to the previous behavior, a client can issue a PORT, PASV, or similar command. Another alternative is to reinitialize the session with REIN but this terminates the user and resets all parameters.

5.1. FEAT Command Response for LOCK Command

When replying to the FEAT command [RFC2389], a server-FTP process that supports the LOCK command, as specified here, MUST include, a line containing exactly the string "LOCK". This string is case insensitive, and MAY be sent in any mixture of upper or lower case, however it SHOULD be sent in upper case. That is, the response SHOULD be:

```
C> FEAT
S> 211-Extensions supported:
S>  ...
S>  LOCK
S>  ...
S> 211 END
```

The ellipses indicate place holders where other features may be included, and are not required. The one-space indentation of the feature lines is mandatory [RFC2389].

```
lock-feat = SP "LOCK" CRLF
```

5.2. User-PI usage of LOCK

The user-PI issues the FEAT command to query the server-PI if it supports the LOCK command.

```
C> FEAT
S> 211-Extensions supported:
S>  ...
S>  LOCK
S>  ...
S> 211 END
```

Next, the user-PI issues the LIST command.

```
   C> LOCK
   S> 200 LOCK OK to current port
   C> LIST
   S> 125 Transfer starting.
   S> --separator189dhde78b287734237842g3847g
      [Server sends raw binary data]
   S> --separator189dhde78b287734237842g3847g--
   S> 226 Transfer complete.
```

The client requests that data transfers will be over the current connection, instead of opening another port, and RETRieves a file.

```
   C> TYPE I
   S> 200 Type set to I.
   C> LOCK
   S> 200 LOCK OK to current port
   C> RETR filename.ext
   S> 150 Opening BINARY mode
   S> boundary=separator189dhde78b287734237842g3847g
   S> --separator189dhde78b287734237842g3847g
      [Server sends raw binary data]
   S> --separator189dhde78b287734237842g3847g--
   S> 226 Transfer complete.
```

In this example, the client requests that data transfers will be over the current connection, instead of opening another port, and STORes a file.

```
   C> TYPE I
   S> 200 Type set to I.
   C> LOCK
   S> 200 LOCK OK to current port
   C> STOR metalinkc.py
   S> 125 Transfer starting.
   C> boundary=separator189dhde78b287734237842g3847g
   C> --separator189dhde78b287734237842g3847g
      [Client sends raw binary data]
   C> --separator189dhde78b287734237842g3847g--
   S> 226 Transfer complete.
```

5.3. LOCK Command Errors

The server-PI SHOULD reply with a 500 reply if the LOCK command is unrecognized or unimplemented.

The server-PI SHOULD reply with a 552 reply if the user is not

allowed to use the LOCK command.

6. IANA Considerations

This new command is added to the "FTP Commands and Extensions" registry created by [RFC5797].

Command Name: LOCK

Description: Single port data transfers for FTP.

FEAT String: LOCK

Command Type: Service execution/parameter setting

Conformance Requirements: Optional

Reference: This specification

7. Security Considerations

FTP can be secured with TLS by encrypting the control connection and data connection, as per [RFC4217]. Under optimal conditions with LOCK, no data connection is opened and all data transfers occur over the control connection. When LOCK is in use, if data transfers need to be encrypted, then the control connection MUST be encrypted (instead of the data connection, as with non-LOCK transfers).

8. References

8.1. Normative References

[RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 0959, October 1985.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2389] Hethmon, P. and R. Elz, "Feature negotiation mechanism for the File Transfer Protocol", RFC 2389, August 1998.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

8.2. Informative References

[RFC4217] Ford-Hutchinson, P., "Securing FTP with TLS", RFC 4217, October 2005.

[RFC5797] Klensin, J. and A. Hoenes, "FTP Command and Extension Registry", RFC 5797, March 2010.

Appendix A. Acknowledgements and Contributors

Thanks to the FTPEXT2 Working Group, Robert McMurray, and Alun Jones.

Appendix B. Two Port FTP

[RFC0959] already defines a mechanism - Block Mode over the default port - which does not require PORT or PASV to operate, and does not close connections between files. This means that under such a scheme, firewall administration is simpler - port 21 is open inbound, and port 20 on the server allows outbound connections.

Appendix C. Document History

[[to be removed by the RFC editor before publication as an RFC.]]

Known issues concerning this draft:

- <https://github.com/antbryan/internetdraft/issues>

draft-bryan-ftp-lock-02 : January 30, 2013.

- FTPEXT2 mailing list feedback.

draft-bryan-ftp-lock-01 : January 19, 2013.

- Draft had expired.

draft-bryan-ftp-lock-00 : June 7, 2011.

- Initial draft.

Authors' Addresses

Anthony Bryan
Pompano Beach, FL
USA

EMail: anthonybryan@gmail.com
URI: http://www.metalinker.org

Daniel Stenberg

EMail: daniel@haxx.se
URI: http://www.haxx.se/

Tatsuhiro Tsujikawa
Shiga
Japan

EMail: tatsuhiro.t@gmail.com
URI: http://aria2.sourceforge.net