Network Working Group
Request for Comments: 2428
Category: Standards Track

M. Allman
NASA Lewis/Sterling Software
S. Ostermann
Ohio University
C. Metz
The Inner Net

September 1998

FTP Extensions for IPv6 and NATs

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The specification for the File Transfer Protocol assumes that the underlying network protocol uses a 32-bit network address (specifically IP version 4). With the deployment of version 6 of the Internet Protocol, network addresses will no longer be 32-bits. This paper specifies extensions to FTP that will allow the protocol to work over IPv4 and IPv6. In addition, the framework defined can support additional network protocols in the future.

1. Introduction

The keywords, such as MUST and SHOULD, found in this document are used as defined in RFC 2119 [Bra97].

The File Transfer Protocol [PR85] only provides the ability to communicate information about IPv4 data connections. FTP assumes network addresses will be 32 bits in length. However, with the deployment of version 6 of the Internet Protocol [DH96] addresses will no longer be 32 bits long. RFC 1639 [Pis94] specifies extensions to FTP to enable its use over various network protocols. Unfortunately, the mechanism can fail in a multi-protocol environment. During the transition between IPv4 and IPv6, FTP needs the ability to negotiate the network protocol that will be used for data transfer.

This document provides a specification for a way that FTP can communicate data connection endpoint information for network protocols other than IPv4. In this specification, the FTP commands PORT and PASV are replaced with EPRT and EPSV, respectively. This document is organized as follows. Section 2 outlines the EPRT command and Section 3 outlines the EPSV command. Section 4 defines the utilization of these two new FTP commands. Section 5 briefly presents security considerations. Finally, Section 6 provides conclusions.

2. The EPRT Command

The EPRT command allows for the specification of an extended address for the data connection. The extended address MUST consist of the network protocol as well as the network and transport addresses. The format of EPRT is:

EPRT<space><d><net-prt><d><net-addr><d><tcp-port><d>

The EPRT command keyword MUST be followed by a single space (ASCII 32). Following the space, a delimiter character (<d>) MUST be specified. The delimiter character MUST be one of the ASCII characters in range 33-126 inclusive. The character "|" (ASCII 124) is recommended unless it coincides with a character needed to encode the network address.

The <net-prt> argument MUST be an address family number defined by IANA in the latest Assigned Numbers RFC (RFC 1700 [RP94] as of the writing of this document). This number indicates the protocol to be used (and, implicitly, the address length). This document will use two of address family numbers from [RP94] as examples, according to the following table:

AF Number	Protocol
1	Internet Protocol, Version 4 [Pos81a]
2	Internet Protocol, Version 6 [DH96]

The <net-addr> is a protocol specific string representation of the network address. For the two address families specified above (AF Number 1 and 2), addresses MUST be in the following format:

AF Number	Address Format	Example
1	dotted decimal	132.235.1.2
2	IPv6 string representations defined in [HD96]	1080::8:800:200C:417A

The <tcp-port> argument must be the string representation of the number of the TCP port on which the host is listening for the data connection.

The following are sample EPRT commands:

```
EPRT |1|132.235.1.2|6275|

EPRT |2|1080::8:800:200C:417A|5282|
```

The first command specifies that the server should use IPv4 to open a data connection to the host "132.235.1.2" on TCP port 6275. The second command specifies that the server should use the IPv6 network protocol and the network address "1080::8:800:200C:417A" to open a TCP data connection on port 5282.

Upon receipt of a valid EPRT command, the server MUST return a code of 200 (Command OK). The standard negative error code 500 and 501 [PR85] are sufficient to handle most errors (e.g., syntax errors) involving the EPRT command. However, an additional error code is needed. The response code 522 indicates that the server does not support the requested network protocol. The interpretation of this new error code is:

```
5yz Negative Completion
x2z Connections
xy2 Extended Port Failure - unknown network protocol
```

The text portion of the response MUST indicate which network protocols the server does support. If the network protocol is unsupported, the format of the response string MUST be:

```
<text stating that the network protocol is unsupported> \
    (prot1,prot2,...,protn)
```

Both the numeric code specified above and the protocol information between the characters '(' and ')' are intended for the software automata receiving the response; the textual message between the numeric code and the '(' is intended for the human user and can be any arbitrary text, but MUST NOT include the characters '(' and ')'. In the above case, the text SHOULD indicate that the network protocol in the EPRT command is not supported by the server. The list of protocols inside the parenthesis MUST be a comma separated list of address family numbers. Two example response strings follow:

```
Network protocol not supported, use (1)

Network protocol not supported, use (1,2)
```

3. The EPSV Command

The EPSV command requests that a server listen on a data port and wait for a connection. The EPSV command takes an optional argument. The response to this command includes only the TCP port number of the listening connection. The format of the response, however, is similar to the argument of the EPRT command. This allows the same parsing routines to be used for both commands. In addition, the format leaves a place holder for the network protocol and/or network address, which may be needed in the EPSV response in the future. The response code for entering passive mode using an extended address MUST be 229. The interpretation of this code, according to [PR85] is:

```
2yz Positive Completion
x2z Connections
xy9 Extended Passive Mode Entered
```

The text returned in response to the EPSV command MUST be:

```
<text indicating server is entering extended passive mode> \
    (<d><d><d><tcp-port><d>)
```

The portion of the string enclosed in parentheses MUST be the exact string needed by the EPRT command to open the data connection, as specified above.

The first two fields contained in the parenthesis MUST be blank. The third field MUST be the string representation of the TCP port number on which the server is listening for a data connection. The network protocol used by the data connection will be the same network protocol used by the control connection. In addition, the network address used to establish the data connection will be the same network address used for the control connection. An example response string follows:

```
Entering Extended Passive Mode (|||6446|)
```

The standard negative error codes 500 and 501 are sufficient to handle all errors involving the EPSV command (e.g., syntax errors).

When the EPSV command is issued with no argument, the server will choose the network protocol for the data connection based on the protocol used for the control connection. However, in the case of proxy FTP, this protocol might not be appropriate for communication between the two servers. Therefore, the client needs to be able to request a specific protocol. If the server returns a protocol that is not supported by the host that will be connecting to the port, the

client MUST issue an ABOR (abort) command to allow the server to close down the listening connection. The client can then send an EPSV command requesting the use of a specific network protocol, as follows:

EPSV<space><net-prt>

If the requested protocol is supported by the server, it SHOULD use the protocol. If not, the server MUST return the 522 error messages as outlined in section 2.

Finally, the EPSV command can be used with the argument "ALL" to inform Network Address Translators that the EPRT command (as well as other data commands) will no longer be used. An example of this command follows:

EPSV<space>ALL

Upon receipt of an EPSV ALL command, the server MUST reject all data connection setup commands other than EPSV (i.e., EPRT, PORT, PASV, et al.). This use of the EPSV command is further explained in section 4.

4. Command Usage

For all FTP transfers where the control and data connection(s) are being established between the same two machines, the EPSV command MUST be used. Using the EPSV command benefits performance of transfers that traverse firewalls or Network Address Translators (NATs). RFC 1579 [Bel94] recommends using the passive command when behind firewalls since firewalls do not generally allow incoming connections (which are required when using the PORT (EPRT) command). In addition, using EPSV as defined in this document does not require NATs to change the network address in the traffic as it is forwarded. The NAT would have to change the address if the EPRT command was used. Finally, if the client issues an "EPSV ALL" command, NATs may be able to put the connection on a "fast path" through the translator, as the EPRT command will never be used and therefore, translation of the data portion of the segments will never be needed. When a client only expects to do two-way FTP transfers, it SHOULD issue this command as soon as possible. If a client later finds that it must do a three-way FTP transfer after issuing an EPSV ALL command, a new FTP session MUST be started.

5. Security Issues

The authors do not believe that these changes to FTP introduce new security problems. A companion Work in Progress [AO98] is a more general discussion of FTP security issues and techniques to reduce these security problems.

6. Conclusions

The extensions specified in this paper will enable FTP to operate over a variety of network protocols.

References

[AO98] Allman, M., and S. Ostermann, "FTP Security Considerations", Work in Progress.

[Bel94] Bellovin, S., "Firewall-Friendly FTP", RFC 1579, February 1994.

[Bra97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[DH96] Deering, S., and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, December 1995.

[HD96] Hinden, R., and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

[Pis94] Piscitello, D., "FTP Operation Over Big Address Records (FOOBAR)", RFC 1639, June 1994.

[Pos81a] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[Pos81b] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[PR85] Postel, J., and J. Reynolds, "File Transfer Protocol (FTP)", STD 9, RFC 959, October 1985.

[RP94] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994. See also: http://www.iana.org/numbers.html

Authors' Addresses

Mark Allman
NASA Lewis Research Center/Sterling Software
21000 Brookpark Rd. MS 54-2
Cleveland, OH 44135

Phone: (216) 433-6586
EMail: mallman@lerc.nasa.gov
http://gigahertz.lerc.nasa.gov/~mallman/

Shawn Ostermann
School of Electrical Engineering and Computer Science
Ohio University
416 Morton Hall
Athens, OH 45701

Phone: (740) 593-1234
EMail: ostermann@cs.ohiou.edu

Craig Metz
The Inner Net
Box 10314-1954
Blacksburg, VA 24062-0314

Phone: (DSN) 754-8590
EMail: cmetz@inner.net

Full Copyright Statement