Behavior Engineering for Hindrance Avoidance
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2010

I. van Beijnum
IMDEA Networks

October 19, 2009

IPv6-to-IPv4 translation FTP considerations
draft-van-beijnum-behave-ftp64-06

Status of this Memo

Copyright Notice

Abstract

The File Transfer Protocol has a very long history, and despite the fact that today, other options exist to perform file transfers, FTP

is still in common use. As such, it is important that in the situation where some client computers are IPv6-only while many servers are still IPv4-only and IPv6-to-IPv4 translators are used to bridge that gap, FTP is made to work through these translators as best it can.

FTP has an active and a passive mode, both as original commands that are IPv4-specific, and as extended, IP version agnostic commands. The only FTP mode that works without changes through an IPv6-to-IPv4 translator is extended passive. However, many existing FTP servers don't support this mode, and some clients don't ask for it. This document describes server, client and middlebox (if any) behavior that minimizes this problem.

Table of Contents

1. Introduction

[RFC0959] specifies two modes of operation for FTP: active mode, in which the server connects back to the client, usually to a client-provided port number, and passive mode, where the server opens a port for the client to connect to. Without additional action, active mode with a client-supplied port doesn't work through NATs or firewalls. And in both cases, an IPv4 address is specified, making both the original passive mode and the original active mode incompatible with IPv6. These issues were solved in [RFC2428], which introduces the EPSV (extended passive) mode that only specifies a port number and the EPRT (extended port) command which allows the client to supply an IPv6 address (and a port) to the server.

A survey done in April of 2009 of 25 randomly picked and/or well-known FTP sites reachable over IPv4 showed that only 12 of them supported EPSV over IPv4. Additionally, only 2 of those 12 indicated that they supported EPSV in response to the FEAT command ([RFC2389]) that asks the server to list its supported features. One supported EPSV but not FEAT. In 5 cases, issuing the EPSV command to the server led to a significant delay, in 3 cases followed by a control channel reset. It appears that in these cases, the server did support EPSV but a middlebox didn't. All 25 servers were able to successfully complete a transfer in PASV traditional passive mode as required by [RFC1123]. More tests showed that the use of an address family argument with the EPSV command is widely mis- or unimplemented in servers. The additional tests with more servers showed that approximately 65% of FTP servers support EPSV successfully and around 96% support PASV successfully. Clients weren't extensively tested, but previous experience from the author suggests that most clients support PASV, with the notable exception of the command line client included with Windows, which only supports active mode. It uses the original PORT command when running over IPv4 and EPRT when running over IPv6.

Based on the tests, this document updates [RFC0959] and [RFC2428] in order to disallow certain unusual modes of operation that are incompatible with IPv6-to-IPv4 translation as well as requiring the EPSV capability from all servers. All IPv6 FTP clients are required to implement EPSV. Further, it is recommended that IPv6 clients retry with PASV when EPSV fails under the assumption that they are communicating through an IPv6-to-IPv4 translator.

Additionally, there are guidelines for operators choosing to implement an application layer gateway functionality to provide connectivity between unupdated servers and/or clients. It is not required to implement an ALG to conform to this specification. Clients that want to engage in more complex behavior, such as server-

to-server transfers, may make an FTP ALG go into transparent mode by issuing an AUTH command.

The requirements and recommendations in this document apply to all forms of IPv6-to-IPv4 translation, including stateless translation such as [RFC2765] and stateful translation such as [I-D.bagnulo-behave-nat64].

The FTP protocol allows for complex interactions, such as the situation where a client connects to two servers and directs the servers to exchange data between them. These types of interactions are out of scope for this document.

2. Notational Conventions

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 [RFC2119].

3. Terminology

Within the context of this document, the words "client" and "server" refer to FTP client and server implementations, respectively. An FTP server is understood to be an implementation of the FTP protocol running on a server, waiting for clients to connect and issue commands and start data transfers. Clients are pieces of software designed to interact with servers using the FTP protocol, and store (upload) or retrieve (download) files to/from one or more servers, either interactively under control of a user, or as an unattended background process. Most operating systems provide a web browser that implements a basic FTP client, as well as a command line client. Third-party FTP clients are also widely available.

Other terminology is derived from the documents listed in the reference section.

4. Server requirements

All FTP servers MUST support EPSV mode. Further, if a server allows configuration of any kind, it MUST also be configurable to respond with a 502 (command not implemented) error to EPSV and EPRT commands, even though the server is capable of EPSV and/or EPRT. This way, FTP servers residing behind firewalls or other middleboxes that break EPSV or EPRT functionality can be made to trigger clients to fall back to PASV or PORT immediately rather than potentially suffering a

timeout.

All FTP servers MUST only use the local address used for the control channel session in PASV responses. This allows an ALG to translate the response to the PASV command to an EPSV response without loss of information. Note that according to tests performed by Dan Wing, this requirement is already met in practice.

All FTP servers that support the FEAT command (which is highly RECOMMENDED for all servers) MUST indicate support for EPSV and/or EPRT when available in response to the FEAT command and MUST NOT list EPSV and/or EPRT in response to the FEAT command when EPSV and/or EPRT is administratively disabled as outlined above.

5. Client requirements

All FTP clients MUST support EPSV when communicating over IPv6.

It is highly RECOMMENDED that FTP clients react by retrying with PASV or EPRT when the EPSV command fails, either because of an error response by the server (40x, 42x, 50x and 52x responses), because the data connection couldn't be created or because the control channel session was terminated. In the latter two cases, a client MAY cache the name or address of the FTP server and issue PASV rather than EPSV in future sessions. In that case, the cache entry SHOULD be cleared if older than 7 days and the server indicates EPSV support in its FEAT response where it previously did not indicate EPSV support in its FEAT response. There is always a risk that an error was the result of a condition unrelated to IPv6-to-IPv4 translation. However, retrying with a PASV request has little potential for harm, so unless the error is clearly unrelated, retrying with PASV is the appropriate reaction.

When an FTP client is communicating over IPv6 and it is unable to use the EPSV (and possibly EPRT) command successfully, it SHOULD retry with PASV. However, the server will respond to the PASV command with an IPv4 address that the client must use to connect to for the data connection. Even if the client has IPv4 reachability, it SHOULD ignore the server-supplied address and set up a data connection towards the IPv6 address of the server that is used for the control channel session. However, the port number used for the data connection is taken from the 227 response to the PASV command.

Clients MUST NOT use any arguments with the EPSV command; many IPv4 FTP servers react adversely to "EPSV 1". Clients SHOULD assume that servers are unaware of the IP version used by clients. This may be the result from a server implementation that uses the updated IPv6

socket API with IPv4-mapped addresses, or because the server resides behind an IPv6-to-IPv4 translator.

6. ALG functionality

The author recognizes that FTP application layer gateways for compatibility with IPv6-to-IPv4 translators is rejected by many within the IETF community. As such, it is RECOMMENDED to update FTP clients and servers as required for IPv6-to-IPv4 translation support where possible, to allow proper operation of the FTP protocol without the need for ALGs.

On the other hand, network operators often have little influence over the FTP clients their customers run, let alone the FTP servers used throughout the Internet. For those operators, deploying an ALG may be the only way to provide a satisfactory customer experience. So, even though not the preferred solution, this document describes the functionality of such an ALG in order to promote consistent behavior between ALGs in an effort to minimize their harmful effects. However, the situation with regard to FTP server and -clients, especially in IPv6-heavy deployments, may change fast, so within relatively little time it may become feasible to stop running an ALG. Operators are encouraged to keep revisiting the issue.

Note that the translation of EPSV through all translators and EPRT through a stateless translator is relatively simple and translation of EPRT through a stateful translator relatively difficult. As such, an ALG used with a stateful translator MAY choose to support only EPSV. However, an ALG used with a stateless translator SHOULD also support EPRT.

6.1. Control channel translation

The IPv6-to-IPv4 FTP ALG intercepts all TCP sessions towards IPv4 port 21 destinations. The FTP ALG implements the Telnet protocol ([RFC0854]) used for control channel interactions to the degree necessary to interpret commands and responses and re-issue those commands and responses, modifying them as outlined below. Option negotiation attempts by either the client or the server, except for those allowed by [RFC1123], SHOULD be rejected by the FTP ALG without relaying those attempts. This avoids the situation where the client and the server negotiate options unknown to the FTP ALG.

There are two ways to implement the control channel ALG:

1. The ALG terminates the IPv6 TCP session, sets up a new IPv4 TCP session towards the IPv4 FTP server, and relays commands and

responses back and forth between the two sessions.

2. Packets that are part of the control channel are translated individually.

In the second case, an implementation must have the ability to track and update TCP sequence numbers when translating packets and break up packets into smaller packets after translation, as the control channel translation may modify the length of the payload portion of the packets in question. Also, FTP commands/responses or Telnet negotiations may straddle packet boundaries, so in order to be able to perform the ALG function, it may be necessary to reconstitute Telnet negotiations and FTP commands and responses from multiple packets.

If the client issues the AUTH command the client is attempting to negotiate [RFC2228] security mechanisms which are likely to be incompatible with the FTP ALG function. In this situation, the FTP ALG MUST switch to transparently forwarding all data on the control channel in both directions until the end of the control channel session. This requirement applies regardless of the response from the server. I.e., it is the fact that the client attempts the AUTH negotiation that requires the ALG to become transparent, not whether or not the attempt is successful.

There have been FTP ALGs for the purpose of making active FTP work through IPv4 NATs for a long time. Another type of ALG would be one that imposes restrictions required by security policies. Multiple ALGs can be implemented as a single entity. Should such a multi-purpose ALG forbid the use of the AUTH command for policy reasons, the side effect of making the ALG stop performing the translations described here, as well as other possible interventions related to IPv6-to-IPv4 translation, MUST be retained even if the ALG responds to the AUTH command with an error and doesn't propagate the command to the server. (Implementers are further advised that IPv6 hosts using an IPv6-to-IPv4 translator will normally have the ability to execute FTP over IPv6 without interference from the ALG.)

6.2. EPSV to PASV translation

Although many IPv4 FTP servers support the EPSV command, some servers react adversely to this command, and there is no reliable way to detect in advance that this will happen. As such, an FTP ALG MAY translate all occurrences of the EPSV command issued by the client to the PASV command, and reformat a 227 response as a corresponding 229 response.

For instance, if the client issues EPSV (or EPSV 2 to indicate IPv6

as the network protocol), this is translated to the PASV command. If the server with address 192.0.2.31 then responds with:

227 Entering Passive Mode (192,0,2,31,237,19)

The FTP ALG reformats this as:

229 Entering Extended Passive Mode (|||60691|)

If the server's 227 response contains an IPv4 address that doesn't match the destination of the control channel, the FTP ALG SHOULD send the following response to the client:

425 Can't open data connection.

It is important that the response is in the 4xx range to indicate a temporary condition.

If the client issues an EPSV command with a numeric argument other than 2, the ALG MUST NOT pass the command on to the server, but rather respond with a 522 error.

If the client issues EPSV ALL, the FTP ALG MUST NOT pass this command to the server, but respond with:

202 Command not implemented.

This avoids the situation where an FTP server may react adversely to receiving a PASV command after the client indicated that it will only use EPSV during this session.

6.3. EPRT to PORT translation

Should the IPv6 client issue an EPRT command, the FTP ALG MAY translate this EPRT command to a PORT command. The translation is different depending on whether the translator is a stateless one-to-one translator or a stateful one-to-many translator.

6.3.1. Stateless EPRT translation

If the address specified in the EPRT command is the client's IPv6 address, then the FTP ALG reformats the EPRT command into a PORT command with the IPv4 address that maps to the client's IPv6 address. The port number MUST be preserved for compatibility with stateless translators.

If the address specified in the EPRT command is not the client's IPv4 address, the ALG's response is undefined. It may pass along the

command unchanged, respond with an error, or attempt to perform an appropriate translation.

6.3.2. Stateful EPRT translation

If the address in the EPRT command is the IPv6 address of the control channel client's address, the stateful translator selects an unused port number in combination with the IPv4 address used for the control channel towards the FTP server, and sets up a mapping from that transport address to the one specified by the client in the EPRT command. The PORT command with the IPv4 address and port used on the IPv4 side of the mapping is only issued towards the server once the mapping is created. Initially, the mapping is such that either any transport address or the FTP server's IPv4 address with any port number is accepted as a source, but once the three-way handshake is complete, the mapping is narrowed to only match the negotiated TCP session.

If the address in the EPRT command is not the client's IPv6 address, the ALG's response is undefined.

6.4. Default port 20 translation

If the client doesn't issue an EPSV/PASV or EPRT/PORT command, it is invoking the default active FTP behavior where the server sets up a TCP session towards the client. In this situation, the source port number is the default FTP data port (port 20) and the destination port is the port the client uses as the source port in the control channel session.

In the case of a stateless translator, this doesn't pose any problems. In the case of a stateful translator, the translator SHOULD accept incoming connection requests from the server on the IPv4 side if the transport addresses match that of an existing FTP control channel session, with the exception that the control channel session uses port 21 and the new session port 20. In this case, a mapping is set up towards the same transport address on the IPv6 side that is used for the matching FTP control channel session.

So for instance, the client is 2001:db8:31::6 and the server is 192.0.2.4. The translator has prefix 2001:db8:ffff:fffff::/96 as its translator prefix and 10.0.0.1 as its IPv4 address. On the IPv6 side, the transport addresses for an FTP control channel session could then be 2001:db8:31::6,49152 to 2001:db8:ffff:ffff::c000:204,21 on the IPv6 side and 10.0.0.1,60000 to 192.0.2.4,21 on the IPv4 side. If then the FTP server initiates a session from 192.0.2.4,20 to 10.0.0.1,60000, the translator sets up a mapping from those addresses to source 2001:db8:ffff:ffff::c000:204,20 destination 2001:db8:31::

6,49152.

If there is no (unambiguous) match for an existing data channel session when an incoming session request on port 20 arrives, the connection is refused with a TCP RST.

6.5. Both PORT and PASV

[RFC0959] allows a client to issue both PORT and PASV to use non-default ports on both sides of the connection. However, this is incompatible with the notion that with PASV the data connection is made from the client to the server, while PORT reaffirms the default behavior where the server connects to the client. As such, the behavior of an ALG is undefined when a client issues both PASV and PORT.

6.6. Timeouts

Wherever possible, control channels SHOULD NOT time out while there is an active data channel. A timeout of at least 30 seconds is recommended for mappings created by the FTP ALG that are waiting for initial packets.

Whenever a command from the client isn't propagated to the server, the FTP ALG instead issues a NOOP command in order to keep the keepalive state between the client and the server synchronized. The response to the NOOP command is not sent back to the client.

7. IANA considerations

None.

8. Security considerations

In the majority of cases, FTP is used without further security mechanisms. This allows a passive attacker to obtain the login credentials, and an attacker that can modify packets to change the data transferred. However, FTP can be used with TLS in order to solve these issues. IPv6-to-IPv4 translation and the FTP ALG don't impact the security issues in the former case nor the use of TLS in the latter case. However, if FTP is used with TLS or another authentication mechanism, the ALG function is not performed so only passive transfers from a server that implements EPSV or a client that supports PASV will succeed.

9. References

9.1. Normative References

[RFC0854] Postel, J. and J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, May 1983.

[RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[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.

[RFC2228] Horowitz, M., "FTP Security Extensions", RFC 2228, October 1997.

[RFC2428] Allman, M., Ostermann, S., and C. Metz, "FTP Extensions for IPv6 and NATs", RFC 2428, September 1998.

9.2. Informative References

[RFC2765] Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.

[I-D.bagnulo-behave-nat64] Bagnulo, M., Matthews, P., and I. Beijnum, "NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-bagnulo-behave-nat64-03 (work in progress), March 2009.

[I-D.liu-behave-ftp64] Liu, D. and Z. Cao, "IPv6 IPv4 translation FTP considerations", draft-liu-behave-ftp64-03 (work in progress), August 2009.

Appendix A. Acknowledgements

Kentaro Ebisawa, Remi Denis-Courmont, Mayuresh Bakshi, Sarat Kamisetty, Reinaldo Penno, Alun Jones, Dave Thaler, Mohammed Boucadair, Mikael Abrahamsson and Dapeng Liu contributed ideas and comments. Dan Wing ran experiments with a large number of FTP

servers that were very illuminating; many of the choices underlying this document are based on his results. Versions -05 and -06 of this document adopts several important design decisions from [I-D.liu-behave-ftp64].

Iljitsch van Beijnum is partly funded by Trilogy, a research project supported by the European Commission under its Seventh Framework Program.

Appendix B. Document and discussion information

The latest version of this document will always be available at http://www.muada.com/drafts/. Please direct questions and comments to the BEHAVE or apps area mailinglists or directly to the author.

Author's Address

Iljitsch van Beijnum
IMDEA Networks
Avda. del Mar Mediterraneo, 22
Leganes, Madrid 28918
Spain

Email: iljitsch@muada.com