Behavior Engineering for Hindrance Avoidance	I. van Beijnum IMDEA Networks
Internet-Draft	September 1, 2010
Intended status: Standards Track	
Expires: March 5, 2011	

TOC

An FTP ALG for IPv6-to-IPv4 translation
draft-ietf-behave-ftp64-05

Abstract

The File Transfer Protocol (FTP) 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 do not support this mode, and some clients do not ask for it. This document describes specifies a middlebox that may solve this mismatch.

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 March 5, 2011.

Copyright Notice

Table of Contents

---

1. Introduction

TOC

[RFC0959] (Postel, J. and J. Reynolds, “File Transfer Protocol,” October 1985.) specifies two modes of operation for FTP: active mode, in which the server connects back to the client 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 does not work through NATs or firewalls. With active mode, the PORT command has an IPv4 address as its argument, and in passive mode, the server responds to the PASV command with an IPv4 address. This makes both the passive and active modes as originally specified in [RFC0959] (Postel, J. and J. Reynolds, “File Transfer Protocol,” October 1985.) incompatible with IPv6. These issues were solved in [RFC2428] (Allman, M., Ostermann, S., and C. Metz, “FTP Extensions for IPv6 and NATs,” September 1998.), which introduces the EPSV

(extended passive) command, where the server only responds with a port number, and the EPRT (extended port) command, which allows the client to supply either an IPv4 or 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 introduced in [RFC2389] (Hethmon, P. and R. Elz, "Feature negotiation mechanism for the File Transfer Protocol," August 1998.) 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. All 25 servers were able to successfully complete a transfer in traditional passive PASV mode as required by [RFC1123] (Braden, R., "Requirements for Internet Hosts - Application and Support," October 1989.). 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 were not 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. This FTP client uses the original PORT command when running over IPv4 and EPRT when running over IPv6.

Although these issues can and should be addressed by modifying, where necessary, clients and servers support EPSV successfully, such modifications may not appear widely in a timely fashion. Also, network operators who may want to deploy IPv6-to-IPv4 translation generally don't have control over client or server implementations. As such, this document standardizes an FTP Application Layer Gateway that will allow unmodified IPv6 FTP clients to interact with unmodified IPv4 FTP servers successfully when using FTP for simple file transfers between a single client and a single server.

Clients that want to engage in more complex behavior, such as server-to-server transfers, may make an FTP application layer gateway (ALG) go into transparent mode by issuing an AUTH or ALGS commands as explained in Section 5 (Control channel translation).

The recommendations and specifications in this document apply to all forms of IPv6-to-IPv4 translation, including stateless translation such as [RFC2765] (Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)," February 2000.) or [I-D.ietf-behave-v6v4-xlate] (Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm," December 2009.) as well as stateful translation such as [I-D.ietf-behave-v6v4-xlate-stateful] (Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers," July 2010.).

TOC

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] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

Where these words appear in lower case, they SHOULD NOT be interpreted within the context of [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.), but rather, in accordance with regular English usage.

---

3. Terminology

TOC

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 system with a stable address, waiting for clients to connect and issue commands that eventually start data transfers. Clients interact with servers using the FTP protocol, and store (upload files) and retrieve (download files) to and from one or more servers. This either happens interactively under control of a user, or is done 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. Note that this document cannot be fully understood on its own; it depends on background and terminology outlined in the references.

---

4. ALG overview

TOC

The most robust way to solve an IP version mismatch between FTP clients and FTP servers would be by changing clients and servers rather than using an IPv6-to-IPv4 translator for the data channel and using an application layer gateway on the control channel. 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 standardizes the functionality of such an ALG in order to promote

consistent behavior between ALGs in an effort to minimize their harmful effects.

Operators are encouraged to only deploy an FTP ALG for IPv6-to-IPv4 translation when the FTP ALG is clearly needed. In the presence of the ALG, EPSV commands that could be handled directly by conforming servers are translated into PASV commands, introducing unnecessary complexity and reducing robustness. As such a "set and forget" policy on ALGs is not recommended.

Note that the translation of EPSV through all translators and EPRT through a stateless translator is relatively simple but supporting translation of EPRT through a stateful translator is relatively difficult, because in the latter case a translation mapping must be set up for each data transfer using parameters that must be learned from the client/server interaction on the control channel. This needs to happen before the EPRT command can be translated into a PORT command and passed on to the server. As such, an ALG used with a stateful translator MUST support EPSV and MAY support EPRT. However, an ALG used with a stateless translator SHOULD also support EPRT.

The ALG functionality is described as a function separate from the IPv6-to-IPv4 translation function. However, in the case of EPRT translation, the ALG and translator functions need to be tightly coupled, so if EPRT translation is supported, it is assumed that the ALG and IPv6-to-IPv4 translation functions are integrated within a single device.

---

5. Control channel translation

TOC

The IPv6-to-IPv4 FTP ALG intercepts all TCP sessions towards port 21 for IPv6 destination addresses that map to IPv4 destinations reachable through an IPv6-to-IPv4 translator. The FTP ALG implements the Telnet protocol ([RFC0854] (Postel, J. and J. Reynolds, “Telnet Protocol Specification,” May 1983.)) 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. Telnet option negotiation attempts by either the client or the server, except for those allowed by [RFC1123] (Braden, R., “Requirements for Internet Hosts - Application and Support,” October 1989.), MUST be rejected by the FTP ALG without relaying those attempts. This avoids the situation where the client and the server negotiate Telnet options that are unimplemented by 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.

As they ultimately provide the same result either implementation strategy (or any other that is functionally equivalent) MAY be used.

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 could modify the length of the payload portion of the packets in question. Also, FTP commands/responses or Telnet negotiations could straddle packet boundaries, so in order to be able to perform the ALG function, it can prove necessary to reconstitute Telnet negotiations and FTP commands and responses from multiple packets.

If the client issues the AUTH command, then the client is attempting to negotiate [RFC2228] (Horowitz, M., “FTP Security Extensions,” October 1997.) 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. In other words, it is the fact that the client attempts the AUTH negotiation that requires the ALG to become transparent, whether or not the attempt is successful. The transparency requirement applies to the commands and responses flowing between the client and the server. It is possible that commands or responses that were sent through the ALG before the AUTH command was issued were changed in length so TCP sequence numbers in packets entering the ALG and packets exiting the ALG no longer match. In transparent mode, the ALG MUST continue to adjust sequence numbers if it was doing so before entering transparent mode as the result of the AUTH command. The ALGS command (Section 11 (The ALGS command)) MAY have a similar effect as the AUTH command, depending on the argument used.

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 of different types can be implemented as a single entity. If such a multi-purpose ALG forbids 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 does not propagate the command to the server. This way, any time a client issues the AUTH command, it knows that an ALG will be in transparent mode afterwards. Implementers are further advised that unlike hosts behind an IPv4 NAT, IPv6 hosts using an IPv6-to-IPv4 translator will normally have the ability to execute FTP over IPv6 without interference from the IPv6-to-IPv4 translator or the ALG, so an IPv6-to-IPv4 translation FTP ALG is not the best place to implement security policies.

TOC

6. 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 MUST 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. However, an ALG MAY forego EPSV to PASV translation if it has positive knowledge, either through administrative configuration or learned dynamically, that EPSV will be successful without translation to PASV.

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|)

The ALG SHOULD ignore the IPv4 address in the server's 227 response. This is the behavior that is exhibited by most clients and is needed to work with servers that include [RFC1918] (Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, “Address Allocation for Private Internets,” February 1996.) addresses in their 227 responses. However, if the 227 response contains an IPv4 address that does not match the destination of the control channel, the FTP ALG MAY send the following response to the client instead of the 229 response:

> 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:

> 522 Network protocol not supported

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

> 504 Command not implemented for that parameter

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

7. 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.

7.1. Stateless EPRT translation

TOC

If the address specified in the EPRT command is the IPv6 address used by the client for the control channel session, 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. For instance, if the client with IPv6 address 2001:db8:2::31 issues the following EPRT command:

```
EPRT |2|2001:db8:2::31|5282|
```

Assuming the IPv4 address that goes with 2001:db8:2::31 is 192.0.2.31, the FTP ALG reformats this as:

```
PORT 192,0,2,31,20,162
```

If the address specified in the EPRT command is an IPv4 address or an IPv6 address that is not the IPv6 address used by the client for the control session, the ALG SHOULD NOT attempt any translation, but pass along the command unchanged.

7.2. Stateful EPRT translation

TOC

If the address in the EPRT command is the IPv6 address used by the client for the control channel, 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 SHOULD be narrowed to only match the negotiated TCP session.

If the address specified in the EPRT command is an IPv4 address or an IPv6 address that is not the IPv6 address used by the client for the control session, the ALG SHOULD NOT attempt any translation, but pass along the command unchanged.

If the client with IPv6 address 2001:db8:2::31 issues the EPRT command:

```
EPRT |2|2001:db8:2::31|5282|
```

And the stateful translator uses the address 192.0.2.31 on its IPv4 interface, a mapping with destination address 192.0.2.31 and destination port 60192 towards 2001:db8:2::31 port 5282 may be created, after which the FTP ALG reformats the EPRT command as:

```
PORT 192,0,2,31,235,32
```

---

8. Default port 20 translation

TOC

If the client does not issue an EPSV/PASV or EPRT/PORT command prior to initiating a file transfer, 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 for the control channel session.

In the case of a stateless translator, this does not pose any problems. In the case of a stateful translator, the translator MAY 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.

Note that if multiple clients are using the same IPv6-to-IPv4 translator to communicate with the same FTP server, and for each client the IPv6-to-IPv4 translator uses the same source address on its IPv4 side, it may not be possible to correlate incoming FTP data channel sessions with the intended IPv6 host unambiguously. In such cases, the IPv6-to-IPv4 translator SHOULD refuse the connection establishment attempt by returning a TCP reset packet. An ALG/ translator MAY monitor the progress of FTP control channels and only attempt to perform a mapping when an FTP client has started a file transfer without issuing the EPSV, PASV, EPRT or PORT commands.

---

9. Both PORT and PASV

TOC

[RFC0959] (Postel, J. and J. Reynolds, “File Transfer Protocol,” October 1985.) 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. Implementations SHOULD NOT try to detect the situation where both PASV and PORT commands are issued prior to a command that initiates a transfer, but rather, apply the same translation they would have if there had not been a PASV command prior to a PORT command or a PORT command prior to a PASV command.

10. Default behavior

TOC

Whenever the client issues a command which the ALG is not set up to translate, either because the command is not specified in this document, the command is not part of any FTP specification, the ALG functionality is disabled administratively for the command in question, or translation does not apply for any other reason, the command MUST be passed on to the server without modification, and the server response MUST be passed on to the client without modification. For example, if the client issues the PASV command, this command is passed on to the server transparently, and the server's response to the client.

11. The ALGS command

TOC

ALGs SHOULD support the new ALGS (ALG status) command that allows clients to query and set the ALG's status. Note that this command MUST NOT be implemented in FTP servers. If those recognize the command, the best course of action would be to return a 202 response:

> 202 Command not implemented, superfluous at this site

However, as FTP servers don't implement the command, there is no reason for them to specifically recognize this command, and returning any 50x response that is normally returned when commands are not recognized is appropriate. A client can use the ALGS command to request the ALG's status and to enable and disable EPSV to PASV and, if implemented, EPRT to PORT translation. There are three possible arguments to the ALGS command:

> **ALGS STATUS64** The ALG is requested to return the EPSV and EPRT translation status.
>
> **ALGS ENABLE64** The ALG is requested to enable translation.
>
> **ALGS DISABLE64** The ALG is requested to disable translation.

The ALG SHOULD enable or disable translation as requested. If EPRT to PORT translation is supported, ALGS ENABLE64 enables it and ALGS DISABLE64 disables it along with enabling or disabling EPSV to PASV translation. If EPRT to PORT translation is not supported, ALGS ENABLE64 only enables EPSV to PASV translation. After an ALGS command with any of the three supported arguments, the ALG returns a

216 response indicating the type of translation that will be performed. There are four possible keywords that follow the 216 response code:

> **216 NONE** Neither EPSV nor EPRT translation is performed.
>
> **216 EPSV** EPSV is translated to PASV, no EPRT translation is performed.
>
> **216 EPRT** EPRT is translated to PORT, no EPSV translation is performed.
>
> **216 EPSVEPRT** EPSV is translated to PASV, EPRT is translated to PORT.

The translation type MAY be followed by a space and additional descriptive text until end-of-line. Failure to set the requested translation mode is not an error condition, and is thus indicated by the applicable keyword following the 216 response, and not with an error code response.

If the ALGS command is not implemented, the command SHOULD be passed on to the server without modification. If there is no argument to the ALGS command, or the argument is not one of STATUS64, ENABLE64 or DISABLE64 (or an argument specified by a supported newer document), a 504 error SHOULD be returned.

The Augmented Backus-Naur Form (ABNF) notation (see [RFC5234] (Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," January 2008.)) of the ALGS command and its response are as follows:

```
  algs-command      = "ALGS" SP algs-token CRLF
  algs-token        = "STATUS64" / "ENABLE64" / "DISABLE64"
                      / algs-ext-token
  algs-ext-token    = *VCHAR

  algs-response     = (ok-response / error-response) CRLF
  ok-response       = "216" SP response-token [ freetext ]
  response-token    = "NONE" / "EPSV" / "EPRT" / "EPSVEPRT"
  error-response    = not-implemented / invalid-parameter
  not-implemented   = "502" [ freetext ]
  invalid-parameter = "504" [ freetext ]
  freetext          = (SP *VCHAR)
```

---

12. Timeouts and translating to NOOP

TOC

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 data channel mappings created by the FTP ALG that are waiting for initial packets.

Whenever a command from the client is not 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 MUST NOT be relayed back to the client. An implementation MAY wait for the server to return the 200 response to the NOOP and translate that 200 response into the response the ALG is required to return to the client. This way, the ALG never has to create new packets to send to the client, but it can limit itself to modifying packets transmitted by the server. If the server responds with something other than 200 to the NOOP command, the ALG MUST tear down the control channel session and log an error.

---

13. IANA considerations

TOC

IANA is requested to add to the FTP Commands and Extensions registry the following entry:

> **Command Name** ALGS
>
> **FEAT Code** -N/A-
>
> **Description** FTP64 ALG status
>
> **Command Type** -N/A-
>
> **Conformance Requirements** o
>
> **Reference** RFC TBD Section 11 (The ALGS command)

[TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/ftp-commands-extensions/ftp-commands-extensions.xhtml]

---

14. Security considerations

TOC

In the majority of cases, FTP is used without further security mechanisms. This allows an attacker with passive interception capabilities 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 do not 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.

For general FTP security considerdations, see [RFC2577] (Allman, M. and S. Ostermann, “FTP Security Considerations,” May 1999.).

15. Contributors

TOC

Dan Wing, Kentaro Ebisawa, Remi Denis-Courmont, Mayuresh Bakshi, Sarat Kamisetty, Reinaldo Penno, Alun Jones, Dave Thaler, Mohammed Boucadair, Mikael Abrahamsson, Dapeng Liu, Michael Liu, Andrew Sullivan, Anthony Bryan and Ed Jankiewicz 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. This document adopts several design decisions from [I-D.liu-behave-ftp64] (Liu, D. and Z. Cao, "IPv6 IPv4 translation FTP considerations," August 2009.).

16. Acknowledgements

TOC

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

17. References

TOC

17.1. Normative References

TOC

[RFC0854]	Postel, J. and J. Reynolds, "Telnet Protocol Specification," STD 8, RFC 854, May 1983 (TXT).
[RFC0959]	Postel, J. and J. Reynolds, "File Transfer Protocol," STD 9, RFC 959, October 1985 (TXT).
[RFC1123]	Braden, R., "Requirements for Internet Hosts - Application and Support," STD 3, RFC 1123, October 1989 (TXT).
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2228]	Horowitz, M., "FTP Security Extensions," RFC 2228, October 1997 (TXT, HTML, XML).
[RFC2428]	Allman, M., Ostermann, S., and C. Metz, "FTP Extensions for IPv6 and NATs," RFC 2428, September 1998 (TXT, HTML, XML).
[RFC5234]	Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," STD 68, RFC 5234, January 2008 (TXT).

17.2. Informative References

TOC

[RFC1918]	Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets," BCP 5, RFC 1918, February 1996 (TXT).
[RFC2389]	Hethmon, P. and R. Elz, "Feature negotiation mechanism for the File Transfer Protocol," RFC 2389, August 1998 (TXT, HTML, XML).
[RFC2577]	Allman, M. and S. Ostermann, "FTP Security Considerations," RFC 2577, May 1999 (TXT).
[RFC2765]	Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)," RFC 2765, February 2000 (TXT).
[I-D.ietf-behave-v6v4-xlate-stateful]	Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers," draft-ietf-behave-v6v4-xlate-stateful-12 (work in progress), July 2010 (TXT).
[I-D.ietf-behave-v6v4-xlate]	Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm," draft-ietf-behave-v6v4-xlate-05 (work in progress), December 2009 (TXT).
[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 (TXT).
[Bernstein]	Bernstein, D., "PASV security and PORT security," 2000.

Author's Address

TOC

	Iljitsch van Beijnum
	IMDEA Networks
	Avda. del Mar Mediterraneo, 22
	Leganes, Madrid 28918
	Spain
Email:	iljitsch@muada.com