Internet Engineering Task Force (IETF)
Request for Comments: 6156
Category: Standards Track
ISSN: 2070-1721

G. Camarillo
O. Novo
Ericsson
S. Perreault, Ed.
Viagenie

April 2011

Traversal Using Relays around NAT (TURN) Extension for IPv6

Abstract

This document adds IPv6 support to Traversal Using Relays around NAT (TURN). IPv6 support in TURN includes IPv4-to-IPv6, IPv6-to-IPv6, and IPv6-to-IPv4 relaying. This document defines the REQUESTED-ADDRESS-FAMILY attribute for TURN. The REQUESTED-ADDRESS-FAMILY attribute allows a client to explicitly request the address type the TURN server will allocate (e.g., an IPv4-only node may request the TURN server to allocate an IPv6 address).

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6156.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Traversal Using Relays around NAT (TURN) [RFC5766] is a protocol that allows for an element behind a NAT to receive incoming data over UDP or TCP. It is most useful for elements behind NATs without Endpoint-Independent Mapping [RFC4787] that wish to be on the receiving end of a connection to a single peer.

The base specification of TURN [RFC5766] only defines IPv4-to-IPv4 relaying. This document adds IPv6 support to TURN, which includes IPv4-to-IPv6, IPv6-to-IPv6, and IPv6-to-IPv4 relaying. This document defines the REQUESTED-ADDRESS-FAMILY attribute, which is an extension to TURN that allows a client to explicitly request the address type the TURN server will allocate (e.g., an IPv4-only node may request the TURN server to allocate an IPv6 address). This document also defines and registers new error response codes.

2. Terminology

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. Overview of Operation

When a user wishes a TURN server to allocate an address of a specific type, it sends an Allocate request to the TURN server with a REQUESTED-ADDRESS-FAMILY attribute. TURN can run over UDP and TCP, and it allows for a client to request address/port pairs for receiving both UDP and TCP.

After the request has been successfully authenticated, the TURN server allocates a transport address of the type indicated in the REQUESTED-ADDRESS-FAMILY attribute. This address is called the relayed transport address.

The TURN server returns the relayed transport address in the response to the Allocate request. This response contains an XOR-RELAYED-ADDRESS attribute indicating the IP address and port that the server allocated for the client.

TURN servers allocate a single relayed transport address per allocation request. Therefore, Allocate requests cannot carry more than one REQUESTED-ADDRESS-FAMILY attribute. Consequently, a client that wishes to allocate more than one relayed transport address at a TURN server (e.g., an IPv4 and an IPv6 address) needs to perform several allocation requests (one allocation request per relayed transport address).

A TURN server that supports a set of address families is assumed to be able to relay packets between them. If a server does not support the address family requested by a client, the server returns a 440 (Address Family not Supported) error response.

4. Creating an Allocation

The behavior specified here affects the processing defined in Section 6 of [RFC5766].

4.1. Sending an Allocate Request

A client that wishes to obtain a relayed transport address of a specific address type includes a REQUESTED-ADDRESS-FAMILY attribute, which is defined in Section 4.1.1, in the Allocate request that it sends to the TURN server. Clients MUST NOT include more than one REQUESTED-ADDRESS-FAMILY attribute in an Allocate request. The mechanisms to formulate an Allocate request are described in Section 6.1 of [RFC5766].

Clients MUST NOT include a REQUESTED-ADDRESS-FAMILY attribute in an Allocate request that contains a RESERVATION-TOKEN attribute.

4.1.1. The REQUESTED-ADDRESS-FAMILY Attribute

The REQUESTED-ADDRESS-FAMILY attribute is used by clients to request the allocation of a specific address type from a server. The following is the format of the REQUESTED-ADDRESS-FAMILY attribute. Note that TURN attributes are TLV (Type-Length-Value) encoded, with a 16-bit type, a 16-bit length, and a variable-length value.

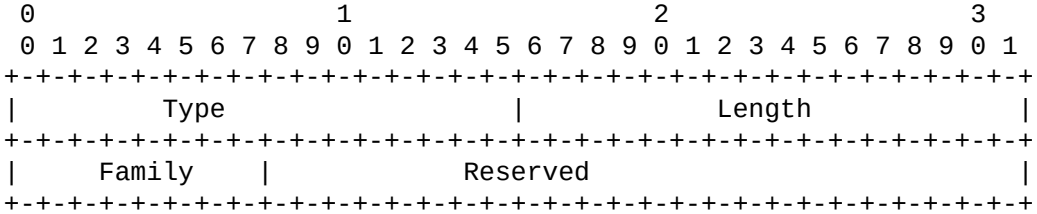

Figure 1: Format of REQUESTED-ADDRESS-FAMILY Attribute

Type: the type of the REQUESTED-ADDRESS-FAMILY attribute is 0x0017. As specified in [RFC5389], attributes with values between 0x0000 and 0x7FFF are comprehension-required, which means that the client or server cannot successfully process the message unless it understands the attribute.

Length: this 16-bit field contains the length of the attribute in bytes. The length of this attribute is 4 bytes.

Family: there are two values defined for this field and specified in [RFC5389], Section 15.1: 0x01 for IPv4 addresses and 0x02 for IPv6 addresses.

Reserved: at this point, the 24 bits in the Reserved field MUST be set to zero by the client and MUST be ignored by the server.

The REQUEST-ADDRESS-TYPE attribute MAY only be present in Allocate requests.

4.2. Receiving an Allocate Request

Once a server has verified that the request is authenticated and has not been tampered with, the TURN server processes the Allocate request. If it contains both a RESERVATION-TOKEN and a REQUESTED-ADDRESS-FAMILY, the server replies with a 400 (Bad Request) Allocate error response. Following the rules in [RFC5389], if the server does not understand the REQUESTED-ADDRESS-FAMILY attribute, it generates an Allocate error response, which includes an ERROR-CODE attribute with 420 (Unknown Attribute) response code. This response will contain an UNKNOWN-ATTRIBUTE attribute listing the unknown REQUESTED-ADDRESS-FAMILY attribute.

If the server can successfully process the request, it allocates a transport address for the TURN client, called the relayed transport address, and returns it in the response to the Allocate request.

As specified in [RFC5766], the Allocate response contains the same transaction ID contained in the Allocate request, and the XOR-RELAYED-ADDRESS attribute is set to the relayed transport address.

The XOR-RELAYED-ADDRESS attribute indicates the allocated IP address and port. It is encoded in the same way as the XOR-MAPPED-ADDRESS [RFC5389].

If the REQUESTED-ADDRESS-FAMILY attribute is absent, the server MUST allocate an IPv4-relayed transport address for the TURN client. If allocation of IPv4 addresses is disabled by local policy, the server returns a 440 (Address Family not Supported) Allocate error response.

If the server does not support the address family requested by the client, it MUST generate an Allocate error response, and it MUST include an ERROR-CODE attribute with the 440 (Address Family not Supported) response code, which is defined in Section 4.2.1.

4.2.1. Unsupported Address Family

This document defines the following new error response code:

440 (Address Family not Supported): The server does not support the address family requested by the client.

4.3. Receiving an Allocate Error Response

If the client receives an Allocate error response with the 440 (Unsupported Address Family) error code, the client MUST NOT retry its request.

5. Refreshing an Allocation

The behavior specified here affects the processing defined in Section 7 of [RFC5766].

5.1. Sending a Refresh Request

To perform an allocation refresh, the client generates a Refresh Request as described in Section 7.1 of [RFC5766]. The client MUST NOT include any REQUESTED-ADDRESS-FAMILY attribute in its Refresh Request.

5.2. Receiving a Refresh Request

If a server receives a Refresh Request with a REQUESTED-ADDRESS-FAMILY attribute, and the attribute's value doesn't match the address family of the allocation, the server MUST reply with a 443 (Peer Address Family Mismatch) Refresh error response.

6. CreatePermission

The behavior specified here affects the processing defined in Section 9 of [RFC5766].

6.1. Sending a CreatePermission Request

The client MUST only include XOR-PEER-ADDRESS attributes with addresses of the same address family as that of the relayed transport address for the allocation.

6.2. Receiving a CreatePermission Request

If an XOR-PEER-ADDRESS attribute contains an address of an address family different than that of the relayed transport address for the allocation, the server MUST generate an error response with the 443 (Peer Address Family Mismatch) response code, which is defined in Section 6.2.1.

6.2.1. Peer Address Family Mismatch

This document defines the following new error response code:

443 (Peer Address Family Mismatch): A peer address was of a different address family than that of the relayed transport address of the allocation.

7. Channels

The behavior specified here affects the processing defined in Section 11 of [RFC5766].

7.1. Sending a ChannelBind Request

The client MUST only include an XOR-PEER-ADDRESS attribute with an address of the same address family as that of the relayed transport address for the allocation.

7.2. Receiving a ChannelBind Request

If the XOR-PEER-ADDRESS attribute contains an address of an address family different than that of the relayed transport address for the allocation, the server MUST generate an error response with the 443 (Peer Address Family Mismatch) response code, which is defined in Section 6.2.1.

8. Packet Translations

The TURN specification [RFC5766] describes how TURN relays should relay traffic consisting of IPv4 packets (i.e., IPv4-to-IPv4 translations). The relay translates the IP addresses and port numbers of the packets based on the allocation's state data. How to translate other header fields is also specified in [RFC5766]. This document addresses IPv4-to-IPv6, IPv6-to-IPv4, and IPv6-to-IPv6 translations.

TURN relays performing any translation MUST translate the IP addresses and port numbers of the packets based on the allocation's state information as specified in [RFC5766]. The following sections specify how to translate other header fields.

As discussed in Section 2.6 of [RFC5766], translations in TURN are designed so that a TURN server can be implemented as an application that runs in "user-land" under commonly available operating systems and that does not require special privileges. The translations specified in the following sections follow this principle.

The descriptions below have two parts: a preferred behavior and an alternate behavior. The server SHOULD implement the preferred behavior. Otherwise, the server MUST implement the alternate behavior and MUST NOT do anything else.

8.1. IPv4-to-IPv6 Translations

Traffic Class

Preferred behavior: as specified in Section 4 of [RFC6145].

Alternate behavior: the relay sets the Traffic Class to the default value for outgoing packets.

Flow Label

Preferred behavior: the relay sets the Flow label to 0. The relay can choose to set the Flow label to a different value if it supports the IPv6 Flow Label field [RFC3697].

Alternate behavior: the relay sets the Flow label to the default value for outgoing packets.

Hop Limit

Preferred behavior: as specified in Section 4 of [RFC6145].

Alternate behavior: the relay sets the Hop Limit to the default value for outgoing packets.

Fragmentation

Preferred behavior: as specified in Section 4 of [RFC6145].

Alternate behavior: the relay assembles incoming fragments. The relay follows its default behavior to send outgoing packets.

For both preferred and alternate behavior, the DONT-FRAGMENT attribute ([RFC5766], Section 14.8) MUST be ignored by the server.

Extension Headers

Preferred behavior: the relay sends the outgoing packet without any IPv6 extension headers, with the exception of the Fragment Header as described above.

Alternate behavior: same as preferred.

8.2. IPv6-to-IPv6 Translations

Flow Label

The relay should consider that it is handling two different IPv6 flows. Therefore, the Flow label [RFC3697] SHOULD NOT be copied as part of the translation.

Preferred behavior: the relay sets the Flow label to 0. The relay can choose to set the Flow label to a different value if it supports the IPv6 Flow Label field [RFC3697].

Alternate behavior: the relay sets the Flow label to the default value for outgoing packets.

Hop Limit

Preferred behavior: the relay acts as a regular router with respect to decrementing the Hop Limit and generating an ICMPv6 error if it reaches zero.

Alternate behavior: the relay sets the Hop Limit to the default value for outgoing packets.

Fragmentation

Preferred behavior: if the incoming packet did not include a Fragment Header and the outgoing packet size does not exceed the outgoing link's MTU, the relay sends the outgoing packet without a Fragment Header.

If the incoming packet did not include a Fragment Header and the outgoing packet size exceeds the outgoing link's MTU, the relay drops the outgoing packet and sends an ICMP message of Type 2, Code 0 ("Packet too big") to the sender of the incoming packet.

If the packet is being sent to the peer, the relay reduces the MTU reported in the ICMP message by 48 bytes to allow room for the overhead of a Data indication.

If the incoming packet included a Fragment Header and the outgoing packet size (with a Fragment Header included) does not exceed the outgoing link's MTU, the relay sends the outgoing packet with a Fragment Header. The relay sets the fields of the Fragment Header as appropriate for a packet originating from the server.

If the incoming packet included a Fragment Header and the outgoing packet size exceeds the outgoing link's MTU, the relay MUST fragment the outgoing packet into fragments of no more than 1280 bytes. The relay sets the fields of the Fragment Header as appropriate for a packet originating from the server.

Alternate behavior: the relay assembles incoming fragments. The relay follows its default behavior to send outgoing packets.

For both preferred and alternate behavior, the DONT-FRAGMENT attribute MUST be ignored by the server.

Extension Headers

Preferred behavior: the relay sends the outgoing packet without any IPv6 extension headers, with the exception of the Fragment Header as described above.

Alternate behavior: same as preferred.

8.3. IPv6-to-IPv4 Translations

Type of Service and Precedence

Preferred behavior: as specified in Section 5 of [RFC6145].

Alternate behavior: the relay sets the Type of Service and Precedence to the default value for outgoing packets.

Time to Live

Preferred behavior: as specified in Section 5 of [RFC6145].

Alternate behavior: the relay sets the Time to Live to the default value for outgoing packets.

Fragmentation

Preferred behavior: as specified in Section 5 of [RFC6145]. Additionally, when the outgoing packet's size exceeds the outgoing link's MTU, the relay needs to generate an ICMP error (ICMPv6 Packet Too Big) reporting the MTU size. If the packet is being sent to the peer, the relay SHOULD reduce the MTU reported in the ICMP message by 48 bytes to allow room for the overhead of a Data indication.

Alternate behavior: the relay assembles incoming fragments. The relay follows its default behavior to send outgoing packets.

For both preferred and alternate behavior, the DONT-FRAGMENT attribute MUST be ignored by the server.

9. Security Considerations

Translation between IPv4 and IPv6 creates a new way for clients to obtain IPv4 or IPv6 access that they did not have before. For example, an IPv4-only client having access to a TURN server implementing this specification is now able to access the IPv6 Internet. This needs to be considered when establishing security and monitoring policies.

The loop attack described in [RFC5766], Section 17.1.7, may be more easily done in cases where address spoofing is easier to accomplish over IPv6. Mitigation of this attack over IPv6 is the same as for IPv4.

All the security considerations applicable to STUN [RFC5389] and TURN [RFC5766] are applicable to this document as well.

9.1. Tunnel Amplification Attack

An attacker might attempt to cause data packets to loop numerous times between a TURN server and a tunnel between IPv4 and IPv6. The attack goes as follows.

Suppose an attacker knows that a tunnel endpoint will forward encapsulated packets from a given IPv6 address (this doesn't necessarily need to be the tunnel endpoint's address). Suppose he then spoofs these two packets from this address:

1. An Allocate request asking for a v4 address, and

2. A ChannelBind request establishing a channel to the IPv4 address of the tunnel endpoint

Then he has set up an amplification attack:

- The TURN relay will re-encapsulate IPv6 UDP data in v4 and send it to the tunnel endpoint.

- The tunnel endpoint will decapsulate packets from the v4 interface and send them to v6.

So, if the attacker sends a packet of the following form:

```
IPv6: src=2001:db9::1 dst=2001:db8::2
UDP:  <ports>
TURN: <channel id>
IPv6: src=2001:db9::1 dst=2001:db8::2
UDP:  <ports>
TURN: <channel id>
IPv6: src=2001:db9::1 dst=2001:db8::2
UDP:  <ports>
TURN: <channel id>
...
```

Then the TURN relay and the tunnel endpoint will send it back and forth until the last TURN header is consumed, at which point the TURN relay will send an empty packet that the tunnel endpoint will drop.

The amplification potential here is limited by the MTU, so it's not huge: IPv6+UDP+TURN takes 334 bytes, so you could get a four-to-one amplification out of a 1500-byte packet. But the attacker could still increase traffic volume by sending multiple packets or by establishing multiple channels spoofed from different addresses behind the same tunnel endpoint.

The attack is mitigated as follows. It is RECOMMENDED that TURN relays not accept allocation or channel binding requests from addresses known to be tunneled, and that they not forward data to such addresses. In particular, a TURN relay MUST NOT accept Teredo or 6to4 addresses in these requests.

10. IANA Considerations

IANA registered the following values under the "STUN Attributes" registry and under the "STUN Error Codes" registry.

10.1. New STUN Attribute

```
0x0017: REQUESTED-ADDRESS-FAMILY
```

10.2. New STUN Error Codes

440 Address Family not Supported
443 Peer Address Family Mismatch

11. Acknowledgements

The authors would like to thank Alfred E. Heggestad, Dan Wing, Magnus Westerlund, Marc Petit-Huguenin, Philip Matthews, and Remi Denis-Courmont for their feedback on this document.

12. References

12.1. Normative References

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

[RFC3697] Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, "IPv6 Flow Label Specification", RFC 3697, March 2004.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

12.2. Informative References

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

Authors' Addresses

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Oscar Novo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: Oscar.Novo@ericsson.com

Simon Perreault (editor)
Viagenie
2600 boul. Laurier, suite D2-630
Quebec, QC G1V 2M2
Canada

Phone: +1 418 656 9254
EMail: simon.perreault@viagenie.ca
URI: http://www.viagenie.ca