IDR Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 22, 2020

C. Loibl, Ed.
next layer Telekom GmbH
R. Raszuk, Ed.
Bloomberg LP
S. Hares, Ed.
Huawei

April 20, 2020

Dissemination of Flow Specification Rules for IPv6
draft-ietf-idr-flow-spec-v6-11

Abstract

Dissemination of Flow Specification Rules [I-D.ietf-idr-rfc5575bis] provides a protocol extension for propagation of traffic flow information for the purpose of rate limiting or filtering. The [I-D.ietf-idr-rfc5575bis] specifies those extensions for IPv4 protocol data packets only.

This specification extends [I-D.ietf-idr-rfc5575bis] and defines changes to the original document in order to make it also usable and applicable to IPv6 data packets.

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 https://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 October 22, 2020.

Table of Contents

1. Introduction

The growing amount of IPv6 traffic in private and public networks requires the extension of tools used in the IPv4 only networks to be also capable of supporting IPv6 data packets.

In this document authors analyze the differences of IPv6 [RFC2460] flows description from those of traditional IPv4 packets and propose subset of new encoding formats to enable Dissemination of Flow Specification Rules [I-D.ietf-idr-rfc5575bis] for IPv6.

This specification should be treated as an extension of base [I-D.ietf-idr-rfc5575bis] specification and not its replacement. It only defines the delta changes required to support IPv6 while all other definitions and operation mechanisms of Dissemination of Flow Specification Rules will remain in the main specification and will not be repeated here.

1.1. Definitions of Terms Used in This Memo

AFI - Address Family Identifier.

AS - Autonomous System.

NLRI - Network Layer Reachability Information.

SAFI - Subsequent Address Family Identifier.

VRF - Virtual Routing and Forwarding instance.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. IPv6 Flow Specification encoding in BGP

The [I-D.ietf-idr-rfc5575bis] defines new SAFIs 133 (Dissemination of Flow Specification) and 134 (L3VPN Dissemination of Flow Specification) applications in order to carry corresponding to each such application Flow Specification.

Implementations wishing to exchange IPv6 Flow Specifications MUST use BGP's Capability Advertisement facility to exchange the Multiprotocol Extension Capability Code (Code 1) as defined in [RFC4760]. While [I-D.ietf-idr-rfc5575bis] specifies Flow Specification for IPv4 (AFI=1) only, the (AFI, SAFI) pair carried in the Multiprotocol Extension Capability MUST be: (AFI=2, SAFI=133) for IPv6 Flow Specification, and (AFI=2, SAFI=134) for VPNv6 Flow Specification.

For both SAFIs the indication to which address family they are referring to will be recognized by AFI value (AFI=1 for IPv4 or VPNv4, AFI=2 for IPv6 and VPNv6 respectively).

It needs to be observed that such choice of proposed encoding is compatible with filter validation against routing reachability information as described in section 6 of [I-D.ietf-idr-rfc5575bis].

3. IPv6 Flow Specification components

The following components are redefined or added for the purpose of accommodating the IPv6 header encoding. Unless otherwise specified all other components defined in [I-D.ietf-idr-rfc5575bis] Section 4.2.2 also apply to IPv6 Flow Specification.

3.1. Type 1 - Destination IPv6 Prefix

Encoding: <type (1 octet), length (1 octet), offset (1 octet), prefix (variable)>

Defines the destination prefix to match. The offset has been defined to allow for flexible matching on part of the IPv6 address where it is required to skip (don't care) of N first bits of the address. This can be especially useful where part of the IPv6 address consists of an embedded IPv4 address and matching needs to happen only on the embedded IPv4 address. The encoded prefix contains enough octets for the bits used in matching (length minus offset bits).

3.2. Type 2 - Source IPv6 Prefix

Encoding: <type (1 octet), length (1 octet), offset (1 octet), prefix (variable)>

Defines the source prefix to match. The length, offset and prefix are the same as in Section 3.1

3.3. Type 3 - Next Header

Encoding: <type (1 octet), [numeric_op, value]+>

Contains a list of {numeric_op, value} pairs that are used to match the last Next Header ([RFC2460] Section 3) value octet in IPv6 packets.

This component uses the Numeric Operator (numeric_op) described in [I-D.ietf-idr-rfc5575bis] Section 4.2.1.1. Type 3 component values SHOULD be encoded as single byte (numeric_op len=00).

Note: While IPv6 allows for more then one Next Header field in the packet the main goal of Type 3 flow specification component is to match on the subsequent IP protocol value. Therefor the definition is limited to match only on last Next Header field in the packet.

3.4. Type 7 - ICMPv6 type

Encoding: <type (1 octet), [numeric_op, value]+>

Defines a list of {numeric_op, value} pairs used to match the type field of an ICMPv6 packet (see also [RFC4443] Section 2.1).

This component uses the Numeric Operator (numeric_op) described in [I-D.ietf-idr-rfc5575bis] Section 4.2.1.1. Type 7 component values SHOULD be encoded as single byte (numeric_op len=00).

In case of the presence of the ICMPv6 type (code) component only ICMPv6 packets can match the entire Flow Specification. The ICMPv6 type (code) component, if present, never matches when the packet's last Next Header field value is not 58 (ICMPv6), if the packet is fragmented and this is not the first fragment, or if the system is unable to locate the transport header. Different implementations may or may not be able to decode the transport header.

3.5. Type 8 - ICMPv6 code

Encoding: <type (1 octet), [numeric_op, value]+>

Defines a list of {numeric_op, value} pairs used to match the code field of an ICMPv6 packet (see also [RFC4443] Section 2.1).

This component uses the Numeric Operator (numeric_op) described in [I-D.ietf-idr-rfc5575bis] Section 4.2.1.1. Type 8 component values SHOULD be encoded as single byte (numeric_op len=00).

The last paragraph of Section 3.4 also applies to this component.

3.6. Type 12 - Fragment

Encoding: <type (1 octet), [bitmask_op, bitmask]+>

Defines a list of {bitmask_op, bitmask} pairs used to match specific IP fragments.

This component uses the Bitmask Operator (bitmask_op) described in [I-D.ietf-idr-rfc5575bis] Section 4.2.1.2. The Type 12 component bitmask MUST be encoded as single byte bitmask (bitmask_op len=00).

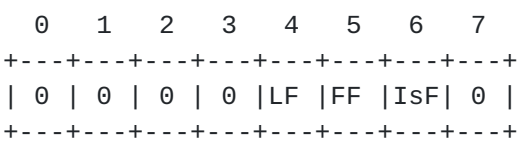

Figure 1: Fragment Bitmask Operand

Bitmask values:

IsF - Is a fragment - match if IPv6 Fragment Header ([RFC2460] Section 4.5) Fragment Offset is not 0

FF - First fragment - match if IPv6 Fragment Header ([RFC2460] Section 4.5) Fragment Offset is 0 AND M flag is 1

LF - Last fragment - match if IPv6 Fragment Header ([RFC2460] Section 4.5) Fragment Offset is not 0 AND M flag is 0

0 - SHOULD be set to 0 on NLRI encoding, and MUST be ignored during decoding

3.7. Type 13 - Flow Label (new)

Encoding: <type (1 octet), [numeric_op, value]+>

Contains a list of {numeric_op, value} pairs that are used to match the 20-bit Flow Label IPv6 header field ([RFC2460] Section 3).

This component uses the Numeric Operator (numeric_op) described in [I-D.ietf-idr-rfc5575bis] Section 4.2.1.1. Type 13 component values SHOULD be encoded as 1-, 2-, or 4-byte quantities (numeric_op len=00, len=01 or len=10).

3.8. Encoding Example

3.8.1. Example 1

The following example demonstrates the prefix encoding for: "all packets to ::1234:5678:9A00:0/64-104 from 100::/8 and port 25".

length	destination	source	port
0x0f	01 68 40 12 34 56 78 9A	02 08 00 01	04 81 19

Decoded:

Value		
0x0f	length	16 octets (len<240 1-octet)
0x01	type	Type 1 - Dest. IPv6 Prefix
0x68	length	104 bit
0x40	offset	64 bit
0x12	prefix	
0x34	prefix	
0x56	prefix	
0x78	prefix	
0x9A	prefix	
0x02	type	Type 2 - Source IPv6 Prefix
0x08	length	8 bit
0x00	offset	0 bit
0x01	prefix	
0x04	type	Type 4 - Port
0x81	numeric_op	end-of-list, value size=1, =
0x19	value	25

This constitutes a NLRI with a NLRI length of 16 octets.

3.8.2. Example 2

The following example demonstrates the prefix encoding for: "all packets to ::1234:5678:9A00:0/65-104".

length	destination
0x08	01 68 41 24 68 ac f1 34

Decoded:

Value		
0x08	length	8 octets (len<240 1-octet)
0x01	type	Type 1 - Dest. IPv6 Prefix
0x68	length	104 bit
0x41	offset	65 bit
0x24	prefix	starting with the 66ths bit
0x68	prefix	
0xac	prefix	
0xf1	prefix	
0x34	prefix	

This constitutes a NLRI with a NLRI length of 8 octets.

4. Ordering of Flow Specifications

The definition for the order of traffic filtering rules from [I-D.ietf-idr-rfc5575bis] Section 5.1 can be reused with new consideration for the IPv6 prefix offset. As long as the offsets are equal, the comparison is the same, retaining longest-prefix-match semantics. If the offsets are not equal, the lowest offset has precedence, as this flow matches the most significant bit.

The code in Appendix A shows a Python3 implementation of the resulting comparison algorithm. The full code was tested with Python 3.7.2 and can be obtained at https://github.com/stoffi92/flowspec-cmp-v6 [1].

5. Validation Procedure

The validation procedure is the same as specified in [I-D.ietf-idr-rfc5575bis] Section 6 with the exception that item a) of the validation procedure should now read as follows:

> a) A destination prefix component with offset=0 is embedded in the Flow Specification

6. IPv6 Traffic Filtering Action changes

Traffic Filtering Actions from [I-D.ietf-idr-rfc5575bis] Section 7 can also be applied to IPv6 Flow Specifications. To allow an IPv6 address specific route-target, a new Traffic Filtering Action IPv6 address specific extended community is specified in Section 6.1 below:

6.1. Redirect IPv6 (rt-redirect-ipv6) Type/Sub-Type 0x80/TBD

The redirect IPv6 address specific extended community allows the traffic to be redirected to a VRF routing instance that lists the specified IPv6 address specific route-target in its import policy. If several local instances match this criteria, the choice between them is a local matter (for example, the instance with the lowest Route Distinguisher value can be elected).

This extended community uses the same encoding as the IPv6 address specific Route Target extended community [RFC5701] Section 2 with the high-order octet of the Type always set to 0x80 and the Sub-Type always TBD.

Interferes with: All BGP Flow Specification redirect Traffic Filtering Actions (with itself and those specified in [I-D.ietf-idr-rfc5575bis] Section 7.4).

7. Security Considerations

No new security issues are introduced to the BGP protocol by this specification over the security considerations in [I-D.ietf-idr-rfc5575bis]

8. IANA Considerations

This section complies with [RFC7153]

IANA is requested to create and maintain a new registry entitled: "Flow Spec IPv6 Component Types" containing the initial entries as specified in Table 1.

Value	Name	Reference
1	Destination IPv6 Prefix	[this document]
2	Source IPv6 Prefix	[this document]
3	Next Header	[this document]
4	Port	[this document]
5	Destination port	[this document]
6	Source port	[this document]
7	ICMPv6 type	[this document]
8	ICMPv6 code	[this document]
9	TCP flags	[this document]
10	Packet length	[this document]
11	DSCP	[this document]
12	Fragment	[this document]
13	Flow Label	[this document]

Table 1: Registry: Flow Spec IPv6 Component Types

IANA maintains a registry entitled "Generic Transitive Experimental Use Extended Community Sub-Types". For the purpose of this work, IANA is requested to assign a new value:

Sub-Type Value	Name	Reference
TBD	Flow spec rt-redirect-ipv6 format	[this document]

Table 2: Registry: Generic Transitive Experimental Use Extended Community Sub-Types

9. Acknowledgements

Authors would like to thank Pedro Marques, Hannes Gredler and Bruno Rijsman, Brian Carpenter, and Thomas Mangin for their valuable input.

10. Contributors

Danny McPherson
Verisign, Inc.

Email: dmcpherson@verisign.com

Burjiz Pithawala
Individual

Email: burjizp@gmail.com

Andy Karch
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: akarch@cisco.com

11. References

11.1. Normative References

[I-D.ietf-idr-rfc5575bis]
Loibl, C., Hares, S., Raszuk, R., McPherson, D., and M. Bacher, "Dissemination of Flow Specification Rules", draft-ietf-idr-rfc5575bis-22 (work in progress), April 2020.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <https://www.rfc-editor.org/info/rfc4760>.

[RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, DOI 10.17487/RFC5492, February 2009, <https://www.rfc-editor.org/info/rfc5492>.

[RFC5701] Rekhter, Y., "IPv6 Address Specific BGP Extended Community Attribute", RFC 5701, DOI 10.17487/RFC5701, November 2009, <https://www.rfc-editor.org/info/rfc5701>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <https://www.rfc-editor.org/info/rfc6437>.

[RFC7153] Rosen, E. and Y. Rekhter, "IANA Registries for BGP Extended Communities", RFC 7153, DOI 10.17487/RFC7153, March 2014, <https://www.rfc-editor.org/info/rfc7153>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative References

[RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <https://www.rfc-editor.org/info/rfc5095>.

11.3. URIs

[1] https://github.com/stoffi92/flowspec-cmp-v6

Appendix A. Python code: flow_rule_cmp_v6

```
<CODE BEGINS>
"""
Copyright (c) 2019 IETF Trust and the persons identified as authors of
```

```
import itertools
import ipaddress

def flow_rule_cmp_v6(a, b):
    for comp_a, comp_b in itertools.zip_longest(a.components,
                                                b.components):
        # If a component type does not exist in one rule
        # this rule has lower precedence
        if not comp_a:
            return B_HAS_PRECEDENCE
        if not comp_b:
            return A_HAS_PRECEDENCE
        # Higher precedence for lower component type
        if comp_a.component_type < comp_b.component_type:
            return A_HAS_PRECEDENCE
        if comp_a.component_type > comp_b.component_type:
            return B_HAS_PRECEDENCE
        # component types are equal -> type specific comparison
        if comp_a.component_type in (IP_DESTINATION, IP_SOURCE):
            if comp_a.offset < comp_b.offset:
                return A_HAS_PRECEDENCE
            if comp_a.offset < comp_b.offset:
                return B_HAS_PRECEDENCE
            # both components have the same offset
            # assuming comp_a.value, comp_b.value of type
            # ipaddress.IPv6Network
            # and the offset bits are reset to 0 (since they are not
            # represented in the NLRI)
            if comp_a.value.overlaps(comp_b.value):
                # longest prefixlen has precedence
                if comp_a.value.prefixlen > comp_b.value.prefixlen:
                    return A_HAS_PRECEDENCE
                if comp_a.value.prefixlen < comp_b.value.prefixlen:
                    return B_HAS_PRECEDENCE
                # components equal -> continue with next component
            elif comp_a.value > comp_b.value:
                return B_HAS_PRECEDENCE
            elif comp_a.value < comp_b.value:
                return A_HAS_PRECEDENCE
```

```
          else:
              # assuming comp_a.value, comp_b.value of type bytearray
              if len(comp_a.value) == len(comp_b.value):
                  if comp_a.value > comp_b.value:
                      return B_HAS_PRECEDENCE
                  if comp_a.value < comp_b.value:
                      return A_HAS_PRECEDENCE
                  # components equal -> continue with next component
              else:
                  common = min(len(comp_a.value), len(comp_b.value))
                  if comp_a.value[:common] > comp_b.value[:common]:
                      return B_HAS_PRECEDENCE
                  elif comp_a.value[:common] < comp_b.value[:common]:
                      return A_HAS_PRECEDENCE
                  # the first common bytes match
                  elif len(comp_a.value) > len(comp_b.value):
                      return A_HAS_PRECEDENCE
                  else:
                      return B_HAS_PRECEDENCE
      return EQUAL
  <CODE ENDS>
```

Authors' Addresses

Christoph Loibl (editor)
next layer Telekom GmbH
Mariahilfer Guertel 37/7
Vienna 1150
AT

Phone: +43 664 1176414
Email: cl@tix.at

Robert Raszuk (editor)
Bloomberg LP
731 Lexington Ave
New York City, NY 10022
USA

Email: robert@raszuk.net

Susan Hares (editor)
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com