Network Working Group C. Filsfils, Ed.
Internet-Draft Cisco Systems, Inc.
Intended status: Standards Track S. Previdi
Expires: August 8, 2019 Huawei
J. Leddy
Individual
S. Matsushima
Softbank
D. Voyer, Ed.
Bell Canada
February 4, 2019

# IPv6 Segment Routing Header (SRH)
## draft-ietf-6man-segment-routing-header-16

Abstract

Segment Routing can be applied to the IPv6 data plane using a new type of Routing Extension Header. This document describes the Segment Routing Extension Header and how it is used by Segment Routing capable nodes.

Requirements Language

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.

Status of This Memo

Copyright Notice



Table of Contents

## 1. Introduction

Segment Routing can be applied to the IPv6 data plane using a new type of Routing Extension Header (SRH). This document describes the Segment Routing Extension Header and how it is used by Segment Routing capable nodes.

The Segment Routing Architecture [RFC8402] describes Segment Routing and its instantiation in two data planes MPLS and IPv6.

SR with the MPLS data plane is defined in [I-D.ietf-spring-segment-routing-mpls].

SR with the IPv6 data plane is defined in [I-D.filsfils-spring-srv6-network-programming].

The encoding of MPLS labels and label stacking are defined in [RFC3032].

The encoding of IPv6 segments in the Segment Routing Extension Header is defined in this document.

Terminology used within this document is defined in detail in [RFC8402]. Specifically, these terms: Segment Routing, SR Domain, SRv6, Segment ID (SID), SRv6 SID, Active Segment, and SR Policy.

## 2. Segment Routing Extension Header

Routing Headers are defined in [RFC8200]. The Segment Routing Header has a new Routing Type (suggested value 4) to be assigned by IANA.

The Segment Routing Header (SRH) is defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Hdr Ext Len  | Routing Type  | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Last Entry   |     Flags     |              Tag              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[0] (128 bits IPv6 address)            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
                              ...
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[n] (128 bits IPv6 address)            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                                                             //
//         Optional Type Length Value objects (variable)       //
//                                                             //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

o  Next Header: Defined in [RFC8200]

o  Hdr Ext Len: Defined in [RFC8200]

o Routing Type: TBD, to be assigned by IANA (suggested value: 4).

o Segments Left: Defined in [RFC8200]

o Last Entry: contains the index (zero based), in the Segment List, of the last element of the Segment List.

o Flags: 8 bits of flags. Following flags are defined:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|U U U U U U U U|
+-+-+-+-+-+-+-+-+
```

U: Unused and for future use. MUST be 0 on transmission and ignored on receipt.

o Tag: tag a packet as part of a class or group of packets, e.g., packets sharing the same set of properties. When tag is not used at source it MUST be set to zero on transmission. When tag is not used during SRH Processing it SHOULD be ignored. The allocation and use of tag is outside the scope of this document.

o Segment List[n]: 128 bit IPv6 addresses representing the nth segment in the Segment List. The Segment List is encoded starting from the last segment of the SR Policy. I.e., the first element of the segment list (Segment List [0]) contains the last segment of the SR Policy, the second element contains the penultimate segment of the SR Policy and so on.

o Type Length Value (TLV) are described in Section 2.1.

## 2.1. SRH TLVs

This section defines TLVs of the Segment Routing Header.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-----------------------
|     Type      |    Length     | Variable length data
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-----------------------
```

Type: An 8 bit value. Unrecognized Types MUST be ignored on receipt.

Length: The length of the Variable length data. It is RECOMMENDED that the total length of new TLVs be multiple of 8 bytes to avoid the use of Padding TLVs.

Variable length data: Length bytes of data that is specific to the Type.

Type Length Value (TLV) contain OPTIONAL information that may be used by the node identified in the Destination Address (DA) of the packet.

Each TLV has its own length, format and semantic. The code-point allocated (by IANA) to each TLV Type defines both the format and the semantic of the information carried in the TLV. Multiple TLVs may be encoded in the same SRH.

TLVs may change en route at each segment. To identify when a TLV type may change en route the most significant bit of the Type has the following significance:

0: TLV data does not change en route

1: TLV data does change en route

Identifying which TLVs change en route, without having to understand the Type, is required for Authentication Header Integrity Check Value (ICV) computation. Any TLV that changes en route is considered mutable for the purpose of ICV computation, the Type Length and Variable Length Data is ignored for the purpose of ICV Computation as defined in [RFC4302].

The "Length" field of the TLV is used to skip the TLV while inspecting the SRH in case the node doesn't support or recognize the Type. The "Length" defines the TLV length in octets, not including the "Type" and "Length" fields.

The following TLVs are defined in this document:

Padding TLV

HMAC TLV

Additional TLVs may be defined in the future.

### 2.1.1. Padding TLVs

There are two types of padding TLVs, pad0 and padN, the following applies to both:

Padding TLVs are used to pad the TLVs to a multiple of 8 octets.

More than one Padding TLV MUST NOT appear in the SRH.

The Padding TLVs are used to align the SRH total length on the 8 octet boundary.

When present, a single Pad0 or PadN TLV MUST appear as the last TLV.

When present, a PadN TLV MUST have a length from 0 to 5 in order to align the SRH total length on a 8-octet boundary.

Padding TLVs are ignored by a node processing the SRH TLV, even if more than one is present.

Padding TLVs are ignored during ICV calculation.

#### 2.1.1.1. PAD0

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|     Type      |
+-+-+-+-+-+-+-+-+
```

Type: to be assigned by IANA (Suggested value 128)

A single Pad0 TLV MUST be used when a single byte of padding is required. If more than one byte of padding is required a Pad0 TLV MUST NOT be used, the PadN TLV MUST be used.

#### 2.1.1.2. PADN

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |      Padding (variable)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                     Padding (variable)                       //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: to be assigned by IANA (suggested value 129).

Length: 0 to 5

Padding: Length octets of padding. Padding bits have no semantics. They MUST be set to 0 on transmission and ignored on receipt.

The PadN TLV MUST be used when more than one byte of padding is required.

### 2.1.2. HMAC TLV

The keyed Hashed Message Authentication Code (HMAC) TLV is OPTIONAL and has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |    Length     |          RESERVED             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      HMAC Key ID (4 octets)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                              //
|                      HMAC (32 octets)                        //
|                                                              //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

o  Type: to be assigned by IANA (suggested value 5).

o  Length: 38.

o  RESERVED: 2 octets.  MUST be 0 on transmission and ignored on receipt.

o  HMAC Key ID: A 4 octet opaque number which uniquely identifies the pre-shared key and algorithm used to generate the HMAC.  If 0, the HMAC is not included.

o  HMAC: 32 octets of keyed HMAC, not present if Key ID is 0.

The HMAC TLV is used to verify the source of a packet is permitted to use the current segment in the destination address of the packet, and ensure the segment list is not modified in transit.

#### 2.1.2.1. HMAC Generation and Verification

Local policy determines when to check for an HMAC and potentially provides an alternate composition of Text, and a requirement on where the HMAC TLV must appear (e.g. first TLV).  This local policy is outside the scope of this document.  It may be based on the active segment at an SR Segment endpoint node, the result of an ACL that considers incoming interface, HMAC Key ID, or other packet fields.

The HMAC field is the output of the HMAC computation as defined in [RFC2104], using:

- key: the pre-shared key identified by HMAC Key ID

- HMAC algorithm: identified by the HMAC Key ID

- Text: a concatenation of the following fields from the IPv6 header and the SRH, as it would be received at the node verifying the HMAC:

  * IPv6 header: source address (16 octets)

  * SRH: Last Entry (1 octet)

  * SRH: Flags (1 octet)

  * SRH: HMAC Key-id (4 octets)

  * SRH: all addresses in the Segment List (variable octets)

The HMAC digest is truncated to 32 octets and placed in the HMAC field of the HMAC TLV.

For HMAC algorithms producing digests less than 32 octets, the digest is placed in the lowest order octets of the HMAC field. Remaining octets MUST be set to zero.

If HMAC verification is successful, the packet is forwarded to the next segment.

If HMAC verification fails, an ICMP error message (parameter problem, error code 0, pointing to the HMAC TLV) SHOULD be generated (but rate limited) and SHOULD be logged.

#### 2.1.2.2. **HMAC Pre-Shared Key Algorithm**

The HMAC Key ID field allows for the simultaneous existence of several hash algorithms (SHA-256, SHA3-256 ... or future ones) as well as pre-shared keys.

The HMAC Key ID field is opaque, i.e., it has neither syntax nor semantic except as an identifier of the right combination of pre-shared key and hash algorithm, and except that a value of 0 means that there is no HMAC field.

At the HMAC TLV verification node the Key ID uniquely identifies the pre-shared key and HMAC algorithm.

At the HMAC TLV generating node the Key ID and destination address uniquely identify the pre-shared key and HMAC algorithm. Utilizing

the destination address with the Key ID allows for overlapping key IDs amongst different HMAC verification nodes. The Text for the HMAC computation is set to the IPv6 header fields and SRH fields as they would appear at the verification node, not necessarily the same as the source node sending a packet with the HMAC TLV.

Pre-shared key roll-over is supported by having two key IDs in use while the HMAC TLV generating node and verifying node converge to a new key.

SRH implementations can support multiple hash functions but MUST implement SHA-2 [FIPS180-4] in its SHA-256 variant.

The selection of pre-shared key and algorithm, and their distribution is outside the scope of this document, some options may include:

- o in the configuration of the HMAC generating or verifying nodes, either by static configuration or any SDN oriented approach

- o dynamically using a trusted key distribution protocol such as [RFC6407]

## 3. SR Nodes

There are different types of nodes that may be involved in segment routing networks: source SR nodes originate packets with a segment in the destination address of the IPv6 header, transit nodes that forward packets destined to a remote segment, and SR segment endpoint nodes that process a local segment in the destination address of an IPv6 header.

### 3.1. Source SR Node

A Source SR Node is any node that originates an IPv6 packet with a segment (i.e. SRv6 SID) in the destination address of the IPv6 header. The packet leaving the source SR Node may or may not contain an SRH. This includes either:

A host originating an IPv6 packet.

An SR domain ingress router encapsulating a received packet in an outer IPv6 header, followed by an optional SRH.

The mechanism through which a segment in the destination address of the IPv6 header and the Segment List in the SRH, is derived is outside the scope of this document.

### 3.2. Transit Node

A transit node is any node forwarding an IPv6 packet where the destination address of that packet is not locally configured as a segment nor a local interface. A transit node is not required to be capable of processing a segment nor SRH.

### 3.3. SR Segment Endpoint Node

A SR segment endpoint node is any node receiving an IPv6 packet where the destination address of that packet is locally configured as a segment or local interface.

## 4. Packet Processing

This section describes SRv6 packet processing at the SR source, Transit and SR segment endpoint nodes.

### 4.1. Source SR Node

A Source node steers a packet into an SR Policy. If the SR Policy results in a segment list containing a single segment, and there is no need to add information to SRH flag or TLV, the DA is set to the single segment list entry and the SRH MAY be omitted.

When needed, the SRH is created as follows:

Next Header and Hdr Ext Len fields are set as specified in [RFC8200].

Routing Type field is set as TBD (to be allocated by IANA, suggested value 4).

The DA of the packet is set with the value of the first segment.

The first element of the SRH Segment List is the ultimate segment. The second element is the penultimate segment and so on.

The Segments Left field is set to n-1 where n is the number of elements in the SR Policy.

The Last Entry field is set to n-1 where n is the number of elements in the SR Policy.

HMAC TLV may be set according to Section 7.

The packet is forwarded toward the packet's Destination Address (the first segment).

#### 4.1.1. Reduced SRH

When a source does not require the entire SID list to be preserved in the SRH, a reduced SRH may be used.

A reduced SRH does not contain the first segment of the related SR Policy (the first segment is the one already in the DA of the IPv6 header), and the Last Entry field is set to n-2 where n is the number of elements in the SR Policy.

### 4.2. Transit Node

As specified in [RFC8200], the only node allowed to inspect the Routing Extension Header (and therefore the SRH), is the node corresponding to the DA of the packet. Any other transit node MUST NOT inspect the underneath routing header and MUST forward the packet toward the DA according to its IPv6 routing table.

When a SID is in the destination address of an IPv6 header of a packet, it's routed through an IPv6 network as an IPv6 address. SIDs, or the prefix(es) covering SIDs, and their reachability may be distributed by means outside the scope of this document. For example, [RFC5308] or [RFC5340] may be used to advertise a prefix covering the SIDs on a node.

### 4.3. SR Segment Endpoint Node

Without constraining the details of an implementation, the SR segment endpoint node creates Forwarding Information Base (FIB) entries for its local SIDs.

When an SRv6-capable node receives an IPv6 packet, it performs a longest-prefix-match lookup on the packets destination address. This lookup can return any of the following:

```
    A FIB entry that represents a locally instantiated SRv6 SID
    A FIB entry that represents a local interface, not locally
                                  instantiated as an SRv6 SID
    A FIB entry that represents a non-local route
    No Match
```

#### 4.3.1. FIB Entry Is Locally Instantiated SRv6 END SID

This document, and section, defines a single SRv6 SID called END. Future documents may define additional SRv6 SIDs. In which case, the entire content of this section will be defined in that document.

If the FIB entry represents a locally instantiated SRv6 SID, process the next header of the IPv6 header as defined in section 4 of [RFC8200]

The following sections describe the actions to take while processing next header fields.

#### 4.3.1.1. SRH Processing

```
S01. When an SRH is processed {
S02.   If Segments Left is equal to zero {
S03.     Proceed to process the next header in the packet,
         whose type is identified by the Next Header field in
         the Routing header.
S04.   }
S05.   Else {
S06.     If local policy requires TLV processing {
S07.       Perform TLV processing (see TLV Processing)
S08.     }
S09.     max_last_entry  =  ( Hdr Ext Len /  2 ) - 1
S10.     If  ((Last Entry > max_last_entry) or
S11.          (Segments Left is greater than (Last Entry+1)) {
S12.       Send an ICMP Parameter Problem, Code 0, message to
           the Source Address, pointing to the Segments Left
           field, and discard the packet.
S13.     }
S14.     Else {
S15.       Decrement Segments Left by 1.
S16.       Copy Segment List[Segments Left] from the SRH to the
           destination address of the IPv6 header.
S17.       If the IPv6 Hop Limit is less than or equal to 1 {
S18.         Send an ICMP Time Exceeded -- Hop Limit Exceeded in
             Transit message to the Source Address and discard
             the packet.
S19.       }
S20.       Else {
S21.         Decrement the Hop Limit by 1
S22.         Resubmit the packet to the IPv6 module for transmission
             to the new destination.
S23.       }
S24.     }
S25.   }
S26. }
```

#### 4.3.1.1.1. TLV Processing

Local policy determines how TLV's are to be processed when the Active Segment is a local END SID. The definition of local policy is outside the scope of this document.

For illustration purpose only, two example local policies that may be associated with an END SID are provided below.

```
Example 1:
For any packet received from interface I2
  Skip TLV processing

Example 2:
For any packet received from interface I1
  If first TLV is HMAC {
    Process the HMAC TLV
  }
  Else {
    Discard the packet
  }
```

#### 4.3.1.2. Upper-layer Header or No Next Header

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 END SID.

```
IF (Upper-layer Header is IPv4 or IPv6) and local policy permits {
  Perform IPv6 decapsulation
  Resubmit the decapsulated packet to the IPv4 or IPv6 module
}
ELSE {
  Send an ICMP parameter problem message to the Source Address and
  discard the packet.  Error code (TBD by IANA) "SR Upper-layer
  Header Error", pointer set to the offset of the upper-layer
  header.
}
```

A unique error code allows an SR Source node to recognize an error in SID processing at an endpoint.

### 4.3.2. FIB Entry is a Local Interface

If the FIB entry represents a local interface, not locally instantiated as an SRv6 SID, the SRH is processed as follows:

If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing Header.

If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

#### 4.3.3. FIB Entry Is A Non-Local Route

Processing is not changed by this document.

#### 4.3.4. FIB Entry Is A No Match

Processing is not changed by this document.

## 5. Intra SR Domain Deployment Model

The use of the SIDs exclusively within the SR Domain and solely for packets of the SR Domain is an important deployment model.

This enables the SR Domain to act as a single routing system.

This section covers:

o  securing the SR Domain from external attempt to use its SIDs

o  SR Domain as a single system with delegation between components

o  handling packets of the SR Domain

### 5.1. Securing the SR Domain

Nodes outside the SR Domain are not trusted: they cannot directly use the SID's of the domain.  This is enforced by two levels of access control lists:

1.  Any packet entering the SR Domain and destined to a SID within the SR Domain is dropped.  This may be realized with the following logic, other methods with equivalent outcome are considered compliant:

    *  allocate all the SID's from a block S/s

    *  configure each external interface of each edge node of the domain with an inbound infrastructure access list (IACL) which drops any incoming packet with a destination address in S/s

   * Failure to implement this method of ingress filtering exposes the SR Domain to source routing attacks as described and referenced in [RFC5095]

2. The distributed protection in #1 is complemented with per node protection, dropping packets to SIDs from source addresses outside the SR Domain. This may be realized with the following logic, other methods with equivalent outcome are considered compliant:

   * assign all interface addresses from prefix A/a

   * at node k, all SIDs local to k are assigned from prefix Sk/sk

   * configure each internal interface of each SR node k in the SR Domain with an inbound IACL which drops any incoming packet with a destination address in Sk/sk if the source address is not in A/a.

## 5.2. SR Domain as a single system with delegation among components

All intra SR Domain packets are of the SR Domain. The IP v6 header is originated by a node of the SR Domain, and is destined to a node of the SR Domain.

All inter domain packets are encapsulated for the part of the packet journey that is within the SR Domain. The outer IPv6 header is originated by a node of the SR Domain, and is destined to a node of the SR Domain.

As a consequence, any packet within the SR Domain is of the SR Domain.

The SR Domain is a system in which the operator may want to distribute or delegate different operations of the outer most header to different nodes within the system.

An operator of an SR domain may choose to delegate SRH addition to a host node within the SR domain, and validation of the contents of any SRH to a more trusted router or switch attached to the host. Consider a top of rack switch (T) connected to host (H) via interface (I). H receives an SRH (SRH1) with a computed HMAC via some SDN method outside the scope of this document. H classifies traffic it sources and adds SRH1 to traffic requiring a specific SLA. T is configured with an IACL on I requiring verification of the SRH for any packet destined to the SID block of the SR Domain (S/s). T checks and verifies that SRH1 is valid, contains an HMAC TLV and verifies the HMAC.

## 5.3. MTU Considerations

Within the SR Domain, well known mitigation techniques are RECOMMENDED, such as deploying a greater MTU value within the SR Domain than at the ingress edges.

## 5.4. AH ICV

Within the domain, an SR source node which includes SRH and AH extension headers can predict the content of the SRH and calculate the ICV at the SR source node, ensuring it can be confirmed at the destination.

## 5.5. ESP ICV

Within the domain, an SR source node may include SRH and ESP extension headers. Only the data following the ESP header is included in ICV computation. SRH precedes ESP.

## 5.6. ICMP Error Processing

ICMP error packets generated within the SR Domain are sent to source nodes within the SR Domain. The invoking packet in the ICMP error message may contain an SRH. Since the destination address of a packet with an SRH changes as each segment is processed, it may not be the destination used by the socket or application that generated the invoking packet.

For the source of an invoking packet to process the ICMP error message, the correct destination address must be determined. The following logic is used to determine the destination address for use by protocol error handlers.

- o Walk all extension headers of the invoking IPv6 packet to the routing extension header preceding the upper layer header.
  - * If routing header is type 4 (SRH)
    - + Use the 0th segment in the segment list as the destination address of the invoking packet.

ICMP errors are then processed by upper layer transports as defined in [RFC4443].

For IP packets encapsulated in an outer IPv6 header, ICMP error handling is as defined in [RFC2473].

### 5.7. Load Balancing and ECMP

For any inter domain packet, the SR Source node MUST impose a flow label computed based on the inner packet. The computation of the flow label is as recommended in [RFC6438] for the sending Tunnel End Point.

At any transit node within an SR domain, the flow label MUST be used as defined in [RFC6438] to calculate the ECMP hash toward the destination address. If flow label is not used, the transit node may hash all packets between a pair of SR Edge nodes to the same link.

At an SR segment endpoint node, the flow label MUST be used as defined in [RFC6438] to calculate any ECMP hash used to forward the processed packet to the next segment.

### 5.8. Other Deployments

Other deployment models and their implications on security, MTU, HMAC, ICMP error processing and interaction with other extension headers are outside the scope of this document.

## 6. Illustrations

This section provides illustrations of SRv6 packet processing at SR source, transit and SR segment endpoint nodes.

### 6.1. Abstract Representation of an SRH

For a node k, its IPv6 address is represented as Ak, its SRv6 SID is represented as Sk.

IPv6 headers are represented as the tuple of (source, destination). For example, a packet with source address A1 and destination address A2 is represented as (A1,A2). The payload of the packet is omitted.

An SR Policy is a list of segments. A list of segments is represented as <S1,S2,S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit.

(SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:

- o Source Address is SA, Destination Addresses is DA, and next-header is SRH.
- o SRH with SID list <S1, S2, S3> with SegmentsLeft = SL.

- o Note the difference between the <> and () symbols. <S1, S2, S3> represents a SID list where the leftmost segment is the first segment. Whereas, (S3, S2, S1; SL) represents the same SID list but encoded in the SRH Segment List format where the leftmost segment is the last segment. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of detailed behavior, the (S3, S2, S1; SL) notation is more convenient.

At its SR Policy headend, the Segment List <S1,S2,S3> results in SRH (S3,S2,S1; SL=2) represented fully as:

```
Segments Left=2
Last Entry=2
Flags=0
Tag=0
Segment List[0]=S3
Segment List[1]=S2
Segment List[2]=S1
```

## 6.2. Example Topology

The following topology is used in examples below:

```
        + * * * * * * * * * * * * * * * * * * * * +

        *         [8]                [9]          *
                   |                  |
        *          |                  |           *
[1]----[3]--------[5]----------------[6]---------[4]---[2]
        *          |                  |           *
                   |                  |
        *          |                  |           *
                   +--------[7]-------+
        *                                         *

        + * * * * * * * *  SR Domain  * * * * * * * +
```

Figure 3

- o 3 and 4 are SR Domain edge routers

- o 5, 6, and 7 are all SR Domain routers

- o 8 and 9 are hosts within the SR Domain

- o 1 and 2 are hosts outside the SR Domain

o The SR domain is secured as per Section 5.1 and no external packet can enter the domain with a destination address equal to a segment of the domain.

## 6.3. Source SR Node

### 6.3.1. Intra SR Domain Packet

When host 8 sends a packet to host 9 via an SR Policy <S7,A9> the packet is

P1: (A8,S7)(A9,S7; SL=1)

#### 6.3.1.1. Reduced Variant

When host 8 sends a packet to host 9 via an SR Policy <S7,A9> and it wants to use a reduced SRH, the packet is

P2: (A8,S7)(A9; SL=1)

### 6.3.2. Inter SR Domain Packet - Transit

When host 1 sends a packet to host 2, the packet is

P3: (A1,A2)

The SR Domain ingress router 3 receives P3 and steers it to SR Domain egress router 4 via an SR Policy <S7, S4>. Router 3 encapsulates the received packet P3 in an outer header with an SRH. The packet is

P4: (A3, S7)(S4, S7; SL=1)(A1, A2)

If the SR Policy contains only one segment (the egress router 4), the ingress Router 3 encapsulates P3 into an outer header (A3, S4). The packet is

P5: (A3, S4)(A1, A2)

#### 6.3.2.1. Reduced Variant

The SR Domain ingress router 3 receives P3 and steers it to SR Domain egress router 4 via an SR Policy <S7, S4>. If router 3 wants to use a reduced SRH, Router 3 encapsulates the received packet P3 in an outer header with a reduced SRH. The packet is

P6: (A3, S7)(S4; SL=1)(A1, A2)

### 6.3.3. Inter SR Domain Packet - Internal to External

When host 8 sends a packet to host 1, the packet is encapsulated for the portion of its journey within the SR Domain.  From 8 to 3 the packet is

P7: (A8,S3)(A8,A1)

In the opposite direction, the packet generated from 1 to 8 is

P8: (A1,A8)

At node 3 P8 is encapsulated for the portion of its journey within the SR domain.  Resulting in

P9: (A3,S8)(A1,A8)

At node 9 the outer IPv6 header is removed by S6 processing then processed again when received by A8.

## 6.4. Transit Node

Nodes 5 acts as transit nodes for packet P1, and sends packet

P1: (A8,S7)(A9,S7;SL=1)

on the interface toward node 7.

## 6.5. SR Segment Endpoint Node

Node 7 receives packet P1 and, using the logic in Section 4.3.1, sends packet

P7: (A8,A9)(A9,S7; SL=0)

on the interface toward router 6.

## 6.6. Delegation of Function with HMAC Verification

This section describes how a function may be delegated within the SR Domain to non SR source nodes.  In the following sections consider a host 8 connected to a top of rack 5.

### 6.6.1. SID List Verification

An operator may prefer to add the SRH at source 8, while 5 verifies the SID list is valid.

For illustration purpose, an SDN controller provides 8 an SRH terminating at node 9, with segment list <S5,S7,S6,A9>, and HMAC TLV computed for the SRH. The HMAC key is shared with 5, node 8 does not know the key. Node 5 is configured with an IACL applied to the interface connected to 8, requiring HMAC verification for any packet destined to S/s.

Node 8 originates packets with the received SRH with HMAC TLV.

P15:(A8,S5)(A9,S6,S7,S5;SL=3;HMAC)

Node 5 receives and verifies the HMAC for the SRH, then forwards the packet to the next segment

P16:(A8,S7)(A9,S6,S7,S5;SL=2;HMAC)

Node 6 receives

P17:(A8,S6)(A9,S6,S7,S5;SL=1;HMAC)

Node 9 receives

P18:(A8,A9)(A9,S6,S7,S5;SL=0;HMAC)

This use of an HMAC is particularly valuable within an enterprise based SR Domain [SRN].

## 7. Security Considerations

This section reviews security considerations related to the SRH, given the SRH processing and deployment models discussed in this document.

As describe in Section 5, it is necessary to filter packets ingress to the SR Domain destined to segments within the SR Domain. This ingress filtering is via an IACL at SR Domain ingress border nodes. Additional protection is applied via an IACL at each SR Segment Endpoint node, filtering packets not from within the SR Domain, destined to SIDs in the SR Domain. ACLs are easily supported for small numbers of prefixes, making summarization important, and when the prefixes requiring filtering is kept to a seldom changing set.

Additionally, ingress filtering of IPv6 source addresses as recommended in BCP38 SHOULD be used.

### 7.1. Source Routing Attacks

[RFC5095] deprecates the Type 0 Routing header due to a number of significant attacks that are referenced in that document. Such attacks include bypassing filtering devices, reaching otherwise unreachable Internet systems, network topology discovery, bandwidth exhaustion, and defeating anycast.

Because this document specifies that the SRH is for use within an SR domain protected by ingress filtering via IACLs; such attacks cannot be mounted from outside an SR Domain. As specified in this document, SR Domain ingress edge nodes drop packets entering the SR Domain destined to segments within the SR Domain.

Additionally, this document specifies the use of IACL on SR Segment Endpoint nodes within the SR Domain to limit the source addresses permitted to send packets to a SID in the SR Domain.

Such attacks may, however, be mounted from within the SR Domain, from nodes permitted to source traffic to SIDs in the domain. As such, these attacks and other known attacks on an IP network (e.g. DOS/DDOS, topology discovery, man-in-the-middle, traffic interception/siphoning), can occur from compromised nodes within an SR Domain.

### 7.2. Service Theft

Service theft is defined as the use of a service offered by the SR Domain by a node not authorized to use the service.

Service theft is not a concern within the SR Domain as all SR Source nodes and SR segment endpoint nodes within the domain are able to utilize the services of the Domain. If a node outside the SR Domain learns of segments or a topological service within the SR domain, IACL filtering denies access to those segments.

### 7.3. Topology Disclosure

The SRH may contains SIDs of some intermediate SR-nodes in the path towards the destination, this reveals those addresses to attackers if they are able to intercept packets containing SRH.

This is applicable within an SR Domain but the disclosure is less relevant as an attacker has other means of learning topology.

### 7.4. ICMP Generation

The generation of ICMPv6 error messages may be used to attempt denial-of-service attacks by sending an error-causing destination address or SRH in back-to-back packets. An implementation that correctly follows Section 2.4 of [RFC4443] would be protected by the ICMPv6 rate-limiting mechanism.

## 8. IANA Considerations

This document makes the following registrations in the Internet Protocol Version 6 (IPv6) Parameters "Routing Type" registry maintained by IANA:

| Suggested Value | Description | Reference |
|---|---|---|
| 4 | Segment Routing Header (SRH) | This document |

This document request IANA to create and maintain a new Registry: "Segment Routing Header TLVs"

### 8.1. Segment Routing Header Flags Register

This document requests the creation of a new IANA managed registry to identify SRH Flags Bits. The registration procedure is "Expert Review" as defined in [RFC8126]. Suggested registry name is "Segment Routing Header Flags". Flags is 8 bits, the following bits are defined in this document:

| Suggested Bit | Description | Reference |
|---|---|---|
| 4 | HMAC | This document |

### 8.2. Segment Routing Header TLVs Register

This document requests the creation of a new IANA managed registry to identify SRH TLVs. The registration procedure is "Expert Review" as defined in [RFC8126]. Suggested registry name is "Segment Routing Header TLVs". A TLV is identified through an unsigned 8 bit codepoint value. The following codepoints are defined in this document:

| Suggested Value | Description | Reference |
|---|---|---|
| 5 | HMAC TLV | This document |
| 128 | Pad0 TLV | This document |
| 129 | PadN TLV | This document |

## 9. Implementation Status

This section is to be removed prior to publishing as an RFC.

### 9.1. Linux

Name: Linux Kernel v4.14

Status: Production

Implementation: adds SRH, performs END processing, supports HMAC TLV

Details: https://irtf.org/anrw/2017/anrw17-final3.pdf and [I-D.filsfils-spring-srv6-interop]

### 9.2. Cisco Systems

Name: IOS XR and IOS XE

Status: Pre-production

Implementation: adds SRH, performs END processing, no TLV processing

Details: [I-D.filsfils-spring-srv6-interop]

### 9.3. FD.io

Name: VPP/Segment Routing for IPv6

Status: Production

Implementation: adds SRH, performs END processing, no TLV processing

Details: https://wiki.fd.io/view/VPP/Segment_Routing_for_IPv6 and [I-D.filsfils-spring-srv6-interop]

### 9.4. Barefoot

Name: Barefoot Networks Tofino NPU

Status: Prototype

Implementation: performs END processing, no TLV processing

Details: [I-D.filsfils-spring-srv6-interop]

### 9.5. Juniper

Name: Juniper Networks Trio and vTrio NPU's

Status: Prototype & Experimental

Implementation: SRH insertion mode, Process SID where SID is an interface address, no TLV processing

### 9.6. Huawei

Name: Huawei Systems VRP Platform

Status: Production

Implementation: adds SRH, performs END processing, no TLV processing

## 10. Contributors

Kamran Raza, Darren Dukes, Brian Field, Daniel Bernier, Ida Leung, Jen Linkova, Ebben Aries, Tomoya Kosugi, Eric Vyncke, David Lebrun, Dirk Steinberg, Robert Raszuk, Dave Barach, John Brzozowski, Pierre Francois, Nagendra Kumar, Mark Townsley, Christian Martin, Roberta Maglione, James Connolly, Aloys Augustin contributed to the content of this document.

## 11. Acknowledgements

The authors would like to thank Ole Troan, Bob Hinden, Ron Bonica, Fred Baker, Brian Carpenter, Alexandru Petrescu, Punit Kumar Jaiswal, and David Lebrun for their comments to this document.

## 12. References

### 12.1. Normative References

[FIPS180-4]
National Institute of Standards and Technology, "FIPS 180-4 Secure Hash Standard (SHS)", March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[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>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

[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>.

[RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <https://www.rfc-editor.org/info/rfc6407>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

## 12.2. Informative References

[I-D.filsfils-spring-srv6-interop] Filsfils, C., Clad, F., Camarillo, P., Abdelsalam, A., Salsano, S., Bonaventure, O., Horn, J., and J. Liste, "SRv6 interoperability report", draft-filsfils-spring-srv6-interop-01 (work in progress), September 2018.

[I-D.filsfils-spring-srv6-network-programming] Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-06 (work in progress), October 2018.

[I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-18 (work in progress), December 2018.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[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>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <https://www.rfc-editor.org/info/rfc5308>.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <https://www.rfc-editor.org/info/rfc5340>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[SRN] and , "Software Resolved Networks: Rethinking Enterprise Networks with IPv6 Segment Routing", 2018, <https://inl.info.ucl.ac.be/system/files/sosr18-final15-embedfonts.pdf>.

## Authors' Addresses

Clarence Filsfils (editor)
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Stefano Previdi
Huawei
Italy

Email: stefano@previdi.net

John Leddy
Individual
US

Email: john@leddy.net

Satoru Matsushima
Softbank

Email: satoru.matsushima@g.softbank.co.jp

Daniel Voyer (editor)
Bell Canada

Email: daniel.voyer@bell.ca