Compressed SRv6 Segment List Encoding

draft-ietf-spring-srv6-srh-compression

Weiqiang Cheng, Clarence Filsfils, Zhenbin Li, Bruno Decraene, Francois Clad

IETF 119, Brisbane, March 2024

Status update

- Addressed reviews from Adrian and Boris (revision -10)
- Closed issue #1 “Clarity on one or multiple data plane solutions” (Jan 18)
- Working-group last call (Jan 21 – Feb 4)
  - Pablo Camarillo accepted to be the document shepherd. Thanks!
  - Lots of support. Thanks to all!
  - Addressed comments from Martin and Dhruv (revision -12)
  - One outstanding comment from Andrew
- New review from Alvaro Retana (Feb 7)
  - Tentatively addressed in revisions -13 and -14

Main changes since revision -09

- Section 4
  - Added operational guidance on C-SID flavor selection

> SRv6 is intended for use in a variety of networks that require different prefix lengths and SID numbering spaces. Each of the two flavors introduced in this document comes with its own recommendations for Locator-Block and C-SID length, as specified in Section 4.1 and Section 4.2. These flavors are best suited for different environments, depending on the requirements of the network. For instance, larger C-SID lengths may be more suitable for networks requiring ample SID numbering space, while smaller C-SID lengths are better for compression efficiency. The two compression flavors allow the compressed segment list encoding to adapt to a range of requirements, with support for multiple compression levels. Network operators can choose the flavor that best suits their use case, deployment design, and network scale.

  - Clarified Locator-Block and C-SID length implementation requirements and operational guidance

> An implementation MUST support a 32-bit Locator-Block length (LBL) and a 16-bit C-SID length (LNFL) for NEXT-C-SID flavor SIDs, and may support any other Locator-Block and C-SID length.
>
> A deployment should use a consistent Locator-Block length and C-SID length for all SIDs of the SR domain. Heterogeneous lengths, while possible, may impact the compression efficiency.

> The REPLACE-C-SID flavor SIDs support any Locator-Block length (LBL), depending on the needs of the operator, as long as it does not exceed 128-LNFL-ceiling(log_2(128/LNFL)) (ceiling(x) is the least integer greater than or equal to x [GKP94]), so that enough bits remain available for the C-SID and Argument. A Locator-Block length of 48, 56, 64, 72, or 80 bits is recommended for easier reading in operation.

> This document defines the REPLACE-C-SID flavor for 16-bit and 32-bit C-SID lengths (LNFL). An implementation MUST support a 32-bit C-SID length for REPLACE-C-SID flavor SIDs.

> A deployment should use a consistent Locator-Block length and C-SID length for all SIDs of the SR domain. Heterogeneous C-SID lengths, while possible, may impact the compression efficiency.

Main changes since revision -09

- Section 6
  - Clarified that SIDs with a missing or invalid SID structure are incompressible

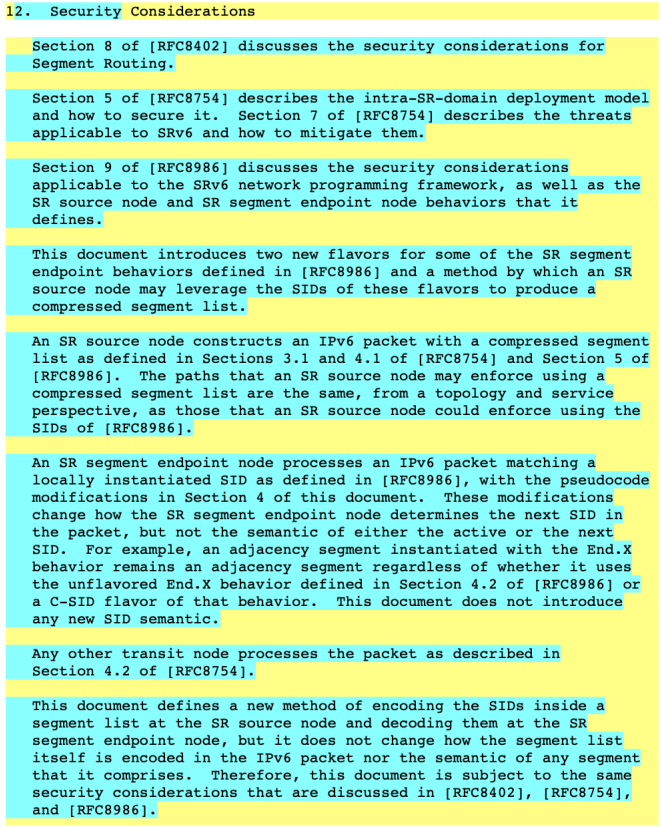
> When compressing a segment list, the SR source node MUST treat an invalid SID structure as unknown, and treats the SID as incompressible.

  - Added a sub-section on upper-layer checksum calculation at the SR source node

> 6.5. Upper-Layer Checksums
>
> The Destination Address used in the IPv6 pseudo-header (Section 8.1 of [RFC8200]) is that of the ultimate destination.
>
> At the originating node, that address will be the Destination Address as it is expected to be received by the ultimate destination. When the last element in the compressed segment list is a C-SID container, this address can be obtained from the last element in the uncompressed segment list or by repeatedly applying the segment behavior as described in Section 9.2. This applies regardless of whether an SRH is present in the IPv6 packet or omitted.
>
> At the recipient(s), that address will be in the Destination Address field of the IPv6 header.

Main changes since revision -09 (2)

- Section 7
  - Reworked to align with the rest of the document
- Section 12
  - Expanded to explain how C-SID fits within the security model of RFC 8402, 8754, and 8986
- Appendix
  - Removed “Open Issues”
  - Added full pseudocodes for all the SIDs of this document



12. Security Considerations

Section 8 of [RFC8402] discusses the security considerations for Segment Routing.

Section 5 of [RFC8754] describes the intra-SR-domain deployment model and how to secure it. Section 7 of [RFC8754] describes the threats applicable to SRv6 and how to mitigate them.

Section 9 of [RFC8986] discusses the security considerations applicable to the SRv6 network programming framework, as well as the SR source node and SR segment endpoint node behaviors that it defines.

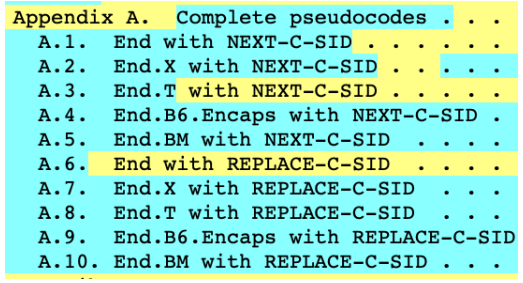
This document introduces two new flavors for some of the SR segment endpoint behaviors defined in [RFC8986] and a method by which an SR source node may leverage the SIDs of these flavors to produce a compressed segment list.

An SR source node constructs an IPv6 packet with a compressed segment list as defined in Sections 3.1 and 4.1 of [RFC8754] and Section 5 of [RFC8986]. The paths that an SR source node may enforce using a compressed segment list are the same, from a topology and service perspective, as those that an SR source node could enforce using the SIDs of [RFC8986].

An SR segment endpoint node processes an IPv6 packet matching a locally instantiated SID as defined in [RFC8986], with the pseudocode modifications in Section 4 of this document. These modifications change how the SR segment endpoint node determines the next SID in the packet, but not the semantic of either the active or the next SID. For example, an adjacency segment instantiated with the End.X behavior remains an adjacency segment regardless of whether it uses the unflavored End.X behavior defined in Section 4.2 of [RFC8986] or a C-SID flavor of that behavior. This document does not introduce any new SID semantic.

Any other transit node processes the packet as described in Section 4.2 of [RFC8754].

This document defines a new method of encoding the SIDs inside a segment list at the SR source node and decoding them at the SR segment endpoint node, but it does not change how the segment list itself is encoded in the IPv6 packet nor the semantic of any segment that it comprises. Therefore, this document is subject to the same security considerations that are discussed in [RFC8402], [RFC8754], and [RFC8986].



Appendix A. Complete pseudocodes . . .
A.1. End with NEXT-C-SID . . . . . .
A.2. End.X with NEXT-C-SID . . . . . .
A.3. End.T with NEXT-C-SID . . . . . .
A.4. End.B6.Encaps with NEXT-C-SID .
A.5. End.BM with NEXT-C-SID . . . . .
A.6. End with REPLACE-C-SID . . . . .
A.7. End.X with REPLACE-C-SID . . . .
A.8. End.T with REPLACE-C-SID . . . .
A.9. End.B6.Encaps with REPLACE-C-SID
A.10. End.BM with REPLACE-C-SID . . .

Andrew Alston — L4 checksum

- Andrew Alston:
  - “[…] the behavior described in this document break the L4 checksum as defined in RFC8200.”
  - “I call breaking any middleware that does checksum validation a problem - and a big one”
- Robert Raszuk:
  - “If I read RFC8200 L4 checksum is computed by the packet *originator and validated by the packet's ultimate receiver.”
  - “[…] from my experience the "middleboxes" RFC3234 talks about are placed at the edges or peripherals of the core networks (example DMZs). In those parts of the network rarely anyone runs any form of SRv6 be it with or without SRH.”
- Antoine Fressancourt:
  - “I tend to agree with Andrew that the fact that the verification of a L4 checksum by a middlebox breaks is a problem. But I think this is a huge problem with the middleboxes, not with SRv6.”
- Tal Mizrahi:
  - “I am curious to know whether there are existing middleboxes that can verify the L4 checksum for packets with an SRH.”
  - “RFC 8200 […] does not say anything about middleware.”
- Nick Buraglio:
  - “What I have not seen is a middle box that does, as far as I know, L4 checksum checking, or that understands SRv6 […].”
- Mark Smith:
  - “If the payload of the SRv6 packet is another IP packet or layer 2 frame e.g. Ethernet […], then the layer 4 checksum issue probably wouldn't occur […]”
  - “However, if SRv6 was used to to directly carry an upper layer transport layer protocol PDU like UDP, TCP or ICMPv6, then that's when the checksum/middlebox issue arises […].”
  - “[…] I don't really see how including an SRH in OAM packets solves the problem unless the middlebox is SRv6 aware.”

Next Steps

- Close WGLC