SPRING Y. Ueno, Ed.
Internet-Draft NTT Communications Corporation
Intended status: Standards Track R. Nakamura
Expires: April 29, 2020 The University of Tokyo
T. Kamata
Cisco Systems, Inc.
October 27, 2019

SRv6 Tagging proxy
draft-eden-srv6-tagging-proxy-00

Abstract

This document describes the tagging method of SRv6 proxy. SRv6 proxy is an SR endpoint behavior for processing SRv6 traffic on behalf of an SR-unaware service.

include Simplified BSD License text as described in Section 4.e of

Table of Contents

1. Introduction

Segment Routing (SR) is a source routing architecture defined in [RFC8402]. SR uses segment identifiers (SIDs) to identify each entity in an SR network. SR can be applied two types of data plane, MPLS and IPv6. IPv6 based SR is called Segment Routing IPv6 (SRv6) and its header format is defined in [I-D.ietf-6man-segment-routing-header]. As for the SRv6 packets, the SIDs are embedded in packets in the form of a list with the current index of the list, called SegmentsLeft (SL). Packets with Segment Routing Headers (SRHs) are steered through the ordered list of SIDs. Note that the proxy behavior defined in this document can only be applied for SRv6 packets.

Because SR can steer packets through arbitrary SR nodes, SR can be applied to Service Function Chaining (SFC). SFC, defined in [RFC7665], is an architecture that realizes the on-demand instantiation of an ordered set of service functions. Although there are differences in the specific packet steering method, SR defined in [RFC8402] can realize SFC and [I-D.xuclad-spring-sr-service-programming] describes SR proxy behaviors to integrate SR-unaware services to it.

This document describes a new SRv6 proxy, called tagging proxy. The tagging proxy, which is a variant of the dynamic SR proxy, supports

both IPv4 and IPv6 and multiple service chains by one proxy instance without state management.

2. Terminology

This document leverages the terminology proposed in [RFC8402], [I-D.ietf-spring-segment-routing-policy], and [I-D.xuclad-spring-sr-service-programming].

3. SRv6 Tagging proxy

The proxy is a variant of the dynamic proxy defined in [I-D.xuclad-spring-sr-service-programming]. The dynamic proxy caches the outer IPv6 header and SRH before removing it from the incoming traffic. After removal of the outer IPv6 and SRH headers, the dynamic proxy sends the traffic to an associating service and the same headers are re-attached to the traffic returning from the service.

For caching outer headers, the tagging proxy uses arguments of SRv6 SIDs as indexes for cache entires. The arguments are determined by the operator to correspond one-to-one with the service chains, and the process could be automated by the network controllers. Upon receiving a packet whose active segment matches a tagging SR proxy function, the proxy node caches the IPv6 header and SRH. Corresponding cache entry for a packet is indicated by an argument part of the SRv6 SID. Every time a packet arrives, a corresponding cache entry is updated.

The tagging proxy removes the IPv6 header and SRH for sending the inner packet to the SR-unaware service. At that time the tagging proxy treats the index as a "tag", that is embedded into the inner packet. As a field to embed the tag, Type of Service (ToS) is used for IPv4 packets and Traffic Class (TC) is used for IPv6 packets. Note that the argument length of the SID for tagging proxy cannot be greater than 8-bit because of the length of ToS and TC fields.

When the proxy node receives the packet returning from the SR-unawre service, the proxy node pushes the IPv6 header and SRH onto the packet. The headers are retrieved from the cache entry that corresponds to the tag extracted from the ToS or TC field of the packet.

A tagging SR proxy segment is associated with the following mandatory parameters:

- NH-ADDR: Next hop Ethernet address (only for inner type IPv4 and IPv6)

- IFACE-OUT: Local interface for sending traffic towards the service

- IFACE-IN: Local interface receiving the traffic coming back from the service

A tagging SR proxy segment is thus defined for a specific service. It is also bound to a pair of directed interfaces on the proxy. These may be both directions of a single interface, or opposite directions of two different interfaces. The latter is recommended in case the service is to be used as part of a bi-directional SR SC policy. If the proxy and the service both support 802.1Q, IFACE-OUT and IFACE-IN can also represent sub-interfaces.

3.1. SRv6 pseudocode

3.1.1. Tagging proxy for inner type IPv4

Upon receiving an IPv6 packet destined for S, where S is an IPv6 tagging proxy segment for IPv4 traffic, a node N does:

```
1.   IF NH=SRH & SL > 0 & ENH == 4 THEN
2.       Cache IPv6 Header and SRH into CACHE[ARG]
3.       Remove the (outer) IPv6 header and its extension headers
4.       Embed ARG into the ToS field of the (inner) IPv4 header
5.       Forward the exposed packet on IFACE-OUT towards NH-ADDR
6.   ELSE
7.       Drop the packet
```

Upon receiving a non-link-local IPv4 packet on IFACE-IN, a node N does:

```
1.   IF CACHE[ToS] THEN
2.       Set ToS value to 0
3.       Decrement TTL and update checksum of the inner IPv4 header
4.       Push the IPv6 header and SRH in CACHE[ToS]
5.       Set ENH value to 4
6.       Update the payload length of the outer IPv6 header
7.       Lookup outer DA in appropriate table and proceed accordingly
8.   ELSE
9.       Drop the packet
```

Note that the proxy may cache and restore the ToS value of inner IPv4 packet in addition to outer IPv6 header and SRH if the service chain uses single ToS value.

3.1.2. Tagging proxy for inner type IPv6

Upon receiving an IPv6 packet destined for S, where S is an IPv6 tagging proxy segment for IPv6 traffic, a node N does:

```
1.   IF NH=SRH & SL > 0 & ENH == 41 THEN
2.       Cache IPv6 Header and SRH into CACHE[ARG]
3.       Remove the (outer) IPv6 header and its extension headers
4.       Embed ARG into the TC field of the (inner) IPv6 header
5.       Forward the exposed packet on IFACE-OUT towards NH-ADDR
6.   ELSE
7.       Drop the packet
```

Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N does:

```
1.   IF CACHE[TC] THEN
2.       Set TC value to 0
3.       Decrement Hop Limit of the inner IPv6 header
4.       Push the IPv6 header and SRH in CACHE[TC]
5.       Set ENH value to 41
6.       Update the payload length of the outer IPv6 header
7.       Lookup outer DA in appropriate table and proceed
accordingly
8.   ELSE
9.       Drop the packet
```

Note that the proxy may cache and restore the TC value of inner IPv6 packet in addition to outer IPv6 header and SRH if the service chain uses single ToS value.

4. Implementation status

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

The tagging SR proxy is available on the below open-source implementations.

- Linux XDP based implementation by Yukito Ueno
- Linux kernel based implementation (out-of-tree) by Ryo Nakamura

Also, both implementations were operated for the traffic of exhibitors and visitors at Interop Tokyo 2019 ShowNet.

5. Discussion

This tagging proxy uses ToS or Traffic Class field as a container of an index of a cache entry. Upon receiving a packet returning from an SR-unaware service, the index is needed for the proxy node to decide which cache entry should be pushed to the packet. On the other hand, the usage is different from the original purpose of ToS and TC fields.

6. Acknowledgements

The authors would like to thank all the members and contributors of Interop Tokyo 2019 ShowNet. The authors are also thankful to Francois Clad for his comments.

7. IANA Considerations

7.1. SRv6 Endpoint Behaviors

This I-D requests the IANA to allocate, within the "SRv6 Endpoint Behaviors" sub-registry belonging to the top-level "Segment-routing with IPv6 dataplane (SRv6) Parameters" registry, the following allocations:

Value	Description	Reference
TBA	End.AT - Tagging proxy	[This.ID]

8. Security Considerations

The security requirements and mechanisms described in [RFC8402], [I-D.ietf-6man-segment-routing-header] and [I-D.filsfils-spring-srv6-network-programming] also apply to this document. This document does not introduce any new security vulnerabilities.

9. References

9.1. Normative References

[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-07 (work in progress), February 2019.

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-26 (work in progress), October 2019.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-03 (work in progress), May 2019.

[I-D.xuclad-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca, d., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", draft-xuclad-spring-sr-service-programming-02 (work in progress), April 2019.

[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>.

9.2. Informative References

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

Authors' Addresses

Yukito Ueno (editor)
NTT Communications Corporation
Tokyo
JP

Phone: +80 90 3085 5274
Email: yukito.ueno@ntt.com

Ryo Nakamura
The University of Tokyo
Tokyo
JP

Phone: +81 3 5841 2710
Email: upa@haeena.net

Teppei Kamata
Cisco Systems, Inc.
Tokyo
JP

Email: tkamata@cisco.comt