Workgroup: Network Working Group
Internet-Draft: draft-menon-svr-00
Published: 1 October 2021
Intended Status: Informational
Expires: 4 April 2022
A A. Menon M. Baj
u Juniper Networks Juniper Networks
t
h
o
r
s
:
P. Timmons H. Kaplan
Juniper Networks Juniper Networks

Secure Vector Routing (SVR)

Abstract

This document describes Secure Vector Routing (SVR). Secure Vectors contain application layer metadata used for authentication and communicating network intent between data routers. The metadata is extensible, and could be used to transmit information, network routes, security policies, and quality policies securely across network boundaries. Boundaries include those formed by private network RFC1918 networks with the IPv4 public internet, and the IPv6 public internet.

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 4 April 2022.

Table of Contents

1. Introduction

There exists a need to extend network intent across multiple IP networks and paths to provide an end-to-end experience. Selection of specific paths and their attributes is a key intent. There is also a need for applications to communicate their intent to networks. This need is increasing as workloads move to public cloud and the numbers of cloud locations increase. The standard practice today is to use an overlay network of tunnels to create a virtual network. SVR is being proposed as an alternative to using tunnels, which simplifies the network by virtue of having only one network to manage and securely transport traffic with encryption and authentication; also, the absence of tunneling overhead reduces bandwidth. Since SVR specifies intent abstractly, it also has the capability to interwork policies between different networks and address spaces.

Most networks are deployed with a virtual private network (VPN) across IP backbone facilities. VPNs have the significant disadvantage of carrying additional payload on top of the actual packet thereby leading to IP fragmentation as well as reduced bandwidth. In this document, SVR is being proposed as a new technique which is session aware and does not have the overhead of IP tunnels.

1.1. Overview

A Secure Vector Route (SVR) describes a network intent and shares this intent in the form of metadata with a routing peer. The intent to a peer router is conveyed by means of a cookie, often referred to as first packet metadata, which is placed on the first packet that is targeted towards the peer. SVR is session aware on every router and sets up a biflow (forward and reverse flows) based on the intent. Once the session is set up, the cookie is not sent for the subsequent packets.

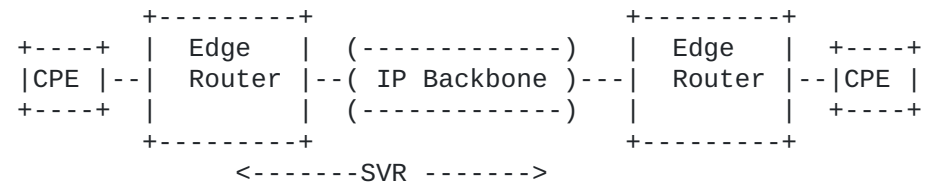


Figure 1

This intent must include:

*Authority: This defines the owner of the namespace to be used for defining policies. Each namespace owner can allocate Tenant names (representing collections of network endpoints sharing common network enforcement policy), and Service names (representing accessible destinations and traffic treatment policy). Authority namespace allocation will be managed like any other domain name reservation.

*Context: This is the original "5-tuple" of an IP packet, including source IP, source port, destination IP, destination port, and protocol. Optionally, Layer 2 information such as MAC Address or VLAN tags may be included for certain use cases if required.

*Signature: The metadata must be cryptographically signed using HMAC by the source router, so the next hop router can authenticate the sender of the data. The portion of the packet that is signed must not include the IP header, as it may go through a NAT or IPv4 to IPv6 conversion.

*Direction: This is the intended client to server direction. The initial network packet of a communication session indicates the direction. For example, a TCP SYN packet would travel from client to server, defining the direction of an intent. Forward direction is always client to server, and reverse is always server to client. These directions have nothing to do with a network topology (for example, hub and spoke), and a single network path could have forward sessions going bi-directionally -- traffic going from node A to node B may represent the forward direction for some sessions and the reverse direction for other sessions.

*Tenant(s): This is a textual description defining network endpoints that share common access policy (allowlists or blocklists to network destinations). These may be mapped using any known technique including source IP address mask, a VLAN tag, ingress interface, provided by an authentication system, or even client supplied, and this mapping is outside the scope of this document. Often these are location specific definitions, but the tenant has global meaning within an authority. Tenant names can conform to domain name syntax, and be expressed as hierarchical structures (i.e., location.department.company).

*Service(s): This is a textual description of what server(s) can be accessed with this intent. Examples include Zoom, or Office365/Outlook. Although outside the scope of this document, these could be defined with any known technique, including URLs, IP address(es) protocol(s) and port(s), CIDR block(s), etc. Having a single text name to describe a network destination makes applying network intent easier. This Service name can be used to define all other attributes of network intent including:

-Quality Policy: Associated or imputed by the Service, a description of the bandwidth quality that is required. These can be implementation specific.

-Security Policy: Associated or imputed by the Service, a description of what kinds of encryption may be required and security processing that is required. These can be implementation specific.

These additional attributes are outside the scope of this protocol as they are locally defined and associated locally to the specific Service.

When performing HMAC signatures, or payload encryption, key management is also outside the scope of this document. Standard IKEv2 techniques could be applied, or integration with authentication systems may be appropriate.

1.2. Qualified Service Name (QSN)

A Secure Vector Route can be described as a Qualified Service Name (QSN). This is shorthand for a human understanding of a specific network intent definition. This shorthand is used in use case examples, but is not part of the specific protocol.

```
QSN://[subtenant.]tenant.authority/service/subservice
```

The QSN conforms with the definition of a URL. The QSN describes the Direction, Tenant, and Service of an SVR. The QSN introduces the concept of a naming authority, which controls the namespace for subtenants, tenants, services, and subservices. Examples of QSNs include the following:

```
QSN://engineering.juniper/github/project
QSN://engineering.acme/github
```

These examples indicate that juniper and acme are different authorities, and can each completely describe a QSN without defining overlapping intents in any way. The first example is a name for the intent definition for engineering at juniper to access github for a specific project. In the second case, engineering at acme can access any resources defined as github. QSNs will be used in this document and are useful in describing a high-level intent.

2. Theory of operation of Secure Vector Routing

The SVR metadata must be inserted into existing packets directly after the L4 header, even if the resulting increase in packet size would cause the packet to require fragmentation. The metadata must be in the very first packet of a new session (TCP or UDP bidirectional flow) to have any role in path selection or security. Metadata may be sent in any subsequent packet to change/update the networking intent. The metadata is inserted into the payload portion of a packet to guarantee it makes it unchanged through the network. Packet lengths and checksums must be adjusted accordingly, although adjusting TCP sequence numbers is not necessary.

2.1. SVR Peer Definition

A SVR Peer is any client, server, or network element that can understand and participate in SVR. Having pre-shared cryptographic keys, and the ability to authenticate and decrypt metadata is a

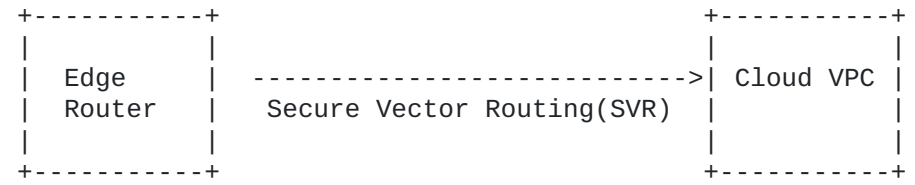
prerequisite to being a peer. Peers do not have to be directly adjacent, but reachable via IP networking, even across network boundaries. Examples of peers include:

A branch router and a data center edge router:

A client desktop computer and a nearby router:

A WIFI AP and an edge router:

A data center edge router and a VPC based router in a public cloud:



2.2. SVR Peer Paths

SVR Peers may have many different possible pathways between them. Each path must be discovered, and its service state should be known prior to sending metadata. Techniques such as BFD (RFC 5580) could be used to determine path availability. Each of these individual paths between peers is called a "peer path." Secure Vector routes are always defined and used between peers and can support session resiliency across multiple paths.

Peer paths are defined by their IP addresses or hostnames. Routers with multiple interfaces may have multiple peer paths to a next hop router. The IP addresses routers use on their interfaces are also referred to as waypoints. Every peer path can be defined by a pair of waypoint addresses. To avoid confusion when branch or edge devices obtain new IP addresses dynamically, these peer paths are assigned a hostname which is used instead. The named pathway and associated keys(ip-address/hostname and peer name) are provisioned and presumed available in this specification. Details of this provisioning are outside the scope of this document.

2.3. Directionality

The protocol utilizes the concept of direction. Direction is either forward or reverse; it is not tied to network topology, but rather the direction of session establishment. For TCP, the forward direction is always the client side towards the server side. For UDP, the forward direction is from the sender of the first packet. Reverse is the opposite direction.

These directions can be used to qualify a path between a pair of SVR Peers, the ingress and the egress (forward is from ingress to egress); or to qualify metadata (forward metadata is inserted by the ingress).

2.4. Metadata Handshake

To ensure the metadata is received and understood between peers, a handshake is performed. A router that supports SVR peer paths inserts metadata for each packet flow in the following circumstances:

* It is a "forward" packet representing a new session and the ingress node has not yet received any reverse metadata from the recipient egress node.

* It is a "reverse" packet from the recipient egress node to the initiating ingress node and recipient egress node has not received forward packets from this session without metadata.

These two comprise what is known as the "metadata handshake" -- that is, the initiating router includes metadata in all packets it sends to the recipient router until it receives a reverse packet with metadata from that recipient. Likewise, the recipient continues to send metadata to the initiating router until it receives a packet without metadata. This is how two routers acknowledge receipt of metadata from their counterparts: the absence of metadata in a packet indicates that it has received metadata from its counterpart.

2.5. Path Obstructions

Firewalls and middleboxes that sit along a peer path may not tolerate TCP SYN messages with data in the payload, or may verify sequence numbers in TCP streams (which are invalidated due to the inclusion of SVR metadata). The two devices that represent the peer path endpoints may determine through testing if there is a firewall, NAT, or other active middlebox between the two routers. Procedures for this (like STUN/TURN/ICE) are well known, and not included in this document. If a middlebox is detected, the packets can be UDP-transformed ie; the protocol byte can be changed from TCP to UDP by the transmitting router and restored to TCP by the receiving router for packets flowing in both directions. The sequence number of the TCP packet will be copied to the TCP checksum location as it will be overwritten by the UDP header. The original protocol in use is part of the context in the metadata and can be restored by the receiving peer.

2.6. Metadata removal

To prevent breaking any applications, there must be a 100% guarantee that metadata inserted by a participating SVR device is removed prior to the consumption of the data by the application service. If the client and server support metadata, then the network intent can be sent end-to-end. When a mid-stream packet router wants to insert SVR metadata, it must guarantee that the packet is directed to a next hop device that will understand and remove the metadata.

2.7. Modification of transport addresses

To guarantee that the packet will go to a specific device, the destination address for the packet is changed to the waypoint address of the next hop. Because the original addresses are stored in the context field, they can be recovered if needed. This is similar to IPv6 segment routing or a LISP RLOC with the exception that the original addresses are stored in metadata within the payload portion of the packet, and not the IP Header.

To communicate to the next hop the origin of the packet and to define and open a return path, the source address of the packet is NATted to the interface of the source router. This is called a source waypoint address. This provides a return path and can be used to guarantee symmetric flows if desired. Once again, the original addresses are state information that the source router must maintain.

2.8. Unique 5-Tuples for Every Session

To avoid sharing a hash with all traffic, and to make sessions completely independent, the source port and destination port can be assigned any values that are unique by the source router. When there are no NATs between the two router interfaces, this permits 2^32 (4,294,967,296) different unique sessions on a peer path. If there are source NATs, this will be reduced to 2^16 (65,536) different unique sessions. Ports can be reassigned if not in active use. It is also possible that middle boxes will limit what destination ports are permissible, reducing the number of possibilities. The range of ports that can be used could be discovered at run time by testing a peer path, provisioned, or both.

Note: The ingress SVR peer (client side) assigns both source and destination ports. The ingress side always chooses even ports for local (source port) and odd ports for remote(destination port) This provides total uniqueness between any two peers, with no negotiation or collision possibilities. Think of the two ports as a Sessions Identification Tag. Even if a session traveling in the opposite direction was allocated the same exact ports, because the source address and destination addresses would be swapped, the 5 tuples on the wire remain unique.

2.9. Session State Requirements

Each participant (peer) in secure vector routing must maintain state for every active session. This includes the full set of original addresses and translations required. This allows participants to stop sending metadata once it has been received by the peer. Should

one side lose state information, it can notify the other that it must send metadata to restore a session. Typically a path change will trigger metadata to be turned on for the subsequent forward packets so that the new session state including the waypoints can be maintained by the next hop router.

2.10. Waypoint Addresses

Each SVR router (peer) must statefully remember the source address that a session with metadata was received on. This may not be the same address the router sent a packet from due to a NAT or Firewall in the pathway. Routers use provisioned waypoint addresses, but statefully record and store the actual waypoint addresses.

3. Metadata Format and Composition

The format of metadata has both Header attributes as well as Payload attributes. Header attributes are always guaranteed to be unencrypted. These headers may be inspected by intermediate network elements but can't be changed. Payload attributes optionally can be encrypted by the sender. The pre-existing security association and key sharing is outside the scope of this document. Each attribute listed below will indicate whether it is a header attribute (unencrypted) or payload attribute (optionally encrypted). There are no attributes that can exist in both sections.

3.1. Metadata Header

The metadata header is shown below. A well-known "cookie" (0x4c48dbc6ddf6670c in network byte order or 0x0c67f6ddc6db484c in host byte order) is built into the header which is used in concert with contextual awareness of the packet itself to determine the presence of metadata within a packet. This is an eight-byte pattern that immediately follows the L4 header and is an indicator to a receiving router that a packet contains metadata. NOTE: Because packets are not normally directed to a routers interface, all packets arriving on a waypoint should include metadata or they will be dropped.

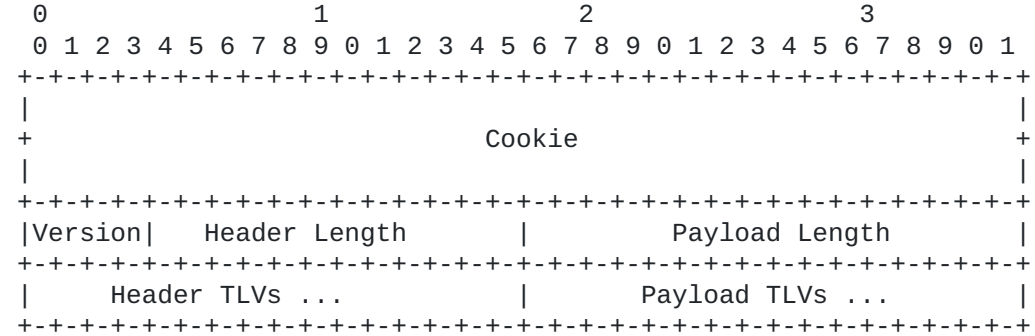


Figure 2

Cookie (8 bytes) The fingerprint of metadata. This value is used to determine the existence of metadata within a packet.

Version (4 bits) Field representing the version of the metadata header. The current version of metadata is 0x1.

Header Length (12 bits)
Length of the metadata header including any added optional attributes that are guaranteed to be unencrypted. The value of the number of bytes in the header.

Payload Length (2 bytes) Length of data following the metadata header, not including the size of the header. This data could be encrypted. The value of this field is the number of bytes in the payload.

3.1.1. False Positives

Given that no byte sequence is truly unique in the payload of a packet, in the scenario where the original payload after the L4 header contained the same byte sequence as the cookie, false positive logic is enacted on the packet. If the metadata signature can't verify that the metadata is valid, then a false positive metadata header is added to the packet to indicate that the first eight bytes of the payload matches the cookie. The structure of a false positive metadata packet is one which has a metadata header length that is the same as the base header size as well as having zero payload length. The receiving side of a packet with false positive metadata will strip out the metadata header if the next hop of the packet is not expecting a metadata header.

In the scenario where a router receives a false positive metadata header but intends to add metadata to the packet, the false positive metadata header is modified to contain the newly added attributes. Once attributes are added, the metadata header is no longer considered to be false positive.

3.1.2. Header Attributes

Header attributes are unencrypted and not associated with any one specific session, and do not have a forward or reverse direction. These are tied instead to the router peer path itself. The metadata header contains a 12-bit field associated with the header length of the metadata header. The field represents the overall length of the header. Its base value is 0xC which is the initial length of the metadata header.

The value 0xC comes from:

```
   64 bits  Cookie Length
    4 bits  Metadata Version
   12 bits  Header Length
+ 16 bits  Payload Length
--------------------------
96 bits = 12 bytes decimal = 0xC hex
```

Any value greater than 0xC would indicate that there are optional attributes associated with this metadata header which are guaranteed to be unencrypted.

An example of an optional attribute which would reside in the guaranteed unencrypted section of metadata would be per-packet fabric fragment attribute. This attribute is not associated with any session or encryption schema and as such must not be encrypted.

3.1.3. Payload Attributes

The metadata header contains a two-byte payload length field which is associated with attributes that can be encrypted.

3.1.4. Forward and Reverse Attributes

Each metadata payload attribute may be valid in the forward direction, the reverse direction, or both. If not valid, it is ignored quietly by the receiving side.

3.2. Header Attributes

3.2.1. Fragment

When a packet is fragmented to insert metadata, a new fragmentation mechanism must be added to prevent fragmentation attacks and to support reassembly (which requires protocol and port information). If a packet is received that IS a fragment, and it must transit through a metadata signaled pathway, it must also have this metadata attached to properly bind the fragment with the correct session.

All fragments will have a metadata header and the fragment TLV added to the guaranteed unencrypted portion of the metadata header. If the original packet already has a metadata header on it, the fragment TLV will be added to it.

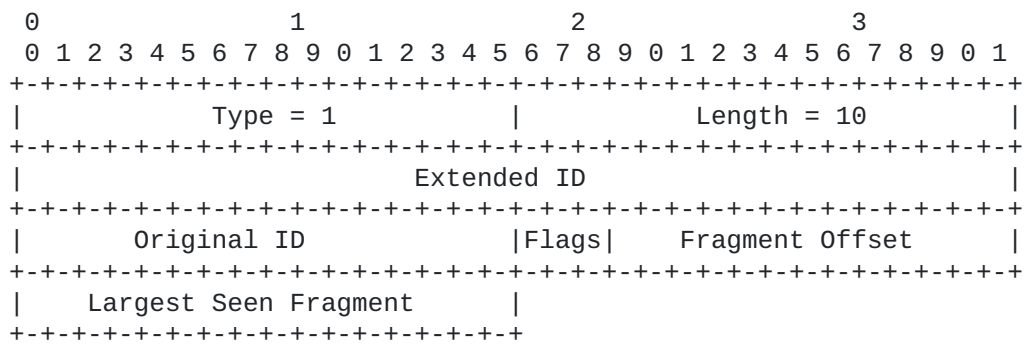


Figure 3

Type = 1 decimal (0x01) (2 bytes) Identifier for the Fragment TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Extended ID (4 bytes) Value used for differentiating fragment tuples.

Original ID (2 bytes) Original identification value of the L3 header.

Flags (3 bits) Field used for identifying fragment attributes. They are (in order, from most significant to least significant):

bit 0: Reserved; must be zero.

bit 1: Don't fragment (DF).

bit 2: More fragments (MF).

Fragment Offset (13 bits) Field associated with the number of eight-byte segments the fragment payload contains.

Largest Seen Fragment (2 bytes) Used along with a given egress network interface MTU to determine the fragment size of a reassembled packet.

3.2.2. Security Identifier

A versioning identifier used to determine which security key version should be used when handling features dealing with security and authenticity of a packet.

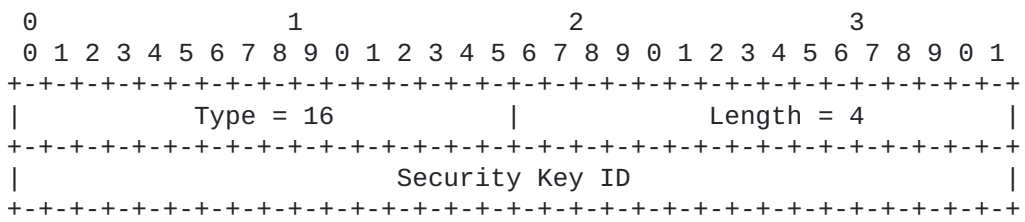


Figure 4

Type = 16 Decimal (0x10) (2 bytes) Identifier for the Security Identifier TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Security Identifier (4 bytes) This is a four-byte security key version identifier. This is used to identify the algorithmic version used for metadata authentication and encryption.

3.2.3. Disable Forward Metadata

An indication that forward metadata should be disabled. This is sent in the reverse metadata to acknowledge receipt of the metadata. This is the second part of the metadata handshake.

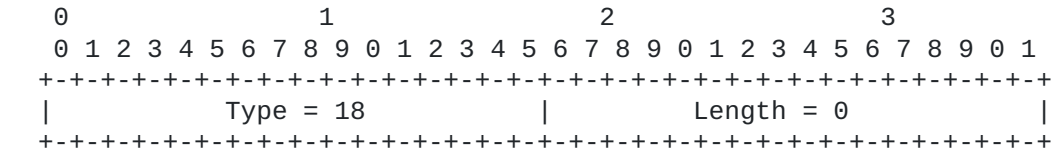


Figure 5

Type = 18 Decimal (0x12) (2 bytes) Note: The binding of the forward metadata to this is based on the 5-tuple address on the wire. Every session first packet with metadata will have a unique 5-tuple as it leaves the router. This unique address is used to bind a forward and reverse metadata. See metadata handshake in Section 2.4.

Length (2 bytes)
Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Note: This type does not contain any value as its existence in metadata indicates a true value.

3.2.4. IPv4 Router Egress Source Address

The IP address of the originating packet on the wire. This is sent by the source to the destination because there may be a NAT between the two routers. By recognizing the actual source packet on the wire, and comparing it with this field, a router can detect a NAT is present.

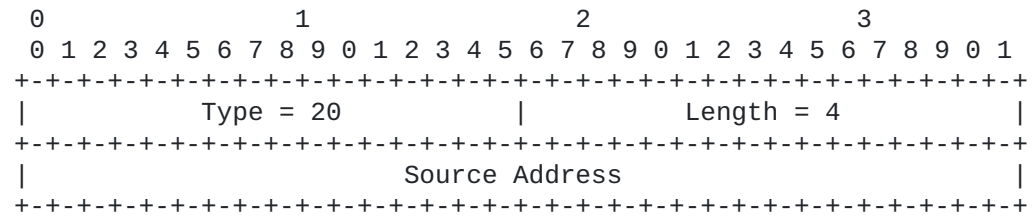


Figure 6

Type = 20 decimal (0x14) (2 bytes) Identifier for the Source Address (IPv4) TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Source Address (4 bytes) Original IPv4 source address of the originating router.

3.2.5. IPv4 Router Egress Source Address

The IP address of the originating packet. This is sent by the source to the destination because there may be a NAT between the two routers. By recognizing the actual source packet on the wire, and comparing it with this field, a router can detect a NAT is present.

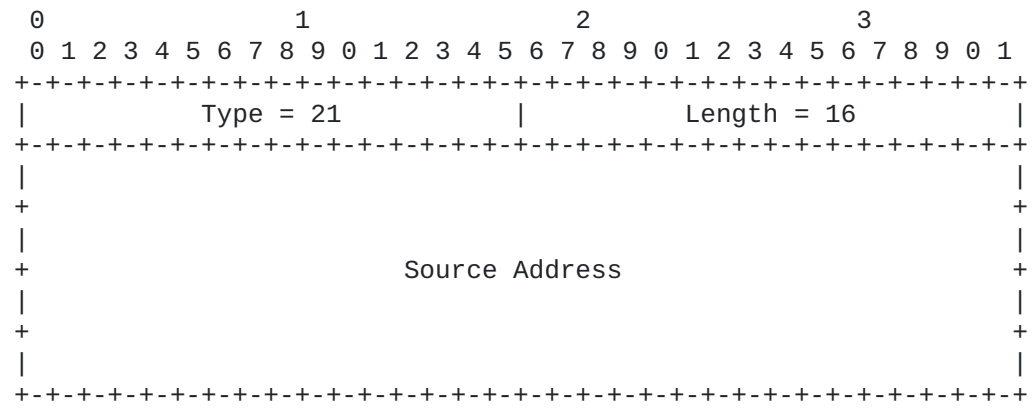


Figure 7

Type 21 decimal (Ox15) (2 bytes)
Identifier for the Source Address (IPv6) TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Source Address (16 bytes) Original IPv6 source address of the originating router.

3.2.6. Selective Acknowledgement

Used for TCP session optimization between peers. Allows sending side in long latency situations to queue packets and retransmit them to optimize TCP traffic.

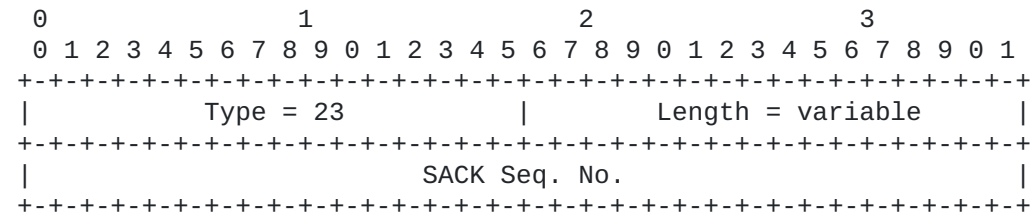


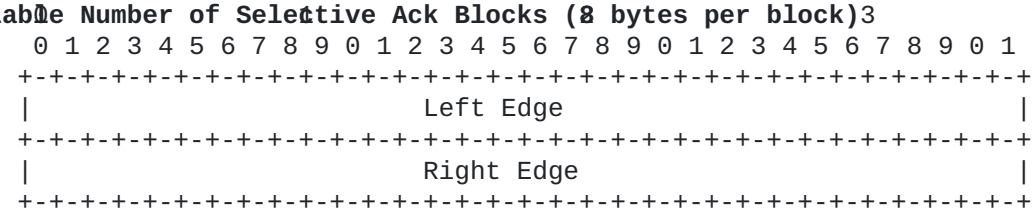
Figure 8

Type = 23 decimal (0x17) (2 bytes) Identifier for the Selective Acknowledgement TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

ACK Sequence Number (4 bytes) Sequence number for the last TCP message received.

Variable Number of Selective Ack Blocks (8 bytes per block)



Left Edge (4 bytes) Left Edge of the Selective Ack Block.

Right Edge (4 bytes) Right Edge of the Selective Ack Block.

3.2.7. SVR Protocol Message

The SVR Protocol Message is a force drop message that is used as an out-of-band signaling mechanism between 128T routers. These are messages that are internally generated and not intended to be forwarded through the router to another destination. Any sessions utilizing the 5 tuples defined by the received packet with metadata may be acted upon based on the drop reason.

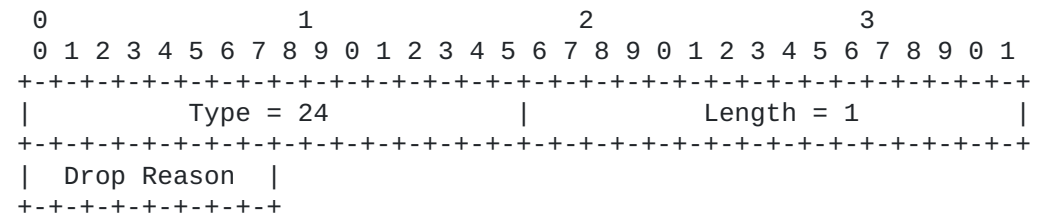


Figure 9

Type 24 decimal (0x18) (2 bytes) Identifier for the force drop TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Drop Reason (1 byte) Reason why this packet should be dropped.

- *0 = Unknown. This value is reserved and used for backwards compatibility.
- *1 = Keep Alive. A packet that is dropped by the receiving node. Used only to keep NAT pinholes alive on middleboxes.
- *2 = Enable Metadata. When packets are received on a 128T's wayport address and does not have metadata, the receiving 128T treats the packet as a new session, requesting the sending 128T to resend the packet with Metadata in the packet. If state is lost, this can help with recovery.
- *3 = Disable Metadata. An indication that forward metadata should be disabled. This is sent in the reverse metadata to acknowledge receipt of the metadata. If recovery is complete, this can be sent.

3.2.8. Path Metrics

Inline flow performance metrics to compute jitter, latency and loss per service, per path, per traffic class.

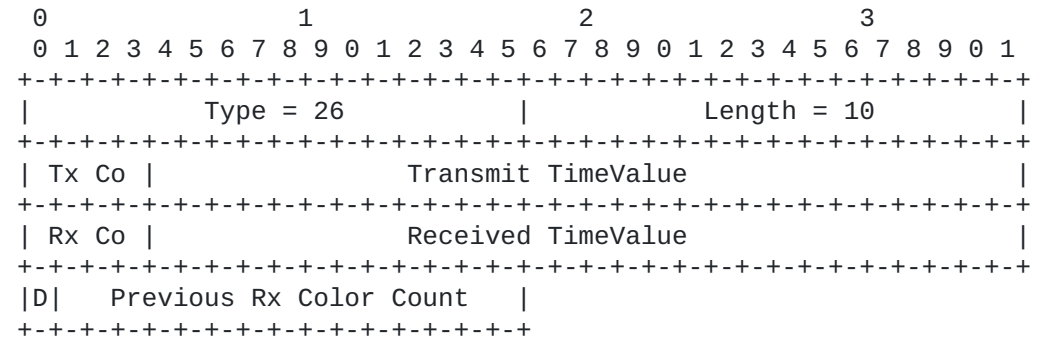


Figure 10

Type = 26 decimal (Ox1A) (2 bytes)
Identifier for the Path Metrics TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Transmit Color (4 bits) Current color of a transmitting node.

Transmit Time Value (28 bits) Current time value in milliseconds at time of marking. This time value represents the amount of time which has elapsed since the start of a transmit color.

Received Color (4 bits) Most recently received color from a remote node.

Receive Time Value (28 bits) Cached time value in milliseconds from adjacent node adjusted by the elapsed time between caching of the value and current time. This time value is associated with the received color.

Drop Bit (1 bit) Should this packet be dropped.

Previous Rx Color Count (15 bits) Number of packets received from the previous color block. This count is in reference to the color previous to the current RX color which is defined above.

3.2.9. Modify Request

A modify operation on an existing session, which diverts the packet to the service area for additional treatment and processing. This forces an "inline" session modify instead of allocating new waypoints to cause the same behavior.

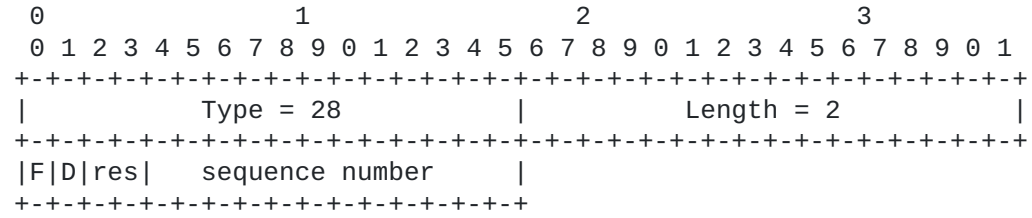


Figure 11

Type = 28 decimal (0x1C) (2 bytes) Identifier for the Modify Request TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

F (1 bit) Direction - forward or reverse. 0 = reverse. 1 = forward.

D (1 bit) Divert - Whether the modify can be handled inline in the fastpath or needs to be diverted to the service area. 1 = divert. 0 = do not divert.

Reserved (2 bits) Reserved for future use. Value is 00.

Sequence Number (12 bits)
The number of modifications the session has undergone.

3.3. Payload Attributes

Payload attributes are used for session establishment and can be encrypted to provide privacy. Encryption can be disabled for debugging.

3.3.1. Forward Context IPv4

The context contains a five-tuple associated with the original addresses, ports, and protocol of the packet. This is also known as the Forward Session Key.

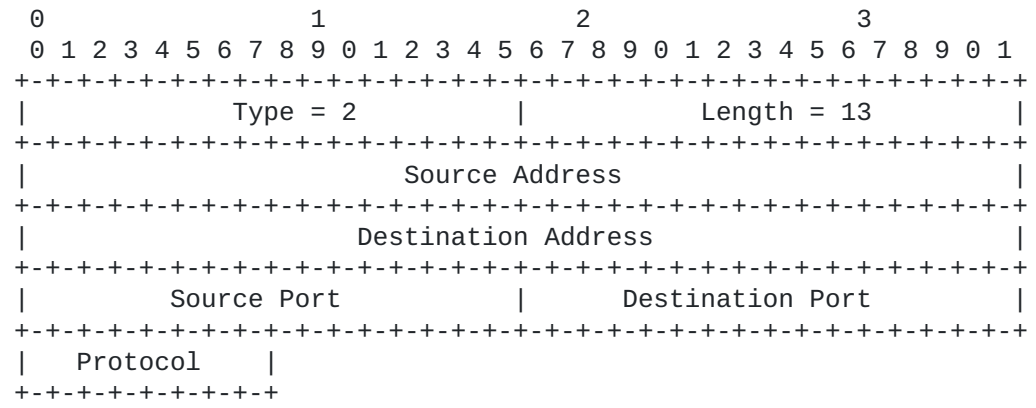


Figure 12

Type = 2 decimal (0x02) (2 bytes) Identifier for the Forward Session Key (IPv4) TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Source Address (4 bytes) Original IPv4 source address of the packet.

Destination Address (4 bytes) Original IPv4 destination address of the packet.

Source Port (2 bytes) Original source port of the packet.

Destination Port (2 bytes) Original destination port of the packet.

Protocol (1 byte) Original protocol of the packet.

3.3.2. Forward Context IPv6

A five-tuple associated with the original addresses, ports, and protocol of the packet for IPv6.

Figure 13

Type = 3 decimal (0x03) (2 bytes) Identifier for the Forward Session Key (IPv6) TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Source Address (16 bytes) Original IPv6 source address of the packet.

Destination Address (16 bytes) Original IPv6 destination address of the packet.

Source Port (2 bytes) Original source port of the packet.

Destination Port (2 bytes) Original destination port of the packet.

Protocol (1 byte) Original protocol of the packet.

3.3.3. Reverse Context IPv4

Five-tuple associated with the egress (router) addresses, ports, and protocol of the packet. Forward context and reverse context session keys are not guaranteed to be symmetrical due to functions which apply source NAT, destination NAT, or both to a packet before leaving the router.

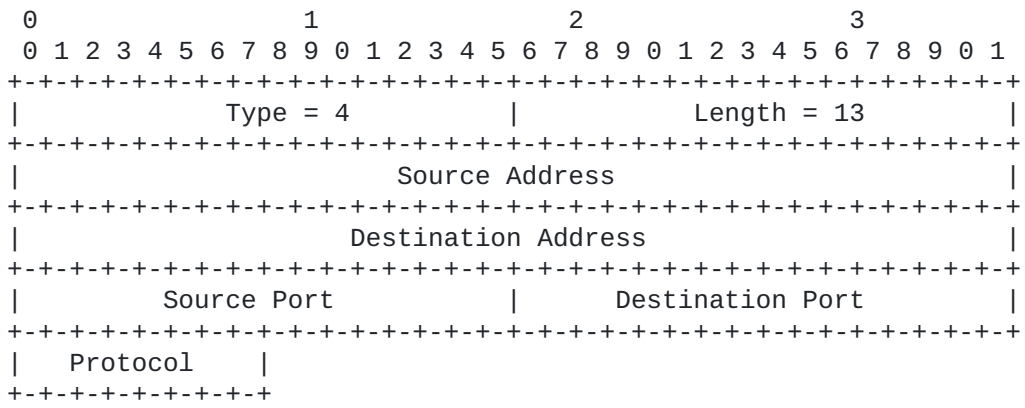


Figure 14

Type = 4 decimal (0x04) (2 bytes) Identifier for the Reverse Session Key (IPv4) TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Source Address (4 bytes) Egress IPv4 source address of the packet.

Destination Address (4 bytes) Egress IPv4 destination address of the packet.

Source Port (2 bytes) Egress source port of the packet.

Destination Port (2 bytes) Egress destination port of the packet.

Protocol (1 byte) Original protocol of the packet.

3.3.4. Reverse Context IPv6

Five-tuple associated with the egress (router) addresses, ports, and protocol of the packet. Forward and reverse session keys are not guaranteed to be symmetrical due to functions which apply source NAT, destination NAT, or both to a packet before leaving the router.

Figure 15

Type = 3 decimal (0x03) (2 bytes) Identifier for the Reverse Session Key (IPv6) TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Source Address (16 bytes) Egress IPv6 source address of the packet.

Destination Address (16 bytes) Egress IPv6 destination address of the packet.

Source Port (2 bytes) Egress source port of the packet.

Destination Port (2 bytes) Egress destination port of the packet.

Protocol (1 byte) Original protocol of the packet.

3.3.5. Session UUID

Unique identifier of a session. This is assigned by the peer that is initiating a session. Once assigned, it is maintained through all participating routers end-to-end.

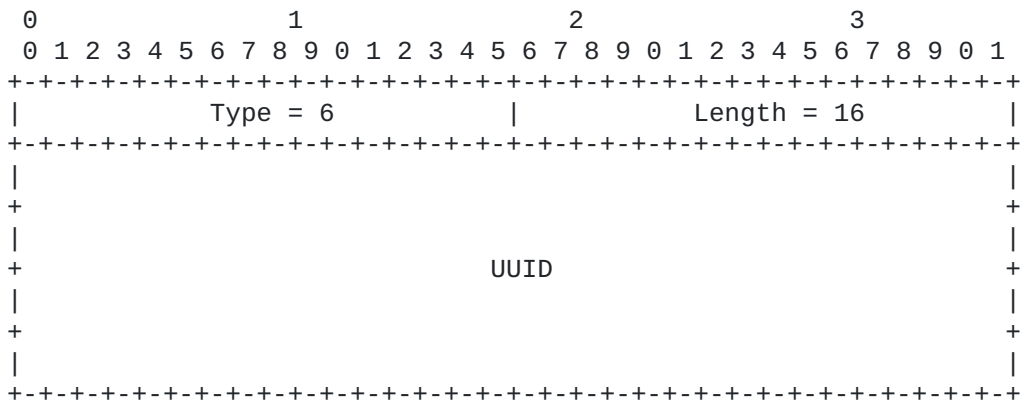


Figure 16

Type = 6 decimal (0x06) (2 bytes) Identifier for the Session UUID TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

UUID (16 bytes) Unique identifier of a session.

3.3.6. Source Tenant Name

An alphanumeric ASCII string which dictates what tenancy the session belongs to.

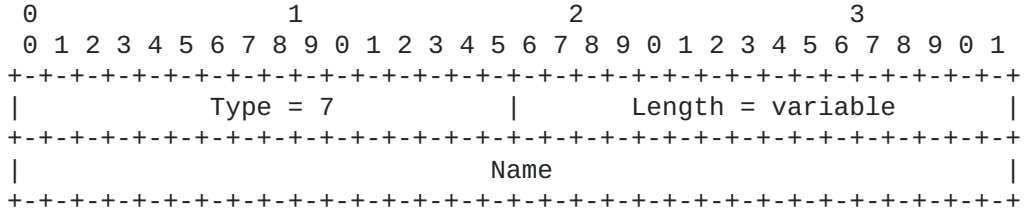


Figure 17

Type 7 decimal (0x07) (2 bytes) Identifier for the Tenant ID TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Name (variable length) The tenant name represented as a string.

3.3.7. Service Name

An alphanumeric string which dictates what service the session belongs to. This attribute must be present in all forward metadata packets.

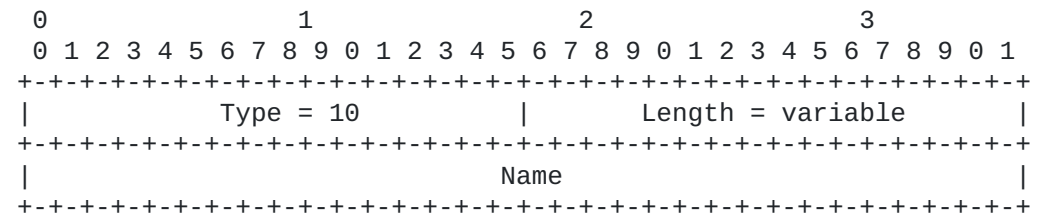


Figure 18

Type = 10 decimal (0x0A) (2 bytes) Identifier for the Service Name TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Name (variable length) The service name represented as a string.

3.3.8. Session Encrypted

Indicates if the session is having its payload encrypted. This enables payload encryption with tenant specific attributes and keys. Details of key management and encryption types are outside the scope of this standard.

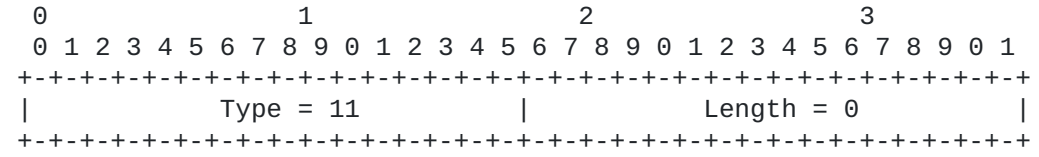


Figure 19

Type = 11 decimal (0x0B) (2 bytes) Identifier for the Session Encrypted TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Note: This type does not contain any value as its existence in metadata indicates a value.

3.3.9. TCP SYN Packet

Indicates if the packet in question is a TCP SYN packet. Due to the potential that this packet could have had its TCP header transformed to UDP temporarily, the flags in the TCP header may not be accessible.

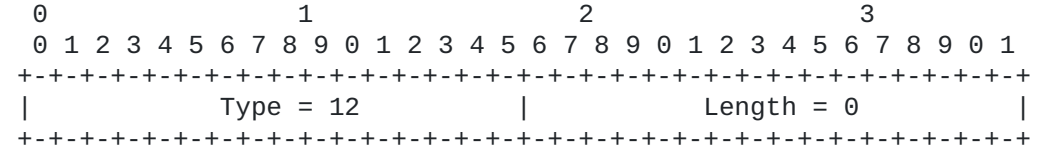


Figure 20

Type = 12 decimal (0x0C) (2 bytes)
Identifier for the TCP SYN packet TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Note: This type does not contain any value as its existence in metadata indicates a value.

3.3.10. Source Router Name

An alphanumeric string which dictates which source router the packet is originating from. This attribute may be present in all forward metadata packets if needed.

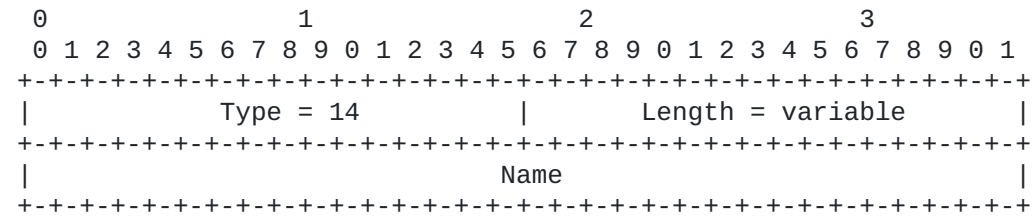


Figure 21

Type = 14 decimal (0x0E) (2 bytes) Identifier for the Source Router Name TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Name (variable length) The router name represented as a string.

3.3.11. Source Router Security Name

An alphanumeric string which dictates what security policy to be used for encrypting metadata when sending reverse packets back to this router. This attribute is optional.

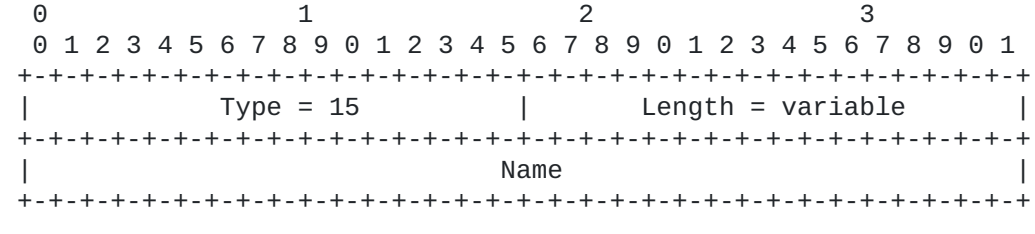


Figure 22

Type = 15 decimal (0x0F) (2 bytes) Identifier for the Source Router Security Name TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Name (variable length)
The peer security name represented as a string.

3.3.12. Destination Router Name

An alphanumeric string which dictates which destination router the packet is sent to. This attribute may be present if needed.

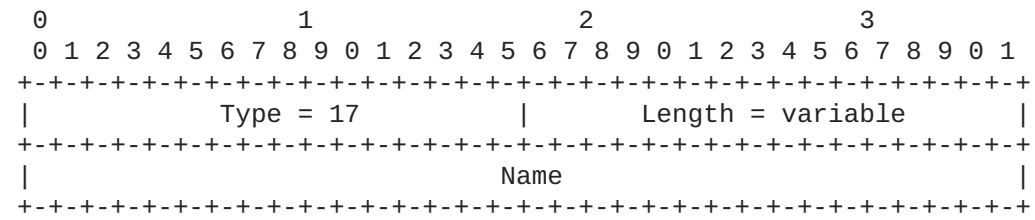


Figure 23

Type = 17 decimal (0x11) (2 bytes) Identifier for the Destination Router Name TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Name (variable length) The destination router name represented as a string.

3.3.13. Peer Path ID

An alphanumeric string which dictates which router path has been chosen for a packet. This name can be the hostname or IP address of the egress interface of the originating router. Any two routers may have multiple pathways between them. This enables association of policies with specific paths.

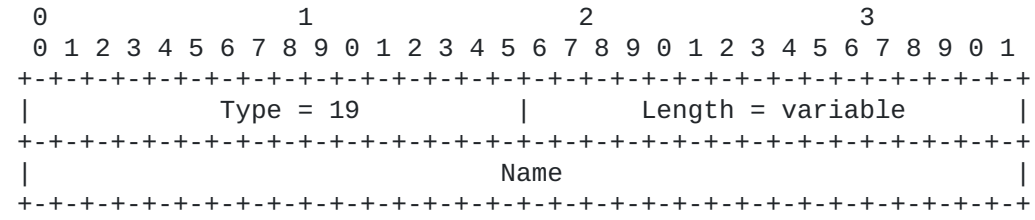


Figure 24

Type = 19 decimal (0x13) (2 bytes) Identifier for the Peer Path ID TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Name (variable length) The peer path name which is represented as a string.

3.3.14. IPv4 Source NAT Address

An alphanumeric string which dictates which router path has been chosen for a packet. This name can be the hostname or IP address of the egress interface of the originating router. Any two routers may have multiple pathways between them. This enables association of policies with specific paths.

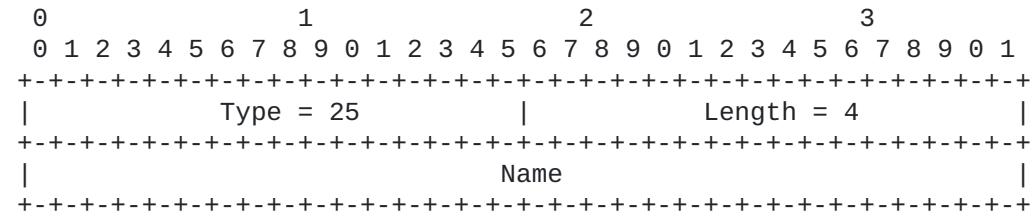


Figure 25

Type = 25 decimal (0x19) (2 bytes) Identifier for the Source NAT Address (IPv4) TLV.

Length (2 bytes) Number of bytes associated with the length of the value (not including the 4 bytes associated with the type and length fields).

Source Address (4 bytes) Source NATed address of the packet.

4. Metadata Exchanges and Use Cases

Metadata can be inserted and used to share network intent between routers. Below are examples for specific use cases. The metadata is not limited to these use cases, these are just illustrative.

4.1. Establishing a Peer Path

Before a client or router can send metadata to a next hop router or server, it must know apriori of the next hop router's existence (and interface IP Address or waypoint). The client or router must also have a routed path to the IP Address (as determined by L2 and L3 routing protocols). The mechanism to obtain the router's address is outside the scope of this document. This could be through configuration, a DNS lookup, a LISP like RLOC query, or through a registry service. Routing protocols can be used to share topology as needed to locate supporting routers, and to obtain and share cryptographic keys.

Client software that does not have access to routing layers can generate packets to test if the default route supports metadata by sending specially constructed packets to the default gateway.

4.1.1. Peer Path Status and Performance

If a router or client learns the IP address of a participating peer, it can then verify the path and performance by using BFD or other quality measurement techniques. Once connectivity is established (at L2, L3, and SVR), the peer path status and performance measurements can be measured. This measured criteria can be used to select which peer path to use. This is outside the scope of this document.

4.1.2. Sharing Path Metrics between routers

Two routers that can send metadata between each other are said to be "peers". The interface addresses they use to talk to each other are called "waypoints". A router can send its "peer" a quality report of what it has measured from its point of view with a peer. This provides a view from the peers point of view that may be useful in deciding algorithmically what peer path to select. To send information to a peer, one can include a Path Metrics Header as defined in Section 3.2.8.

This header can be included in any metadata sent, at any time. The receiving side can use this information as needed. There are no standard timeframes for sending this information, and it is implementation specific.

4.1.3. Testing Path Availability

Testing path availability is outside the scope of this document. BFD is frequently used to verify that a path to a router is available. Lack of a metadata response when sending sessions to a router indicates that path is unavailable as well. Since routers are also running routing protocols, should the waypoint address become unreachable, the path is also declared down.

4.2. New Session Initiation

When a client or router detects that a new session is being established, metadata must be inserted into the first packet to communicate intent to a next hop router. Detecting a new session is protocol specific. For TCP, it's a new 5 Tuple packet (new flow) that is a SYN packet. For UDP, it's simply a new 5 Tuple packet not currently in use. These are easily detected because:

* The packet search key generates a flow miss which indicates it's a new 5 tuple to the router.

* Any packet arriving at the router's interface address that is a flow miss MUST have metadata to be processed.

Because it is a flow miss, the router will forward the packet internally to a branch of code that will look up routes and insert metadata.

4.2.1. First Packet Processing

For illustrative purposes, assume that the packet is an engineer at Juniper Networks attempting to access a Zoom video conference service. The resulting QSN is: QSN://engineer.juniper/zoom.

Locally, the source IPv4 address for the engineer is 10.0.1.1. Engineers are assigned to vlan 10, and DNS resolves the Zoom address to be 3.208.72.109. When the packet arrives at the branch router with the name "Branch 1", the following steps will be performed:

* Determine the Tenant. This is just a text name which describes the routes and policies that are available for a group of source IP addresses. In our example, the "engineer" is based upon VLAN

10, and the tenant will be "engineer" as named by the authority "juniper".

* Using the destination IP address, determine a Service. Some services have a single IP Address, while others have multiple CIDR blocks. The Service name can also be obtained by using DPI (SNI/URL/CN from Certificate) or by proxying DNS resolutions.

The determination of a service is essential. With Zoom by example, the current published list of IP addresses, protocol, and ports include over 230 CIDR block ranges where the most common block is a /25. Describing network intent would be difficult using IP Addresses.

Once the tenant and service are known, the routing intent can be determined. The routing policies are checked, specifically:

* Are engineers allowed to use Zoom.

* Is there a valid L3 route to 3.208.72.109. If no valid route exists, is there another Zoom service address configured?

* Look up security policies for Zoom.

* Look up QoS policies for Zoom. Assign Video Conferencing QoS policy.

* Select a peer to send the session to:

  - The Source Peer/Destination Peer defines a peer path that has been chosen. This establishes the source and destination IP Addresses on the wire.

  - There may be more than one path acceptable, and they are rank ordered for the forwarding plane to use.

* Choose a peer path, and establish fast path rules for subsequent packets:

  - Pick the best peer path for communication. The algorithm for selecting the best peer path could include any criteria or algorithm and is not part of this protocol definition.

  - Allocate source and destination[AM2] ports for the waypoint IP addresses chosen for each peer path. to create a unique 5 tuple session between peers.

  - Install fast path NAT rules for the packet in the forward direction.

  - Install fast path NAT rules for packets that will come in the reverse direction.

Once all of these local L2/L3 parameters are mapped to Tenant/ Service routing intent, then metadata can be constructed an inserted into the first packet (for TCP A SYN Packet).

The metadata attributes that will be inserted includes:

*Header: Security Identifier

*Payload: Forward Context

*Payload: Source Tenant Name

*Payload: Service Name

*Payload: Session UUID

*Payload: Source Router Name

*Payload: Source Router Security Name

*Payload: Peer Path Name

When the first packet is sent, the sending side will include the same exact metadata on every packet until a packet in the opposite direction (reverse direction) arrives with metadata indicating a complete handshake. For TCP, the SYN packet contains metadata, and typically a SYN-ACK from the server side responds with metadata, and there is no further metadata inserted in a session.

```
Client ---->    TCP SYN w/Metadata  ----> Server
Server <---- TCP SYN-ACK w/Metadata <---- Server
```

For UDP, metadata can be inserted in packets until there is a reverse flow packet.

4.2.2. Signing the Metadata

The metadata is signed with an HMAC signature that is a defined number of bytes long. The signature used is defined by RFC2104. The HMAC signature is placed at the very end of the packet, extending the packet length by the number of bytes. The shared keys used for signing and verifying the authenticity of the packet are outside the scope of this document, but could be any well known and trusted key distribution scheme.

4.2.3. Encryption of the Metadata

The metadata Payload Headers are encrypted utilizing AES256, with the initialization vector (IV) supplied directly after the encrypted data to guarantee the receiver can decrypt the information statelessly.

4.2.4. Receiving the First Response Packet

When a next hop router receives a packet directed to an interface the router owns that has a uniquely new 5 tuple, it can reasonably expect the packet (if not a routing protocol packet) contains metadata. By performing DPI on the received packet, the router can determine positively if metadata is present. If so, the metadata can be authenticated, decrypted, parsed for network intent, and removed. The packet is forwarded (possibly through a second SVR). When a first reverse packet is received for the session, the router will

construct a block of metadata in the reverse direction. The purpose of this is:

*Indicate to the sender that it can stop sending metadata.

*Provide backward information about the service for routing of future instances.

The reverse metadata attributes that will be inserted include:

*Header: Security Identifier

*Payload: Reverse Context

*Payload: Tenant Name

*Payload: Destination Router Name

*Payload: Peer Path ID

4.2.5. Subsequent Packet Processing

As soon as the initiating router gets confirmation that the next hop router understands the metadata, it can stop sending metadata. In the case of the TCP transport layer, this occurs with the 2nd packet of the session, first reverse packet, the TCP SYN-ACK response.

4.2.6. Session Termination

No metadata is sent upon normal session termination. The router can monitor the TCP state machine and have a guard timer after seeing a FIN/ACK or RST exchange. After the guard timer, the session can be removed from the system. If a new session arrives during this period (A TCP SYN), then it will cause immediate termination of the existing session. In addition, all protocols also have an associated inactivity timeout, after which the session gets terminated if no packets flow in either direction.

4.2.7. Unicast and Asymmetric Flows

When there are multicast flows, or asymmetry, the sender must assume for TCP when the sequence number is advancing, that there is end-to-end communication, and one can stop sending metadata. For UDP asymmetry the sending router will send a maximum of 11 packets with metadata and if no reverse packets are seen, the receiving peer router will trigger a disable metadata packet to the originating router to turn off metadata for subsequent forward packets.

4.3. Moving a Session

To change the pathway of a session between two routers, one simply reinserts the metadata described in section 3.2.1. but retains the same Session UUID. This will cause the upstream router to do the following:

*Update its Fast Packet forwarding tables to reflect the new IP addresses and ports for transport.

When the sender of the metadata sees packets returning on the new interface, the old fast path entries can be removed.

5. Security Considerations

5.1. HMAC Authentication

HMAC signatures are mandatory for the packets that contain Metadata to guarantee the contents were not changed, and that the router sending it is known to the receiver. Any HMAC algorithm can be used, from SHA128, SHA256 or MD5, as long as both sides agree. HMAC is always performed on the layer 4 payload of the packet.

5.2. Replay Prevention

Optional HMAC signatures can be added to every packet. This prevents any mid-stream attempts to corrupt or impact sessions that are ongoing. The signature must include all of the packet after Layer 4, and include a current time of day to prevent replay attacks.

Both the sending and receiving routers must agree on these optional HMAC signatures, details of which are outside the scope of this document.

5.3. Payload Encryption

Payload encryption can use AES-CBC-128 or AES-CBC-256 ciphers which can be configured. Since these are block-ciphers, the payload should be divisible by 16. If the actual payload length is divisible by 16, then the last 16 bytes will be all 0s. If the actual payload is not divisible by 16, then the remaining data will be padded by 0's and the last byte will indicate the length.

5.4. DDoS and Unexpected Traffic on Waypoint Addresses

Waypoint addresses could be addressed by any client at any time. IP Packets that arrive on the router's interface will be processed with the assumption that they MUST contain metadata OR be part of an existing established routing protocol.

Routers will only accept metadata from routers that they are provisioned to speak with. As such an ACL on incoming source addresses is limited to routers provisioned to communicate. All other packets are dropped.

When a packet is received the "cookie" in the metadata header is reviewed first. If the cookie isn't correct, the packet is dropped.

The HMAC signature is checked. If the signature validates, the packet is assumed good, and processing continues. If the HMAC fails, the packet is dropped.

These methods prevent distributed denial of service attacks on the waypoint addresses of routers.

6. Acknowledgements

The authors would like to thank Anya Yungelson, Scott McCulley, and Patrick MeLampy for their input into this document.

7. Normative References

[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>.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, DOI 10.17487/RFC2234, November 1997, <https://www.rfc-editor.org/info/rfc2234>.

Authors' Addresses

Abilash Menon
Juniper Networks
200 Summit Dr #600
Burlington, MA 01803
United States of America

Email: abilashm@juniper.net

Michael Baj
Juniper Networks
200 Summit Dr #600
Burlington, MA 01803
United States of America

Email: mbaj@juniper.net

Patrick Timmons
Juniper Networks
200 Summit Dr #600
Burlington, MA 01803
United States of America

Email: ptimmons@juniper.net

Hadriel Kaplan
Juniper Networks
200 Summit Dr #600
Burlington, MA 01803
United States of America

Email: hkaplan@juniper.net