VFR and Underlay Fragmentation

Feature history for VFR and underlay fragmentation

Table 1: Feature History

Feature Name	Release Information	Description
VFR and Underlay Fragmentation	Cisco IOS XE Catalyst SD-WAN Release 17.12.1a Cisco Catalyst SD-WAN Manager Release 20.12.1	In Cisco Catalyst SD-WAN networks, the VFR (Virtual Fragmentation Reassembly) actively fragments and reassembles packets. The packets undergo fragmentation to improve transportation efficiency while passing through a VFR-enabled Cisco IOS XE Catalyst SD-WAN device. The VFR reassembles the fragmented packets to match the original incoming packet. The reassembled packet contains critical Layer 4 or Layer 7 information necessary for proper reception by the destination device. Underlay fragmentation refers to the process of breaking down a large data packet into smaller fragments at the network layer. Underlay fragmentation allows the successful transmission of packets that exceed the MTU limitations by breaking them down into manageable fragments and ensuring their reliable delivery.

Virtual Fragmentation Reassembly

A Virtual Fragmentation Reassembly (VFR) is a Cisco feature that

- allows a router to collect IP fragments even if they lack necessary Layer 4 information,
- performs dynamic access control list (ACL) analysis for security, and
- enables security features such as Cisco IOS Firewall and NAT to inspect traffic.

While transmitting data across a network, due to various network constraints, the original data packets fragment into smaller fragments to facilitate seamless transmission. While the packets travel through the Cisco IOS XE Catalyst SD-WAN device, they are fragmented. VFR allows fragmented packets to be reassembled efficiently before reaching their destination.

Packet reassembly modes

In Cisco Catalyst SD-WAN network, data packets undergo reassembly in two modes:

- Default mode: Packets are virtually reassembled by default. Upon the delivery of the first fragment, each feature in the network receives the entire payload of the virtually reassembled packet. When the last fragment is received, the remaining features reassemble the packet. The original packet is fragmented, and the internal fragment information structure is shared. The fragments are then queued for refragmentation based on the fragment-offset sequence. The VFR mechanism reconstructs the packets using information from the fragment headers, such as fragment identifiers, sequence numbers, and offsets.
- Reassembly mode: Packets undergo physical reassembly, and fragment header information isn't saved. Upon receiving the last fragment, the fragments reassemble via a metapacket, and the internal fragment information structure is released.

If the packets were originally fragmented using the default mode, they undergo reassembly as if they were the original incoming packets. On the other hand, when the reassembly mode is utilized to virtually fragment the packets, they experience fragmentation based on the MTU of the egress interface before reassembly.

Some features (such as NAT, Cisco IOS XE Firewall, IPSec) automatically enable VFR to obtain Layer 4 or Layer 7 information.

When a particular interface enables VFR, it overrides the existing firewall or NAT's VFR mode configuration by default, ensuring interoperability with the firewall or NAT.

Underlay fragmentation

An underlay fragmentation is a network layer process that

- breaks down large data packets that exceed the Maximum Transmission Unit (MTU) size supported by the Cisco Catalyst SD-WAN network infrastructure,
- enables the transmission of packets that exceed MTU limitations by fragmenting them into smaller parts, and
- ensures the successful delivery of these fragments across the network.

Benefits of VFR and underlay fragmentation

- VFR enables the Cisco IOS XE Firewall to create appropriate dynamic access control lists (ACLs) to protect the network from various fragmentation attacks.
- VFR is responsible for detecting and preventing various types of fragment attacks.
- VFR drops all fragments within a fragment chain if an overlap of a fragment is detected.

Prerequisites for configuring VFR and underlay fragmentation

Properly configure the Maximum Transmission Unit (MTU) size on all network devices. The MTU defines the maximum packet size that can be transmitted without fragmentation. Ensure the MTU is set appropriately on every device along the network path to avoid unintended underlay fragmentation.

Restrictions for configuring VFR and underlay fragmentation

Fragment handling requirements

- The VFR process requires all fragments within an IP datagram to be present. If load balancing causes fragments to be sent to different devices, VFR may fail and drop fragments.
- If any fragments in a series of fragmented packets are lost or arrive out of order, the reassembly process may fail, resulting in incomplete or corrupted packets.

Feature integration

- VFR is designed to operate with features that require fragment reassembly, such as Cisco Catalyst SD-WAN NAT and IPsec. By default, NAT, Crypto-based IPSec, and NAT64 internally enable or disable VFR on an interface when these features are activated. If multiple features enable VFR on the same interface, VFR uses a reference count to track the number of enabling features. VFR is automatically disabled when the reference count reaches zero.
- The VFR CLIs are unavailable under port-channel sub-interfaces.

Underlay fragmentation

- The underlay fragmentation mechanism operates only at the network layer and is limited to the underlying network infrastructure. It does not manage fragmentation and reassembly across multiple network segments or provide end-to-end fragmentation handling.

Use cases for VFR and underlay fragmentation

Networks such as long-distance connections such as a connection between an airplane and aiport signal towers, can experience interruptions, due to the time it takes for large packets to traverse these links. When VFR is enabled, the fragments reassemble into a complete datagram, and then are fragmented within the Cisco Catalyst SD-WAN tunnel interface. With this, the first fragment is sent out first and there is no interruption in receiving the packets.

Underlay fragmentation helps in fragmenting large packets into smaller sizes, and reconstruct the packet back into the original one. This improves the overall application performance.

Boost mode

The boost mode helps in resolving one of the identified bottlenecks related to the memory management of fragments within the data plane of the network.

The boost mode is disabled by default on Cisco IOS XE Catalyst SD-WAN devices.

Before Cisco IOS XE Catalyst SD-WAN Release 17.12.1a	Cisco IOS XE Catalyst SD-WAN Release 17.12.1a and later
The memory allocation to reassembly of fragments occured from a global chunk, necessitating a lock in period for the memory until the reassembly is complete. This leads to potential competition among multiple threads for the same global chunk and results in waiting for the same memory.	The boost mode enhances performance by utilizing CVLA, an alternative data plane memory infrastructure. Unlike the chunk mechanism, CVLA is lock-free and is an efficient memory management mechanism within Cisco IOS XE devices.

Configure VFR and underlay fragmentation

Use one of these methods to configure VFR and underlay fragmentation:

- Configuration groups
- CLI commands

Configure underlay fragmentation using a configuration group

Before you begin

Minimum supported releases: Cisco IOS XE Catalyst SD-WAN Release 17.15.1a

Procedure

Step 1 From the Cisco SD-WAN Manager, choose **Configuration** > **Configuration Groups**.

Step 2 Click **Transport & Management Profile**.

Step 3 Select the desired transport profile and click **Edit**.

Step 4 Click **Edit Ethernet Interface** > **Tunnel**.

Step 5 Enable **Allow Fragmentation** and **MTU To Max**.

Step 6 Click **Save**.

Configure VFR using CLI commands

Enable virtual fragment reassembly (VFR) on interfaces to prevent fragmentation attacks and ensure correct packet delivery.

You can configure VFR using a CLI template. For more information about using CLI templates, see *CLI Add-On Feature Templates* and *CLI Templates*.

Note By default, CLI templates execute commands in global config mode.

Procedure

Enable VFR.

- Enable VFR for IPv4 packets
- Enable VFR for IPv6 packets

To enable VFR for IPv4 packets on Inbound Interface Traffic perform the following steps:

a) Configure an interface type and enter interface configuration mode.

   interface *interface-type interface-number*

b) Enable VFR on the interface and specify the maximum threshold values.

   ip virtual-reassembly [**max-reassemblies** *number*] [**max-fragments** *number*] [**timeout** *seconds*] [**mode** *modes*][**drop-fragments**]

Example:

Here is the complete configuration example to enable VFR for IPv4 packets:

```
interface GigabitEthernet5
ip virtual-reassembly max-reassemblies 64 max-fragments 16 mode default timeout 5
```

To enable VFR for IPv4 packets on outbound interface traffic perform the following steps:

a) Configure an interface type and enter interface configuration mode.

   interface *interface-type interface-number*

b) Enable VFR for outbound interface traffic on the interface and specify the maximum threshold values.

   ip virtual-reassembly-out [**max-reassemblies** *number*] [**max-fragments** *number*] [**timeout** *seconds*] [**mode** *modes*][**drop-fragments**]

Example:

Here is the complete configuration example to enable VFR for IPv4 packets:

```
interface GigabitEthernet 5
ip virtual-reassembly-out mode default max-fragments 64
```

To enable VFR for IPv6 packets on inbound interface traffic perform the following steps:

a) Configure an interface type and enter interface configuration mode.

   interface *interface-type interface-number*

b) Enable VFR for IPv6 packets on inbound interface traffic.

ipv6 virtual-reassembly [**in**|**out**] [**max-reassemblies** *number*] [**max-fragments** *number*] [**timeout** *seconds*] [**mode** *modes*][**drop-fragments**]

Example:

Here is the complete configuration example to enable VFR for IPv6 packets:

```
interface GigabitEthernet 5
ipv6 virtual-reassembly in mode default max-fragments 25
max-reassemblies 1024
```

To enable VFR for IPv6 packets on outbound interface traffic perform the following steps:

a) Configure an interface type and enter interface configuration mode.

   interface *interface-type* *interface-number*

b) Enable VFR for IPv6 packets on outbound interface traffic.

   ipv6 virtual-reassembly [**in**|**out**] [**max-reassemblies** *number*] [**max-fragments** *number*] [**timeout** *seconds*] [**mode** *modes*][**drop-fragments**]

Example:

Here is the complete configuration example to enable VFR for IPv6 packets:

```
interface GigabitEthernet 5
ipv6 virtual-reassembly out mode default max-fragments 25
```

Configure underlay fragmentation using CLI commands

You can configure underlay fragmentation using CLI templates. For more information about using CLI templates, see *CLI Add-On Feature Templates* and *CLI Templates*.

Note By default, CLI templates execute commands in global config mode.

This section provides example CLI configurations to configure underlay fragmentation.

Procedure

Step 1 Enter the config-sdwan mode

Example:

```
sdwan
```

Step 2 Configure an interface type and enter interface configuration mode.

Example:

```
interface interface-name interface-number
```

Step 3 Configure the tunnel interface.

Example:

```
tunnel-interface
```

Step 4 Skip Layer 3 fragmentation and clear overlay DF bit.

Example:

```
inner-fragmentation-disable
```

Step 5 Perform the encapsulation for the GRE interface of the TLOC.

Example:

```
encapsulation gre
```

Only GRE encapsulation is supported for underlay fragmentation in Cisco IOS XE Catalyst SD-WAN Release 17.12.1a.

Here is the complete configuration example to enable underlay fragmentation:

```
sdwan
interface GigabitEthernet1
tunnel-interface
inner-fragmentation-disable
encapsulation gre
```

Enable boost mode using CLI commands

You can enable boost mode using a CLI template. For more information about using CLI templates, see *CLI Add-On Feature Templates* and *CLI Templates*.

Note By default, CLI templates execute commands in global config mode.

Procedure

Enable the boost mode.

Example:

```
platform ipreass boost-mode
```

Here is the complete configuration example to enable the boost mode:

```
platform ipreass boost-mode
```

Verify VFR and underlay fragments

Boost mode

The following is a sample output of the **show platform hardware qfp active infrastructure cvla client handles** command:

```
Device# show platform hardware qfp active infrastructure cvla client handles
Handles for cpp 0:

---------------------

Entity name: IPREASS_CVLA_0

Handle: 0xeea45000

Number of allocations: 0

Memory allocated: 0


Entity name: FNF_AOR

Handle: 0xeea0d000

Number of allocations: 0

Memory allocated: 0


Entity name: NBAR_CVLA_ENTITY

Handle: 0xee946000

Number of allocations: 0

Memory allocated: 0


Entity name: FNF Chunk 2

Handle: 0xef929000

Number of allocations: 0

Memory allocated: 0


Entity name: FNF Chunk 1

Handle: 0xef928000

Number of allocations: 0

Memory allocated: 0
```

```
---------------------
```

The boost mode is disabled if entity for **IPREASS_CVLA_*** is not displayed. Once the boost mode is disabled, the **IPREASS_CVLA_*** disappears after 64 seconds.

VFR for IPv4 packets

The following is a sample output from the **show ip virtual-reassembly** command:

```
Device# show ip virtual-reassembly GigabitEthernet 5
GigabitEthernet5:

   Virtual Fragment Reassembly (VFR) is ENABLED [out]

   Concurrent reassemblies (max-reassemblies): 16

   Fragments per reassembly (max-fragments): 32

   Reassembly timeout (timeout): 3 seconds

   Drop fragments: OFF


   Current reassembly count:0

   Current fragment count:0

   Total reassembly count:12

   Total reassembly timeout
```

The example shows if VFR for IPv4 is enabled or not. **Virtual Fragment Reassembly (VFR) is ENABLED [out]** signifies that VFR is enabled. The total packets that underwent reassembly are also displayed.

VFR for IPv6 packets

The following is a sample output from the **show ipv6 virtual-reassembly** command:

```
Device# show ipv6 virtual-reassembly GigabitEthernet 5
GigabitEthernet5:

   IPv6 Virtual Fragment Reassembly (IPV6VFR) is ENABLED [out]

   IPv6 configured concurrent reassemblies (max-reassemblies): 64

   IPv6 configured fragments per reassembly (max-fragments): 16

   IPv6 configured reassembly timeout (timeout): 3 seconds

   IPv6 configured drop fragments: OFF


   IPv6 current reassembly count:0
   IPv6 current fragment count:0

   IPv6 total reassembly count:12

   IPv6 total reassembly timeout count:0
```

The example shows if VFR for IPv6 is enabled or not. **Virtual Fragment Reassembly (VFR) is ENABLED [out]** signifies that VFR is enabled. The total packets that underwent reassembly are also displayed.

Underlay fragmentation

The following is a sample output from the **show ip traffic interface GigabitEthernet 1** command:

```
Device# show ip traffic interface GigabitEthernet 1
GigabitEthernet 1 statistics :

  Rcvd:  11048818 total, 749458331 total_bytes

         0 format errors, 0 hop count exceeded

         0 bad header, 0 no route

         0 bad destination, 0 not a router

         0 no protocol, 0 truncated

         0 forwarded

         0 fragments, 0 total reassembled

         0 reassembly timeouts, 0 reassembly failures

         0 discards, 0 delivers

  Sent:  0 total, 0 total_bytes 0 discards

         0 generated, 0 forwarded

         0 fragmented into, 0 fragments, 0 failed

  Mcast: 0 received, 0 received bytes

         0 sent, 0 sent bytes

  Bcast: 0 received, 1256 sent
```

The example shows the number of packets that were sent and received, including the total number of packets. A change from the previous number of packet tranfer indicates that underlay fragmentation is enabled.

The following is a sample output from **show sdwan ftm tloc-list** command:

```
Device# show sdwan ftm tloc-list

--- LOCAL  TLOC LIST ---




Id: 32775 (binosId=0xf808007f), Tenant Id: 0      LocalTLOC, num-nhops: 0   ,hash: 0, ref:
 1    SLA 0x0:0x0 Inner-fragmentation
```

```
-disable: No


[TOTAL-LOCAL-TLOC:1]




--- REMOTE TLOC LIST ---




Id: 32768 (binosId=0xf808000f), Tenant Id: 0       SLAClass, num-nhops: 0   ,hash: 0, ref:
 1    SLA 0x0:0x0
 num-active-nhops: 0



Id: 32774 (binosId=0xf808006f), Tenant Id: 0       SLAClass, num-nhops: 1   ,hash: 0, ref:
 1    SLA 0x1:0x0
 [nhop1] nhop-Id: 19   , Type: IPsec     , Encap: IPSEC SLA 0x1:0x0hw_record_index:   5
198.100.1.5/12366->198.100.1.6/12346 pr
oto 0x800 hash 0x13 wan-if 3 tloc 32774 R-color mpls local-tloc 32775 L-color mpls BFD UP
tloc-capability 0 SLA 0x1:0x0 weight
1   pref 0

 num-active-nhops: 1


[TOTAL-REMOTE-TLOC:2]


--- PENDING TLOC LIST (is_pending_updates:FALSE)---




[TOTAL-PENDING-TLOC:0]




--- UNMATCHED TLOC LIST (is_pending_updates:FALSE)---




[TOTAL-UNMATCHED-TLOC:0]



--- TENANT LOCAL  TLOC LIST ---
```

The example displays all the local TLOCs in the network.

The following is a sample output from **show platform software sdwanR0 next-hop overlay all** command:

```
Device# show platform software sdwan R0 next-hop overlay all

Show sdwan next-hop oce all :


OCE ID: 0xf800013f, OCE Type: SDWAN_NH_OVERLAY

Overlay: client_handle (nil), ppe addr (nil)

  overlay encap: ipsec

  src-ip: 198.100.1.5, src-port: 12366

  dst-ip: 198.100.1.6, dst-port: 12346

  flags: 0x0, linktype: MCP_LINK_IP, ifhandle: 15, encap type: MCP_ET_NULL

  encap rewrite: 00

  mtu: 1446, fixup: 0x0, fixup_flags_2: 0x0, color: mpls, phy_oce_handle: 31, nh_overlay_h:
 0xf800013f
    Overlay_CFG:

    encap type: ipsec

    src-ip: 198.100.1.5, src-port: 12366

    dst-ip: 198.100.1.6, dst-port: 12346

    local_system_ip: 1.1.1.1

    remote_system_ip: 2.2.2.2

    local_color: 2 [mpls], remote_color: 2 [mpls]

    wan_ifindex: 8 [GigabitEthernet2], tun_ifindex: 15 [Tunnel0]

    tun_adj_id: 0, l2_adj_id: 0x1f, tunnel_qos_dpidx: 0x0

    bfd-ld: 20005, ipsec_flow_id: 603979786, session_id: 5

    Inner-fragmentation-disable: yes
```

The example demonstrates whether the inner fragmentation is disabled or enabled in a particular next-hop overlay.

The following is a sample output from **show platform software sdwan F0 next-hop overlay all** command:

```
Device# show platform software sdwan F0 next-hop overlay all

OCE ID: 0xf800013f, OCE Type: SDWAN_NH_OVERLAY

Overlay: client_handle 0x63d321350ba0, ppe addr db910710

  overlay encap: ipsec

  src-ip: 198.100.1.5, src-port: 12366

  dst-ip: 198.100.1.6, dst-port: 12346
  flags: 0x0, linktype: MCP_LINK_SDWAN, ifhandle: 15, encap type: MCP_ET_ARPA

  encap rewrite: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

  mtu: 1446, fixup: 0x0, fixup_flags_2: 0x800000, color: mpls, phy_oce_handle: 31,
nh_overlay_h: 0xf800013f
    Overlay_CFG:

    encap type: ipsec

    src-ip: 198.100.1.5, src-port: 12366

    dst-ip: 198.100.1.6, dst-port: 12346

    local_system_ip: 1.1.1.1

    remote_system_ip: 2.2.2.2

    local_color: 2 [mpls], remote_color: 2 [mpls]

    wan_ifindex: 8 [GigabitEthernet2], tun_ifindex: 15 [Tunnel0]

    tun_adj_id: 0, l2_adj_id: 0x1f, tunnel_qos_dpidx: 0x0

    bfd-ld: 20005, ipsec_flow_id: 603979786, session_id: 5

    Inner-fragmentation-disable: yes
```

The example demonstrates whether the inner fragmentation is disabled or enabled in all the available overlays.