Troubleshoot with the IOS-XE Datapath Packet Trace Feature

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Background Information

Reference Topology

Packet Tracing in Use

- Quick Start Guide
- Enable Platform Conditional Debugs
- Enable Packet Trace
  - Egress Condition Limitation with Packet Traces
- Display the Packet Trace Results
- FIA Trace
- Display the Packet Trace Results
  - Check the FIA Associated with an Interface
- Dump the Traced Packets
- Drop Trace
  - Example Drop Trace Scenario
- Inject and Punt Traces
  - IOSd Drop Tracing
  - IOSd Egress Path Tracing
  - LFTS packet tracing
  - Packet trace pattern matching based on User Defined Filter (ASR1000 platform only)

Packet Trace Examples

- Packet Trace Example - NAT
- Packet Trace Example - VPN

Performance Impact

Introduction

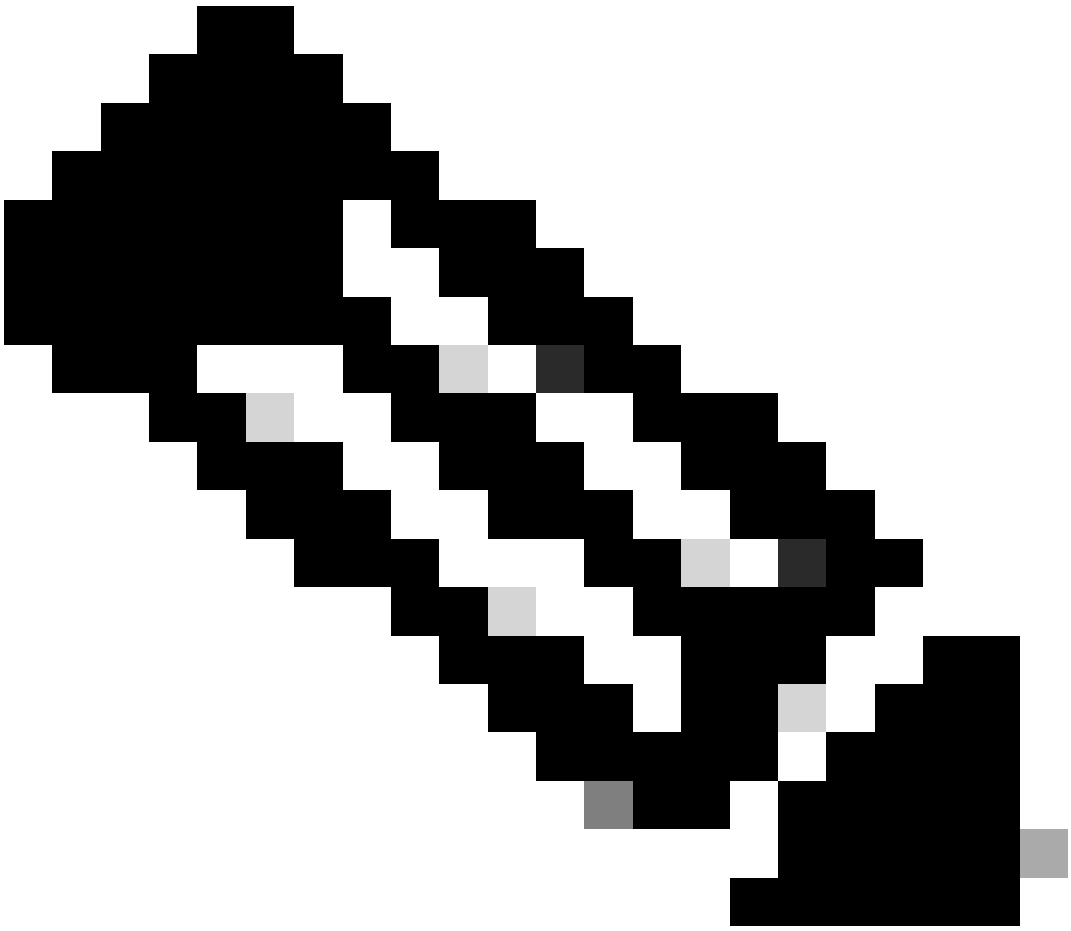
This document describes how to perform datapath packet tracing for Cisco IOS® XE software via the Packet Trace feature.

Prerequisites

Requirements

Cisco recommends that you have knowledge of this information:

The packet-trace feature is available in Cisco IOS-XE version 3.10 and later releases on the QFP (Quantum Flow Processor) based routing platforms, which include the ASR1000, ISR4000, ISR1000, Catalyst 1000, Catalyst 8000, CSR1000v, and Catalyst 8000v series routers. This feature is not supported on the ASR900 series aggregation services routers or the Catalyst series switches that run Cisco IOS-XE software.



> **Note**: The packet-trace feature does not work on the dedicated management interface, GigabitEthernet0 on the ASR1000 series routers, since packets forwarded on that interface are not process by the QFP.

Components Used

The information in this document is based on these software and hardware versions:

- Cisco IOS-XE Software Release 3.10S (15.3(3)S) and later
- ASR1000 series router

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

In order to identify issues such as misconfiguration, capacity overload, or even the ordinary software bug while troubleshooting, it is necessary to understand what happens to a packet within a system. The Cisco IOS-XE Packet Trace feature addresses this need. It provides a field-safe method that is used for accounting and in order to capture the per-packet process details based on a class of user-defined conditions.

Reference Topology

This diagram illustrates the topology that is used for the examples that are described in this document:



Packet Tracing in Use

In order to illustrate the use of the packet trace feature, the example that is used throughout this section describes a trace of the Internet Control Message Protocol (ICMP) traffic from the local workstation 172.16.10.2 (behind the ASR1K) to the remote host 172.16.20.2 in the ingress direction on interface GigabitEthernet0/0/1 on the ASR1K.

You can trace packets on the ASR1K with these two steps:

1. Enable the platform conditional debugs in order to select the packets or traffic that you want to trace on the ASR1K.

2. Enable the platform packet trace with either the path-trace or Feature Invocation Array (FIA) trace option.

Quick Start Guide

Here is a quick start guide if you are already familiar with the contents of this document, and want a section for a quick look at the CLI. These are only a few examples to illustrate the use of the tool. Refer to the later sections that discuss the syntaxes in detail, and ensure you use the configuration that is appropriate to your requirement.

1. Configure platform conditions:

<#root>

```
debug platform condition ipv4 10.0.0.1/32 both
```

```
--> matches in and out packets with source
 or destination as 10.0.0.1/32
```

```
debug platform condition ipv4 access-list 198 egress
```

```
--> (Ensure access-list 198 is
 defined prior to configuring this command) - matches egress packets corresponding
 to access-list 198
```

```
debug platform condition interface gig 0/0/0 ingress

--> matches all ingress packets
 on interface gig 0/0/0
```

```
debug platform condition mpls 10 1 ingress

--> matches MPLS packets with top ingress
 label 10
```

```
debug platform condition ingress

--> matches all ingress packets on all interfaces
 (use cautiously)
```

After a platform condition is configured, start platform conditions with this CLI command:

<#root>

```
debug platform condition start
```

2. Configure packet trace:

<#root>

```
debug platform packet-trace packet 1024

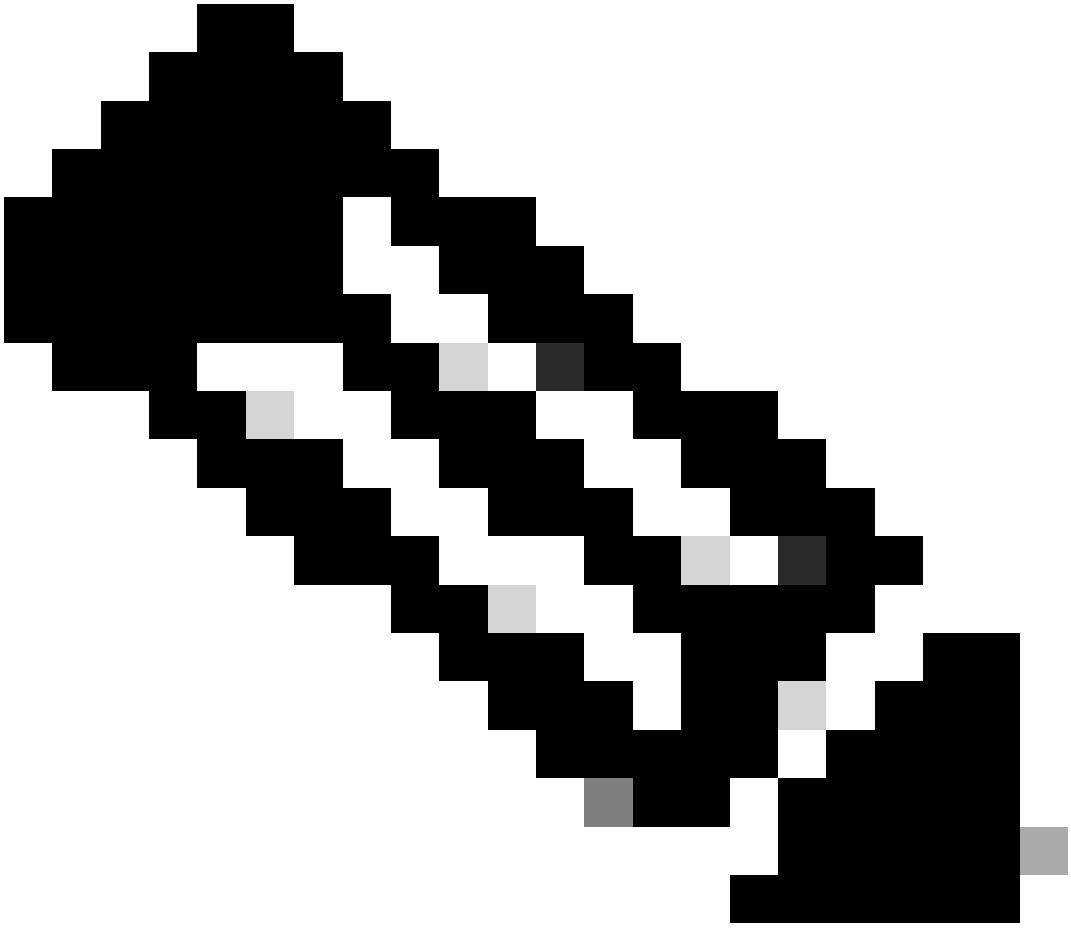
  -> basic path-trace, and automatically stops
 tracing packets after 1024 packets. You can use "circular" option if needed
```

```
debug platform packet-trace packet 1024 fia-trace -

> enables detailed fia trace, stops
 tracing packets after 1024 packets
```

```
debug platform packet-trace drop [code <dropcode>]

  -> if you want to trace/capture only
 packets that are dropped. Refer to Drop Trace section for more details.
```



Note: In earlier Cisco IOS-XE 3.x releases, the command debug platform packet-trace enable is also required to start the packet-trace feature. This is no longer required in Cisco IOS-XE 16.x releases.

Enter this command in order to clear the trace buffer and reset packet-trace:

<#root>

```
clear platform packet-trace statistics
```

```
  --> clear the packet trace buffer
```

The command to clear both platform conditions and the packet trace configuration is:

<#root>

```
clear platform condition all
```

```
  --> clears both platform conditions and the packet trace configuration
```

Show Commands

Verify the platform condition and packet trace configuration after you apply the previous commands in order to ensure you have what you need.

<#root>

show platform conditions

```
 --> shows the platform conditions configured
```

show platform packet-trace configuration

```
 --> shows the packet-trace configurations
```

show debugging

```
 --> this can show both platform conditions and platform packet-trace configured
```

Here are the commands to check the traced/captured packets:

<#root>

show platform packet-trace statistics

```
--> statistics of packets traced
```

show platform packet-trace summary

```
  --> summary of all the packets traced, with input and
 output interfaces, processing result and reason.
```

show platform packet-trace packet 12

```
 -> Display path trace of FIA trace details for the 12th packet in the trace buffer
```

Enable Platform Conditional Debugs

The Packet Trace feature relies on the conditional debug infrastructure in order to determine the packets to be traced. The conditional debug infrastructure provides the ability to filter traffic based on:

- Protocol
- IP address and mask
- Access Control List (ACL)
- Interface
- Traffic direction (ingress or egress)

These conditions define where and when the filters are applied to a packet.

For the traffic that is used in this example, enable platform conditional debugs in the ingress direction for ICMP packets from 172.16.10.2 to 172.16.20.2. In other words, select the traffic that you want to trace. There are various options that you can use in order to select this traffic.

```
<#root>

ASR1000#

debug platform condition

 ?
  egress     Egress only debug
  feature    For a specific feature
  ingress    Ingress only debug
  interface  Set interface for conditional debug
  ipv4       Debug IPv4 conditions
  ipv6       Debug IPv6 conditions
  start      Start conditional debug
  stop       Stop conditional debug
```

In this example, an access-list is used in order to define the condition, as shown here:

```
<#root>

ASR1000#

show access-list 150


Extended IP access list 150
 10 permit icmp host 172.16.10.2 host 172.16.20.2
ASR1000#

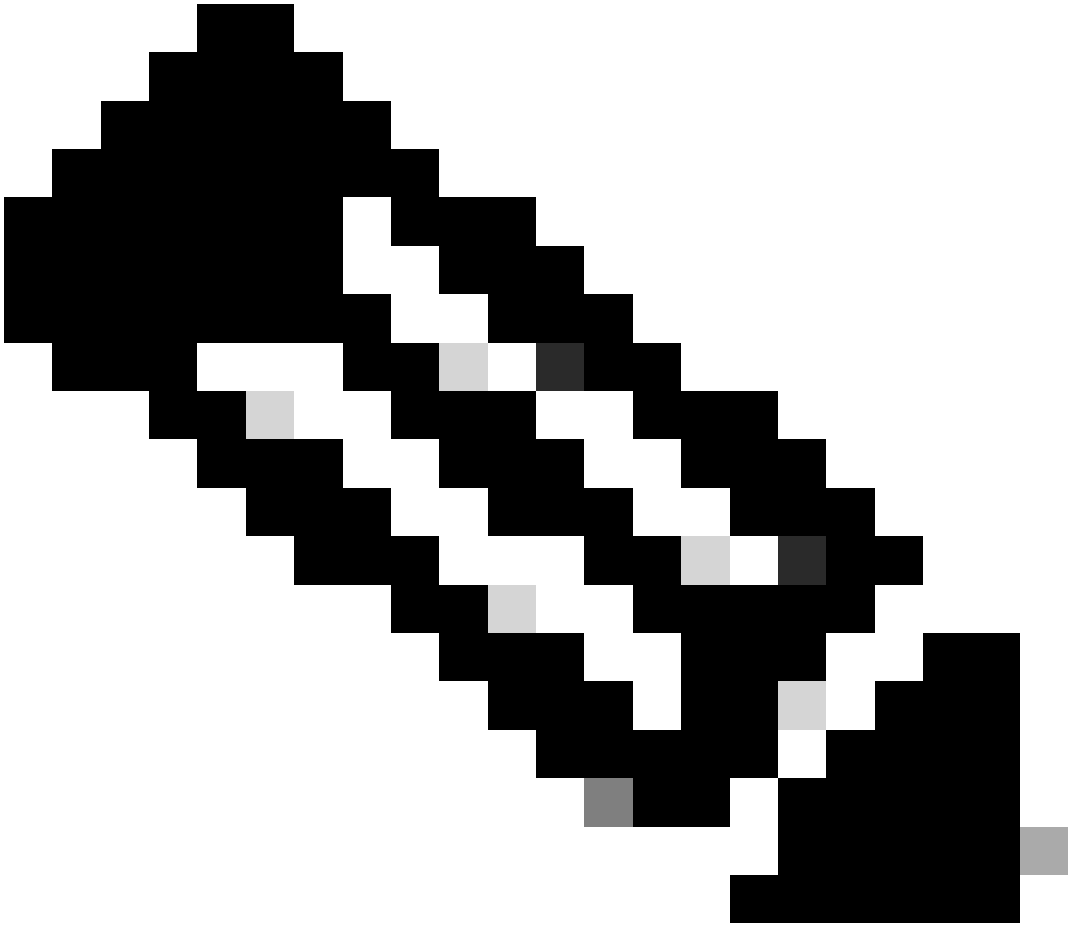
debug platform condition interface gig 0/0/1 ipv4
 access-list 150 ingress
```

In order to begin conditional debugging, enter this command:

```
<#root>

ASR1000#

debug platform condition start
```



Note: In order to stop or disable the conditional debugging infrastructure, enter the debug platform condition stop command.

In order to view the conditional debug filters that are configured, enter this command:

```
<#root>

ASR1000#

show platform conditions


Conditional Debug Global State:

Start

Conditions                                                          Direction
--------------------------------------------------------------------|---------
GigabitEthernet0/0/1                      & IPV4 ACL [150]           ingress
```

```
Feature Condition        Format                  Value
----------------------|-----------------------|--------------------------------

ASR1000#
```

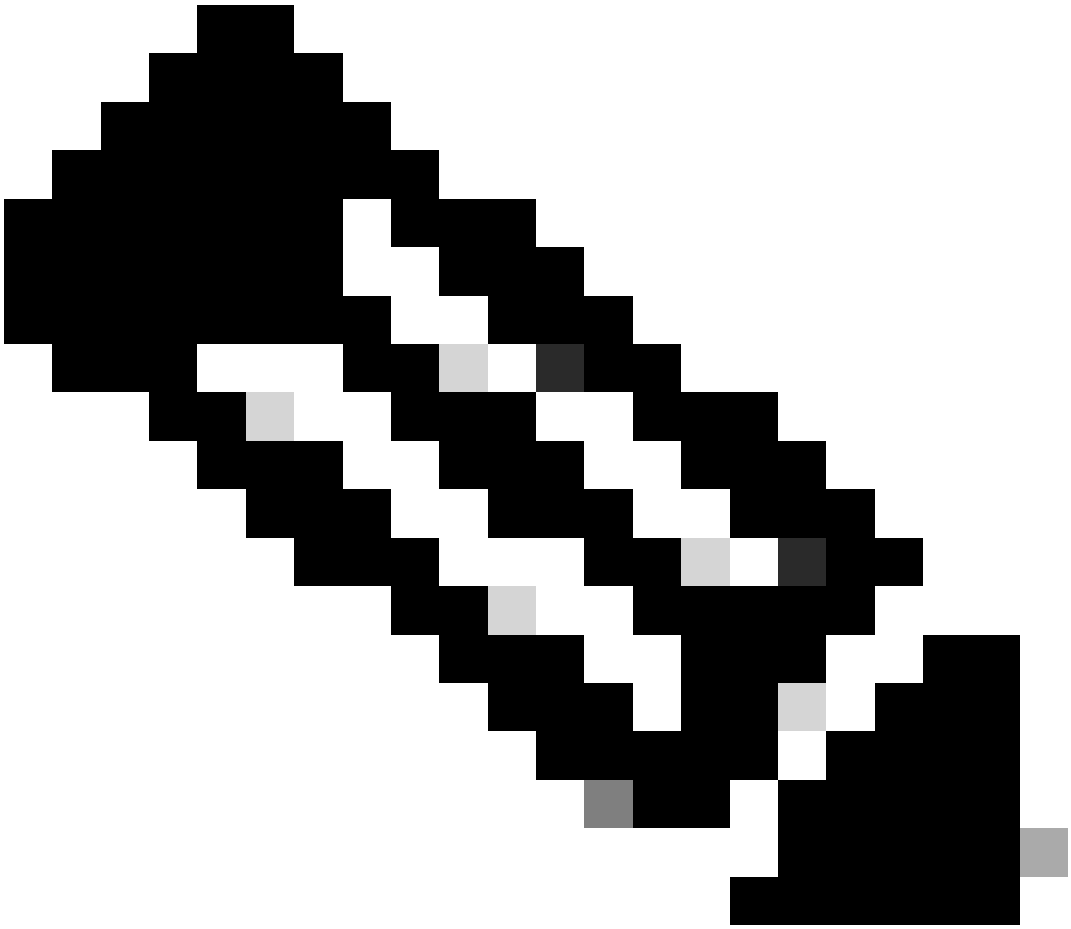
In summary, this configuration has been applied thus far:

<#root>

```
access-list 150 permit icmp host 172.16.10.2 host 172.16.20.2
```

debug platform condition interface gig 0/0/1 ipv4 access-list 150 ingress
debug platform condition start

Enable Packet Trace



Note: This section describes the packet and copy options in detail, and the other options are

described later in the document.

Packet traces are supported on both the physical and the logical interfaces, such as Tunnel or Virtual-access interfaces.

Here is the packet trace CLI syntax:

<#root>

```
ASR1000#
```

debug platform packet-trace

```
 ?
  copy    Copy packet data
  drop    Trace drops only
  inject  Trace injects only
  packet  Packet count
  punt    Trace punts only
```

<#root>

debug platform packet-trace packet <pkt-size/pkt-num> [fia-trace | summary-only]
[circular] [data-size <data-size>]

Here are descriptions for the keywords of this command:

- **pkt-num** - The Packet Number specifies the maximum number of packets that are maintained at one time.
- **summary-only** - This specifies that only the summary data is captured. The default is to capture both summary data and feature-path data.
- **fia-trace** - This optionally performs an FIA trace in addition to the path data information.
- **data-size** - This allows you to specify the size of the path data buffer, from 2,048 to 16,384 bytes. The default is **2,048** bytes.

<#root>

debug platform packet-trace copy packet {in | out | both} [L2 | L3 | L4]
[size <num-bytes>]

Here are descriptions for the keywords of this command:

- **in/out** - This specifies the direction of the packet flow to be copied - ingress and/or egress.
- **L2/L3/L4** - This allows you to specify the location that the copy of the packet starts. Layer 2 (L2) is the default location.

- **size** - This allows you to specify the maximum number of octets that are copied. The default is 64 octets.

For this example, this is the command used in order to enable packet trace for the traffic that is selected with the conditional debug infrastructure:

```
<#root>

ASR1000#

debug platform packet-trace packet 16
```

In order to review the packet trace configuration, enter this command:

```
<#root>

ASR1000#

show platform packet-trace configuration


debug platform packet-trace packet 16 data-size 2048
```

You can also enter the **show debugging** command in order to view both the platform conditional debugs and the packet trace configurations:

```
<#root>

ASR1000#

show debugging


IOSXE Conditional Debug Configs:



Conditional Debug Global State: Start

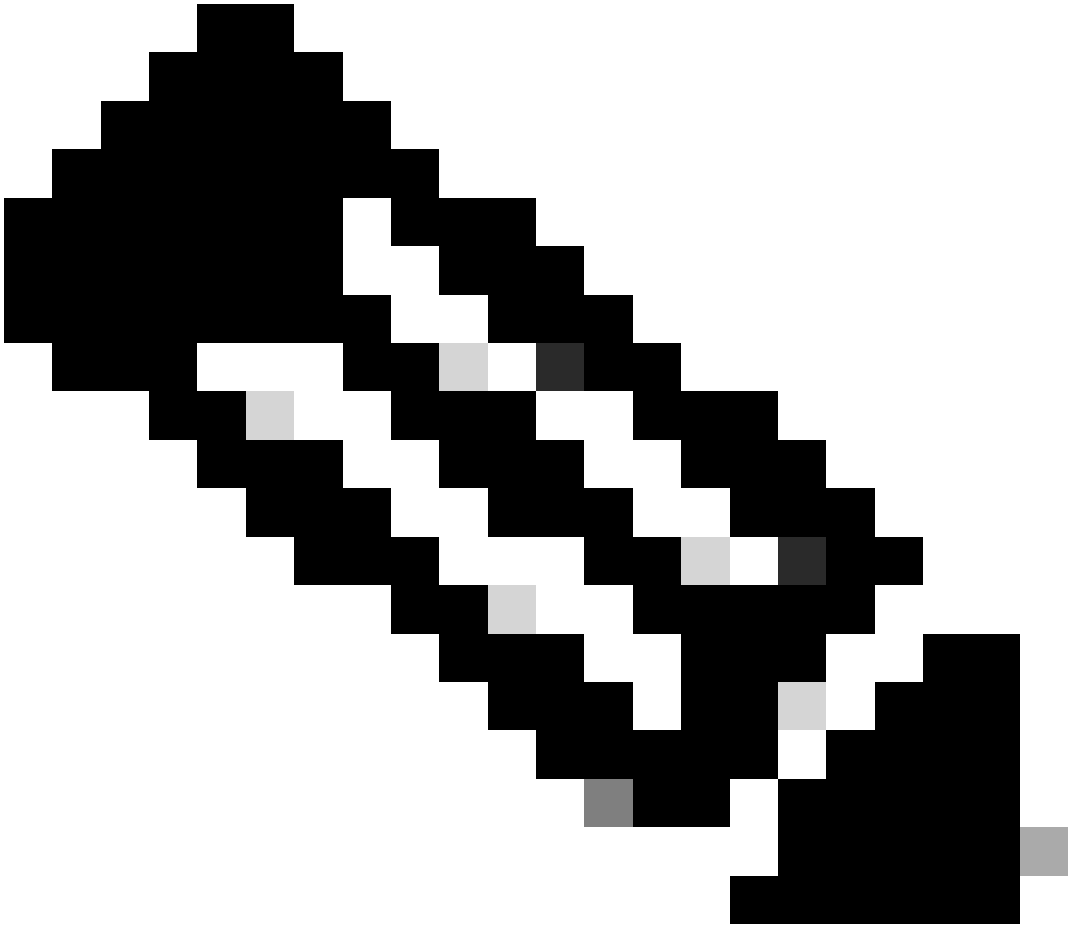
Conditions
                                                                Direction
------------------------------------------------------------------------|---------
GigabitEthernet0/0/1                     & IPV4 ACL [150]                ingress
...
IOSXE Packet Tracing Configs:

Feature Condition        Format                  Value
----------------------|----------------------|--------------------------------

Feature  Type           Submode                                          Level
-------|--------------|-----------------------------------------------|----------


IOSXE Packet Tracing Configs:
```

```
debug platform packet-trace packet 16 data-size 2048
```



> **Note**: Enter the clear platform condition all command in order to clear all of the platform debug conditions and the packet trace configurations and data.

In summary, this configuration data has been used thus far in order to enable packet-trace:

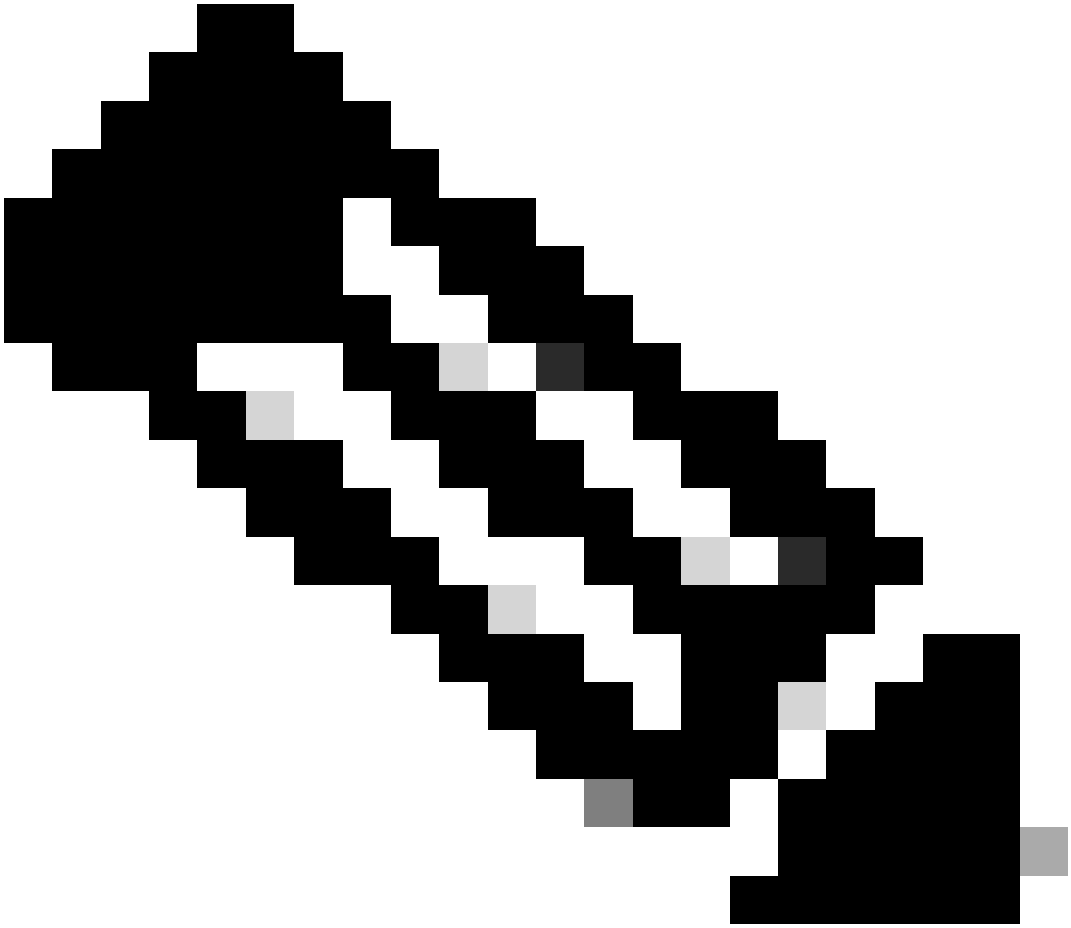
<#root>

debug platform packet-trace packet 16

Egress Condition Limitation with Packet Traces

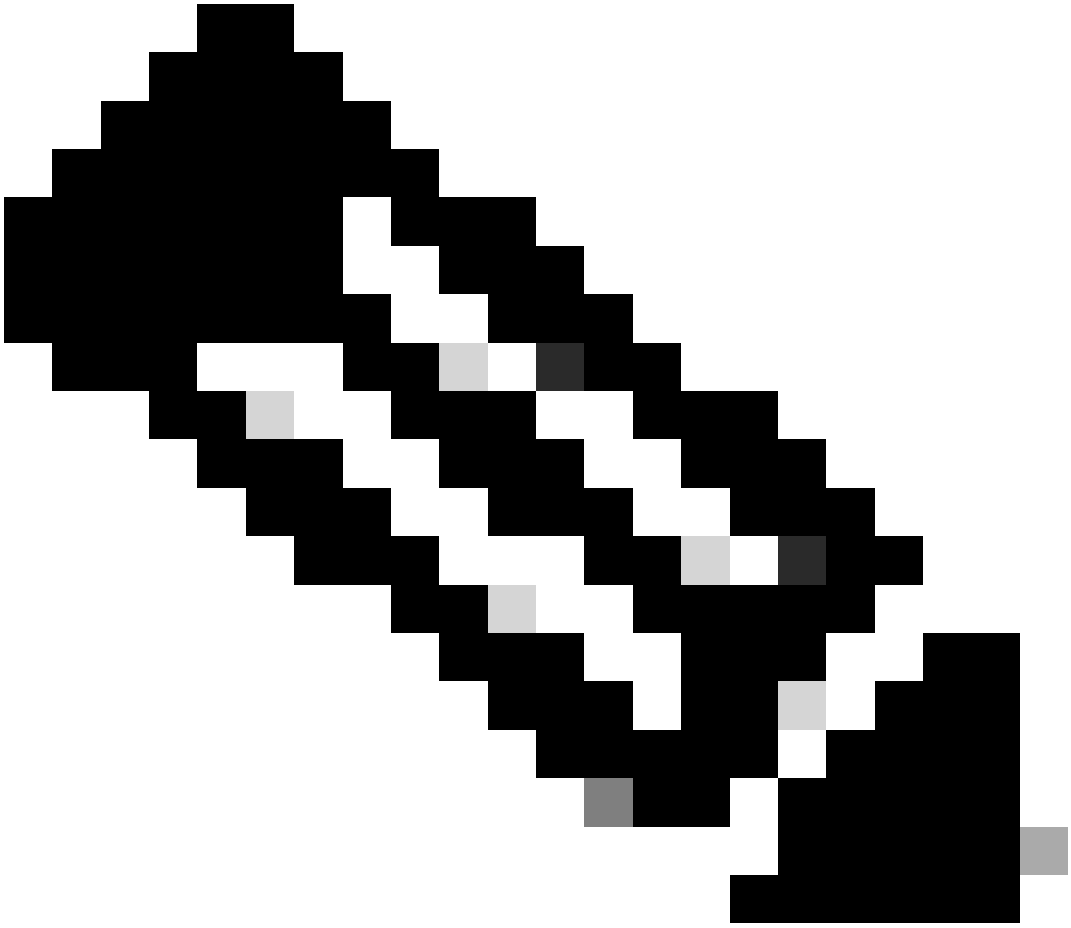
The conditions define the conditional filters and when they are applied to a packet. For example, **debug platform condition interface g0/0/0 egress** means that a packet is identified as a match when it reaches the

output FIA on interface g0/0/0, so any packet processing that takes place from ingress until that point is missed.



> **Note**: Cisco highly recommends that you use ingress conditions for packet traces in order to get the most complete and meaningful data possible. The egress conditions can be used, but be aware of the limitations.

Display the Packet Trace Results



Note: This section assumes that path-trace is enabled.

Three specific levels of inspection are provided by the packet trace:

- Accounting
- Per-packet summary
- Per-packet path data

When five ICMP request packets are sent from 172.16.10.2 to 172.16.20.2, these commands can be used in order to view the packet trace results:

<#root>

ASR1000#

show platform packet-trace statistics

Packets Traced: 5

```
 Ingress  5
 Inject   0
 Forward  5
 Punt     0
 Drop     0
 Consume  0

 ASR1000#
```

show platform packet-trace summary

```
Pkt

   Input            Output           State  Reason

0

     Gi0/0/1          Gi0/0/0          FWD
 1     Gi0/0/1          Gi0/0/0          FWD
 2     Gi0/0/1          Gi0/0/0          FWD
 3     Gi0/0/1          Gi0/0/0          FWD
 4     Gi0/0/1          Gi0/0/0          FWD

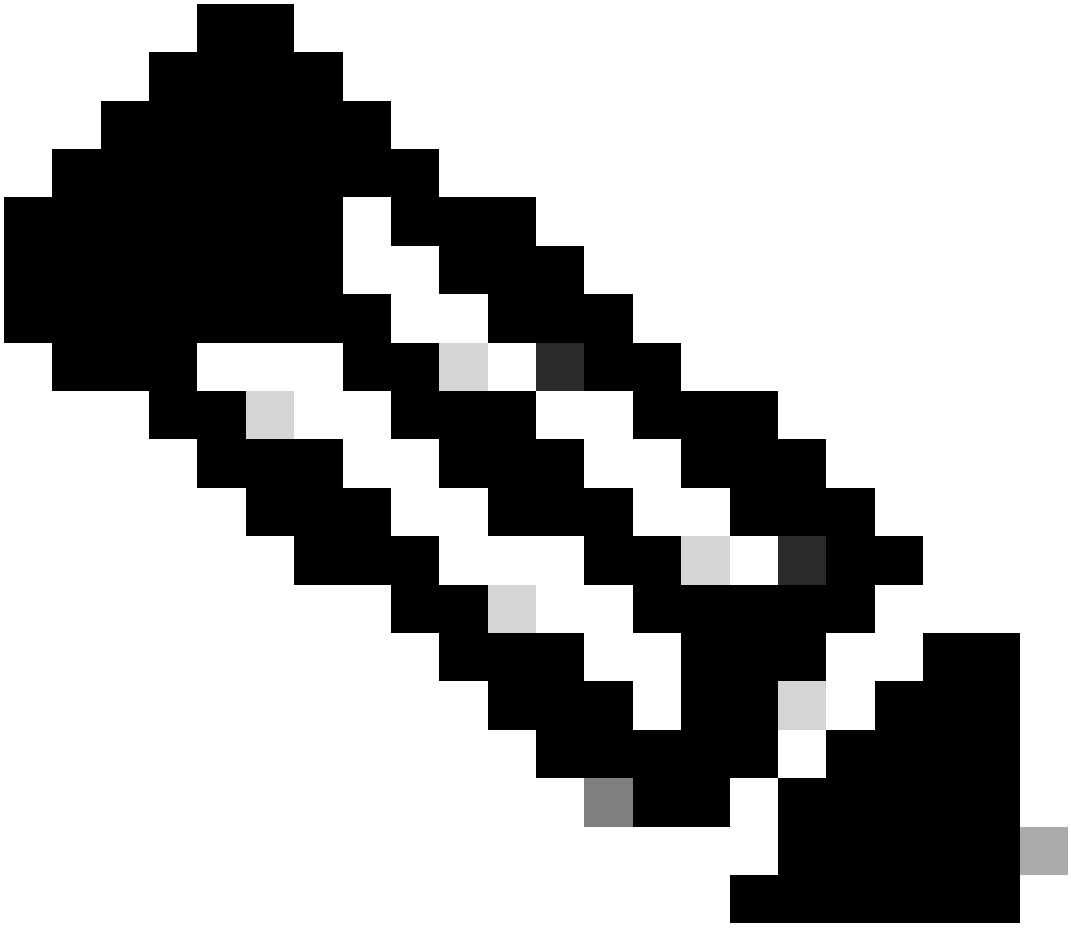
 ASR1000#
```

show platform packet-trace packet 0

```
 Packet: 0

          CBUG ID: 4
 Summary
 Input     : GigabitEthernet0/0/1
 Output    : GigabitEthernet0/0/0
 State     : FWD
 Timestamp
    Start   : 1819281992118 ns (05/17/2014 06:42:01.207240 UTC)
    Stop    : 1819282095121 ns (05/17/2014 06:42:01.207343 UTC)
 Path Trace
 Feature: IPV4
 Source      : 172.16.10.2
 Destination : 172.16.20.2
 Protocol    : 1 (ICMP)

 ASR1000#
```



Note: The third command provides an example that illustrates how to view the packet trace for each packet. In this example, the first packet traced is shown.

From these outputs, you can see that five packets are traced and that you can view the input interface, the output interface, the state, and the path trace.

State	Remark
FWD	The packet is scheduled/queued for delivery, to be forwarded to next hop via an egress interface.
PUNT	The packet is punted from the Forwarding Processor (FP) to the Route Processor (RP) (control plane).
DROP	The packet is dropped on the FP. Run FIA trace, use global drop counters, or use datapath debugs in order to find more details for drop reasons.
CONS	The packet is consumed during a packet process, such as during the ICMP ping request or the crypto packets.

The **ingress** and **inject** counters in the packet trace statistics output correspond to the packets that enter via an external interface and packets that are seen as injected from the control plane, respectively.

FIA Trace

The FIA holds the list of features that are executed sequentially by the Packet Processor Engines (PPE) in the Quantum Flow Processor (QFP) when a packet is forwarded either ingress or egress. The features are based on the configuration data that is applied on the machine. Thus, a FIA trace helps to understand the flow of the packet through the system as the packet is processed.

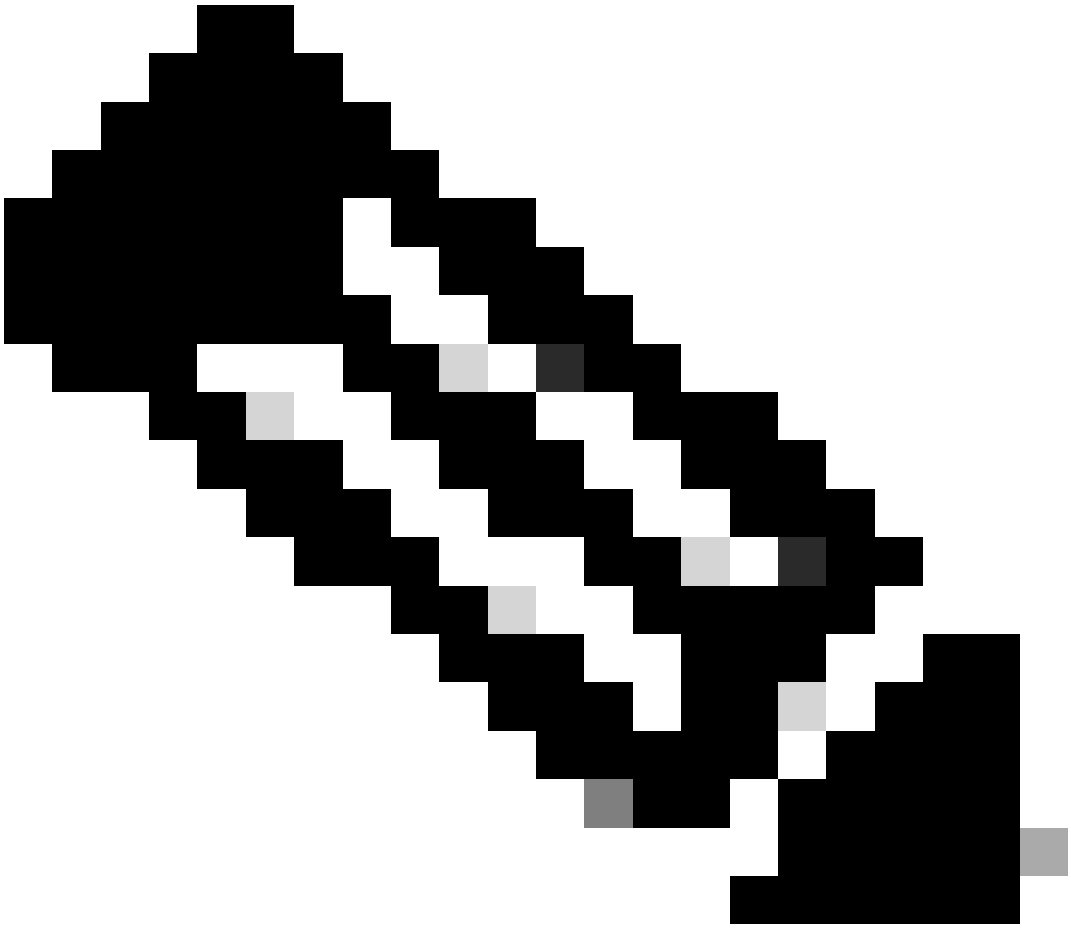
You must apply this configuration data in order to enable packet trace with FIA:

<#root>

ASR1000#

debug platform packet-trace packet 16 fia-trace

Display the Packet Trace Results



Note: This section assumes that FIA trace is enabled. Also, when you add or modify the current

packet trace commands, the buffered packet trace details are cleared, so you must send some traffic again so that you can trace it.

Send five ICMP packets from 172.16.10.2 to 172.16.20.2 after you enter the command that is used in order to enable the FIA trace, as described in the previous section.

<#root>

ASR1000#

show platform packet-trace summary

```
Pkt   Input           Output          State  Reason
0     Gi0/0/1         Gi0/0/0         FWD
1     Gi0/0/1         Gi0/0/0         FWD
2     Gi0/0/1         Gi0/0/0         FWD
3     Gi0/0/1         Gi0/0/0         FWD
4     Gi0/0/1         Gi0/0/0         FWD
```

ASR1000#

show platform packet-trace packet 0

```
Packet: 0           CBUG ID: 9
Summary
  Input     : GigabitEthernet0/0/1
  Output    : GigabitEthernet0/0/0
  State     : FWD
  Timestamp
    Start   : 1819281992118 ns (05/17/2014 06:42:01.207240 UTC)
    Stop    : 1819282095121 ns (05/17/2014 06:42:01.207343 UTC)
Path Trace
  Feature: IPV4
    Source      : 172.16.10.2
    Destination : 172.16.20.2
    Protocol    : 1 (ICMP)
  Feature: FIA_TRACE
    Entry     : 0x8059dbe8 - DEBUG_COND_INPUT_PKT
    Timestamp : 3685243309297
  Feature: FIA_TRACE
    Entry     : 0x82011a00 - IPV4_INPUT_DST_LOOKUP_CONSUME
    Timestamp : 3685243311450
  Feature: FIA_TRACE
    Entry     : 0x82000170 - IPV4_INPUT_FOR_US_MARTIAN
    Timestamp : 3685243312427
  Feature: FIA_TRACE
    Entry     : 0x82004b68 - IPV4_OUTPUT_LOOKUP_PROCESS
    Timestamp : 3685243313230
  Feature: FIA_TRACE
    Entry     : 0x8034f210 - IPV4_INPUT_IPOPTIONS_PROCESS
    Timestamp : 3685243315033
  Feature: FIA_TRACE
    Entry     : 0x82013200 - IPV4_OUTPUT_GOTO_OUTPUT_FEATURE
    Timestamp : 3685243315787
  Feature: FIA_TRACE
    Entry     : 0x80321450 - IPV4_VFR_REFRAG
    Timestamp : 3685243316980
```

```
  Feature: FIA_TRACE
    Entry     : 0x82014700 - IPV6_INPUT_L2_REWRITE
    Timestamp : 3685243317713
  Feature: FIA_TRACE
    Entry     : 0x82000080 - IPV4_OUTPUT_FRAG
    Timestamp : 3685243319223
  Feature: FIA_TRACE
    Entry     : 0x8200e500 - IPV4_OUTPUT_DROP_POLICY
    Timestamp : 3685243319950
  Feature: FIA_TRACE
    Entry     : 0x8059aff4 - PACTRAC_OUTPUT_STATS
    Timestamp : 3685243323603
  Feature: FIA_TRACE
    Entry     : 0x82016100 - MARMOT_SPA_D_TRANSMIT_PKT
    Timestamp : 3685243326183

ASR1000#
```

Check the FIA Associated with an Interface

When you enable the platform conditional debugs, conditional debugging is added to the FIA as a feature. Based on the feature order of processing on the interface, the conditional filter needs to be set accordingly, for example, whether the pre- or post-NAT address must be used in the conditional filter.

This output shows the order of the features in the FIA for the platform conditional debugging that is enabled in the ingress direction:

<#root>

```
ASR1000#
```

show platform hardware qfp active interface if-name GigabitEthernet 0/0/1

```
 General interface information
 Interface Name: GigabitEthernet0/0/1
 Interface state: VALID
 Platform interface handle: 10
 QFP interface handle: 8
 Rx uidb: 1021
 Tx uidb: 131064
 Channel: 16
 Interface Relationships

 BGPPA/QPPB interface configuration information
 Ingress: BGPPA/QPPB not configured. flags: 0000
 Egress : BGPPA not configured. flags: 0000

 ipv4_input enabled.
 ipv4_output enabled.
 layer2_input enabled.
 layer2_output enabled.
 ess_ac_input enabled.

 Features Bound to Interface:
 2 GIC FIA state
 48 PUNT INJECT DB
```

39 SPA/Marmot server
40 ethernet
1 IFM
31 icmp_svr
33 ipfrag_svr
34 ipreass_svr
36 ipvfr_svr
37 ipv6vfr_svr
12 CPP IPSEC
Protocol 0 - ipv4_input
FIA handle - CP:0x108d99cc DP:0x8070f400
IPV4_INPUT_DST_LOOKUP_ISSUE (M)
IPV4_INPUT_ARL_SANITY (M)

CBUG_INPUT_FIA

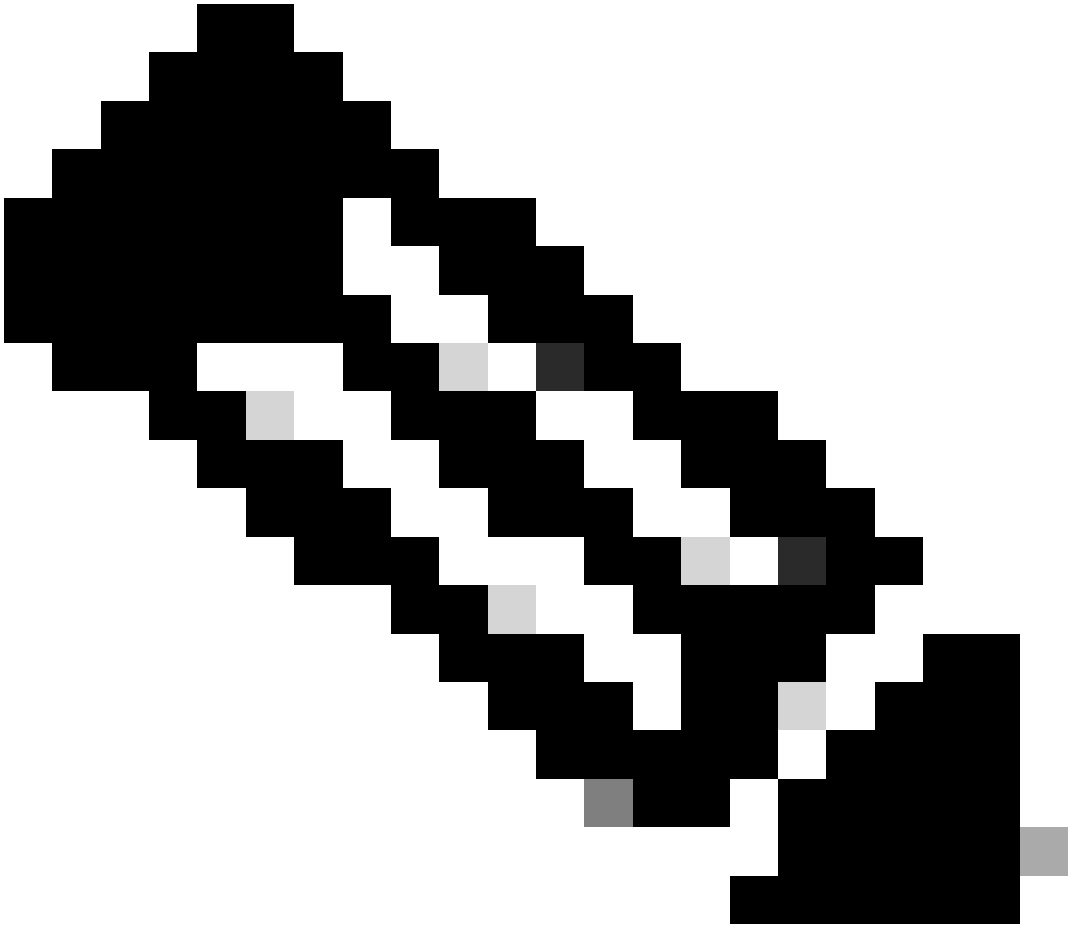
DEBUG_COND_INPUT_PKT

IPV4_INPUT_DST_LOOKUP_CONSUME (M)
IPV4_INPUT_FOR_US_MARTIAN (M)
IPV4_INPUT_IPSEC_CLASSIFY
IPV4_INPUT_IPSEC_COPROC_PROCESS
IPV4_INPUT_IPSEC_RERUN_JUMP
IPV4_INPUT_LOOKUP_PROCESS (M)
IPV4_INPUT_IPOPTIONS_PROCESS (M)
IPV4_INPUT_GOTO_OUTPUT_FEATURE (M)
Protocol 1 - ipv4_output
FIA handle - CP:0x108d9a34 DP:0x8070eb00
IPV4_OUTPUT_VFR
MC_OUTPUT_GEN_RECYCLE (D)
IPV4_VFR_REFRAG (M)
IPV4_OUTPUT_IPSEC_CLASSIFY
IPV4_OUTPUT_IPSEC_COPROC_PROCESS
IPV4_OUTPUT_IPSEC_RERUN_JUMP
IPV4_OUTPUT_L2_REWRITE (M)
IPV4_OUTPUT_FRAG (M)
IPV4_OUTPUT_DROP_POLICY (M)
PACTRAC_OUTPUT_STATS
MARMOT_SPA_D_TRANSMIT_PKT
DEF_IF_DROP_FIA (M)
Protocol 8 - layer2_input
FIA handle - CP:0x108d9bd4 DP:0x8070c700
LAYER2_INPUT_SIA (M)
CBUG_INPUT_FIA
DEBUG_COND_INPUT_PKT
LAYER2_INPUT_LOOKUP_PROCESS (M)
LAYER2_INPUT_GOTO_OUTPUT_FEATURE (M)
Protocol 9 - layer2_output
FIA handle - CP:0x108d9658 DP:0x80714080
LAYER2_OUTPUT_SERVICEWIRE (M)
LAYER2_OUTPUT_DROP_POLICY (M)
PACTRAC_OUTPUT_STATS
MARMOT_SPA_D_TRANSMIT_PKT
DEF_IF_DROP_FIA (M)
Protocol 14 - ess_ac_input
FIA handle - CP:0x108d9ba0 DP:0x8070cb80
PPPOE_GET_SESSION
ESS_ENTER_SWITCHING

```
PPPOE_HANDLE_UNCLASSIFIED_SESSION
DEF_IF_DROP_FIA (M)

QfpEth Physical Information
DPS Addr: 0x11215eb8
Submap Table Addr: 0x00000000
VLAN Ethertype: 0x8100
QOS Mode: Per Link

ASR1000#
```



Note: The CBUG_INPUT_FIA and the DEBUG_COND_INPUT_PKT correspond to the conditional debug features that are configured on the router.

Dump the Traced Packets

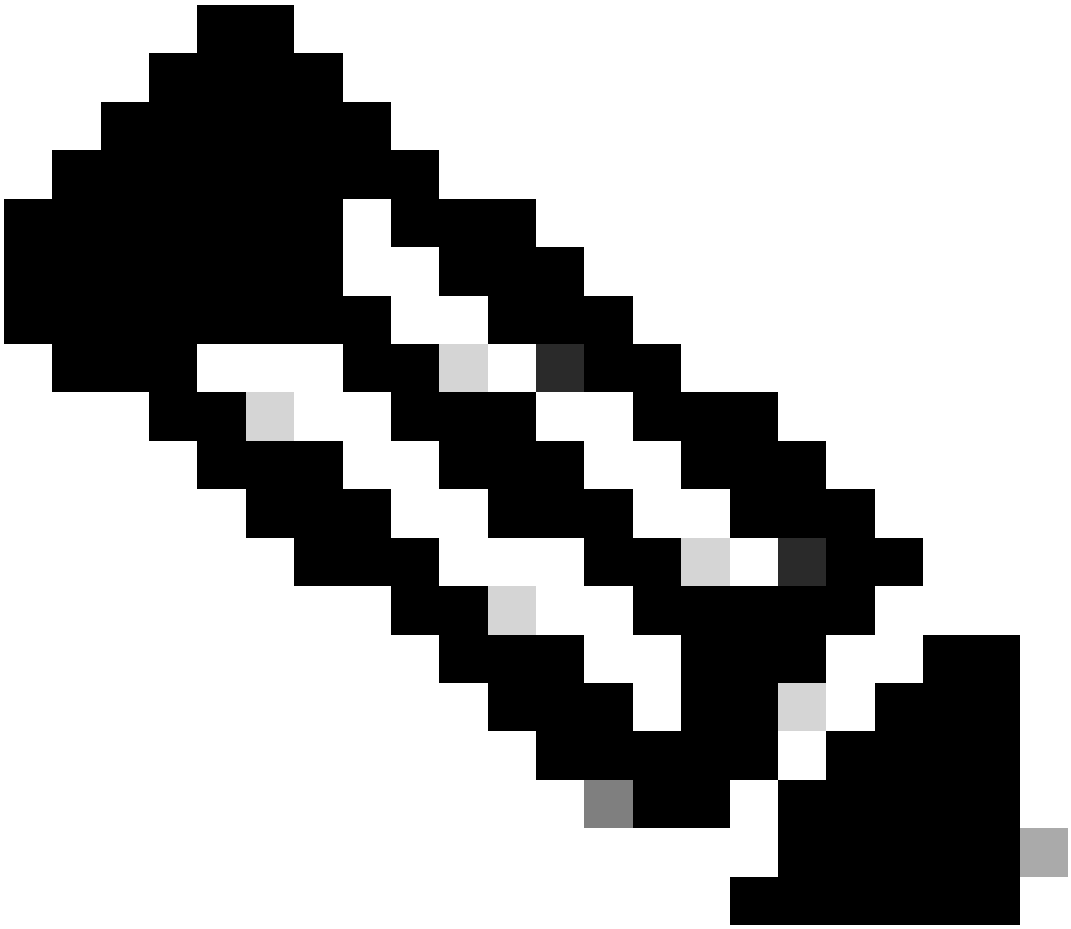
You can copy and dump the packets as they are traced, as this section describes. This example shows how to copy a maximum of 2,048 bytes of the packets in the ingress direction (172.16.10.2 to 172.16.20.2).

Here is the additional command that is needed:

<#root>

ASR1000#

debug platform packet-trace copy packet input size 2048



Note: The size of the packet that is copied is in the range of 16 to 2,048 bytes.

Enter this command in order to dump the copied packets:

<#root>

ASR1000#

show platform packet-trace packet 0

```
Packet: 0           CBUG ID: 14
Summary
Input     : GigabitEthernet0/0/1
Output    : GigabitEthernet0/0/0
State     : FWD
Timestamp
    Start   : 1819281992118 ns (05/17/2014 06:40:01.207240 UTC)
    Stop    : 1819282095121 ns (05/17/2014 06:40:01.207343 UTC)
Path Trace
Feature: IPV4
Source      : 172.16.10.2
Destination : 172.16.20.2
Protocol    : 1 (ICMP)
Feature: FIA_TRACE
Entry     : 0x8059dbe8 - DEBUG_COND_INPUT_PKT
Timestamp : 4458180580929

<some content excluded>

Feature: FIA_TRACE
Entry     : 0x82016100 - MARMOT_SPA_D_TRANSMIT_PKT
Timestamp : 4458180593896


Packet Copy In


a4934c8e 33020023 33231379 08004500 00640160 0000ff01 5f16ac10 0201ac10
01010800 1fd40024 00000000 000184d0 d980abcd abcdabcd abcdabcd abcdabcd
abcdabcd abcdabcd abcdabcd abcdabcd abcdabcd abcdabcd abcdabcd abcdabcd
abcdabcd abcdabcd abcdabcd abcdabcd abcd

ASR1000#
```

Drop Trace

Drop trace is available in Cisco IOS-XE Software Release 3.11 and later. It enables packet trace only for dropped packets. Here are some highlights of the feature:

- It optionally allows you to specify the retention of packets for a specific drop code.
- It can be used without global or interface conditions in order to capture drop events.
- A drop event capture means that only the drop itself is traced, not the life of the packet. However, it still allows you to capture summary data, tuple data, and the packet in order to help refine conditions or provide clues to the next debug step.

Here is the command syntax that is used in order to enable drop-type packet traces:

<#root>

debug platform packet-trace drop [code <code-num>]

The drop code is the same as the drop ID, as reported in the **show platform hardware qfp active statistics drop detail** command output:

<#root>

```
ASR1000#
show platform hardware qfp active statistics drop detail

-------------------------------------------------------------------------------
   ID
  Global Drop Stats                          Packets                  Octets
-------------------------------------------------------------------------------
   60
  IpTtlExceeded                                3                     126
    8
  Ipv4Acl                                     32                    3432
```

Example Drop Trace Scenario

Apply this ACL on the Gig 0/0/0 interface of the ASR1K in order to drop traffic from 172.16.10.2 to 172.16.20.2:

```
access-list 199 deny ip host 172.16.10.2 host 172.16.20.2
access-list 199 permit ip any any
interface Gig 0/0/0
  ip access-group 199 out
```

With the ACL in place, which drops the traffic from the local host to the remote host, apply this drop-trace configuration:

<#root>

```
debug platform condition interface Gig 0/0/1 ingress


debug platform condition start


debug platform packet-trace packet 1024 fia-trace


debug platform packet-trace drop
```

Send five ICMP request packets from 172.16.10.2 to 172.16.20.2. The drop trace captures these packets that are dropped by the ACL, as shown:

<#root>

```
ASR1000#
show platform packet-trace statistics


 Packets Summary
 Matched  5
 Traced   5
 Packets Received
 Ingress  5
 Inject   0
 Packets Processed
 Forward  0
 Punt     0


Drop     5
 Count       Code  Cause
 5           8     Ipv4Acl


 Consume  0


 ASR1000#
show platform packet-trace summary


 Pkt   Input           Output          State  Reason
 0     Gi0/0/1         Gi0/0/0         DROP   8   (Ipv4Acl)
 1     Gi0/0/1         Gi0/0/0         DROP   8   (Ipv4Acl)
 2     Gi0/0/1         Gi0/0/0         DROP   8   (Ipv4Acl)
 3     Gi0/0/1         Gi0/0/0         DROP   8   (Ipv4Acl)
 4     Gi0/0/1         Gi0/0/0         DROP   8   (Ipv4Acl)


 ASR1K#
debug platform condition stop



 ASR1K#
show platform packet-trace packet 0


 Packet: 0           CBUG ID: 140
 Summary
 Input     : GigabitEthernet0/0/1
 Output    : GigabitEthernet0/0/0


State     : DROP 8   (Ipv4Acl)

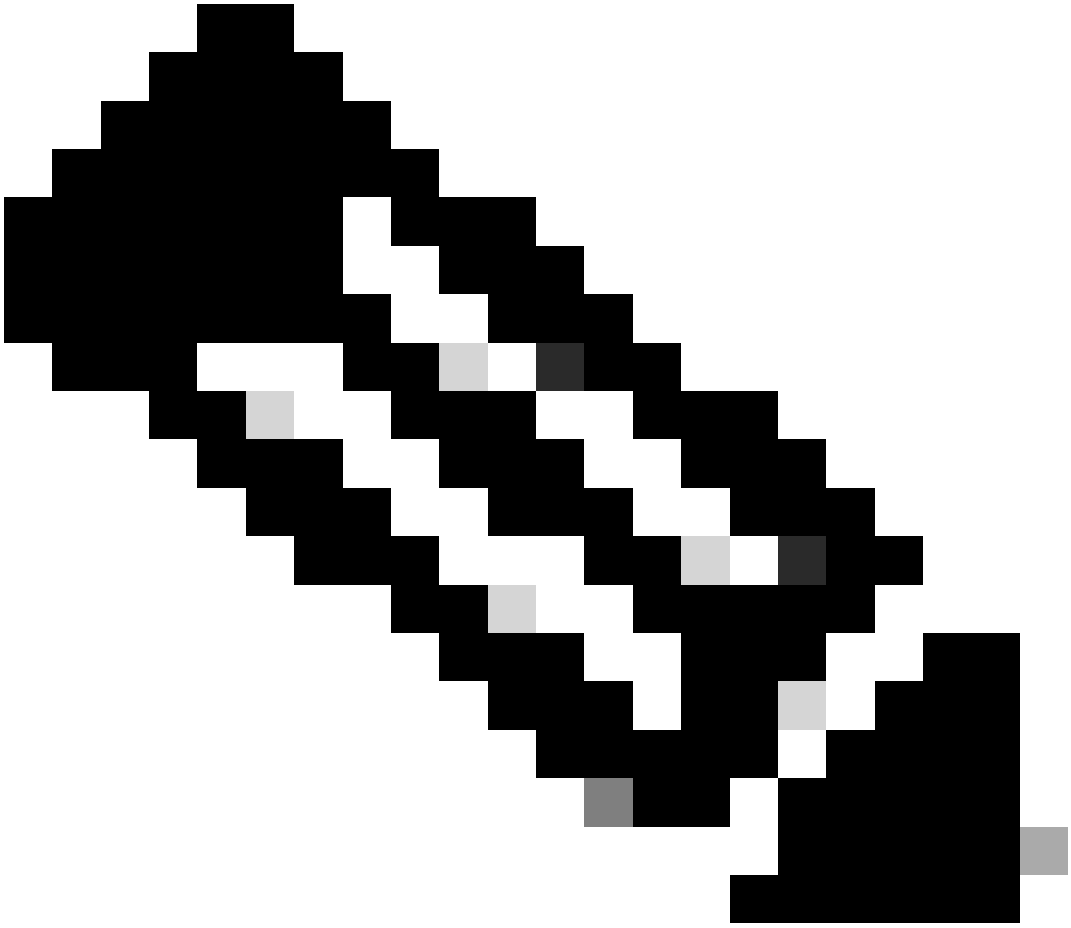

 Timestamp
 Start   : 1819281992118 ns (05/17/2014 06:42:01.207240 UTC)
 Stop    : 1819282095121 ns (05/17/2014 06:42:01.207343 UTC)
 Path Trace
 Feature: IPV4
 Source      : 172.16.10.2
 Destination : 172.16.20.2
 Protocol    : 1 (ICMP)
```

```
Feature: FIA_TRACE
Entry      : 0x806c7eac - DEBUG_COND_INPUT_PKT
Lapsed time: 1031 ns
Feature: FIA_TRACE
Entry      : 0x82011c00 - IPV4_INPUT_DST_LOOKUP_CONSUME
Lapsed time: 657 ns
Feature: FIA_TRACE
Entry      : 0x806a2698 - IPV4_INPUT_ACL
Lapsed time: 2773 ns
Feature: FIA_TRACE
Entry      : 0x82000170 - IPV4_INPUT_FOR_US_MARTIAN
Lapsed time: 1013 ns
Feature: FIA_TRACE
Entry      : 0x82004500 - IPV4_OUTPUT_LOOKUP_PROCESS
Lapsed time: 2951 ns
Feature: FIA_TRACE
Entry      : 0x8041771c - IPV4_INPUT_IPOPTIONS_PROCESS
Lapsed time: 373 ns
Feature: FIA_TRACE
Entry      : 0x82013400 - MPLS_INPUT_GOTO_OUTPUT_FEATURE
Lapsed time: 2097 ns
Feature: FIA_TRACE
Entry      : 0x803c60b8 - IPV4_MC_OUTPUT_VFR_REFRAG
Lapsed time: 373 ns
Feature: FIA_TRACE
Entry      : 0x806db148 - OUTPUT_DROP
Lapsed time: 1297 ns
Feature: FIA_TRACE
Entry      : 0x806a0c98 - IPV4_OUTPUT_ACL
Lapsed time: 78382 ns

ASR1000#
```

Inject and Punt Traces

The inject and punt packet trace feature was added in Cisco IOS-XE Software Release 3.12 and later in order to trace punt (packets that are received on the FP that are punted to the control plane) and inject (packets that are injected to the FP from the control plane) packets.



Note: The punt trace can work without the global or interface conditions, just like a drop trace. However, the conditions must be defined for an inject trace to work.

Here is an example of a punt and **i**nject packet trace when you ping from the ASR1K to an adjacent router:

<#root>

ASR1000#

debug platform condition ipv4 172.16.10.2/32 both

ASR1000#

debug platform condition start

ASR1000#

debug platform packet-trace punt

ASR1000#

```
debug platform packet-trace inject


 ASR1000#
debug platform packet-trace packet 16


 ASR1000#
 ASR1000#ping 172.16.10.2
 Type escape sequence to abort.
 Sending 5, 100-byte ICMP Echos to 172.16.10.2, timeout is 2 seconds:
 !!!!!
 Success rate is 100 percent (5/5), round-trip min/avg/max = 14/14/15 ms
 ASR1000#
```

Now you can verify the punt and **i**nject trace results:

<#root>

```
ASR1000#
show platform packet-trace summary


 Pkt   Input            Output           State  Reason
 0     INJ.2            Gi0/0/1          FWD
 1     Gi0/0/1          internal0/0/rp:0  PUNT   11  (For-us data)
 2     INJ.2            Gi0/0/1          FWD
 3     Gi0/0/1          internal0/0/rp:0  PUNT   11  (For-us data)
 4     INJ.2            Gi0/0/1          FWD
 5     Gi0/0/1          internal0/0/rp:0  PUNT   11  (For-us data)
 6     INJ.2            Gi0/0/1          FWD
 7     Gi0/0/1          internal0/0/rp:0  PUNT   11  (For-us data)
 8     INJ.2            Gi0/0/1          FWD
 9     Gi0/0/1          internal0/0/rp:0  PUNT   11  (For-us data)

 ASR1000#
show platform packet-trace packet 0


 Packet: 0           CBUG ID: 120
 Summary


Input     : INJ.2


 Output    : GigabitEthernet0/0/1
 State     : FWD
 Timestamp
 Start   : 115612780360228 ns (05/29/2014 15:02:55.467987 UTC)
 Stop    : 115612780380931 ns (05/29/2014 15:02:55.468008 UTC)
 Path Trace
 Feature: IPV4
 Source      : 172.16.10.1
 Destination : 172.16.10.2
 Protocol    : 1 (ICMP)

 ASR1000#
```

```
 ASR1000#
```

`show platform packet-trace packet 1`

```
 Packet: 1           CBUG ID: 121
 Summary
 Input     : GigabitEthernet0/0/1
 Output    : internal0/0/rp:0

State     : PUNT 11  (For-us data)

 Timestamp
 Start   : 115612781060418 ns (05/29/2014 15:02:55.468687 UTC)
 Stop    : 115612781120041 ns (05/29/2014 15:02:55.468747 UTC)
 Path Trace
 Feature: IPV4
 Source      : 172.16.10.2
 Destination : 172.16.10.1
 Protocol    : 1 (ICMP)
```

Packet Trace Enhancement with IOSd and LFTS Punt/Inject Trace and UDF Matching (New In 17.3.1)

The packet trace feature is further enhanced to provide additional trace information for packets originated or destined to IOSd or other BinOS processes in Cisco IOS-XE release 17.3.1.

IOSd Drop Tracing

With this enhancement, packet tracing is extended into IOSd, and can provide information about any packet drops inside of IOSd, which are usually reported in the ***show ip traffic*** output. There is no additional configuration required to enable IOSd drop tracing. Here is an example of a UDP packet dropped by IOSd due to bad checksum error:

<#root>

```
Router#debug platform condition ipv4 10.118.74.53/32 both
Router#debug platform condition start
Router#debug platform packet-trace packet 200
Packet count rounded up from 200 to 256

Router#
Router#show plat pack pa 0
Packet: 0           CBUG ID: 674
Summary
  Input     : GigabitEthernet1
  Output    : internal0/0/rp:0
  State     : PUNT 11  (For-us data)
  Timestamp
    Start   : 17756544435656 ns (06/29/2020 18:19:17.326313 UTC)
    Stop    : 17756544469451 ns (06/29/2020 18:19:17.326346 UTC)
Path Trace
  Feature: IPV4(Input)
    Input       : GigabitEthernet1
    Output      : <unknown>
    Source      : 10.118.74.53
```

```
    Destination : 172.18.124.38
    Protocol    : 17 (UDP)
      SrcPort   : 2640
      DstPort   : 500

IOSd Path Flow: Packet: 0    CBUG ID: 674
  Feature: INFRA
  Pkt Direction: IN
    Packet Rcvd From DATAPLANE

  Feature: IP
  Pkt Direction: IN
    Packet Enqueued in IP layer
    Source      : 10.118.74.53
    Destination : 172.18.124.38
    Interface   : GigabitEthernet1

  Feature: IP
  Pkt Direction: IN
  FORWARDED To transport layer
    Source        : 10.118.74.53
    Destination   : 172.18.124.38
    Interface     : GigabitEthernet1

  Feature: UDP
  Pkt Direction: IN


DROPPED
    UDP: Checksum error: dropping

    Source      : 10.118.74.53(2640)
    Destination : 172.18.124.38(500)
```

IOSd Egress Path Tracing

Packet trace is enhanced to show the path trace and protocol processing information as the packet is originated from IOSd and sent in the egress direction towards the network. There is no additional configuration required to capture the IOSd egress path trace information. Here is an example of egress path tracing for an SSH packet egressing the router:

<#root>

```
Router#show platform packet-trace packet 2
Packet: 2           CBUG ID: 2


IOSd Path Flow:

  Feature: TCP
  Pkt Direction: OUTtcp0: O SYNRCVD 172.18.124.38:22 172.18.124.55:52774 seq 3052140910 OPTS 4 ACK 2346

  Feature: TCP
  Pkt Direction: OUT
  FORWARDED
    TCP: Connection is in SYNRCVD state
    ACK         : 2346709419
    SEQ         : 3052140910
    Source      : 172.18.124.38(22)
```

```
    Destination : 172.18.124.55(52774)


  Feature: IP
  Pkt Direction: OUTRoute out the generated packet.srcaddr: 172.18.124.38, dstaddr: 172.18.124.55

  Feature: IP
  Pkt Direction: OUTInject and forward successful srcaddr: 172.18.124.38, dstaddr: 172.18.124.55

  Feature: TCP
  Pkt Direction: OUTtcp0: O SYNRCVD 172.18.124.38:22 172.18.124.55:52774 seq 3052140910 OPTS 4 ACK 2346
Summary
  Input     : INJ.2
  Output    : GigabitEthernet1
  State     : FWD
  Timestamp
    Start   : 490928006866 ns (06/29/2020 13:31:30.807879 UTC)
    Stop    : 490928038567 ns (06/29/2020 13:31:30.807911 UTC)
Path Trace
  Feature: IPV4(Input)
    Input       : internal0/0/rp:0
    Output      : <unknown>
    Source      : 172.18.124.38
    Destination : 172.18.124.55
    Protocol    : 6 (TCP)
      SrcPort   : 22
      DstPort   : 52774
  Feature: IPSec
    Result    : IPSEC_RESULT_DENY
    Action    : SEND_CLEAR
    SA Handle : 0
    Peer Addr : 172.18.124.55
    Local Addr: 172.18.124.38
```

LFTS packet tracing

LFTS (Linux Forwarding Transport Service) is a transport mechanism to forward packets punted from the CPP into applications other than IOSd. LFTS packet tracing enhancement added tracing information for such packets in the path trace output. There is no additional configuration required to obtain the LFTS tracing information. Here is an example output of LFTS tracing for punted packet to the NETCONF application:

<#root>

```
Router#show plat packet-trace pac 0
Packet: 0           CBUG ID: 461
Summary
  Input     : GigabitEthernet1
  Output    : internal0/0/rp:0
  State     : PUNT 11  (For-us data)
  Timestamp
    Start   : 647999618975 ns (06/30/2020 02:18:06.752776 UTC)
    Stop    : 647999649168 ns (06/30/2020 02:18:06.752806 UTC)
Path Trace
  Feature: IPV4(Input)
    Input       : GigabitEthernet1
    Output      : <unknown>
    Source      : 10.118.74.53
```

```
    Destination : 172.18.124.38
    Protocol    : 6 (TCP)
      SrcPort   : 65365
      DstPort   : 830


LFTS Path Flow: Packet: 0    CBUG ID: 461

  Feature: LFTS
  Pkt Direction: IN
    Punt Cause  : 11
     subCause : 0
```

Packet trace pattern matching based on User Defined Filter (ASR1000 platform only)

In Cisco IOS-XE release17.3.1, a new packet matching mechanism is also added to the ASR1000 product families to match on arbitrary field in a packet based on the User Defined Filter (UDF) infrastructure. This allows flexible packet matching based on fields that is not part of the standard L2/L3/L4 header structure. The next example shows a UDF definition that matches on 2 bytes of user defined pattern of 0x4D2 that starts from an offset of 26 bytes from the L3 outer protocol header.

```
udf grekey header outer l3 26 2
ip access-list extended match-grekey
 10 permit ip any any udf grekey 0x4D2 0xFFFF

debug plat condition ipv4 access-list match-grekey both
debug plat condition start
debug plat packet-trace pack 100
```

Packet Trace Examples

This section provides some examples where the packet trace feature is useful for troubleshooting purposes.

Packet Trace Example - NAT

With this example, an interface source Network Address Translation (NAT) is configured on the WAN interface of an ASR1K (Gig0/0/0) for the local subnet (172.16.10.0/24).

Here is the platform condition and packet trace configuration that is used in order to trace the traffic from 172.16.10.2 to 172.16.20.2, which becomes translated (NAT) on the Gig0/0/0 interface:

```
debug platform condition interface Gig 0/0/1 ingress
debug platform condition start
debug platform packet-trace packet 1024 fia-trace
```

When five ICMP packets are sent from 172.16.10.2 to 172.16.20.2 with an interface source NAT configuration, these are the packet trace results:

<#root>

ASR1000#

show platform packet-trace summary

```
 Pkt   Input           Output          State  Reason
 0     Gi0/0/1         Gi0/0/0         FWD
 1     Gi0/0/1         Gi0/0/0         FWD
 2     Gi0/0/1         Gi0/0/0         FWD
 3     Gi0/0/1         Gi0/0/0         FWD
 4     Gi0/0/1         Gi0/0/0         FWD
```

ASR1000#

show platform packet-trace statistics

```
 Packets Summary
 Matched  5
 Traced   5
 Packets Received
 Ingress  5
 Inject   0
 Packets Processed
 Forward  5
 Punt     0
 Drop     0
 Consume  0
```

ASR1000#

show platform packet-trace packet 0

```
 Packet: 0           CBUG ID: 146
 Summary
 Input     : GigabitEthernet0/0/1
 Output    : GigabitEthernet0/0/0
 State     : FWD
 Timestamp
 Start   : 3010217805313 ns (05/17/2014 07:01:52.227836 UTC)
 Stop    : 3010217892847 ns (05/17/2014 07:01:52.227923 UTC)
 Path Trace
 Feature: IPV4
 Source      : 172.16.10.2
 Destination : 172.16.20.2
 Protocol    : 1 (ICMP)
 Feature: FIA_TRACE
 Entry      : 0x806c7eac - DEBUG_COND_INPUT_PKT
 Lapsed time: 1031 ns
 Feature: FIA_TRACE
 Entry      : 0x82011c00 - IPV4_INPUT_DST_LOOKUP_CONSUME
 Lapsed time: 462 ns
 Feature: FIA_TRACE
 Entry      : 0x82000170 - IPV4_INPUT_FOR_US_MARTIAN
 Lapsed time: 355 ns
 Feature: FIA_TRACE
 Entry      : 0x803c6af4 - IPV4_INPUT_VFR
 Lapsed time: 266 ns
 Feature: FIA_TRACE
 Entry      : 0x82004500 - IPV4_OUTPUT_LOOKUP_PROCESS
 Lapsed time: 942 ns
 Feature: FIA_TRACE
```

```
 Entry      : 0x8041771c - IPV4_INPUT_IPOPTIONS_PROCESS
 Lapsed time: 88 ns
 Feature: FIA_TRACE
 Entry      : 0x82013400 - MPLS_INPUT_GOTO_OUTPUT_FEATURE
 Lapsed time: 568 ns
 Feature: FIA_TRACE
 Entry      : 0x803c6900 - IPV4_OUTPUT_VFR
 Lapsed time: 266 ns


Feature: NAT
 Direction   : IN to OUT
 Action      : Translate Source
 Old Address : 172.16.10.2  00028
 New Address : 192.168.10.1 00002


 Feature: FIA_TRACE
 Entry      : 0x8031c248 - IPV4_NAT_OUTPUT_FIA
 Lapsed time: 55697 ns
 Feature: FIA_TRACE
 Entry      : 0x801424f8 - IPV4_OUTPUT_THREAT_DEFENSE
 Lapsed time: 693 ns
 Feature: FIA_TRACE
 Entry      : 0x803c60b8 - IPV4_MC_OUTPUT_VFR_REFRAG
 Lapsed time: 88 ns
 Feature: FIA_TRACE
 Entry      : 0x82014900 - IPV6_INPUT_L2_REWRITE
 Lapsed time: 444 ns
 Feature: FIA_TRACE
 Entry      : 0x82000080 - IPV4_OUTPUT_FRAG
 Lapsed time: 88 ns
 Feature: FIA_TRACE
 Entry      : 0x8200e600 - IPV4_OUTPUT_DROP_POLICY
 Lapsed time: 1457 ns
 Feature: FIA_TRACE
 Entry      : 0x82017980 - MARMOT_SPA_D_TRANSMIT_PKT
 Lapsed time: 7431 ns
 ASR1000#
```

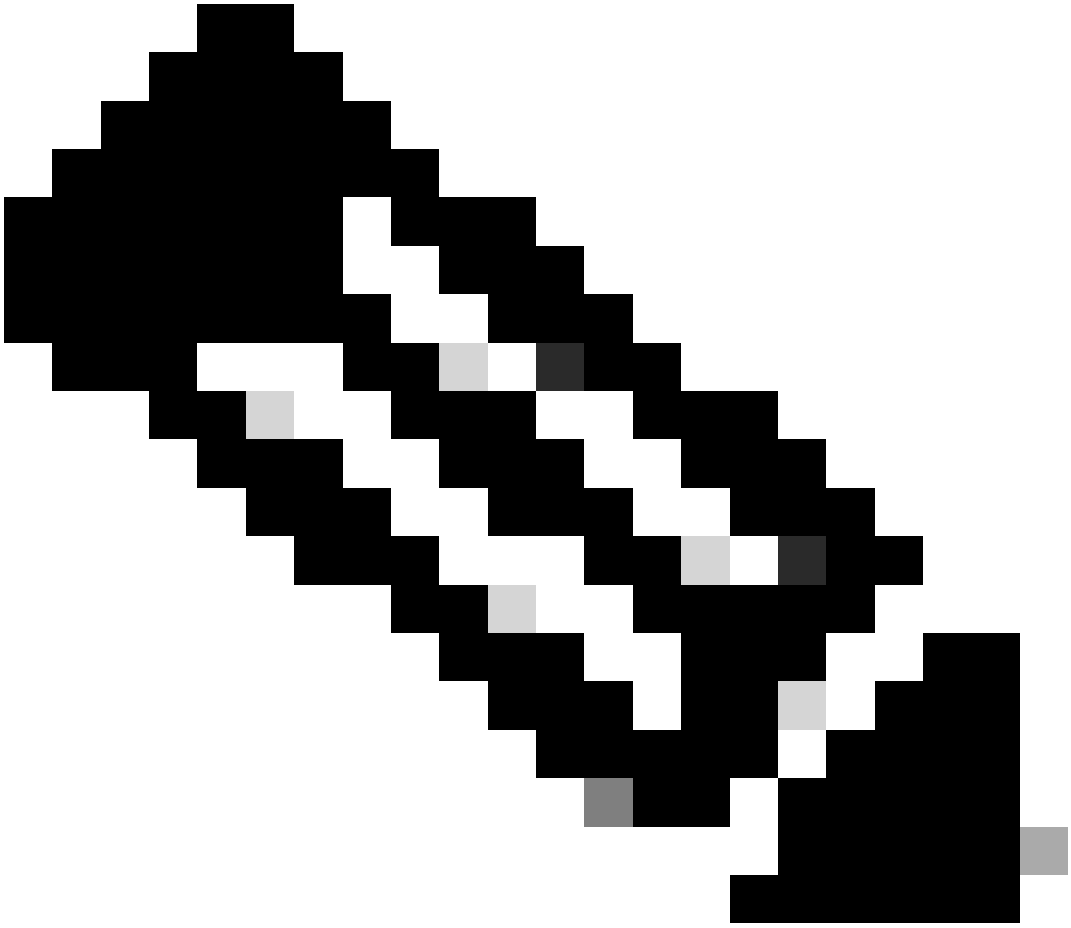
Packet Trace Example - VPN

With this example, a site-to-site VPN tunnel is used between the ASR1K and the Cisco IOS router in order to protect the traffic that flows between 172.16.10.0/24 and 172.16.20.0/24 (local and remote subnets).

Here is the platform condition and packet trace configuration that is used in order to trace the VPN traffic that flows from 172.16.10.2 to 172.16.20.2 on the Gig 0/0/1 interface:

```
debug platform condition interface Gig 0/0/1 ingress
debug platform condition start
debug platform packet-trace packet 1024 fia-trace
```

When five ICMP packets are sent from 172.16.10.2 to 172.16.20.2, which are encrypted by the VPN tunnel between the ASR1K and the Cisco IOS router in this example, these are the packet trace outputs:



Note: The packet traces show the QFP Security Association (SA) handle in the trace that is used in order to encrypt the packet, which is useful when you troubleshoot IPsec VPN issues in order to verify that the correct SA is used for encryption.

<#root>

ASR1000#

show platform packet-trace summary

```
 Pkt   Input           Output          State  Reason
 0     Gi0/0/1         Gi0/0/0         FWD
 1     Gi0/0/1         Gi0/0/0         FWD
 2     Gi0/0/1         Gi0/0/0         FWD
 3     Gi0/0/1         Gi0/0/0         FWD
 4     Gi0/0/1         Gi0/0/0         FWD
```

ASR1000#

show platform packet-trace packet 0

```
 Packet: 0          CBUG ID: 211
 Summary
 Input     : GigabitEthernet0/0/1
 Output    : GigabitEthernet0/0/0
 State     : FWD
 Timestamp
 Start   : 4636921551459 ns (05/17/2014 07:28:59.211375 UTC)
 Stop    : 4636921668739 ns (05/17/2014 07:28:59.211493 UTC)
 Path Trace
 Feature: IPV4
 Source      : 172.16.10.2
 Destination : 172.16.20.2
 Protocol    : 1 (ICMP)
 Feature: FIA_TRACE
 Entry      : 0x806c7eac - DEBUG_COND_INPUT_PKT
 Lapsed time: 622 ns
 Feature: FIA_TRACE
 Entry      : 0x82011c00 - IPV4_INPUT_DST_LOOKUP_CONSUME
 Lapsed time: 462 ns
 Feature: FIA_TRACE
 Entry      : 0x82000170 - IPV4_INPUT_FOR_US_MARTIAN
 Lapsed time: 320 ns
 Feature: FIA_TRACE
 Entry      : 0x82004500 - IPV4_OUTPUT_LOOKUP_PROCESS
 Lapsed time: 1102 ns
 Feature: FIA_TRACE
 Entry      : 0x8041771c - IPV4_INPUT_IPOPTIONS_PROCESS
 Lapsed time: 88 ns
 Feature: FIA_TRACE
 Entry      : 0x82013400 - MPLS_INPUT_GOTO_OUTPUT_FEATURE
 Lapsed time: 586 ns
 Feature: FIA_TRACE
 Entry      : 0x803c6900 - IPV4_OUTPUT_VFR
 Lapsed time: 266 ns
 Feature: FIA_TRACE
 Entry      : 0x80757914 - MC_OUTPUT_GEN_RECYCLE
 Lapsed time: 195 ns
 Feature: FIA_TRACE
 Entry      : 0x803c60b8 - IPV4_MC_OUTPUT_VFR_REFRAG
 Lapsed time: 88 ns


Feature: IPSec
 Result    : IPSEC_RESULT_SA
 Action    : ENCRYPT
 SA Handle : 6
 Peer Addr : 192.168.20.1
 Local Addr: 192.168.10.1


 Feature: FIA_TRACE
 Entry      : 0x8043caec - IPV4_OUTPUT_IPSEC_CLASSIFY
 Lapsed time: 9528 ns
 Feature: FIA_TRACE
 Entry      : 0x8043915c - IPV4_OUTPUT_IPSEC_DOUBLE_ACL
 Lapsed time: 355 ns
 Feature: FIA_TRACE
 Entry      : 0x8043b45c - IPV4_IPSEC_FEATURE_RETURN
 Lapsed time: 657 ns
 Feature: FIA_TRACE
 Entry      : 0x8043ae28 - IPV4_OUTPUT_IPSEC_RERUN_JUMP
 Lapsed time: 888 ns
 Feature: FIA_TRACE
```

```
Entry      : 0x80436f10 - IPV4_OUTPUT_IPSEC_POST_PROCESS
Lapsed time: 2186 ns
Feature: FIA_TRACE
Entry      : 0x8043b45c - IPV4_IPSEC_FEATURE_RETURN
Lapsed time: 675 ns
Feature: FIA_TRACE
Entry      : 0x82014900 - IPV6_INPUT_L2_REWRITE
Lapsed time: 1902 ns
Feature: FIA_TRACE
Entry      : 0x82000080 - IPV4_OUTPUT_FRAG
Lapsed time: 71 ns
Feature: FIA_TRACE
Entry      : 0x8200e600 - IPV4_OUTPUT_DROP_POLICY
Lapsed time: 1582 ns
Feature: FIA_TRACE
Entry      : 0x82017980 - MARMOT_SPA_D_TRANSMIT_PKT
Lapsed time: 3964 ns
ASR1000#
```

Performance Impact

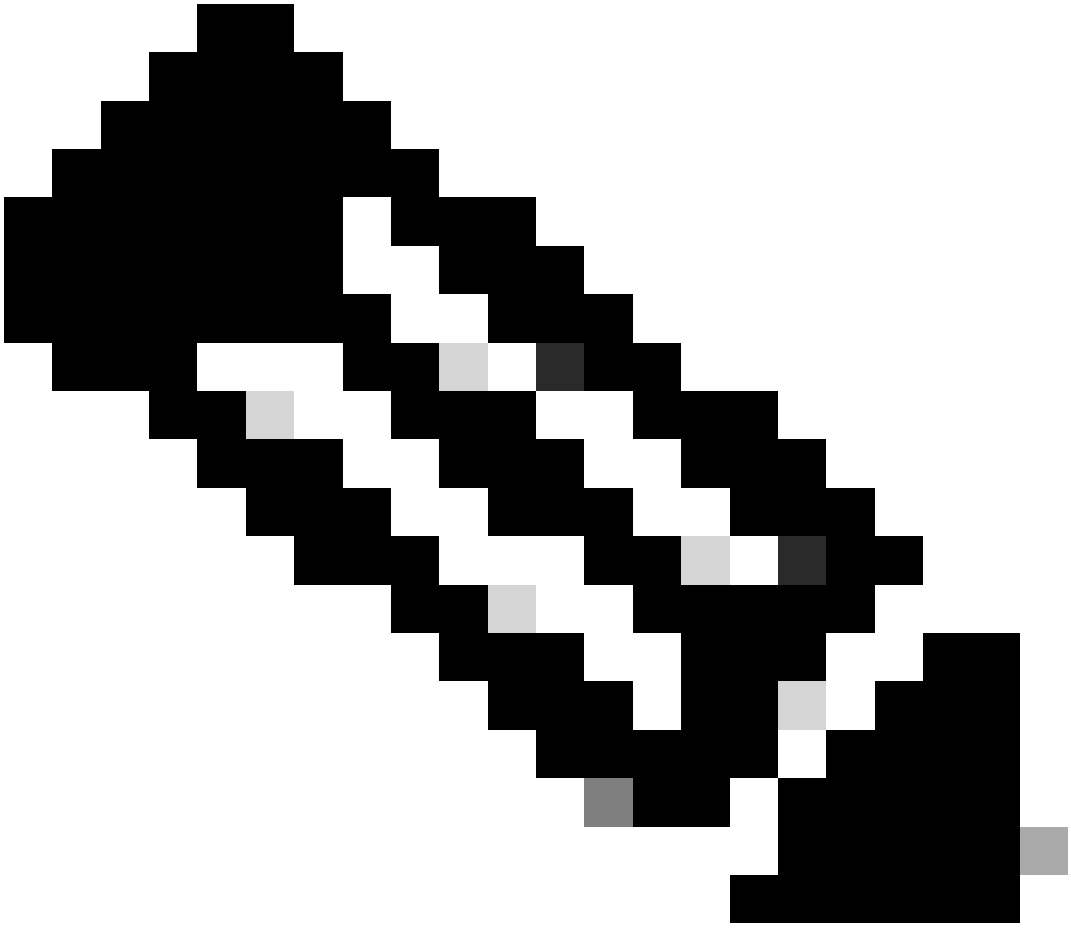
Packet trace buffers consume QFP DRAM, so be mindful of the amount of memory that a configuration requires and the amount of memory that is available.

The performance impact varies, dependent upon the packet trace options that are enabled. The packet trace only affects the forwarding performance of the packets that are traced, such as those packets that match the user-configured conditions. The more granular and detailed information that you configure the packet trace to capture, the greater it can impact resources.

As with any troubleshooting, it is best to take an iterative approach and only enable the more-detailed trace options when a debug situation warrants it.

The QFP DRAM usage can be estimated with this formula:

memory needed = (stats overhead) + num of pkts * (summary size + path data size + copy size)



Note: Where the **stats overhead** and **summary size** are fixed at 2 KB and 128 B, respectively, the **path data size** and **copy size** are user-configurable.

Related Information

- **Cisco ASR1000 Series Aggregation Series Routers Software Configuration Guide - Packet Trace**
- **Packet Drops on Cisco ASR1000 Series Service Routers**
- **Cisco Technical Support & Downloads**