Troubleshoot Punt Keepalive Failures in Cisco IOS XE

Contents

Introduction

This document describes how to troubleshoot punt keep alive failures.

Prerequisites

Requirements

Basic knowledge in Cisco IOS® XE.

Components Used

This document is based on Cisco IOS XE routers like CSR8000v, ASR1000 and ISR4000 Series.

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

The punt path in Cisco IOS XE based systems is an internal data path. This is the path where communication between control plane and data plane takes place.

This internal path is used to transmit control plane packets for the router consumption.

When this path fails, you can see this type of error in log.

```
%IOSXE_INFRA-4-NO_PUNT_KEEPALIVE:  Keepalive not received for 60 seconds
```

The keep alive messages are messages that monitor the health of the path between the QFP and the RP.

This path is critical for the system to operate.

If these keep alives are not received within 5 minutes, you can see a critical log like this:

```
%IOSXE_INFRA-2-FATAL_NO_PUNT_KEEPALIVE:  Keepalive not received for 300 seconds resetting
```

System resets in order to recover from this condition.

The Punt Debug Log File

On the event of punt keep alive failures and resets due to it, the system creates a file called punt_debug.log which collects relevant data to understand the behavior at issue time.

Note: Ensure to have the system up-to-date with the latest Cisco IOS XE software release for file punt_debug.log to be generated.

This file contains these commands execution multiple times in order to understand different counters.

show platform software infra punt-keepalive

show platform software infra lsmpi

show platform software infrastructure lsmpi driver

show platform software infra lsmpi bufusage

show platform software punt-policer

show platform software status control-processor brief

show process cpu platform sorted

show platform software infrastructure punt

show platform hardware qfp active statistics drop

show platform hardware qfp active infra punt statistics type per-cause

show platform hardware qfp active infrastructure bqs queue output default all

Note: Within the punt_debug.log, you focus on error indicators and large quantity of packets that can cause the issue.

The Linux Shared Memory Punt Interface (LSMPI)

This component is used to transmit packets and messages from forwarding processor to routing processor.

The Punt Policer

The punt policer is a control plane protection mechanism that allows the system to protect and police control plane packets.

With the command **show platform software punt-policer**, you can see the conform packets and the dropped due to this policer.

```
----------------- show platform software punt-policer ------------------
```

```
Per Punt-Cause Policer Configuration and Packet Counters


Punt                                   Config Rate(pps)     Conform Packets                  Dropped Pack
Cause  Description                      Normal   High     Normal           High             Normal
-------------------------------------------------------------------------------------------------------
  2    IPv4 Options                     874      655      0                0                0
  3    Layer2 control and legacy        8738     2185     0                0                0
  4    PPP Control                      437      1000     0                0                0
  -- snip : output omitted for brevity  --
```

The command **show platform software infrastructure punt** shows counter data about punt causes.

```
------------------ show platform software infrastructure punt ------------------


LSMPI interface internal stats:
enabled=0, disabled=0, throttled=0, unthrottled=0, state is ready
Input Buffers = 51181083
Output Buffers = 51150283
-- snip : output omitted for brevity  --
EPC CP RX Pkt cleansed 0
Punt cause out of range 0
IOSXE-RP Punt packet causes:
     3504959 ARP request or response packets
          27 Incomplete adjacency packets
-- snip : output omitted for brevity  --

   FOR_US Control IPv4 protcol stats:
        2369262 TCP packets

   FOR_US Control IPv6 protcol stats:
           6057 ICMPV6 packets
Packet histogram(500 bytes/bin), avg size in 119, out 95:
 Pak-Size      In-Count       Out-Count
      0+:      51108211         51144723
    500+:         22069             2632
   1000+:          2172                0
   1500+:          3170                0
```

This data is relevant to understand what can be impacting the punt keep alive path.

Embedded Event Manager (EEM) for Data Collection

In the case that the punt_debug.log does not provide enough data to diagnose the problem, EEM scripting can be used to get more data points at issue time.

```
event manager applet punt_script authorization bypass
event syslog pattern "IOSXE_INFRA-4-NO_PUNT_KEEPALIVE" maxrun 1000
action 0.0 cli command "enable"
action 0.1 set i "0"
action 0.2 cli command "test platform software punt-keepalive ignore-fault"
action 0.3 while $i lt 10
action 0.4 syslog msg "iteration $i"
action 0.9 cli command "show clock | append bootflash:qfp_lsmpi.txt"
action 1.0 cli command "show platform software infrastructure lsmpi | append bootflash:qfp_lsmpi.txt"
action 1.1 cli command "show platform software infrastructure lsmpi driver | append bootflash:qfp_lsmpi
action 1.2 cli command "show platform software infrastructure lsmpi driver 0 | append bootflash:qfp_lsm
action 1.3 cli command "show platform software infrastructure lsmpi bufusage | append bootflash:qfp_lsm
action 1.4 cli command "show platform software infrastructure lsmpi bufusage 0 | append bootflash:qfp_l
action 1.5 cli command "show platform software infrastructure punt-keepalive | append bootflash:qfp_lsm
action 1.6 cli command "show platform software infrastructure punt | append bootflash:qfp_lsmpi.txt"
action 1.7 cli command "show platform software punt-policer | append bootflash:qfp_lsmpi.txt"
action 1.8 cli command "show platform hardware qfp active infrastructure punt stat type per-cause | app
action 1.9 cli command "show platform hardware qfp active infrastructure punt statistics type punt-drop
action 1.a cli command "show platform hardware qfp active infrastructure punt statistics type inject-dr
action 1.b cli command "show platform hardware qfp active infrastructure bqs queue output default inter
action 1.c cli command "show platform hardware qfp active statistics drop | append bootflash:qfp_lsmpi.
action 1.d cli command "show platform hardware qfp active datapath utilization | append bootflash:qfp_l
action 1.e cli command "show platform hardware qfp active datapath infrastructure sw-hqf | append bootf
action 1.f cli command "show platform hardware qfp active datapath infrastructure sw-distrib | append b
action 1.g cli command "show platform hardware qfp active datapath infrastructure sw-pktmem | append bo
action 1.h cli command "show platform software status control-processor brief | append bootflash:qfp_ls
action 2.0 increment i
action 2.1 wait 3
action 2.4 end
action 3.0 syslog msg "End of data collection. Please transfer the file at bootflash:qfp_lsmpi.txt"
action 5.0 cli command "debug platform hardware qfp active datapath crashdump"
```

Note: The commands contained within the script vary depending on the platform where it is configured.

This script allows you to understand the lsmpi, resources, and punt state during issue time.

The EEM script includes the command **debug platform hardware qfp active datapath crashdump** that generates the qfp core dump, needed by developer team and TAC.

Note: If filing a case with Cisco TAC, please provide the core file generated by the script.

If a packet trace is needed, this amendment can be added to the script:

First, setup the packet trace configuration, which can be done out of the EEM script:

```
debug platform packet-trace packet 8192 fia-trace circular
debug platform condition both
debug platform packet-trace copy packet both L2
```

Then, start and stop it with these actions within the EEM script:

```
action 6.2 cli command "debug platform condition start"
action 6.3 wait 8
action 6.4 cli command "debug platform condition stop"
```

Then, dump the data with these commands in a separate file:

```
action 6.5 cli command "show platform packet-trace statistics | append bootflash:traceAll.txt"
action 6.6 cli command "show platform packet-trace summary | append bootflash:traceAll.txt"
```

action 6.7 cli command "show platform packet-trace packet all decode | append bootflash:traceAll.txt"

This packet trace actions logic is added right after the end statement of the while cycle within the EEM script.

This script allows you to understand what type of packets can be causing the issue.

Packet trace is a feature documented at Troubleshoot with the IOS XE Datapath Packet Trace Feature

A Practical Example

A CSR8000v is constantly rebooting.

After extracting the system report, you can observe a crashdump, and an iosd core file indicating punt keep alive related functions within the stack trace.

Note: For the stack trace decoding, TAC assistance is required.

However, the crashinfo file is in clear text and you can see these symptoms:

```
Jan 15 14:29:41.756 AWST: %IOSXE_INFRA-4-NO_PUNT_KEEPALIVE:  Keepalive not received for 160 seconds
Jan 15 14:30:01.761 AWST: %IOSXE_INFRA-4-NO_PUNT_KEEPALIVE:  Keepalive not received for 180 seconds
Jan 15 14:30:21.766 AWST: %IOSXE_INFRA-4-NO_PUNT_KEEPALIVE:  Keepalive not received for 200 seconds
Jan 15 14:30:41.776 AWST: %IOSXE_INFRA-4-NO_PUNT_KEEPALIVE:  Keepalive not received for 220 seconds
Jan 15 14:31:01.780 AWST: %IOSXE_INFRA-4-NO_PUNT_KEEPALIVE:  Keepalive not received for 240 seconds
Jan 15 14:31:41.789 AWST: %IOSXE_INFRA-4-NO_PUNT_KEEPALIVE:  Keepalive not received for 280 seconds
Jan 15 14:32:01.791 AWST: %IOSXE_INFRA-4-NO_PUNT_KEEPALIVE:  Keepalive not received for 300 seconds
Jan 15 14:32:01.791 AWST: %IOSXE_INFRA-2-FATAL_NO_PUNT_KEEPALIVE:  Keepalive not received for 300 secon

%Software-forced reload

Exception to IOS Thread:
Frame pointer 0x7F0AE0EE29A8, PC = 0x7F0B342C16D2

UNIX-EXT-SIGNAL: Aborted(6), Process = PuntInject Keepalive Process
-Traceback= 1#7b5996c3
```

The process impacted is PuntInject Keepalive Process.

The system must trigger an abort signal when the keepalive reaches the 300 seconds threshold mark.

The punt_debug.log reveals some transmit failures within the command **show platform software infrastructure lsmpi driver**:

```
Reason for TX drops (sticky):
    Bad packet len   : 0
    Bad buf len      : 0
    Bad ifindex      : 0
    No device        : 0
    No skbuff        : 0
    Device xmit fail : 82541   >>>>>>>>>>>>>>>>>>>>> Tx failure
```

This is a generic failure.

This counter increases within multiple samples taken in the file.

The EEM script was provided to get more data about the resources, punt data path and other infrastructure related commands.

By checking the lsmpi traffic punt counters you see that EIGRP control plane packets are remarkable. These are packets identified as for us packets:

```
17660574 For-us data packets
543616 RP<->QFP keepalive packets
1004 Glean adjacency packets
3260636 BFD control packets
122523839 For-us control packets<<<<

FOR_US Control IPv4 protcol stats:
```

```
153551 TCP packets
2663105 GRE packets
104394559 EIGRP packets<<<<
```

Later on, it was found that the hypervisor was oversubscribed, impacting the underlying compute resources.

The CSR8000v was deployed in another hypervisor, and this helped to mitigate the problem.

Enhancements

Enhancement for automatic qfp core file generation was introduced starting Cisco IOS XE 17.15 version via Cisco bug ID CSCwf85505