Fehlerbehebung bei Catalyst Switches der Serie 3850 Hohe CPU-Auslastung

Inhalt

Einleitung

In diesem Dokument wird beschrieben, wie Sie auf der neuen Cisco IOS®-XE-Plattform Probleme bei der CPU-Auslastung beheben, die hauptsächlich auf Unterbrechungen zurückzuführen sind.

Hintergrundinformationen

Es ist wichtig zu verstehen, wie Cisco IOS®-XE aufgebaut ist. Mit Cisco IOS®-XE ist Cisco auf einen Linux-Kernel umgestiegen, und alle Subsysteme wurden in Prozesse unterteilt. Alle Subsysteme, die sich zuvor in Cisco IOS® befanden, wie z. B. die Modultreiber, Hochverfügbarkeit usw., werden jetzt als Softwareprozesse innerhalb des Linux-Betriebssystems ausgeführt. Cisco IOS® selbst wird als Daemon innerhalb des Linux-Betriebssystems (IOSd) ausgeführt. Cisco IOS®-XE bietet nicht nur dasselbe Erscheinungsbild wie das klassische Cisco IOS®, sondern auch Betrieb, Support und Management.

Darüber hinaus enthält das Dokument eine Reihe neuer Befehle auf dieser Plattform, die zur Behebung von CPU-Nutzungsproblemen unverzichtbar sind.

Hier einige nützliche Definitionen:

- Forwarding Engine Driver (FED): Dieser Treiber bildet das Herzstück des Cisco Catalyst Switches der Serie 3850 und ist für die gesamte Hardware-Programmierung/-Weiterleitung zuständig.

- Cisco IOSd®: Dies ist der Cisco IOS®-Daemon, der auf dem Linux-Kernel ausgeführt wird. Es wird als ein Softwareprozess innerhalb des Kernels ausgeführt.

- Packet Delivery System (PDS): Dies ist die Architektur und der Prozess für die Zustellung von Paketen zu und von verschiedenen Subsystemen. Beispielsweise steuert es, wie Pakete von

der FED an das IOSd und umgekehrt übermittelt werden.

- Handle: Ein Handle kann als Zeiger gedacht werden. Es ist ein Mittel, um detailliertere Informationen über bestimmte Variablen zu finden, die in den von der Box erzeugten Ausgaben verwendet werden. Dies ähnelt dem Konzept der LTL-Indizes (Local Target Logic) beim Cisco Catalyst Switch der Serie 6500.

Anwenderbericht: Address Resolution Protocol (ARP)-Interrupts

Der in diesem Abschnitt beschriebene Fehlerbehebungs- und Verifizierungsprozess kann bei einer hohen CPU-Auslastung aufgrund von Unterbrechungen umfassend genutzt werden.

Schritt 1: Identifizieren des Prozesses, der CPU-Zyklen beansprucht

Der Befehl **show process cpu** zeigt auf natürliche Weise an, wie die CPU derzeit aussieht. Beachten Sie, dass der Cisco Catalyst Switch der Serie 3850 vier Kerne verwendet. Hier wird die CPU-Auslastung für alle vier Kerne aufgelistet:

```
3850-2#show processes cpu sort | exclude 0.0
Core 0: CPU utilization for five seconds: 53%; one minute: 39%;  five minutes: 41%
Core 1: CPU utilization for five seconds: 43%; one minute: 57%;  five minutes: 54%
Core 2: CPU utilization for five seconds: 95%; one minute: 60%;  five minutes: 58%
Core 3: CPU utilization for five seconds: 32%; one minute: 31%;  five minutes: 29%
PID    Runtime(ms) Invoked  uSecs  5Sec     1Min     5Min     TTY   Process
8525   472560      2345554  7525   31.37    30.84    30.83    0     iosd
5661   2157452     9234031  698    13.17    12.56    12.54    1088  fed
6206   19630       74895    262    1.83     0.43     0.10     0     eicored
6197   725760      11967089 60     1.41     1.38     1.47     0     pdsd
```

Aus der Ausgabe wird deutlich, dass der Cisco IOS®-Daemon einen Großteil der CPU zusammen mit der FED verbraucht, dem Herzstück dieser Box. Wenn die CPU-Auslastung aufgrund von Interrupts hoch ist, sehen Sie, dass Cisco IOSd® und FED einen Großteil der CPU verwenden und dass diese Unterprozesse (oder eine Teilmenge davon) die CPU verwenden:

- FED-Stempel TX
- FED-Stempel RX
- FED-Stanzerauffüllung
- FED-Stempel - TX abgeschlossen

Mit dem Befehl **show process cpu detail<*process*>** können Sie einen beliebigen dieser Prozesse vergrößern. Da Cisco IOSd® für den Großteil der CPU-Auslastung verantwortlich ist, sehen wir uns das genauer an.

```
3850-2#show processes cpu detailed process iosd sort | ex 0.0
Core 0: CPU utilization for five seconds: 36%; one minute: 39%; five minutes: 40%
Core 1: CPU utilization for five seconds: 73%; one minute: 52%; five minutes: 53%
Core 2: CPU utilization for five seconds: 22%; one minute: 56%; five minutes: 58%
Core 3: CPU utilization for five seconds: 46%; one minute: 40%; five minutes: 31%
PID    T C  TID  Runtime(ms)Invoked uSecs 5Sec   1Min   5Min  TTY  Process
                                          (%)    (%)    (%)
8525   L          556160     2356540 7526  30.42   30.77  30.83  0   iosd
8525   L 1  8525  712558     284117  0     23.14   23.33  23.38  0   iosd
59     I          1115452    4168181 0     42.22   39.55  39.33  0   ARP Snoop
198    I          3442960    4168186 0     25.33   24.22 24.77  0   IP Host Track Proce
```

```
30     I          3802130    4168183 0     24.66   27.88  27.66  0   ARP Input
283    I          574800     3225649 0     4.33    4.00   4.11   0   DAI Packet Process

3850-2#show processes cpu detailed process fed sorted | ex 0.0
Core 0: CPU utilization for five seconds: 45%; one minute: 44%; five minutes: 44%
Core 1: CPU utilization for five seconds: 38%; one minute: 44%; five minutes: 45%
Core 2: CPU utilization for five seconds: 42%; one minute: 41%; five minutes: 40%
Core 3: CPU utilization for five seconds: 32%; one minute: 30%; five minutes: 31%
PID    T C  TID  Runtime(ms)Invoked uSecs 5Sec   1Min   5Min  TTY  Process
                                          (%)    (%)    (%)
5638   L          612840     1143306 536   13.22  12.90   12.93 1088 fed
5638   L 3  8998  396500     602433  0     9.87   9.63    9.61  0     PunjectTx
5638   L 3  8997  159890     66051   0     2.70   2.70    2.74  0     PunjectRx
```

Die Ausgabe (Cisco IOSd® CPU-Ausgabe) zeigt, dass ARP Snoop, IP Host Track Process und ARP Input hoch sind. Dies tritt häufig auf, wenn die CPU aufgrund von ARP-Paketen unterbrochen wird.

Schritt 2: Bestimmen der CPU-Warteschlange, die die hohe CPU-Auslastung verursacht

Der Cisco Catalyst Switch der Serie 3850 verfügt über eine Reihe von Warteschlangen, die verschiedenen Pakettypen gerecht werden (die FED unterhält 32 RX CPU-Warteschlangen, d. h. Warteschlangen, die direkt an die CPU gehen). Es ist wichtig, diese Warteschlangen zu überwachen, um festzustellen, welche Pakete an die CPU gesendet und welche vom Cisco IOSd® verarbeitet werden. Diese Warteschlangen gelten pro ASIC.

> **Hinweis**: Es gibt zwei ASICs: 0 und 1. Die Ports 1 bis 24 gehören zu ASIC 0.

Um die Warteschlangen anzuzeigen, geben Sie den Befehl **show platform punt statistics port-asic** *<port-asic>* **cpuq** *<queue>* **direction** *<rx|tx>ein*.

Im Befehl **show platform punt statistics port-asic 0 cpuq -1 direction rx** listet das Argument -1 alle Warteschlangen auf. Daher listet dieser Befehl alle Empfangswarteschlangen für Port-ASIC 0 auf.

Nun müssen Sie ermitteln, welche Warteschlange eine große Anzahl von Paketen mit hoher Geschwindigkeit weiterleitet. In diesem Beispiel ergab eine Untersuchung der Warteschlangen diesen Täter:

```
<snip>
RX (ASIC2CPU) Stats (asic 0 qn 16 lqn 16):
RXQ 16: CPU_Q_PROTO_SNOOPING
--------------------------------------
Packets received from ASIC     : 79099152
Send to IOSd total attempts    : 79099152
Send to IOSd failed count      : 1240331
RX suspend count               : 1240331
RX unsuspend count             : 1240330
RX unsuspend send count        : 1240330
RX unsuspend send failed count : 0
RX dropped count               : 0
RX conversion failure dropped  : 0
RX pkt_hdr allocation failure  : 0
RX INTACK count                : 0
RX packets dq'd after intack   : 0
Active RxQ event               : 9906280
```

```
RX spurious interrupt          : 0
<snip>
```

Die Warteschlangennummer lautet 16, und der Warteschlangenname lautet CPU_Q_PROTO_SNOOPING.

Eine andere Möglichkeit, die verantwortliche Warteschlange zu erkennen, besteht darin, den Befehl **show platform punt client** einzugeben.

```
3850-2#show platform punt client
 tag      buffer        jumbo    fallback     packets   received   failures
                                              alloc   free  bytes    conv  buf
 27       0/1024/2048     0/5       0/5         0     0          0     0     0
 65536    0/1024/1600     0/0       0/512       0     0          0     0     0
 65537    0/ 512/1600     0/0       0/512    1530  1530     244061     0     0
 65538    0/   5/5        0/0       0/5         0     0          0     0     0
 65539    0/2048/1600     0/16      0/512       0     0          0     0     0
 65540    0/ 128/1600     0/8       0/0         0     0          0     0     0
 65541    0/ 128/1600     0/16      0/32        0     0          0     0     0
 65542    0/ 768/1600     0/4       0/0         0     0          0     0     0
 65544    0/  96/1600     0/4       0/0         0     0          0     0     0
 65545    0/  96/1600     0/8       0/32        0     0          0     0     0
 65546    0/ 512/1600     0/32      0/512       0     0          0     0     0
 65547    0/  96/1600     0/8       0/32        0     0          0     0     0
 65548    0/ 512/1600     0/32      0/256       0     0          0     0     0
 65551    0/ 512/1600     0/0       0/256       0     0          0     0     0
 65556    0/  16/1600     0/4       0/0         0     0          0     0     0
 65557    0/  16/1600     0/4       0/0         0     0          0     0     0
 65558    0/  16/1600     0/4       0/0         0     0          0     0     0
 65559    0/  16/1600     0/4       0/0         0     0          0     0     0
 65560    0/  16/1600     0/4       0/0         0     0          0     0     0
s65561  421/ 512/1600     0/0       0/128   79565859 131644697  478984244     0 37467
 65563    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65564    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65565    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65566    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65581    0/   1/1        0/0       0/0         0     0          0     0     0
 131071   0/  96/1600     0/4       0/0         0     0          0     0     0
fallback pool: 98/1500/1600
jumbo pool:    0/128/9300
```

Bestimmen Sie den Tag, dem die meisten Pakete zugewiesen wurden. In diesem Beispiel ist dies 65561.

Geben Sie dann den folgenden Befehl ein:

```
3850-2#show pds tag all | in Active|Tags|65561
  Active   Client Client
    Tags   Handle Name                 TDA       SDA         FDA   TBufD       TBytD
   65561  7296672 Punt Rx Proto Snoop  79821397  79821397    0  79821397   494316524
```

Diese Ausgabe zeigt, dass es sich bei der Warteschlange um Rx Proto Snoop handelt.

Das s vor dem 65561 in der Ausgabe des Befehls **show platform punt client** bedeutet, dass der FED-Handle angehalten und durch die Anzahl eingehender Pakete überlastet wird. Wenn das s nicht verschwindet, bedeutet dies, dass die Warteschlange dauerhaft feststeckt.

Schritt 3: Verwerfen Sie das an die CPU gesendete Paket.

In den Ergebnissen des Befehls **show pds tag all** wird ein Handle, 7296672, neben dem Punt Rx Proto Snoop angezeigt.

Verwenden Sie dieses Handle im Befehl **show pds client *<handle>* packet last sink**. Beachten Sie, dass Sie debug pds pktbuf-last aktivieren müssen, bevor Sie den Befehl verwenden. Andernfalls tritt der folgende Fehler auf:

```
3850-2#show pds client 7296672 packet last sink
% switch-2:pdsd:This command works in debug mode only. Enable debug using
"debug pds pktbuf-last" command
```

Wenn das Debuggen aktiviert ist, wird folgende Ausgabe angezeigt:

```
3850-2#show pds client 7296672 packet last sink
Dumping Packet(54528) # 0 of Length 60
---------------------------------------
Meta-data
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0010 00 00 16 1d 00 00 00 00 00 00 00 00 55 5a 57 f0  ............UZW.
0020 00 00 00 00 fd 01 10 df 00 5b 70 00 00 10 43 00  .........[p...C.
0030 00 10 43 00 00 41 fd 00 00 41 fd 00 00 00 00 00  ..C..A...A......
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0050 00 00 00 3c 00 00 00 00 00 01 00 19 00 00 00 00  ...<............
0060 01 01 b6 80 00 00 00 4f 00 00 00 00 00 00 00 00  .......O........
0070 01 04 d8 80 00 00 00 33 00 00 00 00 00 00 00 00  .......3........
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0090 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00a0 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 00  ................
Data
0000 ff ff ff ff ff ff aa bb cc dd 00 00 08 06 00 01  ................
0010 08 00 06 04 00 01 aa bb cc dd 00 00 c0 a8 01 0a  ................
0020 ff ff ff ff ff ff c0 a8 01 14 00 01 02 03 04 05  ................
0030 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11              ............
```

Mit diesem Befehl wird das letzte von der Senke empfangene Paket ausgelesen, in diesem Beispiel IOSd. Dies zeigt, dass der Header ausgelesen und mit Terminal-based Wireshark (TShark) decodiert werden kann. Die Meta-Daten sind für die interne Verwendung durch das System vorgesehen, aber die Datenausgabe liefert tatsächliche Paketinformationen. Die Meta-Daten bleiben jedoch äußerst nützlich.

Beachten Sie die Zeile, die mit 0070 beginnt. Verwenden Sie die ersten 16 Bit danach, wie hier dargestellt:

```
3850-2#show platform port-asic ifm iif-id 0x0104d88000000033
Interface Table
Interface IIF-ID        : 0x0104d88000000033
Interface Name          : Gi2/0/20
Interface Block Pointer : 0x514d2f70
Interface State         : READY
Interface Stauts        : IFM-ADD-RCVD, FFM-ADD-RCVD
Interface Ref-Cnt       : 6
Interface Epoch         : 0
Interface Type          : ETHER
       Port Type         : SWITCH PORT
       Port Location     : LOCAL
Slot              : 2
       Unit              : 20
       Slot Unit         : 20
```

```
       Acitve            : Y
       SNMP IF Index     : 22
       GPN               : 84
       EC Channel        : 0
       EC Index          : 0
 ASIC              : 0
       ASIC Port         : 14
       Port LE Handle    : 0x514cd990
Non Zero Feature Ref Counts
       FID : 48(AL_FID_L2_PM), Ref Count : 1
       FID : 77(AL_FID_STATS), Ref Count : 1
       FID : 51(AL_FID_L2_MATM), Ref Count : 1
       FID : 13(AL_FID_SC), Ref Count : 1
       FID : 26(AL_FID_QOS), Ref Count : 1
Sub block information
       FID : 48(AL_FID_L2_PM), Private Data &colon; 0x54072618
       FID : 26(AL_FID_QOS), Private Data &colon; 0x514d31b8
```

Die verantwortliche Schnittstelle wird hier identifiziert. In Gig2/0/20 gibt es einen Traffic-Generator, der ARP-Datenverkehr pumpt. Wenn Sie diese herunterfahren, wird das Problem behoben und die CPU-Auslastung minimiert.

Schritt 4: Verwenden der FED-Ablaufverfolgung

Der einzige Nachteil der im letzten Abschnitt beschriebenen Methode besteht darin, dass nur das letzte Paket, das in die Senke gelangt, ausgegeben wird und nicht der Schuldige sein kann.

Eine bessere Möglichkeit zur Fehlerbehebung wäre die Verwendung einer Funktion namens FED-Tracing. Die Ablaufverfolgung ist eine Paketerfassungsmethode (unter Verwendung verschiedener Filter), die von der FED an die CPU übermittelt wird. Die FED-Ablaufverfolgung ist jedoch nicht so einfach wie die Netdr-Funktion auf dem Cisco Catalyst Switch der Serie 6500.

Hier wird der Prozess in Schritte unterteilt:

1. Aktivieren Sie die Detailverfolgung. Standardmäßig ist die Ereignisablaufverfolgung aktiviert. Sie müssen die detaillierte Ablaufverfolgung aktivieren, um die tatsächlichen Pakete zu erfassen:

   ```
   3850-2#set trace control fed-punject-detail enable
   ```

2. Feinabstimmung des Erfassungspuffers Bestimmen Sie, wie tief Ihre Puffer für die detaillierte Nachverfolgung sind, und erhöhen Sie sie nach Bedarf.

   ```
   3850-2#show mgmt-infra trace settings fed-punject-detail
   One shot Trace Settings:

   Buffer Name: fed-punject-detail
   Default Size: 32768
   Current Size: 32768
   Traces Dropped due to internal error: No
   Total Entries Written: 0
   One shot mode: No
   One shot and full: No
   Disabled: False
   ```

Sie können die Puffergröße mit dem folgenden Befehl ändern:

```
3850-2#set trace control fed-punject-detail buffer-size
```

Folgende Werte stehen Ihnen zur Verfügung:

```
3850-2#set trace control fed-punject-detail buffer-size ?
 <8192-67108864>  The new desired buffer size, in bytes
 default          Reset trace buffer size to default
```

3. Fügen Sie Erfassungsfilter hinzu. Sie müssen nun verschiedene Filter für die Erfassung hinzufügen. Sie können verschiedene Filter hinzufügen und entweder alle oder einen beliebigen Filter für die Erfassung zuordnen.

   Mit diesem Befehl werden Filter hinzugefügt:

   ```
   3850-2#set trace fed-punject-detail direction rx filter_add
   ```

   Diese Optionen sind derzeit verfügbar:

   ```
   3850-2#set trace fed-punject-detail direction rx filter_add ?
    cpu-queue  rxq 0..31
    field      field
    offset     offset
   ```

   Jetzt müssen Sie die Dinge miteinander verknüpfen. Erinnern Sie sich an die Warteschlange, die in Schritt 2 dieses Fehlerbehebungsprozesses identifiziert wurde? Da die Warteschlange 16 eine große Anzahl von Paketen an die CPU weiterleitet, ist es sinnvoll, diese Warteschlange zu verfolgen und festzustellen, welche Pakete von ihr an die CPU weitergeleitet werden.

   Mit dem folgenden Befehl können Sie jede Warteschlange verfolgen:

   ```
    3850-2#set trace fed-punject-detail direction rx filter_add cpu-queue
   ```

   Hier ist der Befehl für dieses Beispiel:

   ```
   3850-2#set trace fed-punject-detail direction rx filter_add cpu-queue 16 16
   ```

   Sie müssen entweder eine Übereinstimmung für alle oder eine Übereinstimmung für Ihre Filter auswählen und dann die Ablaufverfolgung aktivieren:

```
3850-2#set trace fed-punject-detail direction rx match_all
3850-2#set trace fed-punject-detail direction rx filter_enable
```

4. Gefilterte Pakete anzeigen. Sie können die erfassten Pakete mit dem Befehl **show mgmt-infra trace messages feed-punject-detail** anzeigen.

```
3850-2#show mgmt-infra trace messages fed-punject-detail
[11/25/13 07:05:53.814 UTC 2eb0c9 5661]
00 00 00 00 00 4e 00 40 07 00 02 08 00 00 51 3b
00 00 00 00 00 01 00 00 03 00 00 00 00 00 00 01
00 00 00 00 20 00 00 0e 00 00 00 00 00 01 00 74
00 00 00 04 00 54 41 02 00 00 00 00 00 00 00 00

[11/25/13 07:05:53.814 UTC 2eb0ca 5661]
ff ff ff ff ff ff aa bb cc dd 00 00 08 06 00 01
08 00 06 04 00 01 aa bb cc dd 00 00 c0 a8 01 0a
ff ff ff ff ff ff c0 a8 01 14 00 01 02 03 04 05
06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 f6 b9 10 32
[11/25/13 07:05:53.814 UTC 2eb0cb 5661] Frame descriptors:
[11/25/13 07:05:53.814 UTC 2eb0cc 5661]
=========
fdFormat=0x4    systemTtl=0xe
loadBalHash1=0x8        loadBalHash2=0x8
spanSessionMap=0x0      forwardingMode=0x0
destModIndex=0x0        skipIdIndex=0x4
srcGpn=0x54     qosLabel=0x41
srcCos=0x0      ingressTranslatedVlan=0x3
bpdu=0x0        spanHistory=0x0
sgt=0x0 fpeFirstHeaderType=0x0
srcVlan=0x1     rcpServiceId=0x2
wccpSkip=0x0    srcPortLeIndex=0xe
cryptoProtocol=0x0      debugTagId=0x0
vrfId=0x0       saIndex=0x0
pendingAfdLabel=0x0     destClient=0x1
appId=0x0       finalStationIndex=0x74
decryptSuccess=0x0      encryptSuccess=0x0
rcpMiscResults=0x0      stackedFdPresent=0x0
spanDirection=0x0       egressRedirect=0x0
redirectIndex=0x0       exceptionLabel=0x0
destGpn=0x0     inlineFd=0x0
suppressRefPtrUpdate=0x0        suppressRewriteSideEfects=0x0
cmi2=0x0        currentRi=0x1
currentDi=0x513b        dropIpUnreachable=0x0
srcZoneId=0x0   srcAsicId=0x0
originalDi=0x0  originalRi=0x0
srcL3IfIndex=0x2        dstL3IfIndex=0x0
dstVlan=0x0     frameLength=0x40
fdCrc=0x7       tunnelSpokeId=0x0

=========
[11/25/13 07:05:53.814 UTC 2eb0cd 5661]
[11/25/13 07:05:53.814 UTC 2eb0ce 5661] PUNT PATH (fed_punject_rx_process_packet:
830):RX: Q: 16, Tag: 65561
[11/25/13 07:05:53.814 UTC 2eb0cf 5661] PUNT PATH (fed_punject_get_physical_iif:
579):RX: Physical IIF-id 0x104d88000000033
[11/25/13 07:05:53.814 UTC 2eb0d0 5661] PUNT PATH (fed_punject_get_src_l3if_index:
434):RX: L3 IIF-id 0x101b6800000004f
```

```
[11/25/13 07:05:53.814 UTC 2eb0d1 5661] PUNT PATH (fed_punject_fd_2_pds_md:478):
RX: l2_logical_if = 0x0
[11/25/13 07:05:53.814 UTC 2eb0d2 5661] PUNT PATH (fed_punject_get_source_cos:638):
RX: Source Cos 0
[11/25/13 07:05:53.814 UTC 2eb0d3 5661] PUNT PATH (fed_punject_get_vrf_id:653):
RX: VRF-id 0
[11/25/13 07:05:53.814 UTC 2eb0d4 5661] PUNT PATH (fed_punject_get_src_zoneid:667):
RX: Zone-id 0
[11/25/13 07:05:53.814 UTC 2eb0d5 5661] PUNT PATH (fed_punject_fd_2_pds_md:518):
RX: get_src_zoneid failed
[11/25/13 07:05:53.814 UTC 2eb0d6 5661] PUNT PATH (fed_punject_get_acl_log_direction:
695): RX: : Invalid CMI2
[11/25/13 07:05:53.814 UTC 2eb0d7 5661] PUNT PATH (fed_punject_fd_2_pds_md:541):RX:
get_acl_log_direction failed
[11/25/13 07:05:53.814 UTC 2eb0d8 5661] PUNT PATH (fed_punject_get_acl_full_direction:
724):RX: DI 0x513b ACL Full Direction 1
[11/25/13 07:05:53.814 UTC 2eb0d9 5661] PUNT PATH (fed_punject_get_source_sgt:446):
RX: Source SGT 0
[11/25/13 07:05:53.814 UTC 2eb0da 5661] PUNT PATH (fed_punject_get_first_header_type:680):
RX: FirstHeaderType 0
[11/25/13 07:05:53.814 UTC 2eb0db 5661] PUNT PATH (fed_punject_rx_process_packet:916):
RX: fed_punject_pds_send packet 0x1f00 to IOSd with tag 65561
[11/25/13 07:05:53.814 UTC 2eb0dc 5661] PUNT PATH (fed_punject_rx_process_packet:744):
RX: **** RX packet 0x2360 on qn 16, len 128 ****
[11/25/13 07:05:53.814 UTC 2eb0dd 5661]
buf_no 0 buf_len 128

<snip>
```

Diese Ausgabe enthält viele Informationen und kann in der Regel ausreichen, um festzustellen, woher die Pakete stammen und was in ihnen enthalten ist.

Der erste Teil des Header Dump sind wiederum die Meta-Daten, die vom System verwendet werden. Der zweite Teil ist das eigentliche Paket.

```
ff ff ff ff ff ff - destination MAC address
aa bb cc dd 00 00 - source MAC address
```

Sie können diese Quell-MAC-Adresse verfolgen, um den zuständigen Port zu ermitteln (sobald Sie festgestellt haben, dass dies die Mehrzahl der Pakete ist, die aus der Warteschlange 16 ausgelesen werden; diese Ausgabe zeigt nur eine Instanz des Pakets an und die anderen Ausgaben/Pakete werden ausgeschnitten).

Es gibt jedoch einen besseren Weg. Beachten Sie, dass Protokolle, die nach den Kopfzeileninformationen vorhanden sind:

```
[11/25/13 07:05:53.814 UTC 2eb0ce 5661] PUNT PATH (fed_punject_rx_process_packet:
830):RX: Q: 16, Tag: 65561
[11/25/13 07:05:53.814 UTC 2eb0cf 5661] PUNT PATH (fed_punject_get_physical_iif:
579):RX: Physical IIF-id 0x104d88000000033
```

Das erste Protokoll zeigt deutlich an, aus welcher Warteschlange und mit welchem Tag

dieses Paket stammt. Wenn Sie die Warteschlange nicht früher kennen, können Sie auf diese Weise leicht feststellen, um welche Warteschlange es sich handelt.

Das zweite Protokoll ist sogar noch nützlicher, da es die physische Interface ID Factory (IIF)-ID für die Quellschnittstelle bereitstellt. Der Hexadezimalwert ist ein Handle, mit dem Informationen über diesen Port ausgelesen werden können:

```
3850-2#show platform port-asic ifm iif-id 0x0104d88000000033
Interface Table
Interface IIF-ID        : 0x0104d88000000033
Interface Name          : Gi2/0/20
Interface Block Pointer : 0x514d2f70
Interface State         : READY
Interface Stauts        : IFM-ADD-RCVD, FFM-ADD-RCVD
Interface Ref-Cnt       : 6
Interface Epoch         : 0
Interface Type          : ETHER
       Port Type         : SWITCH PORT
       Port Location     : LOCAL
Slot              : 2
       Unit              : 20
       Slot Unit         : 20
       Active           : Y
       SNMP IF Index     : 22
       GPN               : 84
       EC Channel        : 0
       EC Index          : 0
 ASIC              : 0
       ASIC Port         : 14
       Port LE Handle    : 0x514cd990
Non Zero Feature Ref Counts
       FID : 48(AL_FID_L2_PM), Ref Count : 1
       FID : 77(AL_FID_STATS), Ref Count : 1
       FID : 51(AL_FID_L2_MATM), Ref Count : 1
       FID : 13(AL_FID_SC), Ref Count : 1
       FID : 26(AL_FID_QOS), Ref Count : 1
Sub block information
       FID : 48(AL_FID_L2_PM), Private Data &colon; 0x54072618
       FID : 26(AL_FID_QOS), Private Data &colon; 0x514d31b8
```

Sie haben erneut die Quellschnittstelle und den Schuldigen identifiziert.

Tracing ist ein leistungsstarkes Tool, das zur Behebung von Problemen mit hoher CPU-Auslastung unerlässlich ist und umfangreiche Informationen bereitstellt, um eine solche Situation erfolgreich zu lösen.

EEM-Beispielskript (Embedded Event Manager) für Cisco Catalyst Switches der Serie 3850

Verwenden Sie diesen Befehl, um ein Protokoll zu erstellen, das bei einem bestimmten Schwellenwert generiert wird:

```
process cpu threshold type total rising interval
switch
```

Das mit dem Befehl generierte Protokoll sieht wie folgt aus:

```
*Jan 13 00:03:00.271: %CPUMEM-5-RISING_THRESHOLD: 1 CPUMEMd[6300]:  Threshold: :
50, Total CPU Utilzation(total/Intr) :50/0, Top 3 processes(Pid/Util) : 8622/25,
5753/12, 9663/0
```

Das generierte Protokoll liefert folgende Informationen:

- Die gesamte CPU-Auslastung zum Zeitpunkt des Triggers. Dies wird in diesem Beispiel durch die Gesamt-CPU-Auslastung (gesamt/Intern) :50/0 gekennzeichnet.
- Top Prozesse - diese sind im Format PID/CPU% aufgelistet. In diesem Beispiel sind dies:

```
8622/25 - 8622 is PID for IOSd and 25 implies that this process is using 25% CPU.
5753/12 - 5733 is PID for FED and 12 implies that this process is using 12% CPU.
```

Das EEM-Skript wird hier angezeigt:

```
event manager applet highcpu
event syslog pattern "%CPUMEM-5-RISING_THRESHOLD"
 action 0.1 syslog msg "high CPU detected"
 action 0.2 cli command "enable"
 action 0.3 cli command "show process cpu sorted | append nvram:<filename>.txt"
 action 0.4 cli command "show process cpu detailed process <process name|process ID>
sorted | nvram:<filename>.txt"
 action 0.5 cli command "show platform punt statistics port-asic 0 cpuq -1
direction rx | append nvram:<filename>.txt"
 action 0.6 cli command "show platform punt statistics port-asic 1 cpuq -1
direction rx | append nvram:<filename>.txt"
 action 0.7 cli command "conf t"
 action 0.8 cli command "no event manager applet highcpu"
```

> **Hinweis**: Der Befehl **process cpu threshold** funktioniert derzeit nicht in der 3.2.x-Folge. Ein weiterer zu beachtender Punkt ist, dass dieser Befehl die durchschnittliche CPU-Auslastung zwischen den vier Kernen untersucht und ein Protokoll generiert, wenn dieser Durchschnitt den im Befehl definierten Prozentsatz erreicht.

Cisco IOS-XE 16.x oder spätere Versionen

Wenn Sie Catalyst 3850-Switches mit Cisco® IOS-XE Softwareversion 16.x oder höher verwenden, finden Sie weitere Informationen unter Fehlerbehebung bei hoher CPU-Auslastung auf Catalyst-Switch-Plattformen mit IOS-XE 16.x.

Zugehörige Informationen

- Was ist Cisco IOS XE?
- Cisco Catalyst Switches der Serie 3850 - Datenblätter und Produktliteratur
- Technischer Support und Dokumentation für Cisco Systeme

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.