Configure and Verify Netflow, AVC, and ETA on Catalyst 9000 Series Switches

Contents

Introduction

This document describes how to configure and validate NetFlow, Application Visibility and Control (AVC), and Encrypted Traffic Analytics (ETA).

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- NetFlow
- AVC
- ETA

Components Used

The information in this document is based on a Catalyst 9300 switch that runs Cisco IOS® XE software 16.12.4.

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.

Related Products

This document can also be used with these hardware and software versions:

- 9200 (Supports NetFlow and AVC only)
- 9300 (Supports NetFlow, AVC, and ETA)
- 9400 (Supports NetFlow, AVC, and ETA)
- 9500 (Supports NetFlow and AVC only)
- 9600 (Supports NetFlow and AVC only)
- Cisco IOS XE 16.12 and later

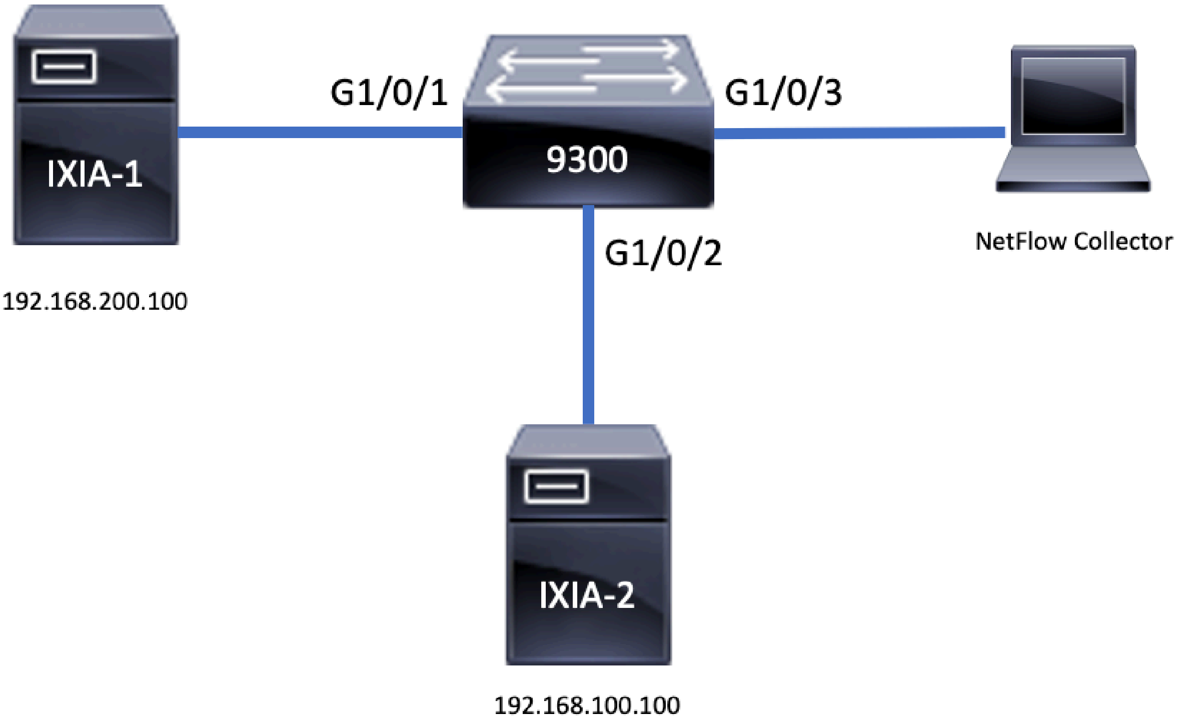
Background Information

- Flexible NetFlow is the next-generation in flow technology that collects and measures data to allow all routers or switches in the network to become a source of telemetry.
- Flexible NetFlow allows extremely granular and accurate traffic measurements and high-level aggregated traffic collection.
- Flexible NetFlow uses flows to provide statistics for accounting, network monitoring, and network

planning.
- A flow is a unidirectional stream of packets that arrives on a source interface and has the same values for the keys. A key is an identified value for a field within the packet. You create a flow via a flow record to define the unique keys for your flow.

Note: Platform (fed) commands can vary. Command can be `show platform fed <active|standby>` versus `show platform fed switch <active|standby>` . If the syntax noted in the examples do not parse out, please try the variant.

Network Diagram



Configure

Components

NetFlow configuration is comprised of **three main components** that can be used together is several variations to perform traffic analysis and data export.

Flow Record

- A record is a combination of key and nonkey fields. Flexible NetFlow records are assigned to Flexible NetFlow flow monitors to define the cache that is used for storage of flow data.
- Flexible NetFlow includes several predefined records that can be used to monitor traffic.
- Flexible NetFlow also allows custom records to be defined for a Flexible NetFlow flow monitor cache by specification of key and nonkey fields to customize the data collection to your specific requirements.

As shown in the example, flow record configuration details:

```
flow record TAC-RECORD-IN
match flow direction
match ipv4 source address
match interface input
match ipv4 destination address
match ipv4 protocol
collect counter packets long
collect counter bytes long
collect timestamp absolute last
collect transport tcp flags

flow record TAC-RECORD-OUT
match flow direction
match interface output
match ipv4 source address
match ipv4 destination address
match ipv4 protocol
collect counter packets long
collect counter bytes long
collect timestamp absolute last
collect transport tcp flags
```

Flow Exporter

- Flow exporters are used to export the data in the flow monitor cache to a remote system (server that functions as a NetFlow collector), for analysis and storage.
- Flow exporters are assigned to flow monitors to provide data export capability for the flow monitors.

As shown in the example, flow exporter configuration details:

```
flow exporter TAC-EXPORT
destination 192.168.69.2
source Vlan69
```

Flow Monitor

- Flow monitors are the Flexible NetFlow component that is applied to interfaces to perform network traffic monitoring.
- Flow data is collected from the network traffic and added to the flow monitor cache while the process runs. The process is based on the key and nonkey fields in the flow record.

As shown in the example, flow monitor configuration details:

```
flow monitor TAC-MONITOR-IN
exporter TAC-EXPORT
record TAC-RECORD-IN
```

```
flow monitor TAC-MONITOR-OUT
exporter TAC-EXPORT
record TAC-RECORD-OUT

Switch#show run int g1/0/1
Building configuration...

Current configuration : 185 bytes
!
interface GigabitEthernet1/0/1
switchport access vlan 42
switchport mode access
ip flow monitor TAC-MONITOR-IN input
ip flow monitor TAC-MONITOR-OUT output
load-interval 30
end
```

Flow Sampler (Optional)

- Flow samplers are created as separate components in a router’s configuration.
- Flow samplers limit the number of packets that are selected for analysis to reduce the load on the device that uses Flexible NetFlow.
- Flow samplers are used to reduce the load on the device that uses Flexible NetFlow achieved through the limit of the number of packets that are selected for analysis.
- Flow samplers exchange accuracy for router performance. If there is a reduction in the number of packets that are analyzed by the flow monitor, accuracy of the information stored in the flow moniitor cache can be impacted.

As shown in the example, example flow sampler configuration:

<#root>

```
sampler SAMPLE-TAC
description Sample at 50%
mode random 1 out-of 2

Switch(config)#

interface GigabitEthernet1/0/1


Switch(config-if)#

ip flow monitor TAC-MONITOR-IN sampler SAMPLE-TAC input


Switch(config-if)#

end
```

Restrictions

- DNA Addon license is required for full Flexible NetFlow, otherwise Sampled NetFlow is only available.
- Flow-exporters cannot use the management port as a source.

This is not an inclusive list, consult the configuration guide for the appropriate platform and code.

Verify

Platform Independent Verification

Verify the configuration and confirm that the required NetFlow components are present:

1. **Flow Record**
2. **Flow Exporter**
3. **Flow Monitor**
4. **Flow Sampler (Optional)**

Tip: To view the flow record, flow exporter, and flow monitor output in one command, run show running-config flow monitor <flow monitor name> expand .

As shown in the example, the flow monitor tied to the input direction and its associated components:

<#root>

```
Switch#

show running-config flow monitor TAC-MONITOR-IN expand

Current configuration:
!

flow record TAC-RECORD-IN

 match ipv4 protocol
 match ipv4 source address
 match ipv4 destination address
 match interface input
 match flow direction
 collect transport tcp flags
 collect counter bytes long
 collect counter packets long
 collect timestamp absolute last
!

flow exporter TAC-EXPORT

 destination 192.168.69.2
 source Vlan69
!

flow monitor TAC-MONITOR-IN

 exporter TAC-EXPORT
 record TAC-RECORD-IN
!
```

As shown in the example, the flow monitor tied to the output direction and its associated components:

<#root>

```
Switch#

show run flow monitor TAC-MONITOR-OUT expand

Current configuration:
!

flow record TAC-RECORD-OUT

 match ipv4 protocol
 match ipv4 source address
 match ipv4 destination address
 match interface output
 match flow direction
 collect transport tcp flags
 collect counter bytes long
 collect counter packets long
 collect timestamp absolute last
!

flow exporter TAC-EXPORT

 destination 192.168.69.2
 source Vlan69
!

flow monitor TAC-MONITOR-OUT

 exporter TAC-EXPORT
 record TAC-RECORD-OUT
!
```

Run the command show flow monitor <flow monitor name> statistics**.** This output is helpful to confirm that data is recorded:

<#root>

```
Switch#

show flow monitor TAC-MONITOR-IN statistics

  Cache type:                                Normal (Platform cache)
  Cache size:                                 10000
  Current entries:                                1

  Flows added:                                    1
  Flows aged:                                     0
```

Run the command show flow monitor <flow monitor name> cacheto confirm that the NetFlow cache has output:

<#root>

```
Switch#

show flow monitor TAC-MONITOR-IN cache

  Cache type:                                Normal (Platform cache)
  Cache size:                                 10000
  Current entries:                                1
  Flows added:                                  1
  Flows aged:                                   0

IPV4 SOURCE ADDRESS:       192.168.200.100
IPV4 DESTINATION ADDRESS:  192.168.100.100
INTERFACE INPUT:           Gi1/0/1
FLOW DIRECTION:            Input
IP PROTOCOL:               17
tcp flags:                 0x00
counter bytes long:        4606617470
counter packets long:      25311085
timestamp abs last:        22:44:48.579
```

Run the command show flow exporter <exporter name> statisticsto confirm that the exporter sent packets:

<#root>

```
Switch#

show flow exporter TAC-EXPORT statistics

Flow Exporter TAC-EXPORT:
  Packet send statistics (last cleared 00:08:38 ago):
    Successfully sent:         2                     (24 bytes)

  Client send statistics:
    Client: Flow Monitor TAC-MONITOR-IN
      Records added:           0
      Bytes added:             12
        - sent:                12

    Client: Flow Monitor TAC-MONITOR-OUT
      Records added:           0
      Bytes added:             12
        - sent:                12
```

Platform Dependent Verification

NetFlow Initialization - NFL Partition Table

- NetFlow partitions are initialized for different features with 16 partitions per direction (Input vs Output).
- NetFlow partition table configuration is divided into the global bank allocation, which is further subdivided into the ingress and egress flow banks.

Key Fields

- Number of partitions
- Partition enable status
- Partition limit
- Current partition usage

To view the NetFlow Partition Table, you can run the command show platform software fed switch active|standby|member|

fnf sw-table-sizes asic <asic number> shadow 0 .

Note: Flows that are created are specific to the switch and asic core when they are created. The switch number (active, standby, etc) needs to specified accordingly. The ASIC number that is input is tied to the respective interface, use show platform software fed switch active|standby|member ifm mappings to determine ASIC that corresponds to the interface. For the shadow option, always use "0".

<#root>

```
Switch#

show platform software fed switch active fnf sw-table-sizes asic 0 shadow 0


---------------------------------
     Global Bank Allocation
---------------------------------
Ingress Banks : Bank 0 Bank 1
Egress Banks  : Bank 2 Bank 3
---------------------------------


Global flow table Info



<--- Provides the number of entries used per direction

INGRESS    usedBankEntry           0  usedOvfTcamEntry     0
 EGRESS    usedBankEntry           0  usedOvfTcamEntry     0
---------------------------------
    Flows Statistics
INGRESS    TotalSeen=0 MaxEntries=0 MaxOverflow=0
EGRESS     TotalSeen=0 MaxEntries=0 MaxOverflow=0

---------------------------------
       Partition Table
---------------------------------
## Dir  Limit  CurrFlowCount  OverFlowCount  MonitoringEnabled
 0 ING      0          0               0           0


1 ING  16640          0               0           1          <-- Current flow count in hardware

 2 ING      0          0               0           0
 3 ING  16640          0               0           0
 4 ING      0          0               0           0
 5 ING   8192          0               0           1
 6 ING      0          0               0           0
 7 ING      0          0               0           0
 8 ING      0          0               0           0
 9 ING      0          0               0           0
10 ING      0          0               0           0
11 ING      0          0               0           0
12 ING      0          0               0           0
13 ING      0          0               0           0
14 ING      0          0               0           0
15 ING      0          0               0           0
 0 EGR      0          0               0           0


1 EGR  16640          0               0           1          <-- Current flow count in hardware
```

```
 2  EGR      0           0               0           0
 3  EGR  16640           0               0           0
 4  EGR      0           0               0           0
 5  EGR   8192           0               0           1
 6  EGR      0           0               0           0
 7  EGR      0           0               0           0
 8  EGR      0           0               0           0
 9  EGR      0           0               0           0
10  EGR      0           0               0           0
11  EGR      0           0               0           0
12  EGR      0           0               0           0
13  EGR      0           0               0           0
14  EGR      0           0               0           0
15  EGR      0           0               0           0
```

Flow Monitor

Flow monitor configuration includes the following:

1. NetFlow ACL Configuration, which results in creation of an entry within the ACL TCAM table.

The ACL TCAM entry is comprised of:

- Lookup matching keys
- Result parameters used for NetFlow lookup, which includes the following:
  - Profile ID
  - NetFlow ID

2. Flow Mask Configuration, which results in creation of an entry in NflLookupTable and NflFlowMaskTable.

- Indexed by NetFlow ACL result parameters to find the flow mask for netflow lookup

NetFlow ACL

To view the NetFlow ACL configuration run the command `show platform hardware fed switch active fwd-asic resource tcam table nfl_acl asic <asic number>` .

Tip: If there is a Port ACL (PACL), the entry gets created on the ASIC where the interface is mapped to. In the case of a Router ACL (RACL), the entry is present on all ASIC(s).

- In this output there are NFCMD0 and NFCMD1, that are 4 bit values. In order to calculate the Profile ID, convert the values into binary.
- In this output, NFCMD0 is 1, NFCMD1 is 2. When converted to binary: 000100010
- In Cisco IOS XE 16.12 and onwards within the combined 8 bits, the first 4 bits is the profile ID, and the 7th bit indicates that the lookup is enabled. So in the example, **0001**0010, the profile ID is 1.
- In Cisco IOS XE 16.11 and older versions of code, within the combined 8 bits, the first 6 bits is the profile ID, and the 7th bit indicates that the lookup is enabled. So in this example, **000100**10, the profile ID is 4.

<#root>

```
Switch#

show platform hardware fed switch active fwd-asic resource tcam table nfl_acl asic 0

Printing entries for region INGRESS_NFL_ACL_CONTROL (308) type 6 asic 0
========================================================
Printing entries for region INGRESS_NFL_ACL_GACL (309) type 6 asic 0
========================================================
Printing entries for region INGRESS_NFL_ACL_PACL (310) type 6 asic 0
========================================================
TAQ-2 Index-32 (A:0,C:0) Valid StartF-1 StartA-1 SkipF-0 SkipA-0
Input IPv4 NFL PACL

Labels  Port   Vlan   L3If   Group
M:      00ff   0000   0000   0000
V:      0001   0000   0000   0000

   vcuResults l3Len l3Pro l3Tos SrcAddr  DstAddr  mtrid vrfid  SH
M:  00000000   0000  00    00  00000000 00000000  00   0000   0000
V:  00000000   0000  00    00  00000000 00000000  00   0000   0000

   RMAC RA MEn IPOPT MF NFF DF SO DPT TM DSEn  l3m
M:  0    0  0    0    0  0   0  0  0   0  0  0
V:  0    0  0    0    0  0   0  0  0   0  0  0

   SrcPort DstPortIITypeCode TCPFlags TTL ISBM QosLabel ReQOS S_P2P D_P2P
M:  0000    0000              00    00    0000    00      0     0     0
V:  0000    0000              00    00    0000    00      0     0     0

   SgEn SgLabel    AuthBehaviorTag  l2srcMiss  l2dstMiss ipTtl SgaclDeny
M:  0    000000    0  0   0 0     0
V:  0    000000    0  0   0 0     0


NFCMD0 NFCMD1 SMPLR LKP1 LKP2 PID QOSPRI MQLBL MPLPRO LUT0PRI CPUCOPY
     1      2     0    1    0   0      0     0      0 0x0000f       0

Start/Skip Word: 0x00000003
Start Feature, Terminate

-----------------------------------------
Printing entries for region INGRESS_NFL_ACL_VACL (311) type 6 asic 0
========================================================
Printing entries for region INGRESS_NFL_ACL_RACL (312) type 6 asic 0
========================================================
Printing entries for region INGRESS_NFL_ACL_SSID (313) type 6 asic 0
========================================================
Printing entries for region INGRESS_NFL_CATCHALL (314) type 6 asic 0
========================================================
TAQ-2 Index-224 (A:0,C:0) Valid StartF-1 StartA-1 SkipF-0 SkipA-0
Input IPv4 NFL RACL

Labels  Port   Vlan   L3If   Group
M:      0000   0000   0000   0000
V:      0000   0000   0000   0000

   vcuResults l3Len l3Pro l3Tos SrcAddr  DstAddr  mtrid vrfid  SH
M:  00000000   0000  00    00  00000000 00000000  00   0000   0000
V:  00000000   0000  00    00  00000000 00000000  00   0000   0000

   RMAC RA MEn IPOPT MF NFF DF SO DPT TM DSEn  l3m
M:  0    0  0    0    0  0   0  0  0   0  0  0
```

```
V:  0    0  0    0    0  0   0  0  0   0  0  0

   SrcPort DstPortIITypeCode TCPFlags TTL ISBM QosLabel ReQOS S_P2P D_P2P
M:  0000    0000              00    00    0000    00       0     0     0
V:  0000    0000              00    00    0000    00       0     0     0

   SgEn SgLabel    AuthBehaviorTag  l2srcMiss  l2dstMiss ipTtl SgaclDeny
M:  0    000000    0  0   0 0     0
V:  0    000000    0  0   0 0     0

NFCMD0 NFCMD1 SMPLR LKP1 LKP2 PID QOSPRI MQLBL MPLPRO LUTOPRI CPUCOPY
     0      0     0    0    0   0      0     0      0 0x00000       0
Start/Skip Word: 0x00000003
Start Feature, Terminate

-----------------------------------------
TAQ-2 Index-225 (A:0,C:0) Valid StartF-0 StartA-0 SkipF-0 SkipA-0
Input IPv4 NFL PACL

Labels  Port   Vlan   L3If   Group
M:      0000   0000   0000   0000
V:      0000   0000   0000   0000

   vcuResults l3Len l3Pro l3Tos SrcAddr  DstAddr  mtrid vrfid  SH
M:  00000000   0000  00    00  00000000 00000000  00   0000   0000
V:  00000000   0000  00    00  00000000 00000000  00   0000   0000

   RMAC RA MEn IPOPT MF NFF DF SO DPT TM DSEn  l3m
M:  0    0  0    0    0  0   0  0  0   0  0  0
V:  0    0  0    0    0  0   0  0  0   0  0  0

   SrcPort DstPortIITypeCode TCPFlags TTL ISBM QosLabel ReQOS S_P2P D_P2P
M:  0000    0000              00    00    0000    00       0     0     0
V:  0000    0000              00    00    0000    00       0     0     0

   SgEn SgLabel    AuthBehaviorTag  l2srcMiss  l2dstMiss ipTtl SgaclDeny
M:  0    000000    0  0   0 0     0
V:  0    000000    0  0   0 0     0

NFCMD0 NFCMD1 SMPLR LKP1 LKP2 PID QOSPRI MQLBL MPLPRO LUTOPRI CPUCOPY
     0      0     0    0    0   0      0     0      0 0x00000       0
Start/Skip Word: 0x00000000
No Start, Terminate

-----------------------------------------
TAQ-2 Index-226 (A:0,C:0) Valid StartF-0 StartA-0 SkipF-0 SkipA-0
Input IPv6 NFL PACL

Labels  Port   Vlan   L3If   Group
Mask    0x0000 0x0000 0x0000 0x0000
Value   0x0000 0x0000 0x0000 0x0000

vcuResult dstAddr0 dstAddr1 dstAddr2 dstAddr3 srcAddr0
00000000  00000000 00000000 00000000 00000000 00000000
00000000  00000000 00000000 00000000 00000000 00000000

srcAddr1 srcAddr2 srcAddr3 TC HL l3Len fLabel vrfId toUs
00000000 00000000 00000000 00 00  0000  00000  000   0
00000000 00000000 00000000 00 00  0000  00000  000   0

l3Pro mtrId AE FE RE HE MF NFF SO IPOPT RA MEn RMAC DPT TMP l3m
 00    00   0  0  0  0  0   0  0    0   0   0   0    0   0    0
 00    00   0  0  0  0  0   0  0    0   0   0   0    0   0    0

DSE srcPort dstPortIITypeCode tcpFlags IIPresent cZId dstZId
 0   0000    0000      00      00     00     00
 0   0000    0000      00      00     00     00

v6RT AH ESP mREn ReQOS QosLabel PRole VRole  AuthBehaviorTag
M: 0   0   0    0    0      00         0     0 0
V: 0   0   0    0    0      00         0     0 0

   SgEn SgLabel
M:  0    000000
V:  0    000000

NFCMD0 NFCMD1 SMPLR LKP1 LKP2 PID QOSPRI MQLBL MPLPRO LUTOPRI CPUCOPY
     0      0     0    0    0   0      0     0      0 0x00000       0
Start/Skip Word: 0x00000000
No Start, Terminate

-----------------------------------------
TAQ-2 Index-228 (A:0,C:0) Valid StartF-0 StartA-0 SkipF-0 SkipA-0
conversion to string vmr l2p not supported
-----------------------------------------
TAQ-2 Index-230 (A:0,C:0) Valid StartF-0 StartA-0 SkipF-0 SkipA-0
Input MAC NFL PACL

Labels  Port   Vlan   L3If   Group
M:      0000   0000   0000   0000
V:      0000   0000   0000   0000

   arpSrcHwAddr  arpDestHwAddr  arpSrcIpAddr  arpTargetIp  arpOperation
M: 000000000000  000000000000   00000000       00000000     0000
V: 000000000000  000000000000   00000000       00000000     0000


   TRUST  SNOOP  SVALID  DVALID
M:    0     0      0      0
V:    0     0      0      0


   arpHardwareLength  arpHardwareType  arpProtocolLength  arpProtocolType
M:   00000000           00000000             00000000           00000000
V:   00000000           00000000             00000000           00000000


   VlanId l2Encap l2Protocol cosCFI   srcMAC      dstMAC    ISBM  QosLabel
M:  000    0        0000     0     000000000000 000000000000 00     00
V:  000    0        0000     0     000000000000 000000000000 00     00


  ReQOS isSnap  isLLC  AuthBehaviorTag
M:  0     0      0       0
V:  0     0      0        0

NFCMD0 NFCMD1 SMPLR LKP1 LKP2 PID QOSPRI MQLBL MPLPRO LUTOPRI CPUCOPY
     0      0     0    0    0   0      0     0      0 0x00000       0
Start/Skip Word: 0x00000000
No Start, Terminate

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

Flow Mask

Run the command show platform software fed switch active|standby|member fnf fmask-entry asic <asic number> entry 1 to view that the flow mask is installed in hardware. The number of list of key fields can also be found here.

<#root>

```
Switch#

show platform software fed switch active fnf fmask-entry asic 1 entry 1


---------------------------------
mask0_valid : 1
Mask hdl0   : 1
Profile ID  : 0
Feature 0   : 148
Fmsk0 RefCnt: 1
Mask M1     :
[511:256] => :00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
[255:000] => :FFFFFFFF 00000000 FFFFFFFF 03FF0000 00000000 00FF0000 00000000 C00000FF

Mask M2     :

Key Map     :

Source   Field-Id   Size   NumPFields   Pfields
 002       090       04        01       (0 1 1 1)
 002       091       04        01       (0 1 1 0)
 002       000       01        01       (0 1 0 7)
 000       056       08        01       (0 0 2 4)
 001       011       11        04       (0 0 0 1) (0 0 0 0) (0 1 0 6) (0 0 2 0)
 000       067       32        01       (0 1 12 0)
 000       068       32        01       (0 1 12 2)
```

Flow Stats and Timestamp Offload Data

Run the command show platform software fed switch active fnf flow-record asic <asic number> start-index <index number> num-flows <number of flows> to view netflow statistics as well as timestamps

<#root>

```
Switch#

show platform software fed switch active fnf flow-record asic 1 start-index 1 num-flows 1

1 flows starting at 1 for asic 1:-------------------------------------------------
Idx 996 :
{90, ALR_INGRESS_NET_FLOW_ACL_LOOKUP_TYPE1 = 0x01}
{91, ALR_INGRESS_NET_FLOW_ACL_LOOKUP_TYPE2 = 0x01}
{0, ALR_INGRESS_NFL_SPECIAL1 = 0x00}
{56, PHF_INGRESS_L3_PROTOCOL = 0x11}
{11 PAD-UNK = 0x0000}
{67, PHF_INGRESS_IPV4_DEST_ADDRESS = 0xc0a86464}
{68, PHF_INGRESS_IPV4_SRC_ADDRESS = 0xc0a8c864}

FirstSeen = 0x4b2f, LastSeen = 0x4c59, sysUptime = 0x4c9d
```

```
PKT Count = 0x000000000102d5df, L2ByteCount = 0x00000000ca371638
```

```
Switch#
```

```
show platform software fed switch active fnf flow-record asic 1 start-index 1 num-flows 1
```

```
1 flows starting at 1 for asic 1:-------------------------------------------------
Idx 996 :
{90, ALR_INGRESS_NET_FLOW_ACL_LOOKUP_TYPE1 = 0x01}
{91, ALR_INGRESS_NET_FLOW_ACL_LOOKUP_TYPE2 = 0x01}
{0, ALR_INGRESS_NFL_SPECIAL1 = 0x00}
{56, PHF_INGRESS_L3_PROTOCOL = 0x11}
{11 PAD-UNK = 0x0000}
{67, PHF_INGRESS_IPV4_DEST_ADDRESS = 0xc0a86464}
{68, PHF_INGRESS_IPV4_SRC_ADDRESS = 0xc0a8c864}
```

```
FirstSeen = 0x4b2f, LastSeen = 0x4c5b, sysUptime = 0x4c9f
PKT Count = 0x0000000001050682, L2ByteCount = 0x00000000cbed1590
```

Application Visibility and Control (AVC)

Background Information

- Application Visibility and Control (AVC) is a solution that leverages Network-Based Recognition Version 2 (**NBAR2**), **NetFlow V9**, and various report and management tools (**Cisco Prime**) to help classify applications via deep packet inspection (DPI).
- AVC can be configured on wired access ports for standalone switches or switch stacks.
- AVC can also be used on Cisco wireless controllers to identify applications based on DPI and then mark it with a specific DSCP value. It can also collect various wireless performance metrics such as bandwidth usage in terms of applications and clients.

Performance and Scale

Performance: Each switch member is able to handle 500 connections per second (CPS) at less than 50% CPU utilization. Beyond this rate, AVC service is not guaranteed.

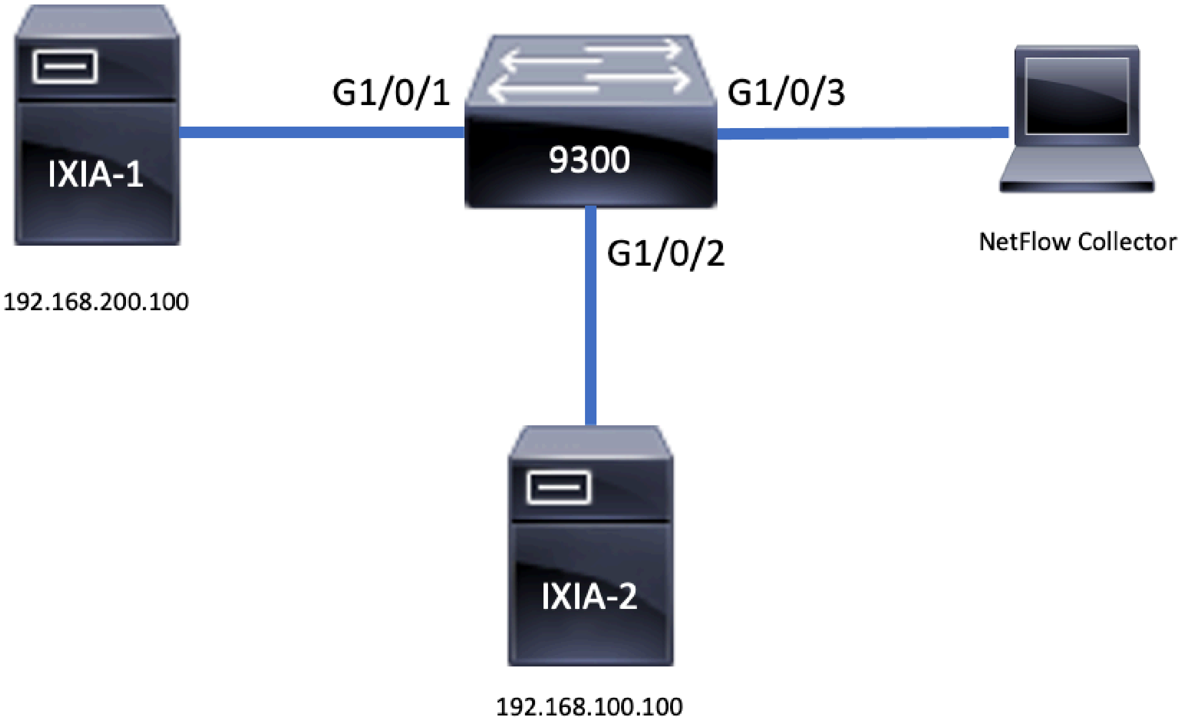
Scale: Ability to handle up to 5000 bi-directional flows per 24 access ports (approximately 200 flows per access port).

Wired AVC Restrictions

- AVC and Encrypted Traffic Analytics (ETA) cannot be configured together at the same time on the same interface.
- Packet classification is only supported for unicast IPv4 (TCP/UDP) traffic.
- NBAR based QoS policy configuration is only supported on wired physical ports. This includes Layer 2 access and trunk ports and Layer 3 routed ports.
- NBAR based QoS policy configuration is not supported on port-channel members, Switch Virtual Interfaces (SVIs) or sub-interfaces.
- NBAR2 based classifiers (**match protocol**), only support QoS actions of marking and policing.
- "Match protocol" is limited to 255 different protocols in all policies (8 bit hardware limitation)

Note: This is not an exhaustive list of all restrictions, consult the appropriate AVC configuration guide for your platform and version of code.

Network Diagram



Components

AVC configuration is comprised of **three main components**that make up the solution:

Visibility: Protocol Discovery

- Protocol discovery is achieved through NBAR, which provides per interface, direction and application bytes/packets statistics.
- Protocol discovery is enabled for a specific interface through the interface configuration: ip nbar protocol-discovery.

As shown in the output, how to enable protocol discovery:

<#root>

```
Switch(config)#

interface fi4/0/5


Switch(config-if)#

ip nbar protocol-discovery


Switch(config-if)#exit
```

```
Switch#

show run int fi4/0/5


Building configuration...

Current configuration : 70 bytes
!
interface FiveGigabitEthernet4/0/5
ip nbar protocol-discovery
end
```

Control: Application Based QoS

When compared to traditional QoS which matches on IP address and UDP/TCP port, AVC achieves finer control through application based QoS, which allows you to match on application, and provides more granular control through QoS actions such as marking and policing.

- Actions are performed on aggregated traffic (not per-flow).
- Application Based QoS is achieved by creation of a class map, match of a protocol, and then creation of a policy map.
- The Application Based QoS policy is attached to an interface.

As shown in the output, example configuration for application based QoS:

<#root>

```
Switch(config)#

class-map WEBEX


Switch(config-cmap)#

match protocol webex-media


Switch(config)#

end



Switch(config)#

policy-map WEBEX


Switch(config-pmap)#

class WEBEX


Switch(config-pmap-c)#

set dscp af41


Switch(config)#

end
```

```
Switch(config)#

interface fi4/0/5


Switch(config-if)#

service-policy input WEBEX


Switch(config)#

end



Switch#

show run int fi4/0/5


Building configuration...

Current configuration : 98 bytes
!
interface FiveGigabitEthernet4/0/5
service-policy input WEBEX
ip nbar protocol-discovery
end
```

Application-Based Flexible NetFlow

Wired AVC FNF supports two types of predefined flow records: **legacy bidirectional flow records** and the new **directional flow records**.

Bidirectional flow records keep track of client/server application statistics.

As shown in the output, example configuration of a bidirectional flow record.

<#root>

```
Switch(config)#

flow record BIDIR-1


Switch(config-flow-record)#

match ipv4 version


Switch(config-flow-record)#

match ipv4 protocol


Switch(config-flow-record)#

match application name
```

```
Switch(config-flow-record)#

match connection client ipv4 address

Switch(config-flow-record)#

match connection server ipv4 address

Switch(config-flow-record)#

match connection server transport port

Switch(config-flow-record)#

match flow observation point

Switch(config-flow-record)#

collect flow direction

Switch(config-flow-record)#

collect connection initiator

Switch(config-flow-record)#

collect connection new-connections

Switch(config-flow-record)#

collect connection client counter packets long

Switch(config-flow-record)#

connection client counter bytes network long

Switch(config-flow-record)#

collect connection server counter packets long

Switch(config-flow-record)#

connection server counter bytes network long

Switch(config-flow-record)#

collect timestamp absolute first

Switch(config-flow-record)#

collect timestamp absolute last

Switch(config-flow-record)#

end
```

```
Switch#

show flow record BIDIR-1


flow record BIDIR-1:
Description: User defined
No. of users: 0
Total field space: 78 bytes
Fields:
match ipv4 version
match ipv4 protocol
match application name
match connection client ipv4 address
match connection server ipv4 address
match connection server transport port
match flow observation point
collect flow direction
collect timestamp absolute first
collect timestamp absolute last
collect connection initiator
collect connection new-connections
collect connection server counter packets long
collect connection client counter packets long
collect connection server counter bytes network long
collect connection client counter bytes network long
```

Directional records are application-stats for input/output.

As shown in the output, configuration examples of input and output directional records:

Note: The command match interface input specifies a match to the input interface. The command match interface output specifies a match to the output interface. The command match application name is mandatory for AVC support.

<#root>

```
Switch(config)#

flow record APP-IN


Switch(config-flow-record)#

match ipv4 version


Switch(config-flow-record)#

match ipv4 protocol


Switch(config-flow-record)#

match ipv4 source address


Switch(config-flow-record)#
```

```
match ipv4 destination address

Switch(config-flow-record)#

match transport source-port

Switch(config-flow-record)#

match transport destination-port

Switch(config-flow-record)#

match interface input

Switch(config-flow-record)#

match application name

Switch(config-flow-record)#

collect interface output

Switch(config-flow-record)#

collect counter bytes long

Switch(config-flow-record)#

collect counter packets long

Switch(config-flow-record)#

collect timestamp absolute first

Switch(config-flow-record)#

collect timestamp absolute last

Switch(config-flow-record)#

end


Switch#

show flow record APP-IN


flow record APP-IN:
Description: User defined
No. of users: 0
Total field space: 58 bytes
Fields:
match ipv4 version
match ipv4 protocol
match ipv4 source address
match ipv4 destination address
match transport source-port
match transport destination-port
match interface input
match application name
collect interface output
collect counter bytes long
collect counter packets long
collect timestamp absolute first
collect timestamp absolute last
```

```
Switch(config)#

flow record APP-OUT


Switch(config-flow-record)#

match ipv4 version


Switch(config-flow-record)#

match ipv4 protocol


Switch(config-flow-record)#

match ipv4 source address


Switch(config-flow-record)#

match ipv4 destination address


Switch(config-flow-record)#

match transport source-port


Switch(config-flow-record)#

match transport destination-port


Switch(config-flow-record)#

match interface output


Switch(config-flow-record)#

match application name


Switch(config-flow-record)#

collect interface input


Switch(config-flow-record)#

collect counter bytes long


Switch(config-flow-record)#

collect counter packets long
```

```
Switch(config-flow-record)#
```

collect timestamp absolute first

```
Switch(config-flow-record)#
```

collect timestamp absolute last

```
Switch(config-flow-record)#
```

end

```
Switch#
```

show flow record APP-OUT

```
flow record APP-OUT:
Description: User defined
No. of users: 0
Total field space: 58 bytes
Fields:
match ipv4 version
match ipv4 protocol
match ipv4 source address
match ipv4 destination address
match transport source-port
match transport destination-port
match interface output
match application name
collect interface input
collect counter bytes long
collect counter packets long
collect timestamp absolute first
collect timestamp absolute last
```

Flow Exporter

Create a flow exporter to define export parameters.

As shown in the output, example configuration of the flow exporter:

<#root>

```
Switch(config)#
```

flow exporter AVC

```
Switch(config-flow-exporter)#
```

destination 192.168.69.2

```
Switch(config-flow-exporter)#
```

```
source vlan69

Switch(config-flow-exporter)#

end



Switch#

show run flow exporter AVC


Current configuration:
!
flow exporter AVC
destination 192.168.69.2
source Vlan69
!
```

Flow Monitor

Create a flow monitor to associate it to a flow record.

As shown in the output, example configuration of the flow monitor:

<#root>

```
Switch(config)#

flow monitor AVC-MONITOR


Switch(config-flow-monitor)#

record APP-OUT


Switch(config-flow-monitor)#

exporter AVC


Switch(config-flow-monitor)#

end



Switch#

show run flow monitor AVC-MONITOR


Current configuration:
!
flow monitor AVC-MONITOR
exporter AVC
record APP-OUT
```

Associate Flow Monitor to an Interface

You can attach up to two different AVC monitors with different predefined records to an interface at the same time.

As shown in the output, example configuration of the flow monitor:

<#root>

```
Switch(config)#

interface fi4/0/5


Switch(config-if)#

ip flow monitor AVC-MONITOR out


Switch(config-if)#

end



Switch#

show run interface fi4/0/5


Building configuration...
Current configuration : 134 bytes
!
interface FiveGigabitEthernet4/0/5
ip flow monitor AVC-MONITOR output
service-policy input WEBEX
ip nbar protocol-discovery
end
```

NBAR2

NBAR2 Dynamic Hitless Protocol Pack Upgrade

Protocol packs are software packages that update the NBAR2 protocol support on a device without replacement of the Cisco software on the device. A protocol pack contains information on applications officially supported by NBAR2 which are compiled and packed together. For each application, the protocol-pack includes information on application signatures and application attributes. Each software release has a built-in protocol-pack bundled with it.

- NBAR2 provides a way to update the protocol-packet without any traffic or service interruption and without the need to modigy the software image on the device(s).
- NBAR2 protocol-packets are available for download on Cisco Software Center at: NBAR2 Protocol Pack Library .

NBAR2 Protocol Pack Upgrade

Before installation of a new protocol pack, you must copy the protocol packet to the flash on all switch(es).

To load the new protocol pack, use the command ip nbar protocol-pack flash:<Pack Name>.

You do not need to reload the switch(es) to have the NBAR2 upgrade to occur.

As shown in the output, example configuration of how to load the NBAR2 Protocol Pack:

<#root>

```
Switch(config)#

ip nbar protocol-pack flash:newProtocolPack
```

To revert to the built-in protocol pack, use the command default ip nbar protocol-pack .

As shown in the output, example configuration of how to revert back to the built-in protocol pack:

<#root>

```
Switch(config)#

default ip nbar protocol-pack
```

Display NBAR2 Protocol Pack Information

To display protocol pack information use the commands listed:

- **show ip nbar version**
- show ip nbar protocol-pack active detail

As shown in the output, example output of those commands:

<#root>

```
Switch#

show ip nbar version


NBAR software

version: 37


NBAR minimum backward compatible version: 37
NBAR change ID: 293126

Loaded Protocol Pack(s):
Name: Advanced Protocol Pack
Version: 43.0
Publisher: Cisco Systems Inc.
NBAR Engine Version: 37
State: Active

Switch#show ip nbar protocol-pack active detail
Active Protocol Pack:
```

```
Name: Advanced Protocol Pack
Version: 43.0
Publisher: Cisco Systems Inc.
NBAR Engine Version: 37
State: Active
```

NBAR2 Custom Applications

NBAR2 supports the use of custom protocols to identify custom applications. Custom protocols support protocols and applications that NBAR2 does not currently support.

These can include the following:

- Specific application to an organization
- Applications specific to a geography

NBAR2 provides a way to manually customize applications through the command `ip nbar custom <myappname>`.

Note: Custom applications take precedence over built-in protocols

There are various types of application customization:

Generic protocol customization

- HTTP
- SSL
- DNS

Composite: Customization based on multiple protocols –**server-name**.

Layer3/Layer4 customization

- IPv4 address
- DSCP values
- TCP/UDP ports
- Flow source or destination direction

Byte Offset:Customization based on specific byte values in the payload

HTTP Customization

HTTP customization could be based on a combination of HTTP fields from:

- **cookie** - HTTP Cookie
- **host** - Host name of Origin Server that contains the resource
- **method** - HTTP method
- **referrer** - Address the resource request was obtained from

- **url** - Uniform Resource Locator path
- **user-agent** - Software used by agent that sends the request
- **version** - HTTP version
- **via** - HTTP via field

Example custom application called MYHTTP that uses the HTTP host *mydomain.com with Selector ID 10.

<#root>

```
Switch(config)#

ip nbar custom MYHTTP http host *mydomain.com id 10
```

SSL Customization

Customization can be done for SSL encrypted traffic through information extracted from the SSL Server Name Indication (SNI) or Common Name (CN).

Example custom application called MYSSL that uses SSL unique-name mydomain.com with selector ID 11.

<#root>

```
Switch(config)#

ip nbar custom MYSSL ssl unique-name *mydomain.com id 11
```

DNS Customization

NBAR2 examines DNS request and response traffic, and can correlate the DNS response to an application. The IP address returned from the DNS response is cached and used for later packet flows associated with that specific application.

The command ip nbar customapplication-namedns domain-nameidapplication-id is used for DNS customization. To extend an application, use the command ip nbar custom application-name dns domain-name domain-name extends existing-application.

Example custom application called MYDNS that uses the DNS domain name “mydomain.com” with selector ID 12.

<#root>

```
Switch(config)#

ip nbar custom MYDNS dns domain-name *mydomain.com id 12
```

Composite Customization

NBAR2 provides a way to customize applications based on domain names that appear in HTTP, SSL or

DNS.

Example custom application called MYDOMAIN that uses HTTP, SSL or DNS domain name mydomain.com with selector ID 13.

```
<#root>

Switch(config)#

ip nbar custom MYDOMAIN composite server-name *mydomain.com id 13
```

L3/L4 Customization

Layer3/Layer4 customization is based on the packet tuple and is always matched on the first packet of a flow.

Example custom application LAYER4CUSTOM that matches IP addresses 10.56.1.10 and 10.56.1.11, TCP and DSCP ef with selector ID 14.

```
<#root>

Switch(config)#

ip nbar custom LAYER4CUSTOM transport tcp id 14


Switch(config-custom)#

ip address 10.56.1.10 10.56.1.11


Switch(config-custom)#

dscp ef


Switch(config-custom)#

end
```

Monitor Custom Applications

To monitor custom applications utilize the show commands listed:

show ip nbar protocol-id | inc Custom

```
<#root>

Switch#

show ip nbar protocol-id | inc Custom


LAYER4CUSTOM              14            Custom
MYDNS                     12            Custom
MYDOMAIN                  13            Custom
```

```
MYHTTP                    10            Custom
MYSSL                     11            Custom
```

show ip nbar protocol-id CUSTOM_APP

<#root>

```
Switch#

show ip nbar protocol-id MYSSL

Protocol Name              id            type
----------------------------------------------
MYSSL                     11            Custom
```

Verify AVC

There are multiple steps to validate the functionality of AVC, this section provides commands and example output.

To validate that NBAR is active, you can run the command show ip nbar control-plane.

Key Areas:

- NBAR state must be **activated** in a correct scenario
- NBAR config state must be **ready** in a correct scenario

<#root>

```
Switch#

show ip nbar control-plane


NGCP Status:
============

graph sender info:
NBAR state is

ACTIVATED


NBAR config send mode is ASYNC
NBAR config state is

READY



NBAR update ID 3
NBAR batch ID ACK 3
NBAR last batch ID ACK clients 1 (ID: 4)
Active clients 1 (ID: 4)
NBAR max protocol ID ever 1935
NBAR Control-Plane Version: 37
```

```
<snip>
```

Validate that each switch member has an active data plane with the command show platform software fed switch active|standby|member wdavc function wdavc_stile_cp_show_info_ui:

Is DP activated must be **TRUE** in a correct scenario

<#root>

```
Switch#

show platform software fed switch active wdavc function wdavc_stile_cp_show_info_ui



Is DP activated :

TRUE


MSG ID : 3
Maximum number of flows: 262144
Current number of graphs: 1
Requests queue state : WDAVC_STILE_REQ_QUEUE_STATE_UP
Number of requests in queue :

0


Max number of requests in queue (TBD): 1
Counters:
activate_msgs_rcvd : 1
graph_download_begin_msgs_rcvd : 3
stile_config_msgs_rcvd : 1584
graph_download_end_msgs_rcvd : 3
deactivate_msgs_rcvd : 0
intf_proto_disc_msgs_rcvd : 1
intf_attach_msgs_rcvd : 2
cfg_response_msgs_sent : 1593
num_of_handle_msg_from_fmanfp_events : 1594
num_of_handle_request_from_queue : 1594
num_of_handle_process_requests_events : 1594
```

Utilize the command show platform software fed switch active|standby|member wdavc flowsto display key information:

<#root>

```
Switch#

show platform software fed switch active wdavc flows



CurrFlows=1, Watermark=1

IX |IP1           |IP2           |PORT1|PORT2|L3   |L4   |VRF |TIMEOUT|APP     |TUPLE|FLOW     |IS FIF
   |              |              |     |     |PROTO|PROTO|VLAN|SEC    |NAME    |TYPE |TYPE     |SWAPPED
--------------------------------------------------------------------------------------
```

```
1  |192.168.100.2 |192.168.200.2 |68   |67   |1    |17   |0   |360    |unknown |Full |Real Flow|Yes
```

Key Fields:

CurrFlows: Demonstrates how many active flows that are tracked by AVC

Watermark: Demonstrates the largest number of flows historically tracked by AVC

TIMEOUT SEC: Inactivity timeout based on the identified application

APP NAME: Identified application

FLOW TYPE: Real Flow indicates this was created as a result of inbound data. Pre Flow indicates this flow is created as a result of inbound data. Pre-flows are used for anticipated media flows

TUPLE TYPE: Real flows are always full tuple, Pre-flows are either full tuple or half tuple

BYPASS: If set to TRUE, indicates that no more packets are required by software in order to identify this flow

FINAL: If set to TRUE, indicates that the application does not change anymore for this flow

BYPASS PKT: How many packets were needed in order to get to final classification

#PKTS: How many packets were actually punted to software for this flow

View additional details about current flows, you can utilize the command **show platform software fed switch active wdavc function wdavc_ft_show_all_flows_seg_ui**.

<#root>

```
Switch#

show platform software fed switch active wdavc function wdavc_ft_show_all_flows_seg_ui


CurrFlows=1, Watermark=1

IX |IP1           |IP2           |PORT1|PORT2|L3    |L4    |VRF |TIMEOUT|APP      |TUPLE |FLOW      |IS FIF
   |              |              |     |     |PROTO|PROTO|VLAN|SEC    |NAME     |TYPE  |TYPE      |SWAPPED
------------------------------------------------------------------------------------------------------------
1  |192.168.100.2 |192.168.200.2|68   |67   |1    |17   |0   |360    |unknown |Full  |Real Flow|Yes

   SEG IDX |I/F ID |OPST I/F |SEG DIR |FIF DIR |Is SET |DOP ID |NFL HDL  |BPS PND |APP PND |FRST TS  |L
   ---------------------------------------------------------------------------------------------------------
   0       |9      |----     |Ingress |True    |True   |0      |50331823 |0       |0       |177403000|1
```

Key Fields

I/F ID: Specifies the Interface ID

SEG DIR: Specifies ingress of egress direction

FIF DIR: Determines whether or not this is the flow initiator direction

NFL HDL: Flow ID in hardware

To view the entry in hardware run the command show platform software fed switch active fnf flow-record asic <number> start-index <number> num-flows <number of flows>.

Note: To choose the ASIC, it is the ASIC instance which the port is mapped to. To identify the ASIC, utilize the command show platform software fed switch active|standby|member ifm mappings. The start-index can be set to 0 if you are not interested in a specific flow. Otherwise, the start-index needs to be specified. For num-flows, that specifies the number of flows that can be viewed, maximum 10.

<#root>

```
Switch#

show platform software fed switch active fnf flow-record asic 3 start-index 0 num-flows 1
```

```
1 flows starting at 0 for asic 3:------------------------------------------------
Idx 175 :
{90, ALR_INGRESS_NET_FLOW_ACL_LOOKUP_TYPE1 = 0x01}
{91, ALR_INGRESS_NET_FLOW_ACL_LOOKUP_TYPE2 = 0x01}
{0, ALR_INGRESS_NFL_SPECIAL1 = 0x00}
{11 PAD-UNK = 0x0000}
{94, PHF_INGRESS_DEST_PORT_OR_ICMP_OR_IGMP_OR_PIM_FIRST16B = 0x0043}
{93, PHF_INGRESS_SRC_PORT = 0x0044}
{67, PHF_INGRESS_IPV4_DEST_ADDRESS = 0xc0a8c802}
{68, PHF_INGRESS_IPV4_SRC_ADDRESS = 0xc0a86402}
{56, PHF_INGRESS_L3_PROTOCOL = 0x11}
FirstSeen = 0x2b4fb, LastSeen = 0x2eede, sysUptime = 0x2ef1c
PKT Count = 0x000000000001216f, L2ByteCount = 0x0000000001873006
```

Look for Various Errors and Warnings in the Data Path

Utilize the command show platform software fed switch active|standby|member wdavc function wdavc_ft_show_stats_ui | inc err|warn|fail to view potential flow table errors:

<#root>

```
Switch#

show platform software fed switch active wdavc function wdavc_ft_show_stats_ui | inc err|warn|fail
```

```
Bucket linked exceed max error : 0
extract_tuple_non_first_fragment_warn : 0
ft_client_err_alloc_fail : 0
ft_client_err_detach_fail : 0
ft_client_err_detach_fail_intf_attach : 0
ft_inst_nfl_clock_sync_err : 0
ft_ager_err_invalid_timeout : 0
ft_intf_err_alloc_fail : 0
ft_intf_err_detach_fail : 0
ft_inst_err_unreg_client_all : 0
ft_inst_err_inst_del_fail : 0
```

```
ft_flow_seg_sync_nfl_resp_pend_del_warn : 0
ager_sm_cb_bad_status_err : 0
ager_sm_cb_received_err : 0
ft_ager_to_time_no_mask_err : 0
ft_ager_to_time_latest_zero_ts_warn : 0
ft_ager_to_time_seg_zero_ts_warn : 0
ft_ager_to_time_ts_bigger_curr_warn : 0
ft_ager_to_ad_nfl_resp_error : 0
ft_ager_to_ad_req_all_recv_error : 0
ft_ager_to_ad_req_error : 0
ft_ager_to_ad_resp_error : 0
ft_ager_to_ad_req_restart_timer_due_err : 0
ft_ager_to_flow_del_nfl_resp_error : 0
ft_ager_to_flow_del_all_recv_error : 0
ft_ager_to_flow_del_req_error : 0
ft_ager_to_flow_del_resp_error : 0
ft_consumer_timer_start_error : 0
ft_consumer_tw_stop_error : 0
ft_consumer_memory_error : 0
ft_consumer_ad_resp_error : 0
ft_consumer_ad_resp_fc_error : 0
ft_consumer_cb_err : 0
ft_consumer_ad_resp_zero_ts_warn : 0
ft_consumer_ad_resp_zero_pkts_bytes_warn : 0
ft_consumer_remove_on_count_zero_err : 0
ft_ext_field_ref_cnt_zero_warn : 0
ft_ext_gen_ref_cnt_zero_warn : 0
```

Utilize the command `show platform software fed switch active wdavc function wdavc_stile_stats_show_ui | inc err` to view any potential NBAR errors:

<#root>

```
Switch#

show platform software fed switch active wdavc function wdavc_stile_stats_show_ui | inc err


find_flow_error : 0
add_flow_error : 0
remove_flow_error : 0
detach_fo_error : 0
is_forward_direction_error : 0
set_flow_aging_error : 0
ft_process_packet_error : 0
sys_meminfo_get_error : 0
```

Verify that Packets are Cloned to CPU

Utilize the command `show platform software fed switch active punt cpuq 21 | inc received` to verify that packets are cloned to the CPU for NBAR processing:

Note: In the lab this number did not increment.

<#root>

```
Switch#

show platform software fed switch active punt cpuq 21 | inc received


Packets received from ASIC : 63
```

Identify CPU Congestion

In times of congestion, packets can be dropped before sent to WDAVC process. Utilize the command **show platform software fed switch active wdavc function fed_wdavc_show_ots_stats_ui**to validate:

<#root>

```
Switch#

show platform software fed switch active wdavc function fed_wdavc_show_ots_stats_ui


OTS Limits
----------------------------------------------
ots_queue_max : 20000
emer_bypass_ots_queue_stress : 4000
emer_bypass_ots_queue_normal : 200
OTS Statistics
----------------------------------------------
total_requests : 40
total_non_wdavc_requests : 0
request_empty_field_data_error : 0
request_invalid_di_error : 0
request_buf_coalesce_error : 0
request_invalid_format_error : 0
request_ip_version_error : 0
request_empty_packet_error : 0
memory_allocation_error : 0
emergency_bypass_requests_warn : 0
dropped_requests : 0
enqueued_requests : 40
max_ots_queue : 0
```

Tip: To clear the punt drop counter utilize the command show platform software fed switch active wdavc function fed_wdavc_clear_ots_stats_ui.

Identify Scale Issues

If there are no free FNF entries in hardware, traffic is not subject to NBAR2 classification. Utilize the command show platform software fed switch active fnf sw-table-sizes asic <number> shadow 0 to confirm:

Note: Flows that are created are specific to the switch and asic core when they are created. The switch number (active, standby, etc) needs to specified accordingly. The ASIC number that is input is tied to the respective interface, use show platform software fed switch active|standby|member ifm mappings to determine ASIC that corresponds to the interface. For the shadow option, always use 0.

<#root>

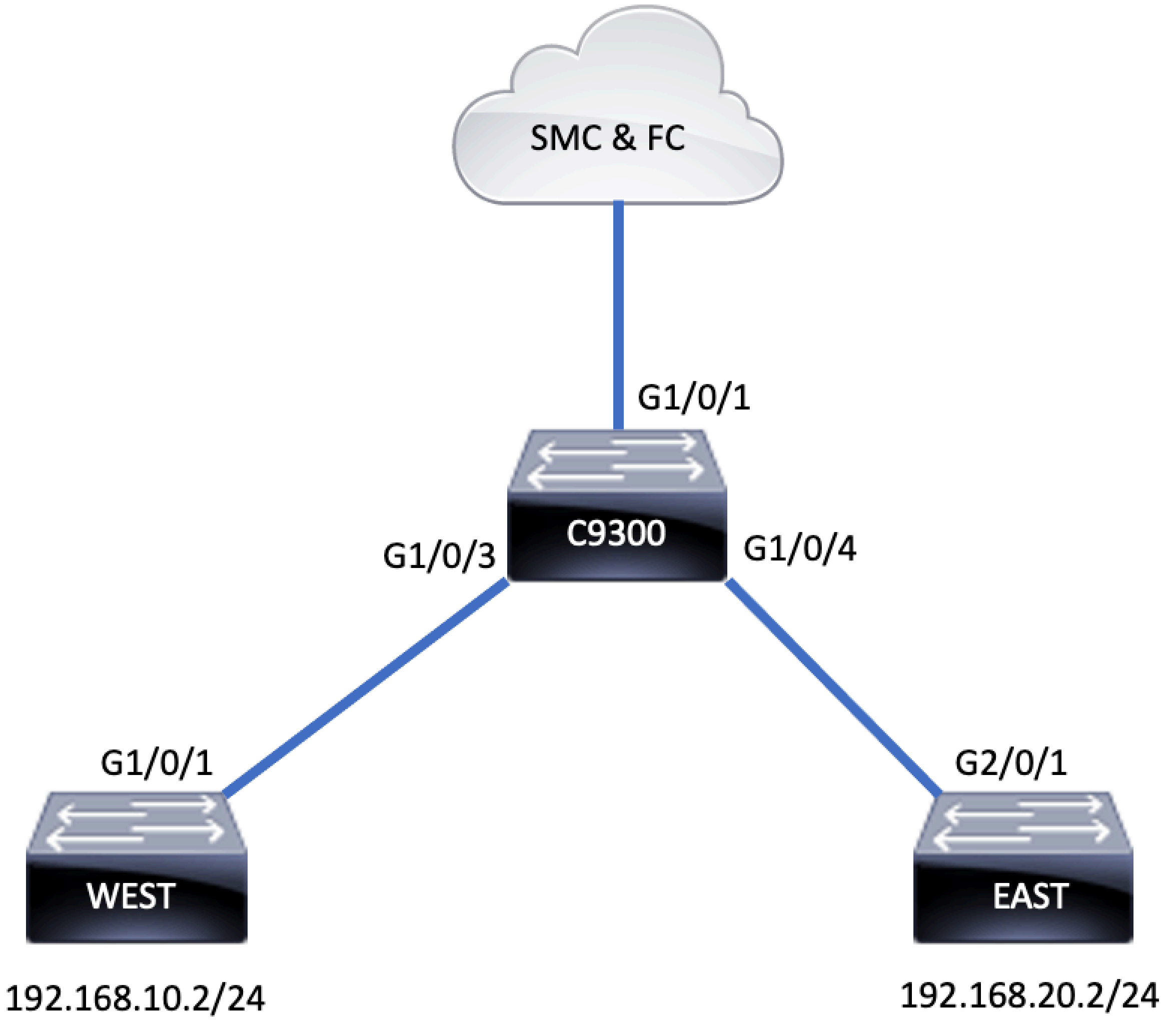
```
Switch#

show platform software fed switch active fnf sw-table-sizes asic 3 shadow 0



---------------------------------
Global Bank Allocation
---------------------------------
Ingress Banks : Bank 0
Egress Banks : Bank 1
---------------------------------
Global flow table Info
INGRESS usedBankEntry 1 usedOvfTcamEntry 0
EGRESS usedBankEntry 0 usedOvfTcamEntry 0

<-- 256 means TCAM entries are full


---------------------------------
Flows Statistics
INGRESS TotalSeen=1 MaxEntries=1 MaxOverflow=0
EGRESS TotalSeen=0 MaxEntries=0 MaxOverflow=0

---------------------------------
Partition Table
---------------------------------
## Dir  Limit CurrFlowCount OverFlowCount MonitoringEnabled
 0 ING      0         0             0           0
 1 ING  16640         1             0           1
 2 ING      0         0             0           0
 3 ING  16640         0             0           0
 4 ING      0         0             0           0
 5 ING   8192         0             0           1
 6 ING      0         0             0           0
 7 ING      0         0             0           0
 8 ING      0         0             0           0
 9 ING      0         0             0           0
10 ING      0         0             0           0
11 ING      0         0             0           0
12 ING      0         0             0           0
13 ING      0         0             0           0
14 ING      0         0             0           0
15 ING      0         0             0           0
 0 EGR      0         0             0           0
 1 EGR  16640         0             0           1
 2 EGR      0         0             0           0
 3 EGR  16640         0             0           0
 4 EGR      0         0             0           0
 5 EGR   8192         0             0           1
 6 EGR      0         0             0           0
 7 EGR      0         0             0           0
 8 EGR      0         0             0           0
 9 EGR      0         0             0           0
10 EGR      0         0             0           0
11 EGR      0         0             0           0
12 EGR      0         0             0           0
13 EGR      0         0             0           0
14 EGR      0         0             0           0
15 EGR      0         0             0           0
```

Encrypted Traffic Analytics (ETA)

Background Information

- ETA focuses on identification of malware communication in encrypted traffic through passive monitoring, extraction of relevant data elements, and a combination of behavioral modeling and machine learning with cloud-based global security.
- ETA leverages telemetry from NetFlow as well as encrypted malware detection and cryptographic compliance and sends this data to Cisco Stealthwatch.
- ETA extracts two main data elements: the Initial Data Packet (IDP) and the Sequence of Packet Length and Time (SPLT).

Network Diagram



Components

ETA is comprised of several different components that are used in conjunction to create the ETA solution:

- NetFlow - Standard that defines data elements exported by network devices that describe the flows on the network.
- Cisco Stealthwatch - Harnesses the power of network telemetry that includes NetFlow, IPFIX, proxy logs, and deep packet inspection of raw packets - to provide advanced network visibility, security

intelligence, and analytics.
- Cisco Cognitive Intelligence - Finds malicious activity that has bypassed security controls or entered through unmonitored channels and inside an organization environment.
- Encrypted Traffic Analytics - Cisco IOS XE feature that uses advanced behavioral algorithms to identify malicious traffic patterns through analysis of inftraflow metadata of encrypted traffic, detects potential threats hidden in encrypted traffic.

Note: This part of the document only focuses on configuration and verification of ETA and NetFlow on the Catalyst 9000 series switch and does not cover Stealthwatch Management Console (SMC) and Flow Collector (FC) deployment to the Cognitive Intelligence Cloud.

Restrictions

- Deployment of ETA requires DNA Advantage to function
- ETA and a transmit (TX) Switched Port Analyzer (SPAN) is not supported on the same interface.

This is not an inclusive list, consult the appropriate configuration guide for the switch and version of code for all restrictions.

Configuration

As show in the output, enable ETA on the switch globally and define the flow export destination:

<#root>

```
C9300(config)#

et-analytics


C9300(config-et-analytics)#

ip flow-export destination 172.16.18.1 2055
```

Tip: You MUST use port 2055, do not use another port number.

Next, configure Flexible NetFlow as show in the output:

Configure Flow Record

<#root>

```
C9300(config)#

flow record FNF-RECORD


C9300(config-flow-record)#

match ipv4 protocol
```

```
C9300(config-flow-record)#
match ipv4 source address

C9300(config-flow-record)#
match ipv4 destination address

C9300(config-flow-record)#
match transport source-port

C9300(config-flow-record)#
match transport destination-port

C9300(config-flow-record)#
collect counter bytes long

C9300(config-flow-record)#
collect counter packets long

C9300(config-flow-record)#
collect timestamp absolute first

C9300(config-flow-record)#
collect timestamp absolute last
```

Configure Flow Monitor

<#root>

```
C9300(config)#
flow exporter FNF-EXPORTER

C9300(config-flow-exporter)#
destination 172.16.18.1

C9300(config-flow-exporter)#
transport udp 2055

C9300(config-flow-exporter)#
template data timeout 30

C9300(config-flow-exporter)#
```

```
option interface-table

C9300(config-flow-exporter)#
option application-table timeout 10

C9300(config-flow-exporter)#
exit
```

Configure Flow Record

```
<#root>
C9300(config)#
flow monitor FNF-MONITOR

C9300(config-flow-monitor)#
exporter FNF-EXPORTER

C9300(config-flow-monitor)#
record FNF-RECORD

C9300(config-flow-monitor)#
end
```

Apply Flow Monitor

```
<#root>
C9300(config)#
int range g1/0/3-4

C9300(config-if-range)#
ip flow mon FNF-MONITOR in

C9300(config-if-range)#
ip flow mon FNF-MONITOR out

C9300(config-if-range)#
end
```

Enable ETA on Switch Interface(s)

<#root>

```
C9300(config)#
```

```
interface range g1/0/3-4
```

```
C9300(config-if-range)#
```

```
et-analytics enable
```

Verify

Verify that the ETA monitor, eta-mon, is active. Confirm that the status is allocated through the command show flow monitor eta-mon .

<#root>

```
C9300#
```

```
show flow monitor eta-mon
```

```
Flow Monitor eta-mon:
Description: User defined
Flow Record: eta-rec
Flow Exporter: eta-exp
Cache:
Type: normal (Platform cache)
Status:
```

```
allocated
```

```
Size: 10000 entries
Inactive Timeout: 15 secs
Active Timeout: 1800 secs
```

Verify that the ETA cache is populated. When NetFlow and ETA are configured on the same interface, utilize **show flow monitor <monitor name> cache** instead of show flow monitor eta-mon cache as the output from show flow monitor eta-mon cache is empty:

<#root>

```
C9300#
```

```
show flow monitor FNF-MONITOR cache
```

```
Cache type: Normal (Platform cache)
Cache size: 10000
Current entries: 4

Flows added: 8
Flows aged: 4
- Inactive timeout ( 15 secs) 4

IPV4 SOURCE ADDRESS: 192.168.10.2
IPV4 DESTINATION ADDRESS: 192.168.20.2
TRNS SOURCE PORT: 0
TRNS DESTINATION PORT: 0
IP PROTOCOL: 1
counter bytes long: 500
counter packets long: 5
timestamp abs first: 21:53:23.390
timestamp abs last: 21:53:23.390

IPV4 SOURCE ADDRESS: 192.168.20.2
IPV4 DESTINATION ADDRESS: 192.168.10.2
TRNS SOURCE PORT: 0
TRNS DESTINATION PORT: 0
IP PROTOCOL: 1
counter bytes long: 500
counter packets long: 5
timestamp abs first: 21:53:23.390
timestamp abs last: 21:53:23.390

IPV4 SOURCE ADDRESS: 192.168.20.2
IPV4 DESTINATION ADDRESS: 192.168.10.2
TRNS SOURCE PORT: 0
TRNS DESTINATION PORT: 0
IP PROTOCOL: 1
counter bytes long: 500
counter packets long: 5
timestamp abs first: 21:53:23.390
timestamp abs last: 21:53:23.390

IPV4 SOURCE ADDRESS: 192.168.10.2
IPV4 DESTINATION ADDRESS: 192.168.20.2
TRNS SOURCE PORT: 0
TRNS DESTINATION PORT: 0
IP PROTOCOL: 1
counter bytes long: 500
counter packets long: 5
timestamp abs first: 21:53:23.390
timestamp abs last: 21:53:23.390
```

Validate that flows are exported towards the SMC and FC with the command show flow exporter eta-exp statistics .

<#root>

```
C9300#

show flow exporter eta-exp statistics


Flow Exporter eta-exp:
Packet send statistics (last cleared 03:05:32 ago):
Successfully sent: 3 (3266 bytes)

Client send statistics:
Client: Flow Monitor eta-mon
Records added: 4
- sent: 4
```

```
Bytes added: 3266
- sent: 3266
```

Confirm that SPLT and IDP are exported to the FC with the command show platform software fed switch active fnf et-analytics-flows.

<#root>

```
C9300#

show platform software fed switch active fnf et-analytics-flows



ET Analytics Flow dump

=================
Total packets received : 20
Excess packets received : 0
Excess syn received : 0
Total eta records added : 4
Current eta records : 0
Total eta splt exported : 2
Total eta IDP exported : 2
```

Validate which interfaces are configured for et-analytics with the command show platform software et-analytics interfaces .

<#root>

```
C9300#

show platform software et-analytics interfaces


ET-Analytics interfaces
GigabitEthernet1/0/3
GigabitEthernet1/0/4

ET-Analytics VLANs
```

Use the command show platform software et-analytics global to view a global state of ETA:

<#root>

```
C9300#

show plat soft et-analytics global


ET-Analytics Global state
=========================
All Interfaces : Off
IP Flow-record Destination : 10.31.126.233 : 2055
```

```
Inactive timer : 15

ET-Analytics interfaces
GigabitEthernet1/0/3
GigabitEthernet1/0/4

ET-Analytics VLANs
```