Verify Contracts and Rules in the ACI Fabric

Document ID: 119023

Contributed by Paul Raytick and Robert Correiro, Cisco TAC
Engineers.

Jun 29, 2015

Contents

Introduction

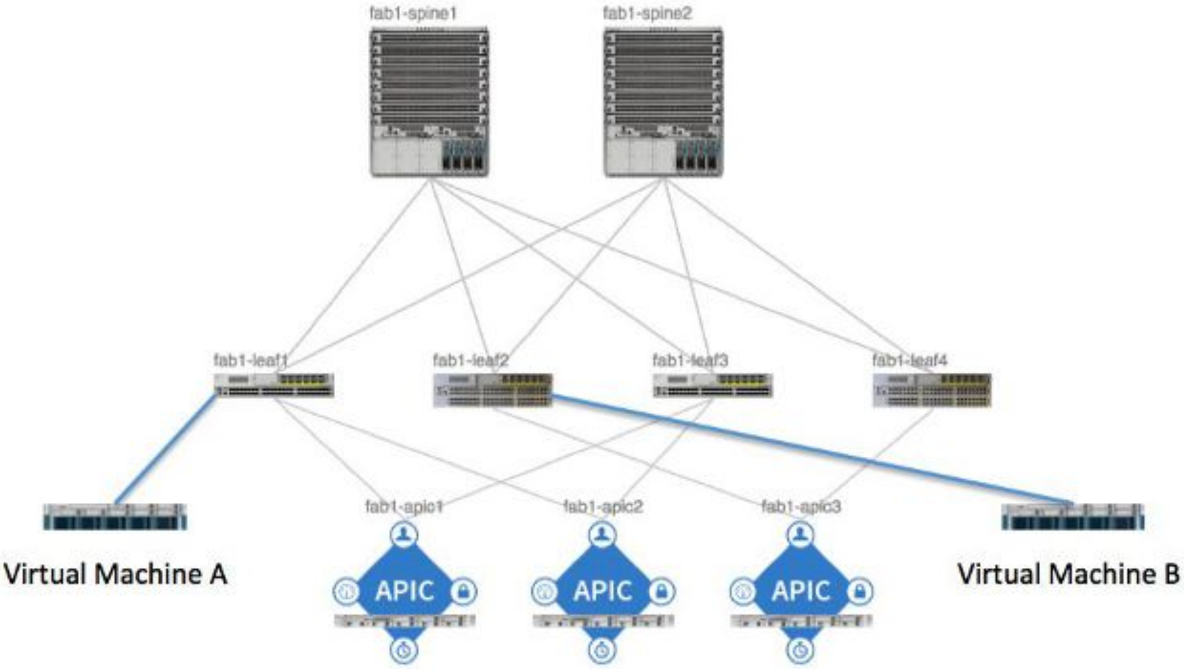
This document describes how to verify that contracts are configured and behave properly in the Application Centric Infrastructure (ACI) fabric.

Note: Verification of the logical and concrete models, as well as the hardware programming, is described in this document.

Topology

In the example that is used throughout this document, Virtual Machine−A (VM) is attached to Leaf1, and a contract is in place that allows it to communicate with VM−B, which is attached to Leaf2. The contract allows both Internet Control Message Protocol (ICMP) and HTTP.

This image illustrates the topology:



Process Overview

This is the policy interaction and flow for contracts and rules:

1. The Policy Manager on the Application Policy Infrastructure Controller (APIC) communicates with the Policy Element Manager on the switch.

2. The Policy Element Manager on the switch programs the Object Store on the switch.

3. The Policy Manager on the switch communicates with the Access Control List Quality of Service (ACLQOS) client on the switch.

4. The ACLQOS client programs the hardware.

Identify the Contract/Zoning Rule Used

Here is an example ***show zoning−rule*** command output from the leaf, before the contract is added for the two End Point Groups (EPGs).

```
fab1_leaf1# show zoning−rule

Rule ID  SrcEPG  DstEPG  FilterID  operSt   Scope     Action

=======  ======  ======  ========  ======   =====     ======

4096     0       0       implicit  enabled  16777200  deny,log

4097     0       0       implicit  enabled  3080192   deny,log

4098     0       0       implicit  enabled  2686976   deny,log

4099     0       49154   implicit  enabled  2686976   permit

4102     0       0       implicit  enabled  2097152   deny,log
```

```
4103     0       32771   implicit  enabled  2097152   permit
4117     16387   16386   12        enabled  2097152   permit
4116     16386   16387   13        enabled  2097152   permit
4100     16386   49154   default   enabled  2097152   permit
4101     49154   16386   default   enabled  2097152   permit
4104     0       32770   implicit  enabled  2097152   permit
4105     49155   16387   13        enabled  2097152   permit
4112     16387   49155   13        enabled  2097152   permit
4113     49155   16387   12        enabled  2097152   permit
4114     16387   49155   12        enabled  2097152   permit
[snip]
```

This is the same command output after the contract is added so that the two EPGs can communicate with each other:

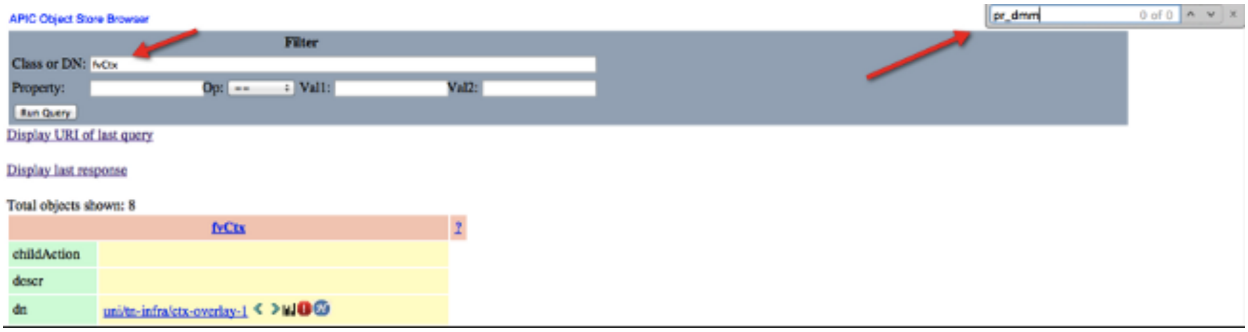
```
fab1_leaf1# show zoning-rule
Rule ID  SrcEPG  DstEPG  FilterID  operSt   Scope     Action
=======  ======  ======  ========  ======   ========  ========
4096     0       0       implicit  enabled  16777200  deny,log
4097     0       0       implicit  enabled  3080192   deny,log
4098     0       0       implicit  enabled  2686976   deny,log
4099     0       49154   implicit  enabled  2686976   permit
4131     49155   32771   7         enabled  2686976   permit
4132     32771   49155   6         enabled  2686976   permit
4102     0       0       implicit  enabled  2097152   deny,log
4103     0       32771   implicit  enabled  2097152   permit
4117     16387   16386   12        enabled  2097152   permit
4116     16386   16387   13        enabled  2097152   permit
4100     16386   49154   default   enabled  2097152   permit
4101     49154   16386   default   enabled  2097152   permit
4104     0       32770   implicit  enabled  2097152   permit
4105     49155   16387   13        enabled  2097152   permit
4112     16387   49155   13        enabled  2097152   permit
4113     49155   16387   12        enabled  2097152   permit
4114     16387   49155   12        enabled  2097152   permit
```

```
[snip]
```

Note: Notice the new rule IDs (***4131*** and ***4132***) that were added, the filter IDs of ***7*** and ***6***, and the scope of ***2686976***.

Caution: This command output allows you to easily locate the rules that you must examine in a lab system; however, this can be cumbersome in a production environment with the dynamic changes that occur.

Another method that you can employ in order to locate the rules of interest is to use *Visore*. Perform a search on the context Managed Object (MO) for ***fvCtx***. You can then search on that screen for your specific context Distinguished Name (DN), as shown here:



APIC Object Store Browser

pr_dmm 0 of 0

Filter

Class or DN: fvCtx

Property: Op: == Val1: Val2:

Run Query

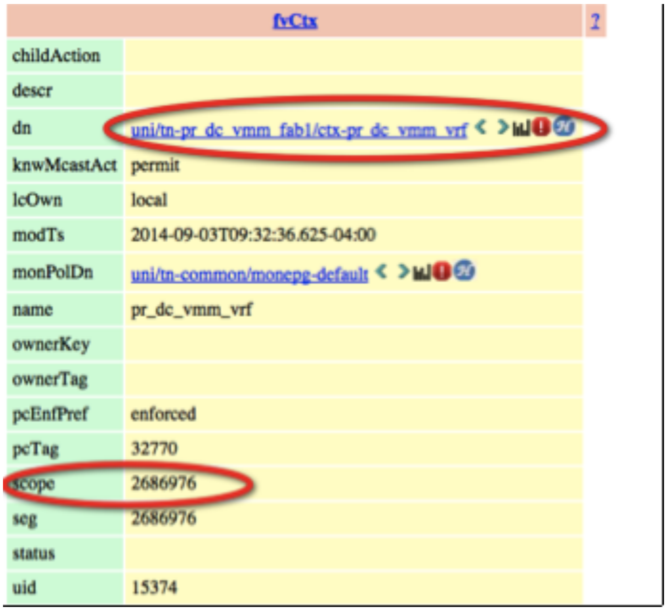
Display URI of last query

Display last response

Total objects shown: 8

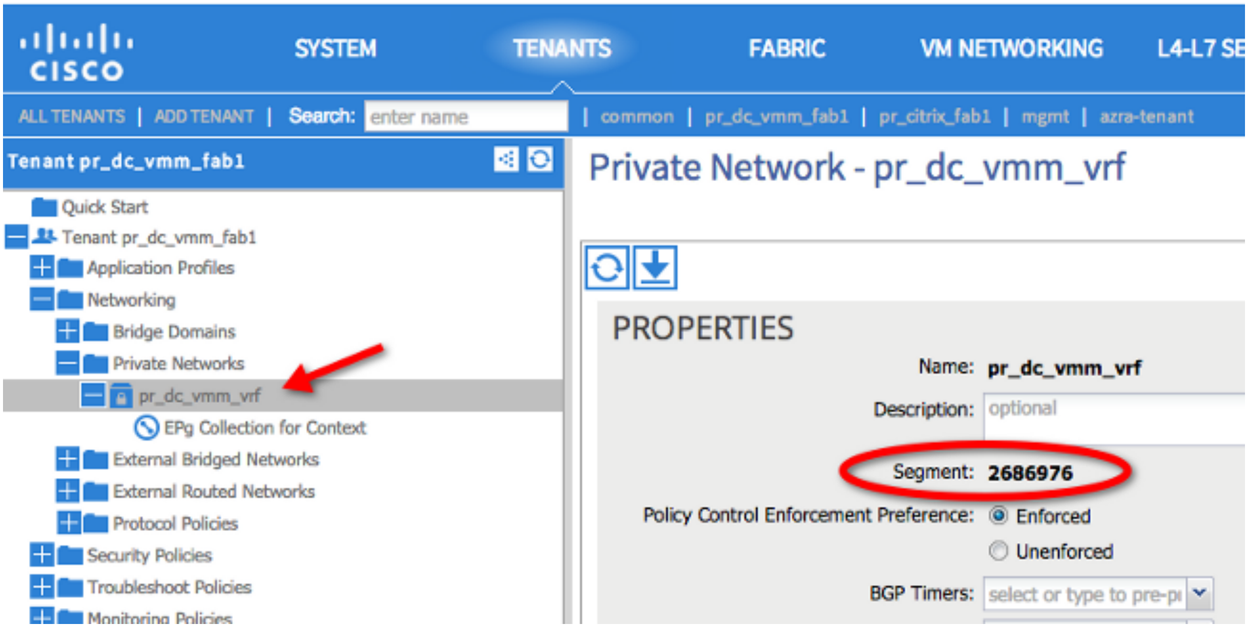
fvCtx	?
childAction	
descr	
dn	uni/tn-infra/ctx-overlay-1

Take note of the scope for that context. You can use this in order to map to the ***show−zoning−rule*** command output so that you can locate the rules that you must query:



fvCtx	?
childAction	
descr	
dn	uni/tn-pr_dc_vmm_fab1/ctx-pr_dc_vmm_vrf
knwMcastAct	permit
lcOwn	local
modTs	2014-09-03T09:32:36.625-04:00
monPolDn	uni/tn-common/monepg-default
name	pr_dc_vmm_vrf
ownerKey	
ownerTag	
pcEnfPref	enforced
pcTag	32770
scope	2686976
seg	2686976
status	
uid	15374

You can also identify the segment ID/scope for the context from the User Interface (UI), as shown here:



Private Network - pr_dc_vmm_vrf

PROPERTIES

Name: pr_dc_vmm_vrf

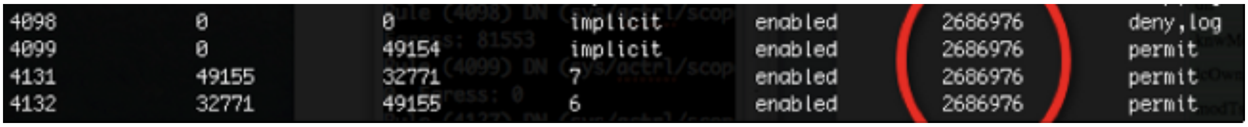
Description: optional

Segment: 2686976

Policy Control Enforcement Preference: ● Enforced ○ Unenforced

BGP Timers: select or type to pre-p

This scope matches that shown in the ***show zoning−rules*** command output:



```
4098    0       0       implicit    enabled    2686976    deny,log
4099    0       49154   implicit    enabled    2686976    permit
4131    49155   32771   7           enabled    2686976    permit
4132    32771   49155   6           enabled    2686976    permit
```

Once you have the scope ID information and you identify the rule and filter IDs, you can use the next command in order to verify that you hit the new filters (and not the *implicit deny* messages between the EPGs). The implicit deny message is included so that by default, the EPGs cannot communicate.

Notice in this command output that Leaf1, Filter−6 (***f−6***) is incrementing:

```
fab1_leaf1# show system internal policy-mgr stats | grep 2686976

Rule (4098) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-any-f-implicit)
 Ingress: 0, Egress: 81553

Rule (4099) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-49154-f-implicit)
 Ingress: 0, Egress: 0

Rule (4131) DN (sys/actrl/scope-2686976/rule-2686976-s-49155-d-32771-f-7)
 Ingress: 0, Egress: 0

Rule (4132) DN (sys/actrl/scope-2686976/rule-2686976-s-32771-d-49155-f-6)
 Ingress: 1440, Egress: 0

fab1_leaf1# show system internal policy-mgr stats | grep 2686976

Rule (4098) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-any-f-implicit)
 Ingress: 0, Egress: 81553

Rule (4099) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-49154-f-implicit)
 Ingress: 0, Egress: 0

Rule (4131) DN (sys/actrl/scope-2686976/rule-2686976-s-49155-d-32771-f-7)
 Ingress: 0, Egress: 0

Rule (4132) DN (sys/actrl/scope-2686976/rule-2686976-s-32771-d-49155-f-6)
 Ingress: 1470, Egress: 0
```

Notice in this command output that Leaf2, Filter−7 (***f−7***) is incrementing:

```
fab1_leaf2# show system internal policy-mgr stats | grep 268697

Rule (4098) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-any-f-implicit)
 Ingress: 0, Egress: 80257

Rule (4099) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-49153-f-implicit)
 Ingress: 0, Egress: 0

Rule (4117) DN (sys/actrl/scope-2686976/rule-2686976-s-32771-d-49155-f-6)
 Ingress: 0, Egress: 0

Rule (4118) DN (sys/actrl/scope-2686976/rule-2686976-s-49155-d-32771-f-7)
 Ingress: 2481, Egress: 0

fab1_leaf2# show system internal policy-mgr stats | grep 268697

Rule (4098) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-any-f-implicit)
 Ingress: 0, Egress: 80257

Rule (4099) DN (sys/actrl/scope-2686976/rule-2686976-s-any-d-49153-f-implicit)
 Ingress: 0, Egress: 0

Rule (4117) DN (sys/actrl/scope-2686976/rule-2686976-s-32771-d-49155-f-6)
 Ingress: 0, Egress: 0

Rule (4118) DN (sys/actrl/scope-2686976/rule-2686976-s-49155-d-32771-f-7)
 Ingress: 2511, Egress: 0
```

Tip: Knowledge of the scope, rule ID, destination and source pcTags, and filter is important with attempts to troubleshoot this issue further. It is also useful to have knowledge of the EPGs between which the rule ID exists.

You can perform a search on the MO with the DN name ***fvAEPg*** and *grep* for the particular pcTag via the ***moquery*** command, as shown here:

```
admin@RTP_Apic1:~> moquery -c fvAEPg | grep 49155 -B 5

dn : uni/tn-Prod/ap-commerceworkspace/epg-Web
lcOwn : local
matchT : AtleastOne
modTs : 2014-10-16T01:27:35.355-04:00
monPolDn : uni/tn-common/monepg-default
pcTag : 49155
```

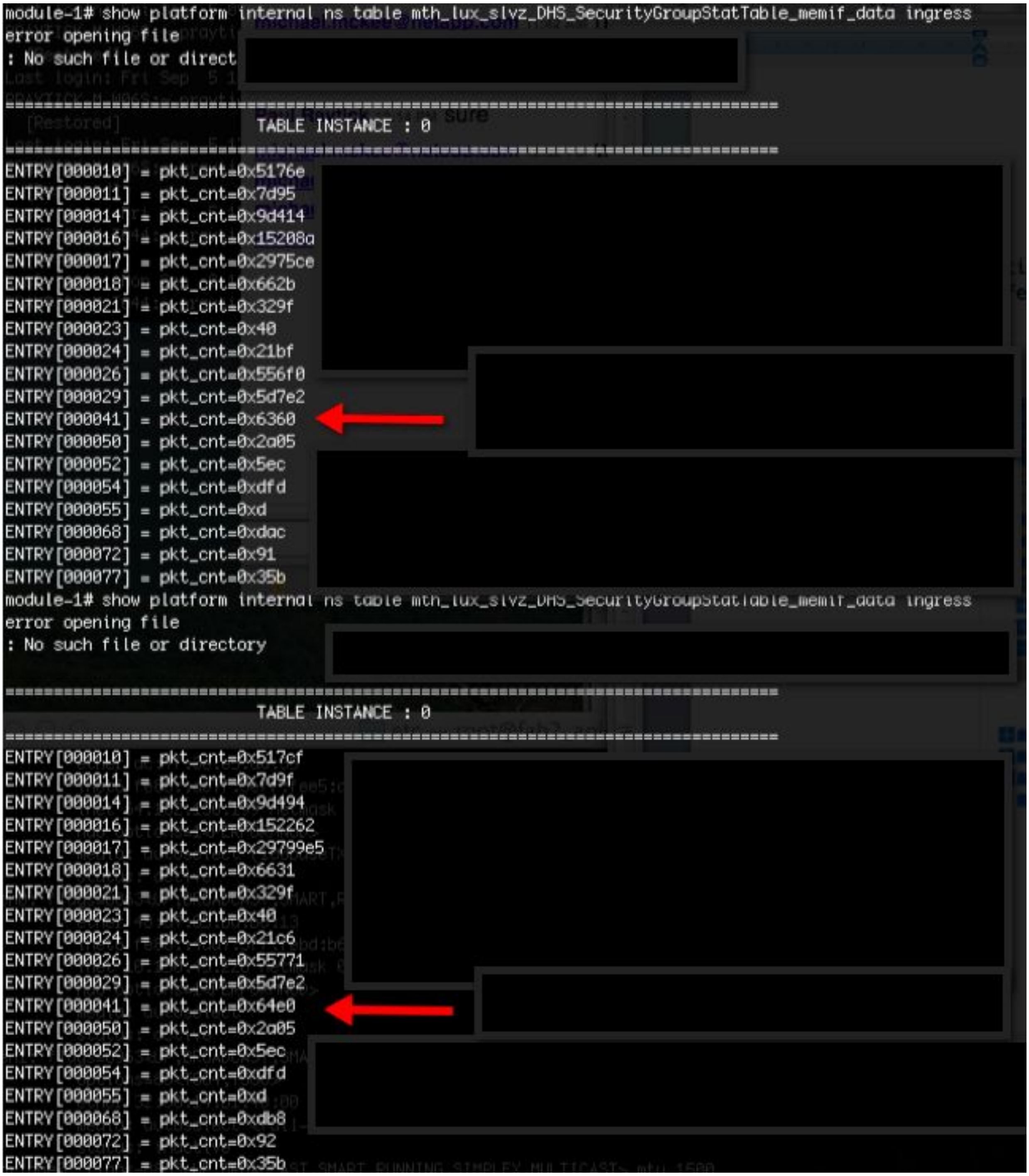
You can also use the *filter* option with the ***moquery*** command, as shown here:

```
admin@RTP_Apic1:~> moquery -c fvAEPg -f 'fv.AEPg.pcTag=="49155"'
Total Objects shown: 1

# fv.AEPg
name : Web
childAction :
configIssues :
configSt : applied
descr :
dn : uni/tn-Prod/ap-commerceworkspace/epg-Web
lcOwn : local
matchT : AtleastOne
modTs : 2014-10-16T01:27:35.355-04:00
monPolDn : uni/tn-common/monepg-default
pcTag : 49155
```

```
prio : unspecified
rn : epg-Web
scope : 2523136
status :
triggerSt : triggerable
uid : 15374
```

Verify Hardware Programming

Now you can verify the hardware entry for the rule. In order to view the hardware information, enter the ***show platform internal ns table mth_lux_slvz_DHS_SecurityGroupStatTable_memif_data ingress*** command (this is a ***vsh_lc*** command):

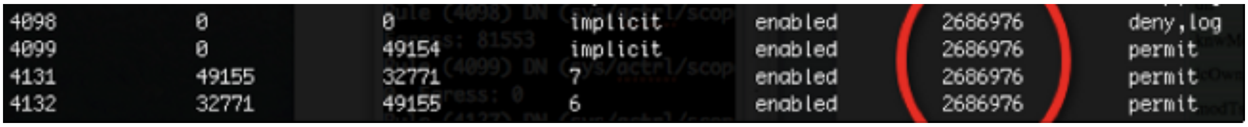


```
module-1# show platform internal ns table mth_lux_slvz_DHS_SecurityGroupStatTable_memif_data ingress
error opening file
: No such file or direct [redacted]

=================================================================================
                          TABLE INSTANCE : 0
=================================================================================
ENTRY[000010] = pkt_cnt=0x5176e  [redacted]
ENTRY[000011] = pkt_cnt=0x7d95
ENTRY[000014] = pkt_cnt=0x9d414
ENTRY[000016] = pkt_cnt=0x15208a
ENTRY[000017] = pkt_cnt=0x2975ce
ENTRY[000018] = pkt_cnt=0x662b
ENTRY[000021] = pkt_cnt=0x329f
ENTRY[000023] = pkt_cnt=0x40
ENTRY[000024] = pkt_cnt=0x21bf
ENTRY[000026] = pkt_cnt=0x556f0      [redacted]
ENTRY[000029] = pkt_cnt=0x5d7e2
ENTRY[000041] = pkt_cnt=0x6360
ENTRY[000050] = pkt_cnt=0x2a05
ENTRY[000052] = pkt_cnt=0x5ec    [redacted]
ENTRY[000054] = pkt_cnt=0xdfd
ENTRY[000055] = pkt_cnt=0xd
ENTRY[000068] = pkt_cnt=0xdac
ENTRY[000072] = pkt_cnt=0x91
ENTRY[000077] = pkt_cnt=0x35b
module-1# show platform internal ns table mth_lux_slvz_DHS_SecurityGroupStatTable_memif_data ingress
error opening file
: No such file or directory    [redacted]

=================================================================================
                          TABLE INSTANCE : 0
=================================================================================
ENTRY[000010] = pkt_cnt=0x517cf  [redacted]
ENTRY[000011] = pkt_cnt=0x7d9f
ENTRY[000014] = pkt_cnt=0x9d494
ENTRY[000016] = pkt_cnt=0x152262
ENTRY[000017] = pkt_cnt=0x29799e5
ENTRY[000018] = pkt_cnt=0x6631
ENTRY[000021] = pkt_cnt=0x329f
ENTRY[000023] = pkt_cnt=0x40
ENTRY[000024] = pkt_cnt=0x21c6
ENTRY[000026] = pkt_cnt=0x55771
ENTRY[000029] = pkt_cnt=0x5d7e2      [redacted]
ENTRY[000041] = pkt_cnt=0x64e0
ENTRY[000050] = pkt_cnt=0x2a05
ENTRY[000052] = pkt_cnt=0x5ec    [redacted]
ENTRY[000054] = pkt_cnt=0xdfd
ENTRY[000055] = pkt_cnt=0xd
ENTRY[000068] = pkt_cnt=0xdb8
ENTRY[000072] = pkt_cnt=0x92
ENTRY[000077] = pkt_cnt=0x35b
```

In this example, hardware entry 41 (***ENTRY [000041]***) is incrementing.

Note: The use of this command is not practical in a production environment, but you can use the other commands that are described in this section instead.

Remember the rule (***4132***) and the scope (***268976***):



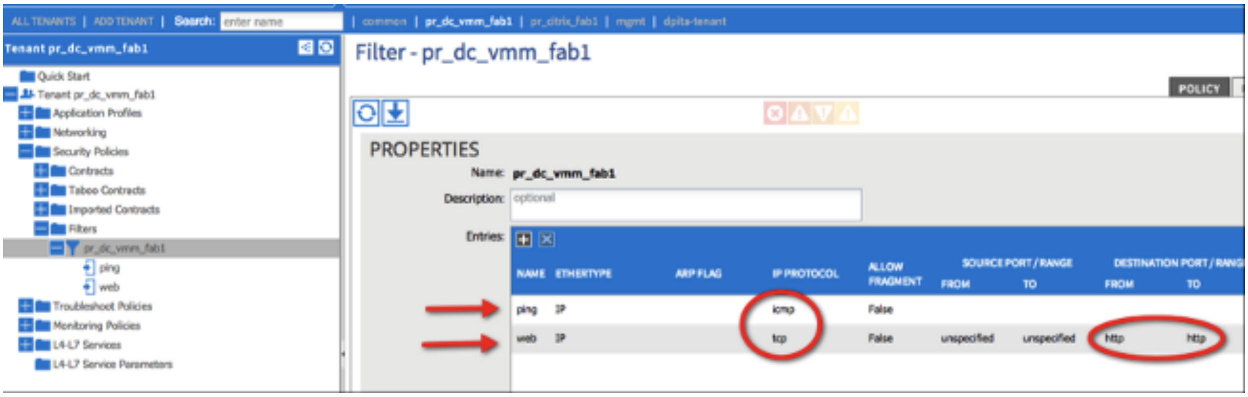
```
4098    0       0       implicit   enabled   2686976   deny,log
4099    0       49154   implicit   enabled   2686976   permit
4131    49155   32771   7          enabled   2686976   permit
4132    32771   49155   6          enabled   2686976   permit
```

Enter this command in order to determine the rule ID to the Ternary Content−Addressable Memory (TCAM) hardware index entry mapping, and filter based on the rule ID and/or filter ID:

```
module-1# show system internal aclqos zoning-rules

[snip]

===========================================
Rule ID: 4131 Scope 4 Src EPG: 49155 Dst EPG: 32771 Filter 7

Curr TCAM resource:
=============================
   unit_id: 0
   === Region priority: 771 (rule prio: 3 entry: 3)===
       sw_index = 62 | hw_index = 40
   === Region priority: 772 (rule prio: 3 entry: 4)===
       sw_index = 63 | hw_index = 45

===========================================
Rule ID: 4132 Scope 4 Src EPG: 32771 Dst EPG: 49155 Filter 6

Curr TCAM resource:
=============================
   unit_id: 0
   === Region priority: 771 (rule prio: 3 entry: 3)===
       sw_index = 66 | hw_index = 41
   === Region priority: 771 (rule prio: 3 entry: 3)===
       sw_index = 67 | hw_index = 42

[snip]
```

For this example, the source and destination EPG combination of interest is ***32771=0x8003, 49155=0xC003***. Therefore, you should consider all of the TCAM entries for these source and destination classes that match the rule IDs (***4131*** and ***4132***) and filter IDs (***6*** and ***7***).

In this example, some of these TCAM entries are dumped. For reference, here is the contract configuration that allows pings and web traffic for these EPGs:



Filter - pr_dc_vmm_fab1

PROPERTIES

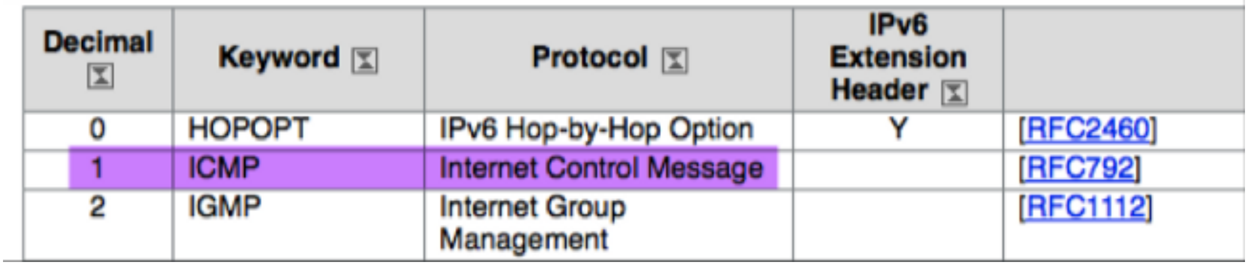
Name: pr_dc_vmm_fab1

Description: optional

Entries:

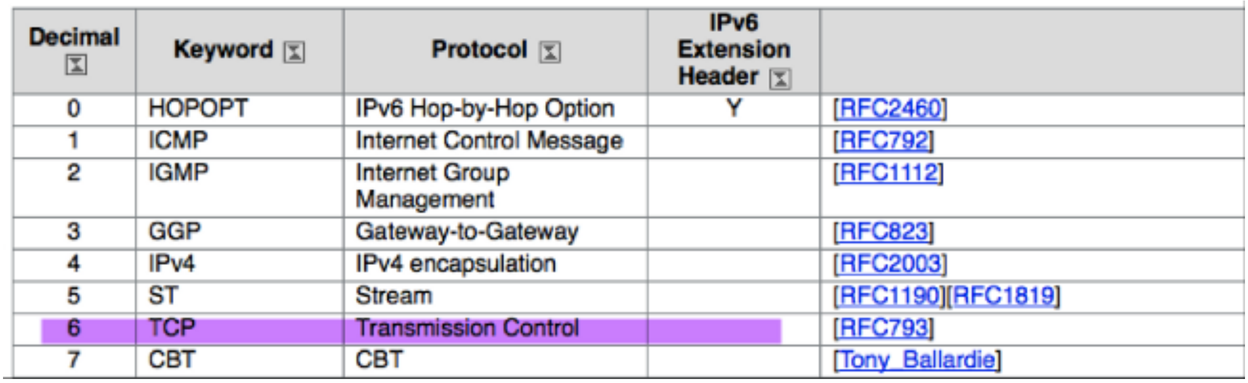
NAME	ETHERTYPE	ARP FLAG	IP PROTOCOL	ALLOW FRAGMENT	SOURCE PORT / RANGE		DESTINATION PORT / RANG	
					FROM	TO	FROM	TO
ping	IP		icmp	False				
web	IP		tcp	False	unspecified	unspecified	http	http

```
module-1# show platform internal ns table mth_lux_slvz_DHS_SecurityGroupKeyTable0
 _memif_data 41

======================================================================
                         TABLE INSTANCE : 0
======================================================================
ENTRY[000041] =
               sg_label=0x4

               sclass=0x8003

               dclass=0xc003

               prot=0x1  (IP Protocol 0x01 = ICMP)
```

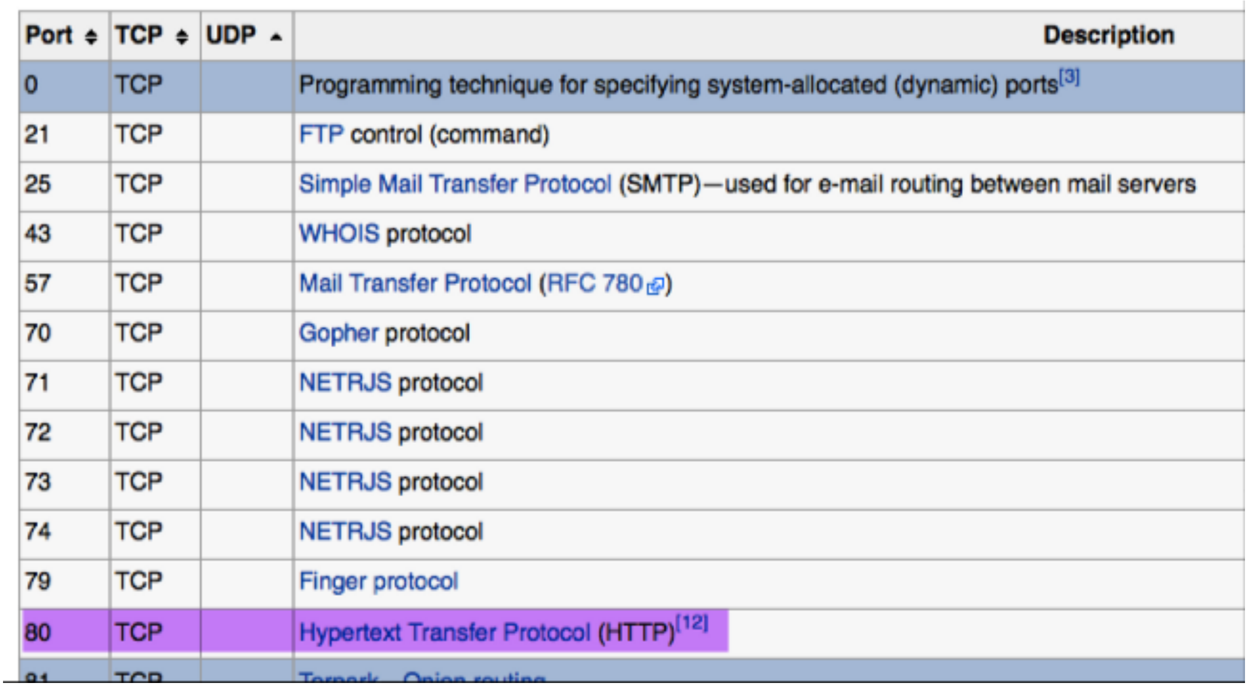


Decimal	Keyword	Protocol	IPv6 Extension Header	
0	HOPOPT	IPv6 Hop-by-Hop Option	Y	[RFC2460]
1	ICMP	Internet Control Message		[RFC792]
2	IGMP	Internet Group Management		[RFC1112]

```
sup_tx_mask=0x1
               src_policy_incomplete_mask=0x1

               dst_policy_incomplete_mask=0x1

               class_eq_mask=0x1

               aclass_mask=0x1ff

               port_dir_mask=0x1

               dport_mask=0xffff

               sport_mask=0xffff

               tcpflags_mask=0xff

               ip_opt_mask=0x1

               ipv6_route_mask=0x1

               ip_fragment_mask=0x1

               ip_frag_offset0_mask=0x1

               ip_frag_offset1_mask=0x1

               ip_mf_mask=0x1

               l4_partial_mask=0x1

               dst_local_mask=0x1

               routeable_mask=0x1

               spare_mask=0x7ff

               v4addr_key_mask=0x1

               v6addr_key_mask=0x1
               valid=0x1


module-1# show platform internal ns table mth_lux_slvz_DHS_SecurityGroupKeyTable0
 _memif_data 42

=======================================================================
                         TABLE INSTANCE : 0
=======================================================================
ENTRY[000042] =

               sg_label=0x4

               sclass=0x8003

               dclass=0xc003

               prot=0x6 <--

               dport=0x50 <--
```



Decimal	Keyword	Protocol	IPv6 Extension Header	
0	HOPOPT	IPv6 Hop-by-Hop Option	Y	[RFC2460]
1	ICMP	Internet Control Message		[RFC792]
2	IGMP	Internet Group Management		[RFC1112]
3	GGP	Gateway-to-Gateway		[RFC823]
4	IPv4	IPv4 encapsulation		[RFC2003]
5	ST	Stream		[RFC1190][RFC1819]
6	TCP	Transmission Control		[RFC793]
7	CBT	CBT		[Tony_Ballardie]



Port	TCP	UDP	Description
0	TCP		Programming technique for specifying system-allocated (dynamic) ports[3]
21	TCP		FTP control (command)
25	TCP		Simple Mail Transfer Protocol (SMTP)—used for e-mail routing between mail servers
43	TCP		WHOIS protocol
57	TCP		Mail Transfer Protocol (RFC 780)
70	TCP		Gopher protocol
71	TCP		NETRJS protocol
72	TCP		NETRJS protocol
73	TCP		NETRJS protocol
74	TCP		NETRJS protocol
79	TCP		Finger protocol
80	TCP		Hypertext Transfer Protocol (HTTP)[12]
81	TCP		Torpark—Onion routing

```
sup_tx_mask=0x1

               src_policy_incomplete_mask=0x1

               dst_policy_incomplete_mask=0x1
```

```
                class_eq_mask=0x1

                aclass_mask=0x1ff

                port_dir_mask=0x1

                sport_mask=0xffff

                tcpflags_mask=0xff

                ip_opt_mask=0x1

                ipv6_route_mask=0x1

                ip_fragment_mask=0x1

                ip_frag_offset0_mask=0x1

                ip_frag_offset1_mask=0x1

                ip_mf_mask=0x1

                l4_partial_mask=0x1

                dst_local_mask=0x1
```

Tip: You can verify each of the TCAM entries with the same method.

Troubleshoot Hardware Programming Issues

This section provides some useful troubleshooting commands and tips.

Useful Troubleshooting Commands

Here are some helpful commands that you can use in order to locate the leaf Policy Manager errors when problems are encountered:

```
fab1_leaf1# show system internal policy-mgr event-history errors


1) Event:E_DEBUG, length:84, at 6132 usecs after Mon Sep 8 13:15:56 2014

   [103] policy_mgr_handle_ctx_mrules(779): ERROR: Failed to process prio(1537):
   (null)


2) Event:E_DEBUG, length:141, at 6105 usecs after Mon Sep 8 13:15:56 2014

   [103] policy_mgr_process_mrule_prio_aces(646): ERROR: Failed to insert iptables
   rule for rule(4120) , fentry(5_0) with priority(1537): (null)


[snip]


fab1_leaf1# show system internal policy-mgr event-histor trace

[1409945922.23737] policy_mgr_ppf_hdl_close_state:562: Got close state callback

[1409945922.23696] policy_mgr_ppf_rdy_ntf_fun:239: StatStoreEnd returned: 0x0(SU

CCESS)
```

```
[1409945922.23502] policy_mgr_ppf_rdy_ntf_fun:208: ppf ready notification: sess_
id: (0xFF0104B400005B51)
[1409945922.23475] policy_mgr_ppf_rdy_ntf_fun:205: Got ready notification callba
ck with statustype (4)
[1409945921.983476] policy_mgr_gwrap_handler:992: Dropped...now purging it...
[1409945921.982882] policy_mgr_ppf_goto_state_fun:481: Sess id (0xFF0104B400005B

[snip]

module-1# show system internal aclqos event-history trace
T [Fri Sep 5 13:18:24.863283] ============= Session End ============
T [Fri Sep 5 13:18:24.862924] Commit phase: Time taken 0.62 ms, usr 0.00 ms,
sys 0.00 ms
T [Fri Sep 5 13:18:24.862302] ppf session [0xff0104b410000087] commit ... npi
nst 1
T [Fri Sep 5 13:18:24.861421] Verify phase: Time taken 0.77 ms, usr 0.00 ms,
sys 0.00 ms
T [Fri Sep 5 13:18:24.860615] ============= Session Begin ============
T [Fri Sep 5 13:18:24.830472] ============= Session End ============
T [Fri Sep 5 13:18:24.830062] Commit phase: Time taken 0.98 ms, usr 0.00 ms,
sys 0.00 ms
T [Fri Sep 5 13:18:24.829085] ppf session [0xff0104b410000086] commit ... npi
nst 1
T [Fri Sep 5 13:18:24.827685] Verify phase: Time taken 2.04 ms, usr 0.00 ms,
sys 0.00 ms
T [Fri Sep 5 13:18:24.825388] ============= Session Begin ============
T [Fri Sep 5 12:32:51.364225] ============= Session End ============
T [Fri Sep 5 12:32:51.363748] Commit phase: Time taken 0.64 ms, usr 0.00 ms,

[snip]
```

Tip: Some of the files are large, so it is easier to send them to the bootflash and examine them in an editor.

```
module-1# show system internal aclqos ?
asic            Asic information
brcm            Broadcam information
database        Database
```

```
event−history   Show various event logs of ACLQOS

mem−stats       Show memory allocation statistics of ACLQOS

prefix          External EPG prefixes

qos             QoS related information

range−resource  Zoning rules L4 destination port range resources

regions         Security TCAM priority regions

span            SPAN related information

zoning−rules    Show zoning rules
```

```
module−1# show system internal aclqos event−history ?

errors        Show error logs of ACLQOS

msgs          Show various message logs of ACLQOS

ppf           Show ppf logs of ACLQOS

ppf−parse     Show ppf-parse logs of ACLQOS

prefix        Show prefix logs of ACLQOS

qos           Show qos logs of ACLQOS

qos−detail    Show detailed qos logs of ACLQOS

span          Show span logs of ACLQOS

span−detail   Show detailed span logs of ACLQOS

trace         Show trace logs of ACLQOS

trace−detail  Show detailed trace logs of ACLQOS

zoning−rules  Show detailed logs of ACLQOS
```

Troubleshooting Tips

Here are some helpful troubleshooting tips:

- If you you seem to experience a TCAM exhaustion problem, check the UI or CLI for faults that are associated with the rule in question. This fault might be reported:

  ```
  Fault F1203 − Rule failed due to hardware programming error.
  ```

  One rule might take more than one TCAM entry in the Application−Specific Integrated Circuit (ASIC). In order to view the number of entries on the ASIC, enter these commands:

  ```
  fab1−leaf1# vsh_lc

  module−1# show platform internal ns table−health
  VLAN STATE curr usage: 0 − size: 4096
  QQ curr usage: 0 − size: 16384
  SEG STATE curr usage: 0 − size: 4096
  SRC TEP curr usage: 0 − size: 4096
  ```

```
POLICY KEY curr usage: 0 - size: 1
SRC VP curr usage: 0 - size: 4096
SEC GRP curr usage: 43 - size: 4096
```

Note: In this example, there are ***43*** entries present. This usage is also reported to the APIC in the *eqptCapacity* class.

- When there are multiple matches, the TCAM lookup returns the ***lower hw−index***. In order to verify the index, enter this command:

  ```
  show system internal aclqos zoning-rule
  ```

  When troubleshooting, you might observe the drop that is caused by the *any−any−implict* rule. This rule is always at the bottom, which means that the packet is dropped because a rule does not exist. This is either due to a misconfiguration, or the Policy Element Manager does not program it as expected.

- The pcTags can have either a *local* or *global* scope:

  - ***Globally scoped pcTag*** This pcTag usually has a lower value (less than four digits in decimal format).

  - ***Locally scoped pcTag*** This pcTag uses a larger value (five digits in decimal format).

When you troubleshoot, a quick look at the length of the value indicates its scope.