Behavior of ACL in PBR on Nexus 7K Containing both L3 and L4 Information

Contents

Introduction

This document describes the behavior of Policy-Based Routing (PBR) on Nexus Switches when you filter based on Layer 3 (L3) and Layer 4 (L4) information.

Background Information

If you add a sequence in PBR in order to match specific L4 information, as a feature N7K creates entries for Access Control Entry (ACEs) and a fragment ACE is created automatically that matches the L3 info specified in the match sequence. In case of fragmented packets, the first packet known as initial fragment contains the L4 header and is matched correctly in the Access Control List (ACL). However, the next fragments known as non-initial fragments do not contain any L4 information and thus if the L3 portion of the ACL entry matches, the non-initial fragment is permitted. So utmost care should be taken, while filtering the traffic based on L4 information, as the non-initial fragments might be wrongly routed in the absence of L4 information.

Topology

The LAN Router is connected to Nexus on interface E2.1, Vlan 700. The requirement is to redirect the traffic that matches Simple Network Management Protocol (SNMP), Web etc. to Optimizer and all other traffic directly in order to interface E2/2 towards Firewall. PBR is configured on Switch Virtual Interface (SVI) Vlan700 on Nexus device. Configuration for the same is provided here. Sequence 70 in the route-map forwards all other traffic to Firewall. There is a new requirement that all the traffic with UDP port 920x needs to go via Optimizer, for this Sequence 50 is added in the route-map.

See here how PBR responds to Fragmented and Non-Fragmented packets that hit in sequence 50 and match both L3 and L4 information.

Here is the configuration on Nexus interface Vlan700 to redirect the traffic that comes on E2/1:

```
interface Vlan700
 no shutdown
 mtu 9000
 vrf member ABC
 no ip redirects
 ip address 10.11.25.25/28
 ip policy route-map In_to_Out


Nexus# show route-map In_to_Out
route-map In_to_Out, permit, sequence 3
 Match clauses:
   ip address (access-lists): Toolbar
 Set clauses:
   ip next-hop 10.3.22.13
route-map In_to_Out, permit, sequence 5
 Match clauses:
   ip address (access-lists): Internet
 Set clauses:
   ip next-hop 10.11.25.19
route-map In_to_Out, permit, sequence 7
 Match clauses:
   ip address (access-lists): Web
 Set clauses:
   ip next-hop 10.11.25.19
route-map In_to_Out, permit, sequence 10
 Match clauses:
   ip address (access-lists): In_to_Out_Internet
 Set clauses:
   ip next-hop 10.11.25.23
route-map In_to_Out, permit, sequence 30
```

```
 Match clauses:
   ip address (access-lists): In_to_Out_www
 Set clauses:
   ip next-hop 10.11.25.23
route-map In_to_Out, permit, sequence 35
 Match clauses:
   ip address (access-lists): In_to_Out_https
 Set clauses:
   ip next-hop 10.11.25.23
route-map In_to_Out, permit, sequence 40
 Match clauses:
   ip address (access-lists): In_to_Out_8080
 Set clauses:
   ip next-hop 10.11.25.23
route-map In_to_Out, permit, sequence 50
 Match clauses:
   ip address (access-lists): UDP_Traffic
 Set clauses:
   ip next-hop 10.11.25.23 >>>>>>>>>>>>>>>>>>>> Towards Optimizer
route-map In_to_Out, permit, sequence 70
 Match clauses:
   ip address (access-lists): To_Firewall
 Set clauses:
    ip next-hop . 10.22.45.63 >>>>>>>>>>>>>>>> Towards Firewall


Nexus# show ip access-lists UDP_Traffic

IP access list UDP_Traffic

10 permit udp any any eq 9201
20 permit udp any any eq 9202

30 permit udp any any eq 9203


Nexus# sh ip access-lists To_Firewall

IP access list To_Firewall
```

```
       10 permit ip any any
```

Once the Policy based routing is configured on SVI, Nexus creates an entry in hardware for the same. Lets now look at the hardware programming for the PBR on module 2 of Nexus:

```
Nexus# show system internal access-list vlan 700 input entries detail module 2

Flags: F - Fragment entry  E - Port Expansion

      D - DSCP Expansion  M - ACL Expansion

      T - Cross Feature Merge Expansion

INSTANCE 0x0

---------------

 Tcam 1 resource usage:

 ----------------------

 Label_b = 0x201

  Bank 0

  ------

    IPv4 Class

      Policies: PBR(GGSN_Toolbar)

      Netflow profile: 0

      Netflow deny profile: 0

      Entries:

        [Index] Entry [Stats]

        ---------------------

 [0019:000f:000f] prec 1 permit-routed ip 0.0.0.0/0 224.0.0.0/4   [0]

 [002d:0024:0024] prec 1 redirect(0x5d)-routed tcp 1.1.22.80/28 0.0.0.0/0 eq 80 flow-label 80
[0]

 [002e:0025:0025] prec 1 redirect(0x5d)-routed tcp 1.1.22.80/28 0.0.0.0/0 fragment   [0]

 [002f:0026:0026] prec 1 redirect(0x5d)-routed tcp 1.1.22.80/28 0.0.0.0/0 eq 8080 flow-label
8080  [0]

 [0030:0027:0027] prec 1 redirect(0x5d)-routed tcp 1.1.22.80/28 0.0.0.0/0 fragment   [0]

 [0031:0028:0028] prec 1 redirect(0x5d)-routed tcp 1.1.22.48/28 0.0.0.0/0 eq 80 flow-label 80
[0]

 [0032:0029:0029] prec 1 redirect(0x5d)-routed tcp 1.1.22.48/28 0.0.0.0/0 fragment   [0]

 [0033:002a:002a] prec 1 redirect(0x5d)-routed tcp 1.1.22.48/28 0.0.0.0/0 eq 8080 flow-label
8080  [0]

 [0034:002b:002b] prec 1 redirect(0x5d)-routed tcp 1.1.22.48/28 0.0.0.0/0 fragment   [0]
```

```
 [0035:002c:002c] prec 1 permit-routed ip 1.1.22.24/29 0.0.0.0/0   [0]

 [0036:002d:002d] prec 1 permit-routed ip 1.1.22.32/28 0.0.0.0/0   [0]

 [0037:002e:002e] prec 1 permit-routed ip 1.1.22.64/28 0.0.0.0/0   [0]

 [0038:002f:002f] prec 1 permit-routed ip 1.1.22.80/28 0.0.0.0/0   [0]

 [003d:0033:0033] prec 1 permit-routed ip 1.1.22.96/28 0.0.0.0/0   [0]

 [003e:0034:0034] prec 1 permit-routed tcp 0.0.0.0/0 196.11.146.149/32 eq 25 flow-label 25  [0]

 [0059:004f:004f] prec 1 permit-routed tcp 0.0.0.0/0 196.11.146.149/32 fragment   [0]

 [005a:0050:0050] prec 1 redirect(0x5e)-routed ip 1.1.22.16/29 0.0.0.0/0   [0]

 [005b:0051:0051] prec 1 redirect(0x5e)-routed tcp 0.0.0.0/0 0.0.0.0/0 eq 80 flow-label 80  [0]

 [005c:0052:0052] prec 1 redirect(0x5e)-routed tcp 0.0.0.0/0 0.0.0.0/0 fragment   [0]

 [005d:0053:0053] prec 1 redirect(0x5e)-routed tcp 0.0.0.0/0 0.0.0.0/0 eq 443 flow-label 443
[0]

 [005e:0054:0054] prec 1 redirect(0x5e)-routed tcp 0.0.0.0/0 0.0.0.0/0 fragment   [0]

 [005f:0055:0055] prec 1 redirect(0x5e)-routed tcp 0.0.0.0/0 0.0.0.0/0 eq 8080 flow-label 8080
[0]

 [0060:0056:0056] prec 1 redirect(0x5e)-routed tcp 0.0.0.0/0 0.0.0.0/0 fragment   [0]

********************Sequence 50 is to match the traffic for UDP ports
9201/9202/9203**********************

 [0061:0057:0057] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 eq 9201 flow-label 9201
[0]

 [0062:0058:0058] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 fragment   [0]

 [0063:0059:0059] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 eq 9202 flow-label 9202
[0]

 [0064:005a:005a] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 fragment   [0]

 [0065:005b:005b] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 eq 9203 flow-label 9203
[0]

 [0066:005c:005c] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 fragment   [0]

********************Sequence 70 is to send all other traffic to Firewall**********************

 [0067:005d:005d] prec 1 permit-routed ip 0.0.0.0/0 0.0.0.0/0   [23]

 [0068:005e:005e] prec 1 permit-routed ip 0.0.0.0/0 0.0.0.0/0   [0]
```

You see that in addition to Access List Entry that matches **udp 0.0.0.0/0 0.0.0.0/0 eq 9201,** there is another entry that matches the fragments **udp 0.0.0.0/0 0.0.0.0/0 fragment** but that entry does not have any UDP port information. This entry is equivalent to any other that matches the UDP packet, so the packets for other UDP ports also get matched in this sequence generated by hardware.

Test Case 1: Traffic Initiated from LAN Router towards Firewall

- The packet that reaches the Nexus was non-fragmented and hence the traffic matched as expected in PBR.
- It was redirected properly to the Firewall and can be seen in debugs run on Firewall.

UDP packet -port 500

```
*Mar 26 04:07:48.959: IP: s=1.1.1.1 (GigabitEthernet0/0), d=3.3.3.3, len 28, rcvd 4  -à
Traffic entering from Nexus interface

*Mar 26 04:07:48.959:     UDP src=500, dst=500
```

TCP packet - port 80

```
*Mar 26 04:07:48.671: IP: s=1.1.1.1 (GigabitEthernet0/1), d=3.3.3.3, len 40, rcvd 4
-à  Traffic entering from Optimizer interface

*Mar 26 04:07:48.671:     TCP src=1720, dst=80, seq=0, ack=0, win=0
```

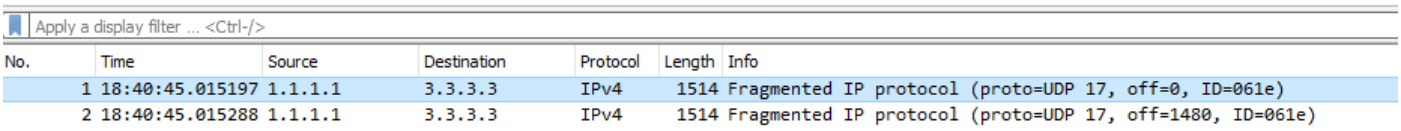
UDP packet -port 9201

```
*Mar 27 09:30:19.879: IP: s=1.1.1.1 (GigabitEthernet0/1), d=3.3.3.3, len 28, input
feature  à Traffic entering from Optimizer interface

*Mar 27 09:30:19.879:     UDP src=6000, dst=9201, MCI Check(80), rtype 0, forus FALSE,
sendself FALSE, mtu 0, fwdchk FALSE
```

Test Case 2: Traffic Initiated via Sniffer File from LAN Router towards Firewall with UDP 500

Traffic with two fragments in the Sniffer File generated here:



1. Initial Fragments with Route-Map:

- The first fragment with **Offset = 0** is known as initial fragment and it contains the UDP header in the packet.
- As the traffic is for UDP 500, it gets matched in sequence 70 to permit **ip any any.**

```
prec 1 permit-routed ip 0.0.0.0/0 0.0.0.0/0   [23]
```

- So the very first packet that has both Layer 3 and Layer 4 information is routed properly.

2. Non-Initial Fragments packets with Route-Map:

- The second fragment with **Offset 0** is known as non-initial fragment and does not contain any UDP header. It is purely IP packet with protocol type UDP (17).
- As there is no Layer 4 information, it matches in sequence 70 : **permit-routed ip 0.0.0.0/0 0.0.0.0/0.**
- However, in sequence 50, there is an Access List that matches traffic for UDP port 920x. The hardware automatically creates an entry to allow the UDP fragments that match the specified Layer 3 information.
- Therefore, every fragmented packet for any Layer 3 information with UDP protocol that is matched in sequence 50.

```
              prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 eq 9201 flow-label 9201  [0]

              prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 fragment [23]
>>>>>>>>>>>>>>>>>>>>
```

- This way, there is one fragment that is routed properly and another routed via wrong sequence.
- The second fragment is modified in order to make **offset = 0**, and it is matched in Sequence 70 as expected.
- This is an expected behavior whenever the Layer 4 fragments are received.
- The intention of creating an extra entry to allow fragments is to permit the non-initial fragments received without Layer 4 information.
- In case, the traffic was for UDP 9201 and there was no entry to allow fragments. Then the second fragment would have matched in Sequence 70 to permit **ip any any** and hence be routed wrongly.

```
Nexus# sh route-map In_to_Out pbr-statistics

route-map In_to_Out, permit, sequence 3

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 5

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 7

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 10

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 30

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 35

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 40
```

```
 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 50 ----------> 2nd Fragment for UDP 500 is matched here

 Policy routing matches: 4397 packets

route-map In_to_Out, permit, sequence 70------------> 1st Fragment for UDP 500 is matched here

 Policy routing matches: 4397 packets
```

- Another sequence 45 is created in order to permit the traffic for UDP 500 and observe that both the fragments are matched in sequence 45.
- The initial fragment matched due to UDP header information and non-initial matched in the fragments line for sequence 45.

```
Nexus# sh route-map In_to_Out pbr-statistics

route-map In_to_Out, permit, sequence 3

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 5

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 7

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 10

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 30

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 35

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 40

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 45----------> Both fragments matched here

 Policy routing matches: 213 packets

route-map In_to_Out, permit, sequence 50

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 70

 Policy routing matches: 0 packets
```

```
Default routing: 0 packets
```

Access List for Sequence 45:

```
Nexus# sh ip access-lists udptraffic

IP access list udptraffic

permit udp any any eq isakmp
```

3. Now lets see how fragments keyword behaves with ACL and Route-Map

- Sequence 5 is applied to permit any random UDP port 56 on the port ACL.

```
Nexus# sh ip access-lists TEST_UDP


IP access list TEST_UDP

statistics per-entry

 5 permit udp any any eq 56 [match=0]

 10 permit udp any any eq isakmp [match=0]

 20 permit ip any any [match=0]
```

- Initiated a traffic stream with fragmented non-initial packet and observed it to be matching in sequence 5. Even though the packet is for UDP 500, it matches in sequence 5 in order to allow UDP 56.

```
Nexus# sh ip access-lists TEST_UDP


IP access list TEST_UDP

statistics per-entry

 5 permit udp any any eq 56 [match=56]

 10 permit udp any any eq isakmp [match=0]

 20 permit ip any any [match=0]
```

- The fragments are denied on the port ACL and it is observed that no packets are matched in the ACL for non-initial as the packet actually gets matched in the entry **udp any any fragments** automatically created by platform.

```
NEXUS# sh ip access-lists TEST_UDP


IP access list TEST_UDP

statistics per-entry
```

```
        fragments deny-all

       5 permit udp any any eq 56 [match=0]

        10 permit udp any any eq isakmp [match=0]

        20 permit ip any any [match=0]
```

```
 [0014:000a:000a] prec 3 permit udp 0.0.0.0/0 0.0.0.0/0 eq 56 flow-label 56  [0]-> Here we are now not seeing any entry to allow UDP fragments

 [0015:000b:000b] prec 3 permit udp 0.0.0.0/0 0.0.0.0/0 eq 500 flow-label 500  [0]

 [0016:000c:000c] prec 3 permit ip 0.0.0.0/0 0.0.0.0/0   [0]

 [0017:000d:000d] prec 3 deny ip 0.0.0.0/0 0.0.0.0/0 fragment   [100]>> Getting matched in fragments deny statement

 [001e:0014:0014] prec 3 deny ip 0.0.0.0/0 0.0.0.0/0   [0]
```

- Denied the fragments in problematic ACL in PBR, however this workaround did not work and packets are still seen to match in both the sequence 50 and 70. This is due to programming behavior of Access list and Route-map.

```
          NEXUS# sh ip access-lists UDP_Traffic

       IP access list UDP_Traffic

       statistics per-entry

        fragments deny-all

       10 permit udp any any eq 9201

        20 permit udp any any eq 9202

        30 permit udp any any eq 9203
```

```
[0061:0057:0057] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 eq 9201 flow-label 9201 [0]

[0062:0058:0058] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 fragment    [8027]

 [0063:0059:0059] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 eq 9202 flow-label 9202 [0]

 [0064:005a:005a] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 fragment    [0]

 [0065:005b:005b] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 eq 9203 flow-label 9203 [0]

 [0066:005c:005c] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 fragment    [0]

  [0067:005d:005d] prec 1 permit-routed ip 0.0.0.0/0 0.0.0.0/0   [8027]

 [0068:005e:005e] prec 1 permit-routed ip 0.0.0.0/0 0.0.0.0/0   [0]
```

- Outputs when fragments deny is applied on both port ACL and PBR ACL:

[0061:0057:0057] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 eq 9201 flow-label 9201 [0]

[0062:0058:0058] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 fragment [8027] --- > **Once the fragments are denied in port CAL, we observed non-initial packets to be getting dropped (See the mismatch in number of packets between UDP and IP counter)**

[0063:0059:0059] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 eq 9202 flow-label 9202 [0]

[0064:005a:005a] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 fragment [0]

[0065:005b:005b] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 eq 9203 flow-label 9203 [0]

[0066:005c:005c] prec 1 redirect(0x5e)-routed udp 0.0.0.0/0 0.0.0.0/0 fragment [0]

[0067:005d:005d] prec 1 permit-routed ip 0.0.0.0/0 0.0.0.0/0 [8214]

[0068:005e:005e] prec 1 permit-routed ip 0.0.0.0/0 0.0.0.0/0 [0]

VDC-1 Ethernet2/1 :

==========

INSTANCE 0x0

Tcam 0 resource usage:

Label_a = 0x200

Bank 0

IPv4 Class

Policies: PACL(TEST_UDP)

Netflow profile: 0

Netflow deny profile: 0

Entries:

[Index] Entry [Stats]

[0014:000a:000a] prec 3 permit udp 0.0.0.0/0 0.0.0.0/0 eq 56 flow-label 56 [8027]

[0015:000b:000b] prec 3 permit udp 0.0.0.0/0 0.0.0.0/0 eq 500 flow-label 500 [8214]

[0016:000c:000c] prec 3 permit ip 0.0.0.0/0 0.0.0.0/0 [0]

```
 [0017:000d:000d] prec 3 deny ip 0.0.0.0/0 0.0.0.0/0 fragment   [100]

 [001e:0014:0014] prec 3 deny ip 0.0.0.0/0 0.0.0.0/0   [0]
```

There are several possible ways to overcome this problem or limitation of fragmented packets with L4 information:

- Route-map can be tweaked in order to allow specific L3 information for particular UDP ports.

In the current configuration, if L3 source and destination information is mentioned then the non-initial packet is routed based on that specific information. However this is useful only when there is no other sequence before it matches the same L3 information.

```
Nexus# show ip access-lists UDP_Traffic

IP access list UDP_Traffic

10 permit udp host 1.1.1.1 host 3.3.3.3 eq 9201
20 permit udp any any eq 9202

30 permit udp any any eq 9203
```

- Path from source to destination can be verified in order to check the MTU so that packet does not get fragmented.
- The workaround of applying another sequence allows UDP above the problematic sequence to work, however, the behavior is same as explained earlier when sequence 45 was applied

```
Nexus# sh route-map In_to_Out pbr-statistics

route-map In_to_Out, permit, sequence 3

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 5

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 7

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 10

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 30

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 35

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 40
```

```
 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 45----------> Both fragments matched here

 Policy routing matches: 213 packets

route-map In_to_Out, permit, sequence 50

 Policy routing matches: 0 packets

route-map In_to_Out, permit, sequence 70

 Policy routing matches: 0 packets
```

Access list for Sequence 45:

```
Nexus# sh ip access-lists udptraffic
```

IP access list udptraffic:

```
permit udp any any eq isakmp
```

Doc Bug: CSCve05428 N7K Doc bug || ACL in PBR that contains both L3 and L4 information.