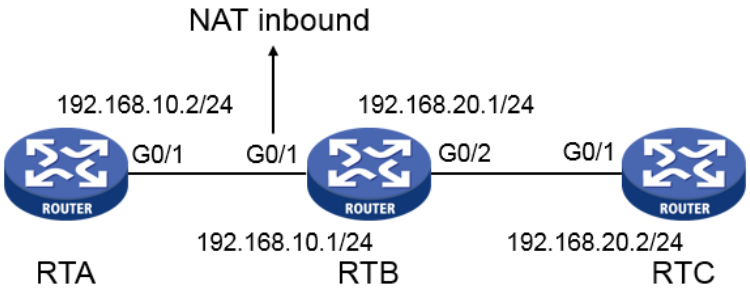
路由器NAT inbound入方向动态地址转换经验案例

NAT 葛松炜 2019-11-28 发表

组网及说明

入方向动态地址转换功能通常与接口上的出方向动态地址转换（nat outbound）、内部服务器（nat server）或出方向静态地址转换（nat static outbound）配合，用于实现双向NAT应用，不建议单独使用。

之所以不建议单独使用，是因为nat inbound在单独使用时无法实现同outbound一样，在公网设备没有私网路由的时候，内网设备也能ping通外网。nat inbound动态地址转换在单独使用时，一般应用于内网环境，在访问某设备时，如果想隐藏自己原本的IP地址，让目的地址看来并不是本来的源地址发起的访问，这种环境下可以使用nat inbound实现。有的局点不愿意使用nat outbound，想要通过nat inbound实现流量入方向进行动态地址的转换。



问题描述

例如如上环境，现场想在RTA经过RTB访问RTC时，在RTB的入方向接口G0/1进行入方向的动态地址转换，在RTC上观察报文，源地址为nat inbound转换后的地址。

过程分析

RTB NAT inbound相关配置：

```
#
interface GigabitEthernet0/1
 port link-mode route
 combo enable copper
 ip address 192.168.10.1 255.255.255.0
 nat inbound 3000 address-group 1
#
#
acl advanced 3000
 rule 0 permit ip
#
nat address-group 1
 address 192.168.30.1 192.168.30.1      //配置NAT地址组地址，前后分别为起始地址和终止地址
#
```

在RTA和RTC上分别配置默认路由，指向RTB（RTC只要有到NAT地址组地址的路由，指向RTB即可），但是配置完路由后发现RTA无法ping通RTC：

```
[RTA]ping 192.168.20.2
Ping 192.168.20.2 (192.168.20.2): 56 data bytes, press CTRL_C to break
Request time out
Request time out
Request time out
Request time out
Request time out

--- Ping statistics for 192.168.20.2 ---
5 packet(s) transmitted, 0 packet(s) received, 100.0% packet loss
```

此时在RTB上通过display nat session verbose命令查看NAT会话（注意需要在设备系统视图下通过ses

sion statistics enable命令打开会话计数统计）

```
[RTB]display nat session verbose
Slot 0:
Initiator:
  Source      IP/port: 192.168.10.2/219
  Destination IP/port: 192.168.20.2/2048
  DS-Lite tunnel peer: -
  VPN instance/VLAN ID/Inline ID: -/-/-
  Protocol: ICMP(1)
  Inbound interface: GigabitEthernet0/1
Responder:
  Source      IP/port: 192.168.20.2/6
  Destination IP/port: 192.168.30.1/0
  DS-Lite tunnel peer: -
  VPN instance/VLAN ID/Inline ID: -/-/-
  Protocol: ICMP(1)
  Inbound interface: GigabitEthernet0/2
State: ICMP_REQUEST
Application: OTHER
Role: -
Failover group ID: -
Start time: 2019-11-28 20:29:10    TTL: 57s
Initiator->Responder:          5 packets       420 bytes
Responder->Initiator:          0 packets         0 bytes
Total sessions found: 1
```

通过会话发现，在RTB的入方向上确实进行了地址转换，RTC在回包时的目的地址并不是192.168.10.2，而是转换后的地址192.168.30.1。但是观察报文时我们发现，RTB上可以记录到初始方向报文，但是回包却没有计数，这是因为在配置入方向动态地址转换时有这样一条限制：

对于入方向动态地址转换，如果指定了add-route参数，则有报文命中该配置时，设备会自动添加路由表项：目的地址为本次地址转换使用的地址组中的地址，出接口为本配置所在接口，下一跳地址为报文的源地址；如果没有指定add-route参数，则用户需要在设备上手工添加路由。由于自动添加路由表项速度较慢，通常建议手工添加路由。

解决方法

根据限制描述，我们有如下两种方法在NAT inbound设备上添加路由：

方法一：手动添加路由，目的地址为转换地址组中的地址，出接口为本配置所在接口，下一跳地址为报文的源地址

[RTB]ip route-static 192.168.30.1 32 GigabitEthernet 0/1 192.168.10.2

方法二：在nat inbound后面指定add-route参数，有报文命中该配置时，设备会自动添加对应的路由表项

[RTB-GigabitEthernet0/1]nat inbound 3000 address-group 1 no-pat add-route

指定add-route参数，RTA发起ping RTC后，display ip routing-table能够看到自动添加的表项：

```
[RTB]display ip routing-table
Destinations : 17       Routes : 17
Destination/Mask   Proto   Pre Cost       NextHop         Interface
0.0.0.0/32         Direct  0   0          127.0.0.1       InLoop0
127.0.0.0/8        Direct  0   0          127.0.0.1       InLoop0
127.0.0.0/32       Direct  0   0          127.0.0.1       InLoop0
127.0.0.1/32       Direct  0   0          127.0.0.1       InLoop0
127.255.255.255/32 Direct  0   0          127.0.0.1       InLoop0
192.168.10.0/24    Direct  0   0          192.168.10.1    GE0/1
192.168.10.0/32    Direct  0   0          192.168.10.1    GE0/1
192.168.10.1/32    Direct  0   0          127.0.0.1       InLoop0
192.168.10.255/32  Direct  0   0          192.168.10.1    GE0/1
192.168.20.0/24    Direct  0   0          192.168.20.1    GE0/2
192.168.20.0/32    Direct  0   0          192.168.20.1    GE0/2
192.168.20.1/32    Direct  0   0          127.0.0.1       InLoop0
192.168.20.255/32  Direct  0   0          192.168.20.1    GE0/2
192.168.30.1/32    Static  1   0          192.168.10.2    GE0/1
```

224.0.0.0/4 Direct 0 0 0.0.0.0 NULL0
224.0.0.0/24 Direct 0 0 0.0.0.0 NULL0
255.255.255.255/32 Direct 0 0 127.0.0.1 InLoop0

但是在ping包时发现，自动添加表项的速度较慢，没有路由表项或路由表项老化后首次发起ping报文时，ping包时因为没有迅速下发表项而导致了丢包，这也是为什么推荐手工添加路由

[RTA]ping 192.168.20.2 //没有路由表项时，自动添加路由表项会产生少量丢包
Ping 192.168.20.2 (192.168.20.2): 56 data bytes, press CTRL_C to break
Request time out
56 bytes from 192.168.20.2: icmp_seq=1 ttl=254 time=1.638 ms
56 bytes from 192.168.20.2: icmp_seq=2 ttl=254 time=1.588 ms
56 bytes from 192.168.20.2: icmp_seq=3 ttl=254 time=1.201 ms
56 bytes from 192.168.20.2: icmp_seq=4 ttl=254 time=1.749 ms

添加路由后在RTB上再次查看NAT会话，地址转换和来回报文计数都正常：
[RTB]display nat session verbose
Slot 0:
Initiator:
Source IP/port: 192.168.10.2/233
Destination IP/port: 192.168.20.2/2048
DS-Lite tunnel peer: -
VPN instance/VLAN ID/Inline ID: -/-/-
Protocol: ICMP(1)
Inbound interface: GigabitEthernet0/1
Responder:
Source IP/port: 192.168.20.2/233
Destination IP/port: 192.168.30.1/0
DS-Lite tunnel peer: -
VPN instance/VLAN ID/Inline ID: -/-/-
Protocol: ICMP(1)
Inbound interface: GigabitEthernet0/2
State: ICMP_REPLY
Application: OTHER
Role: -
Failover group ID: -
Start time: 2019-11-28 21:25:30 TTL: 23s
Initiator->Responder: 5 packets 420 bytes
Responder->Initiator: 5 packets 420 bytes
在RTC入接口上抓包，可以看到源地址成功转换为了192.168.30.1：

93	46661.288072	192.168.30.1	192.168.20.2	ICMP	98	Echo (ping) request id=0x0001, seq=0/0, ttl=254 (reply in 94)
94	46661.288896	192.168.20.2	192.168.30.1	ICMP	98	Echo (ping) reply id=0x0001, seq=0/0, ttl=255 (request in 93)
95	46661.454905	192.168.30.1	192.168.20.2	ICMP	98	Echo (ping) request id=0x0001, seq=1/256, ttl=254 (reply in 96)
96	46661.455254	192.168.20.2	192.168.30.1	ICMP	98	Echo (ping) reply id=0x0001, seq=1/256, ttl=255 (request in 95)
97	46661.623773	192.168.30.1	192.168.20.2	ICMP	98	Echo (ping) request id=0x0001, seq=2/512, ttl=254 (reply in 98)
98	46661.624195	192.168.20.2	192.168.30.1	ICMP	98	Echo (ping) reply id=0x0001, seq=2/512, ttl=255 (request in 97)
99	46661.804331	192.168.30.1	192.168.20.2	ICMP	98	Echo (ping) request id=0x0001, seq=3/768, ttl=254 (reply in 100)
100	46661.804607	192.168.20.2	192.168.30.1	ICMP	98	Echo (ping) reply id=0x0001, seq=3/768, ttl=255 (request in 99)
101	46661.974859	192.168.30.1	192.168.20.2	ICMP	98	Echo (ping) request id=0x0001, seq=4/1024, ttl=254 (reply in 102)
102	46661.975205	192.168.20.2	192.168.30.1	ICMP	98	Echo (ping) reply id=0x0001, seq=4/1024, ttl=255 (request in 101)