策略路由及其对Cisco IOS的ESP和ISAKMP数据包的影响

目录

简介

本文档介绍在使用Cisco IOS®时，应用于封装安全负载(ESP)和互联网安全关联和密钥管理协议(ISAKMP)数据包时，策略型路由(PBR)和本地PBR的效果。

作者：思科TAC工程师Michal Garcarz。

先决条件

要求

Cisco 建议您具有以下主题的基础知识：

- Cisco IOS
- Cisco IOS上的VPN配置

使用的组件

本文档中的信息基于Cisco IOS版本15.x。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您使用的是真实网络，请确保您已经了解所有命令的潜在影响。

背景信息

在建立IPsec隧道之前，路由器会启动ISAKMP交换。由于这些数据包是由路由器生成的，因此这些数据包被视为本地生成的流量，并应用任何本地PBR决策。此外，路由器(增强型内部网关路由协议(EIGRP)、下一跳解析协议(NHRP)、边界网关协议(BGP)或互联网控制消息协议(ICMP)ping)生成的任何数据包也被视为本地生成的流量，并应用本地PBR决策。

路由器转发并通过隧道发送的流量（称为中转流量）不被视为本地生成的流量，并且任何所需的路由策略都必须应用于路由器的入口接口。

这对通过隧道的流量的影响是本地生成的流量遵循PBR，但中转流量不遵循。本文解释了这种行为差异的后果。

对于需要ESP封装的中转流量，不需要任何路由条目，因为PBR在ESP封装前后确定数据包的出口接口。对于需要封装ESP的本地生成流量，必须包含路由条目，因为本地PBR在封装之前仅为数据包确定出口接口，而路由则为封装后的数据包确定出口接口。

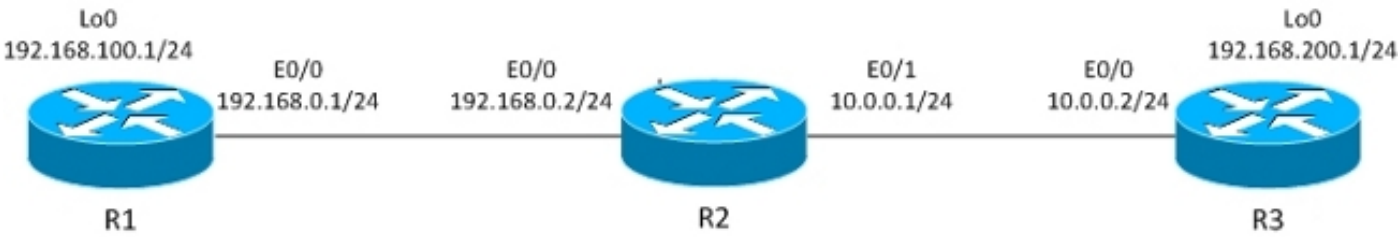
本文档包含一个典型的配置示例，其中使用一台带有两条ISP链路的路由器。一条链路用于访问Internet，另一条链路用于VPN。如果有任何链路故障，流量将通过不同的Internet服务提供商(ISP)链路重新路由。同时也提出了缺陷。

请注意，PBR在思科快速转发(CEF)中执行，而本地PBR是进程交换的。

路由器上本地生成的流量

本节介绍从路由器(R)1发起的流量的行为。该流量由R1封装。

拓扑



R1和R3之间建立IPsec LAN到LAN隧道。

相关流量在R1 Lo0(192.168.100.1)和R3 Lo0(192.168.200.1)之间。

R3路由器有到R2的默认路由。

R1没有路由条目，只有直连网络。

配置

R1对所有流量都有本地PBR:

```
interface Loopback0
 ip address 192.168.100.1 255.255.255.0
!
interface Ethernet0/0
 ip address 192.168.0.1 255.255.255.0
 crypto map CM

track 10 ip sla 10
ip sla 10
 icmp-echo 192.168.0.2 source-ip 192.168.0.1

route-map LOCALPBR permit 10
 set ip next-hop verify-availability 192.168.0.2 1 track 10
ip local policy route-map LOCALPBR
```

调试

当R1处于UP状态时，所有本地生成的流量都会发送到R2。

要验证启用隧道时发生的情况，请发送来自路由器本身的相关流量：

```
R1#debug ip packet
R1#ping 192.168.200.1 source lo0
```

> **警告：debug ip packet**命令可能会生成大量调试，并对CPU使用率产生巨大影响。谨慎使用。

此调试还允许使用访问列表以限制调试处理的流量。**debug ip packet**命令仅显示进程交换的流量。

以下是R1上的调试：

```
IP: s=192.168.100.1 (local), d=192.168.200.1 (Ethernet0/0), len 100, local
```

```
feature, Policy Routing(3), rtype 2, forus FALSE, sendself FALSE, mtu 0, fwdchk
FALSE
IP: s=192.168.100.1 (local), d=192.168.200.1 (Ethernet0/0), len 100, sending
IP: s=192.168.100.1, d=192.168.200.1, pak EF6E8F28 consumed in output feature,
packet consumed, IPSec output classification(30), rtype 2, forus FALSE, sendself
FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, local feature, Policy
Routing(3), rtype 2, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, sending
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, output feature,
IPSec output classification(30), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, output feature,
IPSec: to crypto engine(64), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, output feature,
Post-encryption output features(65), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, post-encap feature,
(1), rtype 2, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, post-encap feature,
FastEther Channel(3), rtype 2, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, sending full packet
```

接下来发生的情况是：

相关流量(192.168.100.1 > 192.168.200.1)由本地PBR匹配，并确定出口接口(E0/0)。 此操作会触发加密代码以启动ISAKMP。该数据包也由本地PBR策略路由，该PBR确定出口接口(E0/0)。 发送ISAKMP流量，并协商隧道

当您再次ping时会发生什么情况？

```
R1#show crypto session
Crypto session current status

Interface: Ethernet0/0
Session status: UP-ACTIVE
Peer: 10.0.0.2 port 500
 IKEv1 SA: local 192.168.0.1/500 remote 10.0.0.2/500 Active
 IPSEC FLOW: permit ip host 192.168.100.1 host 192.168.200.1
       Active SAs: 2, origin: crypto map

R1#ping 192.168.200.1 source lo0 repeat 1

IP: s=192.168.100.1 (local), d=192.168.200.1 (Ethernet0/0), len 100, local
feature, Policy Routing(3), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.100.1 (local), d=192.168.200.1 (Ethernet0/0), len 100, sending
IP: s=192.168.100.1 (local), d=192.168.200.1 (Ethernet0/0), len 100, output
feature, IPSec output classification(30), rtype 2, forus FALSE, sendself FALSE,
mtu 0, fwdchk FALSE
IP: s=192.168.100.1, d=192.168.200.1, pak EEB40198 consumed in output feature,
packet consumed, IPSec: to crypto engine(64), rtype 2, forus FALSE, sendself
FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 172, output feature,
IPSec output classification(30), rtype 1, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 172, output feature,
IPSec: to crypto engine(64), rtype 1, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 172, output feature,
Post-encryption output features(65), rtype 1, forus FALSE, sendself FALSE,
```

```
mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), g=10.0.0.2, len 172,
forward
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 172, post-encap
feature, (1), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 172, post-encap
feature, FastEther Channel(3), rtype 0, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 172, encapsulation
failed.
Success rate is 0 percent (0/1)
```

接下来发生的情况是：

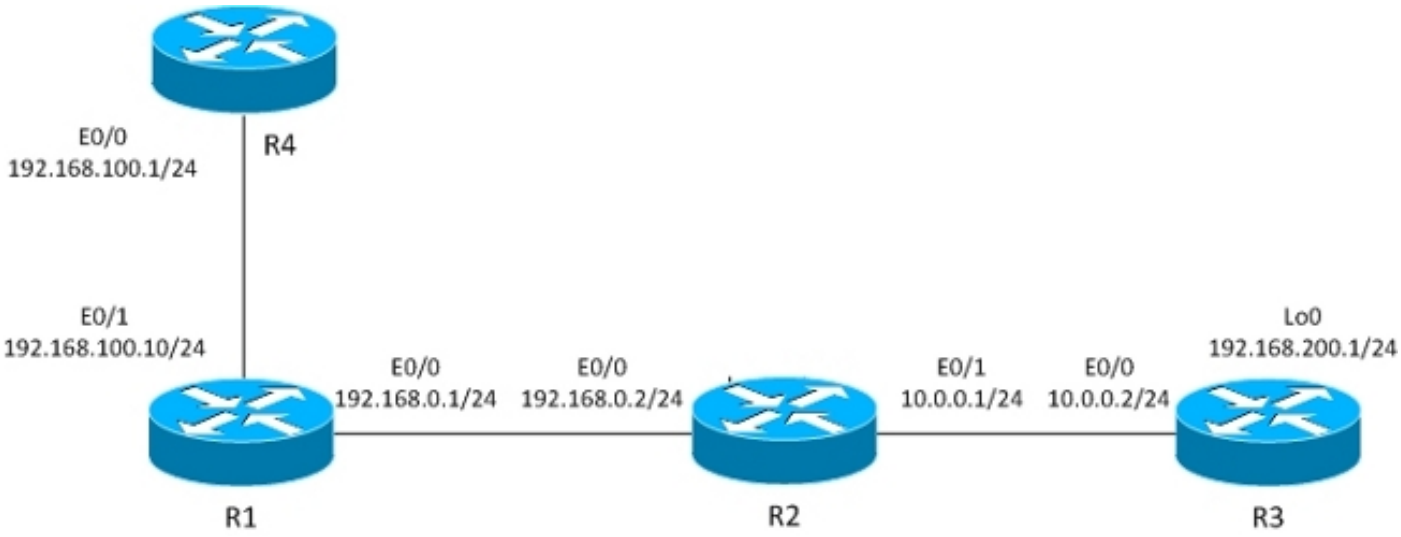
本地生成的相关流量(192.168.100.1 > 192.168.200.1)在本地策略路由，并确定出口接口(E0/0)。 数据包由E0/0上的IPsec输出功能使用并封装。为了确定出口接口，会检查封装的数据包（从192.168.0.1到10.0.0.2）的路由，但R1的路由表中没有任何内容，这就是封装失败的原因。

在此场景中，隧道为UP状态，但流量不会发送，因为在ESP封装后，Cisco IOS检查路由表以确定出口接口。

通过路由器的中转流量

本节介绍通过路由器（由该路由器封装的ESP）的中转流量的行为。

拓扑



L2L隧道在R1和R3之间建立。

相关流量在R4(192.168.100.1)和R3 lo0(192.168.200.1)之间。

R3路由器有到R2的默认路由。

R4路由器有到R1的默认路由。

R1没有路由。

配置

修改上一拓扑是为了显示路由器收到数据包进行加密时的流量（中转流量而不是本地生成的流量）。

现在，从R4收到的相关流量在R1上（通过E0/1上的PBR）进行策略路由，并且所有流量也有本地策略路由：

```
interface Ethernet0/1
 ip address 192.168.100.10 255.255.255.0
 ip policy route-map PBR

route-map LOCALPBR permit 10
 set ip next-hop verify-availability 192.168.0.2 1 track 10
!
route-map PBR permit 10
 set ip next-hop verify-availability 192.168.0.2 1 track 10

ip local policy route-map LOCALPBR
```

调试

要验证在R1上启动隧道时（在收到来自R4的相关流量后）会发生什么情况，请输入：

```
R1#debug ip packet

R4#ping 192.168.200.1
```

以下是R1上的调试：

```
IP: s=192.168.100.1 (Ethernet0/1), d=192.168.200.1 (Ethernet0/0), len 100,
input feature, Policy Routing(68), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.100.1 (Ethernet0/1), d=192.168.200.1 (Ethernet0/0), len 100,
input feature, MCI Check(73), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.100.1, d=192.168.200.1, pak EEB4A9D8 consumed in output feature,
packet consumed, IPSec output classification(30), rtype 2, forus FALSE,
sendself FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, local feature,
Policy Routing(3), rtype 2, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, sending
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, output feature,
IPSec output classification(30), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, output feature,
IPSec: to crypto engine(64), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, output feature,
Post-encryption output features(65), rtype 2, forus FALSE, sendself FALSE,
mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, post-encap
feature, (1), rtype 2, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, post-encap
feature, FastEther Channel(3), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (local), d=10.0.0.2 (Ethernet0/0), len 192, sending full
```

packet
接下来发生的情况是：

相关流量在E0/0上命中PBR并触发加密代码以发送ISAKMP数据包。该ISAKMP数据包在本地策略路由，而出口接口由本地PBR确定。隧道已建立。

以下是从R4对192.168.200.1执行的另一次ping操作：

```
R4#ping 192.168.200.1
```

以下是R1上的调试：

```
IP: s=192.168.100.1 (Ethernet0/1), d=192.168.200.1 (Ethernet0/0), len 100,
input feature, Policy Routing(68), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.100.1 (Ethernet0/1), d=192.168.200.1 (Ethernet0/0), len 100,
input feature, MCI Check(73), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.100.1 (Ethernet0/1), d=192.168.200.1 (Ethernet0/0), len 100,
output feature, IPSec output classification(30), rtype 2, forus FALSE, sendself
FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.100.1, d=192.168.200.1, pak EF722068 consumed in output feature,
packet consumed, IPSec: to crypto engine(64), rtype 2, forus FALSE, sendself
FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (Ethernet0/1), d=10.0.0.2 (Ethernet0/0), len 172, input
feature, Policy Routing(68), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (Ethernet0/1), d=10.0.0.2 (Ethernet0/0), len 172, input
feature, MCI Check(73), rtype 2, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (Ethernet0/1), d=10.0.0.2 (Ethernet0/0), len 172, output
feature, IPSec output classification(30), rtype 2, forus FALSE, sendself FALSE,
mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (Ethernet0/1), d=10.0.0.2 (Ethernet0/0), len 172, output
feature, IPSec: to crypto engine(64), rtype 2, forus FALSE, sendself FALSE,
mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (Ethernet0/1), d=10.0.0.2 (Ethernet0/0), len 172, output
feature, Post-encryption output features(65), rtype 2, forus FALSE, sendself
FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (Ethernet0/1), d=10.0.0.2 (Ethernet0/0), g=192.168.0.2, len
172, forward
IP: s=192.168.0.1 (Ethernet0/1), d=10.0.0.2 (Ethernet0/0), len 172, post-encap
feature, (1), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
IP: s=192.168.0.1 (Ethernet0/1), d=10.0.0.2 (Ethernet0/0), len 172, post-encap
feature, FastEther Channel(3), rtype 0, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
IP: s=192.168.0.1 (Ethernet0/1), d=10.0.0.2 (Ethernet0/0), len 172,
sending full packet
```

接下来发生的情况是：

相关流量在E0/0上到达PBR，而PBR确定出口接口(E0/0)。 在E0/0上，数据包由IPSec使用并封装。根据同一PBR规则检查封装的数据包并确定出口接口后，将正确发送和接收数据包。

行为差异摘要

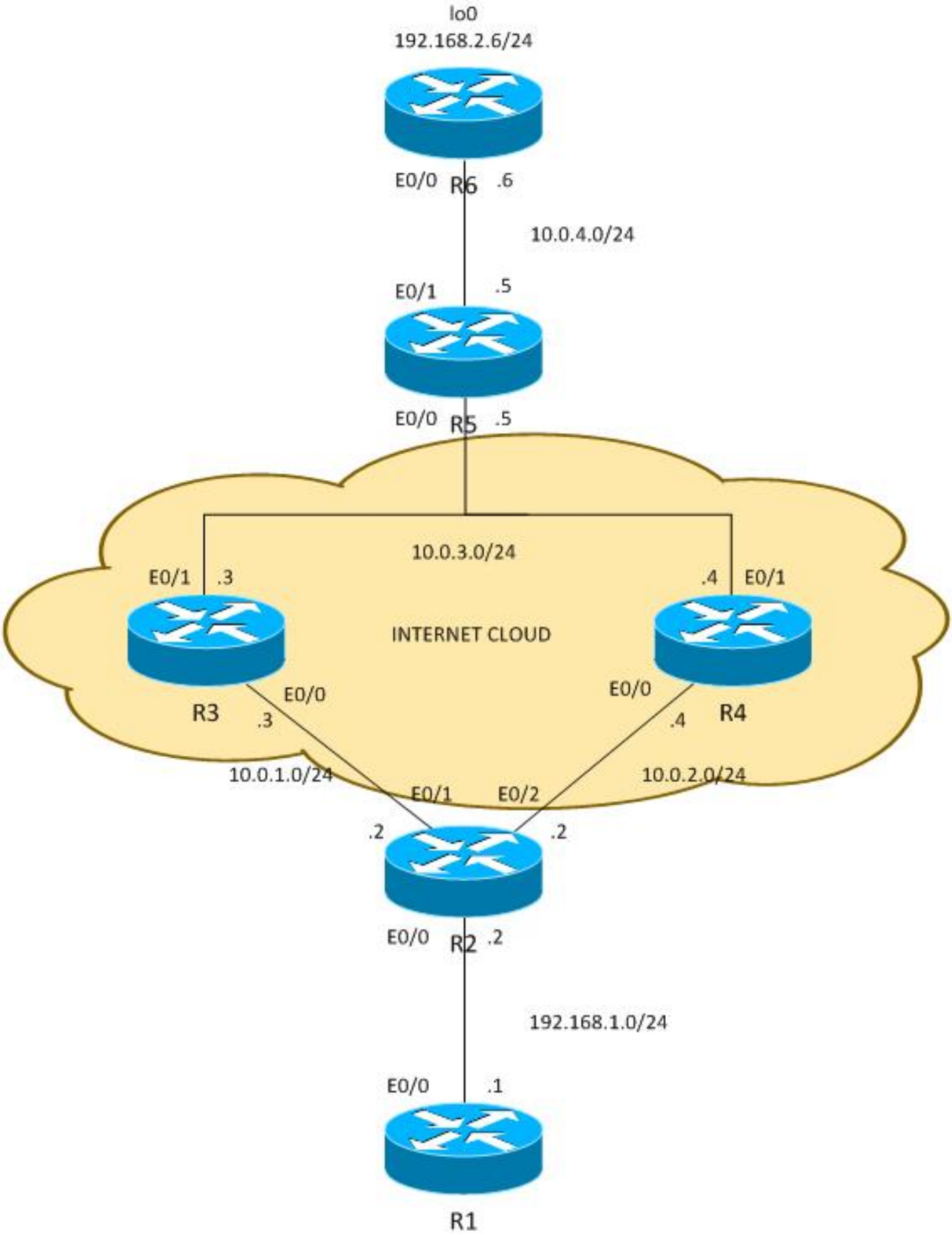
对于本地生成的流量，非封装流量(ISAKMP)的出口接口由本地PBR确定。对于本地生成的流量，后

封装流量(ESP)的出口接口由路由表确定（未选中本地PBR）。 对于中转流量，后封装流量(ESP)的出口接口由接口PBR确定（封装前后两次）。

配置示例

这是一个实用配置示例，它展示了您在PBR和本地PBR与VPN时可能遇到的问题。R2(CE)有两条ISP链路。R6路由器还有CE和一条ISP链路。从R2到R3的第一条链路用作R2的默认路由。到R4的第二条链路仅用于到R6的VPN流量。如果ISP链路发生故障，流量将重新路由到另一条链路。

拓扑



配置

192.168.1.0/24和192.168.2.0/24之间的流量受到保护。开放最短路径优先(OSPF)用于Internet云中通告10.0.0.0/8地址，这些地址被视为ISP分配给客户的公有地址。在现实中，BGP代替OSPF。

R2和R6上的配置基于加密映射。在R2上，E0/0上使用PBR，以便在UP时将VPN流量定向到R4:

```
route-map PBR permit 10
 match ip address cmap
 set ip next-hop verify-availability 10.0.2.4 1 track 20

ip access-list extended cmap
 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255

crypto map cmap 10 ipsec-isakmp
 set peer 10.0.4.6
 set transform-set TS
 match address cmap

interface Ethernet0/0
 ip address 192.168.1.2 255.255.255.0
 ip nat inside
 ip virtual-reassembly in
 ip policy route-map PBR
```

您会看到不需要本地PBR。接口PBR将相关流量路由到10.0.2.4。这会触发加密代码，从正确的接口（到R4的链路）启动ISAKMP，即使路由是通过R3到远程对等点时也是如此。

在R6上，使用两个VPN对等点：

```
crypto map cmap 10 ipsec-isakmp
 set peer 10.0.2.2             !primary
 set peer 10.0.1.2
 set transform-set TS
 match address cmap
```

R2使用IP服务等级协议(SLA)对R3和R4执行ping操作。默认路由为R3。如果R3发生故障，它会选择R4:

```
ip sla 10
 icmp-echo 10.0.1.3
ip sla schedule 10 life forever start-time now
ip sla 20
 icmp-echo 10.0.2.4
ip sla schedule 20 life forever start-time now

track 10 ip sla 10
track 20 ip sla 20

ip route 0.0.0.0 0.0.0.0 10.0.1.3 track 10
ip route 0.0.0.0 0.0.0.0 10.0.2.4 100
```

此外，R2允许所有内部用户访问Internet。为了在ISP到R3关闭的情况下实现冗余，需要路由映射。R3处于UP状态且默认路由指向R3时，它会将内部流量端口地址转换(PAT)转换到不同出口接口（PAT到E0/1接口，当R3关闭且R4用作默认路由时，它会将PAT转换到接口E0/2）。

```
ip access-list extended pat
 deny   ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
 deny   udp any any eq isakmp
 deny   udp any eq isakmp any
 permit ip any any

route-map RMAP2 permit 10
 match ip address pat
```

```
 match interface Ethernet0/2
!
route-map RMAP1 permit 10
 match ip address pat
 match interface Ethernet0/1

ip nat inside source route-map RMAP1 interface Ethernet0/1 overload
ip nat inside source route-map RMAP2 interface Ethernet0/2 overload

interface Ethernet0/0
 ip address 192.168.1.2 255.255.255.0
 ip nat inside
 ip virtual-reassembly in
 ip policy route-map PBR

interface Ethernet0/1
 ip address 10.0.1.2 255.255.255.0
 ip nat outside
 ip virtual-reassembly in
 crypto map cmap

interface Ethernet0/2
 ip address 10.0.2.2 255.255.255.0
 ip nat outside
 ip virtual-reassembly in
 crypto map cmap
```

VPN流量需要从转换中排除，ISAKMP也是如此。如果ISAKMP流量不排除在转换之外，则它会被PAT发送到指向R3的外部接口：

```
R2#show ip nat translation
Pro Inside global      Inside local       Outside local      Outside global
udp 10.0.1.2:500       10.0.2.2:500       10.0.4.6:500       10.0.4.6:500

*Jun  8 09:09:37.779: IP: s=10.0.2.2 (local), d=10.0.4.6, len 196, local
feature, NAT(2), rtype 0, forus FALSE, sendself FALSE, mtu 0, fwdchk FALSE
*Jun  8 09:09:37.779: IP: s=10.0.2.2 (local), d=10.0.4.6 (Ethernet0/1),
len 196, sending
*Jun  8 09:09:37.779: IP: s=10.0.1.2 (local), d=10.0.4.6 (Ethernet0/1), len 196,
output feature, Post-routing NAT Outside(24), rtype 1, forus FALSE, sendself
FALSE, mtu 0, fwdchk FALSE
*Jun  8 09:09:37.779: IP: s=10.0.1.2 (local), d=10.0.4.6 (Ethernet0/1), len 196,
output feature, Common Flow Table(27), rtype 1, forus FALSE, sendself FALSE,
mtu 0, fwdchk FALSE
*Jun  8 09:09:37.779: IP: s=10.0.1.2 (local), d=10.0.4.6 (Ethernet0/1), len 196,
output feature, Stateful Inspection(28), rtype 1, forus FALSE, sendself FALSE,
mtu 0, fwdchk FALSE
*Jun  8 09:09:37.779: IP: s=10.0.1.2 (local), d=10.0.4.6 (Ethernet0/1), len 196,
output feature, IPSec output classification(34), rtype 1, forus FALSE, sendself
FALSE, mtu 0, fwdchk FALSE
*Jun  8 09:09:37.779: IP: s=10.0.1.2 (local), d=10.0.4.6 (Ethernet0/1), len 196,
output feature, NAT ALG proxy(59), rtype 1, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
*Jun  8 09:09:37.779: IP: s=10.0.1.2 (local), d=10.0.4.6 (Ethernet0/1), len 196,
output feature, IPSec: to crypto engine(75), rtype 1, forus FALSE, sendself FALSE,
mtu 0, fwdchk FALSE
*Jun  8 09:09:37.779: IP: s=10.0.1.2 (local), d=10.0.4.6 (Ethernet0/1), len 196,
output feature, Post-encryption output features(76), rtype 1, forus FALSE, sendself
FALSE, mtu 0, fwdchk FALSE
*Jun  8 09:09:37.779: IP: s=10.0.1.2 (local), d=10.0.4.6 (Ethernet0/1), len 196,
pre-encap feature, IPSec Output Encap(1), rtype 1, forus FALSE, sendself FALSE,
mtu 0, fwdchk FALSE
*Jun  8 09:09:37.779: IP: s=10.0.1.2 (local), d=10.0.4.6 (Ethernet0/1), len 196,
```

```
 pre-encap feature, Crypto Engine(3), rtype 1, forus FALSE, sendself FALSE, mtu 0,
fwdchk FALSE
*Jun  8 09:09:37.779: IP: s=10.0.1.2 (local), d=10.0.4.6 (Ethernet0/1), len 196,
sending full packet
```

测试

使用此配置时，存在完全冗余。VPN使用R4链路，其余流量使用R3路由。如果R4发生故障，VPN流量使用R3链路建立（PBR的路由映射不匹配，且使用默认路由）。

在ISP到R4关闭之前，R6会看到来自对等体10.0.2.2的流量：

```
R6#show crypto session
Crypto session current status

Interface: Ethernet0/0
Session status: UP-ACTIVE
Peer: 10.0.2.2 port 500
 IKEv1 SA: local 10.0.4.6/500 remote 10.0.2.2/500 Active
 IPSEC FLOW: permit ip 192.168.2.0/255.255.255.0 192.168.1.0/255.255.255.0
       Active SAs: 2, origin: crypto map
```

在R2将ISP用于R3的VPN流量后，R6看到来自对等体10.0.1.2的流量：

```
R6#show crypto session
Crypto session current status

Interface: Ethernet0/0
Session status: UP-ACTIVE
Peer: 10.0.1.2 port 500
 IKEv1 SA: local 10.0.4.6/500 remote 10.0.1.2/500 Active
 IPSEC FLOW: permit ip 192.168.2.0/255.255.255.0 192.168.1.0/255.255.255.0
       Active SAs: 2, origin: crypto map
```

相反，当到R3的链路断开时，一切正常。VPN流量仍使用到R4的链路。对192.168.1.0/24执行到PAT的网络地址转换(NAT)，以便适当分配外部地址。在R3关闭之前，会转换到10.0.1.2:

```
R2#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
icmp 10.0.1.2:1        192.168.1.1:1      10.0.4.6:1         10.0.4.6:1
```

在R3断开后，仍然有旧转换以及使用指向R4的链路的新转换（到10.0.2.2）：

```
R2#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
icmp 10.0.2.2:0        192.168.1.1:0      10.0.4.6:0         10.0.4.6:0
icmp 10.0.1.2:1        192.168.1.1:1      10.0.4.6:1         10.0.4.6:1
```

陷阱

如果一切正常，陷阱在哪里？他们在细节上。

本地生成的流量

这是需要从R2自身发起VPN流量的场景。此场景要求您在R2上配置本地PBR，以强制R2通过R4发送ISAKMP流量并使隧道启动。但是，出口接口是使用路由表确定的，默认路由指向R3，并且该数据包被发送到R3，而不是R4（用于VPN传输）。要验证该情况，请输入：

```
ip access-list extended isakmp
 permit udp any any eq isakmp
 permit udp any eq isakmp any
 permit icmp any any

route-map LOCAL-PBR permit 10
 match ip address isakmp
 set ip next-hop verify-availability 10.0.2.4 1 track 20

ip local policy route-map LOCAL-PBR
```

在本例中，通过R4强制本地生成的Internet控制消息协议(ICMP)。如果没有，则使用路由表处理从192.168.1.2到192.168.2.5本地生成的流量，并与R3建立隧道。

应用此配置后会发生什么情况？从192.168.1.2到192.168.2.5的ICMP数据包被放置到R4，并通过通向R4的链路启动隧道。隧道设置为：

```
R2#ping 192.168.2.6 source e0/0 repeat 10
Type escape sequence to abort.
Sending 10, 100-byte ICMP Echos to 192.168.2.6, timeout is 2 seconds:
Packet sent with a source address of 192.168.1.2
.!!!!!!!!!
Success rate is 90 percent (9/10), round-trip min/avg/max = 4/4/5 ms

R2#show crypto session detail
Crypto session current status

Code: C - IKE Configuration mode, D - Dead Peer Detection
K - Keepalives, N - NAT-traversal, T - cTCP encapsulation
X - IKE Extended Authentication, F - IKE Fragmentation

Interface: Ethernet0/1
Session status: DOWN
Peer: 10.0.4.6 port 500 fvrf: (none) ivrf: (none)
     Desc: (none)
     Phase1_id: (none)
 IPSEC FLOW: permit ip 192.168.1.0/255.255.255.0 192.168.2.0/255.255.255.0
       Active SAs: 0, origin: crypto map
       Inbound:  #pkts dec"ed 0 drop 0 life (KB/Sec) 0/0
       Outbound: #pkts enc"ed 0 drop 0 life (KB/Sec) 0/0

Interface: Ethernet0/2
Uptime: 00:00:06
Session status: UP-ACTIVE
Peer: 10.0.4.6 port 500 fvrf: (none) ivrf: (none)
     Phase1_id: 10.0.4.6
     Desc: (none)
 IKEv1 SA: local 10.0.2.2/500 remote 10.0.4.6/500 Active
         Capabilities:(none) connid:1009 lifetime:23:59:53
 IKEv1 SA: local 10.0.2.2/500 remote 10.0.4.6/500 Inactive
         Capabilities:(none) connid:1008 lifetime:0
 IPSEC FLOW: permit ip 192.168.1.0/255.255.255.0 192.168.2.0/255.255.255.0
       Active SAs: 2, origin: crypto map
```

```
       Inbound:  #pkts dec"ed 9 drop 0 life (KB/Sec) 4298956/3593
       Outbound: #pkts enc"ed 9 drop 0 life (KB/Sec) 4298956/3593
```

一切似乎都正常。流量随正确的链路E0/2发送到R4。即使R6也显示流量从10.2.2.2（即R4的链路IP地址）接收：

```
R6#show crypto session detail
Crypto session current status

Code: C - IKE Configuration mode, D - Dead Peer Detection
K - Keepalives, N - NAT-traversal, T - cTCP encapsulation
X - IKE Extended Authentication, F - IKE Fragmentation

Interface: Ethernet0/0
Uptime: 14:50:38
Session status: UP-ACTIVE
Peer: 10.0.2.2 port 500 fvrf: (none) ivrf: (none)
     Phase1_id: 10.0.2.2
     Desc: (none)
 IKEv1 SA: local 10.0.4.6/500 remote 10.0.2.2/500 Active
         Capabilities:(none) connid:1009 lifetime:23:57:13
 IPSEC FLOW: permit ip 192.168.2.0/255.255.255.0 192.168.1.0/255.255.255.0
       Active SAs: 2, origin: crypto map
       Inbound:  #pkts dec"ed 1034 drop 0 life (KB/Sec) 4360587/3433
       Outbound: #pkts enc"ed 1029 drop 0 life (KB/Sec) 4360587/3433
```

但实际上，这里有**ESP数据包的非对称**路由。ESP数据包以10.0.2.2作为源发送，但会被放在指向R3的链路上。加密响应通过R4返回。这可以通过检查R3和R4上的计数器来验证：

发送100个数据包之前，E0/0的R3计数器：

```
R3#show int e0/0 | i pack
 5 minute input rate 0 bits/sec, 0 packets/sec
 5 minute output rate 0 bits/sec, 0 packets/sec
     739 packets input, 145041 bytes, 0 no buffer
    0 input packets with dribble condition detected
     1918 packets output, 243709 bytes, 0 underruns
```

发送100个数据包后，计数器相同：

```
R3#show int e0/0 | i pack
 5 minute input rate 0 bits/sec, 0 packets/sec
 5 minute output rate 0 bits/sec, 0 packets/sec
    839 packets input, 163241 bytes, 0 no buffer
    0 input packets with dribble condition detected
    1920 packets output, 243859 bytes, 0 underruns
```

传入数据包的数量增加了100（在指向R2的链路上），但传出数据包的数量仅增加了2。因此R3只看到加密的ICMP回应。

在发送100个数据包之前，R4上会看到响应：

```
R4#show int e0/0 | i packet
 5 minute input rate 0 bits/sec, 0 packets/sec
 5 minute output rate 1000 bits/sec, 1 packets/sec
    793 packets input, 150793 bytes, 0 no buffer
```

```
    0 input packets with dribble condition detected
    1751 packets output, 209111 bytes, 0 underruns
```
发送100个数据包后：

```
R4#show int e0/0 | i packet
 5 minute input rate 0 bits/sec, 0 packets/sec
 5 minute output rate 0 bits/sec, 0 packets/sec
     793 packets input, 150793 bytes, 0 no buffer
    0 input packets with dribble condition detected
    1853 packets output, 227461 bytes, 0 underruns
```
发送到R2的数据包数增加了102（加密ICMP应答），而接收的数据包数增加了0。因此R4只看到加密的ICMP应答。当然，数据包捕获证实了这一点。

为什么会发生这种情况？答案在文章的第一部分。

以下是这些ICMP数据包的流：

1. 从192.168.1.2到192.168.2.6的ICMP因本地PBR而进入E0/2（指向R4的链路）。
2. ISAKMP会话使用10.0.2.2构建，并按照预期连接到E0/2链路。
3. 对于封装后的ICMP数据包，路由器需要确定出口接口，该接口使用指向R3的路由表来完成。这就是为什么源地址为10.0.2.2（指向R4的链路）的加密数据包通过R3发送的原因。
4. R6从10.0.2.2接收ESP数据包（与ISAKMP会话一致），解密数据包，并将ESP响应发送到10.0.2.2。
5. 由于路由，R5通过R4向10.0.2.2发回响应。
6. R2接收并解密，然后接受数据包。

因此，对本地生成的流量格外谨慎非常重要。

在许多网络中，使用单播反向路径转发(uRPF)，从10.0.2.2发出的流量可能会在R3的E0/0上丢弃。在这种情况下，ping不起作用。

此问题是否有解决方案？可以强制路由器将本地生成的流量视为中转流量。为此，本地PBR需要将流量定向到伪造的环回接口，该接口将其路由到该接口，就像中转流量一样。

不建议这样做。

> **注意：**在将NAT与PBR一起使用时，务必格外小心（请参阅上一节有关PAT访问列表中ISKMP流量的内容）。

不使用PBR的配置示例

还有另一种解决方案是折中的。使用与上例相同的拓扑，可以不使用PBR或本地PBR满足所有要求。对于此场景，仅使用路由。在R2上只添加一个路由条目，并删除所有PBR/本地PBR配置：

```
ip route 192.168.2.0 255.255.255.0 10.0.2.4 track 20
```
总的来说，R2具有以下路由配置：

```
ip route 0.0.0.0 0.0.0.0 10.0.1.3 track 10
```

```
ip route 0.0.0.0 0.0.0.0 10.0.2.4 100
ip route 192.168.2.0 255.255.255.0 10.0.2.4 track 20
```

当到R3的链路为UP时，第一个路由条目是指向R3的默认路由。第二个路由条目是到R3的链路断开时通往R4的备用默认路由。第三个条目根据R4链路状态确定向远程VPN网络发送的流量的方式（如果R4链路为UP，则通过R4发送到远程VPN网络的流量）。 使用此配置，无需策略路由。

缺点是什么？不再使用PBR进行精细控制。无法确定源地址。在这种情况下，发往192.168.2.0/24的所有流量在R4处于UP状态时都会发往R4，而不管其来源如何。在上一个示例中，由PBR和特定源控制：已选择192.168.1.0/24。

此解决方案对于哪种情况过于简单？用于多个LAN网络（R2后）。 当某些网络需要以安全方式（加密）和其他不安全方式（未加密）到达192.168.2.0/24时，来自不安全网络的流量仍然会被放在R2的E0/2接口上，并且不会到达加密映射。因此，它通过到R4的链路以未加密方式发送（主要要求是仅对加密流量使用R4）。

这种情况及其要求非常罕见，这就是为什么此解决方案经常使用的原因。

摘要

使用PBR和本地PBR功能以及VPN和NAT可能比较复杂，需要深入了解数据包流。

对于此处介绍的场景，建议使用两台独立的路由器 — 每台路由器都有一个ISP链路。在ISP发生故障时，流量可以轻松重新路由。无需PBR，整体设计更简单。

此外，还有一个折衷解决方案，它不需要使用PBR，而是使用静态浮动路由。

验证

当前没有可用于此配置的验证过程。

故障排除

目前没有针对此配置的故障排除信息。

相关信息

- 技术支持和文档 - Cisco Systems
- 思科IOS 15.3 M&T — 思科系统