# 策略路由和其影响在ESP和ISAKMP信息包有Cisco IOS的

## 目录



## 简介

本文描述基于策略的路由(PBR)和本地PBR效果，当应用到封装安全有效载荷(ESP)和互联网安全协会和密钥管理协议(ISAKMP)数据包，当您使用Cisco IOS时。


贡献用米哈拉Garcarz， Cisco TAC工程师。


## 先决条件

## 要求

Cisco 建议您具有以下主题的基础知识：

- Cisco IOS
- 在Cisco IOS的VPN配置

### 使用的组件

本文档中的信息根据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IPSec隧道建立之前，路由器启动ISAKMP交换。当那些数据包由路由器生成，数据包被对待，当本地产生的数据流和所有本地PBR决策应用。另外，路由器生成的所有数据包(增强的内部网关路由选择协议(EIGRP)、下一跳解析协议(NHRP)、边界网关协议(BGP)或者互联网控制消息协议(ICMP) ping)也凝视作为本地产生的数据流并且有应用的本地PBR决策。

由路由器转发，并且发送在通道间，呼叫中转流量的流量，在路由器的入口接口没有认为本地产生的数据流和任何希望的路由策略必须应用。

这有在流量横断通道是的暗示本地产生的数据流跟随PBR，但是中转流量不。此条款说明此差异结果在行为的。

对于需要是被封装的ESP的中转流量，没有需要有所有路由条目在ESP封装前后，因为PBR确定数据包的出口接口。对于需要是被封装的ESP的本地产生的数据流，有路由条目是必要的，因为本地PBR在封装前确定仅出口接口数据包的，并且路由确定POST被封装的数据包的出口接口。

本文包含使用有两条ISP链路的一个路由器的典型配置示例。一条链路用于为了访问互联网，并且第二是为VPN。在所有链路故障的情况下，流量重路由与一条不同的互联网服务提供商链路。也提交缺陷。

请注意PBR在思科快速转发(CEF)执行，而本地PBR进程交换。

# 在路由器生成的本地流量

此部分描述从路由器启动的流量行为(R)1。流量是ESP由R1封装。

### 拓扑

IPSec LAN到LAN隧道被构建在R1和R3之间。

关注数据流在R1 Lo0 (192.168.100.1)和R3 Lo0 (192.168.200.1)之间。

R3路由器有一个默认路由对R2。

R1没直接地有路由条目，仅连接的网络。

## 配置

R1有所有流量的本地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

当是UP时，在R1的所有本地产生的数据流发送对R2。

为了验证什么发生，当您启动通道，请发送从路由器的关注数据流：

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

> **警告：debug ip packet命令**威力生成很多调试并且有在CPU使用情况的巨大的影响。小心地请使用它。

此调试也准许access-list用于为了限制调试处理的流量总量。进程交换仅的**debug ip packet命令**显示流量。

这是在R1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
```

这是发生了什么：

关注数据流(192.168.100.1 > 192.168.200.1)由本地PBR和出口接口匹配确定(E0/0)。此操作触发crypto代码启动ISAKMP。该数据包由本地PBR也是策略路由的，确定出口接口(E0/0)。ISAKMP流量发送，并且通道协商

当您再， ping什么发生？

```
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)
```

这是发生了什么：

本地生成的关注数据流， 192.168.100.1 > 192.168.200.1，本地策略路由的，并且确定出口接口(E0/0)。数据包由在E0/0的IPsec输出功能消耗并且被封装。封装数据包(从192.168.0.1到10.0.0.2)被检查路由为了确定出口接口，但是那里是没什么在R1里路由表，是封装为什么发生故障

。

在此方案中，通道是UP，但是流量没有发送，因为，在ESP封装以后， Cisco IOS检查路由表为了确定出口接口。

# 中转流量通过路由器

此部分描述通过路由器来，是该路由器封装的ESP的中转流量的行为。

## 拓扑

L2L通道被构建在R1和R3之间。

关注数据流在R4 (192.168.100.1)和R3 lo0 (192.168.200.1)之间。

R3路由器有一个默认路由对R2。

R4路由器有一个默认路由对R1。

R1没有路由。

## 配置

修改上一个拓扑为了显示流，当路由器收到加密的时(而不是本地产生的数据流的中转流量数据包)。

现在，从R4接收的关注数据流是策略路由的在R1 (由在E0/1的PBR)，并且也有所有流量的本地策略路由：

```
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)
```

## 调试

为了验证什么发生，当您启动在R1的通道(在您收到从R4的关注数据流)后，回车：

```
R1#debug ip packetR4#ping 192.168.200.1
```

这是在R1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
```

这是发生了什么：

关注数据流点击在E0/0和触发crypto代码的PBR发送ISAKMP信息包。ISAKMP信息包是本地策略路由的和出口接口取决于本地PBR。通道被构建。

这更是一ping对从R4的192.168.200.1 ：

```
R4#ping 192.168.200.1
```
这是在R1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
```
这是发生了什么：

关注数据流点击在E0/0的PBR，并且那PBR确定出口接口(E0/0)。在E0/0，数据包由IPSec消耗并且被封装。在封装数据包根据同一个PBR规则后核对，并且确定出口接口，数据包正确地发送并且接收。

# 行为差异的摘要

对于本地产生的数据流，本地PBR取决于非被封装的流量的(ISAKMP)出口接口。对于本地产生的数据流，路由表取决于POST被封装的流量的(ESP)出口接口(本地PBR没有被检查)。对于中转流量，接口PBR取决于POST被封装的流量的(ESP)出口接口(两次，在封装前后)。

# 配置示例

这是提交问题您也许面对与PBR和本地PBR与VPN的实用的配置示例。R2 (CE)有两条ISP链路。R6路由器也有CE和一条ISP链路。从R2to R3的第一条链路使用作为默认路由R2。对R4的第二条链

路仅使用对R6的VPN流量。在所有ISP链路故障的情况下，流量被重路由对另一条链路。

## 拓扑

## 配置

192.168.1.0/24和192.168.2.0/24之间的流量保护。开放最短路径优先(OSPF)用于互联网网云为了通告10.0.0.0/8地址，对待ISP分配的公共地址对客户。在真实世界， BGP使用而不是OSPF。

在R2和R6的配置根据加密映射。在R2，如果是UP， PBR在E0/0用于为了处理VPN流量到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
```

您看到本地PBR不是需要的。对10.0.2.4的接口PBR路由关注数据流。既使当路由是到远端对等体点通过R3，那触发crypto代码启动从正确接口(对R4的链路的ISAKMP)。

在R6，使用VPN的两对等体：

```
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使用一IP服务级别协议(SLA)为了ping R3和R4。默认路由是R3。在R3失败的情况下，它选择R4 ：

```
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
```

并且R2允许互联网访问所有里面用户。为了达到在对R3的ISP发生故障的案件的冗余， route-map是必要的。它端口地址转换(轻拍)对一不同的建立接口E0/2，当R3发生故障和R4的出口接口(PAT对E0/1接口，当R3上，并且默认路由指向R3和PAT的里面流量使用作为默认路由)。

```
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流量需要从转换被排除象ISAKMP。如果ISAKMP流量从转换没有被排除，它是PATed对去往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，有全面冗余。VPN使用R4链路，并且流量的其余路由与R3。在R4失败的情况下，VPN流量设立与R3链路(PBR的route-map不配比，并且使用默认路由)。

在对R4的ISP发生故障前， R6看到从对等体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使用ISP对R3 VPN流量后， R6看到从对等体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
```

对于相反的方案，当对R3的链路断开时，一切良好仍然工作。VPN流量仍然使用链路对R4。网络地址转换(NAT)为192.168.1.0/24执行对PAT为了合适外部地址。在R3断开前，有转换对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，仍有旧有转换与该新的转换一起(对10.0.2.2)用途往R4的链路：

```
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
```

## 缺陷

如果一切良好工作，在哪里缺陷？他们在详细信息。

### 生成的流量本地

这是需要启动从R2的VPN流量的方案。此方案要求您配置在R2的本地PBR为了通过R4强制R2to发送ISAKMP流量和造成通道上升。但是确定出口接口与使用路由表，当默认指向R3，并且该数据包发送对R3，而不是R4，使用VPN的传输。为了验证那，回车：

```
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
```

在本例中生成本地的，互联网控制消息协议(ICMP)通过R4是牵强的。没有该，从192.168.1.2生成的本地流量到192.168.2.5处理与使用路由表，并且通道设立与R3。

在您运用此配置后，什么发生？从192.168.1.2的ICMP数据包到192.168.2.5放置往R4，并且通道发起与对R4的链路。通道设置：

```
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
```

一切似乎正确地运作。流量用往R4的正确链路E0/2传送。R6显示流量从10.2.2.2接收，是R4's链路IP地址：

```
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
```

但是实际上，有**ESP数据包的不对称路由**在这里。ESP数据包在往R3的链路上发送与10.0.2.2作为来源，但是把放。一已加密答复通过R4返回。这可以通过检查在R3和R4的计数器验证：

R3 E0/0计数器在发送100数据包前的：

```
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
```

并且同样计数器，在发送100数据包以后：

```
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
```

100增加的流入数据包数量(在往R2的链路)，但是2.仅增加的输出数据包。R3只所以看到已加密ICMP回音。

答复在R4被看到，在发送100数据包前：

```
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
```

在发送100数据包以后：

```
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
```

发送的数据包编号往102增加的R2的(已加密ICMP回复)，而收到的信息包由0增加。R4只所以看到已加密ICMP回复。当然，数据包捕获确认此。

为什么会发生这种情况？答案是在条款的第一部分。

这是那些ICMP数据包流：

1. 从192.168.1.2的ICMP到192.168.2.6在E0/2 (往R4的链路放置)由于本地PBR。
2. ISAKMP会话用10.0.2.2和放置的E0/2链路建立正如所料。
3. 对于在封装以后的ICMP数据包，路由器需要确定出口接口，完成与使用路由表指向R3。这就是为什么有来源的10.0.2.2 (往R4的链路加密的信息包)通过R3发送。
4. R6收到从10.0.2.2的一ESP数据包，是一致与ISAKMP会话，解码数据包，并且发送对10.0.2.2的ESP答复。
5. 由于路由， R5发送回到对10.0.2.2的一答复通过R4。
6. R2接收它和解密，并且数据包接受。

这就是为什么是额外谨慎的对本地产生的数据流是重要的。

在许多网络中，使用单播逆向路径转发(URPF)，并且从10.0.2.2发出的流量在R3 E0/0可能丢弃。在那种情况下， ping不工作。

有没有此问题的任何解决方案？强制路由器处理本地产生的数据流作为中转流量是可能的。为此，本地PBR需要直接数据流对路由类似中转流量的一假回环接口。

这没有建议。

> **注意**：额外小心是重要的，当您与PBR一起时使用NAT (参考关于ISKMP流量的前面部分在访问列表PAT)。

# 没有PBR的配置示例

也有是妥协的另一解决方案。没有使用PBR或本地PBR，使用拓扑和前一个示例一样，满足所有要求是可能的。对于此scenarrio，使用只路由。仅一个路由条目在R2还被添加，并且所有PBR/local PBR配置删除：

```
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
```

总共， R2有此路由配置：

```
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
```

当对R3的链路是UP时，第一个路由条目是往R3的一个默认路由。当对R3的链路发生故障时，第二个路由条目是往R4的一备用默认路由。第三个条目决定对远程VPN网络的哪个方式流量发送，依靠R4链路状态(如果R4链路是UP，对远程VPN网络的流量通过R4发送)。使用此配置，没有对策略路由的需要。

什么是缺点？再没有粒状控制使用PBR。确定源地址是不可能的。在这种情况下，所有流量到

192.168.2.0/24发送往R4，当是UP时，不管来源。在前一个示例中，那是由PBR和特定来源控制的：192.168.1.0/24选择。

对于哪个方案此解决方案是否是太简单？多个LAN网络(在R2后)。当其中一些网络需要到达192.168.2.0/24用一个安全方式(加密)和其他不安全方式(未加密)，从不安全网络的流量在R2 E0/2接口上仍然把放，并且不点击加密映射。因此它发送的未加密通过链路对R4 (和主要需求是仅使用R4加密流量)。

这种方案和其需求是少见的，是为什么相当经常使用此解决方案。

## 摘要

使用PBR和本地PBR功能与VPN和NAT一起也许复杂并且要求数据包流的详细了解。

对于方案例如被提交的那些此处，建议使用两个独立路由器-有一条ISP链路的每个路由器。在ISP失败的情况下，可以容易地重路由流量。没有对PBR的需要，并且总体设计更加简单。

也有不要求使用PBR的妥协，但是用途静态浮点路由。

## 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
- Cisco IOS 15.3 M&T- Cisco系统