Roteamento de política e seu impacto em pacotes ESP e ISAKMP com Cisco IOS

Contents

Introduction

Este documento descreve o efeito do Roteamento Baseado em Políticas (PBR - Policy Based Routing) e do PBR local quando aplicado aos pacotes de Encapsulating Security Payload (ESP - Encapsulating Security Payload) e Internet Security Association and Key Management Protocol (ISAKMP - Associação de Segurança da Internet e Protocolo de Gerenciamento de Chave) quando você usa o Cisco IOS®.

Contribuído por Michal Garcarz, engenheiro do TAC da Cisco.

Prerequisites

Requirements

A Cisco recomenda que você tenha conhecimento básico sobre estes tópicos:

- Cisco IOS
- Configuração de VPN no Cisco IOS

Componentes Utilizados

As informações neste documento são baseadas no Cisco IOS versão 15.x.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Informações de Apoio

Antes do estabelecimento do túnel IPsec, o roteador inicia uma troca ISAKMP. À medida que esses pacotes são gerados pelo roteador, os pacotes são tratados como tráfego gerado localmente e quaisquer decisões PBR locais são aplicadas. Além disso, todos os pacotes gerados pelo roteador (EIGRP (Enhanced Interior Gateway Routing Protocol), NHRP (Next Hop Resolution Protocol), BGP (Border Gateway Protocol) ou pings ICMP (Internet Control Message Protocol) também são considerados como tráfego gerado localmente e têm a decisão PBR local aplicada.

O tráfego que é encaminhado pelo roteador e enviado através do túnel, que é chamado de tráfego de trânsito, não é considerado tráfego gerado localmente, e qualquer política de roteamento desejada deve ser aplicada na interface de entrada do roteador.

As implicações disso no tráfego que atravessa o túnel é que o tráfego gerado localmente segue o PBR, mas o tráfego de trânsito não. Este artigo explica as consequências dessa diferença de comportamento.

Para o tráfego de trânsito que precisa ser encapsulado por ESP, não há necessidade de ter entradas de roteamento porque o PBR determina a interface de saída do pacote antes e depois do encapsulamento ESP. Para o tráfego gerado localmente que precisa ser encapsulado por ESP, é necessário ter entradas de roteamento, pois o PBR local determina a interface de saída somente para o pacote antes do encapsulamento e o roteamento determina a interface de saída para o pacote pós-encapsulado.

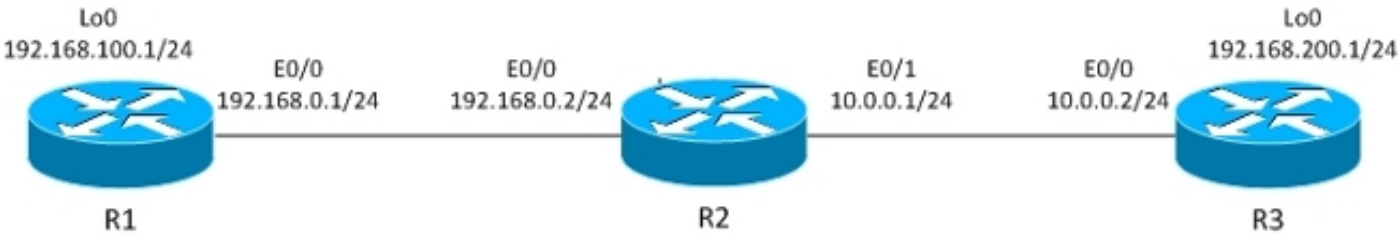
Este documento contém um exemplo de configuração típica em que um roteador com dois links ISP é usado. Um link é usado para acessar a Internet e o segundo é para VPN. Em caso de falha de link, o tráfego é roteado novamente com um link diferente do provedor de serviços de Internet (ISP). As quedas também são apresentadas.

Observe que o PBR é executado no Cisco Express Forwarding (CEF), enquanto o PBR local é comutado por processo.

Tráfego gerado localmente no roteador

Esta seção descreve o comportamento do tráfego iniciado a partir do roteador (R)1. Esse tráfego é encapsulado pelo ESP R1.

Topologia



O túnel de LAN para LAN IPsec é construído entre R1 e R3.

O tráfego interessante está entre R1 Lo0 (192.168.100.1) e R3 Lo0 (192.168.200.1).

O roteador R3 tem uma rota padrão para R2.

R1 não tem entradas de roteamento, apenas redes diretamente conectadas.

Configuração

R1 tem PBR local para todo o tráfego:

```
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
```

Debugs

Todo o tráfego gerado localmente em R1 é enviado para R2 quando está UP.

Para verificar o que ocorre quando você ativa o túnel, envie o tráfego interessante do próprio roteador:

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

> **Caution**: O comando **debug ip packet** pode gerar uma grande quantidade de depurações e tem um enorme impacto no uso da CPU. Use-o com cuidado.

Essa depuração também permite que a lista de acesso seja usada para limitar a quantidade de tráfego processado por depurações. O comando **debug ip packet** exibe apenas o tráfego que é comutado por processo.

Aqui estão as depurações em R1:

```
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
```

Aqui está o que acontece:

O tráfego interessante (192.168.100.1 > 192.168.200.1) é correspondido pelo PBR local, e a interface de saída é determinada (E0/0). Esta ação aciona o código de criptografia para iniciar ISAKMP. Esse pacote também é roteado por políticas pelo PBR local, que determina a interface de saída (E0/0). O tráfego ISAKMP é enviado e o túnel é negociado

O que acontece quando você faz ping novamente?

```
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)
```

Aqui está o que acontece:

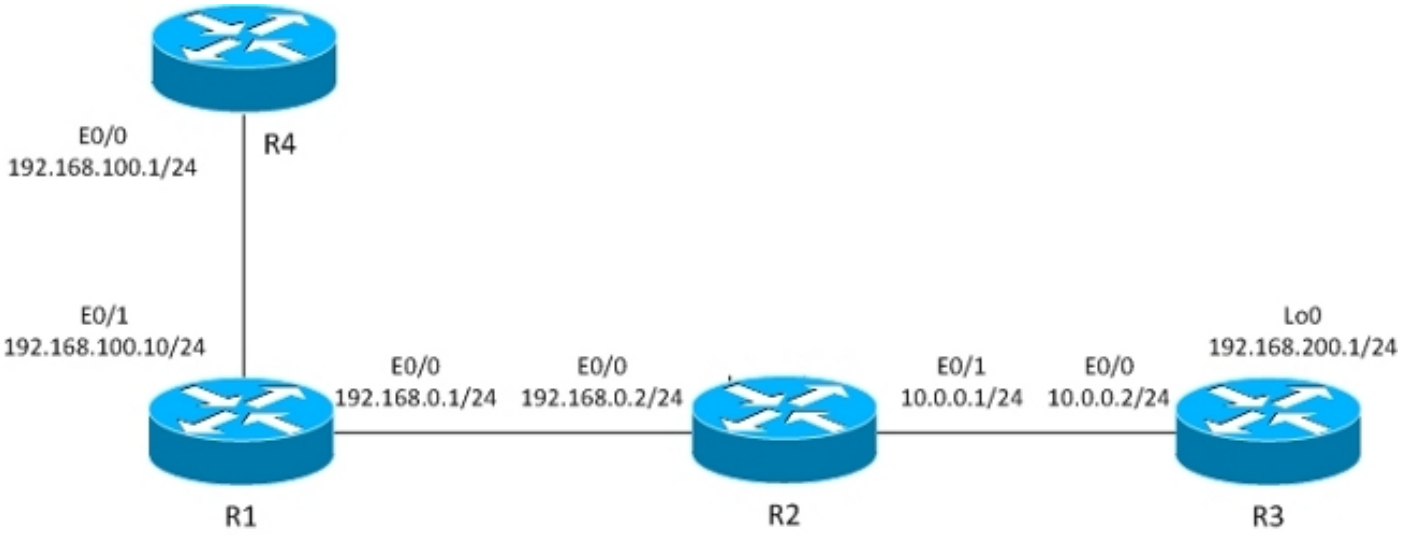
O tráfego interessante gerado localmente, 192.168.100.1 > 192.168.200.1, é roteado por políticas locais e a interface de saída é determinada (E0/0). O pacote é consumido pelo recurso de saída IPsec em E0/0 e encapsulado. O pacote encapsulado (de 192.168.0.1 a 10.0.0.2) é verificado quanto ao roteamento para determinar a interface de saída, mas não há nada nas tabelas de roteamento de R1, razão pela qual o encapsulamento falha.

Neste cenário, o túnel é UP, mas o tráfego não é enviado porque, após o encapsulamento ESP, o Cisco IOS verifica as tabelas de roteamento para determinar a interface de saída.

Tráfego de trânsito através do roteador

Esta seção descreve o comportamento do tráfego de trânsito que vem através do roteador, que é o ESP encapsulado por esse roteador.

Topologia



O túnel L2L é construído entre R1 e R3.

O tráfego interessante está entre R4 (192.168.100.1) e R3 lo0 (192.168.200.1).

O roteador R3 tem uma rota padrão para R2.

O roteador R4 tem uma rota padrão para R1.

R1 não tem roteamento.

Configuração

A topologia anterior é modificada para mostrar o fluxo quando o roteador recebe pacotes para criptografia (tráfego de trânsito em vez de tráfego gerado localmente).

No momento, o tráfego interessante recebido de R4 é roteado por política em R1 (por PBR em E0/1) e também há roteamento de política local para todo o tráfego:

```
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
```

Debugs

Para verificar o que acontece quando você ativa o túnel em R1 (depois de receber o tráfego interessante de R4), insira:

```
R1#debug ip packet
```

```
R4#ping 192.168.200.1
```

Aqui estão as depurações em R1:

```
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
```

Aqui está o que acontece:

O tráfego interessante atinge o PBR em E0/0 e dispara o código de criptografia para enviar o pacote ISAKMP. Esse pacote ISAKMP é roteado localmente por políticas e a interface de saída é determinada pelo PBR local. Um túnel é construído.

Aqui está mais um ping para 192.168.200.1 de R4:

```
R4#ping 192.168.200.1
```

Aqui estão as depurações em R1:

```
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
```

Aqui está o que acontece:

O tráfego interessante atinge o PBR em E0/0 e esse PBR determina a interface de saída (E0/0). Em E0/0, o pacote é consumido pelo IPSec e encapsulado. Depois que o pacote encapsulado é verificado em relação à mesma regra PBR e a interface de saída é determinada, o pacote é enviado e recebido corretamente.

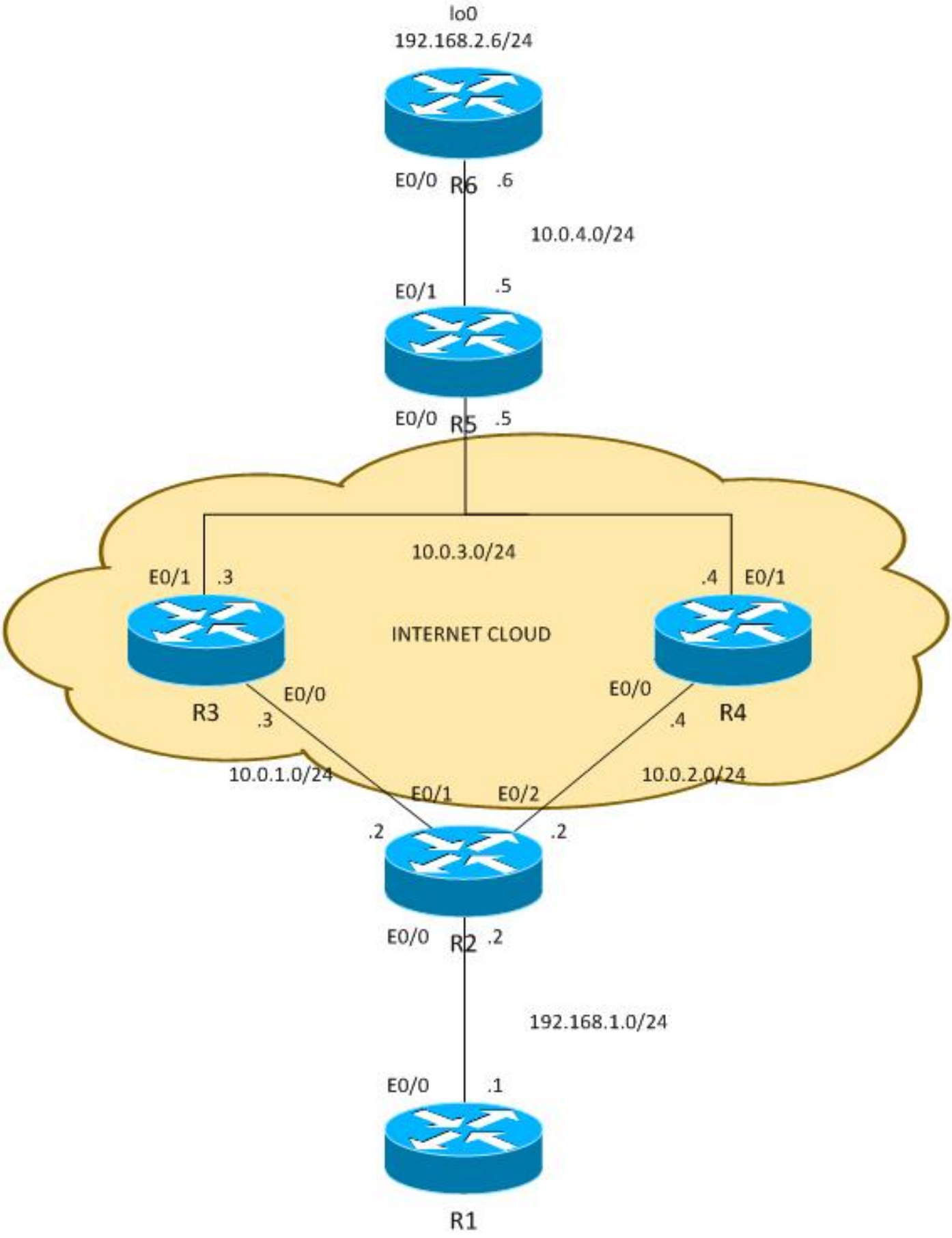
Resumo das diferenças de comportamento

Para o tráfego gerado localmente, a interface de saída para o tráfego não encapsulado (ISAKMP) é determinada pelo PBR local. Para o tráfego gerado localmente, a interface de saída para o tráfego pós-encapsulado (ESP) é determinada pelas tabelas de roteamento (o PBR local não está marcado). Para tráfego de trânsito, a interface de saída para tráfego pós-encapsulado (ESP) é determinada pela interface PBR (duas vezes, antes e depois do encapsulamento).

Exemplo de configuração

Este é um exemplo de configuração prática que apresenta os problemas que você pode enfrentar com PBR e PBR local com VPN. O R2 (CE) tem dois links de ISP. O roteador R6 também tem um link CE e um ISP. O primeiro link de R2 para R3 é usado como uma rota padrão para R2. O segundo link para R4 é usado somente para tráfego VPN para R6. Em caso de falha de link do ISP, o tráfego é roteado novamente para o outro link.

Topologia



Configuração

O tráfego entre 192.168.1.0/24 e 192.168.2.0/24 está protegido. O OSPF (Open Shortest Path First) é usado na nuvem da Internet para anunciar os endereços 10.0.0.0/8, que são tratados

como endereços públicos atribuídos pelo ISP ao cliente. No mundo real, o BGP é usado em vez do OSPF.

A configuração em R2 e R6 é baseada no mapa de criptografia. Em R2, o PBR é usado em E0/0 para direcionar o tráfego VPN para R4 se for UP:

```
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
```

Aqui você vê que o PBR local não é necessário. A interface PBR roteia tráfego interessante para 10.0.2.4. Isso aciona o código de criptografia para iniciar o ISAKMP na interface correta (link para R4), mesmo quando o roteamento é para pontos de peer remotos através de R3.

Em R6, dois peers para a VPN são usados:

```
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 usa um Contrato de Nível de Serviço (SLA - Service Level Agreement) IP para fazer ping em R3 e R4. A rota padrão é R3. Em caso de falha de R3, escolhe 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
```

Além disso, o R2 permite acesso à Internet para todos os usuários internos. Para obter redundância no caso de ISP para R3 estar inoperante, é necessário um mapa de rota. As Conversões de Endereço de Porta (PATs - Port Address Translation) dentro do tráfego para uma interface de saída diferente (PAT para a interface E0/1 quando R3 está UP e a rota padrão aponta para R3, e PAT para a interface E0/2 quando R3 está inativo e R4 é usado como uma rota padrão).

```
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
```

O tráfego VPN precisa ser excluído da tradução como o ISAKMP. Se o tráfego ISAKMP não for excluído da conversão, ele será PATed para a interface externa que vai para R3:

```
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
```

Testando

Com essa configuração, há uma redundância total. A VPN usa o link R4 e o restante do tráfego é roteado com R3. Em caso de falha de R4, o tráfego VPN é estabelecido com o link de R3 (o mapa de rota para PBR não corresponde e o roteamento padrão é usado).

Antes que o ISP para R4 esteja inoperante, o R6 vê o tráfego do peer 10.0.2.2:

```
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
```

Depois que R2 usa ISP para R3 para tráfego VPN, R6 vê o tráfego do peer 10.0.1.2:

```
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
```

No cenário oposto, quando o link para R3 fica inativo, tudo ainda funciona bem. O tráfego VPN ainda usa o link para R4. A Conversão de Endereço de Rede (NAT - Network Address Translation) é executada para 192.168.1.0/24 para PAT a fim de se adequar ao endereço externo. Antes de R3 ser desativado, há uma tradução para 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
```

Depois que R3 é desativado, ainda há a tradução antiga junto com a nova tradução (para 10.0.2.2) que usa o link em direção ao R4:

```
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
```

Armadilhas

Se tudo der certo, onde estão as armadilhas? Eles estão nos detalhes.

Tráfego gerado localmente

Aqui está um cenário que precisa iniciar o tráfego VPN do próprio R2. Esse cenário exige que você configure o PBR local em R2 para forçar R2 a enviar tráfego ISAKMP via R4 e para fazer com que o túnel fique UP. Mas a interface de saída é determinada com o uso de tabelas de roteamento, com o padrão apontando para R3, e esse pacote é enviado para R3, em vez de R4, que é usado para trânsito para VPN. Para verificar isso, insira:

```
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
```

Neste exemplo, o Internet Control Message Protocol (ICMP) que é gerado localmente é forçado através de R4. Sem isso, o tráfego gerado localmente de 192.168.1.2 a 192.168.2.5 é processado com o uso de tabelas de roteamento e um túnel é estabelecido com R3.

O que acontece depois que você aplica essa configuração? O pacote ICMP de 192.168.1.2 a 192.168.2.5 é colocado em direção ao R4 e um túnel é iniciado com o link para R4. O túnel está configurado:

```
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
```

Tudo parece funcionar corretamente. O tráfego é enviado com o link E0/2 correto para R4. Mesmo R6 mostra que o tráfego é recebido de 10.2.2.2, que é o endereço IP do link de R4:

```
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
```

Mas, na verdade, há **roteamento assimétrico para pacotes ESP** aqui. Os pacotes ESP são enviados com 10.0.2.2 como origem, mas são colocados no link em direção a R3. Uma resposta criptografada é retornada por R4. Isso pode ser verificado verificando contadores em R3 e R4:

Contadores R3 de E0/0 antes de enviar 100 pacotes:

```
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
```

E os mesmos contadores, depois de enviar 100 pacotes:

```
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
```

O número de pacotes de entrada aumentou em 100 (no link em direção ao R2), mas os pacotes de saída aumentaram em apenas 2. Portanto, R3 vê apenas o eco ICMP criptografado.

A resposta é vista em R4, antes de enviar 100 pacotes:

```
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
```

Depois de enviar 100 pacotes:

```
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
```

O número de pacotes enviados para R2 aumentou em 102 (resposta ICMP criptografada), enquanto os pacotes recebidos aumentaram em 0. Portanto, R4 vê apenas a resposta ICMP criptografada. Claro, uma captura de pacotes confirma isso.

Por que isso acontece? A resposta está na primeira parte do artigo.

Aqui está o fluxo desses pacotes ICMP:

1. O ICMP de 192.168.1.2 a 192.168.2.6 é colocado em E0/2 (link em direção a R4) devido ao PBR local.
2. A sessão ISAKMP é criada com 10.0.2.2 e colocada no link E0/2, conforme esperado.
3. Para pacotes ICMP após o encapsulamento, o roteador precisa determinar a interface de saída, o que é feito com o uso de tabelas de roteamento que apontam para R3. É por isso que o pacote criptografado com origem 10.0.2.2 (link em direção a R4) é enviado através de R3.
4. O R6 recebe um pacote ESP de 10.0.2.2, que é consistente com a sessão ISAKMP, descriptografa o pacote e envia a resposta ESP para 10.0.2.2.
5. Devido ao roteamento, R5 envia uma resposta de volta para 10.0.2.2 através de R4.
6. O R2 o recebe e descriptografa, e o pacote é aceito.

Épor isso que é importante ter cuidado extra com o tráfego gerado localmente.

Em muitas redes, o Unicast Reverse Path Forwarding (uRPF) é usado e o tráfego originado de 10.0.2.2 pode ser descartado em E0/0 de R3. Nesse caso, o ping não funciona.

Há alguma solução para esse problema? É possível forçar o roteador a tratar o tráfego gerado localmente como tráfego de trânsito. Para isso, o PBR local precisa direcionar o tráfego para uma

interface de loopback falsa da qual é roteado como o tráfego de trânsito.

Isso não é aconselhável.

> **Note**: É importante ter cuidado extra ao usar o NAT junto com o PBR (consulte a seção anterior sobre o tráfego ISKMP na lista de acesso PAT).

Exemplo de configuração sem PBR

Há também outra solução que é um compromisso. Com a mesma topologia do exemplo anterior, é possível satisfazer todos os requisitos sem o uso de PBR ou PBR local. Para esse cenário, somente o roteamento é usado. Apenas mais uma entrada de roteamento é adicionada em R2 e todas as configurações PBR/PBR local são removidas:

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

No total, R2 tem esta configuração de roteamento:

```
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
```

A primeira entrada de roteamento é um roteamento padrão para R3, quando o link para R3 é UP. A segunda entrada de roteamento é uma rota padrão de backup em direção a R4, quando o link para R3 está inoperante. A terceira entrada decide qual caminho o tráfego para a rede VPN remota é enviado, dependendo do estado do link R4 (se o link R4 estiver UP, o tráfego para a rede VPN remota será enviado via R4). Com essa configuração, não há necessidade de roteamento de política.

Qual é a desvantagem? Não há mais controle granular usando PBR. Não é possível determinar o endereço de origem. Nesse caso, todo o tráfego para 192.168.2.0/24 é enviado para R4 quando está UP, independentemente da origem. No exemplo anterior, que era controlado pelo PBR e a origem específica: 192.168.1.0/24 está selecionado.

Para qual cenário essa solução é muito simples? Para várias redes LAN (atrás de R2). Quando algumas dessas redes precisam acessar 192.168.2.0/24 de forma segura (criptografada) e de outras maneiras inseguras (não criptografadas), o tráfego de redes inseguras ainda é colocado na interface E0/2 de R2 e não atinge o mapa de criptografia. Portanto, ele é enviado sem criptografia por meio de um link para R4 (e o requisito principal era usar R4 somente para tráfego criptografado).

Esse tipo de cenário e seus requisitos são raros, por isso essa solução é usada com bastante frequência.

Summary

O uso de recursos PBR e PBR local junto com VPNs e NAT pode ser complexo e exige uma compreensão profunda do fluxo de pacotes.

Para cenários como os apresentados aqui, recomenda-se usar dois roteadores separados - cada roteador com um link ISP. Em caso de falha do ISP, o tráfego pode ser redirecionado facilmente. Não há necessidade de PBR, e o projeto geral é muito mais simples.

Há também uma solução de comprometimento que não exige o uso do PBR, mas usa o roteamento flutuante estático.

Verificar

No momento, não há procedimento de verificação disponível para esta configuração.

Troubleshoot

Atualmente, não existem informações disponíveis específicas sobre Troubleshooting para esta configuração.

Informações Relacionadas

- Suporte Técnico e Documentação - Cisco Systems
- Cisco IOS 15.3 M&T- Cisco Systems