Roteamento de política e seu impacto no ESP e pacotes ISAKMP com Cisco IOS

Índice

Introdução
Pré-requisitos
Requisitos
Componentes Utilizados
Informações de Apoio
Tráfego gerado localmente no roteador
Topologia
Configuração
Debugs
Tráfego de trânsito através do roteador
Topologia
Configuração
Debugs
Sumário para diferenças do comportamento
Exemplo de configuração
Topologia
Configuração
Testando
Armadilhas
Tráfego gerado localmente
Exemplo de configuração sem PBR
Resumo
Verificar
Troubleshooting
Informações Relacionadas

Introdução

Este documento descreve o efeito do Policy Based Routing (PBR) e do PBR local quando aplicado aos pacotes do Encapsulating Security Payload (ESP) e do Internet Security Association and Key Management Protocol (ISAKMP) quando você usa o ® do Cisco IOS.

Contribuído por Michal Garcarz, engenheiro de TAC da Cisco.

Pré-requisitos

Requisitos

Cisco recomenda que você tem o conhecimento básico destes assuntos:

- Cisco IOS
- Configuração de VPN no Cisco IOS

Componentes Utilizados

A informação neste documento é baseada na versão do Cisco IOS 15.x.

As informações neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. Todos os dispositivos utilizados neste documento foram iniciados com uma configuração (padrão) inicial. Se a sua rede estiver ativa, certifique-se de que entende o impacto potencial de qualquer comando.

Informações de Apoio

Antes do estabelecimento do túnel de IPsec, o roteador inicia uma troca ISAKMP. Enquanto aqueles pacotes são gerados pelo roteador, os pacotes estão tratados enquanto o tráfego gerado localmente e todas as decisões locais PBR são aplicados. Além, todos os pacotes gerados pelo roteador (sibilos do Enhanced Interior Gateway Routing Protocol (EIGRP), do Next Hop Resolution Protocol (NHRP), do Border Gateway Protocol (BGP), ou do Internet Control Message Protocol (ICMP)) igualmente são considerados como o tráfego gerado localmente e têm a decisão local PBR aplicada.

Trafique que é enviado pelo roteador e enviado através do túnel, que é chamado tráfego de trânsito, não é considerado tráfego gerado localmente, e toda a política de roteamento desejada deve ser aplicada na interface de ingresso do roteador.

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

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

Este documento contém um exemplo da configuração típica onde um roteador com dois links ISP seja usado. Um link é usado a fim alcançar o Internet e o segundo é para o VPN. Em caso de toda a falha do link, o tráfego é redistribuído com um link diferente do provedor de serviço do Internet (ISP). As armadilhas são apresentadas igualmente.

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

Trafique gerado localmente no roteador

Esta seção descreve o comportamento do tráfego iniciado do roteador (R)1. Que o tráfego é ESP encapsulou pelo r1.

Topologia

O túnel de LAN para LAN de IPSec é construído entre o r1 e o R3.

O tráfego interessante está entre o r1 Lo0 (192.168.100.1) e R3 Lo0 (192.168.200.1).

O roteador R3 tem uma rota padrão ao R2.

O r1 não tem nenhuma entrada de roteamento, somente diretamente redes conectadas.

Configuração

O r1 tem o 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 no r1 está enviado ao R2 quando está ACIMA.

A fim verificar o que ocorre quando você traz acima o túnel, envie o tráfego interessante do roteador próprio:

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

> Cuidado: **O comando debug ip packet** pôde gerar uma grande quantidade de debuga e tem o impacto enorme no USO de CPU. Use-o com cuidado.

Isto debuga igualmente permite que a lista de acesso seja usada a fim limitar a quantidade de

tráfego processada por debuga. O comando debug ip packet indica somente o tráfego que é comutado por processo.

Seja aqui debuga no 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
```

Éaqui o que acontece:

O tráfego interessante (192.168.100.1 > 192.168.200.1) é combinado pelo PBR local, e pela interface de saída é determinado (E0/0). Esta ação provoca o código cripto para iniciar o ISAKMP. Esse pacote é igualmente política-roteado pelo PBR local, que determina a interface de saída (E0/0). O tráfego ISAKMP é enviado, e o túnel é negociado

Que acontece quando você sibila outra vez?

```
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)
```

Éaqui o que acontece:

O tráfego interessante localmente gerado, 192.168.100.1 > 192.168.200.1, é localmente política-roteado, e a interface de saída é determinada (E0/0). O pacote é consumido pelos recursos de emissor do IPsec no E0/0 e encapsulado. O pacote encapsulado (de 192.168.0.1 a 10.0.0.2) não é verificado distribuindo a fim determinar a interface de saída, mas lá é nada nas tabelas de roteamento do r1, que é porque o encapsulamento falha.

Nesta encenação, o túnel está ACIMA, mas o tráfego não é enviado porque, após o encapsulamento ESP, o Cisco IOS verifica as tabelas de roteamento a fim determinar a interface de saída.

Tráfego de trânsito através do roteador

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

Topologia

O túnel L2L é construído entre o r1 e o 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 ao R2.

O roteador R4 tem uma rota padrão ao r1.

O r1 não tem nenhum roteamento.

Configuração

A topologia antiga está alterada a fim mostrar o fluxo quando o roteador recebe pacotes para a criptografia (tráfego de trânsito em vez do tráfego gerado localmente).

Agora, o tráfego interessante recebido do R4 é política-roteado no r1 (pelo PBR em E0/1), e há igualmente um roteamento da política local para todo o tráfego:

```
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)
```

Debugs

A fim verificar o que acontece quando você traz acima o túnel no r1 (depois que você recebe o tráfego interessante do R4), entra:

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

Seja aqui debuga no 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
```

Éaqui o que acontece:

O tráfego interessante bate o PBR no E0/0 e no código cripto dos disparadores para enviar o pacote ISAKMP. Que o pacote ISAKMP é localmente política-roteado, e a interface de saída é determinado pelo PBR local. Um túnel é construído.

Éaqui um mais sibilo a 192.168.200.1 do R4:

```
R4#ping 192.168.200.1
```

Seja aqui debuga no 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
```

Éaqui o que acontece:

O tráfego interessante bate o PBR no E0/0, e esse PBR determina a interface de saída (E0/0). No E0/0, o pacote é consumido pelo IPsec e encapsulado. Depois que o pacote encapsulado está verificado contra a mesma regra PBR e a interface de saída é determinada, o pacote está enviado e recebido corretamente.

Sumário para diferenças do comportamento

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

Exemplo de configuração

Este é um exemplo de configuração prático que apresente as edições que você pôde enfrentar com PBR e o PBR local com VPN. O R2 (CE) tem dois links ISP. O roteador R6 igualmente tem o CE e o um link ISP. O primeiro link do R2 ao R3 é usado como uma rota padrão para o R2. O segundo link ao R4 é usado somente para o tráfego VPN ao R6. Em caso de toda a falha do link ISP, o tráfego é redistribuído ao outro link.

Topologia

Configuração

O tráfego entre 192.168.1.0/24 e 192.168.2.0/24 é protegido. O Open Shortest Path First (OSPF) é usado na nuvem do Internet a fim anunciar os 10.0.0.0/8 endereços, que são tratados como os endereços públicos atribuídos pelo ISP ao cliente. No mundo real, o BGP é usado em vez do OSPF.

A configuração no R2 e no R6 é baseada no mapa cript. No R2, o PBR está usado no E0/0 a fim dirigir o tráfego VPN ao R4 se está ACIMA:

```
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 está precisado. A relação PBR distribui o tráfego interessante a 10.0.2.4. Isso provoca o código cripto para iniciar o ISAKMP da relação correta (link ao R4), mesmo quando o roteamento é aos pontos do peer remoto com o R3.

No R6, dois pares para o 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
```

O R2 usa um acordo do nível de serviço IP (SLA) a fim sibilar o R3 e o R4. A rota padrão é R3. Em caso da falha R3, escolhe o 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
```

Igualmente o R2 permite a acesso ao Internet para tudo usuários do interior. A fim conseguir a Redundância no caso onde o ISP ao R3 está para baixo, um mapa de rotas é necessário. Ele tráfego do interior das traduções de endereço de porta (pancadinhas) a uma interface de saída diferente (a PANCADINHA à relação E0/1 quando o R3 for ASCENDENTE e a rota padrão apontar ao R3, e à PANCADINHA para conectar o E0/2 quando o R3 estiver para baixo e o R4 é usada 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 de ser excluído da tradução como faz ISAKMP. Se o tráfego ISAKMP não é excluído da tradução, é PATed à interface externa que vai para o 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 esta configuração, há uma redundância direta. O VPN usa o link R4, e o resto do tráfego é distribuído com R3. Em caso da falha R4, o tráfego VPN é estabelecido com o link R3 (o mapa de rotas para o PBR não combina e o roteamento padrão é usado).

Antes que o ISP ao R4 esteja para baixo, o R6 vê o tráfego do par 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 o R2 usa o ISP ao R3 para o tráfego VPN, o R6 vê o tráfego do par 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
```

Para a encenação oposta, quando o link ao R3 vai para baixo, tudo ainda trabalha muito bem. O tráfego VPN ainda usa o link ao R4. O Network Address Translation (NAT) é executado para 192.168.1.0/24 à PANCADINHA a fim apropriar o endereço exterior. Antes que o R3 vá para baixo, há uma tradução a 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 o R3 vai para baixo, há ainda a tradução velha junto com a tradução nova (a 10.0.2.2) essa usos o link para o 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 trabalha muito bem, onde estão as armadilhas? Estão nos detalhes.

Tráfego gerado localmente

Está aqui uma encenação que precise de iniciar o tráfego do R2 próprio VPN. Esta encenação exige-o configurar o PBR local no R2 a fim forçar o R2 para enviar o tráfego ISAKMP através do R4 e para fazer com que o túnel vá ACIMA. Mas a interface de saída é determinada com o uso das tabelas de roteamento, com o padrão que aponta ao R3, e esse pacote é enviado ao R3, em vez do R4, que é usado para o trânsito para o VPN. A fim verificar isso, entre:

```
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 do R4. Sem esse, o tráfego gerado localmente de 192.168.1.2 a 192.168.2.5 é processado com o uso das tabelas de roteamento e um túnel é estabelecido com R3.

Que acontece depois que você aplica esta configuração? O pacote ICMP de 192.168.1.2 a 192.168.2.5 é posto para o R4, e um túnel é iniciado com o link ao R4. O túnel estabelece-se:

```
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 trabalhar corretamente. O tráfego é enviado com o link correto E0/2 para o R4. Mesmo o R6 mostra que o tráfego está recebido de 10.2.2.2, que é endereço IP de Um ou Mais Servidores Cisco ICM NT do link do 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 realmente, há um **roteamento assimétrico para pacotes ESP** aqui. Os pacotes ESP são enviados com 10.0.2.2 como uma fonte, mas postos sobre o link para o R3. Uma resposta cifrada é retornada com o R4. Isto pode ser verificado verificando contadores no R3 e no R4:

Contadores R3 do 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, após ter enviado 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 recebidos aumentados por 100 (no link para o R2), mas os pacotes de saída aumentados somente por 2. Assim o R3 vê somente o eco ICMP cifrado.

A resposta é considerada no 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
```

Após ter enviado 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 o R2 aumentado por 102 (resposta de ICMP cifrada), quando os pacotes recebidos aumentaram por 0. Assim o R4 vê somente a resposta de ICMP cifrada. Naturalmente, uma captura de pacote de informação confirma esta.

Por que isso acontece? A resposta é no primeiro parte do artigo.

Está aqui o fluxo daqueles pacotes ICMP:

1. O ICMP de 192.168.1.2 a 192.168.2.6 é posto sobre E0/2 (link para o R4) devido ao PBR local.
2. A sessão ISAKMP é construída com 10.0.2.2 e link E0/2 como esperado sobre posto.
3. Para pacotes ICMP após o encapsulamento, o roteador precisa de determinar a interface de saída, que é feita com o uso das tabelas de roteamento que apontam ao R3. Eis porque o pacote criptografado com fonte 10.0.2.2 (link para o R4) é enviado através do R3.
4. O R6 recebe um pacote ESP de 10.0.2.2, que seja consistente com a sessão ISAKMP, decifre o pacote, e envie a resposta ESP a 10.0.2.2.
5. Devido ao roteamento, o R5 envia para trás uma resposta a 10.0.2.2 com o R4.
6. O R2 recebe o e decrypts, e o pacote é aceitado.

Eis porque é importante ser extremamente cauteloso com tráfego gerado localmente.

Em muitas redes, o Unicast Reverse Path Forwarding (uRPF) é usado e o tráfego originado de 10.0.2.2 poderia ser deixado cair no E0/0 do R3. Nesse caso, o sibilo não trabalha.

Há alguma solução para este problema? É possível forçar o roteador a tratar o tráfego gerado localmente como o tráfego de trânsito. Para o esse, o PBR local precisa o tráfego direto a uma interface de loopback falsa de que é distribuído como o tráfego de trânsito.

Isto não é recomendado.

> Nota: É importante ser extremamente cuidadoso quando você usa o NAT junto com o PBR (refira a seção anterior sobre o tráfego ISKMP na lista de acesso da PANCADINHA).

Exemplo de configuração sem PBR

Há igualmente uma outra solução que seja um acordo. Com a mesma topologia que o exemplo anterior, é possível satisfazer todas as exigências sem o uso do PBR ou do PBR local. Para este scenarrio, somente distribuir é usada. Somente uma mais entrada de roteamento é adicionada no R2, e todas as configurações PBR/local PBR são removidas:

```
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
```

No total, o R2 tem esta configuração de roteamento:

```
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
```

A primeira entrada de roteamento é um roteamento padrão para o R3, quando o link ao R3 está ACIMA. A segunda entrada de roteamento é uma rota padrão alternativa para o R4, quando o link ao R3 está para baixo. A terceira entrada decide que tráfego da maneira à rede VPN remota é enviado, segundo o estado do link R4 (se o link R4 está ACIMA, o tráfego à rede VPN remota é enviado através do R4). Com esta configuração, não há nenhuma necessidade para o roteamento de política.

Que é o inconveniente? Não há nenhum controle granulado usando o PBR any more. Não é possível determinar o endereço de origem. Neste caso, todo o tráfego a 192.168.2.0/24 é enviado para o R4 quando está ACIMA, apesar da fonte. No exemplo anterior, isso foi controlado pelo PBR e pela fonte específica: 192.168.1.0/24 são selecionados.

Para que encenação é esta solução demasiado simples? Para redes da LAN múltipla (atrás do R2). _quando algum aquele rede precisar para alcançar 192.168.2.0/24 um seguro maneira (cifrar) e outro incerto maneira (unencrypted), tráfego rede insegura estar ainda pôr E0/2 relação R2 e fazer não bater mapa cript. É enviado assim unencrypted através de um link ao R4 (e ao requisito principal era usar o R4 somente para o tráfego criptografado).

Este tipo de cenário e suas exigências são raros, que é porque esta solução é usada razoavelmente frequentemente.

Resumo

Usar o PBR e características locais PBR junto com VPN e NAT pôde ser complexa e exige um entendimento aprofundado do fluxo de pacote de informação.

Para encenações tais como aqueles apresentados aqui, recomenda-se para usar dois roteadores separados - cada roteador com um link ISP. Em caso de uma falha ISP, o tráfego pode ser redistribuído facilmente. Não há nenhuma necessidade para o PBR, e o projeto geral é muito mais simples.

Há igualmente uma solução de acordo que não exija o uso do PBR, mas uma distribuição de flutuação estática dos usos pelo contrário.

Verificar

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

Troubleshooting

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

Informações Relacionadas

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