Routing de políticas y su impacto en los paquetes ESP e ISAKMP con Cisco IOS

Contenido

Introducción

Este documento describe el efecto del routing basado en políticas (PBR) y de la PBR local cuando se aplican a los paquetes de carga de seguridad de encapsulación (ESP) y de la asociación de seguridad de Internet y el protocolo de administración de claves (ISAKMP) cuando se utiliza Cisco IOS®.

Colaborado por Michal Garcarz, Ingeniero del TAC de Cisco.

Prerequisites

Requirements

Cisco recomienda que tenga conocimientos básicos sobre estos temas:

- IOS de Cisco
- Configuración de VPN en Cisco IOS

Componentes Utilizados

La información de este documento se basa en la versión 15.x del IOS de Cisco.

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.

Antecedentes

Antes del establecimiento del túnel IPsec, el router inicia un intercambio ISAKMP. A medida que esos paquetes son generados por el router, los paquetes se tratan como tráfico generado localmente y se aplican cualquier decisión de PBR local. Además, cualquier paquete generado por el router (ping de protocolo de routing de gateway interior mejorado (EIGRP), protocolo de resolución de siguiente salto (NHRP), protocolo de gateway fronterizo (BGP) o pings de protocolo de mensajes de control de Internet (ICMP)) también se considera tráfico generado localmente y se aplica la decisión de PBR local.

El tráfico que es reenviado por el router y enviado a través del túnel, que se denomina tráfico de tránsito, no se considera tráfico generado localmente y cualquier política de ruteo deseada se debe aplicar en la interfaz de ingreso del router.

Las implicaciones que esto tiene en el tráfico que atraviesa el túnel es que el tráfico generado localmente sigue PBR, pero el tráfico de tránsito no. Este artículo explica las consecuencias de esta diferencia de comportamiento.

Para el tráfico de tránsito que necesita encapsularse ESP, no hay necesidad de tener entradas de ruteo porque PBR determina la interfaz de egreso para el paquete antes y después de la encapsulación ESP. Para el tráfico generado localmente que necesita ser encapsulado ESP, es necesario tener entradas de ruteo, porque el PBR local determina la interfaz de egreso solamente para el paquete antes de la encapsulación y el ruteo determina la interfaz de egreso para el paquete post encapsulado.

Este documento contiene un ejemplo de configuración típico donde se utiliza un router con dos links ISP. Se utiliza un link para acceder a Internet y el segundo para VPN. En caso de que se produzca una falla en el enlace, el tráfico se vuelve a enrutar con un enlace de proveedor de servicios de Internet (ISP) diferente. También se presentan los obstáculos.

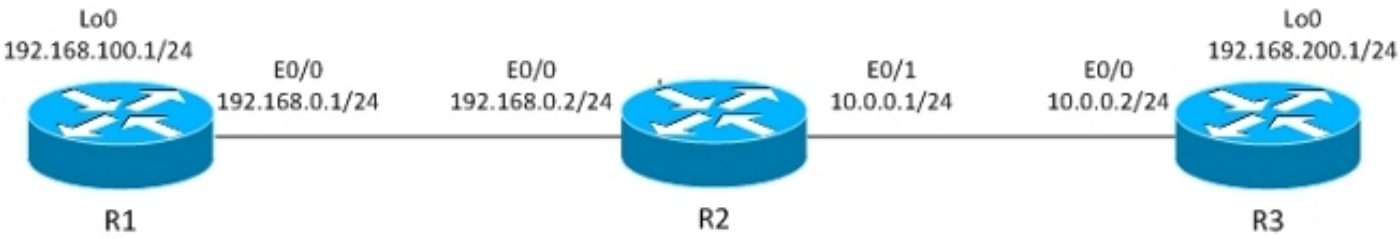
Tenga en cuenta que el PBR se realiza en Cisco Express Forwarding (CEF), mientras que el PBR

local se conmuta por proceso.

Tráfico generado localmente en el router

Esta sección describe el comportamiento del tráfico iniciado desde el router (R)1. Ese tráfico es ESP encapsulado por R1.

Topología



El túnel IPsec de LAN a LAN se construye entre R1 y R3.

El tráfico interesante está entre R1 Lo0 (192.168.100.1) y R3 Lo0 (192.168.200.1).

El router R3 tiene una ruta predeterminada hacia R2.

R1 no tiene entradas de ruteo, sólo redes conectadas directamente.

Configuración

R1 tiene PBR local para todo el tráfico:

```
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
```

Depuraciones

Todo el tráfico generado localmente en R1 se envía a R2 cuando está ACTIVO.

Para verificar qué ocurre cuando se activa el túnel, envíe el tráfico interesante del propio router:

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

> **Precaución:** El comando **debug ip packet** puede generar una gran cantidad de debugs y tiene un enorme impacto en el uso de la CPU. Utilícelo con precaución.

Esta depuración también permite utilizar la lista de acceso para limitar la cantidad de tráfico procesado por depuraciones. El comando **debug ip packet** sólo muestra el tráfico que se conmuta por proceso.

Estas son las depuraciones en 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
```

Esto es lo que sucede:

El tráfico interesante (192.168.100.1 > 192.168.200.1) coincide con el PBR local y se determina la interfaz de salida (E0/0). Esta acción activa el código criptográfico para iniciar ISAKMP. Ese paquete también es ruteado por la PBR local, que determina la interfaz de salida (E0/0). Se envía el tráfico ISAKMP y se negocia el túnel

¿Qué ocurre cuando hace ping de nuevo?

```
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)
```

Esto es lo que sucede:

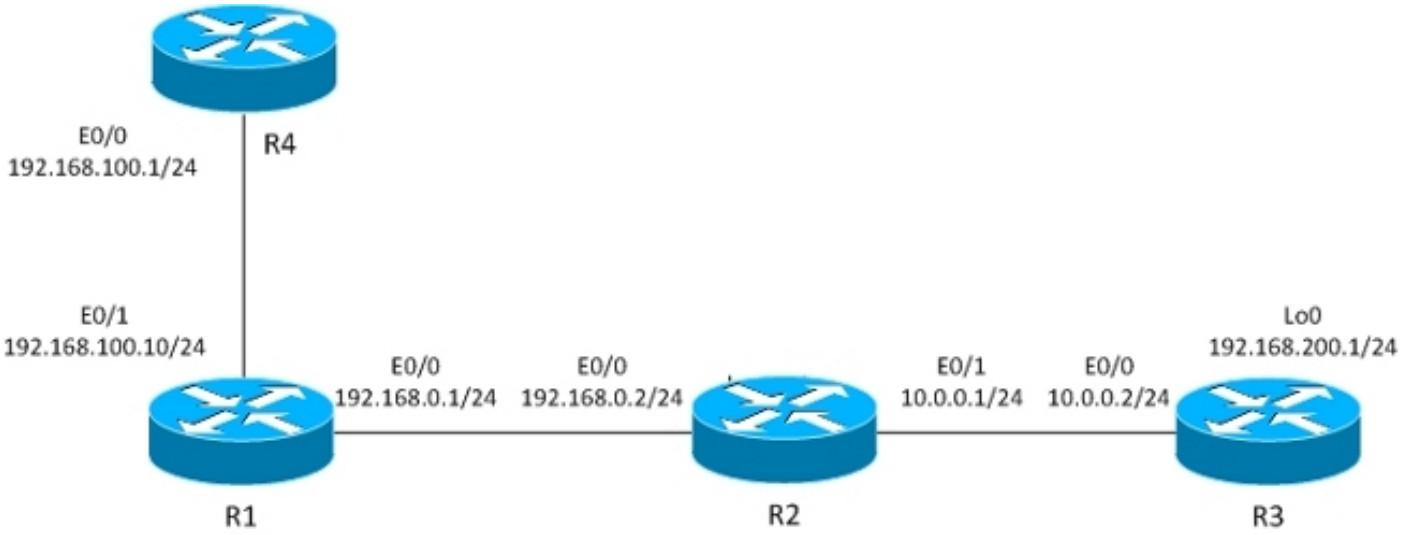
El tráfico interesante generado localmente, 192.168.100.1 > 192.168.200.1, se rutea localmente con políticas y se determina la interfaz de salida (E0/0). El paquete es consumido por la función de salida de IPSec en E0/0 y encapsulado. El paquete encapsulado (de 192.168.0.1 a 10.0.0.2) se verifica en busca de ruteo para determinar la interfaz de salida, pero no hay nada en las tablas de ruteo de R1, razón por la cual falla la encapsulación.

En este escenario, el túnel es ACTIVO, pero el tráfico no se envía porque, después de la encapsulación ESP, Cisco IOS verifica las tablas de ruteo para determinar la interfaz de egreso.

Tráfico de tránsito a través del router

Esta sección describe el comportamiento del tráfico de tránsito que llega a través del router, que es encapsulado por ese router.

Topología



El túnel L2L se construye entre R1 y R3.

El tráfico interesante está entre R4 (192.168.100.1) y R3 lo0 (192.168.200.1).

El router R3 tiene una ruta predeterminada hacia R2.

El router R4 tiene una ruta predeterminada a R1.

R1 no tiene ruteo.

Configuración

La topología anterior se modifica para mostrar el flujo cuando el router recibe paquetes para el cifrado (tráfico de tránsito en lugar de tráfico generado localmente).

En este momento, el tráfico interesante recibido de R4 se enruta mediante políticas en R1 (por PBR en E0/1), y también hay ruteo de políticas locales para todo el tráfico:

```
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
```

Depuraciones

Para verificar qué sucede cuando se activa el túnel en R1 (después de recibir el tráfico interesante de R4), ingrese:

```
R1#debug ip packet
```

```
R4#ping 192.168.200.1
```

Estas son las depuraciones en 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
```

Esto es lo que sucede:

El tráfico interesante llega a PBR en E0/0 y activa el código crypto para enviar el paquete ISAKMP. Ese paquete ISAKMP se rutea localmente mediante políticas y la interfaz de salida se determina mediante PBR local. Se construye un túnel.

Aquí hay un ping más a 192.168.200.1 desde R4:

```
R4#ping 192.168.200.1
```

Estas son las depuraciones en 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
```

Esto es lo que sucede:

El tráfico interesante llega a PBR en E0/0 y ese PBR determina la interfaz de salida (E0/0). En E0/0, IPSec consume el paquete y lo encapsula. Después de que el paquete encapsulado se verifique en relación con la misma regla PBR y se determine la interfaz de egreso, el paquete se envía y recibe correctamente.

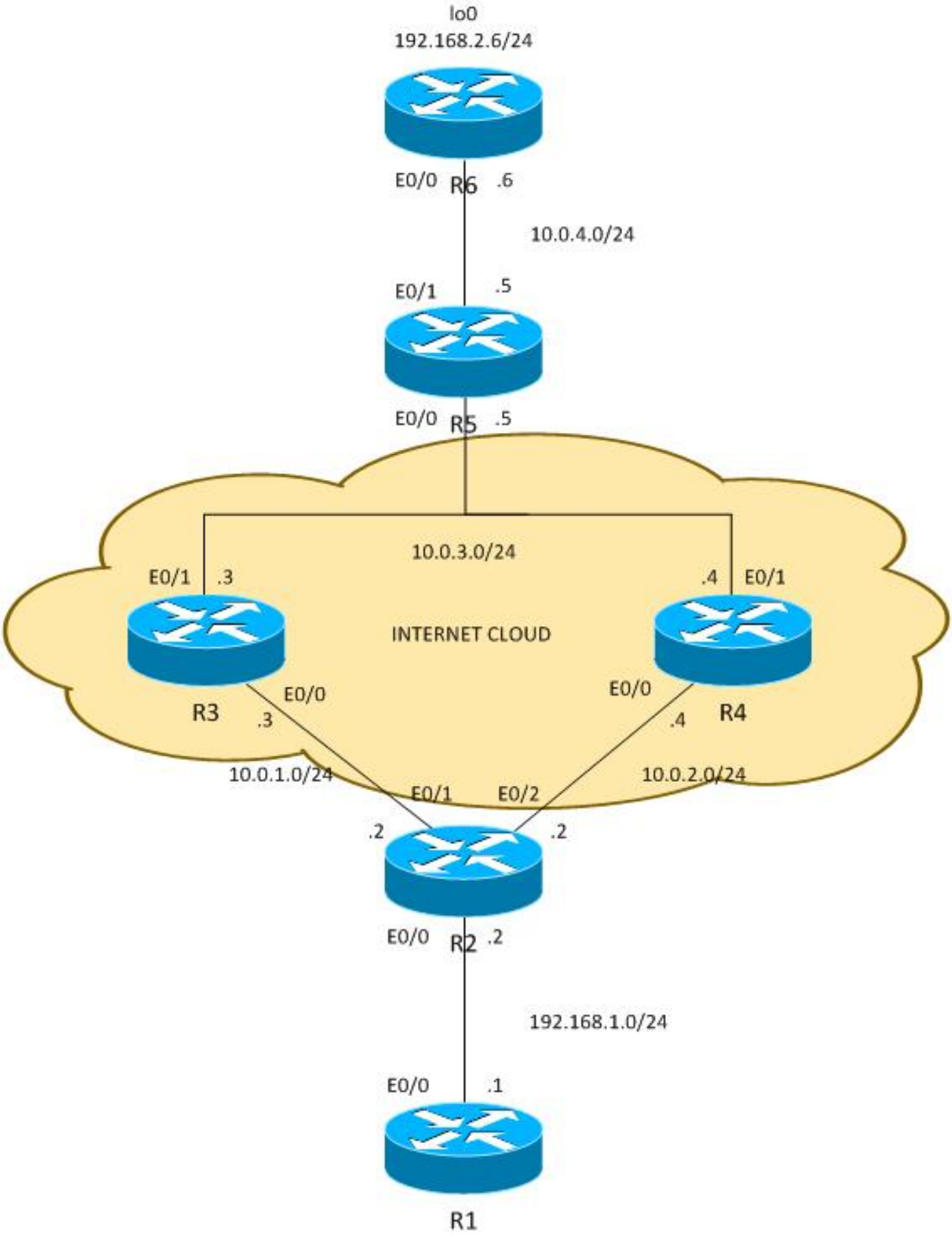
Resumen de diferencias de comportamiento

Para el tráfico generado localmente, la interfaz de salida para el tráfico no encapsulado (ISAKMP) está determinada por la PBR local. Para el tráfico generado localmente, la interfaz de salida para el tráfico post-encapsulado (ESP) se determina mediante las tablas de routing (no se verifica el PBR local). Para el tráfico de tránsito, la interfaz de salida para el tráfico post-encapsulado (ESP) es determinada por la interfaz PBR (dos veces, antes y después de la encapsulación).

Ejemplo de configuración

Este es un ejemplo práctico de configuración que presenta los problemas que podría enfrentar con PBR y PBR local con VPN. El R2 (CE) tiene dos enlaces ISP. El router R6 también tiene CE y un link ISP. El primer link de R2 a R3 se utiliza como ruta predeterminada para R2. El segundo link a R4 se utiliza solamente para el tráfico VPN a R6. En caso de que se produzca una falla en el link ISP, el tráfico se redirige al otro link.

Topología



Configuración

El tráfico entre 192.168.1.0/24 y 192.168.2.0/24 está protegido. Open Shortest Path First (OSPF) se utiliza en la nube de Internet para anunciar las direcciones 10.0.0.0/8, que se tratan como

direcciones públicas asignadas por ISP al cliente. En el mundo real, se utiliza BGP en lugar de OSPF.

La configuración en R2 y R6 se basa en el mapa crypto. En R2, PBR se utiliza en E0/0 para dirigir el tráfico VPN a R4 si es ACTIVO:

```
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
```

Aquí puede ver que no se necesita PBR local. La interfaz PBR enruta el tráfico interesante a 10.0.2.4. Esto activa el código crypto para iniciar ISAKMP desde la interfaz correcta (link a R4), incluso cuando el ruteo es a puntos de peer remotos a través de R3.

En R6, se utilizan dos pares para la VPN:

```
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 utiliza un acuerdo de nivel de servicio (SLA) IP para hacer ping a R3 y R4. La ruta predeterminada es R3. En caso de falla de R3, elige 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
```

También R2 permite el acceso a Internet para todos los usuarios internos. Para lograr la redundancia en el caso en que el ISP a R3 esté inactivo, es necesario un route-map. Traducciones de dirección de puerto (PAT) dentro del tráfico a una interfaz de salida diferente (PAT a interfaz E0/1 cuando R3 está ACTIVO y la ruta predeterminada apunta a R3, y PAT a la interfaz E0/2 cuando R3 está inactivo y R4 se utiliza como ruta predeterminada).

```
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
```

El tráfico VPN debe excluirse de la traducción al igual que ISAKMP. Si el tráfico ISAKMP no se excluye de la traducción, se envía PATed a la interfaz exterior que va hacia 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
```

Prueba

Con esta configuración, hay una redundancia completa. La VPN utiliza el link R4 y el resto del tráfico se enruta con R3. En caso de falla de R4, el tráfico VPN se establece con el link R3 (el route-map para PBR no coincide y se utiliza el ruteo predeterminado).

Antes de que el ISP a R4 esté inactivo, R6 ve el tráfico del 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
```

Después de que R2 utilice ISP a R3 para el tráfico VPN, R6 ve el tráfico del 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
```

En el escenario opuesto, cuando el link a R3 deja de funcionar, todo sigue funcionando bien. El tráfico VPN todavía utiliza el link a R4. La traducción de direcciones de red (NAT) se realiza para 192.168.1.0/24 a PAT para apropiarse de la dirección externa. Antes de que R3 caiga, hay una traducción 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
```

Después de que R3 se interrumpa, todavía hay la traducción antigua junto con la nueva traducción (a 10.0.2.2) que utiliza el link hacia 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
```

Peligros

Si todo funciona bien, ¿dónde están los escollos? Están en los detalles.

Tráfico generado localmente

Este es un escenario que necesita iniciar el tráfico VPN desde el propio R2. Esta situación requiere que configure el PBR local en R2 para obligar a R2 a enviar tráfico ISAKMP a través de R4 y hacer que el túnel se ACTIVE. Pero la interfaz de salida se determina con el uso de tablas de ruteo, con el valor predeterminado apuntando a R3, y ese paquete se envía a R3, en lugar de R4, que se utiliza para el tránsito para VPN. Para verificar que, ingrese:

```
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
```

En este ejemplo, el protocolo de mensajes de control de Internet (ICMP) que se genera localmente se fuerza a través de R4. Sin eso, el tráfico generado localmente entre 192.168.1.2 y 192.168.2.5 se procesa con el uso de tablas de ruteo y se establece un túnel con R3.

¿Qué sucede después de aplicar esta configuración? El paquete ICMP de 192.168.1.2 a 192.168.2.5 se dirige hacia R4, y se inicia un túnel con el link a R4. El 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
```

Todo parece funcionar correctamente. El tráfico se envía con el link E0/2 correcto hacia R4. Incluso R6 muestra que el tráfico se recibe de 10.2.2.2, que es la dirección IP del 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
```

Pero en realidad, hay **ruteo asimétrico para los paquetes ESP** aquí. Los paquetes ESP se envían con 10.0.2.2 como origen, pero se ponen en el link hacia R3. Se devuelve una respuesta cifrada a través de R4. Esto puede verificarse comprobando los contadores en R3 y R4:

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

```
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
```

Y los mismos contadores, después de enviar 100 paquetes:

```
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
```

El número de paquetes entrantes aumentó en 100 (en el link hacia R2), pero los paquetes salientes aumentaron solamente en 2. Así que R3 sólo ve el eco ICMP cifrado.

La respuesta se ve en R4, antes de enviar 100 paquetes:

```
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
```

Después de enviar 100 paquetes:

```
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
```

El número de paquetes enviados hacia R2 aumentó en 102 (respuesta ICMP cifrada), mientras que los paquetes recibidos aumentaron en 0. Así que R4 sólo ve la respuesta ICMP cifrada. Por supuesto, una captura de paquetes lo confirma.

¿Por qué ocurre esto? La respuesta está en la primera parte del artículo.

Este es el flujo de esos paquetes ICMP:

1. El ICMP de 192.168.1.2 a 192.168.2.6 se coloca en E0/2 (link hacia R4) debido a la PBR local.
2. La sesión ISAKMP se construye con 10.0.2.2 y se pone en el link E0/2 como se espera.
3. Para los paquetes ICMP después de la encapsulación, el router necesita determinar la interfaz de egreso, que se realiza con el uso de tablas de ruteo que apuntan a R3. Esta es la razón por la que el paquete cifrado con el origen 10.0.2.2 (link hacia R4) se envía a través de R3.
4. R6 recibe un paquete ESP desde 10.0.2.2, que es consistente con la sesión ISAKMP, descifra el paquete y envía la respuesta ESP a 10.0.2.2.
5. Debido al ruteo, R5 envía una respuesta a 10.0.2.2 a R4.
6. R2 lo recibe y descifra, y se acepta el paquete.

Es por esto que es importante ser más cauteloso con el tráfico generado localmente.

En muchas redes, se utiliza Unicast Reverse Path Forwarding (uRPF) y el tráfico procedente de 10.0.2.2 se puede descartar en E0/0 de R3. En ese caso, el ping no funciona.

¿Hay alguna solución para este problema? Es posible obligar al router a tratar el tráfico generado localmente como tráfico de tránsito. Para ello, el PBR local necesita dirigir el tráfico a una interfaz

de loopback falsa desde la que se rutea como tráfico de tránsito.

Esto no se aconseja.

> **Nota:** Es importante tener más cuidado cuando utiliza NAT junto con PBR (consulte la sección anterior sobre el tráfico ISKMP en la lista de acceso PAT).

Ejemplo de configuración sin PBR

También hay otra solución que supone un compromiso. Con la misma topología que el ejemplo anterior, es posible satisfacer todos los requerimientos sin el uso de PBR o PBR local. Para este escenario, sólo se utiliza el ruteo. Sólo se agrega una entrada de ruteo más en R2 y se eliminan todas las configuraciones PBR/PBR locales:

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

En total, R2 tiene esta configuración de ruteo:

```
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
```

La primera entrada de ruteo es un ruteo predeterminado hacia R3, cuando el link a R3 es UP. La segunda entrada de ruteo es una ruta predeterminada de respaldo hacia R4, cuando el link a R3 está inactivo. La tercera entrada decide de qué manera se envía el tráfico a la red VPN remota, según el estado del link R4 (si el link R4 está ACTIVO, el tráfico a la red VPN remota se envía a través de R4). Con esta configuración, no hay necesidad de ruteo de políticas.

¿Cuál es el inconveniente? Ya no hay control granular con PBR. No es posible determinar la dirección de origen. En este caso, todo el tráfico a 192.168.2.0/24 se envía hacia R4 cuando está ACTIVO, independientemente del origen. En el ejemplo anterior, que estaba controlado por PBR y el origen específico: 192.168.1.0/24 está seleccionado.

¿Para qué escenario es esta solución demasiado simple? Para varias redes LAN (detrás de R2). Cuando algunas de esas redes necesitan alcanzar 192.168.2.0/24 de forma segura (cifrada) y de otras formas inseguras (no cifradas), el tráfico de redes inseguras se sigue colocando en la interfaz E0/2 de R2 y no llega al mapa criptográfico. Por lo tanto, se envía sin cifrar a través de un enlace a R4 (y el requisito principal era utilizar R4 sólo para el tráfico cifrado).

Este tipo de escenario y sus requisitos son raros, por lo que esta solución se utiliza con bastante frecuencia.

Summary

El uso de las funciones PBR y PBR locales junto con las VPN y NAT puede ser complejo y requiere una comprensión profunda del flujo de paquetes.

Para escenarios como los presentados aquí, se recomienda utilizar dos routers independientes: cada router con un link ISP. En caso de una falla del ISP, el tráfico se puede volver a rutear

fácilmente. No se necesita PBR y el diseño general es mucho más sencillo.

También hay una solución de compromiso que no requiere el uso de PBR, sino que utiliza el ruteo flotante estático en su lugar.

Verificación

Actualmente, no hay un procedimiento de verificación disponible para esta configuración.

Troubleshoot

Actualmente, no hay información específica de troubleshooting disponible para esta configuración.

Información Relacionada

- Soporte Técnico y Documentación - Cisco Systems
- Cisco IOS 15.3 M&T - Cisco Systems