Routage des politiques et son impact sur les paquets ESP et ISAKMP avec Cisco IOS

Contenu

Introduction

Ce document décrit l'effet des PBR (Policy Based Routing) et PBR local lorsqu'ils sont appliqués aux paquets ESP (Encapsulating Security Payload) et ISAKMP (Internet Security Association and Key Management Protocol) lorsque vous utilisez Cisco IOS®.

Contribué par Michal Garcarz, ingénieur TAC Cisco.

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- Cisco IOS
- Configuration VPN sur Cisco IOS

Components Used

Les informations de ce document sont basées sur la version 15.x de Cisco IOS.

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.

Informations générales

Avant l'établissement du tunnel IPsec, le routeur initie un échange ISAKMP. Lorsque ces paquets sont générés par le routeur, ils sont traités comme du trafic généré localement et toute décision PBR locale est appliquée. En outre, tous les paquets générés par le routeur (Enhanced Interior Gateway Routing Protocol (EIGRP), Next Hop Resolution Protocol (NHRP), Border Gateway Protocol (BGP) ou Internet Control Message Protocol (ICMP)) sont également considérés comme du trafic généré localement et la décision PBR locale est appliquée.

Le trafic qui est transféré par le routeur et envoyé à travers le tunnel, appelé trafic de transit, n'est pas considéré comme trafic généré localement, et toute politique de routage souhaitée doit être appliquée sur l'interface d'entrée du routeur.

Cela a des conséquences sur le trafic qui traverse le tunnel : le trafic généré localement suit le PBR, mais pas le trafic de transit. Cet article explique les conséquences de cette différence de comportement.

Pour le trafic de transit qui doit être encapsulé ESP, il n'est pas nécessaire d'avoir des entrées de routage car PBR détermine l'interface de sortie du paquet avant et après l'encapsulation ESP. Pour le trafic généré localement qui doit être encapsulé ESP, il est nécessaire d'avoir des entrées de routage, car le PBR local détermine l'interface de sortie uniquement pour le paquet avant l'encapsulation et le routage détermine l'interface de sortie pour le paquet post-encapsulé.

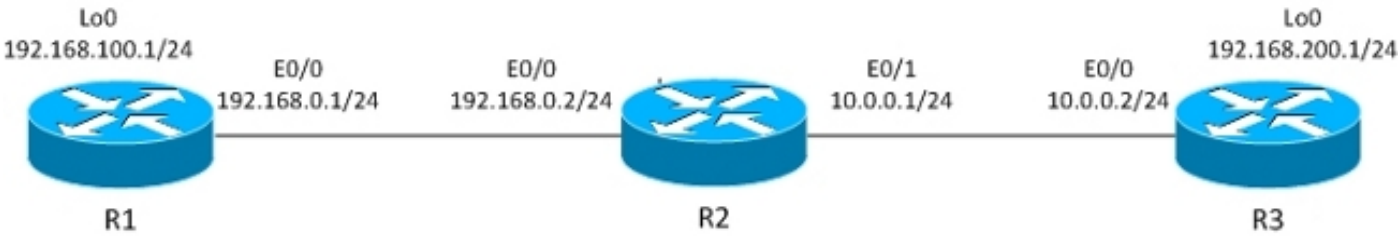
Ce document contient un exemple type de configuration dans lequel un routeur avec deux liaisons ISP est utilisé. Une liaison est utilisée afin d'accéder à Internet et la seconde est pour VPN. En cas de défaillance d’une liaison, le trafic est réacheminé avec une autre liaison du fournisseur d’accès à Internet (FAI). Des pièges sont également présentés.

Notez que le PBR est exécuté dans Cisco Express Forwarding (CEF), alors que le PBR local est commuté par processus.

Trafic généré localement sur le routeur

Cette section décrit le comportement du trafic initié à partir du routeur (R)1. Ce trafic est ESP encapsulé par R1.

Topologie



Le tunnel IPsec LAN à LAN est construit entre R1 et R3.

Le trafic intéressant se situe entre R1 Lo0 (192.168.100.1) et R3 Lo0 (192.168.200.1).

Le routeur R3 dispose d'une route par défaut vers R2.

R1 n'a pas d'entrées de routage, mais seulement des réseaux connectés directement.

Configuration

R1 a un PBR local pour tout le trafic :

```
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
```

Déboguages

Tout le trafic généré localement sur R1 est envoyé à R2 lorsqu'il est UP.

Afin de vérifier ce qui se produit lorsque vous ouvrez le tunnel, envoyez le trafic intéressant à partir du routeur lui-même :

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

> **Attention** : La commande **debug ip packet** peut générer une grande quantité de débogages et avoir un impact énorme sur l'utilisation du CPU. Utilisez-la avec prudence.

Ce débogage permet également d'utiliser la liste d'accès afin de limiter la quantité de trafic traitée par les débogages. La commande **debug ip packet** affiche uniquement le trafic commuté par processus.

Voici les débogages sur 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
```

Voici ce qui se passe :

Le trafic intéressant (192.168.100.1 > 192.168.200.1) est mis en correspondance avec le PBR local et l'interface de sortie est déterminée (E0/0). Cette action déclenche l'initialisation d'ISAKMP par le code de chiffrement. Ce paquet est également routé par un PBR local, qui détermine l'interface de sortie (E0/0). Le trafic ISAKMP est envoyé et le tunnel est négocié

Que se passe-t-il lorsque vous envoyez une nouvelle requête ping ?

```
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)
```

Voici ce qui se passe :

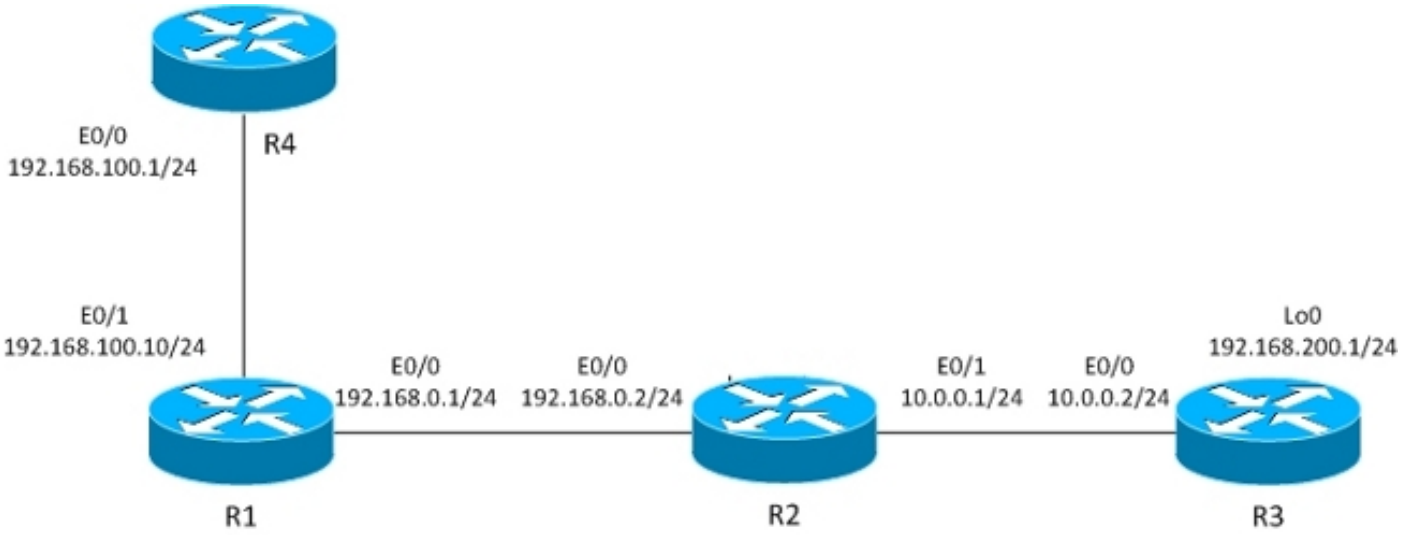
Le trafic intéressant généré localement, 192.168.100.1 > 192.168.200.1, est routé localement par la stratégie et l'interface de sortie est déterminée (E0/0). Le paquet est consommé par la fonctionnalité de sortie IPsec sur E0/0 et encapsulé. Le paquet encapsulé (de 192.168.0.1 à 10.0.0.2) est vérifié pour le routage afin de déterminer l'interface de sortie, mais il n'y a rien dans les tables de routage de R1, ce qui explique pourquoi l'encapsulation échoue.

Dans ce scénario, le tunnel est UP, mais le trafic n'est pas envoyé car, après l'encapsulation ESP, Cisco IOS vérifie les tables de routage afin de déterminer l'interface de sortie.

Trafic de transit via le routeur

Cette section décrit le comportement du trafic de transit qui passe par le routeur, qui est ESP encapsulé par ce routeur.

Topologie



Le tunnel L2L est construit entre R1 et R3.

Le trafic intéressant est entre R4 (192.168.100.1) et R3 lo0 (192.168.200.1).

Le routeur R3 dispose d'une route par défaut vers R2.

Le routeur R4 dispose d'une route par défaut vers R1.

R1 n'a pas de routage.

Configuration

La topologie précédente est modifiée afin d'afficher le flux lorsque le routeur reçoit des paquets pour le chiffrement (trafic de transit au lieu du trafic généré localement).

Pour le moment, le trafic intéressant reçu de R4 est routé par des politiques sur R1 (par PBR sur E0/1), et il existe également un routage de politiques locales pour tout le trafic :

```
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
```

Déboguages

Afin de vérifier ce qui se passe lorsque vous ouvrez le tunnel sur R1 (après avoir reçu le trafic intéressant de R4), entrez :

```
R1#debug ip packet
```

```
R4#ping 192.168.200.1
```

Voici les débogages sur 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
```

Voici ce qui se passe :

Le trafic intéressant atteint PBR sur E0/0 et déclenche l'envoi du code de chiffrement au paquet ISAKMP. Ce paquet ISAKMP est routé localement par une stratégie et l'interface de sortie est déterminée par le PBR local. Un tunnel est construit.

Voici une autre requête ping vers 192.168.200.1 à partir de R4 :

```
R4#ping 192.168.200.1
```

Voici les débogages sur 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
```

Voici ce qui se passe :

Le trafic intéressant atteint PBR sur E0/0 et que PBR détermine l'interface de sortie (E0/0). Sur E0/0, le paquet est consommé par IPSec et encapsulé. Une fois que le paquet encapsulé est vérifié par rapport à la même règle PBR et que l'interface de sortie est déterminée, le paquet est envoyé et reçu correctement.

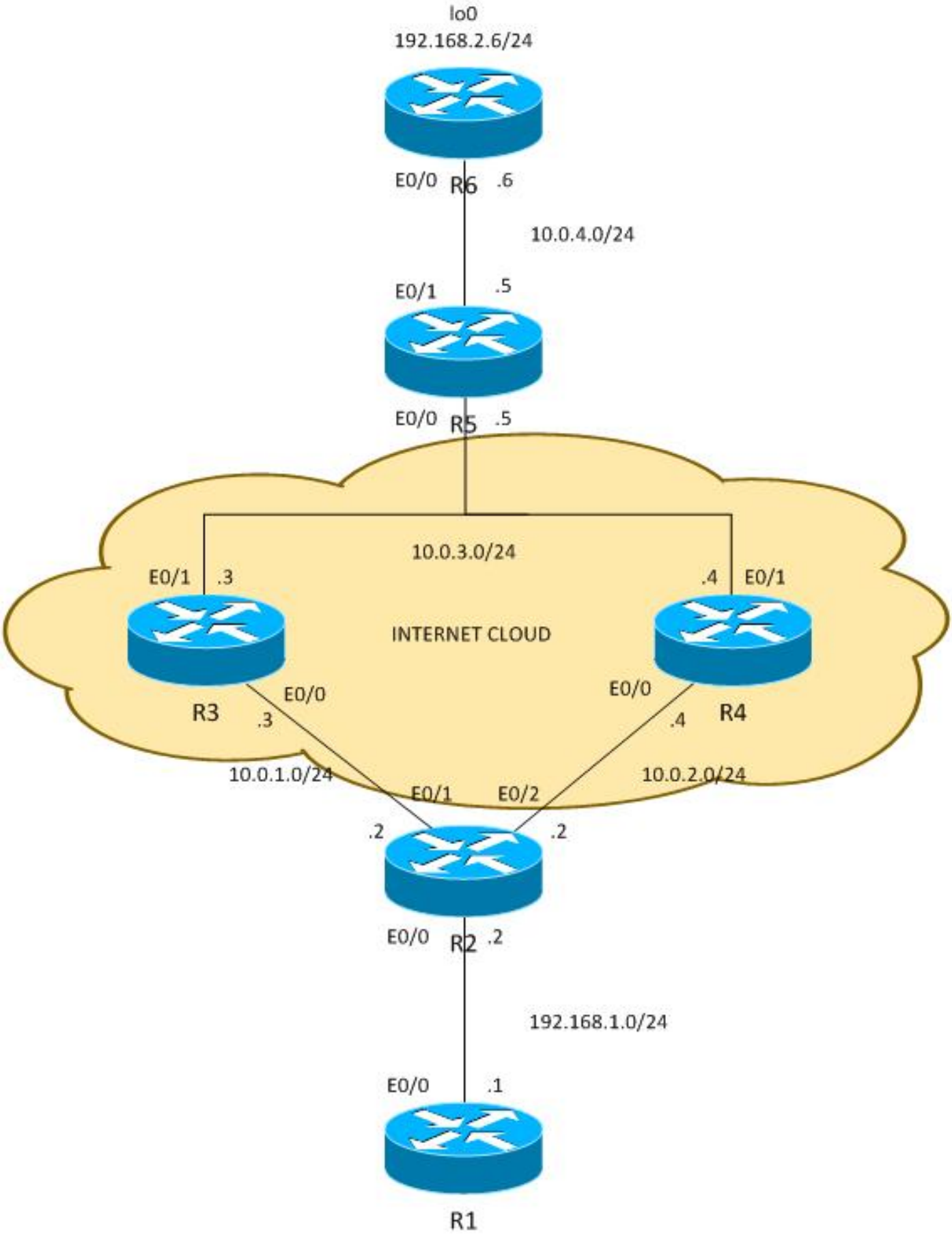
Résumé des différences de comportement

Pour le trafic généré localement, l'interface de sortie pour le trafic non encapsulé (ISAKMP) est déterminée par le PBR local. Pour le trafic généré localement, l'interface de sortie pour le trafic post-encapsulé (ESP) est déterminée par les tables de routage (le PBR local n'est pas vérifié). Pour le trafic de transit, l'interface de sortie pour le trafic post-encapsulé (ESP) est déterminée par le PBR de l'interface (deux fois, avant et après l'encapsulation).

Exemple de configuration

Voici un exemple de configuration pratique qui présente les problèmes que vous pourriez rencontrer avec PBR et PBR local avec VPN. Le routeur R2 (CE) possède deux liaisons ISP. Le routeur R6 dispose également d’une liaison CE et d’une liaison ISP. La première liaison entre R2 et R3 est utilisée comme route par défaut pour R2. La deuxième liaison vers R4 est utilisée uniquement pour le trafic VPN vers R6. En cas de défaillance d’une liaison ISP, le trafic est réacheminé vers l’autre liaison.

Topologie



Configuration

Le trafic entre 192.168.1.0/24 et 192.168.2.0/24 est protégé. Le protocole OSPF (Open Shortest Path First) est utilisé dans le cloud Internet afin d'annoncer les adresses 10.0.0.0/8, qui sont

traitées comme des adresses publiques attribuées par le FAI au client. Dans le monde réel, le protocole BGP est utilisé à la place du protocole OSPF.

La configuration sur R2 et R6 est basée sur la crypto-carte. Sur R2, PBR est utilisé sur E0/0 afin de diriger le trafic VPN vers R4 s'il est 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
```

Ici, vous voyez que le PBR local n'est pas nécessaire. L'interface PBR achemine le trafic intéressant vers 10.0.2.4. Cela déclenche le code de chiffrement pour lancer ISAKMP à partir de l'interface correcte (liaison à R4), même lorsque le routage est destiné à des points homologues distants via R3.

Sur R6, deux homologues pour le VPN sont utilisés :

```
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 utilise un accord de niveau de service (SLA) IP afin d'envoyer une requête ping à R3 et R4. La route par défaut est R3. En cas de défaillance de R3, il choisit 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
```

R2 autorise également l’accès à Internet pour tous les utilisateurs internes. Afin d’obtenir la redondance dans le cas où le routeur ISP vers R3 est en panne, une route-map est nécessaire. Il Port Address Translations (PAT) à l’intérieur du trafic vers une interface de sortie différente (PAT vers interface E0/1 lorsque R3 est UP et que la route par défaut pointe vers R3, et PAT vers l’interface E0/2 lorsque R3 est arrêté et R4 est utilisé comme route par défaut).

```
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
```

Le trafic VPN doit être exclu de la traduction, tout comme ISAKMP. Si le trafic ISAKMP n'est pas exclu de la traduction, c'est PATed à l'interface externe qui va vers 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
```

Test

Avec cette configuration, il y a une redondance complète. Le VPN utilise la liaison R4 et le reste du trafic est routé avec R3. En cas de défaillance de R4, le trafic VPN est établi avec la liaison R3 (la route-map pour PBR ne correspond pas et le routage par défaut est utilisé).

Avant que le FAI de R4 ne soit arrêté, R6 voit le trafic de l'homologue 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
```

Après que R2 utilise ISP à R3 pour le trafic VPN, R6 voit le trafic de l'homologue 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
```

Dans le scénario inverse, lorsque la liaison vers R3 est interrompue, tout fonctionne toujours correctement. Le trafic VPN utilise toujours la liaison vers R4. La traduction d'adresses de réseau (NAT) est effectuée pour 192.168.1.0/24 vers PAT afin d'attribuer l'adresse externe. Avant que R3 ne tombe en panne, il existe une traduction vers 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
```

Après la panne de R3, il reste l'ancienne traduction ainsi que la nouvelle traduction (vers 10.0.2.2) qui utilise la liaison vers 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
```

Les pièges

Si tout fonctionne bien, où sont les pièges ? Ils sont dans les détails.

Trafic généré localement

Voici un scénario qui doit initier le trafic VPN à partir de R2 lui-même. Ce scénario nécessite que vous configuriez un PBR local sur R2 afin de forcer R2 à envoyer le trafic ISAKMP via R4 et de provoquer l'activation du tunnel. Mais l'interface de sortie est déterminée avec l'utilisation de tables de routage, avec la valeur par défaut pointant vers R3, et ce paquet est envoyé à R3, au lieu de R4, qui est utilisé pour le transit pour VPN. Pour vérifier cela, saisissez :

```
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
```

Dans cet exemple, le protocole ICMP (Internet Control Message Protocol) qui est généré localement est forcé via R4. Sans cela, le trafic généré localement de 192.168.1.2 à 192.168.2.5 est traité à l'aide de tables de routage et un tunnel est établi avec R3.

Que se passe-t-il après l'application de cette configuration ? Le paquet ICMP de 192.168.1.2 à 192.168.2.5 est placé vers R4 et un tunnel est initié avec la liaison vers R4. Le tunnel est configuré :

```
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
```

Tout semble fonctionner correctement. Le trafic est envoyé avec la liaison E0/2 correcte vers R4. Même R6 montre que le trafic est reçu de 10.2.2.2, qui est l'adresse IP de liaison 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
```

Mais en fait, il y a un routage asymétrique pour les paquets ESP ici. Les paquets ESP sont envoyés avec 10.0.2.2 comme source, mais sont placés sur la liaison vers R3. Une réponse chiffrée est renvoyée via R4. Vous pouvez vérifier cela en vérifiant les compteurs sur R3 et R4 :

Les compteurs R3 de E0/0 avant l'envoi de 100 paquets :

```
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
```

Et les mêmes compteurs, après avoir envoyé 100 paquets :

```
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
```

Le nombre de paquets entrants a augmenté de 100 (sur la liaison vers R2), mais les paquets sortants n'ont augmenté que de 2. R3 ne voit donc que l'écho ICMP chiffré.

La réponse est visible sur R4, avant d'envoyer 100 paquets :

```
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
```

Après avoir envoyé 100 paquets :

```
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
```

Le nombre de paquets envoyés vers R2 a augmenté de 102 (réponse ICMP chiffrée), tandis que les paquets reçus ont augmenté de 0. R4 ne voit donc que la réponse ICMP chiffrée. Bien sûr, une capture de paquets le confirme.

Que se passe-t-il ? La réponse se trouve dans la première partie de l'article.

Voici le flux de ces paquets ICMP :

1. Le protocole ICMP de 192.168.1.2 à 192.168.2.6 est placé sur E0/2 (liaison vers R4) en raison du PBR local.
2. La session ISAKMP est construite avec 10.0.2.2 et mise sur la liaison E0/2 comme prévu.
3. Pour les paquets ICMP après l'encapsulation, le routeur doit déterminer l'interface de sortie, ce qui se fait avec l'utilisation de tables de routage qui pointent vers R3. C'est pourquoi le paquet chiffré avec la source 10.0.2.2 (liaison vers R4) est envoyé via R3.
4. R6 reçoit un paquet ESP de 10.0.2.2, qui est cohérent avec la session ISAKMP, déchiffre le paquet et envoie la réponse ESP à 10.0.2.2.
5. En raison du routage, R5 renvoie une réponse à 10.0.2.2 via R4.
6. R2 le reçoit et le déchiffre, et le paquet est accepté.

C'est pourquoi il est important d'être très prudent avec le trafic généré localement.

Dans de nombreux réseaux, le protocole uRPF (Unicast Reverse Path Forwarding) est utilisé et le trafic provenant de 10.0.2.2 peut être abandonné sur E0/0 de R3. Dans ce cas, la requête ping ne fonctionne pas.

Y a-t-il une solution à ce problème ? Il est possible de forcer le routeur à traiter le trafic généré localement comme trafic de transit. Pour cela, le PBR local doit diriger le trafic vers une interface

de bouclage erronée à partir de laquelle il est routé comme le trafic de transit.

Ce n'est pas conseillé.

> **Note**: Il est important d'être très prudent lorsque vous utilisez NAT avec PBR (reportez-vous à la section précédente sur le trafic ISKMP dans la liste d'accès PAT).

Exemple de configuration sans PBR

Il y a aussi une autre solution qui est un compromis. Avec la même topologie que l'exemple précédent, il est possible de satisfaire à toutes les exigences sans utiliser de PBR ou de PBR local. Pour ce scénario, seul le routage est utilisé. Une seule entrée de routage supplémentaire est ajoutée sur R2 et toutes les configurations PBR/PBR local sont supprimées :

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

Au total, R2 dispose de cette configuration de routage :

```
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 première entrée de routage est un routage par défaut vers R3, lorsque la liaison vers R3 est UP. La deuxième entrée de routage est une route de secours par défaut vers R4, lorsque la liaison vers R3 est arrêtée. La troisième entrée décide de la manière dont le trafic vers le réseau VPN distant est envoyé, selon l'état de la liaison R4 (si la liaison R4 est UP, le trafic vers le réseau VPN distant est envoyé via R4). Avec cette configuration, il n'est pas nécessaire d'effectuer un routage de stratégie.

Quel est l'inconvénient ? Il n'y a plus de contrôle granulaire utilisant PBR. Il n'est pas possible de déterminer l'adresse source. Dans ce cas, tout le trafic vers 192.168.2.0/24 est envoyé vers R4 lorsqu'il est UP, quelle que soit la source. Dans l'exemple précédent, ceci était contrôlé par PBR et la source spécifique : 192.168.1.0/24 est sélectionné.

Pour quel scénario cette solution est-elle trop simple ? Pour plusieurs réseaux LAN (derrière R2). Lorsque certains de ces réseaux doivent atteindre 192.168.2.0/24 de manière sécurisée (cryptée) et d'autres moyens non sécurisés (non cryptés), le trafic provenant de réseaux non sécurisés est toujours placé sur l'interface E0/2 de R2 et ne touche pas la crypto-carte. Il est donc envoyé non chiffré via une liaison à R4 (et la principale condition requise était d'utiliser R4 uniquement pour le trafic chiffré).

Ce type de scénario et ses exigences sont rares, c'est pourquoi cette solution est utilisée assez souvent.

Résumé

L'utilisation de fonctions PBR et PBR locaux, ainsi que de VPN et de NAT, peut être complexe et nécessite une compréhension approfondie du flux de paquets.

Dans le cas de scénarios tels que ceux présentés ici, il est conseillé d'utiliser deux routeurs distincts, chacun d'entre eux disposant d'une liaison ISP. En cas de défaillance d'un FAI, le trafic peut être réacheminé facilement. Il n'est pas nécessaire d'utiliser un PBR et sa conception globale est beaucoup plus simple.

Il existe également une solution de compromis qui ne nécessite pas l'utilisation de PBR, mais qui utilise le routage statique flottant à la place.

Vérification

Aucune procédure de vérification n'est disponible pour cette configuration.

Dépannage

Il n'existe actuellement aucune information de dépannage spécifique pour cette configuration.

Informations connexes

- Support et documentation techniques - Cisco Systems
- Cisco IOS 15.3 M&T - Cisco Systems