Routage de stratégie et son incidence sur des paquets de l'ESP et de l'ISAKMP avec le Cisco IOS

Contenu

Introduction

Ce document décrit l'effet du Routage à base de règles (PBR) et des gens du pays PBR une fois appliqué aux paquets de Protocole ESP (Encapsulating Security Payload) et de Protocole ISAKMP (Internet Security Association and Key Management Protocol) quand vous utilisez le 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 du VPN sur le Cisco IOS

Composants utilisés

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

Les informations contenues dans ce document ont été créées à partir des périphériques d'un environnement de laboratoire spécifique. Tous les périphériques utilisés dans ce document ont démarré avec une configuration effacée (par défaut). Si votre réseau est opérationnel, assurez-vous que vous comprenez l'effet potentiel de toute commande.

Informations générales

Avant l'établissement de tunnel d'IPsec, le routeur initie un échange d'ISAKMP. Pendant que ces paquets sont générés par le routeur, les paquets sont traités pendant que le trafic localement généré et toutes les décisions des gens du pays PBR sont appliqués. En outre, tous les paquets générés par le routeur (des pings de Protocole EIGPR (Enhanced Interior Gateway Routing Protocol), de Protocole NHRP (Next Hop Resolution Protocol), de Protocole BGP (Border Gateway Protocol), ou de Protocole ICMP (Internet Control Message Protocol)) sont également considérés en tant que trafic localement généré et ont la décision des gens du pays PBR appliquée.

Trafiquez qu'est expédié par le routeur et envoyé à travers le tunnel, qui s'appelle le trafic de transit, n'est pas considéré le trafic localement généré, et n'importe quelle stratégie de acheminement désirée doit être appliquée sur l'interface d'entrée du routeur.

Les implications que ceci a sur le trafic qui traverse le tunnel est que le trafic localement généré suit PBR, mais le trafic de transit ne fait pas. Cet article explique les conséquences de cette différence dans le comportement.

Pour le trafic de transit qui doit être l'ESP encapsulé, il n'y a aucun besoin de n'avoir aucune entrée de routage parce que PBR détermine l'interface de sortie pour le paquet avant et après l'encapsulation de l'ESP. Pour le trafic localement généré qui doit être l'ESP encapsulé, il est nécessaire d'avoir des entrées de routage, parce que les gens du pays PBR déterminent l'interface de sortie seulement pour le paquet avant l'encapsulation et le routage détermine l'interface de sortie pour le paquet POST-encapsulé.

Ce document contient un exemple typique de configuration où un routeur avec deux liens ISP est utilisé. Un lien est utilisé afin d'accéder à l'Internet et le deuxième est pour le VPN. En cas de n'importe quelle panne de lien, le trafic est rerouté avec un lien différent de fournisseur de services

Internet (ISP). Des pièges sont également présentés.

Veuillez noter que PBR est exécuté dans le Technologie Cisco Express Forwarding (CEF), tandis que les gens du pays PBR sont commutés par processus.

Le trafic généré localement sur le routeur

Cette section décrit le comportement du trafic initié du routeur (R)1. Que le trafic est l'ESP s'est encapsulé par R1.

Topologie

Le tunnel entre réseaux locaux d'IPsec est établi entre R1 et R3.

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

Le routeur R3 a un default route à R2.

R1 n'a aucune entrée de routage, seulement directement des réseaux connectés.

Configuration

R1 a les gens du pays PBR 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
```

Debugs

Tout le trafic localement généré sur R1 est envoyé à R2 quand il est EN HAUSSE.

Afin de vérifier ce qui se produit quand vous apportez le tunnel, envoyez le trafic intéressant du routeur lui-même :

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

> **Attention** : La commande de **debug ip packet** pourrait générer l'un grand nombre met au point et a l'incidence énorme sur l'utilisation du CPU. Utilisez-le avec prudence.

Ceci mettent au point permet également la liste d'accès à utiliser afin de limiter le niveau de trafic traité par met au point. Les affichages de commande de **debug ip packet** seulement trafiquent qui est commuté par processus.

Voici met au point 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 produit :

Le trafic intéressant (192.168.100.1 > 192.168.200.1) est apparié par les gens du pays PBR, et l'interface de sortie est déterminé (E0/0). Cette action déclenche le crypto code pour initier l'ISAKMP. Ce paquet stratégie-est également conduit par les gens du pays PBR, qui déterminent l'interface de sortie (E0/0). Le trafic d'ISAKMP est envoyé, et le tunnel est négocié

Que se produit quand vous cinglez de nouveau ?

```
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 produit :

Le trafic intéressant localement généré, 192.168.100.1 > 192.168.200.1, stratégie-est localement conduit, et l'interface de sortie est déterminée (E0/0). Le paquet est consommé par la caractéristique de sortie d'IPsec sur E0/0 et encapsulé. Le paquet encapsulé (de 192.168.0.1 à 10.0.0.2) est routage vérifié afin de déterminer l'interface de sortie, mais là n'est rien dans les tables de routage de R1, qui est pourquoi l'encapsulation échoue.

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

Le trafic de transit par le routeur

Cette section décrit le comportement pour le trafic de transit qui est livré par le routeur, qui est l'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 a un default route à R2.

Le routeur R4 a un default route à R1.

R1 n'a aucun routage.

Configuration

La topologie précédente est modifiée afin d'afficher l'écoulement quand le routeur reçoit des paquets pour le cryptage (le trafic de transit au lieu du trafic localement généré).

En ce moment, le trafic intéressant reçu de R4 stratégie-est conduit sur R1 (par PBR sur E0/1), et il y a également routage local de stratégie pour tout le trafic :

```
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

Afin de vérifier ce qui se produit quand vous apportez le tunnel sur R1 (après que vous recevez le trafic intéressant de R4), entre :

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

Voici met au point 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 produit :

Le trafic intéressant frappe PBR sur E0/0 et code de déclencheurs crypto pour envoyer le paquet d'ISAKMP. Que le paquet d'ISAKMP stratégie-est localement conduit, et l'interface de sortie est déterminé par les gens du pays PBR. Un tunnel est construit.

Voici un davantage ping à 192.168.200.1 de R4 :

```
R4#ping 192.168.200.1
```

Voici met au point 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 produit :

Le trafic intéressant frappe PBR sur E0/0, et ce PBR détermine l'interface de sortie (E0/0). Sur E0/0, le paquet est consommé par IPSec et encapsulé. Après que le paquet encapsulé soit vérifié contre la même règle PBR et l'interface de sortie est déterminée, le paquet est envoyé et reçu correctement.

Résumé pour des différences de comportement

Pour le trafic localement généré, l'interface de sortie pour le trafic non-encapsulé (ISAKMP) est déterminée par les gens du pays PBR. Pour le trafic localement généré, l'interface de sortie pour le trafic POST-encapsulé (ESP) est déterminée par les tables de routage (les gens du pays PBR ne sont pas vérifiés). Pour le trafic de transit, l'interface de sortie pour le trafic POST-encapsulé (ESP) est déterminée par l'interface PBR (deux fois, avant et après l'encapsulation).

Exemple de configuration

C'est un exemple pratique de configuration qui présente les questions que vous pourriez faire face avec PBR et gens du pays PBR avec le VPN. Le R2 (CE) a deux liens ISP. Le routeur R6 a également le CE et un lien ISP. Le premier lien de R2 à R3 est utilisé comme default route pour R2. Le deuxième lien à R4 est utilisé seulement pour le trafic VPN à R6. En cas de n'importe quelle panne de lien ISP, le trafic est rerouté à l'autre lien.

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 nuage Internet afin d'annoncer les 10.0.0.0/8 adresses, qui sont traitées comme annonces publiques assignées par ISP au client. Dans le monde réel, le BGP est utilisé au lieu de l'OSPF.

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

```
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
```

Voici que vous voyez que les gens du pays PBR ne sont pas nécessaires. L'interface PBR conduit le trafic intéressant à 10.0.2.4. Cela déclenche le crypto code pour initier l'ISAKMP de l'interface appropriée (lien à R4), même lorsque le routage est aux points distants de pair par R3.

Sur R6, deux pairs 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 emploie un accord de niveau de service IP (SLA) afin de cingler R3 et R4. Le default route est R3. En cas de panne R3, il choisit 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
```

Également R2 permet à accès Internet pour tous des utilisateurs d'intérieur. Afin de réaliser la Redondance dans le cas où l'ISP à R3 est vers le bas, un route-map est nécessaire. Il le trafic d'intérieur de translations d'adresses d'adresse du port (tapotements) à une interface de sortie différente (PAT à l'interface E0/1 quand R3 est HAUT et le default route indique R3, et à PAT pour relier E0/2 quand R3 est vers le bas et R4 est utilisé comme default route).

```
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 de même que fait ISAKMP. Si le trafic d'ISAKMP n'est pas exclu de la traduction, c'est PATed à l'interface extérieure 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 pleine Redondance. Le VPN utilise le lien R4, et le reste du trafic est conduit avec R3. En cas de panne R4, le trafic VPN est établi avec le lien R3 (le route-map pour PBR ne s'assortit pas et le routage de par défaut est utilisé).

Avant que l'ISP à R4 soit vers le bas, R6 voit le trafic du pair 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 l'ISP à R3 pour le trafic VPN, R6 voit le trafic du pair 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
```

Pour le scénario opposé, quand le lien à R3 descend, tout fonctionne bien toujours. Le trafic VPN utilise toujours le lien à R4. Le Traduction d'adresses de réseau (NAT) est exécuté pour que 192.168.1.0/24 TAPOTENT afin de s'approprier l'adresse d'extérieur. Avant que R3 descende, il y a une traduction à 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 que R3 descende, il reste la vieille traduction avec la nouvelle traduction (à 10.0.2.2) cette des utilisations le lien 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
```

Pièges

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

Le trafic généré localement

Voici un scénario qui doit initier le trafic VPN du R2 lui-même. Ce scénario exige de vous de configurer les gens du pays PBR sur R2 afin de forcer R2 pour envoyer le trafic d'ISAKMP par l'intermédiaire de R4 et pour faire monter le tunnel. Mais l'interface de sortie est déterminée avec l'utilisation des tables de routage, avec le par défaut indiquant R3, et ce paquet est envoyé à R3, au lieu de R4, qui est utilisé pour le transit pour le VPN. Afin de vérifier cela, entrez :

```
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 par l'intermédiaire de R4. Sans ce, le trafic généré localement de 192.168.1.2 à 192.168.2.5 est traité avec l'utilisation des tables de routage et un tunnel est établi avec R3.

Que se produit après que vous appliquiez cette configuration ? Le paquet d'ICMP de 192.168.1.2 à 192.168.2.5 est mis vers R4, et un tunnel est initié avec le lien à R4. Le tunnel est installé :

```
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 le lien correct E0/2 vers R4. Même R6 prouve que le trafic est reçu de 10.2.2.2, qui est adresse IP de lien 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 réellement, il y a **routage asymétrique pour des paquets de l'ESP** ici. Des paquets de l'ESP sont envoyés avec 10.0.2.2 comme source, mais sont mis sur le lien vers R3. Une réponse chiffrée est renvoyée par R4. Ceci peut être vérifié en vérifiant des compteurs sur R3 et R4 :

Compteurs R3 d'E0/0 avant d'envoyer 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 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
    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 entrant a augmenté de 100 (sur le lien vers R2), mais les paquets sortants ont augmenté seulement de 2. Ainsi R3 voit seulement l'écho chiffré d'ICMP.

La réponse est vue 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 envoi de 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 chiffrée d'ICMP), alors que les paquets reçus augmentaient de 0. Ainsi R4 voit seulement la réponse chiffrée d'ICMP. Naturellement, une capture de paquet confirme ceci.

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

Voici l'écoulement de ces paquets d'ICMP :

1. L'ICMP de 192.168.1.2 à 192.168.2.6 est mis sur E0/2 (lien vers R4) en raison des gens du pays PBR.
2. La session d'ISAKMP est établie avec 10.0.2.2 et en fonction mis le lien E0/2 comme prévue.

3. Pour des paquets d'ICMP après l'encapsulation, le routeur doit déterminer l'interface de sortie, qui est faite avec l'utilisation des tables de routage qui indiquent R3. C'est pourquoi le paquet chiffré avec la source 10.0.2.2 (lien vers R4) est envoyé par l'intermédiaire de R3.
4. R6 reçoit un paquet de l'ESP de 10.0.2.2, qui est compatible à la session d'ISAKMP, déchiffre le paquet, et envoie la réponse de l'ESP à 10.0.2.2.
5. En raison du routage, R5 renvoie une réponse à 10.0.2.2 par R4.
6. R2 reçoit lui et des déchiffrages, et le paquet est reçu.

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

Dans beaucoup de réseaux, le Fonction Unicast Reverse Path Forwarding (uRPF) est utilisé et le trafic originaire de 10.0.2.2 pourrait être abandonné sur E0/0 de R3. Dans ce cas, le ping ne fonctionne pas.

Y a-t-il une solution pour ce problème ? Il est possible de forcer le routeur à traiter le trafic localement généré comme trafic de transit. Pour le ce, les gens du pays PBR doivent se diriger le trafic à une interface factice de bouclage de laquelle ils sont conduits comme le trafic de transit.

Ceci n'est pas informé.

> Remarque: Il est important de faire attention extra quand vous utilisez NAT avec PBR (référez-vous à la section précédente au sujet du trafic ISKMP en liste d'accès de PAT).

Exemple de configuration sans PBR

Il y a également une autre solution qui est une compromission. Avec la même topologie que l'exemple précédent, il est possible de répondre à toutes les exigences sans utilisation de PBR ou de gens du pays PBR. Pour ce scenarrio, seulement conduisant est utilisé. Seulement une plus d'entrée de routage est ajoutée sur R2, et toutes les configurations PBR/local PBR sont retirées :

```
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
```

Au total, R2 a cette configuration de routage :

```
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
```

La première entrée de routage est un routage par défaut vers R3, quand le lien à R3 est EN HAUSSE. La deuxième entrée de routage est un default route de sauvegarde vers R4, quand le lien à R3 est en baisse. La troisième entrée décide quel trafic de manière au réseau VPN distant est envoyé, selon l'état de lien R4 (si le lien R4 est EN HAUSSE, le trafic au réseau VPN distant est envoyé par l'intermédiaire de R4). Avec cette configuration, il n'y a aucun besoin de routage de stratégie.

Quel est l'inconvénient ? Il n'y a désormais aucun contrôle granulaire utilisant PBR. Il n'est pas

possible de déterminer l'adresse source. Dans ce cas, tout le trafic à 192.168.2.0/24 est envoyé vers R4 quand il est EN HAUSSE, indépendamment de la source. Dans l'exemple précédent, cela a été contrôlé par PBR et la source spécifique : 192.168.1.0/24 est sélectionnés.

Pour quel scénario cette solution est-elle trop simple ? Pour de plusieurs réseaux de RÉSEAU LOCAL (derrière R2). Quand certains de ces réseaux doivent atteindre 192.168.2.0/24 d'un moyen sûr (chiffré) et d'autres manières non sécurisées (décryptées), le trafic des réseaux non sécurisés est toujours mis sur l'interface E0/2 de R2 et ne frappe pas le crypto map. Ainsi il est envoyé à décrypté par l'intermédiaire d'un lien à R4 (et à l'exigence fondamentale était utiliser R4 seulement pour le trafic chiffré).

Ce genre de scénario et ses conditions requises sont rares, qui est pourquoi cette solution est utilisée assez souvent.

Résumé

Utilisant PBR et caractéristiques locales PBR avec des VPN et NAT pourrait être complexe et exige une compréhension en profondeur de l'écoulement de paquet.

Pour des scénarios comme ceux présentés ici, on lui informe utiliser deux Routeurs distincts - chaque routeur avec un lien ISP. En cas de panne ISP, le trafic peut être rerouté facilement. Il n'y a aucun besoin de PBR, et la conception globale est beaucoup plus simple.

Il y a également une solution de compromission qui n'exige pas l'utilisation de PBR, mais acheminement flottant statique d'utilisations à la place.

Vérifiez

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

Dépannez

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