Policy Routing e il suo impatto sui pacchetti ESP e ISAKMP con Cisco IOS

Sommario

Introduzione

Questo documento descrive l'effetto del PBR (Policy Based Routing) e del PBR locale quando applicati ai pacchetti Encapsulating Security Payload (ESP) e Internet Security Association and Key Management Protocol (ISAKMP) quando si usa Cisco IOS®.

Contributo di Michal Garcarz, Cisco TAC Engineer.

Prerequisiti

Requisiti

Cisco raccomanda la conoscenza di base dei seguenti argomenti:

- Cisco IOS
- Configurazione VPN su Cisco IOS

Componenti usati

Il riferimento delle informazioni contenute in questo documento è Cisco IOS versione 15.x.

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

Premesse

Prima di stabilire il tunnel IPsec, il router avvia uno scambio ISAKMP. Quando questi pacchetti vengono generati dal router, vengono trattati come traffico generato localmente e vengono applicate le decisioni PBR locali. Inoltre, tutti i pacchetti generati dal router (Enhanced Interior Gateway Routing Protocol (EIGRP), Next Hop Resolution Protocol (NHRP), Border Gateway Protocol (BGP) o ping Internet Control Message Protocol (ICMP)) vengono considerati come traffico generato localmente e viene applicata la decisione PBR locale.

Il traffico che viene inoltrato dal router e inviato attraverso il tunnel, definito traffico di transito, non viene considerato traffico generato localmente e qualsiasi criterio di routing desiderato deve essere applicato all'interfaccia in entrata del router.

Le implicazioni per il traffico che attraversa il tunnel sono che il traffico generato localmente segue il PBR, a differenza del traffico di transito. Questo articolo spiega le conseguenze di questa differenza di comportamento.

Per il traffico di transito che deve essere incapsulato da ESP, non è necessario avere voci di routing in quanto il PBR determina l'interfaccia di uscita per il pacchetto prima e dopo l'incapsulamento ESP. Per il traffico generato localmente che deve essere incapsulato tramite ESP, sono necessarie voci di routing, in quanto il PBR locale determina l'interfaccia in uscita solo per il pacchetto prima dell'incapsulamento e il routing determina l'interfaccia in uscita per il pacchetto post-incapsulato.

Questo documento contiene un tipico esempio di configurazione in cui viene usato un router con due collegamenti ISP. Un collegamento viene utilizzato per accedere a Internet e il secondo per la VPN. In caso di errore del collegamento, il traffico viene reindirizzato con un collegamento con un altro provider di servizi Internet (ISP). Presentate anche le insidie.

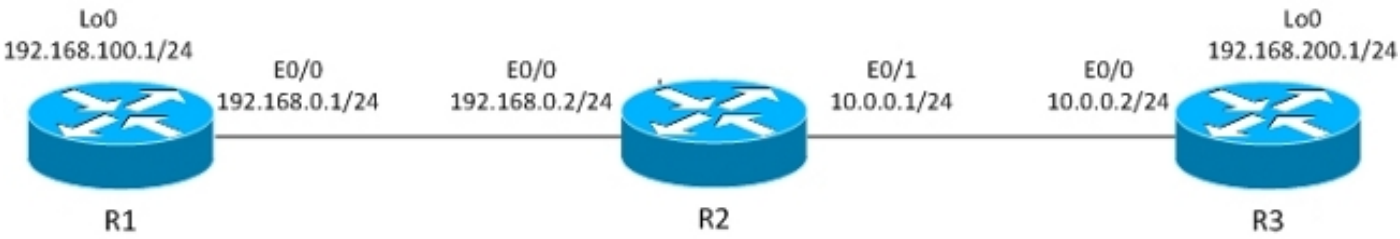
Il PBR viene eseguito in Cisco Express Forwarding (CEF), mentre il PBR locale è a commutazione

di contesto.

Traffico generato localmente sul router

In questa sezione viene descritto il comportamento del traffico avviato dal router (R)1. Tale traffico è ESP incapsulato da R1.

Topologia



Il tunnel IPsec LAN-LAN è costruito tra R1 e R3.

Il traffico interessante è tra R1 Lo0 (192.168.100.1) e R3 Lo0 (192.168.200.1).

Il router R3 ha un percorso predefinito verso R2.

R1 non dispone di voci di routing, ma solo di reti connesse direttamente.

Configurazione

R1 dispone di PBR locale per tutto il traffico:

```
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
```

Debug

Tutto il traffico generato localmente su R1 viene inviato a R2 quando è attivo.

Per verificare cosa succede quando si richiama il tunnel, inviare il traffico interessante dal router stesso:

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

> **Attenzione:** Il comando **debug ip packet** potrebbe generare una grande quantità di debug e avere un impatto notevole sull'utilizzo della CPU. Usatelo con cautela.

Il comando debug consente inoltre di usare l'elenco degli accessi per limitare la quantità di traffico elaborata dai debug. Il comando **debug ip packet** visualizza solo il traffico a commutazione di contesto.

Di seguito sono riportati i debug di 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
```

Ecco cosa succede:

Il traffico interessante (192.168.100.1 > 192.168.200.1) viene abbinato al PBR locale e viene determinata l'interfaccia in uscita (E0/0). Questa azione attiva il codice di crittografia per avviare ISAKMP. Il pacchetto viene inoltre instradato dal PBR locale, che determina l'interfaccia in uscita (E0/0). Il traffico ISAKMP viene inviato e il tunnel viene negoziato

Cosa succede quando si esegue di nuovo il 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)
```

Ecco cosa succede:

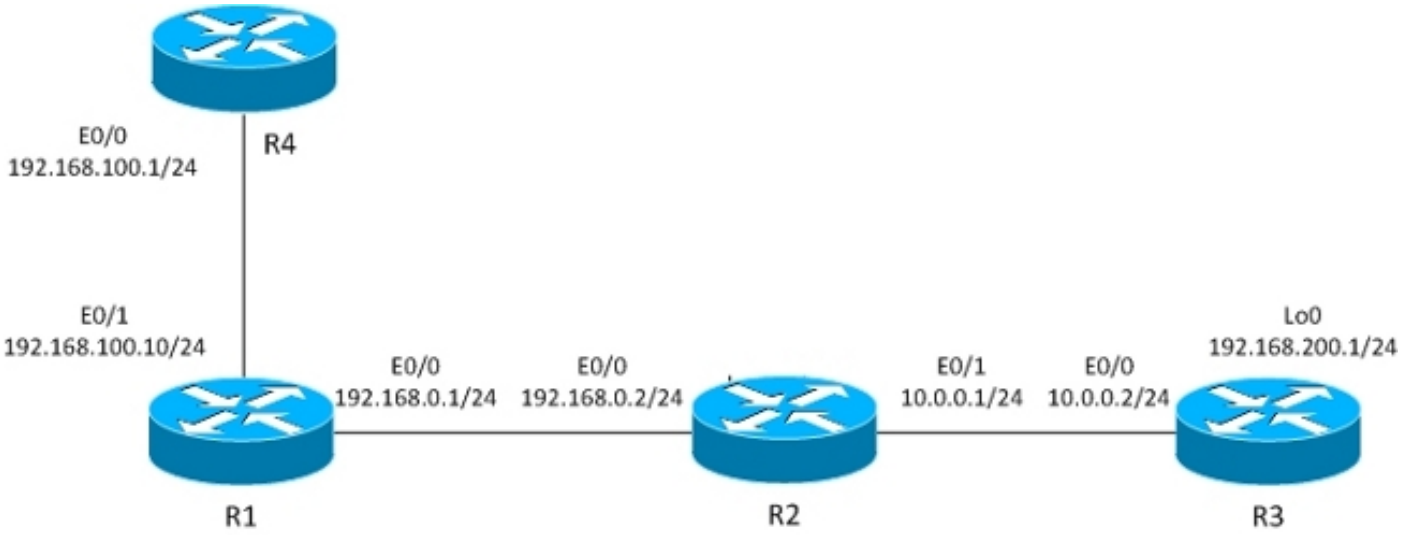
Il traffico interessante generato localmente, 192.168.100.1 > 192.168.200.1, viene instradato localmente sulla base di regole e viene determinata l'interfaccia in uscita (E0/0). Il pacchetto viene consumato dalla funzionalità di output IPsec su E0/0 e incapsulato. Il pacchetto incapsulato (da 192.168.0.1 a 10.0.0.2) viene controllato per verificare la presenza di routing per determinare l'interfaccia di uscita, ma le tabelle di routing di R1 non contengono nulla, ecco perché l'incapsulamento non riesce.

In questo scenario, il tunnel è attivo, ma il traffico non viene inviato perché, dopo l'incapsulamento ESP, Cisco IOS controlla le tabelle di routing per determinare l'interfaccia di uscita.

Traffico di transito attraverso il router

In questa sezione viene descritto il comportamento del traffico di transito che attraversa il router, ossia il protocollo ESP incapsulato da tale router.

Topologia



Il tunnel L2L è costruito tra R1 e R3.

Il traffico interessante è tra R4 (192.168.100.1) e R3 lo0 (192.168.200.1).

Il router R3 ha un percorso predefinito verso R2.

Il router R4 dispone di un percorso predefinito verso R1.

R1 non ha routing.

Configurazione

La topologia precedente viene modificata per mostrare il flusso quando il router riceve i pacchetti per la crittografia (traffico di transito anziché traffico generato localmente).

Al momento, il traffico interessante ricevuto da R4 viene instradato su R1 (da PBR su E0/1), e c'è anche un routing di policy locale per tutto il traffico:

```
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
```

Debug

Per verificare cosa succede quando si apre il tunnel su R1 (dopo aver ricevuto il traffico interessante da R4), immettere:

```
R1#debug ip packet
```

```
R4#ping 192.168.200.1
```

Di seguito sono riportati i debug di 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
```

Ecco cosa succede:

Il traffico interessante raggiunge il PBR su E0/0 e attiva il codice crittografico per inviare il pacchetto ISAKMP. Il pacchetto ISAKMP viene instradato localmente in base ai criteri e l'interfaccia in uscita viene determinata dal PBR locale. Viene costruito un tunnel.

Di seguito viene riportato un altro ping da R4 al numero 192.168.200.1:

```
R4#ping 192.168.200.1
```

Di seguito sono riportati i debug di 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
```

Ecco cosa succede:

Il traffico interessante incontra il PBR su E0/0 e il PBR determina l'interfaccia in uscita (E0/0). Sul router E0/0, il pacchetto viene consumato da IPSec e incapsulato. Dopo aver controllato il pacchetto incapsulato con la stessa regola PBR e aver determinato l'interfaccia in uscita, il pacchetto viene inviato e ricevuto correttamente.

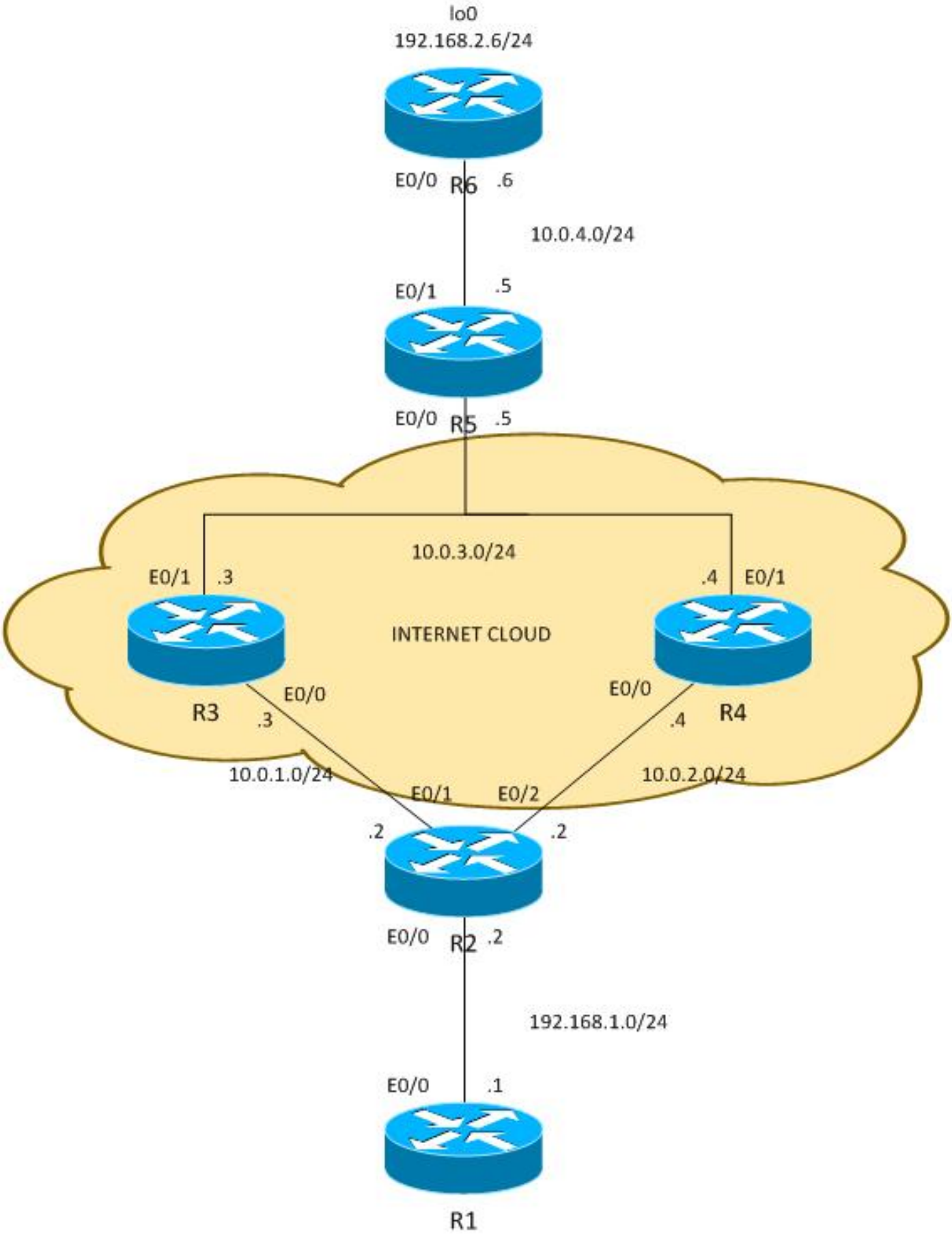
Riepilogo per differenze di comportamento

Per il traffico generato localmente, l'interfaccia in uscita per il traffico non incapsulato (ISAKMP) è determinata dal PBR locale. Per il traffico generato localmente, l'interfaccia in uscita per il traffico post-incapsulato (ESP) è determinata dalle tabelle di routing (il PBR locale non è selezionato). Per il traffico di transito, l'interfaccia in uscita per il traffico post-incapsulato (ESP) è determinata dal PBR dell'interfaccia (due volte, prima e dopo l'incapsulamento).

Esempio di configurazione

Questo è un esempio pratico di configurazione che presenta i problemi che si possono verificare con PBR e PBR locale con VPN. R2 (CE) ha due collegamenti ISP. Il router R6 dispone inoltre di un collegamento CE e di un ISP. Il primo collegamento da R2 a R3 viene utilizzato come percorso predefinito per R2. Il secondo collegamento a R4 viene utilizzato solo per il traffico VPN verso R6. In caso di errore del collegamento ISP, il traffico viene reindirizzato all'altro collegamento.

Topologia



Configurazione

Il traffico tra i siti 192.168.1.0/24 e 192.168.2.0/24 è protetto. Open Shortest Path First (OSPF) viene utilizzato nel cloud Internet per pubblicizzare gli indirizzi 10.0.0.0/8, che vengono trattati

come indirizzi pubblici assegnati dall'ISP al cliente. Nel mondo reale, viene utilizzato BGP al posto di OSPF.

La configurazione su R2 e R6 è basata sulla mappa crittografica. In R2, PBR viene utilizzato su E0/0 per indirizzare il traffico VPN su R4 se è attivo:

```
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
```

Qui vedete che il PBR locale non è necessario. L'interfaccia PBR instrada il traffico interessante alla versione 10.0.2.4. Ciò attiva il codice crittografico per avviare ISAKMP dall'interfaccia corretta (collegamento a R4), anche quando il routing è diretto ai peer remoti tramite R3.

In R6 vengono utilizzati due peer per 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 utilizza un contratto di servizio (SLA, Service Level Agreement) IP per eseguire il ping tra R3 e R4. La route predefinita è R3. In caso di errore di R3, viene scelto 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 consente inoltre l'accesso a Internet a tutti gli utenti interni. Per ottenere la ridondanza nel caso in cui l'ISP per R3 sia inattivo, è necessario un percorso-mappa. I PAT (Port Address Translations) all'interno del traffico verso un'interfaccia in uscita diversa (PAT verso l'interfaccia E0/1 quando R3 è attivo e il percorso predefinito punta a R3 e PAT verso l'interfaccia E0/2 quando R3 è inattivo e R4 è utilizzato come percorso predefinito).

```
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
```

Il traffico VPN deve essere escluso dalla traduzione come ISAKMP. Se il traffico ISAKMP non è escluso dalla traduzione, viene indirizzato all'interfaccia esterna che va verso 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

Con questa configurazione, è presente una ridondanza completa. La VPN utilizza il collegamento R4 e il resto del traffico viene instradato con R3. In caso di errore R4, il traffico VPN viene stabilito con il collegamento R3 (la route-map per PBR non corrisponde e viene utilizzato il routing predefinito).

Prima che l'ISP per R4 non sia attivo, R6 rileva il traffico dal peer 10.0.2.2:

```
R6#show crypto session
Crypto session current status

Interface: Ethernet0/0
Session status: UP-ACTIVE
Peer: 10.0.2.2 port 500
 IKEv1 SA: local 10.0.4.6/500 remote 10.0.2.2/500 Active
 IPSEC FLOW: permit ip 192.168.2.0/255.255.255.0 192.168.1.0/255.255.255.0
       Active SAs: 2, origin: crypto map
```

Dopo che R2 utilizza ISP per R3 per il traffico VPN, R6 rileva il traffico dal peer 10.0.1.2:

```
R6#show crypto session
Crypto session current status

Interface: Ethernet0/0
Session status: UP-ACTIVE
Peer: 10.0.1.2 port 500
 IKEv1 SA: local 10.0.4.6/500 remote 10.0.1.2/500 Active
 IPSEC FLOW: permit ip 192.168.2.0/255.255.255.0 192.168.1.0/255.255.255.0
       Active SAs: 2, origin: crypto map
```

Nello scenario opposto, quando il collegamento a R3 si interrompe, tutto funziona ancora bene. Il traffico VPN utilizza ancora il collegamento a R4. Network Address Translation (NAT) viene eseguito per il percorso 192.168.1.0/24 a PAT per l'impostazione dell'indirizzo esterno. Prima che R3 scenda, c'è una conversione 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
```

Dopo che R3 è sceso, c'è ancora la vecchia traduzione insieme alla nuova traduzione (a 10.0.2.2) che utilizza il collegamento verso 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
```

insidie

Se tutto funziona bene, dove sono le trappole? Sono nei dettagli.

Traffico generato localmente

Di seguito è riportato uno scenario in cui è necessario avviare il traffico VPN dalla stessa R2. Per questo scenario è necessario configurare il PBR locale su R2 in modo da forzare R2 a inviare il traffico ISAKMP tramite R4 e da far salire il tunnel. Tuttavia, l'interfaccia in uscita viene determinata usando delle tabelle di routing, il cui valore predefinito è R3, e il pacchetto viene inviato a R3, anziché a R4, che viene usato per il transito sulla VPN. Per verificarlo, immettere:

```
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
```

Nell'esempio, il protocollo ICMP (Internet Control Message Protocol) generato localmente viene forzato tramite R4. In caso contrario, il traffico generato localmente da 192.168.1.2 a 192.168.2.5 viene elaborato con l'uso di tabelle di routing e viene stabilito un tunnel con R3.

Cosa succede dopo aver applicato la configurazione? Il pacchetto ICMP da 192.168.1.2 a 192.168.2.5 viene indirizzato verso R4. Viene quindi avviato un tunnel con il collegamento a R4. Il tunnel è configurato:

```
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
```

Sembra che tutto funzioni correttamente. Il traffico viene inviato con il collegamento corretto E0/2 verso R4. Anche R6 mostra che il traffico viene ricevuto da 10.2.2.2, che è l'indirizzo IP del collegamento di 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
```

Ma in realtà, c'è un routing asimmetrico per i pacchetti ESP qui. I pacchetti ESP vengono inviati con 10.0.2.2 come origine, ma vengono inseriti sul collegamento verso R3. Viene restituita una risposta crittografata tramite R4. È possibile verificare questa condizione controllando i contatori su R3 e R4:

Contatori R3 di E0/0 prima di inviare 100 pacchetti:

```
R3#show int e0/0 | i pack
 5 minute input rate 0 bits/sec, 0 packets/sec
 5 minute output rate 0 bits/sec, 0 packets/sec
     739 packets input, 145041 bytes, 0 no buffer
```

```
    0 input packets with dribble condition detected
     1918 packets output, 243709 bytes, 0 underruns
```

E gli stessi contatori, dopo aver inviato 100 pacchetti:

```
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
```

Il numero di pacchetti in arrivo è aumentato di 100 (sul collegamento verso R2), ma i pacchetti in uscita sono aumentati solo di 2. Quindi R3 vede solo l'eco ICMP criptato.

La risposta viene verificata su R4, prima di inviare 100 pacchetti:

```
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
```

Dopo l'invio di 100 pacchetti:

```
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
```

Il numero di pacchetti inviati alla R2 è aumentato di 102 (risposta ICMP crittografata), mentre i pacchetti ricevuti sono aumentati di 0. Pertanto, R4 vede solo la risposta ICMP crittografata. Naturalmente, l'acquisizione di un pacchetto conferma questa condizione.

Perché questo accade? La risposta è nella prima parte dell'articolo.

Di seguito viene riportato il flusso di tali pacchetti ICMP:

1. ICMP da 192.168.1.2 a 192.168.2.6 viene inserito su E0/2 (collegamento verso R4) a causa del PBR locale.
2. La sessione ISAKMP è realizzata con la versione 10.0.2.2 e inserita sul collegamento E0/2 come previsto.
3. Per i pacchetti ICMP dopo l'incapsulamento, il router deve determinare l'interfaccia in uscita, usando le tabelle di routing che puntano a R3. Per questo motivo, il pacchetto crittografato con origine 10.0.2.2 (collegamento a R4) viene inviato tramite R3.
4. R6 riceve un pacchetto ESP da 10.0.2.2, che è coerente con la sessione ISAKMP, lo decrittografa e invia la risposta ESP a 10.0.2.2.
5. A causa del routing, R5 restituisce una risposta a 10.0.2.2 tramite R4.
6. R2 lo riceve e lo decripta e il pacchetto viene accettato.

Ecco perché è importante essere particolarmente cauti con il traffico generato localmente.

In molte reti viene utilizzato l'uRPF (Unicast Reverse Path Forwarding) e il traffico proveniente da

10.0.2.2 potrebbe essere interrotto sulla porta E0/0 di R3. In questo caso, il ping non funziona.

Esiste una soluzione per questo problema? È possibile forzare il router a trattare il traffico generato localmente come traffico di transito. Per questo motivo, il PBR locale deve indirizzare il traffico verso un'interfaccia loopback fittizia da cui viene indirizzato come traffico di transito.

Questa operazione non è consigliata.

> **Nota:** È importante prestare particolare attenzione quando si utilizza NAT insieme a PBR (fare riferimento alla sezione precedente sul traffico ISKMP nell'elenco degli accessi PAT).

Esempio di configurazione senza PBR

C'è anche un'altra soluzione che è un compromesso. Con la stessa topologia dell'esempio precedente, è possibile soddisfare tutti i requisiti senza utilizzare il PBR o il PBR locale. Per questo scenario, viene utilizzato solo il routing. In R2 viene aggiunta un'unica voce di routing e vengono rimosse tutte le configurazioni PBR/PBR locale:

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

In totale, R2 presenta la seguente configurazione di routing:

```
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 prima voce corrisponde a un percorso predefinito verso R3, quando il collegamento a R3 è attivo. La seconda voce di routing è un percorso di backup predefinito verso R4, quando il collegamento a R3 è inattivo. La terza voce decide come inviare il traffico alla rete VPN remota, a seconda dello stato del collegamento R4 (se il collegamento R4 è attivo, il traffico alla rete VPN remota viene inviato tramite R4). Con questa configurazione, non è necessario eseguire il routing delle policy.

Qual è lo svantaggio? Non esiste più alcun controllo granulare che utilizzi il PBR. Impossibile determinare l'indirizzo di origine. In questo caso, tutto il traffico diretto a 192.168.2.0/24 viene inviato in direzione R4 quando è attivo, a prescindere dall'origine. Nell'esempio precedente, controllato dal PBR e dall'origine specifica: 192.168.1.0/24.

A quale scenario si riferisce questa soluzione troppo semplice? Per reti LAN multiple (dietro R2). Quando alcune di queste reti devono raggiungere il sito 192.168.2.0/24 in modo sicuro (crittografato) e altri modi non sicuri (non crittografato), il traffico proveniente da reti non sicure viene ancora inserito nell'interfaccia E0/2 di R2 e non colpisce la mappa crittografica. Quindi viene inviato non crittografato tramite un collegamento a R4 (e il requisito principale era usare R4 solo per il traffico crittografato).

Questo tipo di scenario e i suoi requisiti sono rari, motivo per cui questa soluzione viene utilizzata abbastanza spesso.

Riepilogo

L'uso del PBR e delle funzionalità PBR locali insieme alle VPN e al NAT può essere complesso e richiede una comprensione approfondita del flusso dei pacchetti.

In scenari come quelli illustrati di seguito, si consiglia di utilizzare due router separati, ciascuno con un collegamento ISP. In caso di guasto di un ISP, il traffico può essere reindirizzato facilmente. Non c'è bisogno di PBR, e il design generale è molto più semplice.

Esiste inoltre una soluzione di compromesso che non richiede l'utilizzo del PBR, ma utilizza invece il routing mobile statico.

Verifica

Attualmente non è disponibile una procedura di verifica per questa configurazione.

Risoluzione dei problemi

Al momento non sono disponibili informazioni specifiche per la risoluzione dei problemi di questa configurazione.

Informazioni correlate

- Documentazione e supporto tecnico – Cisco Systems
- Cisco IOS 15.3 M e T - Cisco Systems