Richtlinien-Routing und seine Auswirkungen auf ESP- und ISAKMP-Pakete mit Cisco IOS

Inhalt

Einführung

In diesem Dokument werden die Auswirkungen von richtlinienbasiertem Routing (PBR) und lokalem PBR bei der Anwendung auf Pakete mit Encapsulating Security Payload (ESP) und Internet Security Association and Key Management Protocol (ISAKMP) bei der Verwendung von Cisco IOS® beschrieben.

Unterstützt von Michal Garcarz, Cisco TAC Engineer.

Voraussetzungen

Anforderungen

Cisco empfiehlt, über grundlegende Kenntnisse in folgenden Bereichen zu verfügen:

- Cisco IOS
- VPN-Konfiguration in Cisco IOS

Verwendete Komponenten

Die Informationen in diesem Dokument basieren auf Cisco IOS Version 15.x.

Die Informationen in diesem Dokument wurden von den Geräten in einer bestimmten Laborumgebung erstellt. Alle in diesem Dokument verwendeten Geräte haben mit einer leeren (Standard-)Konfiguration begonnen. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die potenziellen Auswirkungen eines Befehls verstehen.

Hintergrundinformationen

Vor der Einrichtung eines IPsec-Tunnels initiiert der Router einen ISAKMP-Austausch. Da diese Pakete vom Router generiert werden, werden die Pakete als lokal generierter Datenverkehr behandelt und alle lokalen PBR-Entscheidungen werden angewendet. Darüber hinaus werden alle vom Router generierten Pakete (EIGRP (Enhanced Interior Gateway Routing Protocol), NHRP (Next Hop Resolution Protocol), Border Gateway Protocol (BGP) oder ICMP-Pings (Internet Control Message Protocol) ebenfalls als lokal generierter Datenverkehr angesehen und unterliegen der lokalen PBR-Entscheidung.

Der vom Router weitergeleitete und über den Tunnel gesendete Datenverkehr, der so genannte Transit-Datenverkehr, wird nicht als lokal generierter Datenverkehr angesehen, und alle gewünschten Routing-Richtlinien müssen auf die Eingangs-Schnittstelle des Routers angewendet werden.

Dies wirkt sich auf den Datenverkehr aus, der den Tunnel passiert, indem der lokal generierte Datenverkehr PBR folgt, der Transitverkehr jedoch nicht. In diesem Artikel werden die Folgen dieser unterschiedlichen Verhaltensweise erläutert.

Für den Transitverkehr, der ESP-gekapselt werden muss, sind keine Routing-Einträge erforderlich, da der PBR die Ausgangsschnittstelle für das Paket vor und nach der ESP-Kapselung bestimmt. Für lokal generierten Datenverkehr, der ESP-gekapselt werden muss, müssen Routing-Einträge vorhanden sein, da das lokale PBR die Ausgangsschnittstelle nur für das Paket vor der Kapselung bestimmt und das Routing die Ausgangsschnittstelle für das nachgekapselte Paket bestimmt.

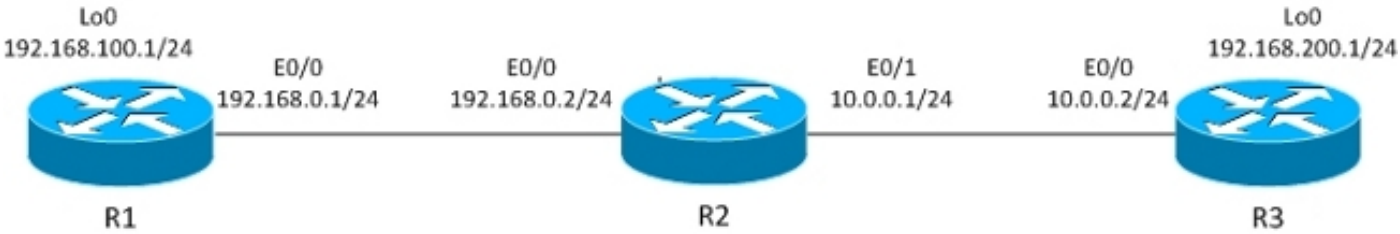
Dieses Dokument enthält ein typisches Konfigurationsbeispiel, in dem ein Router mit zwei ISP-Links verwendet wird. Eine Verbindung wird für den Zugriff auf das Internet verwendet, die andere für VPN. Bei einem Verbindungsausfall wird der Datenverkehr über eine andere Internetdienstanbieter-Verbindung umgeleitet. Es werden auch Pitfälle dargestellt.

Bitte beachten Sie, dass PBR in Cisco Express Forwarding (CEF) ausgeführt wird, während der lokale PBR prozessgeschaltet ist.

Lokaler Datenverkehr auf dem Router

In diesem Abschnitt wird das Verhalten des vom Router (R)1 initiierten Datenverkehrs beschrieben. Dieser Datenverkehr ist durch R1 gekapselt ESP.

Topologie



Der IPsec LAN-to-LAN-Tunnel ist zwischen R1 und R3 aufgebaut.

Der interessante Datenverkehr verläuft zwischen R1 Lo0 (192.168.100.1) und R3 Lo0 (192.168.200.1).

Der R3-Router hat eine Standardroute zu R2.

R1 hat keine Routingeinträge, sondern nur direkt verbundene Netzwerke.

Konfiguration

R1 verfügt über lokalen PBR für den gesamten Datenverkehr:

```
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
```

Debugger

Der gesamte lokal generierte Datenverkehr auf R1 wird bei UP an R2 gesendet.

Um zu überprüfen, was beim Hochfahren des Tunnels geschieht, senden Sie den interessanten Datenverkehr vom Router selbst:

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

> Vorsicht: Der Befehl **debug ip packet** kann eine große Menge an Debuggen generieren und hat enorme Auswirkungen auf die CPU-Nutzung. Verwenden Sie es mit Vorsicht.

Bei diesem Debuggen kann die Zugriffsliste auch verwendet werden, um die Menge des von Debuggen verarbeiteten Datenverkehrs zu begrenzen. Der Befehl **debug ip packet** zeigt nur den prozessgesteuerten Datenverkehr an.

Hier sind die Debug auf 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
```

Folgendes passiert:

Der interessante Datenverkehr (192.168.100.1 > 192.168.200.1) wird mit dem lokalen PBR abgeglichen, und die Ausgangsschnittstelle wird bestimmt (E0/0). Diese Aktion löst den Kryptocode aus, um ISAKMP zu initiieren. Dieses Paket wird auch per Richtlinie über den lokalen PBR geroutet, der die Ausgangsschnittstelle (E0/0) bestimmt. Der ISAKMP-Datenverkehr wird gesendet, und der Tunnel wird ausgehandelt.

Was passiert, wenn Sie erneut pingen?

```
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)
```

Folgendes passiert:

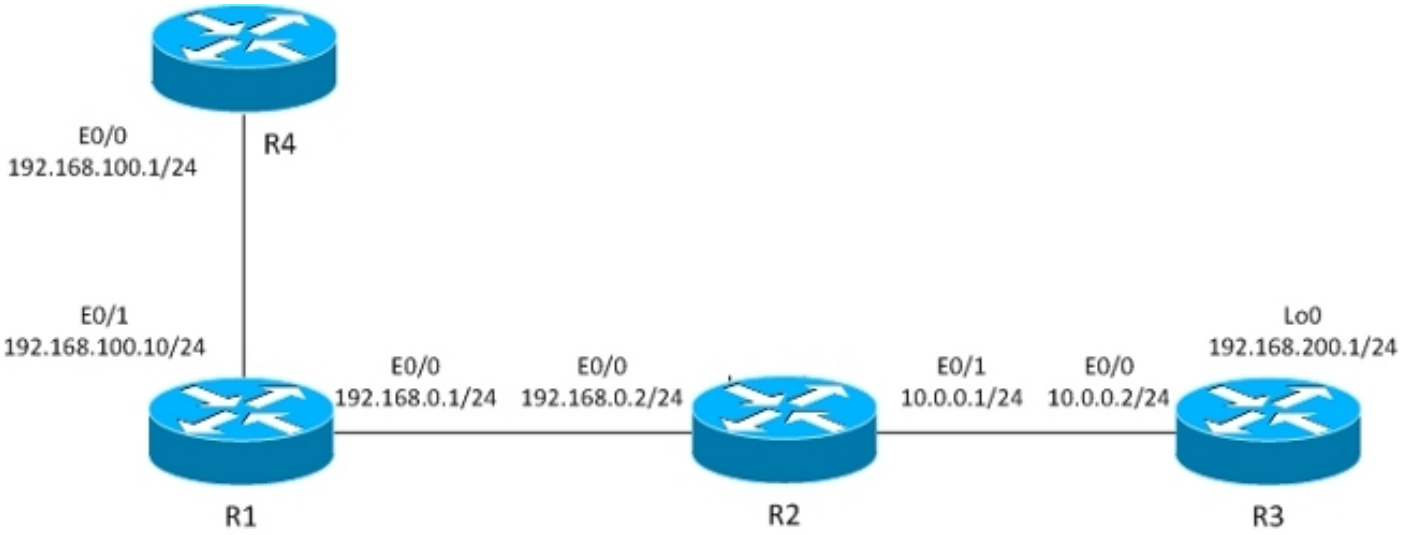
Der lokal generierte interessante Datenverkehr, 192.168.100.1 > 192.168.200.1, wird lokal richtliniengeroutet, und die Ausgangsschnittstelle wird bestimmt (E0/0). Das Paket wird von der IPsec-Ausgabefunktion auf E0/0 verbraucht und gekapselt. Das gekapselte Paket (von 192.168.0.1 bis 10.0.0.2) wird auf Routing überprüft, um die Ausgangsschnittstelle zu bestimmen. Die Routing-Tabellen von R1 enthalten jedoch nichts, weshalb die Kapselung fehlschlägt.

In diesem Szenario ist der Tunnel eingeschaltet, aber der Datenverkehr wird nicht gesendet, da Cisco IOS nach der ESP-Kapselung die Routing-Tabellen überprüft, um die Ausgangsschnittstelle zu ermitteln.

Datenverkehr über den Router leiten

In diesem Abschnitt wird das Verhalten für den Transit-Datenverkehr beschrieben, der über den Router geleitet wird, der von diesem Router gekapselt wird.

Topologie



Der L2L-Tunnel ist zwischen R1 und R3 aufgebaut.

Der interessante Datenverkehr verläuft zwischen R4 (192.168.100.1) und R3 lo0 (192.168.200.1).

Der R3-Router hat eine Standardroute zu R2.

Der R4-Router hat eine Standardroute zu R1.

R1 hat kein Routing.

Konfiguration

Die vorherige Topologie wird geändert, um den Fluss anzuzeigen, wenn der Router Pakete zur Verschlüsselung empfängt (Transit-Datenverkehr statt lokal generierten Datenverkehr).

Derzeit wird der interessante von R4 empfangene Datenverkehr auf R1 (durch PBR auf E0/1) weitergeleitet, und für den gesamten Datenverkehr gibt es auch ein lokales Richtlinienrouting:

```
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
```

Debugger

Geben Sie Folgendes ein, um zu überprüfen, was beim Hochfahren des Tunnels auf R1 geschieht (nachdem Sie den interessanten Datenverkehr von R4 empfangen haben):

```
R1#debug ip packet
```

```
R4#ping 192.168.200.1
```

Hier sind die Debug auf 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
```

Folgendes passiert:

Der interessante Datenverkehr trifft PBR auf E0/0 und löst Kryptocode aus, um das ISAKMP-Paket zu senden. Dieses ISAKMP-Paket wird lokal durch Richtlinien geroutet, und die Ausgangsschnittstelle wird durch den lokalen PBR bestimmt. Es wird ein Tunnel gebaut.

Hier ist ein weiterer Ping von R4 zu 192.168.200.1:

```
R4#ping 192.168.200.1
```

Hier sind die Debug auf 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
```

Folgendes passiert:

Der interessante Datenverkehr trifft auf PBR auf E0/0, und der PBR bestimmt die Ausgangsschnittstelle (E0/0). Auf E0/0 wird das Paket von IPSec verbraucht und gekapselt. Nachdem das gekapselte Paket mit derselben PBR-Regel verglichen und die Ausgangsschnittstelle bestimmt wurde, wird das Paket korrekt gesendet und empfangen.

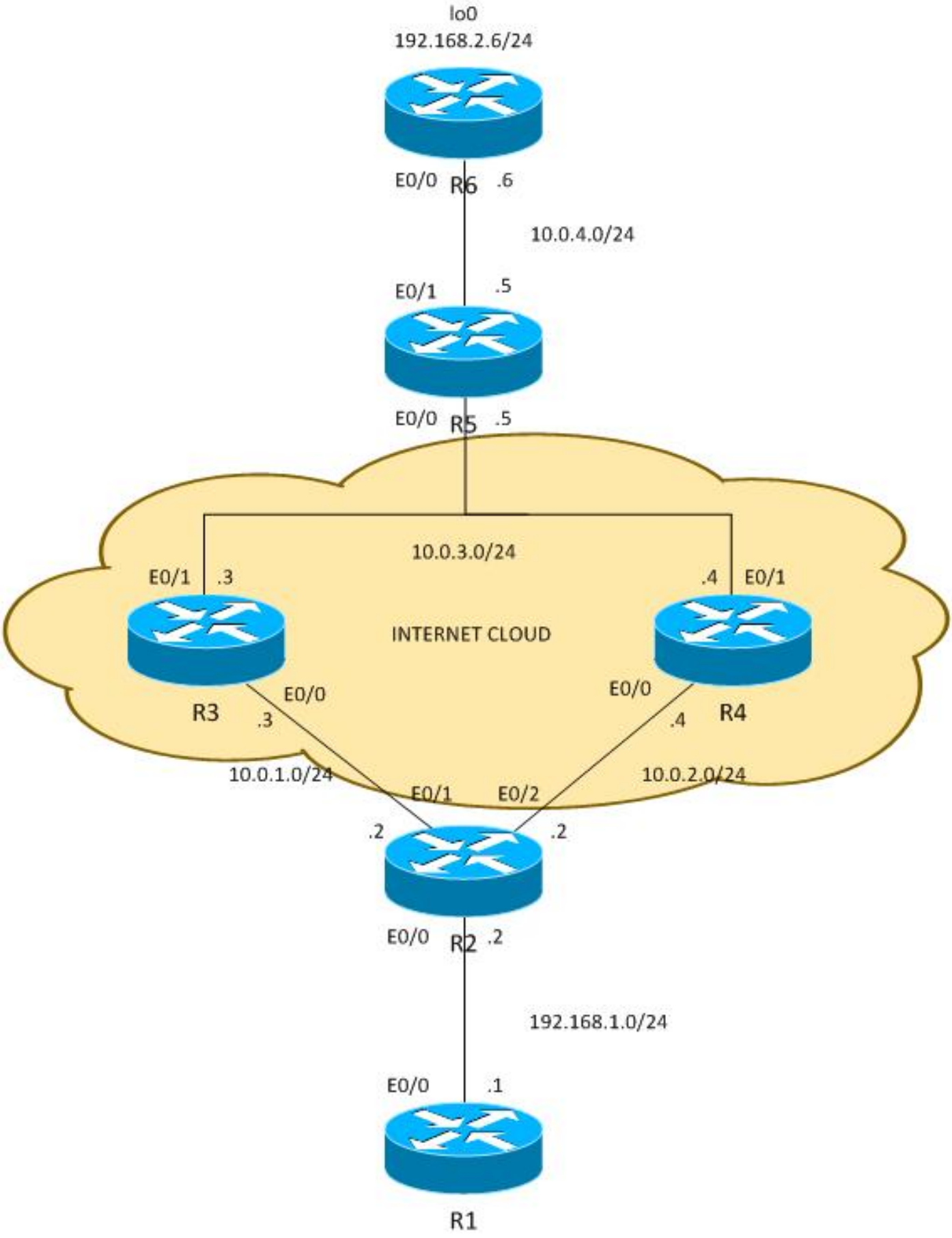
Zusammenfassung zu Verhaltensunterschieden

Für lokal generierten Datenverkehr wird die Ausgangsschnittstelle für nicht gekapselten Datenverkehr (ISAKMP) durch den lokalen PBR bestimmt. Für lokal generierten Datenverkehr wird die Ausgangsschnittstelle für Post-Encapsulated Traffic (ESP) durch die Routing-Tabellen bestimmt (der lokale PBR ist nicht aktiviert). Für den Transitverkehr wird die Egress-Schnittstelle für Postverkapselten Datenverkehr (ESP) durch die Schnittstelle PBR bestimmt (zweimal vor und nach der Kapselung).

Beispielkonfiguration

Dies ist ein praktisches Konfigurationsbeispiel, in dem die Probleme erläutert werden, denen Sie möglicherweise mit dem PBR und dem lokalen PBR mit VPN konfrontiert sind. Der R2 (CE) hat zwei ISP-Verbindungen. Der R6-Router verfügt außerdem über einen CE- und einen ISP-Link. Die erste Verbindung von R2 zu R3 wird als Standardroute für R2 verwendet. Die zweite Verbindung zu R4 wird nur für den VPN-Datenverkehr zu R6 verwendet. Bei einem Ausfall einer ISP-Verbindung wird der Datenverkehr zur anderen Verbindung umgeleitet.

Topologie



Konfiguration

Der Datenverkehr zwischen 192.168.1.0/24 und 192.168.2.0/24 ist geschützt. Open Shortest Path First (OSPF) wird in der Internet-Cloud verwendet, um die 10.0.0.0/8-Adressen anzugeben, die als

öffentliche Adressen behandelt werden, die dem Kunden vom ISP zugewiesen wurden. In der Praxis wird BGP anstelle von OSPF verwendet.

Die Konfiguration auf R2 und R6 basiert auf der Crypto-Map. Auf R2 wird PBR auf E0/0 verwendet, um VPN-Datenverkehr an R4 weiterzuleiten, wenn 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
```

Hier sehen Sie, dass kein lokaler PBR erforderlich ist. Die Schnittstelle PBR leitet interessanten Datenverkehr an 10.0.2.4 weiter. Dies löst den Kryptocode aus, um ISAKMP von der richtigen Schnittstelle (Verbindung zu R4) zu initiieren, selbst wenn das Routing zu Remote-Peer-Points über R3 erfolgt.

Auf R6 werden zwei Peers für das VPN verwendet:

```
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 verwendet ein IP Service Level Agreement (SLA), um R3 und R4 zu pingen. Die Standardroute ist R3. Bei Ausfall von R3 wählt er 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
```

Auch R2 ermöglicht Internetzugang für alle internen Benutzer. Um Redundanz bei einem Ausfall des ISP zu R3 zu erreichen, ist eine Route Map erforderlich. Es Port Address Translations (PATs) im Datenverkehr zu einer anderen Ausgangsschnittstelle (PAT zu E0/1-Schnittstelle, wenn R3 UP ist und die Standardroute zu R3 und PAT zur Schnittstelle E0/2, wenn R3 ausgefallen ist und R4 als Standardroute verwendet wird).

```
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
```

Der VPN-Datenverkehr muss ebenso wie ISAKMP von der Übersetzung ausgeschlossen werden. Wenn ISAKMP-Datenverkehr nicht von der Übersetzung ausgeschlossen wird, wird er an die externe Schnittstelle weitergeleitet, die zu R3 führt:

```
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
```

Tests

Diese Konfiguration bietet eine vollständige Redundanz. Das VPN verwendet die R4-Verbindung, und der restliche Datenverkehr wird mit R3 geroutet. Bei einem Ausfall des R4 wird der VPN-Datenverkehr über die R3-Verbindung eingerichtet (die Routing-Map für den PBR stimmt nicht überein, und das Standard-Routing wird verwendet).

Bevor der ISP zu R4 ausfällt, erkennt R6 den Datenverkehr von 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
```

Nachdem R2 ISP zu R3 für VPN-Datenverkehr verwendet hat, erfasst R6 den Datenverkehr von 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
```

Im Gegenteil: Wenn die Verbindung zu R3 ausfällt, funktioniert alles noch gut. Der VPN-Datenverkehr verwendet weiterhin die Verbindung zu R4. Network Address Translation (NAT) wird für 192.168.1.0/24 in PAT ausgeführt, um die externe Adresse anzugeben. Bevor R3 ausfällt, gibt es eine Übersetzung nach 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
```

Nach dem Ausfall von R3 gibt es noch die alte Übersetzung und die neue Übersetzung (zu

10.0.2.2), die den Link zu R4 verwendet:

```
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
```

Stiche

Wenn alles gut funktioniert, wo sind die Fallstricke? Sie sind im Detail.

Lokal generierter Datenverkehr

In diesem Szenario muss der VPN-Datenverkehr vom R2 selbst initiiert werden. In diesem Szenario müssen Sie das lokale PBR auf R2 konfigurieren, um R2 dazu zu zwingen, ISAKMP-Datenverkehr über R4 zu senden und den Tunnel hochzufahren. Die Ausgangsschnittstelle wird jedoch mithilfe von Routing-Tabellen bestimmt, wobei der Standardwert auf R3 verweist, und dieses Paket wird an R3 anstatt an R4 gesendet, der für die Übertragung von VPNs verwendet wird. Geben Sie Folgendes ein, um dies zu überprüfen:

```
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
```

In diesem Beispiel wird das lokal generierte Internet Control Message Protocol (ICMP) über R4 erzwungen. Andernfalls wird der lokal von 192.168.1.2 bis 192.168.2.5 generierte Datenverkehr mithilfe von Routing-Tabellen verarbeitet, und mit R3 wird ein Tunnel eingerichtet.

Was passiert, nachdem Sie diese Konfiguration angewendet haben? Das ICMP-Paket von 192.168.1.2 bis 192.168.2.5 wird in Richtung R4 verschoben, und es wird ein Tunnel mit der Verbindung zu R4 initiiert. Der Tunnel ist eingerichtet:

```
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
```

Alles scheint richtig zu funktionieren. Der Datenverkehr wird mit der richtigen Verbindung E0/2 in Richtung R4 gesendet. Selbst R6 zeigt, dass der Datenverkehr von 10.2.2.2 empfangen wird, d. h. von der Link-IP-Adresse von 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
```

Hier gibt es jedoch **asymmetrisches Routing für ESP-Pakete**. ESP-Pakete werden mit 10.0.2.2 als Quelle gesendet, werden jedoch auf die Verbindung zu R3 gesetzt. Eine verschlüsselte Antwort wird über R4 zurückgegeben. Dies kann durch Überprüfen der Zähler auf R3 und R4 überprüft werden:

R3-Zähler von E0/0 vor dem Senden von 100 Paketen:

```
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
```

Und die gleichen Zähler, nach dem Senden von 100 Paketen:

```
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
```

Die Anzahl der eingehenden Pakete stieg um 100 (auf der Verbindung zu R2), aber die ausgehenden Pakete stiegen nur um 2. R3 sieht nur das verschlüsselte ICMP-Echo.

Die Antwort wird auf R4 angezeigt, bevor 100 Pakete gesendet werden:

```
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
```

Nach dem Senden von 100 Paketen:

```
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
```

Die Anzahl der an R2 gesendeten Pakete stieg um 102 (verschlüsselte ICMP-Antwort), während die empfangenen Pakete um 0 erhöht wurden. R4 sieht also nur die verschlüsselte ICMP-Antwort. Dies wird natürlich durch eine Paketerfassung bestätigt.

Warum geschieht das? Die Antwort ist im ersten Teil des Artikels.

Der Fluss dieser ICMP-Pakete folgt:

1. Der ICMP von 192.168.1.2 bis 192.168.2.6 wird aufgrund des lokalen PBR auf E0/2 (Verbindung zu R4) gesetzt.
2. Die ISAKMP-Sitzung wurde mit 10.0.2.2 erstellt und wie erwartet auf die E0/2-Verbindung gesetzt.
3. Bei ICMP-Paketen nach der Kapselung muss der Router die Ausgangsschnittstelle bestimmen. Dies geschieht mithilfe von Routing-Tabellen, die auf R3 verweisen. Aus diesem Grund wird das verschlüsselte Paket mit Quelle 10.0.2.2 (Link zu R4) über R3 gesendet.
4. R6 empfängt ein ESP-Paket von 10.0.2.2, das der ISAKMP-Sitzung entspricht, entschlüsselt das Paket und sendet die ESP-Antwort an 10.0.2.2.
5. Aufgrund des Routings sendet R5 eine Antwort an 10.0.2.2 bis R4 zurück.
6. R2 empfängt und entschlüsselt, und das Paket wird akzeptiert.

Daher ist es wichtig, bei lokal generiertem Datenverkehr besonders vorsichtig zu sein.

In vielen Netzwerken wird Unicast Reverse Path Forwarding (uRPF) verwendet, und der von 10.0.2.2 ausgehende Datenverkehr könnte auf E0/0 von R3 verworfen werden. In diesem Fall funktioniert der Ping nicht.

Gibt es eine Lösung für dieses Problem? Es ist möglich, den Router zu zwingen, lokal generierten Datenverkehr als Transitverkehr zu behandeln. Dazu muss der lokale PBR den Datenverkehr an eine falsche Loopback-Schnittstelle weiterleiten, von der er wie der Transitverkehr weitergeleitet wird.

Dies wird nicht empfohlen.

> **Hinweis**: Es ist wichtig, besonders vorsichtig zu sein, wenn Sie NAT zusammen mit PBR verwenden (siehe vorherigen Abschnitt über ISKMP-Datenverkehr in der PAT-Zugriffsliste).

Beispielkonfiguration ohne PBR

Es gibt auch eine andere Lösung, die ein Kompromiss ist. Mit der gleichen Topologie wie im vorherigen Beispiel können alle Anforderungen erfüllt werden, ohne dass ein PBR oder ein lokaler PBR verwendet werden muss. In diesem Szenario wird nur das Routing verwendet. Nur ein weiterer Routing-Eintrag wird für R2 hinzugefügt, und alle PBR/lokalen PBR-Konfigurationen werden entfernt:

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

Insgesamt verfügt R2 über folgende Routing-Konfiguration:

```
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
```

Der erste Routing-Eintrag ist ein Standard-Routing zu R3, wenn die Verbindung zu R3 UP ist. Beim zweiten Routing-Eintrag handelt es sich um eine Backup-Standardroute zu R4, wenn die Verbindung zu R3 unterbrochen ist. Der dritte Eintrag entscheidet, wie Datenverkehr an das Remote-VPN-Netzwerk gesendet wird, abhängig vom R4-Verbindungsstatus (wenn die R4-Verbindung aktiv ist, wird der Datenverkehr über R4 an das Remote-VPN-Netzwerk gesendet). Bei dieser Konfiguration ist kein Richtlinien-Routing erforderlich.

Was ist der Nachteil? Eine präzise Kontrolle über PBR ist nicht mehr gegeben. Es ist nicht möglich, die Quelladresse zu bestimmen. In diesem Fall wird der gesamte Datenverkehr an 192.168.2.0/24 unabhängig von der Quelle an R4 gesendet, wenn er aktiv ist. Im vorherigen Beispiel wurde diese vom PBR und der spezifischen Quelle gesteuert: 192.168.1.0/24 ist ausgewählt.

Für welches Szenario ist diese Lösung zu einfach? Für mehrere LAN-Netzwerke (hinter R2). Wenn einige dieser Netzwerke auf sichere (verschlüsselte) und andere unsichere (unverschlüsselte) Weise 192.168.2.0/24 erreichen müssen, wird der Datenverkehr aus unsicheren Netzwerken weiterhin auf die E0/2-Schnittstelle von R2 gesetzt und nicht auf die Crypto-Map geladen. So wird sie unverschlüsselt über eine Verbindung zu R4 gesendet (und die

Hauptanforderung war, R4 nur für verschlüsselten Datenverkehr zu verwenden).

Diese Art von Szenario und seine Anforderungen sind selten, und deshalb wird diese Lösung ziemlich häufig verwendet.

Zusammenfassung

Die Verwendung von PBR- und lokalen PBR-Funktionen zusammen mit VPNs und NAT kann komplex sein und erfordert ein umfassendes Verständnis des Paketflusses.

In Szenarien wie den hier vorgestellten empfiehlt es sich, zwei separate Router zu verwenden - jeder Router mit einer ISP-Verbindung. Bei einem ISP-Ausfall kann der Datenverkehr problemlos umgeleitet werden. PBR ist nicht erforderlich, und das allgemeine Design ist viel einfacher.

Es gibt auch eine Kompromisslösung, die keine PBR-Funktion erfordert, sondern stattdessen statisches Floating-Routing verwendet.

Überprüfen

Für diese Konfiguration ist derzeit kein Überprüfungsverfahren verfügbar.

Fehlerbehebung

Für diese Konfiguration sind derzeit keine spezifischen Informationen zur Fehlerbehebung verfügbar.

Zugehörige Informationen

- Technischer Support und Dokumentation - Cisco Systems
- Cisco IOS 15.3 M&T - Cisco Systems