# DMVPN 2단계 스포크 투 스포크 터널 문제 해결

## 목차



## 소개

이 문서에서는 2단계 스포크 투 스포크 DMVPN 터널이 설정되지 않은 경우 문제를 해결하는 방법에 대해 설명합니다.

## 사전 요구 사항

### 요구 사항

다음 주제에 대해 알고 있는 것이 좋습니다.

- DMVPN(Dynamic Multipoint Virtual Private Network)
- IKE/IPSEC 프로토콜
- NHRP(Next Hop Resolution Protocol)

### 사용되는 구성 요소

이 문서는 다음 소프트웨어 버전을 기반으로 합니다.

- Cisco CSR1000V(VXE) - 버전 17.03.08

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

# 배경 정보

이 문서에서는 일반적인 DMVPN 문제에 대해 다양한 문제 해결 도구를 구성하고 사용하는 방법에 대해 설명합니다. 이 문제는 2단계 DMVPN 터널의 협상 실패입니다. 소스 스포크가 DMVPN 상태에 대상 스포크에 대한 올바른 NBMA(Non-Broadcast Multi-Access)/터널 매핑이 표시됩니다. 그러나 대상 스포크에 잘못된 매핑이 표시됩니다.

# 이론적 배경

DMVPN 2단계 설정 시 스포크 투 스포크 터널이 설정되는 방식을 이해하는 것이 중요합니다. 이 섹션에서는 이 단계의 NHRP 프로세스에 대한 간단한 이론적 요약을 제공합니다.

DMVPN 2단계에서는 필요에 따라 동적 스포크 투 스포크 터널을 구축할 수 있습니다. 이는 DMVPN 클라우드 내의 모든 디바이스(허브 및 스포크)에서 터널 인터페이스의 모드가 GRE(Generic Routing Encapsulation) 멀티포인트로 변경되기 때문에 가능합니다. 이 단계의 주요 기능 중 하나는 다른 디바이스에서 허브를 next-hop으로 인식하지 못한다는 것입니다. 대신 모든 스포크는 서로 간의 라우팅 정보를 가집니다. 2단계에서 스포크 투 스포크 터널을 설정할 때 스포크가 다른 스포크에 대한 정보를 학습하는 NHRP 프로세스가 트리거되고 NBMA와 터널 IP 주소 간에 매핑이 수행됩니다.

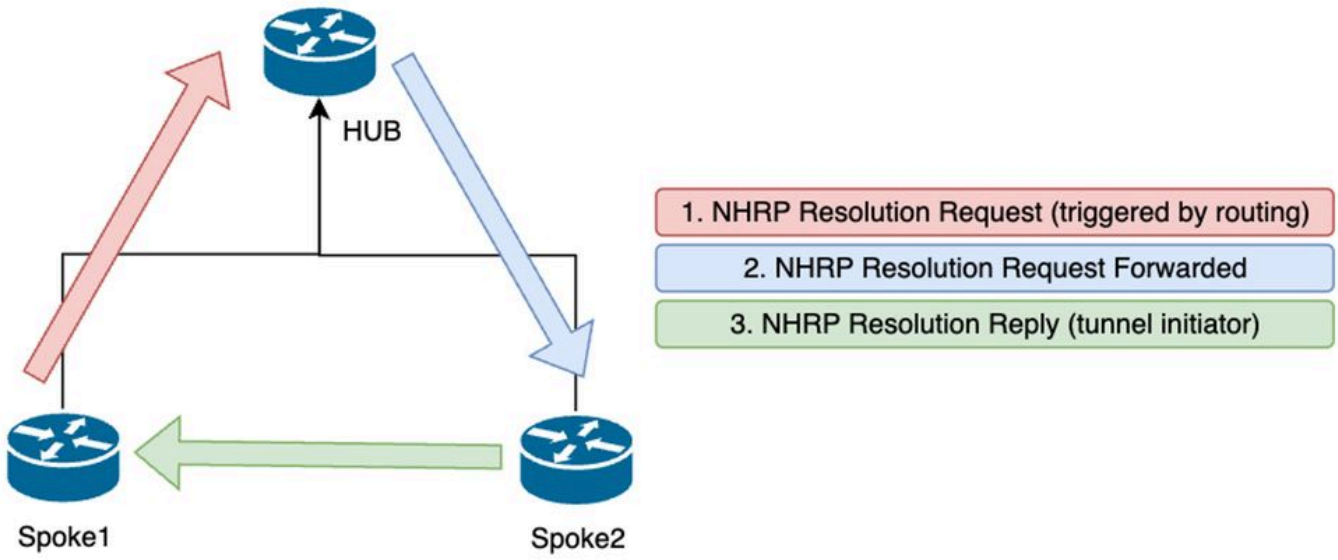
다음 단계는 NHRP 해결 프로세스가 트리거되는 방법을 나열합니다.

1. 소스 스포크가 대상 스포크의 LAN에 연결하려고 하면 해결 요청 메시지를 트리거하는 경로 조회를 수행하여 대상 스포크의 NBMA 주소를 가져옵니다. 소스 스포크가 이 초기 메시지를 허브에 전송합니다.

2. 허브는 해결 요청을 받고 대상 스포크에 전달합니다.

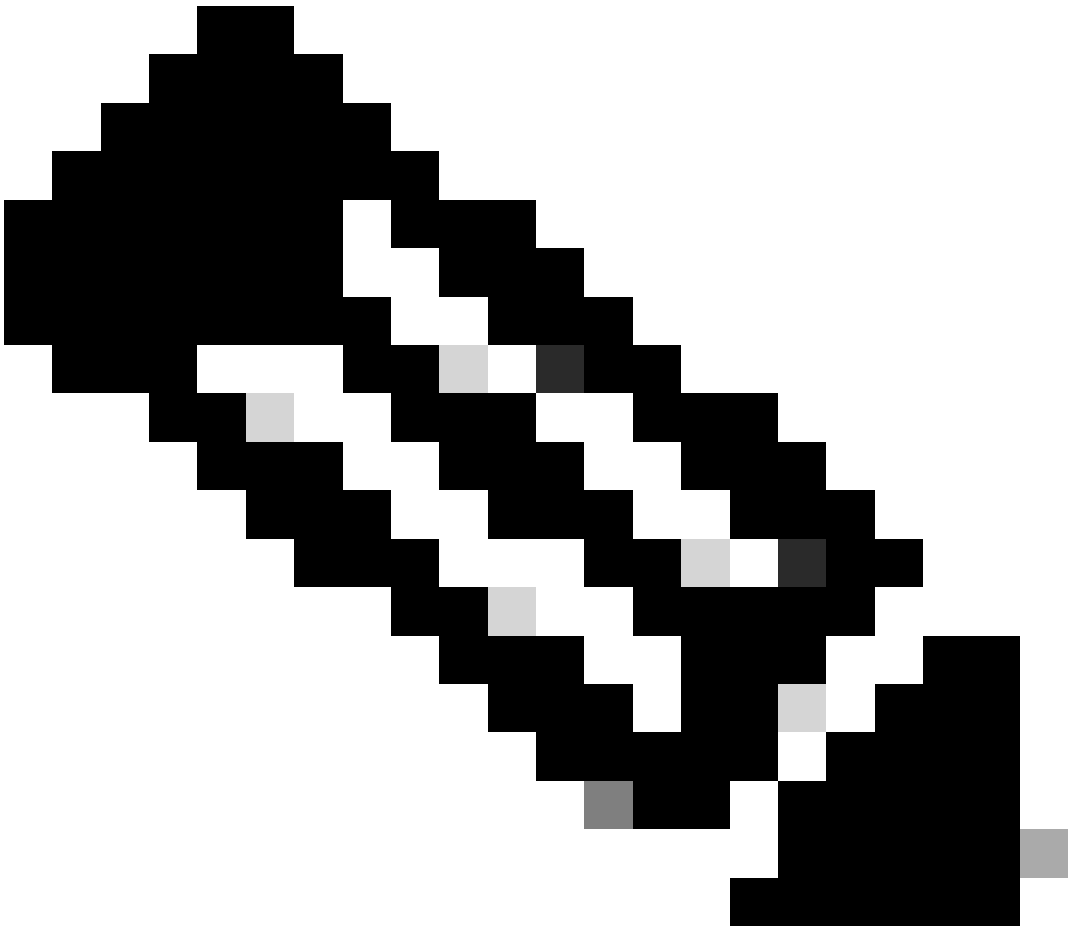
3. 대상 스포크가 원본 스포크에 해결 응답을 보냅니다. 터널 구성에 연결된 IPSEC 프로파일이 있는 경우:

    - NHRP 확인 프로세스는 IKE/IPSEC 프로토콜이 설정될 때까지 지연됩니다.
    - 대상 스포크가 IKE/IPSEC 터널을 시작하고 설정합니다.
    - 그런 다음 NHRP 프로세스가 다시 시작되고 대상 스포크가 IPSEC 터널을 전송 방법으로 사용하여 원본 스포크에 해결 응답을 보냅니다.




2단계의 스포크 간 NHRP 메시지 흐름

---



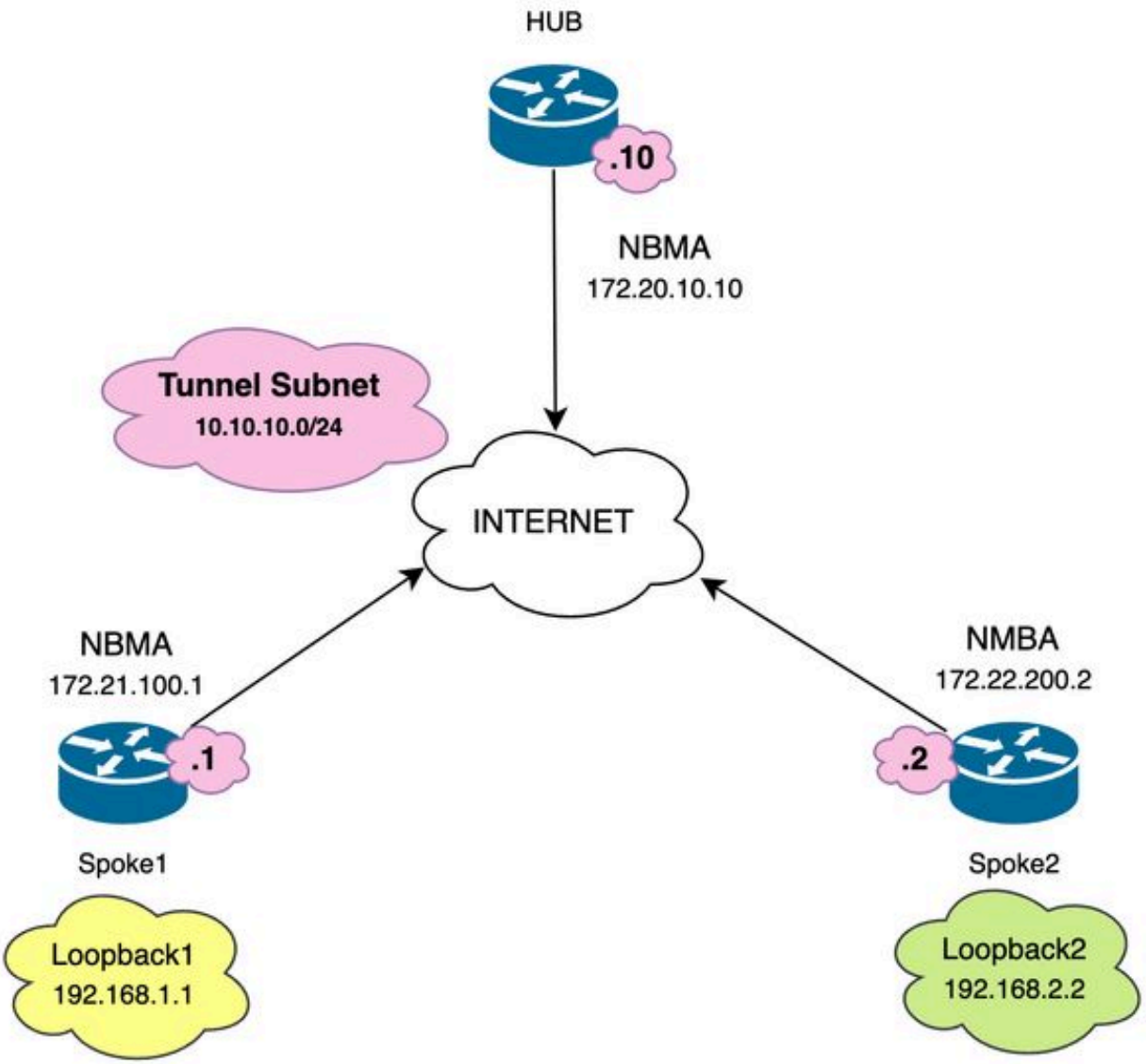
참고: 확인 프로세스를 시작하려면 먼저 모든 스포크가 HUB에 이미 등록되어 있어야 합니

---

다.

## 토폴로지

이 다이어그램은 시나리오에 사용되는 토폴로지를 보여줍니다.




사용된 네트워크 다이어그램 및 IP 서브넷

## 문제 해결 단계

이 시나리오에서는 Spoke1과 Spoke2 간의 스포크 투 스포크 터널이 설정되지 않으므로 서로 연결할 수 없으므로 로컬 리소스(루프백 인터페이스로 표시됨) 간의 통신에 영향을 미칩니다.

```
SPOKE1#ping 192.168.2.2 source loopback1
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.2.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

## 초기 검증

이러한 시나리오가 발생할 경우, 먼저 터널 컨피그레이션을 검증하고 두 디바이스 모두 올바른 값을 갖도록 해야 합니다. 터널 구성을 검토하려면 show running-config interface tunnel<ID> 명령을 실행합니다.

스포크 1 터널 구성:

<#root>

```
SPOKE1#show running-config interface tunnel10
Building configuration...

Current configuration : 341 bytes
!
interface Tunnel10
ip address 10.10.10.1 255.255.255.0
no ip redirects
ip nhrp authentication DMVPN


ip nhrp map 10.10.10.10 172.20.10.10


ip nhrp map multicast 172.20.10.10


ip nhrp network-id 10
ip nhrp nhs 10.10.10.10


tunnel source GigabitEthernet1
tunnel mode gre multipoint


tunnel protection IPSEC profile IPSEC_Profile_1


end
```

스포크 2 터널 구성:

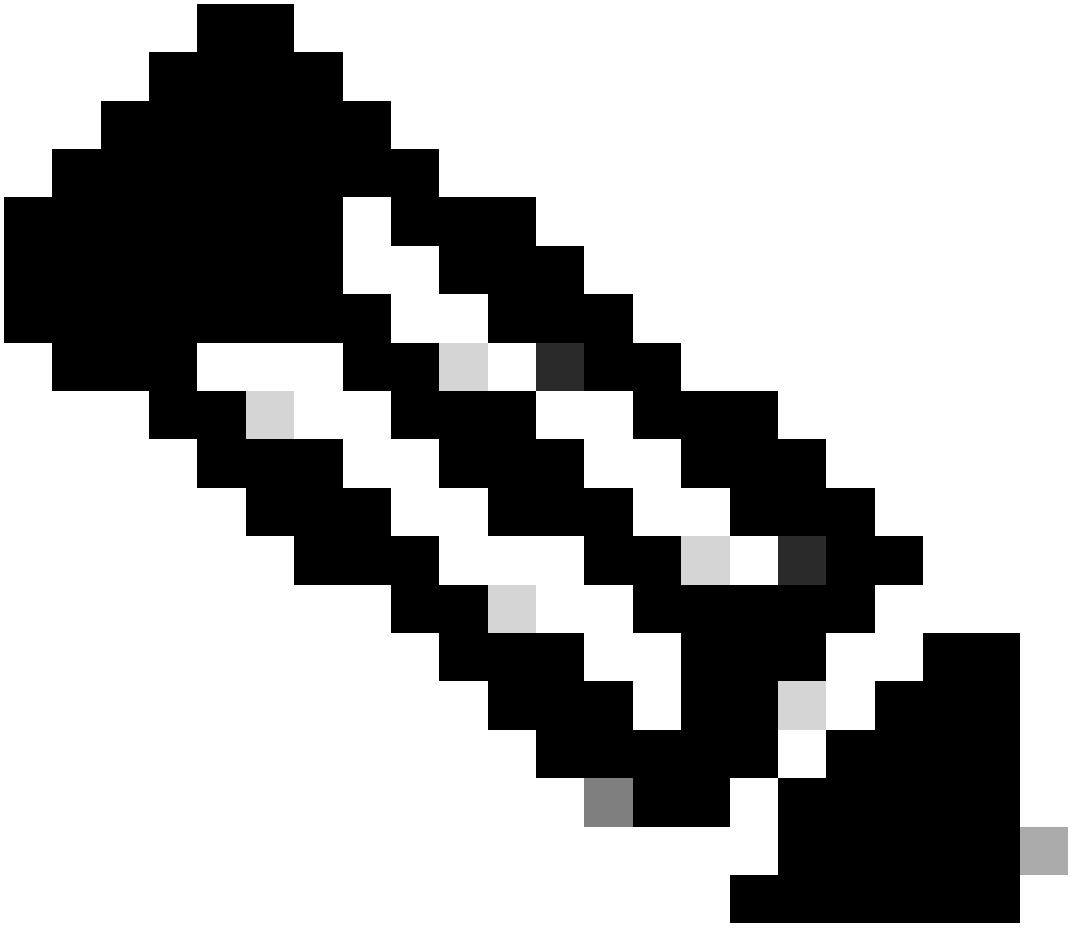
<#root>

```
SPOKE2#show running-config interface tunnel10
```

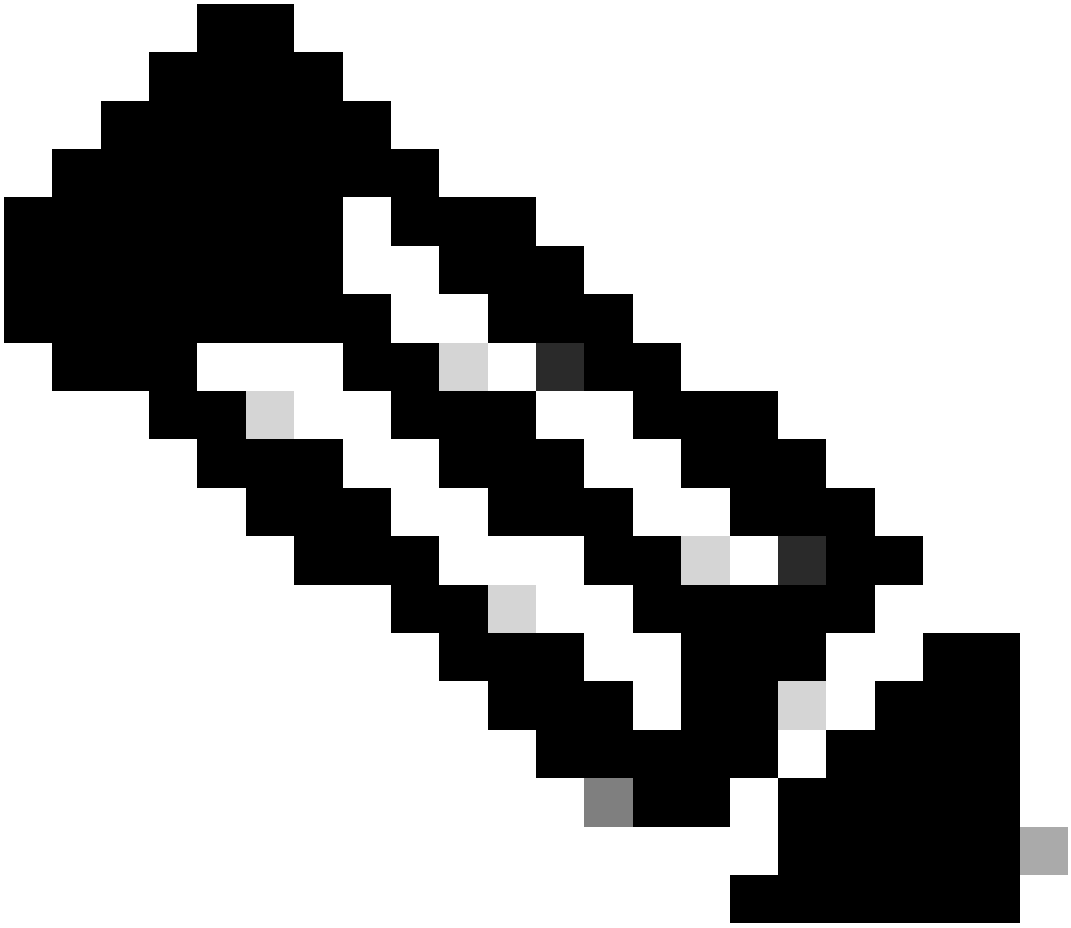
```
Building configuration...

Current configuration : 341 bytes
!
interface Tunnel10
ip address 10.10.10.2 255.255.255.0
no ip redirects
ip nhrp authentication DMVPN


ip nhrp map 10.10.10.10 172.20.10.10


ip nhrp map multicast 172.20.10.10


ip nhrp network-id 10
ip nhrp nhs 10.10.10.10


tunnel source GigabitEthernet1
tunnel mode gre multipoint


tunnel protection IPSEC profile IPSEC_Profile_1


end
```

컨피그레이션에서 HUB에 대한 매핑이 올바른지, NHRP 인증 문자열이 디바이스 간에 일치하는지, 두 스포크가 동일한 DMVPN 단계가 구성되어 있는지, 그리고 IPSEC 보호가 사용되는 경우 올바른 암호화 컨피그레이션이 적용되는지 확인해야 합니다.

컨피그레이션이 올바르고 IPSEC 보호가 포함된 경우 IKE 및 IPSEC 프로토콜이 올바르게 작동하는지 확인해야 합니다. 이는 NHRP가 IPSEC 터널을 전송 방법으로 사용하여 완전히 협상하기 때문입니다. IKE/IPSEC 프로토콜의 상태를 확인하려면 명령을 show crypto IPSEC sa peer x.x.x.x(여기서 x.x.x.x는 터널을 설정하려는 스포크의 NBMA IP 주소)로 실행합니다.



참고: IPSEC 터널이 작동 중인지 확인하려면 인바운드 및 아웃바운드 ESP(Encapsulation Security Payload) 섹션에 터널 정보(SPI, transform-set 등)가 있어야 합니다. 이 섹션에 표시된 모든 값은 양쪽 끝에서 일치해야 합니다.



참고: IKE/IPSEC에 문제가 있는 경우 트러블슈팅은 해당 프로토콜에 중점을 두어야 합니다 .

---

Spoke1의 IKE/IPSEC 터널 상태:

<#root>

SPOKE1#

**show crypto IPSEC sa peer 172.22.200.2**

```
interface: Tunnel10
Crypto map tag: Tunnel10-head-0, local addr 172.21.100.1

protected vrf: (none)
local ident (addr/mask/prot/port): (172.21.100.1/255.255.255.255/47/0)
remote ident (addr/mask/prot/port): (172.22.200.2/255.255.255.255/47/0)
current_peer 172.22.200.2 port 500
PERMIT, flags={origin_is_acl,}
```

**#pkts encaps: 0, #pkts encrypt: 0, #pkts digest: 0**

**#pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0**

#pkts compressed: 0, #pkts decompressed: 0
#pkts not compressed: 0, #pkts compr. failed: 0
#pkts not decompressed: 0, #pkts decompress failed: 0
#send errors 0, #recv errors 0

local crypto endpt.: 172.21.100.1, remote crypto endpt.: 172.22.200.2
plaintext mtu 1458, path mtu 1500, ip mtu 1500, ip mtu idb GigabitEthernet1
current outbound spi: 0x6F6BF94A(1869347146)
PFS (Y/N): N, DH group: none

**inbound esp sas:**

**spi: 0x84502A19(2219846169)**

**transform: esp-256-aes esp-sha256-hmac**

 ,
in use settings ={Transport, }
conn id: 2049, flow_id: CSR:49, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
sa timing: remaining key lifetime (k/sec): (4608000/28716)
IV size: 16 bytes
replay detection support: Y
Status: ACTIVE(ACTIVE)

inbound ah sas:

inbound pcp sas:

**outbound esp sas:**

**spi: 0x6F6BF94A(1869347146)**

**transform: esp-256-aes esp-sha256-hmac**

 ,
in use settings ={Transport, }
conn id: 2050, flow_id: CSR:50, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
sa timing: remaining key lifetime (k/sec): (4608000/28716)
IV size: 16 bytes
replay detection support: Y
Status: ACTIVE(ACTIVE)

outbound ah sas:

outbound pcp sas:

Spoke2의 IKE/IPSEC 터널 상태:

<#root>

SPOKE2#

**show crypto IPSEC sa peer 172.21.100.1**

interface: Tunnel10
Crypto map tag: Tunnel10-head-0, local addr 172.22.200.2

protected vrf: (none)
local ident (addr/mask/prot/port): (172.22.200.2/255.255.255.255/47/0)
remote ident (addr/mask/prot/port): (172.21.100.1/255.255.255.255/47/0)
current_peer 172.21.100.1 port 500
PERMIT, flags={origin_is_acl,}

**#pkts encaps: 16, #pkts encrypt: 16, #pkts digest: 16**

**#pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0**

#pkts compressed: 0, #pkts decompressed: 0
#pkts not compressed: 0, #pkts compr. failed: 0
#pkts not decompressed: 0, #pkts decompress failed: 0
#send errors 0, #recv errors 0

local crypto endpt.: 172.22.200.2, remote crypto endpt.: 172.21.100.1
plaintext mtu 1458, path mtu 1500, ip mtu 1500, ip mtu idb GigabitEthernet1
current outbound spi: 0x84502A19(2219846169)
PFS (Y/N): N, DH group: none

**inbound esp sas:**

**spi: 0x6F6BF94A(1869347146)**

**transform: esp-256-aes esp-sha256-hmac ,**

in use settings ={Transport, }
conn id: 2045, flow_id: CSR:45, sibling_flags FFFFFFFF80004008, crypto map: Tunnel10-head-0
sa timing: remaining key lifetime (k/sec): (4608000/28523)
IV size: 16 bytes
replay detection support: Y
Status: ACTIVE(ACTIVE)

inbound ah sas:

inbound pcp sas:

**outbound esp sas:**

```
spi: 0x84502A19(2219846169)


transform: esp-256-aes esp-sha256-hmac

 ,
in use settings ={Transport, }
conn id: 2046, flow_id: CSR:46, sibling_flags FFFFFFFF80004008, crypto map: Tunnel10-head-0
sa timing: remaining key lifetime (k/sec): (4607998/28523)
IV size: 16 bytes
replay detection support: Y
Status: ACTIVE(ACTIVE)

outbound ah sas:

outbound pcp sas:
```

출력은 두 스포크 모두에서 IPSEC 터널이 작동함을 보여주지만, Spoke2는 암호화된 패킷(캡슐화)은 보여주지만 해독된 패킷(디캡은 보여주지 않습니다. 한편 Spoke1은 IPSEC 터널을 통해 흐르는 패킷을 표시하지 않습니다. 이는 NHRP 프로토콜에 문제가 있을 수 있음을 나타냅니다.

## 문제 해결 툴

초기 검증을 수행하고 컨피그레이션을 확인한 후 IKE/IPSEC 프로토콜(필요한 경우)로 인해 통신 문제가 발생하지 않으면 이 섹션에 나와 있는 툴을 사용하여 트러블슈팅을 계속할 수 있습니다.

유용한 명령

이 명령은 show dmvpn interface tunnel<ID>을 통해 DMVPN별 세션 정보(NBMA/터널 IP 주소, 터널 상태, 작동/작동 중지 시간 및 특성)를 제공합니다. crypto 세션/소켓의 세부 정보를 표시하려면 detail 키워드를 사용할 수 있습니다. 터널의 상태는 양쪽 끝에서 일치해야 한다는 것을 언급하는 것이 중요합니다.

Spoke 1 show dmvpn interface tunnel<ID> 출력:

```
<#root>

SPOKE1#

show dmvpn interface tunnel10


Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
N - NATed, L - Local, X - No Socket
T1 - Route Installed, T2 - Nexthop-override, B - BGP
C - CTS Capable, I2 - Temporary
# Ent --> Number of NHRP entries with same NBMA peer
NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
UpDn Time --> Up or Down Time for a Tunnel
==============================================================================

Interface: Tunnel10, IPv4 NHRP Details
Type:Spoke, NHRP Peers:1,
```

```
# Ent Peer NBMA Addr Peer Tunnel Add  State UpDn Tm  Attrb
----- --------------- --------------- ----- -------- -----
   2
172.20.10.10     10.10.10.2      UP  00:00:51  I2

                        10.10.10.10     UP  02:53:27  S
```

Spoke 2 show dmvpn interface tunnel<ID> 출력:

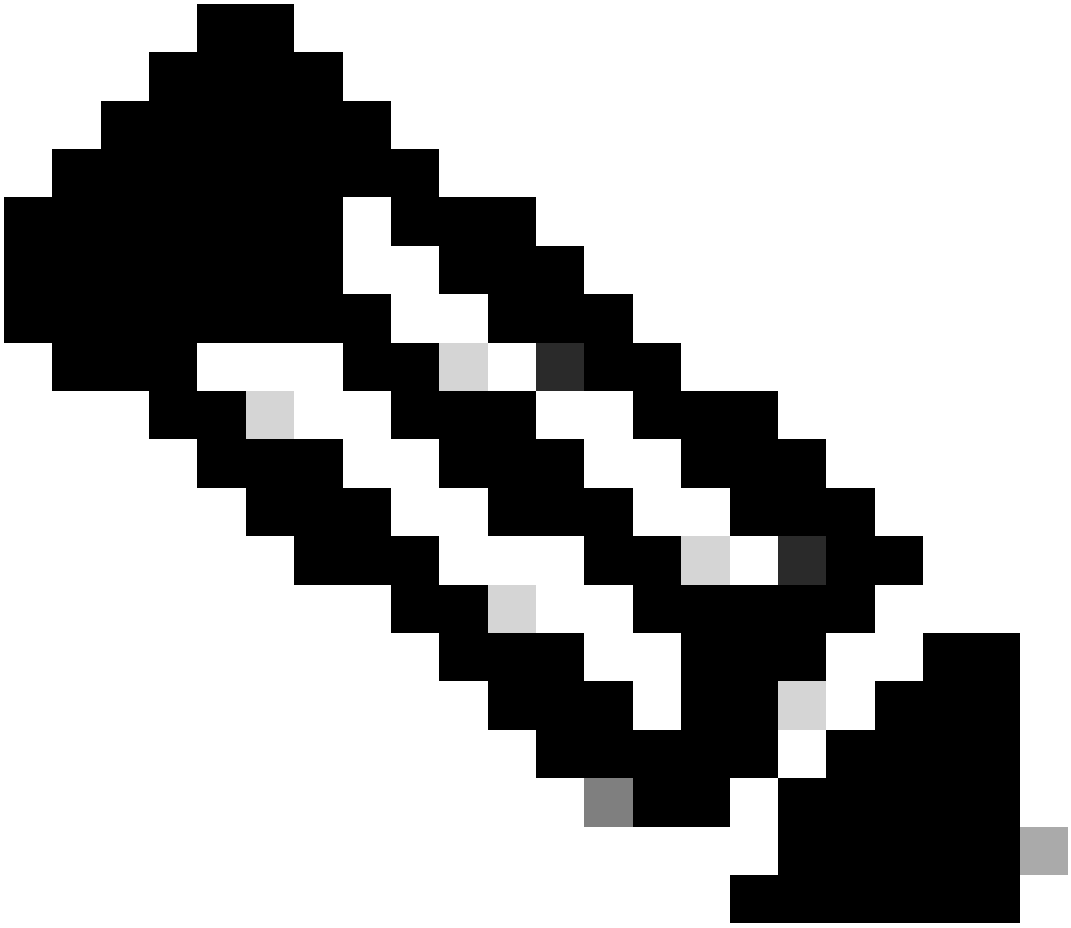
<#root>

```
SPOKE2#
show dmvpn interface tunnel10


Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
N - NATed, L - Local, X - No Socket
T1 - Route Installed, T2 - Nexthop-override, B - BGP
C - CTS Capable, I2 - Temporary
# Ent --> Number of NHRP entries with same NBMA peer
NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
UpDn Time --> Up or Down Time for a Tunnel
==============================================================================

Interface: Tunnel10, IPv4 NHRP Details
Type:Spoke, NHRP Peers:2,

# Ent Peer NBMA Addr Peer Tunnel Add State UpDn Tm Attrb
----- --------------- --------------- ----- -------- -----


1    172.21.100.1     10.10.10.1     UP  00:03:53   D


  1    172.20.10.10     10.10.10.10    UP  02:59:14   S
```

각 장치의 출력에는 스포크마다 다른 정보가 표시됩니다. Spoke1 테이블에서 Spoke 2에 대한 항목에 올바른 NBMA IP 주소가 포함되어 있지 않으며 특성이 불완전한 것(I2)으로 표시됨을 확인할 수 있습니다. 반면, Spoke2 테이블에는 올바른 매핑(NBMA/터널 IP 주소)과 터널이 완전히 협상되었음을 나타내는 up 상태가 표시됩니다.

다음 명령은 문제 해결 프로세스 중에 도움이 될 수 있습니다.

- show ip nhrp: NHRP 매핑 정보 표시
- show ip nhrp traffic interface tunnel10: NHRP 트래픽 통계를 표시합니다.



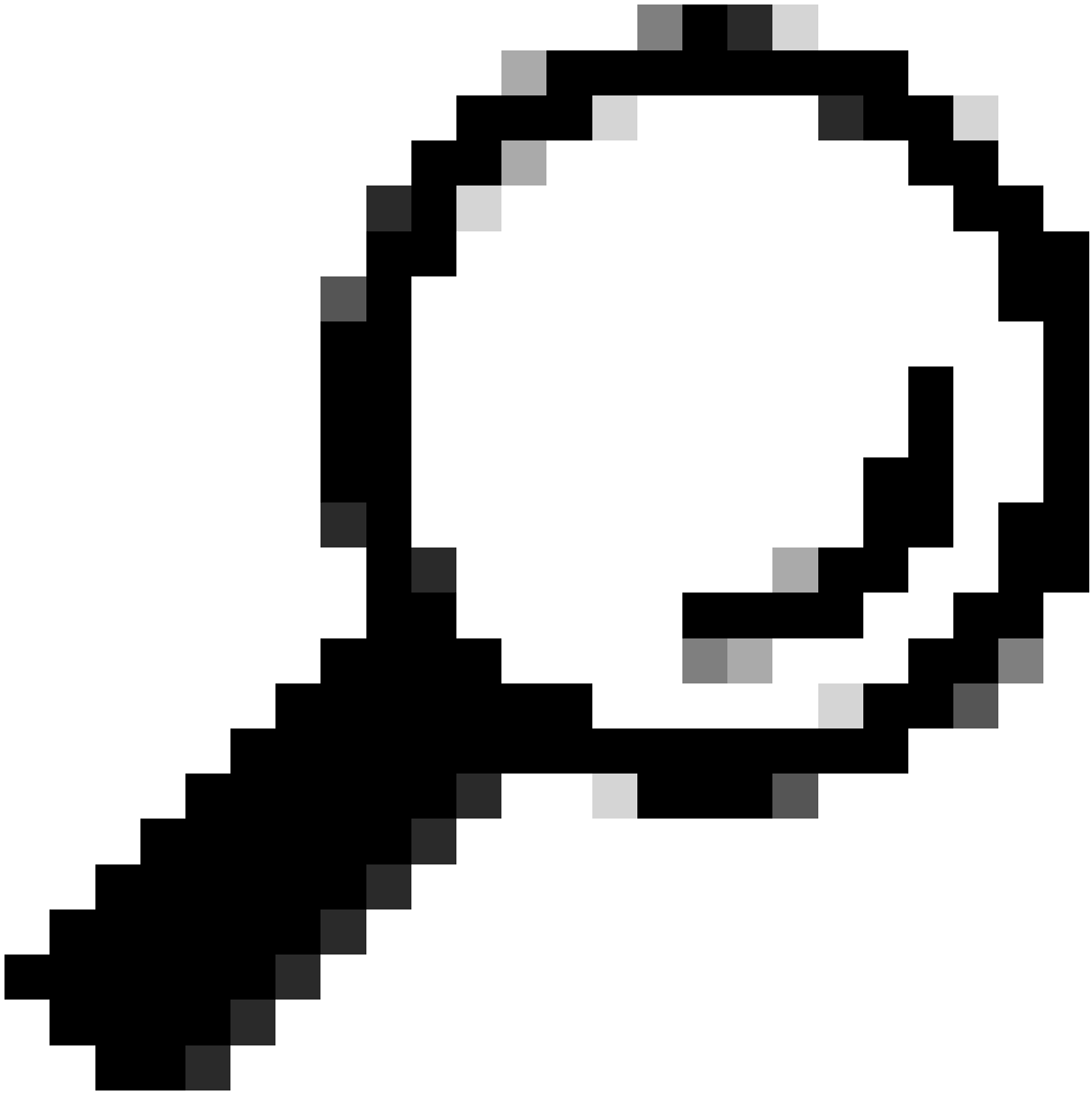
참고: 명령 사양(구문, 설명, 키워드, 예)은 Command Reference: Cisco IOS Security Command Reference: Commands S to Z를 참조하십시오

---

디버그

이전 정보를 확인하고 터널에 협상 문제가 있음을 확인한 후 디버그를 활성화하여 NHRP 패킷이 어떻게 교환되는지 관찰해야 합니다. 관련된 모든 디바이스에서 다음 디버그를 활성화해야 합니다.

1. debug dmvpn condition peer NBMA x.x.x.x(여기서 x.x.x.x는 원격 디바이스 IP 주소).
2. debug dmvpn all all: 이 명령은 ISAKMP, IKEv2, IPSEC, DMVPN 및 NHRP 디버깅 명령을 활성화합니다.



팁: 디버그를 활성화할 때마다 peer condition 명령을 사용하여 해당 특정 터널의 협상을 볼 수 있도록 하는 것이 좋습니다.

---

전체 NHRP 플로우를 보기 위해 각 디바이스에서 다음 debugs 명령을 사용했습니다.

스포크1

```
debug dmvpn condition peer NBMA 172.22.200.2
debug dmvpn condition peer NBMA 172.20.10.10
debug dmvpn all all
```

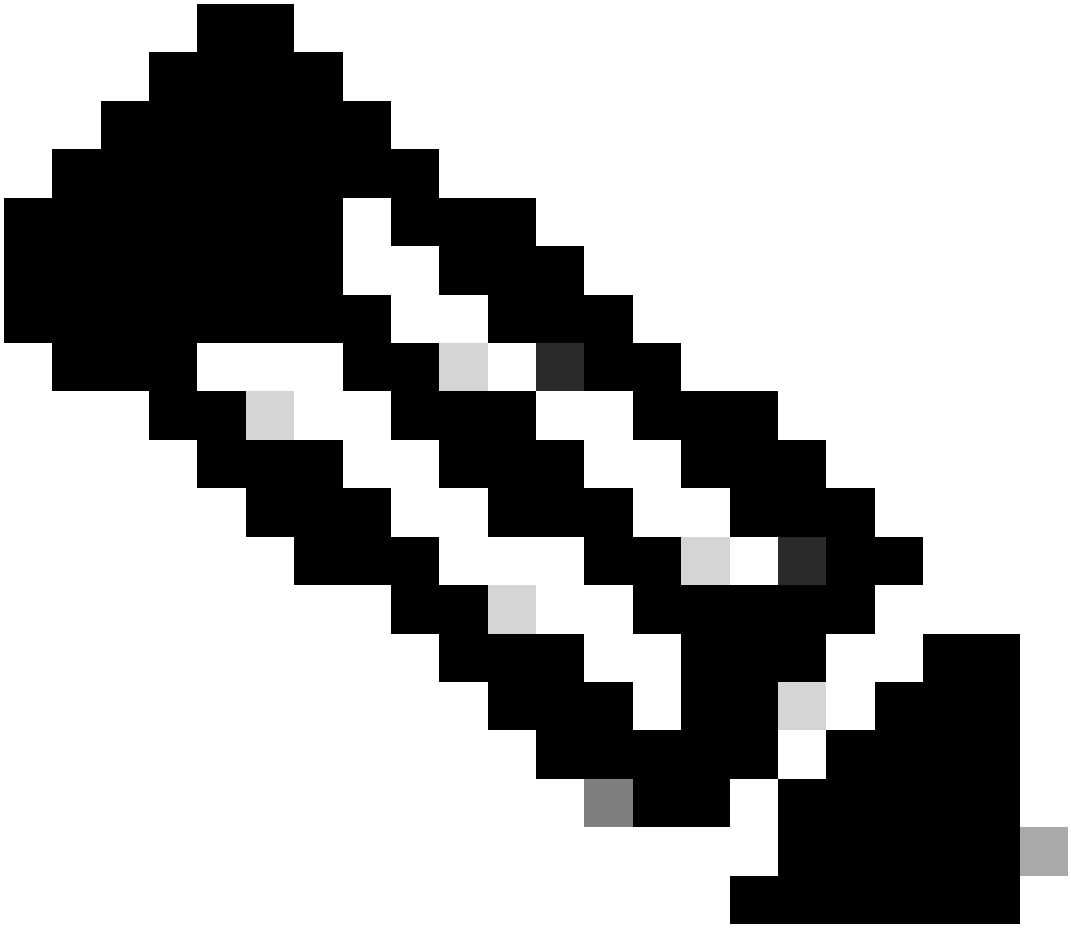
허브

```
debug dmvpn condition peer NBMA 172.21.100.1
debug dmvpn condition peer NBMA 172.22.200.2
debug dmvpn all all
```

스포크2

```
debug dmvpn condition peer NBMA 172.21.100.1
debug dmvpn condition peer NBMA 172.20.10.10
debug dmvpn all all
```

---



참고: 관련된 모든 디바이스에서 디버그를 동시에 활성화하고 수집해야 합니다.

---

모든 디바이스에서 활성화된 디버깅은 show debug 명령을 사용하여 표시됩니다.

<#root>

ROUTER#

**show debug**

```
IOSXE Conditional Debug Configs:

Conditional Debug Global State: Stop


IOSXE Packet Tracing Configs:


Packet Infra debugs:

Ip Address Port
------------------------------------------------------|----------

NHRP:
NHRP protocol debugging is on
NHRP activity debugging is on
NHRP detail debugging is on
NHRP extension processing debugging is on
NHRP cache operations debugging is on
NHRP routing debugging is on
NHRP rate limiting debugging is on
NHRP errors debugging is on
NHRP events debugging is on

Cryptographic Subsystem:
Crypto ISAKMP debugging is on
Crypto ISAKMP Error debugging is on
Crypto IPSEC debugging is on
Crypto IPSEC Error debugging is on
Crypto secure socket events debugging is on
IKEV2:
IKEv2 error debugging is on
IKEv2 default debugging is on
IKEv2 packet debugging is on
IKEv2 packet hexdump debugging is on
IKEv2 internal debugging is on
Tunnel Protection Debugs:
Generic Tunnel Protection debugging is on

DMVPN:
DMVPN error debugging is on
DMVPN UP/DOWN event debugging is on
DMVPN detail debugging is on
DMVPN packet debugging is on
DMVPN all level debugging is on
```

모든 디버그를 수집한 후 소스 스포크(Spoke1)에서 디버깅 분석을 시작해야 합니다. 이렇게 하면 협상을 처음부터 추적할 수 있습니다.

Spoke1 디버그 출력:

<#root>

```
-------------------- [IKE/IPSEC DEBUG OUTPUTS OMITTED]--------------------

*Feb 1 01:31:34.657: ISAKMP: (1016):

Old State = IKE_QM_R_QM2 New State = IKE_QM_PHASE2_COMPLETE


*Feb 1 01:31:34.657: IPSEC(key_engine): got a queue event with 1 KMI message(s)
*Feb 1 01:31:34.657: IPSEC(key_engine_enable_outbound): rec'd enable notify from ISAKMP
*Feb 1 01:31:34.657: CRYPTO_SS(TUNNEL SEC): Sending MTU Changed message
*Feb 1 01:31:34.661: IPSEC-IFC MGRE/Tu10(172.21.100.1/172.22.200.2): Got MTU message mtu 1458
*Feb 1 01:31:34.661: IPSEC-IFC MGRE/Tu10(172.21.100.1/172.22.200.2): connection lookup returned 80007F2
*Feb 1 01:31:34.662: CRYPTO_SS(TUNNEL SEC): Sending Socket Up message
*Feb 1 01:31:34.662: IPSEC-IFC MGRE/Tu10(172.21.100.1/172.22.200.2): connection lookup returned 80007F2
*Feb 1 01:31:34.662: IPSEC-IFC MGRE/Tu10(172.21.100.1/172.22.200.2):

tunnel_protection_socket_up


*Feb 1 01:31:34.662: IPSEC-IFC MGRE/Tu10(172.21.100.1/172.22.200.2): Signalling NHRP
*Feb 1 01:31:36.428: NHRP: Checking for delayed event NULL/10.10.10.2 on list (Tunnel10 vrf: global(0x0
*Feb 1 01:31:36.429: NHRP: No delayed event node found.
*Feb 1 01:31:36.429: NHRP: There is no VPE Extension to construct for the request
*Feb 1 01:31:36.429: NHRP: Sending NHRP Resolution Request for dest: 10.10.10.2 to nexthop: 10.10.10.2
*Feb 1 01:31:36.429: NHRP: Attempting to send packet through interface Tunnel10 via DEST dst 10.10.10.2
*Feb 1 01:31:36.429: NHRP-DETAIL: First hop route lookup for 10.10.10.2 yielded 10.10.10.2, Tunnel10
*Feb 1 01:31:36.429: NHRP:

Send Resolution Request via Tunnel10 vrf: global(0x0), packet size: 85


*Feb 1 01:31:36.429: src: 10.10.10.1, dst: 10.10.10.2
*Feb 1 01:31:36.429: (F) afn: AF_IP(1), type: IP(800), hop: 255, ver: 1
*Feb 1 01:31:36.429: shtl: 4(NSAP), sstl: 0(NSAP)
*Feb 1 01:31:36.429: pktsz: 85 extoff: 52
*Feb 1 01:31:36.429: (M) flags: "router auth src-stable nat ",

reqid: 10


*Feb 1 01:31:36.429:

src NBMA: 172.21.100.1


*Feb 1 01:31:36.429:

src protocol: 10.10.10.1, dst protocol: 10.10.10.2


*Feb 1 01:31:36.429: (C-1) code: no error(0), flags: none
*Feb 1 01:31:36.429: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:36.429: addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 0, pref: 255
*Feb 1 01:31:36.429: Responder Address Extension(3):
*Feb 1 01:31:36.429: Forward Transit NHS Record Extension(4):
*Feb 1 01:31:36.429: Reverse Transit NHS Record Extension(5):
*Feb 1 01:31:36.429: Authentication Extension(7):
*Feb 1 01:31:36.429: type:Cleartext(1),

data:DMVPN


*Feb 1 01:31:36.429: NAT address Extension(9):
```

```
*Feb 1 01:31:36.430: NHRP: Encapsulation succeeded. Sending NHRP Control Packet NBMA Address: 172.20.10
*Feb 1 01:31:36.430: NHRP: 109 bytes out Tunnel10
*Feb 1 01:31:36.430: NHRP-RATE:
```

**Retransmitting Resolution Request for 10.10.10.2, reqid 10, (retrans ivl 4 sec)**

```
*Feb 1 01:31:39.816: NHRP: Checking for delayed event NULL/10.10.10.2 on list (Tunnel10 vrf: global(0x0)
*Feb 1 01:31:39.816: NHRP: No delayed event node found.
*Feb 1 01:31:39.816: NHRP: There is no VPE Extension to construct for the request
*Feb 1 01:31:39.817: NHRP: Sending NHRP Resolution Request for dest: 10.10.10.2 to nexthop: 10.10.10.2
*Feb 1 01:31:39.817: NHRP: Attempting to send packet through interface Tunnel10 via DEST dst 10.10.10.2
*Feb 1 01:31:39.817: NHRP-DETAIL: First hop route lookup for 10.10.10.2 yielded 10.10.10.2, Tunnel10
*Feb 1 01:31:39.817: NHRP:
```

**Send Resolution Request via Tunnel10 vrf: global(0x0), packet size: 85**

```
*Feb 1 01:31:39.817: src: 10.10.10.1, dst: 10.10.10.2
*Feb 1 01:31:39.817: (F) afn: AF_IP(1), type: IP(800), hop: 255, ver: 1
*Feb 1 01:31:39.817: shtl: 4(NSAP), sstl: 0(NSAP)
*Feb 1 01:31:39.817: pktsz: 85 extoff: 52
*Feb 1 01:31:39.817: (M) flags: "router auth src-stable nat ",
```

**reqid: 10**

```
*Feb 1 01:31:39.817:
```

**src NBMA: 172.21.100.1**

```
*Feb 1 01:31:39.817:
```

**src protocol: 10.10.10.1, dst protocol: 10.10.10.2**

```
*Feb 1 01:31:39.817: (C-1) code: no error(0), flags: none
*Feb 1 01:31:39.817: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:39.817: addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 0, pref: 255
*Feb 1 01:31:39.817: Responder Address Extension(3):
*Feb 1 01:31:39.817: Forward Transit NHS Record Extension(4):
*Feb 1 01:31:39.817: Reverse Transit NHS Record Extension(5):
*Feb 1 01:31:39.817: Authentication Extension(7):
*Feb 1 01:31:39.817: type:Cleartext(1),
```

**data:DMVPN**

```
*Feb 1 01:31:39.817: NAT address Extension(9):
*Feb 1 01:31:39.817: NHRP: Encapsulation succeeded. Sending NHRP Control Packet NBMA Address: 172.20.10
*Feb 1 01:31:39.818: NHRP: 109 bytes out Tunnel10
*Feb 1 01:31:39.818: NHRP-RATE:
```

**Retransmitting Resolution Request for 10.10.10.2, reqid 10, (retrans ivl 8 sec)**

```
*Feb 1 01:31:46.039: NHRP: Checking for delayed event NULL/10.10.10.2 on list (Tunnel10 vrf: global(0x0)
*Feb 1 01:31:46.040: NHRP: No delayed event node found.
*Feb 1 01:31:46.040: NHRP: There is no VPE Extension to construct for the request
```

Spoke1 NHRP 프로세스가 시작되면 로그에 디바이스가 NHRP 확인 요청을 전송하고 있음을 보여 줍니다. 패킷에는 소스 스포크(Spoke1)의 NBMA IP 주소와 터널 IP 주소인 src NMBA 및 src 프로토

콜과 같은 몇 가지 중요한 정보가 있습니다. 대상 스포크(Spoke2)의 터널 IP 주소가 있는 dst 프로토콜 값도 볼 수 있습니다. 이는 Spoke1이 매핑을 완료하기 위해 Spoke2의 NBMA 주소를 요청함을 나타냅니다. 또한 패킷에서 경로를 따라 패킷을 추적하는 데 도움이 되는 reqid 값을 찾을 수 있습니다. 이 값은 전체 프로세스를 통해 동일하게 유지되며 NHRP 협상의 특정 흐름을 추적하는 데 도움이 될 수 있습니다. 패킷에 NHRP 인증 문자열과 같이 협상에 중요한 다른 값이 있습니다.

디바이스가 NHRP 확인 요청을 전송한 후, 로그에 재전송이 전송됨을 보여줍니다. 디바이스가 NHRP 확인 응답을 보지 못하므로 패킷을 다시 전송하기 때문입니다. Spoke1에서 응답을 볼 수 없으므로 경로의 다음 디바이스, 즉 HUB에서 해당 패킷을 추적해야 합니다.

허브 디버그 출력:

```
<#root>

*Feb 1 01:31:34.262:

NHRP: Receive Resolution Request via Tunnel10 vrf: global(0x0), packet size: 85


*Feb 1 01:31:34.262: (F) afn: AF_IP(1), type: IP(800), hop: 255, ver: 1
*Feb 1 01:31:34.262: shtl: 4(NSAP), sstl: 0(NSAP)
*Feb 1 01:31:34.263: pktsz: 85 extoff: 52
*Feb 1 01:31:34.263: (M) flags: "router auth src-stable nat ",

reqid: 10


*Feb 1 01:31:34.263:

src NBMA: 172.21.100.1


*Feb 1 01:31:34.263:

src protocol: 10.10.10.1, dst protocol: 10.10.10.2


*Feb 1 01:31:34.263: (C-1) code: no error(0), flags: none
*Feb 1 01:31:34.263: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:34.263: addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 0, pref: 255
*Feb 1 01:31:34.263: Responder Address Extension(3):
*Feb 1 01:31:34.263: Forward Transit NHS Record Extension(4):
*Feb 1 01:31:34.263: Reverse Transit NHS Record Extension(5):
*Feb 1 01:31:34.263: Authentication Extension(7):
*Feb 1 01:31:34.263: type:Cleartext(1), data:DMVPN
*Feb 1 01:31:34.263: NAT address Extension(9):
*Feb 1 01:31:34.263: NHRP-DETAIL: netid_in = 10, to_us = 0
*Feb 1 01:31:34.263: NHRP-DETAIL:

Resolution request for afn 1 received on interface Tunnel10

 , for vrf: global(0x0) label: 0
*Feb 1 01:31:34.263: NHRP-DETAIL: Multipath IP route lookup for 10.10.10.2 in vrf: global(0x0) yielded
*Feb 1 01:31:34.263: NHRP:

Route lookup for destination 10.10.10.2

 in vrf: global(0x0) yielded interface Tunnel10, prefixlen 24
*Feb 1 01:31:34.263: NHRP-DETAIL: netid_out 10, netid_in 10
*Feb 1 01:31:34.263: NHRP: Forwarding request due to authoritative request.
*Feb 1 01:31:34.263: NHRP-ATTR:
```

**NHRP Resolution Request packet is forwarded to 10.10.10.2 using vrf: global(0x0)**

*Feb 1 01:31:34.263: NHRP: Attempting to forward to destination: 10.10.10.2 vrf: global(0x0)
*Feb 1 01:31:34.264: NHRP: Forwarding: NHRP SAS picked source: 10.10.10.10 for destination: 10.10.10.2
*Feb 1 01:31:34.264: NHRP: Attempting to send packet through interface Tunnel10 via DEST dst 10.10.10.2
*Feb 1 01:31:34.264: NHRP-DETAIL: First hop route lookup for 10.10.10.2 yielded 10.10.10.2, Tunnel10
*Feb 1 01:31:34.264: NHRP:

**Forwarding Resolution Request via Tunnel10 vrf: global(0x0), packet size: 105**

*Feb 1 01:31:34.264: src: 10.10.10.10, dst: 10.10.10.2
*Feb 1 01:31:34.264: (F) afn: AF_IP(1), type: IP(800), hop: 254, ver: 1
*Feb 1 01:31:34.264: shtl: 4(NSAP), sstl: 0(NSAP)
*Feb 1 01:31:34.264: pktsz: 105 extoff: 52
*Feb 1 01:31:34.264: (M) flags: "router auth src-stable nat ",

**reqid: 10**

*Feb 1 01:31:34.264:

**src NBMA: 172.21.100.1**

*Feb 1 01:31:34.264:

**src protocol: 10.10.10.1, dst protocol: 10.10.10.2**

*Feb 1 01:31:34.264: (C-1) code: no error(0), flags: none
*Feb 1 01:31:34.264: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:34.264: addr_len: 0(NSAP), subaddr_len: 0(NSAP), proto_len: 0, pref: 255
*Feb 1 01:31:34.264: Responder Address Extension(3):
*Feb 1 01:31:34.264: Forward Transit NHS Record Extension(4):
*Feb 1 01:31:34.264: (C-1)

**code: no error(0)**

, flags: none
*Feb 1 01:31:34.264: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:34.264: addr_len: 4(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 255
*Feb 1 01:31:34.264:

**client NBMA: 172.20.10.10**

*Feb 1 01:31:34.264:

**client protocol: 10.10.10.10**

*Feb 1 01:31:34.264: Reverse Transit NHS Record Extension(5):
*Feb 1 01:31:34.264: Authentication Extension(7):
*Feb 1 01:31:34.264: type:Cleartext(1),

**data:DMVPN**

*Feb 1 01:31:34.265: NAT address Extension(9):
*Feb 1 01:31:34.265: NHRP: Encapsulation succeeded. Sending NHRP Control Packet NBMA Address: 172.22.20
*Feb 1 01:31:34.265: NHRP: 129 bytes out Tunnel10

reqid 값을 사용하여 HUB가 Spoke1에서 전송한 해결 요청을 수신하는지 확인할 수 있습니다. 패킷에서 src NBMA 및 src 프로토콜의 값은 Spoke1의 정보이며, dst 프로토콜의 값은 Spoke1의 디버그에서 볼 수 있듯이 Spoke2의 터널 IP입니다. HUB는 해결 요청을 수신하면 경로 조회를 수행하고 패킷을 Spoke2에 전달합니다. 전달된 패킷에서 HUB는 자체 정보(NBMA IP 주소 및 터널 IP 주소)를 포함하는 확장을 추가합니다.

이전 디버그는 HUB가 확인 요청을 스포크 2로 올바르게 포워딩하고 있음을 보여줍니다. 따라서 다음 단계는 Spoke2가 이를 수신하고 올바르게 처리하여 Spoke1에 해결 응답을 보내고 있는지 확인하는 것입니다.

Spoke2 디버그 출력:

<#root>

```
-------------------- [IKE/IPSEC DEBUG OUTPUTS OMITTED]--------------------

*Feb 1 01:31:34.647: ISAKMP: (1015):

Old State = IKE_QM_IPSEC_INSTALL_AWAIT New State = IKE_QM_PHASE2_COMPLETE


*Feb 1 01:31:34.647: NHRP: Process delayed resolution request src:10.10.10.1 dst:10.10.10.2 vrf: global
*Feb 1 01:31:34.648: NHRP-DETAIL: Resolution request for afn 1 received on interface Tunnel10 , for vrf
*Feb 1 01:31:34.648: NHRP-DETAIL: Multipath IP route lookup for 10.10.10.2 in vrf: global(0x0) yielded
*Feb 1 01:31:34.648: NHRP:

Route lookup for destination 10.10.10.2 in vrf: global(0x0) yielded interface Tunnel10, prefixlen 24


*Feb 1 01:31:34.648: NHRP-ATTR: smart spoke feature and attributes are not configured
*Feb 1 01:31:34.648:

NHRP:


Request was to us. Process the NHRP Resolution Request.


*Feb 1 01:31:34.648: NHRP-DETAIL: Multipath IP route lookup for 10.10.10.2 in vrf: global(0x0) yielded
*Feb 1 01:31:34.648: NHRP: nhrp_rtlookup for 10.10.10.2 in vrf: global(0x0) yielded interface Tunnel10,
*Feb 1 01:31:34.648: NHRP: Request was to us, responding with ouraddress
*Feb 1 01:31:34.648: NHRP: Checking for delayed event 10.10.10.1/10.10.10.2 on list (Tunnel10 vrf: glob
*Feb 1 01:31:34.648: NHRP: No delayed event node found.
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10: Checking to see if we need to delay for src 172.22.200.2 dst
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10: crypto_ss_listen_start already listening
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): Opening a socket with profile IPSE
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): connection lookup returned 80007F1
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): Socket is already open. Ignoring.
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): connection lookup returned 80007F1
*Feb 1 01:31:34.648: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): tunnel is already open!
*Feb 1 01:31:34.648: NHRP: No need to delay processing of resolution event NBMA src:172.22.200.2 NBMA d
*Feb 1 01:31:34.648: NHRP-MEF: No vendor private extension in NHRP packet
*Feb 1 01:31:34.649: NHRP-CACHE: Tunnel10: Cache update for target 10.10.10.1/32 vrf: global(0x0) label
*Feb 1 01:31:34.649: 172.21.100.1 (flags:0x2080)
*Feb 1 01:31:34.649: NHRP:

Adding Tunnel Endpoints (VPN: 10.10.10.1, NBMA: 172.21.100.1)
```

```
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10: crypto_ss_listen_start already listening
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): Opening a socket with profile IPSE
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): connection lookup returned 80007F1
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): Found an existing tunnel endpoint
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): tunnel_protection_stop_pending_tim
*Feb 1 01:31:34.649: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): Socket is already open. Ignoring.
*Feb 1 01:31:34.653:
```

**NHRP: Successfully attached NHRP subblock for Tunnel Endpoints (VPN: 10.10.10.1, NBMA: 172.21.100.1)**

```
*Feb 1 01:31:34.653: NHRP: Peer capability:0
*Feb 1 01:31:34.653: NHRP-CACHE: Inserted subblock node(1 now) for cache: Target 10.10.10.1/32 nhop 10.
*Feb 1 01:31:34.653: NHRP-CACHE: Converted internal dynamic cache entry for 10.10.10.1/32 interface Tun
*Feb 1 01:31:34.653: NHRP-EVE: NHP-UP: 10.10.10.1, NBMA: 172.21.100.1
*Feb 1 01:31:34.653: NHRP-MEF: No vendor private extension in NHRP packet
*Feb 1 01:31:34.653: NHRP-CACHE: Tunnel10: Internal Cache add for target 10.10.10.2/32 vrf: global(0x0)
*Feb 1 01:31:34.653: 172.22.200.2 (flags:0x20)
*Feb 1 01:31:34.653: NHRP: Attempting to send packet through interface Tunnel10 via DEST dst 10.10.10.1
*Feb 1 01:31:34.654: NHRP-DETAIL: First hop route lookup for 10.10.10.1 yielded 10.10.10.1, Tunnel10
*Feb 1 01:31:34.654:
```

**NHRP: Send Resolution Reply via Tunnel10 vrf: global(0x0), packet size: 133**

```
*Feb 1 01:31:34.654: src: 10.10.10.2, dst: 10.10.10.1
*Feb 1 01:31:34.654: (F) afn: AF_IP(1), type: IP(800), hop: 255, ver: 1
*Feb 1 01:31:34.654: shtl: 4(NSAP), sstl: 0(NSAP)
*Feb 1 01:31:34.654: pktsz: 133 extoff: 60
*Feb 1 01:31:34.654: (M) flags: "router auth dst-stable unique src-stable nat ",
```

**reqid: 10**

```
*Feb 1 01:31:34.654:
```

**src NBMA: 172.21.100.1**

```
*Feb 1 01:31:34.654:
```

**src protocol: 10.10.10.1, dst protocol: 10.10.10.2**

```
*Feb 1 01:31:34.654: (C-1) code: no error(0), flags: none
*Feb 1 01:31:34.654: prefix: 32, mtu: 9976, hd_time: 599
*Feb 1 01:31:34.654: addr_len: 4(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 255
*Feb 1 01:31:34.654:
```

**client NBMA: 172.22.200.2**

```
*Feb 1 01:31:34.654:
```

**client protocol: 10.10.10.2**

```
*Feb 1 01:31:34.654: Responder Address Extension(3):
*Feb 1 01:31:34.654: (C) code: no error(0), flags: none
*Feb 1 01:31:34.654: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:34.654: addr_len: 4(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 255
*Feb 1 01:31:34.654:
```

**client NBMA: 172.22.200.2**

```
*Feb 1 01:31:34.654:
client protocol: 10.10.10.2


*Feb 1 01:31:34.654: Forward Transit NHS Record Extension(4):
*Feb 1 01:31:34.654: (C-1) code: no error(0), flags: none
*Feb 1 01:31:34.654: prefix: 0, mtu: 9976, hd_time: 600
*Feb 1 01:31:34.654: addr_len: 4(NSAP), subaddr_len: 0(NSAP), proto_len: 4, pref: 255
*Feb 1 01:31:34.654:
client NBMA: 172.20.10.10


*Feb 1 01:31:34.654:
client protocol: 10.10.10.10


*Feb 1 01:31:34.654: Reverse Transit NHS Record Extension(5):
*Feb 1 01:31:34.654: Authentication Extension(7):
*Feb 1 01:31:34.654: type:Cleartext(1),
data:DMVPN


*Feb 1 01:31:34.655: NAT address Extension(9):
*Feb 1 01:31:34.655: NHRP: Encapsulation succeeded. Sending NHRP Control Packet NBMA Address: 172.21.10
*Feb 1 01:31:34.655: NHRP: 157 bytes out Tunnel10
*Feb 1 01:31:34.655: IPSEC-IFC MGRE/Tu10(172.22.200.2/172.21.100.1): connection lookup returned 80007F1
*Feb 1 01:31:34.655: NHRP-DETAIL: Deleted delayed event on interfaceTunnel10 dest: 172.21.100.1
```

reqid는 이전 출력에서 확인한 값과 일치하며, 이를 통해 Spoke1에서 전송한 NHRP 확인 요청 패킷이 Spoke2에 도달하는 것이 확인됩니다. 이 패킷은 Spoke2에서 경로 조회를 트리거하고 해결 요청이 자신에 대한 것임을 인식하므로 Spoke2는 Spoke1의 정보를 NHRP 테이블에 추가합니다. 해결 응답 패킷을 Spoke1에 다시 보내기 전에 디바이스는 자신의 정보(NBMA IP 주소 및 터널 IP 주소)를 추가하여 Spoke1이 해당 패킷을 사용하여 해당 정보를 데이터베이스에 추가할 수 있도록 합니다.

표시된 모든 디버그를 기반으로 Spoke2에서 보낸 NHRP 해결 회신 이메일이 Spoke1에 도착하지 않습니다. HUB는 NHRP Resolution Request 패킷을 수신하고 예상대로 전달하므로 이 문제에서 폐기될 수 있습니다. 따라서 다음 단계는 Spoke1과 Spoke2 간에 캡처를 수행하여 문제에 대한 자세한 내용을 확인하는 것입니다.

임베디드 패킷 캡처

임베디드 패킷 캡처 기능을 사용하면 디바이스를 통과하는 트래픽을 분석할 수 있습니다. 이를 구성하는 첫 번째 단계는 두 트래픽 흐름(인바운드 및 아웃바운드)에서 캡처할 트래픽을 포함하는 액세스 목록을 생성하는 것입니다.

이 시나리오에서는 NBMA IP 주소가 사용됩니다.

```
ip access-list extended filter
```

```
10 permit ip host 172.21.100.1 host 172.22.200.2
20 permit ip host 172.22.200.2 host 172.21.100.1
```

그런 다음 명령 monitor capture <CAPTURE_NAME> access-list <ACL_NAME> buffer size 10 interface <WAN_INTERFACE> 둘 다를 사용하여 캡처를 구성하고 명령 monitor capture <CAPTURE_NAME> start로 캡처를 시작합니다.

Spoke1 및 Spoke2의 컨피그레이션 캡처:

```
monitor capture CAP access-list filter buffer size 10 interface GigabitEthernet1 both
monitor capture CAP start
```

캡처의 출력을 표시하려면 show monitor capture <CAPTURE_NAME> buffer brief 명령을 사용합니다.

캡처 출력 Spoke1:

<#root>

```
SPOKE1#show monitor capture CAP buffer brief
------------------------------------------------------------------------------
#   size   timestamp     source             destination      dscp    protocol
------------------------------------------------------------------------------
   0  210    0.000000   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   1  150    0.014999   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   2  478    0.028990   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   3  498    0.049985   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   4  150    0.069988   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   5  134    0.072994   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   6  230    0.074993   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   7  230    0.089992   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   8  118    0.100993   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   9  218    0.108988   172.22.200.2    ->  172.21.100.1     48 CS6  ESP
  10   70    0.108988   172.21.100.1    ->  172.22.200.2     0  BE   ICMP
  11  218    1.907994   172.22.200.2    ->  172.21.100.1     48 CS6  ESP
  12   70    1.907994   172.21.100.1    ->  172.22.200.2     0  BE   ICMP
  13  218    5.818003   172.22.200.2    ->  172.21.100.1     48 CS6  ESP
  14   70    5.818003   172.21.100.1    ->  172.22.200.2     0  BE   ICMP
```

**15 218 12.559969 172.22.200.2 -> 172.21.100.1 48 CS6 ESP**

**16 70 12.559969 172.21.100.1 -> 172.22.200.2 0 BE ICMP**

**17 218 26.859001 172.22.200.2 -> 172.21.100.1 48 CS6 ESP**

**18 70 26.859001 172.21.100.1 -> 172.22.200.2 0 BE ICMP**

**19 218 54.378978 172.22.200.2 -> 172.21.100.1 48 CS6 ESP**

**20 70 54.378978 172.21.100.1 -> 172.22.200.2 0 BE ICMP**

캡처 출력 Spoke2:

<#root>

```
SPOKE2#show monitor capture CAP buffer brief
------------------------------------------------------------------------
#   size   timestamp     source              destination      dscp    protocol
------------------------------------------------------------------------
   0  210    0.000000   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   1  150    0.015990   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   2  478    0.027998   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   3  498    0.050992   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   4  150    0.069988   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   5  134    0.072994   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   6  230    0.074993   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
   7  230    0.089992   172.21.100.1    ->  172.22.200.2     48 CS6  UDP
   8  118    0.099986   172.22.200.2    ->  172.21.100.1     48 CS6  UDP
```

**9 218 0.108988 172.22.200.2 -> 172.21.100.1 48 CS6 ESP**

**10 70 0.108988 172.21.100.1 -> 172.22.200.2 0 BE ICMP**

**11 218 1.907994 172.22.200.2 -> 172.21.100.1 48 CS6 ESP**

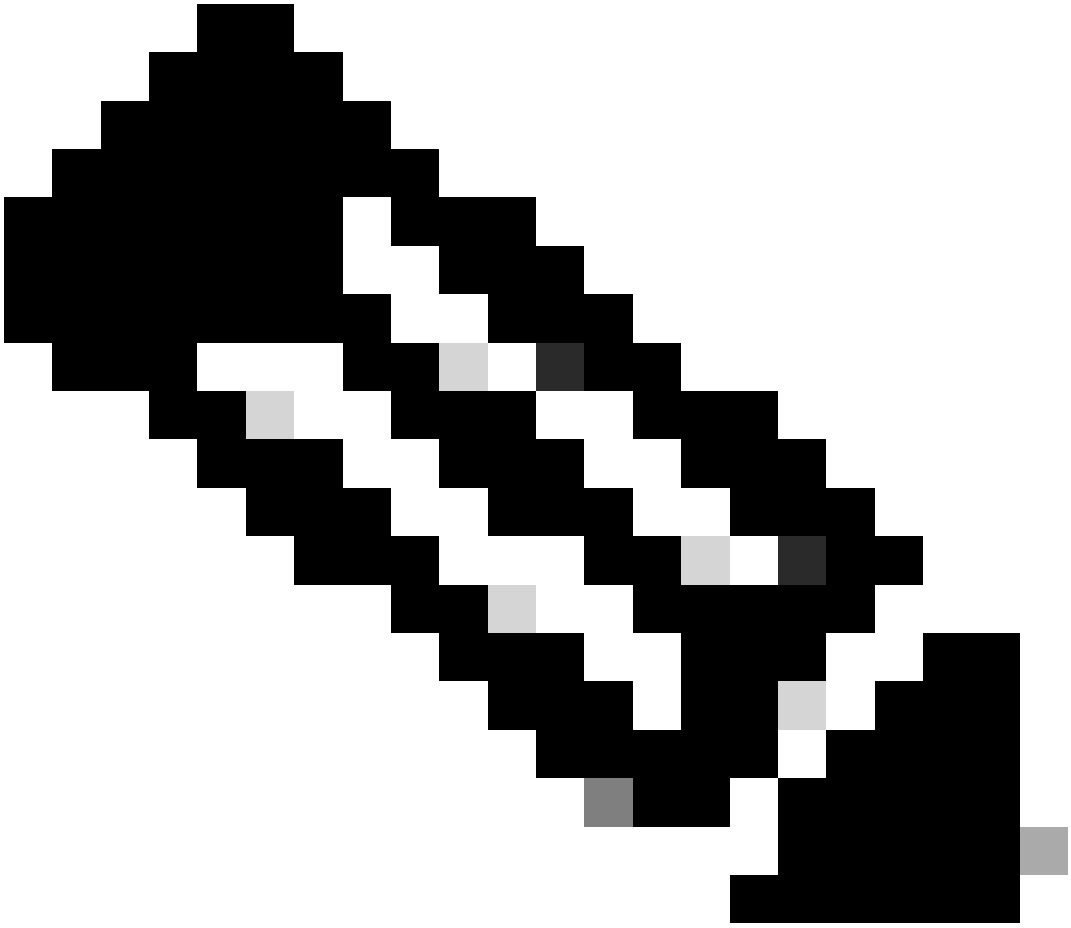
**12 70 1.909001 172.21.100.1 -> 172.22.200.2 0 BE ICMP**

**13 218 5.817011 172.22.200.2 -> 172.21.100.1 48 CS6 ESP**

```
14   70    5.818002   172.21.100.1    ->  172.22.200.2     0  BE   ICMP

15  218   12.559968   172.22.200.2    ->  172.21.100.1     48 CS6  ESP

16   70   12.560960   172.21.100.1    ->  172.22.200.2     0  BE   ICMP

17  218   26.858009   172.22.200.2    ->  172.21.100.1     48 CS6  ESP

18   70   26.859001   172.21.100.1    ->  172.22.200.2     0  BE   ICMP

19  218   54.378978   172.22.200.2   ->  172.21.100.1     48 CS6  ESP

20   70   54.379970   172.21.100.1    ->  172.22.200.2     0  BE   ICMP
```

캡처 출력은 초기 패킷이 UDP 트래픽임을 보여주며, 이는 IKE/IPSEC 협상을 나타냅니다. 그 후 Spoke2는 해결 응답을 Spoke1에 보냅니다. 이 응답은 ESP 트래픽(패킷 9)으로 표시됩니다. 이 이후에 예상되는 트래픽 흐름은 ESP이지만, 다음 패킷은 Spoke1에서 Spoke2로 들어오는 ICMP 트래픽입니다.

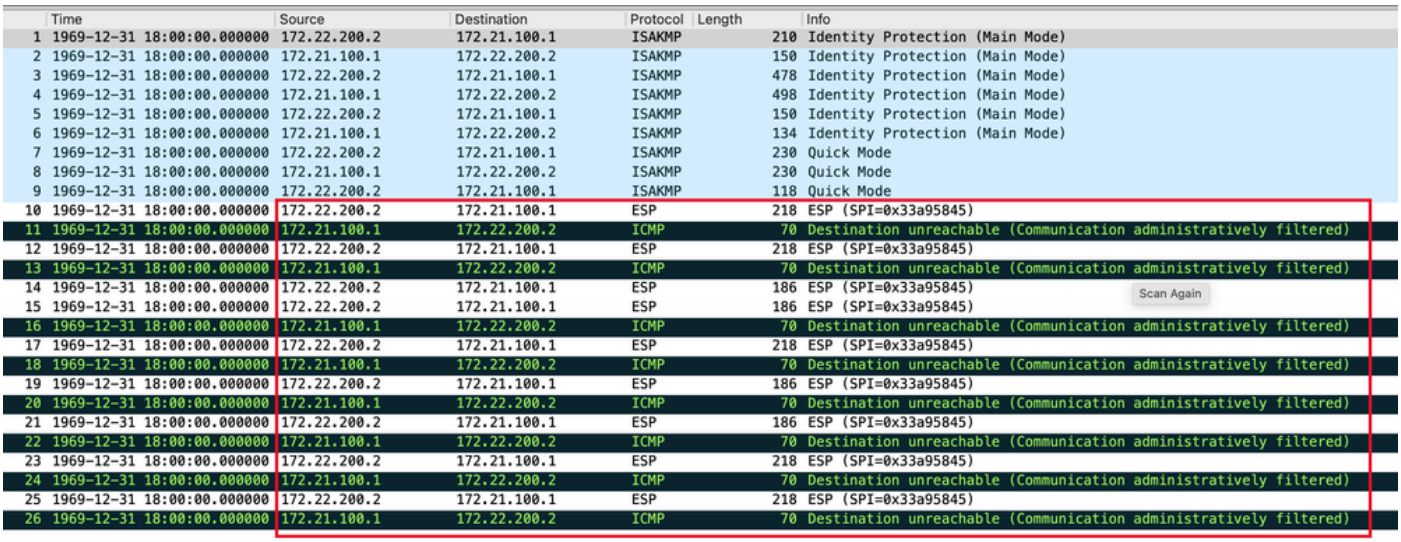
패킷을 심층적으로 분석하려면 show monitor capture <CAPTURE_NAME> 버퍼 덤프 명령을 실행하여 디바이스에서 pcap 파일을 내보낼 수 있습니다. 그런 다음 디코더 도구를 사용하여 덤프 출력을 pcap 파일로 변환하여 Wireshark를 사용하여 열 수 있도록 합니다.



참고: Cisco에는 캡처 구성, 예제 및 디코더를 찾을 수 있는 패킷 분석기가 있습니다. Cisco TAC Tool - Packet Capture Config Generator and Analyzer

---

Wireshark 출력:

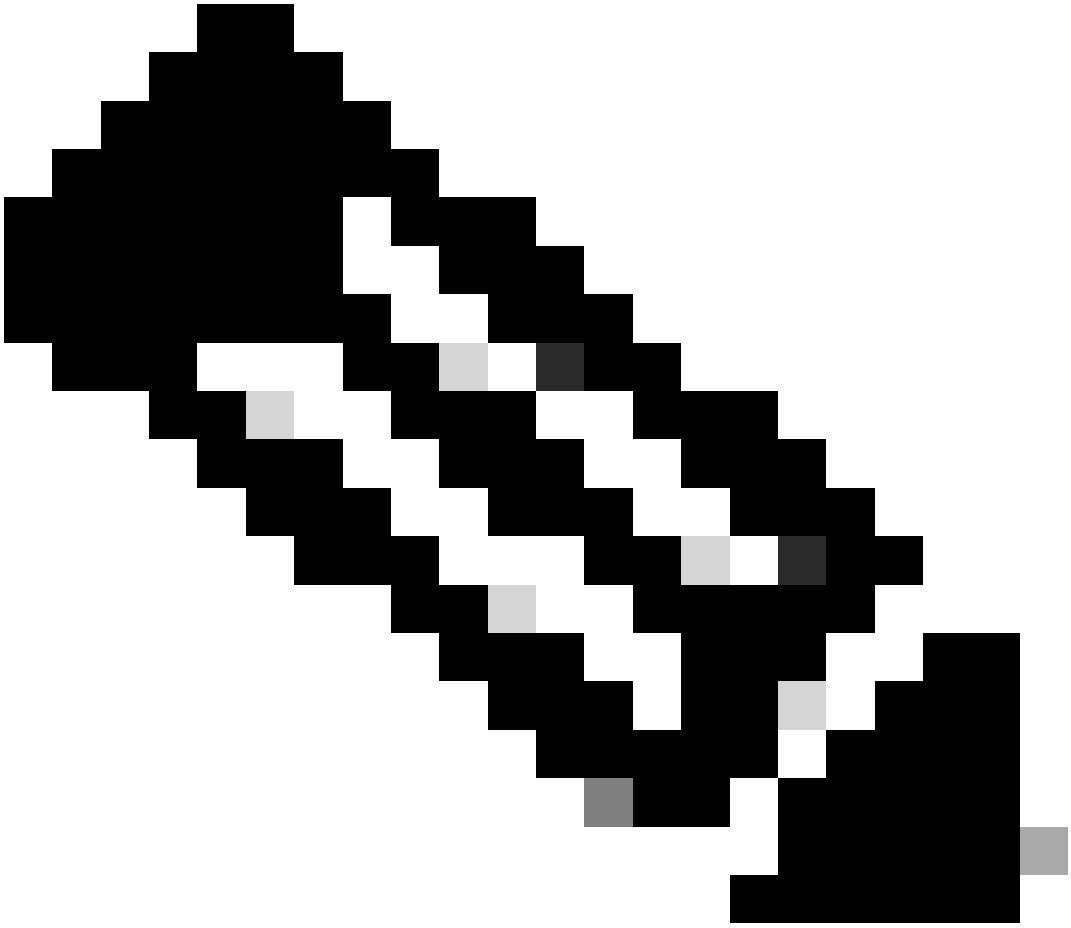


| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ISAKMP | 210 | Identity Protection (Main Mode) |
| 2 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ISAKMP | 150 | Identity Protection (Main Mode) |
| 3 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ISAKMP | 478 | Identity Protection (Main Mode) |
| 4 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ISAKMP | 498 | Identity Protection (Main Mode) |
| 5 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ISAKMP | 150 | Identity Protection (Main Mode) |
| 6 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ISAKMP | 134 | Identity Protection (Main Mode) |
| 7 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ISAKMP | 230 | Quick Mode |
| 8 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ISAKMP | 230 | Quick Mode |
| 9 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ISAKMP | 118 | Quick Mode |
| 10 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ESP | 218 | ESP (SPI=0x33a95845) |
| 11 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ICMP | 70 | Destination unreachable (Communication administratively filtered) |
| 12 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ESP | 218 | ESP (SPI=0x33a95845) |
| 13 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ICMP | 70 | Destination unreachable (Communication administratively filtered) |
| 14 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ESP | 186 | ESP (SPI=0x33a95845) |
| 15 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ESP | 186 | ESP (SPI=0x33a95845) |
| 16 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ICMP | 70 | Destination unreachable (Communication administratively filtered) |
| 17 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ESP | 218 | ESP (SPI=0x33a95845) |
| 18 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ICMP | 70 | Destination unreachable (Communication administratively filtered) |
| 19 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ESP | 186 | ESP (SPI=0x33a95845) |
| 20 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ICMP | 70 | Destination unreachable (Communication administratively filtered) |
| 21 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ESP | 186 | ESP (SPI=0x33a95845) |
| 22 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ICMP | 70 | Destination unreachable (Communication administratively filtered) |
| 23 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ESP | 218 | ESP (SPI=0x33a95845) |
| 24 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ICMP | 70 | Destination unreachable (Communication administratively filtered) |
| 25 | 1969-12-31 18:00:00.000000 | 172.22.200.2 | 172.21.100.1 | ESP | 218 | ESP (SPI=0x33a95845) |
| 26 | 1969-12-31 18:00:00.000000 | 172.21.100.1 | 172.22.200.2 | ICMP | 70 | Destination unreachable (Communication administratively filtered) |

Scan Again

Wireshark에서 출력 캡처

ICMP 패킷의 내용에는 Destination unreachable(Communication administratively filtered)이라는 오류 메시지가 있습니다. 이는 경로를 따라 트래픽에 영향을 주는 라우터 ACL 또는 방화벽과 같은 일종의 필터가 있음을 나타냅니다. 대부분의 경우, 필터는 패킷을 전송하는 디바이스(이 경우 Spoke1)에 구성되지만 중간 디바이스에서도 전송할 수 있습니다.



참고: Wireshark 출력은 두 스포크에서 동일합니다.

---

Cisco IOS® XE Datapath 패킷 추적 기능

Cisco IOS XE 데이터 경로 패킷 추적 기능은 디바이스에서 트래픽을 처리하는 방법을 분석하는 데 사용됩니다. 이를 구성하려면 두 트래픽 흐름(인바운드 및 아웃바운드) 모두에서 캡처할 트래픽을 포함한 액세스 목록을 생성해야 합니다.

이 시나리오에서는 NBMA IP 주소가 사용됩니다.

```
ip access-list extended filter
10 permit ip host 172.21.100.1 host 172.22.200.2
20 permit ip host 172.22.200.2 host 172.21.100.1
```

그런 다음 fia-trace 기능을 구성하고 access-list를 사용하도록 디버깅 조건을 설정합니다. 마지막으

로, 조건을 시작합니다.

```
debug platform packet-trace packet 1024 fia-trace
debug platform condition ipv4 access-list filter both
debug platform condition start
```

- debug platform packet-trace packet <count> fia-trace: 자세한 fia 추적을 활성화하여 구성된 패킷의 양이 캡처되면 중지합니다.
- debug platform condition ipv4 access-list <ACL-NAME> both: 이전에 구성된 access-list를 사용하여 디바이스에 조건을 설정합니다.
- 디버그 플랫폼 조건 시작: 조건을 시작합니다.

fia-trace의 출력을 검토하려면 다음 명령을 사용합니다.

```
show platform packet-trace statistics
show platform packet-trace summary
show platform packet-trace packet <number>
```

Spoke1은 플랫폼 패킷 추적 통계 출력을 표시합니다.

<#root>

```
SPOKE1#show platform packet-trace statistics
Packets Summary
  Matched  18
  Traced   18
Packets Received
  Ingress  11
  Inject   7
    Count       Code  Cause
    4           2     QFP destination lookup
    3           9     QFP ICMP generated packet
Packets Processed
  Forward  7
  Punt     8
    Count       Code  Cause
    5           11    For-us data
    3           26    QFP ICMP generated packet
  Drop     3
    Count       Code  Cause
    3           8     Ipv4Acl
  Consume  0
```

```
          PKT_DIR_IN
             Dropped       Consumed       Forwarded
INFRA              0              0              0
TCP                0              0              0
UDP                0              0              5
IP                 0              0              5
IPV6               0              0              0
ARP                0              0              0

          PKT_DIR_OUT
             Dropped       Consumed       Forwarded
INFRA              0              0              0
TCP                0              0              0
UDP                0              0              0
IP                 0              0              0
IPV6               0              0              0
ARP                0              0              0
```

show platform packet-trace statistics 출력에서 디바이스에서 처리한 패킷의 카운터를 볼 수 있습니다. 이렇게 하면 인바운드 및 아웃바운드 패킷을 확인하고, 삭제 이유와 함께 디바이스에서 패킷을 삭제하고 있는지 확인할 수 있습니다.

표시된 출력에서 Spoke1은 Ipv4Acl 설명이 포함된 일부 패킷을 삭제합니다. 이러한 패킷을 추가로 분석하기 위해 명령 show platform packet-trace summary를 사용할 수 있습니다.

Spoke1 show platform packet-trace summary 출력:

<#root>

```
SPOKE1#show platform packet-trace summary
Pkt    Input                     Output                    State  Reason
0      Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
1      INJ.2                     Gi1                       FWD
2      Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
3      INJ.2                     Gi1                       FWD
4      Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
5      INJ.2                     Gi1                       FWD
6      Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
7      INJ.2                     Gi1                       FWD
8      Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)

9      Gi1                       Gi1                       DROP   8   (Ipv4Acl)


10     Gi1                       internal0/0/recycle:0     PUNT   26  (QFP ICMP generated packet)
11     INJ.9                     Gi1                       FWD

12     Gi1                       Gi1                       DROP   8   (Ipv4Acl)


13     Gi1                       internal0/0/recycle:0     PUNT   26  (QFP ICMP generated packet)
14     INJ.9                     Gi1                       FWD

15     Gi1                       Gi1                       DROP   8   (Ipv4Acl)


16     Gi1                       internal0/0/recycle:0     PUNT   26  (QFP ICMP generated packet)
```

```
17    INJ.9                    Gi1                      FWD
18    Gi1                      Gi1                      DROP   8   (Ipv4Acl)
19    Gi1                      internal0/0/recycle:0    PUNT   26  (QFP ICMP generated packet)
20    INJ.9                    Gi1                      FWD
21    Gi1                      Gi1                      DROP   8   (Ipv4Acl)
22    Gi1                      internal0/0/recycle:0    PUNT   26  (QFP ICMP generated packet)
23    INJ.9                    Gi1                      FWD
24    Gi1                      Gi1                      DROP   8   (Ipv4Acl)
25    Gi1                      internal0/0/recycle:0    PUNT   26  (QFP ICMP generated packet)
26    INJ.9                    Gi1                      FWD
```

이 출력에서는 인그레스 및 이그레스 인터페이스뿐 아니라 디바이스에 도착하고 나가는 각 패킷을 확인할 수 있습니다. 패킷의 상태도 표시되어 패킷이 전달되었는지, 삭제되었는지, 아니면 내부적으로 처리되었는지를 나타냅니다(punt).

이 예에서 이 출력은 디바이스에서 삭제하는 패킷을 식별하는 데 도움이 됩니다. show platform packet-trace packet <PACKET_NUMBER> 명령을 사용하여 디바이스가 특정 패킷을 처리하는 방법을 확인할 수 있습니다.

Spoke1 show platform packet-trace packet <PACKET_NUMBER> 출력:

<#root>

```
SPOKE1#show platform packet-trace packet 9
Packet: 9 CBUG ID: 9
Summary
Input : GigabitEthernet1

Output : GigabitEthernet1

State : DROP 8 (Ipv4Acl)

Timestamp
Start : 366032715676920 ns (02/01/2024 04:30:15.708990 UTC)
Stop : 366032715714128 ns (02/01/2024 04:30:15.709027 UTC)
Path Trace
  Feature: IPV4(Input)

Input : GigabitEthernet1

     Output : <unknown>
```

**Source : 172.22.200.2**

**Destination : 172.21.100.1**

**Protocol : 50 (ESP)**

Feature: DEBUG_COND_INPUT_PKT
Entry : Input - 0x812707d0

**Input : GigabitEthernet1**

**Output : <unknown>**

Lapsed time : 194 ns
Feature: IPV4_INPUT_DST_LOOKUP_ISSUE
Entry : Input - 0x8129bf74

**Input : GigabitEthernet1**

**Output : <unknown>**

Lapsed time : 769 ns
Feature: IPV4_INPUT_ARL_SANITY
Entry : Input - 0x812725cc

**Input : GigabitEthernet1**

**Output : <unknown>**

Lapsed time : 307 ns
Feature: EPC_INGRESS_FEATURE_ENABLE
Entry : Input - 0x812782d0

**Input : GigabitEthernet1**

**Output : <unknown>**

Lapsed time : 6613 ns
Feature: IPV4_INPUT_DST_LOOKUP_CONSUME
Entry : Input - 0x8129bf70

**Input : GigabitEthernet1**

**Output : <unknown>**

Lapsed time : 272 ns
Feature: STILE_LEGACY_DROP
Entry : Input - 0x812a7650

**Input : GigabitEthernet1**

**Output : <unknown>**

Lapsed time : 278 ns
Feature: INGRESS_MMA_LOOKUP_DROP
Entry : Input - 0x812a1278

**Input : GigabitEthernet1**

**Output : <unknown>**

Lapsed time : 697 ns
Feature: INPUT_DROP_FNF_AOR
Entry : Input - 0x81297278

**Input : GigabitEthernet1**

**Output : <unknown>**

Lapsed time : 676 ns
Feature: INPUT_FNF_DROP
Entry : Input - 0x81280f24

**Input : GigabitEthernet1**

**Output : <unknown>**

Lapsed time : 1018 ns
Feature: INPUT_DROP_FNF_AOR_RELEASE
Entry : Input - 0x81297274

**Input : GigabitEthernet1**

**Output : <unknown>**

Lapsed time : 174 ns
**Feature: INPUT_DROP**

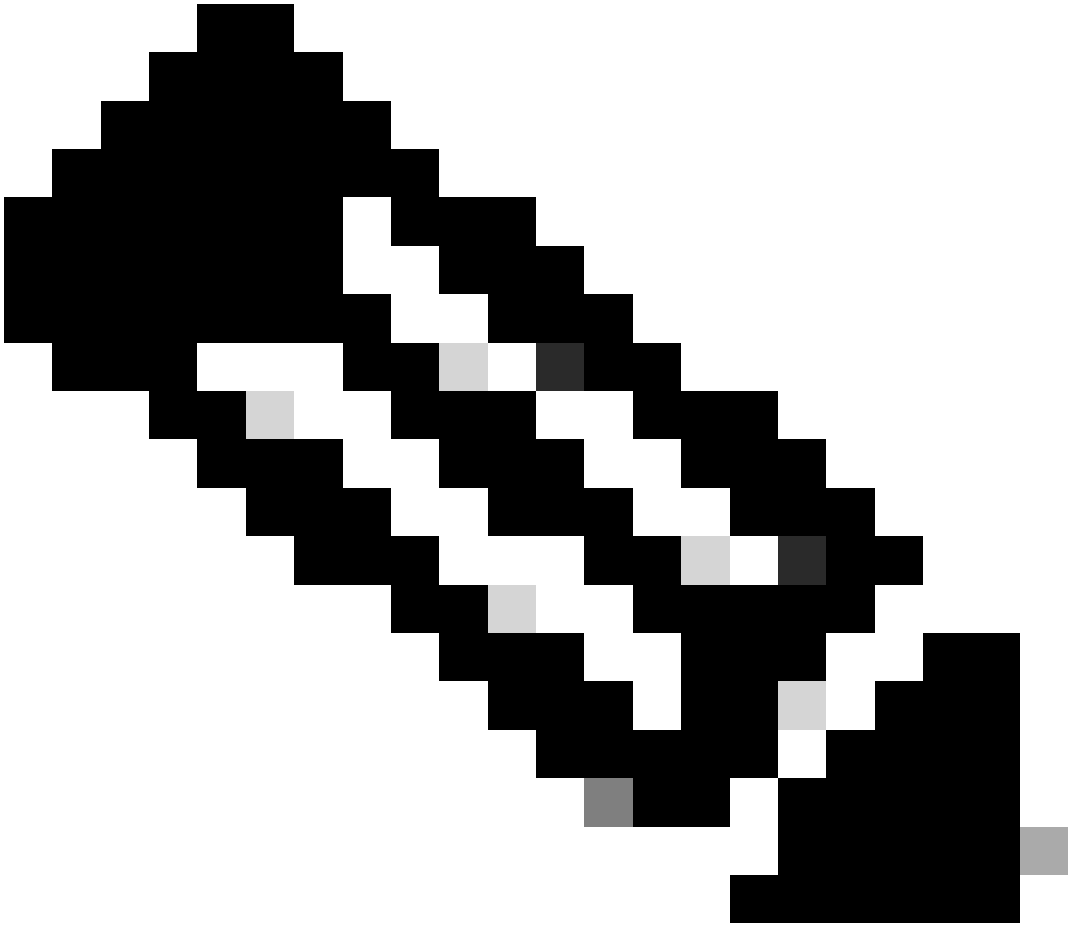
Entry : Input - 0x8126e568

```
Input : GigabitEthernet1

     Output : <unknown>

     Lapsed time : 116 ns

Feature: IPV4_INPUT_ACL

     Entry : Input - 0x81271f70

Input : GigabitEthernet1

     Output : <unknown>

     Lapsed time : 12915 ns
```

첫 번째 부분에서는 인그레스 및 이그레스 인터페이스 및 패킷의 상태를 볼 수 있습니다. 그 다음에는 소스 및 목적지 IP 주소와 프로토콜을 찾을 수 있는 출력의 두 번째 부분이 나옵니다.

각 후속 단계에서는 디바이스가 이 특정 패킷을 처리하는 방법을 보여줍니다. NAT(Network Address Translation), 액세스 목록 등의 컨피그레이션 또는 NAT에 영향을 미칠 수 있는 기타 요인에 대한 통찰력을 제공합니다.

이 경우, 패킷의 프로토콜은 ESP이고, 소스 IP는 Spoke2의 NBMA IP 주소이며, 목적지 IP는 Spoke1의 NBMA IP 주소임을 확인할 수 있다. 이는 NHRP 협상에서 누락된 패킷임을 나타냅니다. 또한 어떤 단계에서도 이그레스 인터페이스가 지정되지 않은 것으로 관찰되어 트래픽이 전달되기 전에 어떤 영향을 받았음을 시사합니다. 마지막 단계에서 디바이스에서 지정된 인터페이스(GigabitEthernet1)의 인바운드 트래픽을 삭제하는 것을 확인할 수 있습니다. 마지막 단계에서는 입력 access-list를 보여주는데, 이는 인터페이스에 어떤 컨피그레이션으로 인해 삭제가 발생할 수 있음을 시사합니다.



참고: 이 문서에 나열된 모든 트러블슈팅 툴을 사용한 후 협상에 관여한 스포크가 트래픽을 삭제하거나 영향을 주고 있다는 징후를 표시하지 않으면 해당 디바이스에서 트러블슈팅이 종료됩니다.

다음 단계는 방화벽, 스위치, ISP와 같은 중간 디바이스를 확인하는 것입니다.

## 솔루션

이러한 시나리오가 표시되면 다음 단계는 이전 출력에 표시된 인터페이스를 확인하는 것입니다. 여기에는 트래픽에 영향을 주는 것이 있는지 확인하기 위한 컨피그레이션 확인이 포함됩니다.

WAN 인터페이스 구성:

<#root>

```
SPOKE1#show running-configuration interface gigabitEthernet1
Building configuration...
```

```
Current configuration : 150 bytes
!
interface GigabitEthernet1
ip address 172.21.100.1 255.255.255.0
ip access-group ESP_TRAFFIC in
negotiation auto
no mop enabled
no mop sysid
end
```

컨피그레이션의 일부로 인터페이스에 액세스 그룹이 적용되었습니다. 액세스 목록에 구성된 호스트가 NHRP 협상에 사용되는 트래픽을 방해하지 않는지 확인해야 합니다.

<#root>

```
SPOKE1#show access-lists ESP_TRAFFIC
Extended IP access list ESP_TRAFFIC
10 deny esp host 172.21.100.1 host 172.22.200.2
20 deny esp host 172.22.200.2 host 172.21.100.1 (114 matches)
30 permit ip any any (22748 matches)
```

액세스 목록의 두 번째 문은 Spoke2의 NBMA IP 주소와 Spoke1의 NBMA IP 주소 간의 통신을 거부하므로 이전에 표시되었던 오류가 발생합니다. 인터페이스에서 액세스 그룹을 제거한 후 두 스포크 간의 통신에 성공했습니다.

```
SPOKE1#ping 192.168.2.2 source loopback1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.2.2, timeout is 2 seconds:
Packet sent with a source address of 192.168.1.1
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/2/3 ms
```

IPSEC 터널이 작동하고 있으며 두 디바이스에서 캡슐과 디캡을 표시합니다.

스포크1:

<#root>

```
SPOKE1#show crypto IPSEC sa peer 172.22.200.2

interface: Tunnel10
    Crypto map tag: Tunnel10-head-0, local addr 172.21.100.1

   protected vrf: (none)
```

```
   local  ident (addr/mask/prot/port): (172.21.100.1/255.255.255.255/47/0)
   remote ident (addr/mask/prot/port): (172.22.200.2/255.255.255.255/47/0)
   current_peer 172.22.200.2 port 500
     PERMIT, flags={origin_is_acl,}


#pkts encaps: 6, #pkts encrypt: 6, #pkts digest: 6



    #pkts decaps: 7, #pkts decrypt: 7, #pkts verify: 7


    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 0, #recv errors 0

     local crypto endpt.: 172.21.100.1, remote crypto endpt.: 172.22.200.2
     plaintext mtu 1458, path mtu 1500, ip mtu 1500, ip mtu idb GigabitEthernet1
     current outbound spi: 0x9392DA81(2475874945)
     PFS (Y/N): N, DH group: none

     inbound esp sas:
      spi: 0xBF8F523D(3213840957)
        transform: esp-256-aes esp-sha256-hmac ,
        in use settings ={Transport, }
        conn id: 2073, flow_id: CSR:73, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
         sa timing: remaining key lifetime (k/sec): (4607998/28783)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:
      spi: 0x9392DA81(2475874945)
        transform: esp-256-aes esp-sha256-hmac ,
        in use settings ={Transport, }
        conn id: 2074, flow_id: CSR:74, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
         sa timing: remaining key lifetime (k/sec): (4607999/28783)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     outbound ah sas:

     outbound pcp sas:
```

스포크2:

<#root>

```
SPOKE2#show crypto IPSEC sa peer 172.21.100.1

interface: Tunnel10
    Crypto map tag: Tunnel10-head-0, local addr 172.22.200.2
```

```
   protected vrf: (none)
   local  ident (addr/mask/prot/port): (172.22.200.2/255.255.255.255/47/0)
   remote ident (addr/mask/prot/port): (172.21.100.1/255.255.255.255/47/0)
   current_peer 172.21.100.1 port 500
     PERMIT, flags={origin_is_acl,}


#pkts encaps: 7, #pkts encrypt: 7, #pkts digest: 7



    #pkts decaps: 6, #pkts decrypt: 6, #pkts verify: 6


    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 0, #recv errors 0

     local crypto endpt.: 172.22.200.2, remote crypto endpt.: 172.21.100.1
     plaintext mtu 1458, path mtu 1500, ip mtu 1500, ip mtu idb GigabitEthernet1
     current outbound spi: 0xBF8F523D(3213840957)
     PFS (Y/N): N, DH group: none

     inbound esp sas:
      spi: 0x9392DA81(2475874945)
        transform: esp-256-aes esp-sha256-hmac ,
        in use settings ={Transport, }
        conn id: 2073, flow_id: CSR:73, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
         sa timing: remaining key lifetime (k/sec): (4607998/28783)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:
      spi: 0xBF8F523D(3213840957)
        transform: esp-256-aes esp-sha256-hmac ,
        in use settings ={Transport, }
        conn id: 2074, flow_id: CSR:74, sibling_flags FFFFFFFF80000008, crypto map: Tunnel10-head-0
         sa timing: remaining key lifetime (k/sec): (4607999/28783)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     outbound ah sas:

     outbound pcp sas:
```

이제 Spoke1의 DMVPN 테이블에 두 항목에 대한 올바른 매핑이 표시됩니다.

<#root>

```
SPOKE1#show dmvpn
```

```
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
        N - NATed, L - Local, X - No Socket
        T1 - Route Installed, T2 - Nexthop-override, B - BGP
        C - CTS Capable, I2 - Temporary
        # Ent --> Number of NHRP entries with same NBMA peer
        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
        UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface: Tunnel10, IPv4 NHRP Details
Type:Spoke, NHRP Peers:2,

# Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb
----- --------------- --------------- ----- -------- -----

1 172.22.200.2     10.10.10.2    UP   00:01:31   D

     1 172.20.10.10     10.10.10.10    UP    1d05h     S
```

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.