

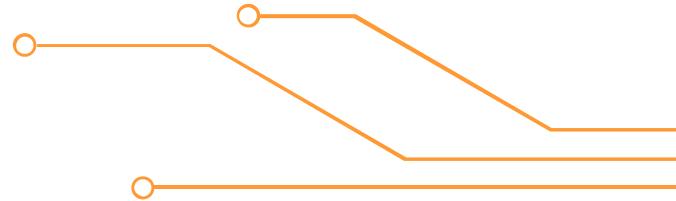


Network & System Defense

Progetto 2

- Jacopo Fabi 0293870
- Michele Salvatori 0306362
- Danilo Dell'Orco 0300229

Roadmap



01 Introduzione

Descrizione Topologia,
Architettura e Componenti

03 OpenVPN

Configurazione e Test Host,
Server e AS200

05 MACsec

Configurazione MACsec e
MACsecKA nella LAN-A1



02 BGP/MPLS VPN

Configurazione e Test Intra-AS
VPN in AS100

04 Firewalls

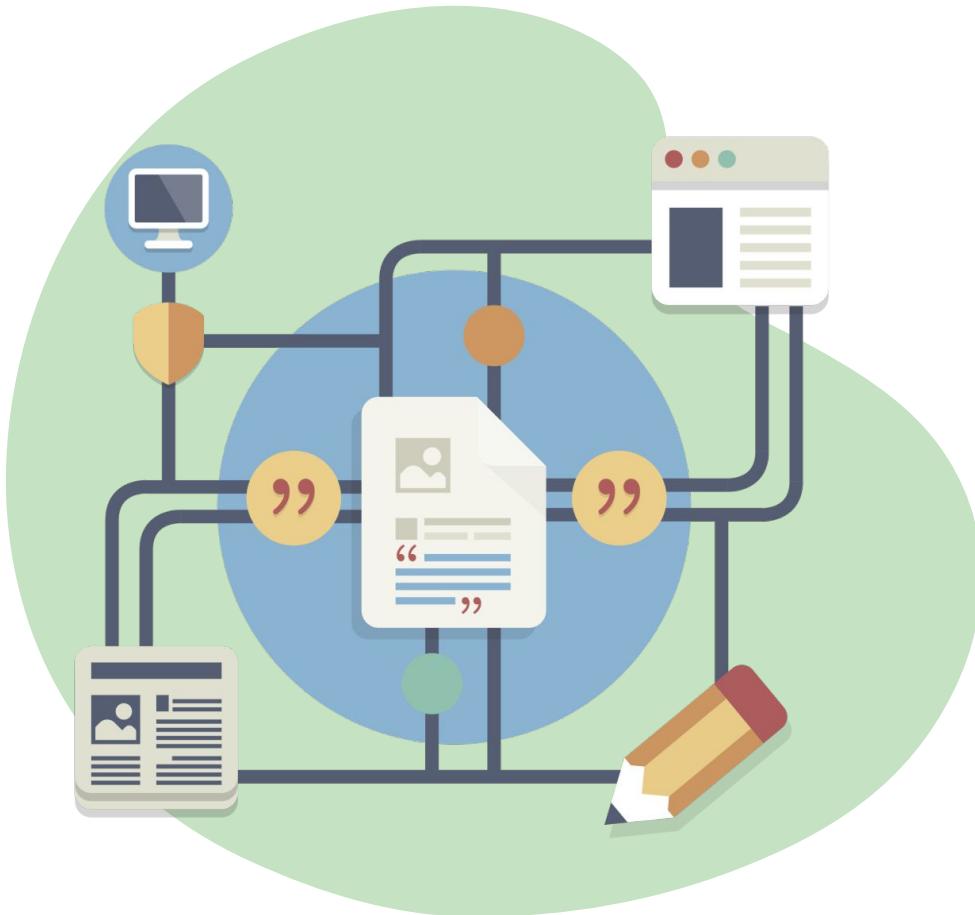
Configurazione Firewall dei
Gateways in LAN-A1 e LAN-A2

06 Scansione Antivirus

Scansione remota sui nodi test,
ed invio report su LAN-A1

...





01

INTRODUZIONE

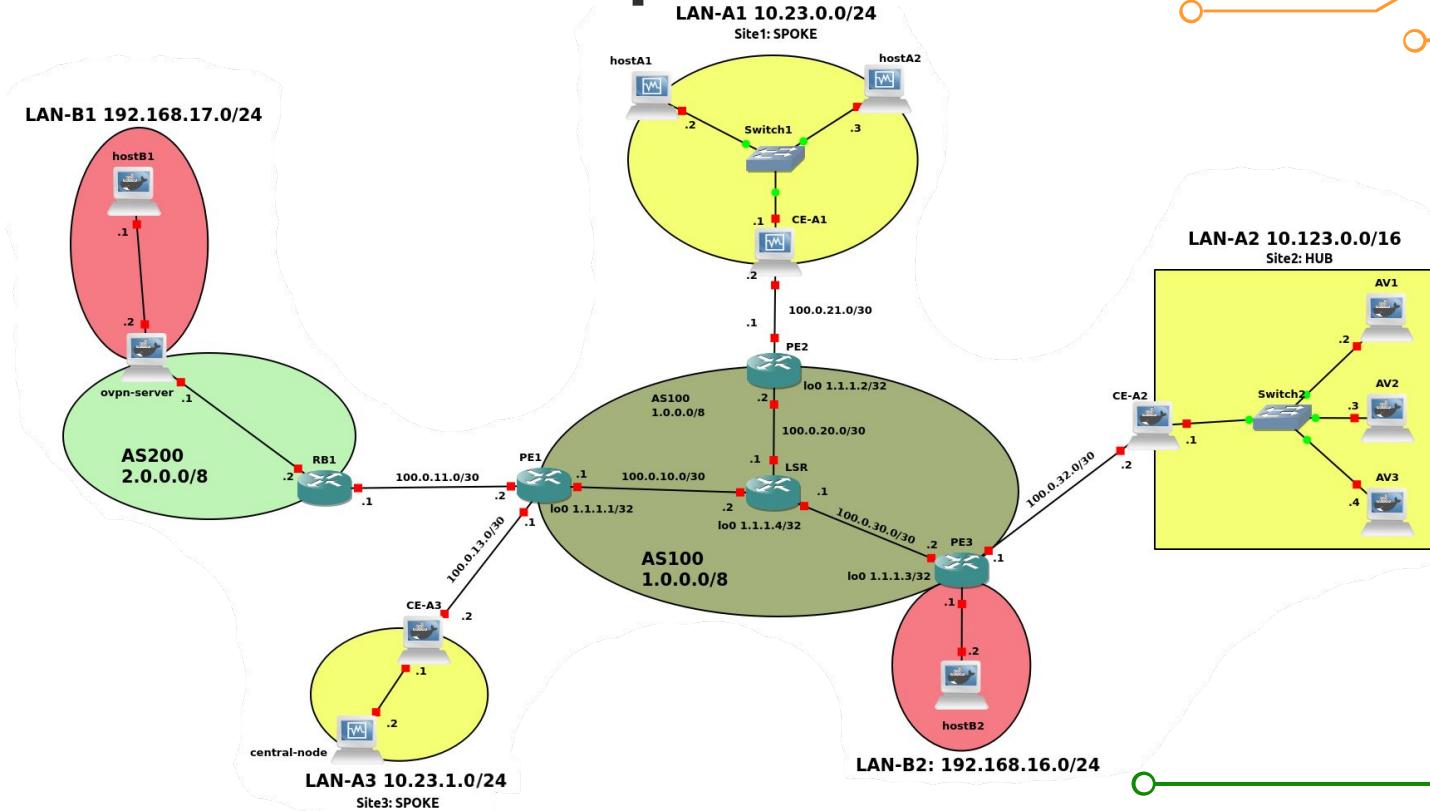
Network Overview

Specifica

In questo progetto, dovete creare un ambiente virtualizzato per testare i binari e verificare la presenza di malware. In particolare, dovete configurare più macchine virtuali o container nel sito 2, ognuno dei quali ospiterà un diverso AV di vostra scelta.

- Un central-node (nel sito 3) consentirà di scaricare un eseguibile e di distribuirlo ai vari nodi di test (nel sito 2) analizzeranno/eseguiranno gli eseguibili e forniranno i risultati al nodo centrale, che creerà un rapporto per l'utente, mostrando quali minacce (se presenti) sono state scoperte nel binario.
- Per quanto riguarda la configurazione di rete, due Sistemi Autonomi forniscono connettività a cinque reti private.
 - AS100 offre un servizio di VPN BGP/MPLS per i tre siti di VPNA.
 - AS200 è un customer di AS100 e hosta un server OpenVPN con un indirizzo IP pubblico, usato per offrire una overlay VPN per il client VPN in LAN-B1.

Specifica





Immagini Docker Utilizzate

- Tutte le immagini docker utilizzate sono state create a partire da [weibeld/ubuntu-networking](#),
- In aggiunta sono stati installati software di base, per la configurazione e gestione dei servizi
 - `telnet`, `python3`, `unzip`.



sixxpain/openvpn

Hosta client e server di OpenVPN.
- openvpn
- easyRSA



sixxpain/nubuntu

Utilizzato per i customer edge
CE-A2 e CE-A3, che non
implementano MACsec.



sixxpain/clamav

Mantiene il servizio di analisi e
report di AV1.
- clamav



sixxpain/jmdav

Mantiene il servizio di analisi e
report di AV2.
- jmdav.py
- pyelftools
- lief



sixxpain/rkhav

Mantiene il servizio di analisi e
report di AV3.
- rkhunter



02

BGP/MPLS VPN

Configurazione e Test Intra-AS VPN in AS100

...

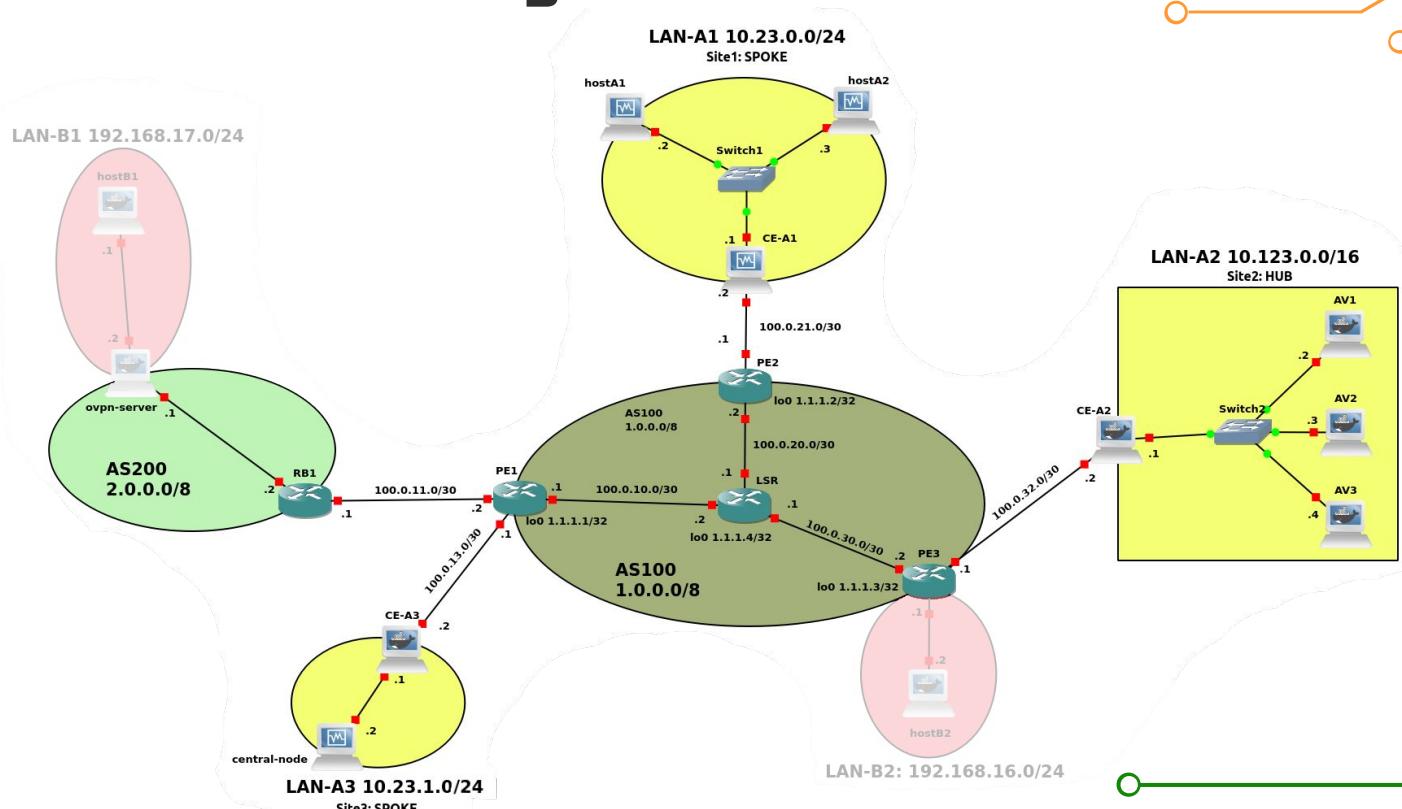
Specifica

Il sistema autonomo AS100 offre connettività ad AS200 e mette a disposizione una BGP/MPLS VPN ad un customer.

Per la realizzazione, è necessario:

- Configurare OSPF in AS100 per permettere ai routers di scambiare le rotte interne al sistema autonomo.
- Configurare BGP sui PE in AS100 e su RB1 in AS200 per abilitare lo scambio di rotte verso destinazioni esterne tramite eBGP, e distribuirle in AS100 tramite iBGP.
- Configurare le tabelle VRF e la loro sincronizzazione per realizzare la BGP/MPLS VPN con topologia Hub-and-Spoke.

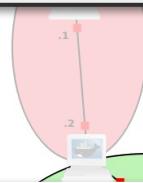
Configurazione di Rete



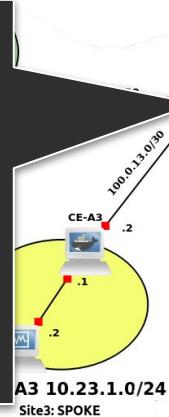
Configurazione di Rete

mpls-ip: abilita l'uso di MPLS su una precisa interfaccia.

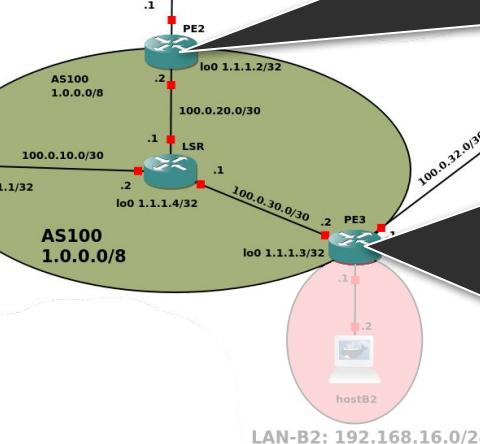
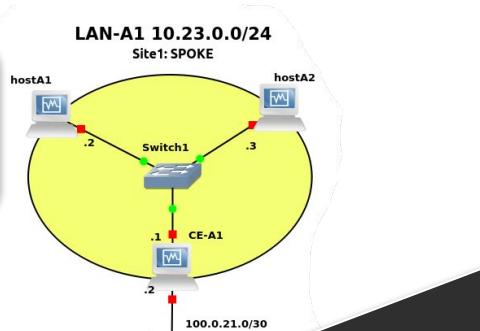
ip vrf forwarding vpnA: abilita il forwarding dei pacchetti per la BGP/MPLS VPN sull'interfaccia del PE connessa al CE.



```
interface Loopback0
 ip address 1.1.1.1 255.255.255.255
!
interface g1/0
 ip address 100.0.10.1 255.255.255.252
 mpls ip
 no shutdown
!
interface g2/0
 ip address 100.0.11.2 255.255.255.252
 no shutdown
!
```



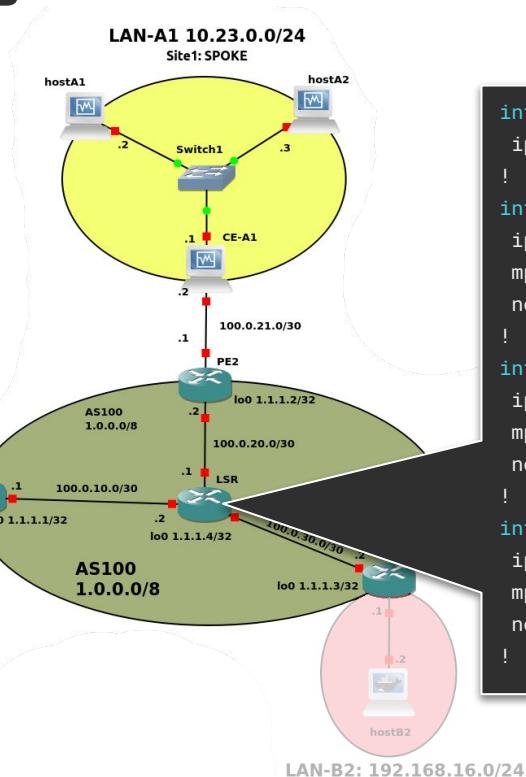
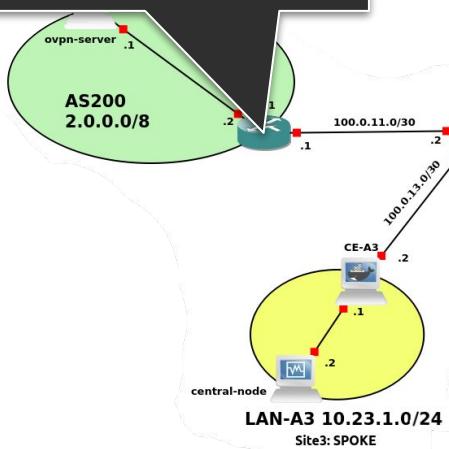
A3 10.23.1.0/24
Site3: SPOKE



```
interface Loopback0
 ip address 1.1.1.2 255.255.255.255
!
interface g1/0
 ip address 100.0.20.2 255.255.255.252
 mpls ip
 no shutdown
!
interface g2/0
 ip vrf forwarding vpnA
 ip address 100.0.21.1 255.255.255.252
 no shutdown
!
interface Loopback0
 ip address 1.1.1.3 255.255.255.255
!
interface g1/0
 ip address 100.0.30.2 255.255.255.252
 mpls ip
 no shutdown
!
interface g2/0
 ip vrf forwarding vpnA
 ip address 100.0.32.1 255.255.255.252
 no shutdown
!
```

Configurazione di Rete

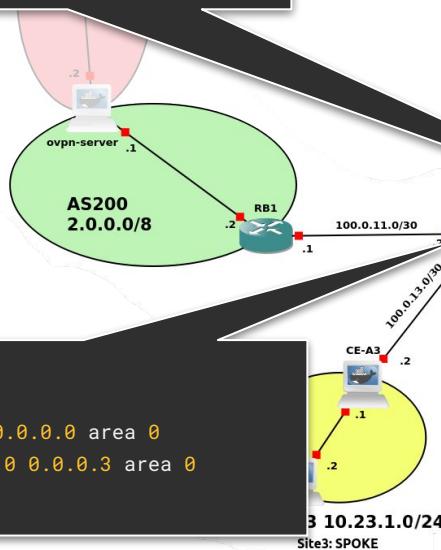
```
interface g1/0
 ip address 100.0.11.1 255.255.255.252
 no shutdown
!
interface g2/0
 ip address 2.0.0.2 255.0.0.0
 no shutdown
!
```



```
interface Loopback0
 ip address 1.1.1.4 255.255.255.255
!
interface g1/0
 ip address 100.0.10.2 255.255.255.252
 mpls ip
 no shutdown
!
interface g2/0
 ip address 100.0.20.1 255.255.255.252
 mpls ip
 no shutdown
!
interface g3/0
 ip address 100.0.30.1 255.255.255.252
 mpls ip
 no shutdown
!
```

Configurazione OSPF

```
router ospf 1
router-id 1.1.1.4
network 1.1.1.4 0.0.0.0 area 0
network 100.0.10.0 0.0.0.3 area 0
network 100.0.20.0 0.0.0.3 area 0
network 100.0.30.0 0.0.0.3 area 0
!
```



```
router ospf 1
router-id 1.1.1.1
network 1.1.1.1 0.0.0.0 area 0
network 100.0.10.0 0.0.0.3 area 0
!
```

OSPF permette ai PE di apprendere le rotte per l'instradamento interno ad AS100.

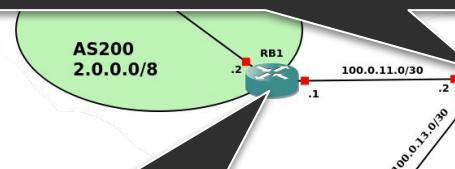
Questo protocollo viene eseguito sfruttando le interfacce di loopback, in quanto sono interfacce stabili che restano UP a prescindere dallo stato dei link fisici.

```
router ospf 1
router-id 1.1.1.2
network 1.1.1.2 0.0.0.0 area 0
network 100.0.20.0 0.0.0.3 area 0
!
```

```
router ospf 1
router-id 1.1.1.3
network 1.1.1.3 0.0.0.0 area 0
network 100.0.30.0 0.0.0.3 area 0
!
```

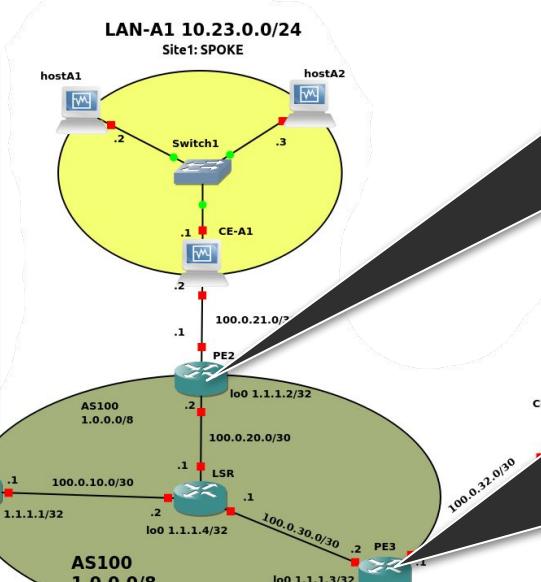
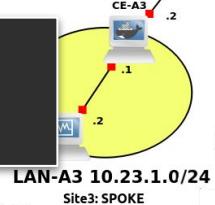
Configurazione BGP

```
router bgp 100
network 1.0.0.0
neighbor 100.0.11.1 remote-as 200
neighbor 1.1.1.2 remote-as 100
neighbor 1.1.1.2 update-source Loopback0
neighbor 1.1.1.2 next-hop-self
neighbor 1.1.1.3 remote-as 100
neighbor 1.1.1.3 update-source Loopback0
neighbor 1.1.1.3 next-hop-self
...
!
```



```
router bgp 200
network 2.0.0.0
neighbor 100.0.11.2 remote-as 100
!

```



```
router bgp 100
network 1.0.0.0
neighbor 1.1.1.1 remote-as 100
neighbor 1.1.1.1 update-source Loopback0
neighbor 1.1.1.1 next-hop-self
neighbor 1.1.1.3 remote-as 100
neighbor 1.1.1.3 update-source Loopback0
neighbor 1.1.1.3 next-hop-self
!

```

```
Site2: HUB
router bgp 100
network 1.0.0.0
neighbor 1.1.1.1 remote-as 100
neighbor 1.1.1.1 update-source Loopback0
neighbor 1.1.1.1 next-hop-self
neighbor 1.1.1.2 remote-as 100
neighbor 1.1.1.2 update-source Loopback0
neighbor 1.1.1.2 next-hop-self
!

```

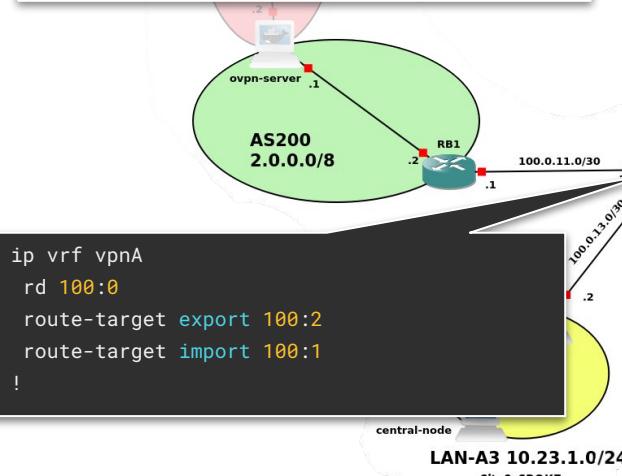
Il protocollo BGP utilizza l'interfaccia fisica PE1->RB1 per il peering eBGP con AS200, e le interfacce di loopback per iBGP

- **eBGP** è abilitato su RB1 e PE1 per scambiare rotte tra AS100 e AS200.
- **iBGP** è abilitato su tutti i PEs per propagare le rotte verso AS200 nei nodi interni della rete.

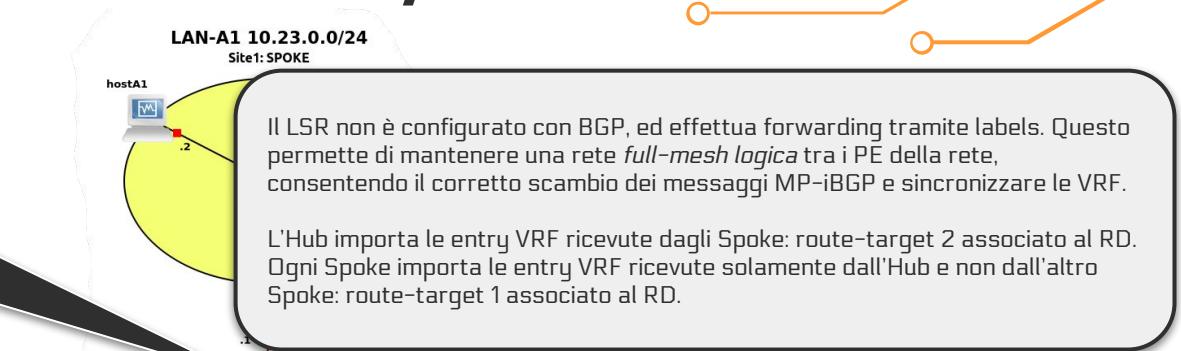
Usiamo l'interfaccia di loopback come sorgente di update BGP perché le rotte in AS100 sono note tramite questi indirizzi, appresi con OSPF.

Configurazione BGP/MPLS VPN

```
ip vrf vpnA
rd 100:0
route-target export 100:2
route-target import 100:1
!
```

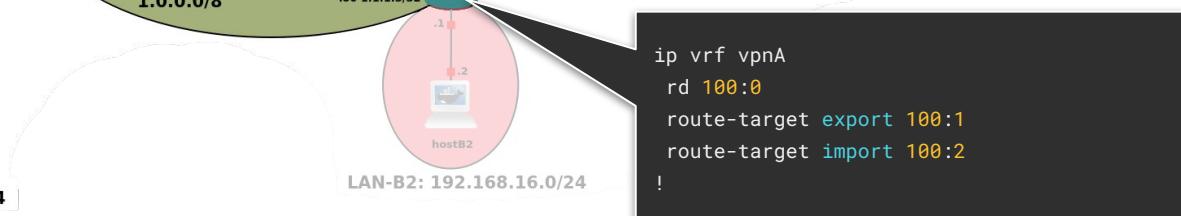


```
ip vrf vpnA
rd 100:0
route-target export 100:2
route-target import 100:1
!
```



Il LSR non è configurato con BGP, ed effettua forwarding tramite labels. Questo permette di mantenere una rete *full-mesh logica* tra i PE della rete, consentendo il corretto scambio dei messaggi MP-iBGP e sincronizzare le VRF.

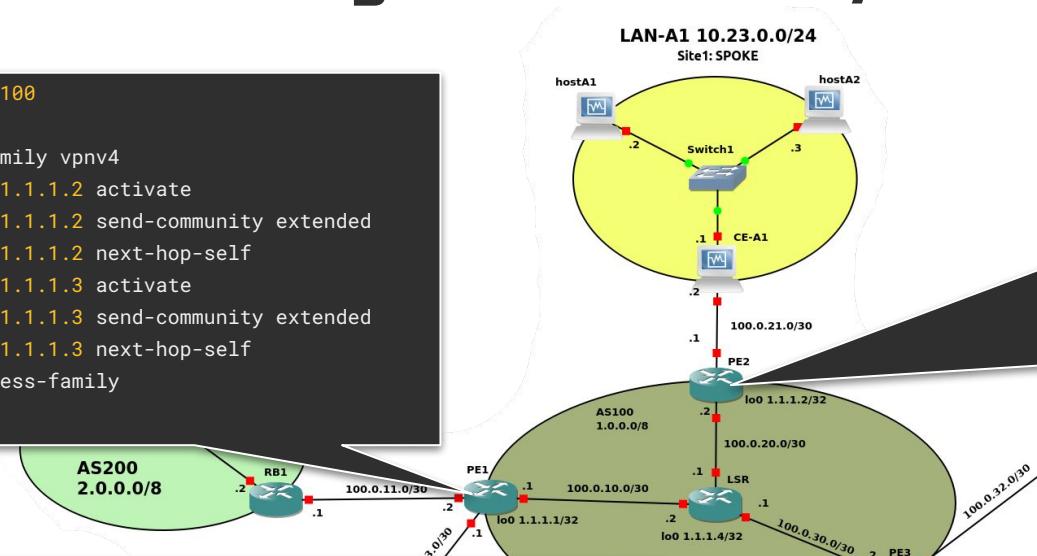
L'Hub importa le entry VRF ricevute dagli Spoke: route-target 2 associato al RD. Ogni Spoke importa le entry VRF ricevute solamente dall'Hub e non dall'altro Spoke: route-target 1 associato al RD.



```
ip vrf vpnA
rd 100:0
route-target export 100:1
route-target import 100:2
!
```

Configurazione BGP/MPLS VPN

```
router bgp 100
...
address-family vpnv4
neighbor 1.1.1.2 activate
neighbor 1.1.1.2 send-community extended
neighbor 1.1.1.2 next-hop-self
neighbor 1.1.1.3 activate
neighbor 1.1.1.3 send-community extended
neighbor 1.1.1.3 next-hop-self
exit-address-family
!
```



Sotto-configurazione di BGP per *indirizzi VPNv4* che ci permette di attivare una nuova relazione di peering tra tutti i PEs che hanno una VPN connessa.

Utilizza il *community attribute extended* per trasportare sia il RD che il RT negli annunci MP-iBGP.

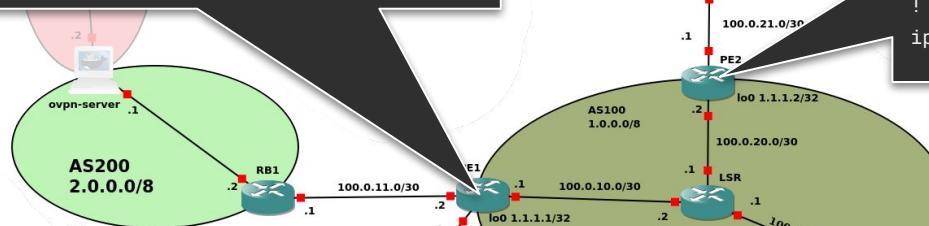
In questo modo siamo in grado di scambiare messaggi MP-iBGP tra i PEs e filtrarli per costruire la topologia hub-and-spoke.

```
router bgp 100
...
address-family vpnv4
neighbor 1.1.1.1 activate
neighbor 1.1.1.1 send-community extended
neighbor 1.1.1.1 next-hop-self
neighbor 1.1.1.3 activate
neighbor 1.1.1.3 send-community extended
neighbor 1.1.1.3 next-hop-self
exit-address-family
!
```

```
router bgp 100
...
address-family vpnv4
neighbor 1.1.1.1 activate
neighbor 1.1.1.1 send-community extended
neighbor 1.1.1.1 next-hop-self
neighbor 1.1.1.2 activate
neighbor 1.1.1.2 send-community extended
neighbor 1.1.1.2 next-hop-self
exit-address-family
!
```

Configurazione BGP/MPLS VPN

```
router bgp 100
...
address-family ipv4 vrf vpnA
network 10.23.1.0 mask 255.255.255.0
exit-address-family
!
ip route vrf vpnA 10.23.1.0 255.255.255.0 100.0.13.2
```

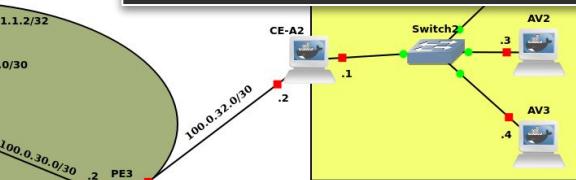


Sotto-configurazione di BGP per specificare le rotte verso i siti della VPN gestiti da un PE che devono essere esportati negli annunci MP-iBGP.

Le rotte nella VRF devono essere inserite manualmente perché non c'è un protocollo di routing sul link PE-CE, quindi bisogna specificare in che modo è possibile raggiungere il sito della VPN.

L'hub esporta anche la rotta per 10.23.0.0/16 per abilitare la comunicazione tra gli spokes.

```
router bgp 100
...
address-family ipv4 vrf vpnA
network 10.23.0.0 mask 255.255.255.0
exit-address-family
!
ip route vrf vpnA 10.23.0.0 255.255.255.0 100.0.21.2
```



```
router bgp 100
...
address-family ipv4 vrf vpnA
network 10.123.0.0 mask 255.255.0.0
network 10.23.0.0 mask 255.255.0.0
exit-address-family
!
ip route vrf vpnA 10.123.0.0 255.255.0.0 100.0.32.2
ip route vrf vpnA 10.23.0.0 255.255.0.0 100.0.32.2
```

TEST HUB-SPOKE BGP/MPLS VPN

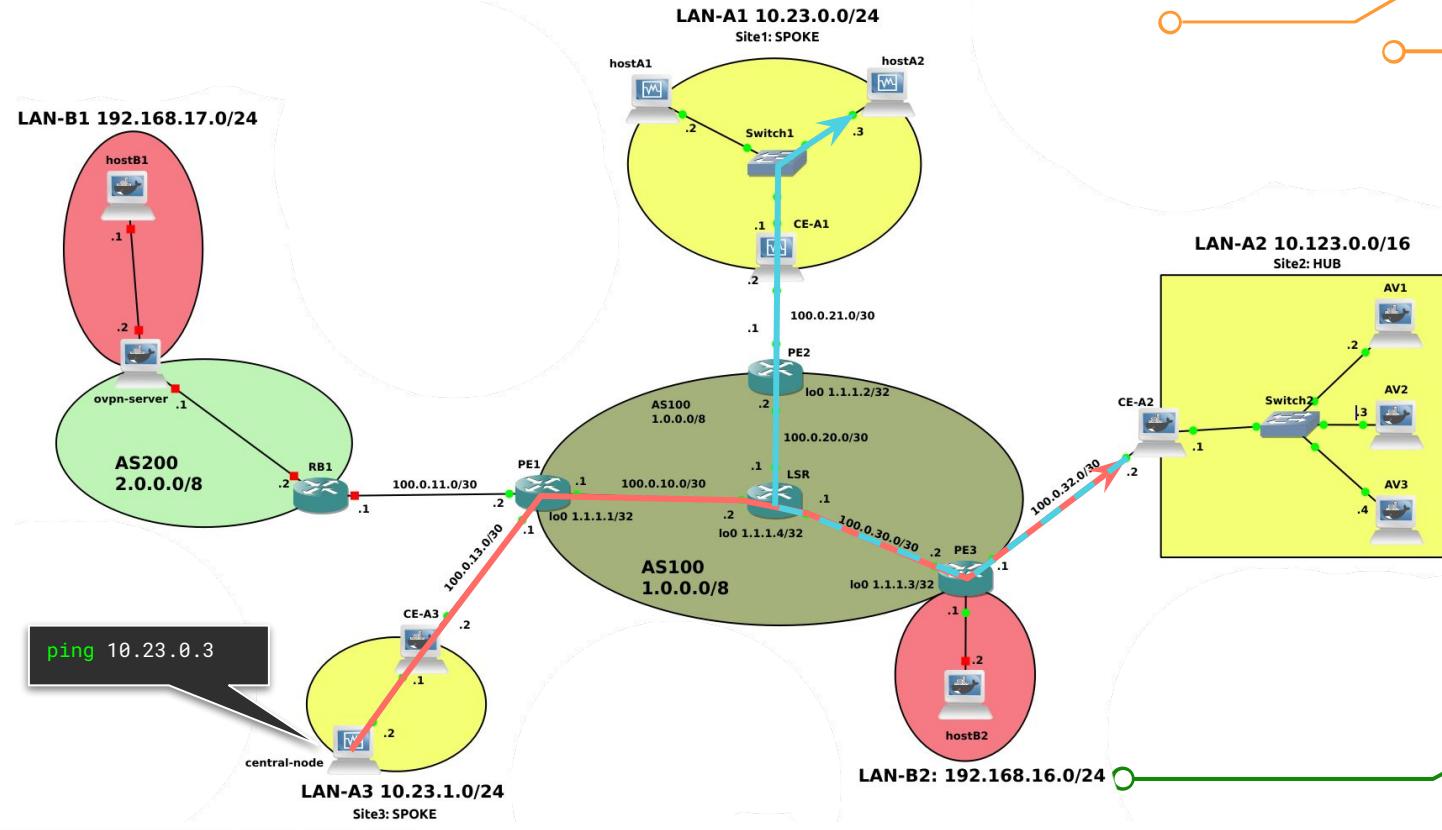
Per testare la configurazione della BGP/MPLS VPN è stato seguito il percorso preso da un pacchetto ICMP generato da **SPOKE_LAN-A3** e destinato a raggiungere **SPOKE_LAN-A2**, così da verificare l'implementazione effettiva della topologia HUB-SPOKE

TEST: SPOKE_LAN-A3-> SPOKE_LAN-A1

- Verifica encapsulamento in MACsec Frame all'ingresso della LAN-A1



TEST: SPOKE -> SPOKE



TEST: SPOKE -> SPOKE

LAN-A1 10.23.0.0/24

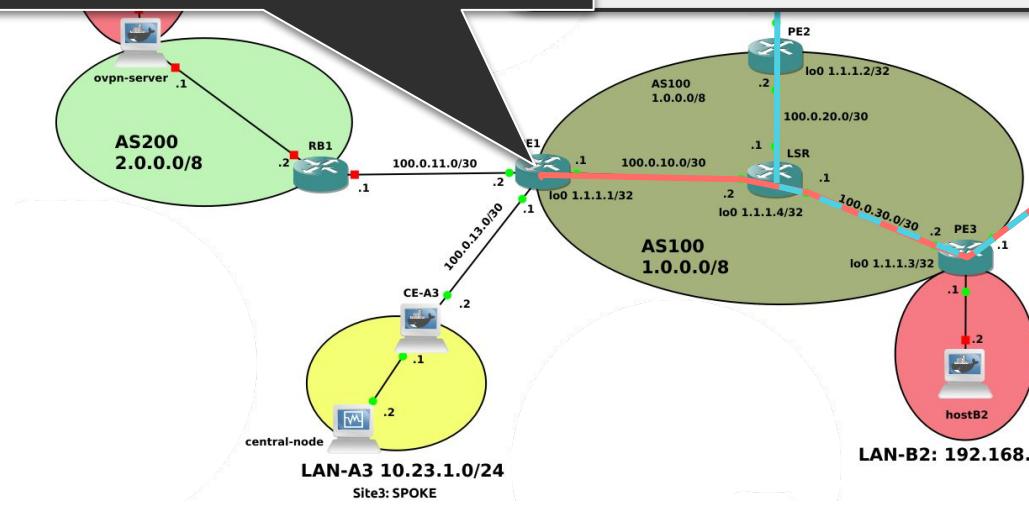
Site1: SPOKE

```
PE1# show ip route vrf vpnA
```

```
100.0.0.0/30 is subnetted, 1 subnets
C      100.0.13.0 is directly connected, GigabitEthernet3/0
10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
S        10.23.1.0/24 [1/0] via 100.0.13.2
B        10.23.0.0/24 [200/0] via 1.1.1.3, 00:05:15
B        10.123.0.0/16 [200/0] via 1.1.1.3, 00:05:15
```

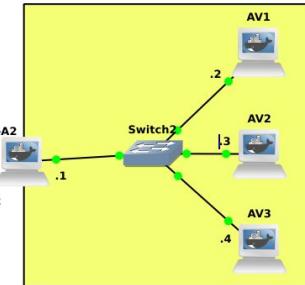
PE1 riceve un pacchetto da un'interfaccia di rete su cui è abilitato il routing attraverso VRF.

Consultando la VRF potrà determinare il next-hop per la rete 10.23.0.0/24



LAN-A2 10.123.0.0/16

Site2: HUB



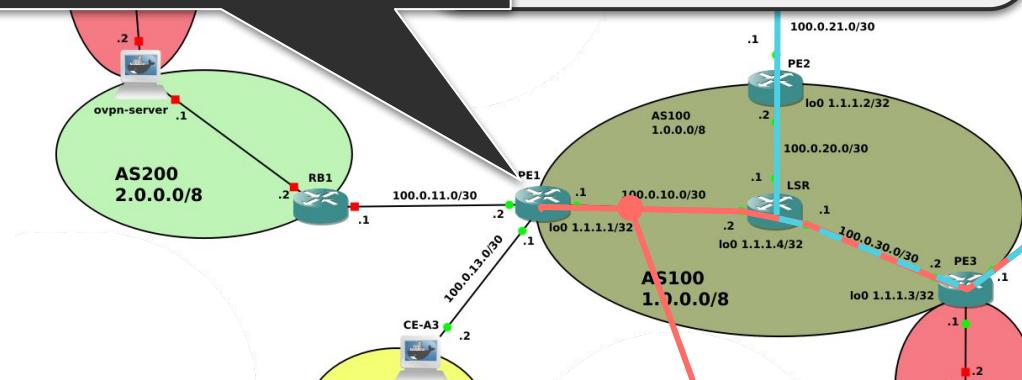
TEST: SPOKE -> SPOKE

LAN-A1 10.23.0.0/24

Site1: SPOKE

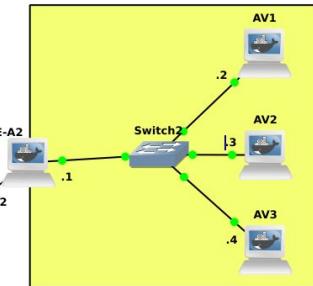
```
PE1# show ip bgp vpngv4 vrf vpnA labels
Network      Next Hop      In label/Out label
Route Distinguisher: 100:0 (vpnA)
  10.23.0.0/16    1.1.1.3        nolabel/21
  10.23.1.0/24   100.0.13.2       21/nolabel
  10.123.0.0/16   1.1.1.3        nolabel/22
```

PE1 applicherà al pacchetto in uscita verso LSR la label 21 per identificare il preciso customer-VPN destinatario



```
Frame 1026: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface -, id 0
Ethernet II, Src: ca:05:6a:59:00:1c (ca:05:6a:59:00:1c), Dst: ca:03:6a:7c:00:1c (ca:03:6a:7c:00:1c)
MultiProtocol Label Switching Header, Label: 18, Exp: 0, S: 0, TTL: 62
MultiProtocol Label Switching Header, Label: 21, Exp: 0, S: 1, TTL: 62
Internet Protocol Version 4, Src: 10.23.1.2, Dst: 10.23.0.3
Internet Control Message Protocol
```

LAN-A2 10.123.0.0/16
Site2: HUB

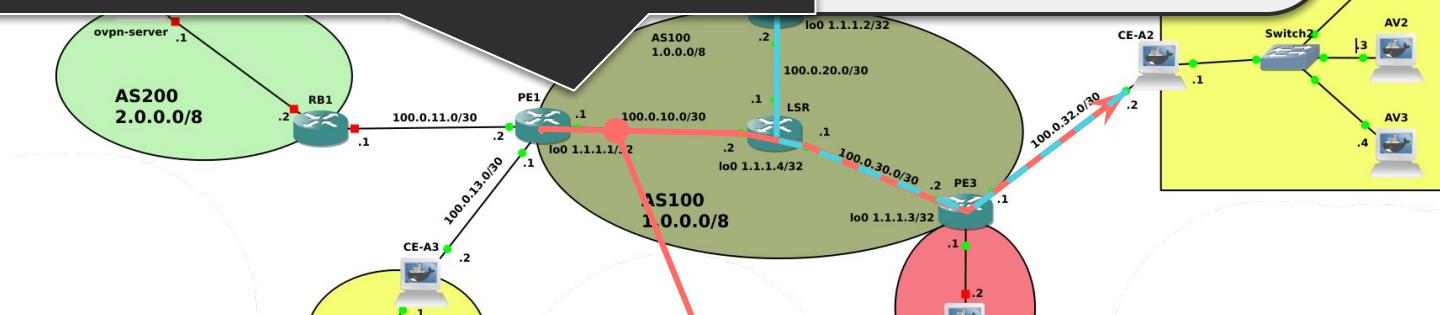


TEST: SPOKE -> SPOKE

```
PE1# show mpls forwarding-table
```

Local Label	Outgoing Label or Tunnel Id	Prefix or Tunnel Id	Bytes	Label Switched	Outgoing interface	Next Hop
16	Pop Label	1.1.1.4/32	0		Gi1/0	100.0.10.2
17	16	1.1.1.2/32	0		Gi1/0	100.0.10.2
18	Pop Label	100.0.30.0/30	0		Gi1/0	100.0.10.2
19	Pop Label	100.0.20.0/30	0		Gi1/0	100.0.10.2
20	18	1.1.1.3/32	0		Gi1/0	100.0.10.2
21	No Label	10.23.1.0/24[V]	714		Gi3/0	100.0.13.2

Per il routing all'interno della MPLS-Core, PE1 consulterà il LIB applicando la label 18



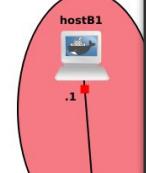
```
Frame 1026: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface -, id 0
Ethernet II. Src: ca:05:6a:59:00:1c (ca:05:6a:59:00:1c). Dst: ca:03:6a:7c:00:1c (ca:03:6a:7c:00:1c)
MultiProtocol Label Switching Header, Label: 18, Exp: 0, S: 0, TTL: 62
MultiProtocol Label Switching Header, Label: 21, Exp: 0, S: 1, TTL: 62
Internet Protocol Version 4, Src: 10.23.1.2, Dst: 10.23.0.3
Internet Control Message Protocol
```

TEST : SPOKE -> SPOKE

LAN-A1 10.23.0.0/24

Site1: SPOKE

LAN-B1 192.168.1

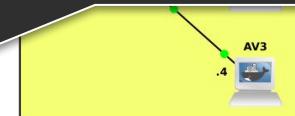
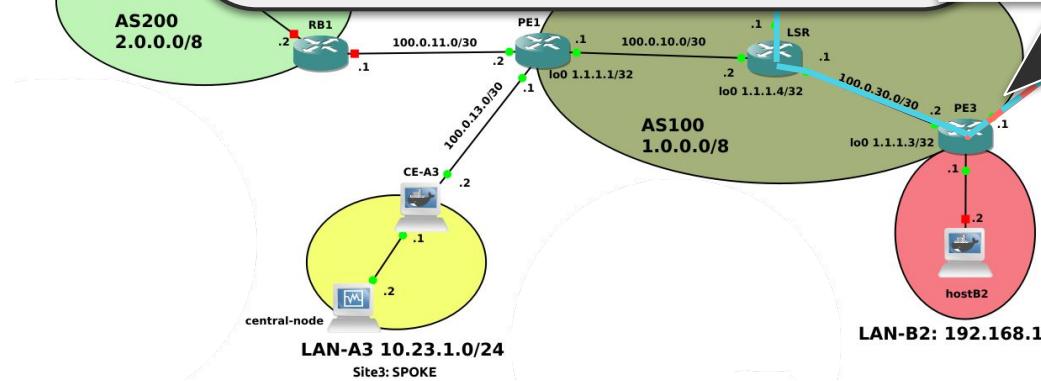


Il pacchetto ricevuto da PE3 non presenterà più l'outer-label che identifica il LSP, poiché LSR ha effettuato il label-switching. Tuttavia il pacchetto ricevuto conterrà l'inner-label [21] che identifica lo Spoke LAN-A1. Di conseguenza, consultando le entry della sua VRF, PE3 instraderà il pacchetto verso l'HUB e non direttamente verso il next-hop per raggiungere lo Spoke destinatario.

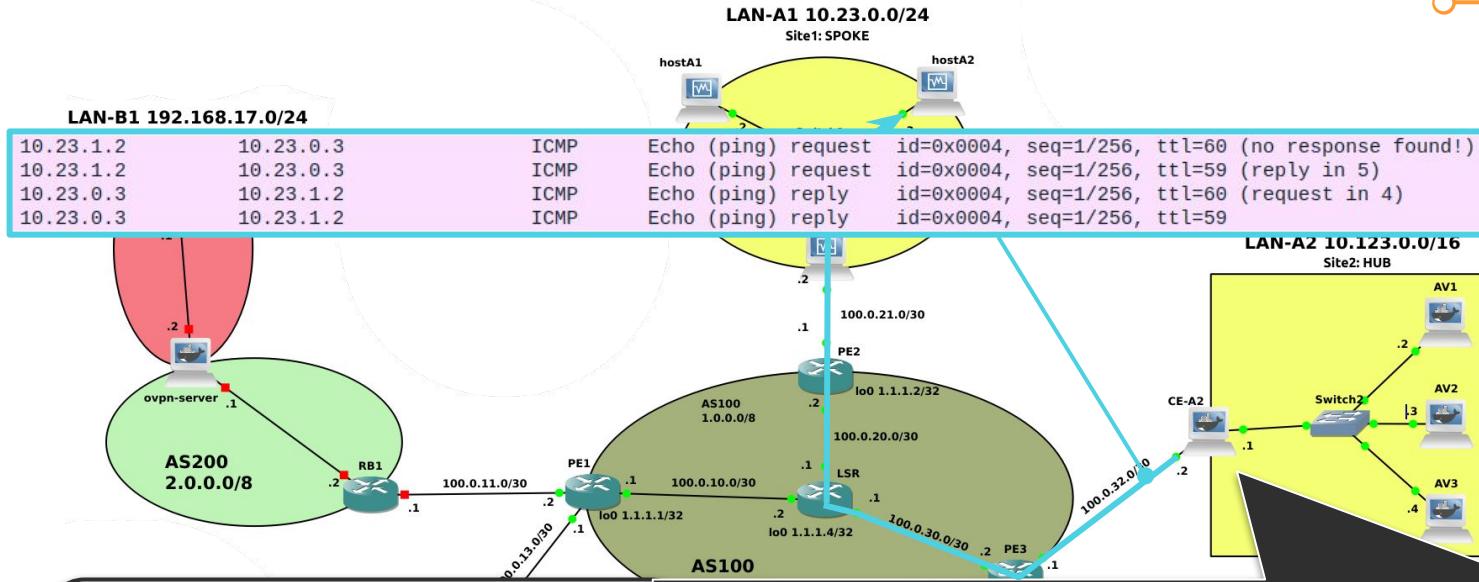
```
PE3# show ip bgp vpnv4 vrf vpnA labels
      Network          Next Hop     In label/Out label
Route Distinguisher: 100:0 (vpnA)
    10.23.0.0/16      1.1.1.2     nolabel/21
    10.23.0.0/24      100.0.32.2   21/nolabel
    10.23.1.0/24      1.1.1.1     nolabel/21
    10.123.0.0/16     100.0.32.2   22/nolabel
```

AS200
2.0.0.0/8

central-node LAN-A3 10.23.1.0/24
Site3: SPOKE



TEST: SPOKE -> SPOKE



Il Client-Edge dell'HUB si limiterà a forwardare i pacchetti ricevuti, e non destinati alla sua LAN, nuovamente verso PE3 utilizzando la sua Default Gateway Route

```
root@CE-A2:~# route
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
default	100.0.32.1	0.0.0.0	UG	0	0	0	eth0
10.123.0.0	0.0.0.0	255.255.0.0	U	0	0	0	eth1
100.0.32.0	0.0.0.0	255.255.255.252	U	0	0	0	eth0

Site3: SPOKE

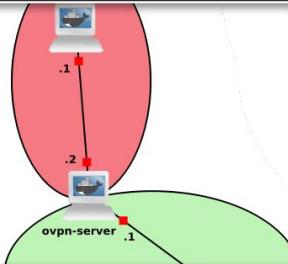
TEST: SPOKE -> SPOKE

```
root@CE-A1:/# route
```

...

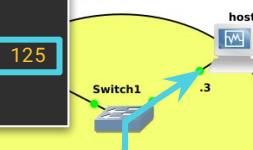
```
10.23.0.0/24 dev macsec0 proto kernel scope link src 10.23.0.1 metric 125
```

...



LAN A1 10.23.0.0/24

Site1: SPOKE



hostA2

Switch1

CE-A1

.3

100.0.21.0/30

.1

PE2

.2

AS100

1.0.0.0/8

100.0.10.0/30

.1

LSR

lo0 1.1.1.2/32

100.0.20.0/30

.1

1.1.4/32

100.0.30.0/30

.2

PE3

.1

100.0.32.0/30

.1

CE-A2

.2

Switch2

AV2

.3

AV3

.4

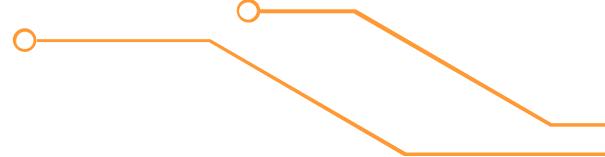
```
PE2# show ip bgp vpnv4 vrf vpnA labels
Network          Next Hop      In label/Out label
10.23.0.0/24    100.0.21.2   21/nolabel
10.23.0.0/16    1.1.1.3      nolabel/22
10.123.0.0/16   1.1.1.3      nolabel/22
```

```
PE3# show ip bgp vpnv4 vrf vpnA labels
Network          Next Hop      In label/Out label
Route Distinguisher: 100:0 (vpnA)
10.23.0.0/16     1.1.1.2      nolabel/21
10.23.0.0/24     100.0.32.2   21/nolabel
10.23.1.0/24     1.1.1.1      nolabel/21
10.123.0.0/16    100.0.32.2   22/nolabel
```

LAN-B2: 192.168.16.0/24



OPENVPN



03



OpenVPN

Configurazione tunnel VPN tramite
OpenVPN



...

Specifica

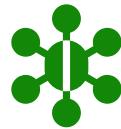
Il sistema autonomo AS200 è il customer di AS100, e mantiene un server OpenVPN, con un indirizzo IP pubblico per offrire overlay VPN per il client nella LAN-B1.

Bisogna configurare OpenVPN con un server ed un client.

- Il server si trova in AS200, con un indirizzo IP pubblico preso dalla rete 2.0.0.0/8.
- Il client è l'HostB2, posto all'interno della rete privata LAN-B2.
- Il server OpenVPN fornisce accesso verso la LAN-B1, verso cui agisce da gateway.



Configurazione Device



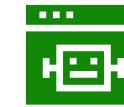
ovpn-server

- sixxpain/openvpn
- Implementa il Server OpenVPN
- Gestisce la CA, la generazione e la firma delle chiavi
- Agisce da gateway per gli host della LAN-B1



HostB1

- sixxpain/openvpn
- Implementa l'host remoto raggiungibile da HostB2 tramite la VPN
- Unaware della VPN



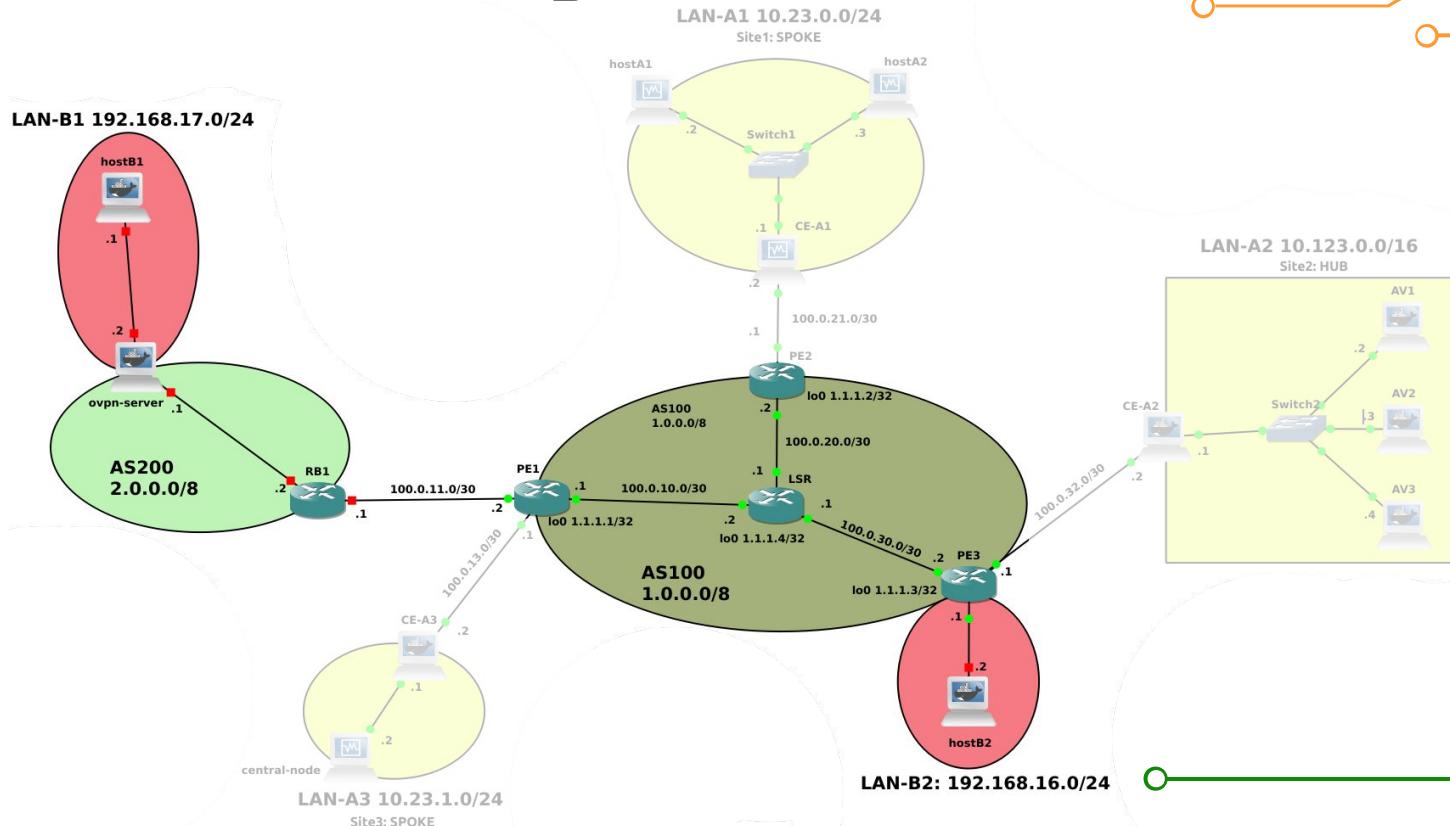
HostB2

- sixxpain/openvpn
- Implementa il Client OpenVPN
- Raggiunge l'HostB1 remoto tramite il tunnel stabilito con il server OpenVPN

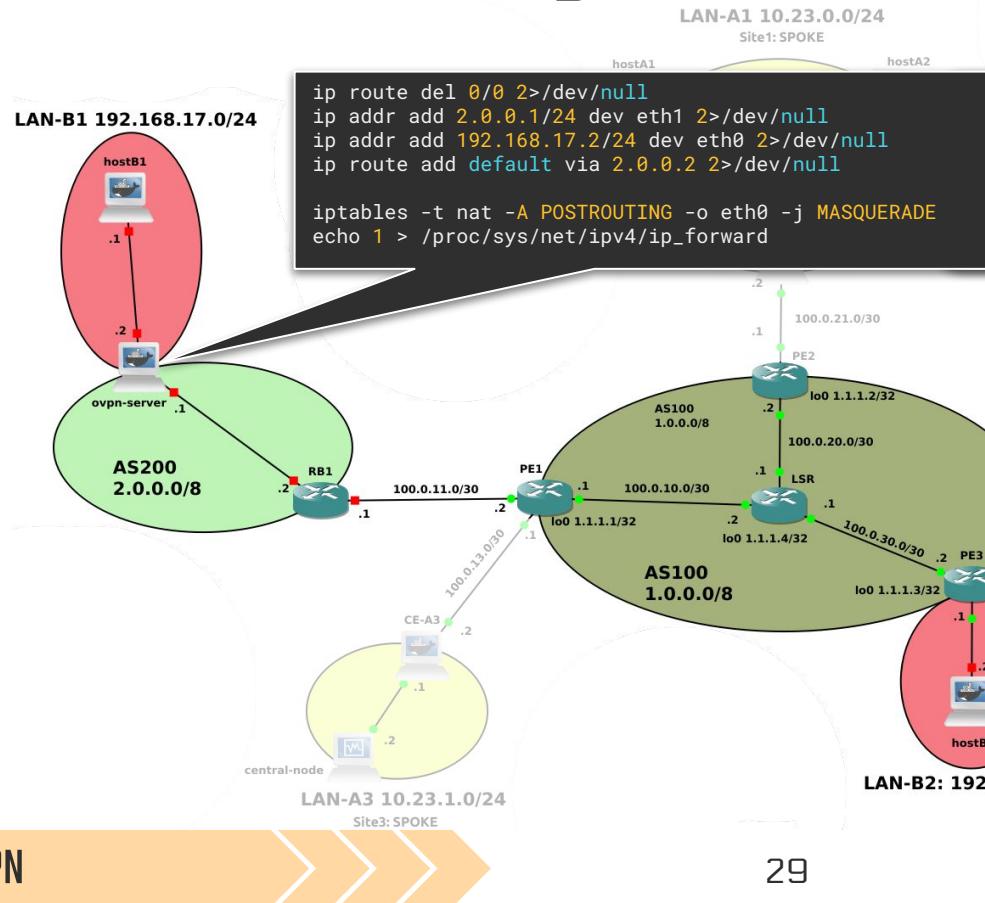
...



Configurazione di Rete

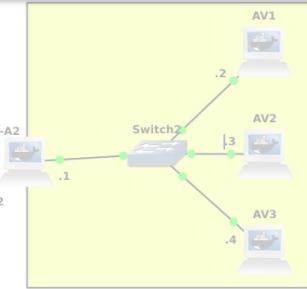


Configurazione di Rete



Abilitiamo il nat sul server, in modo che i pacchetti verso HostB1 vengano mascherati con l'interfaccia privata verso la LAN-B1.

L'HostB1 è unaware della connessione VPN, e senza NAT non potrebbe rispondere ai messaggi di HostB2

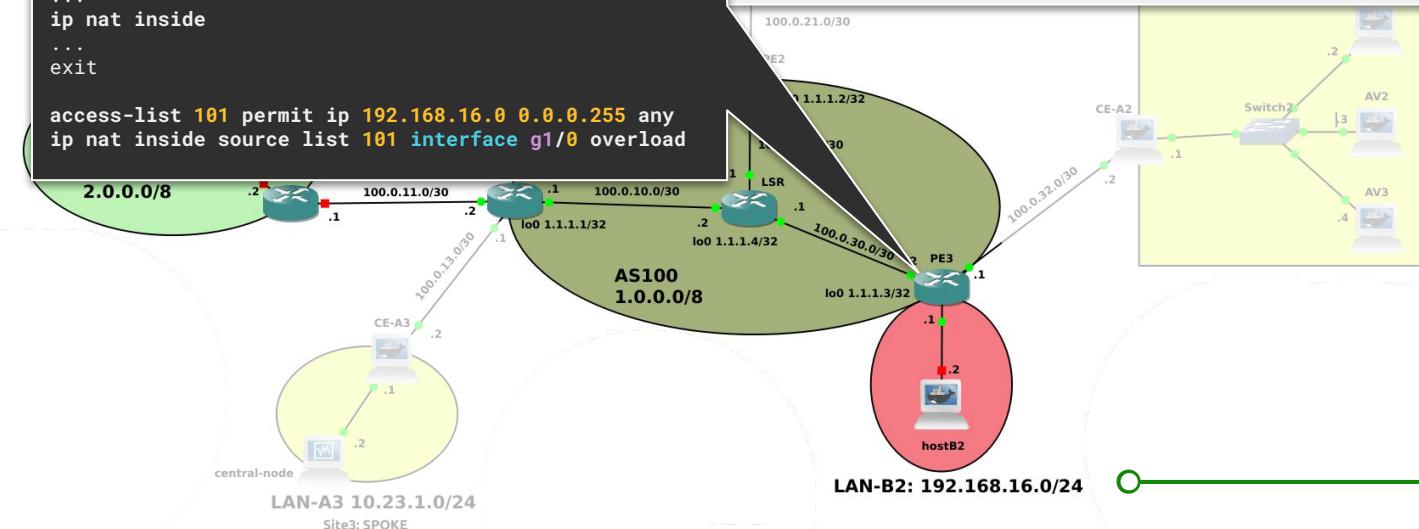


Configurazione di Rete

LAN-A1 10.23.0.0/24

Site1: SPOKE

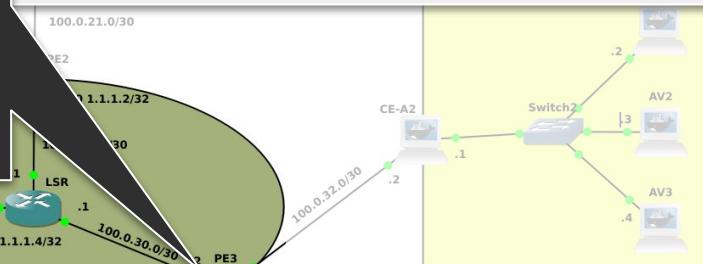
```
configure terminal  
LAN-  
interface g1/0  
...  
ip nat outside  
...  
exit  
  
interface g2/0  
...  
ip nat inside  
...  
exit  
  
access-list 101 permit ip 192.168.16.0 0.0.0.255 any  
ip nat inside source list 101 interface g1/0 overload
```



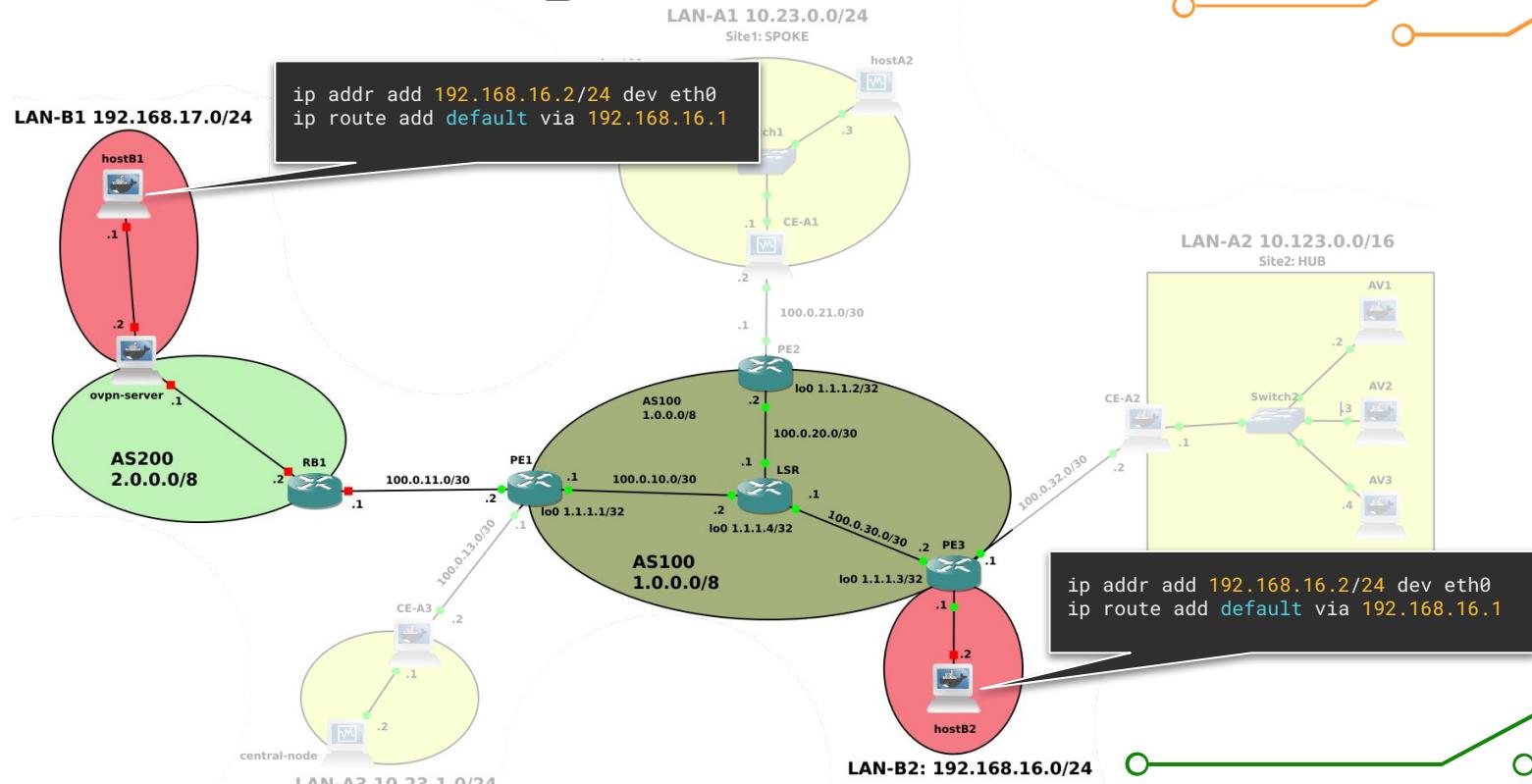
Oltre la configurazione già analizzata per BGP/MPLS, è necessario configurare il NAT su PE3

In questo modo i pacchetti verso HostB1 e ovpn-server saranno mascherati con l'interfaccia di loopback di PE3, ovvero 1.1.1.3

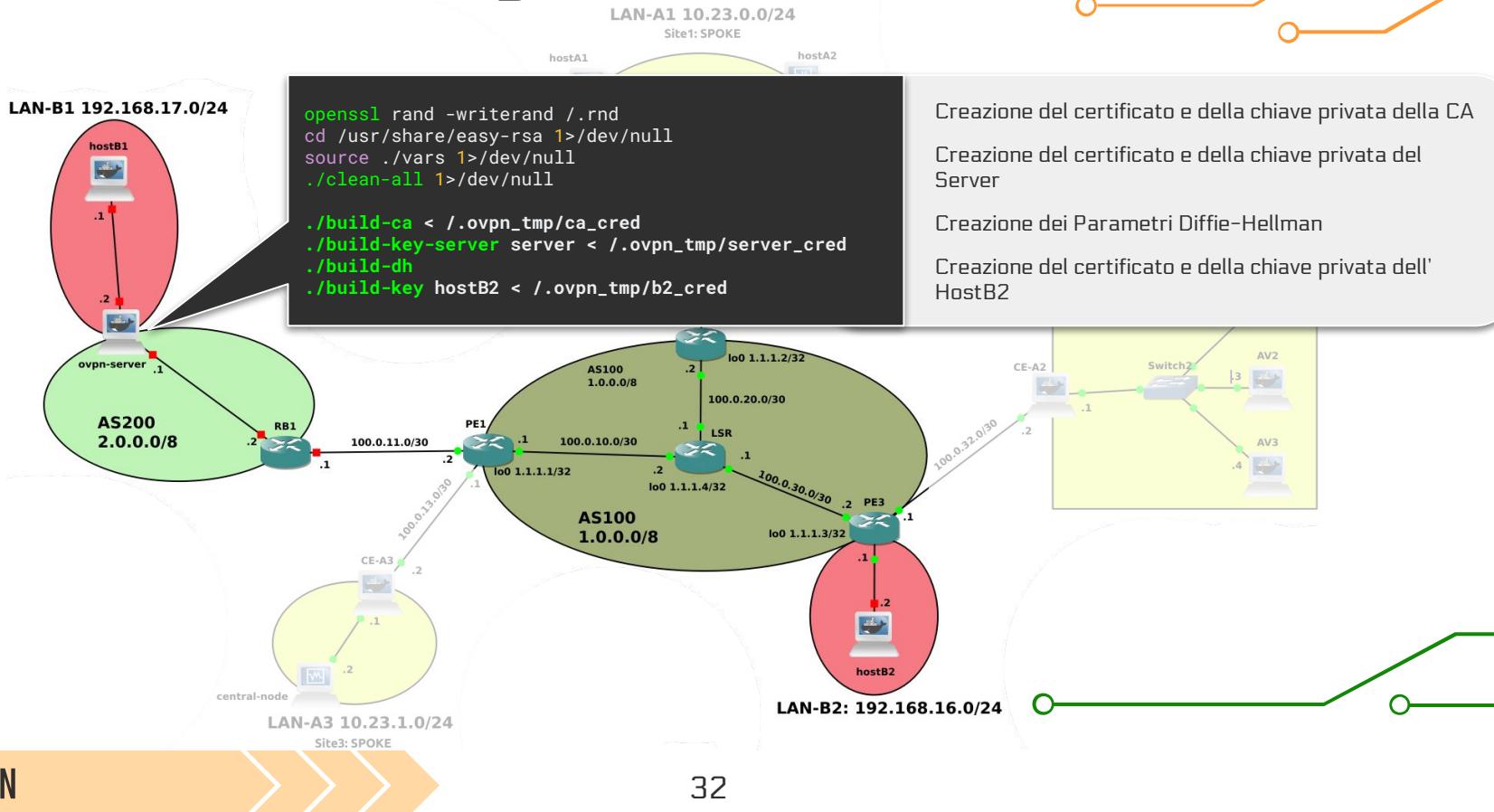
Così facendo possiamo sfruttare le rotte BGP già note per raggiungere AS200



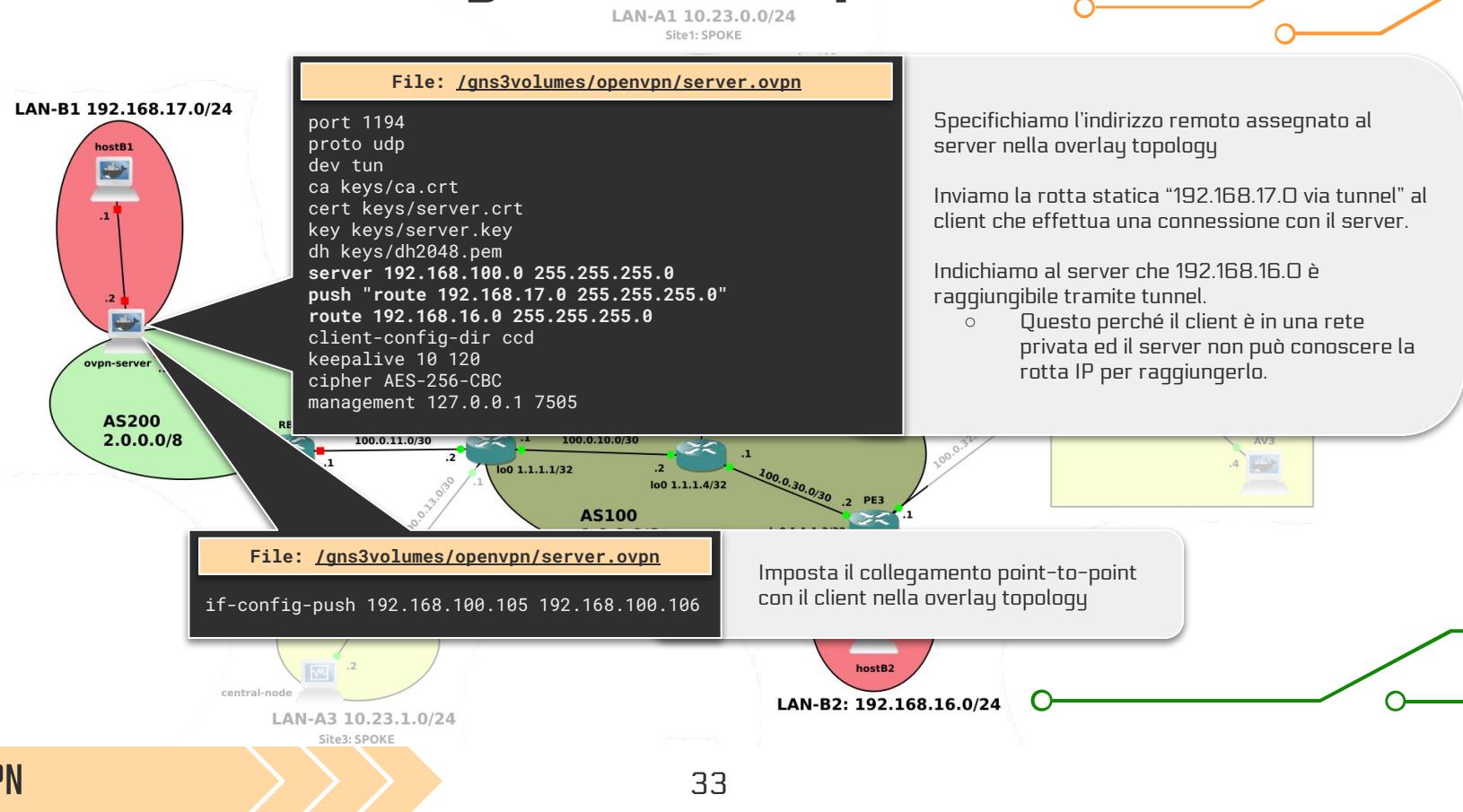
Configurazione di Rete



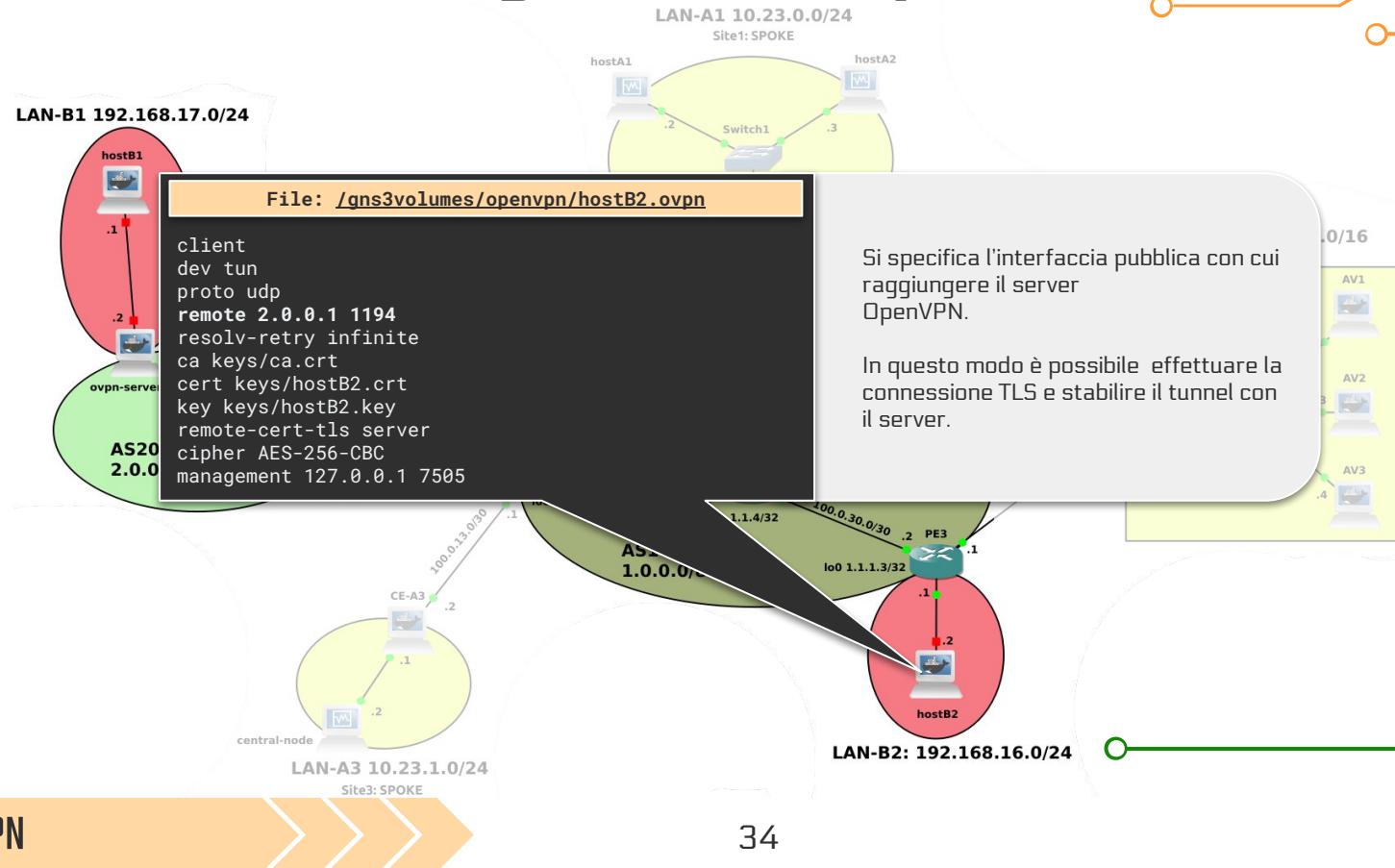
Configurazione di Rete



Configurazione OpenVPN



Configurazione OpenVPN



TEST CONFIGURAZIONE OPENVPN

Per testare la configurazione OpenVPN sono stati effettuati i seguenti test:

TEST 1: TLS Handshake HostB1-> ovpn-server

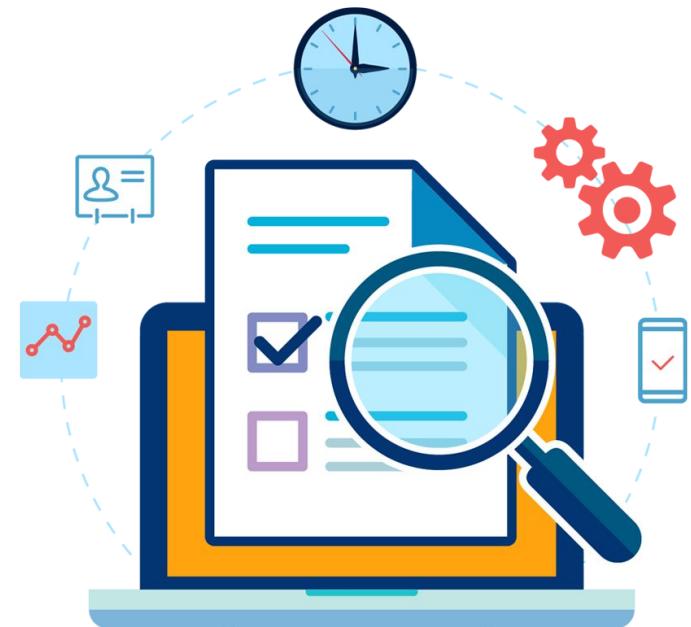
- Verifica NAT nelle richieste verso il Server
- Verifica tunnel stabilito tra Interfaccia di Loopback e Indirizzo pubblico del Server

TEST 2: Ping HostB1 -> ovpn-server

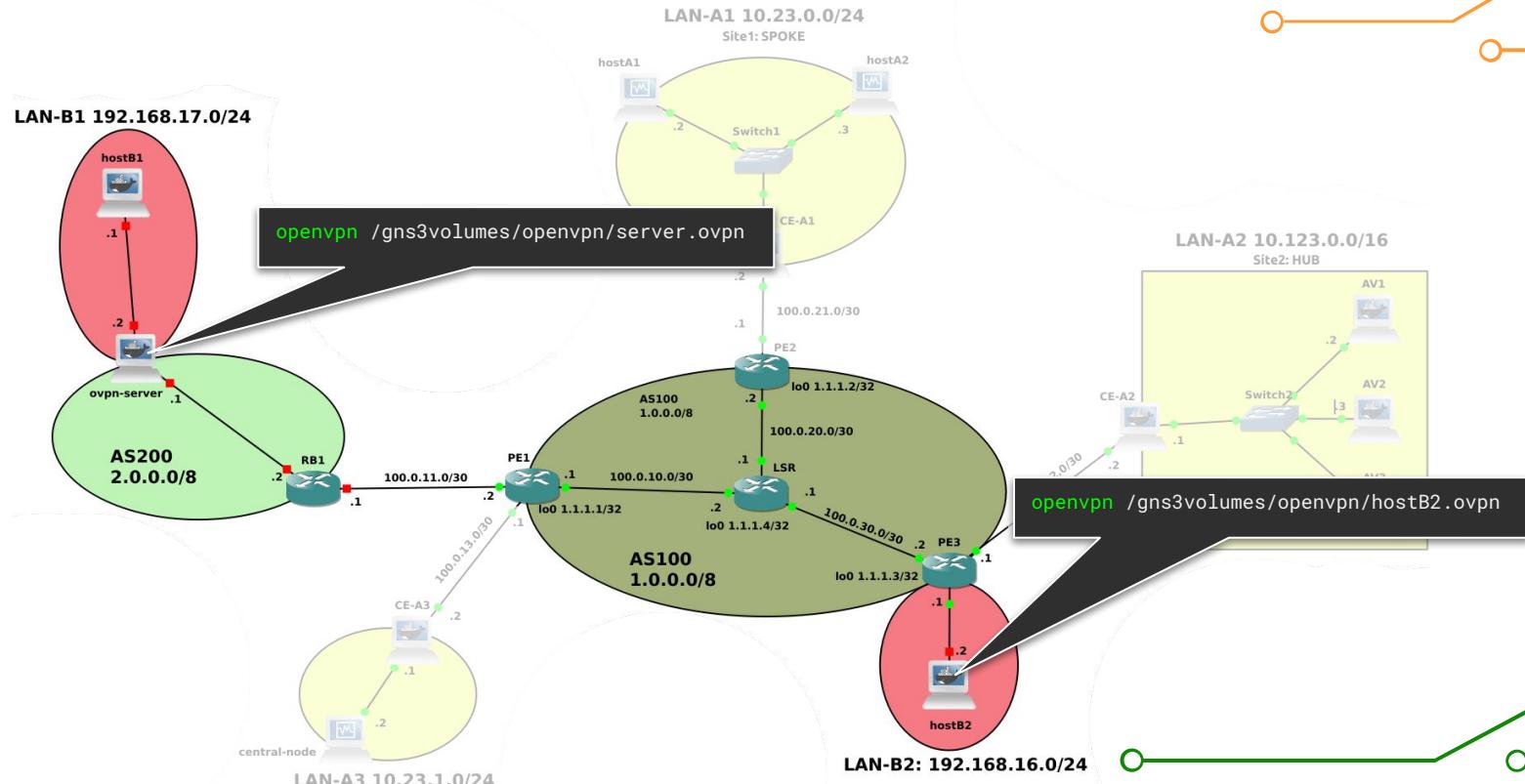
- Verifica Ping ICMP verso 2.0.0.1
- Verifica Ping OpenVPN verso 192.158.100.1

TEST 3: Ping HostB1-> HostB2

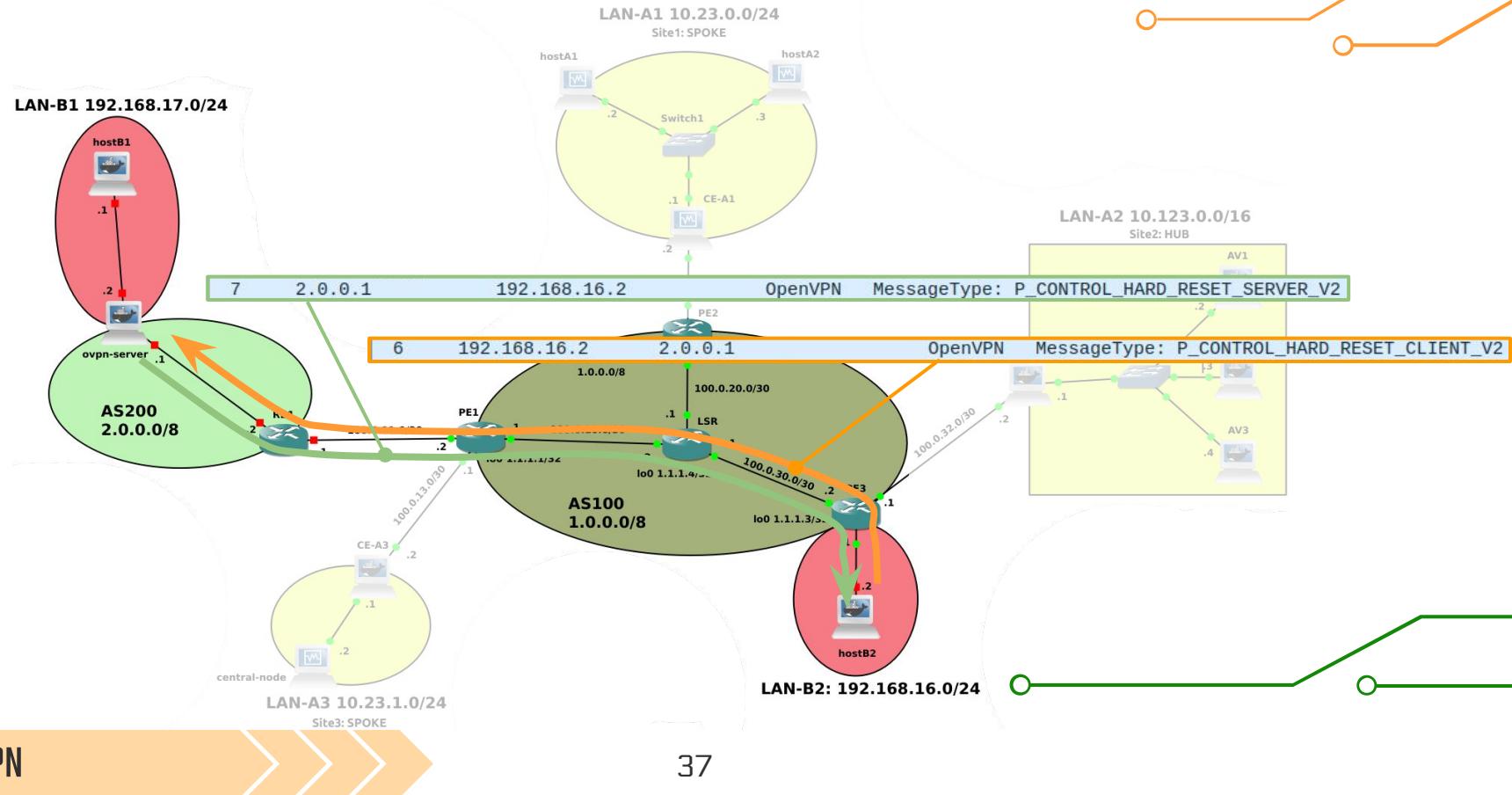
- Verifica encapsulamento dei pacchetti
- Verifica istradamento su ogni link attraversato



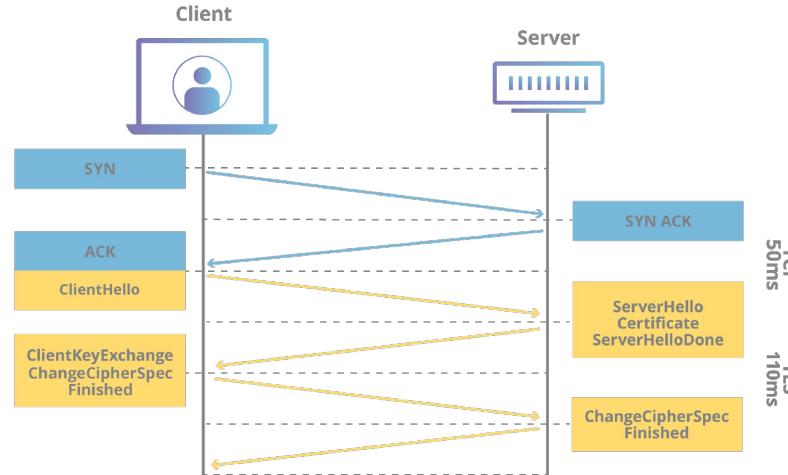
TEST 1: TLS Handshake



TEST 1: TLS Handshake

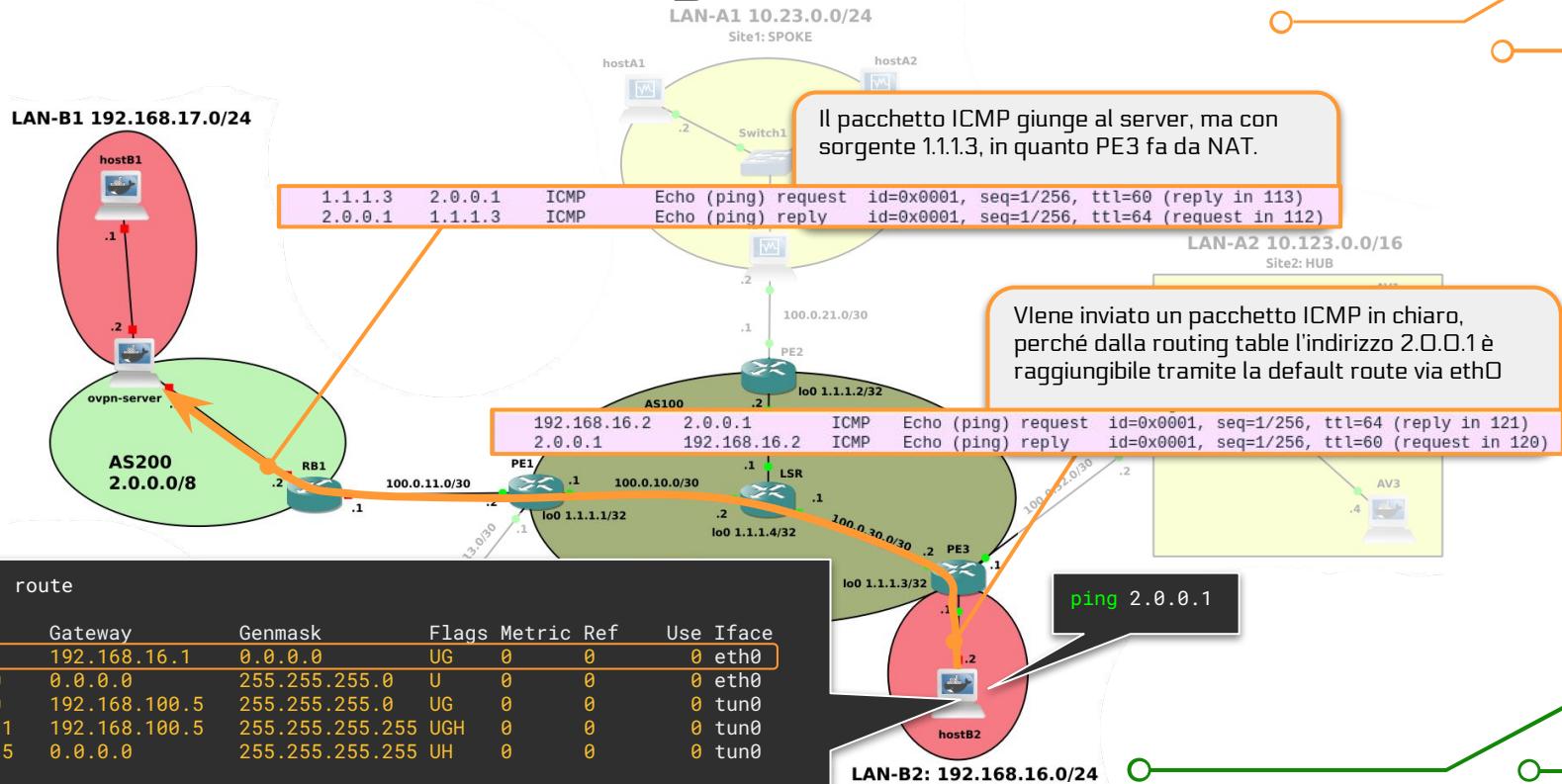


Complete Exchange

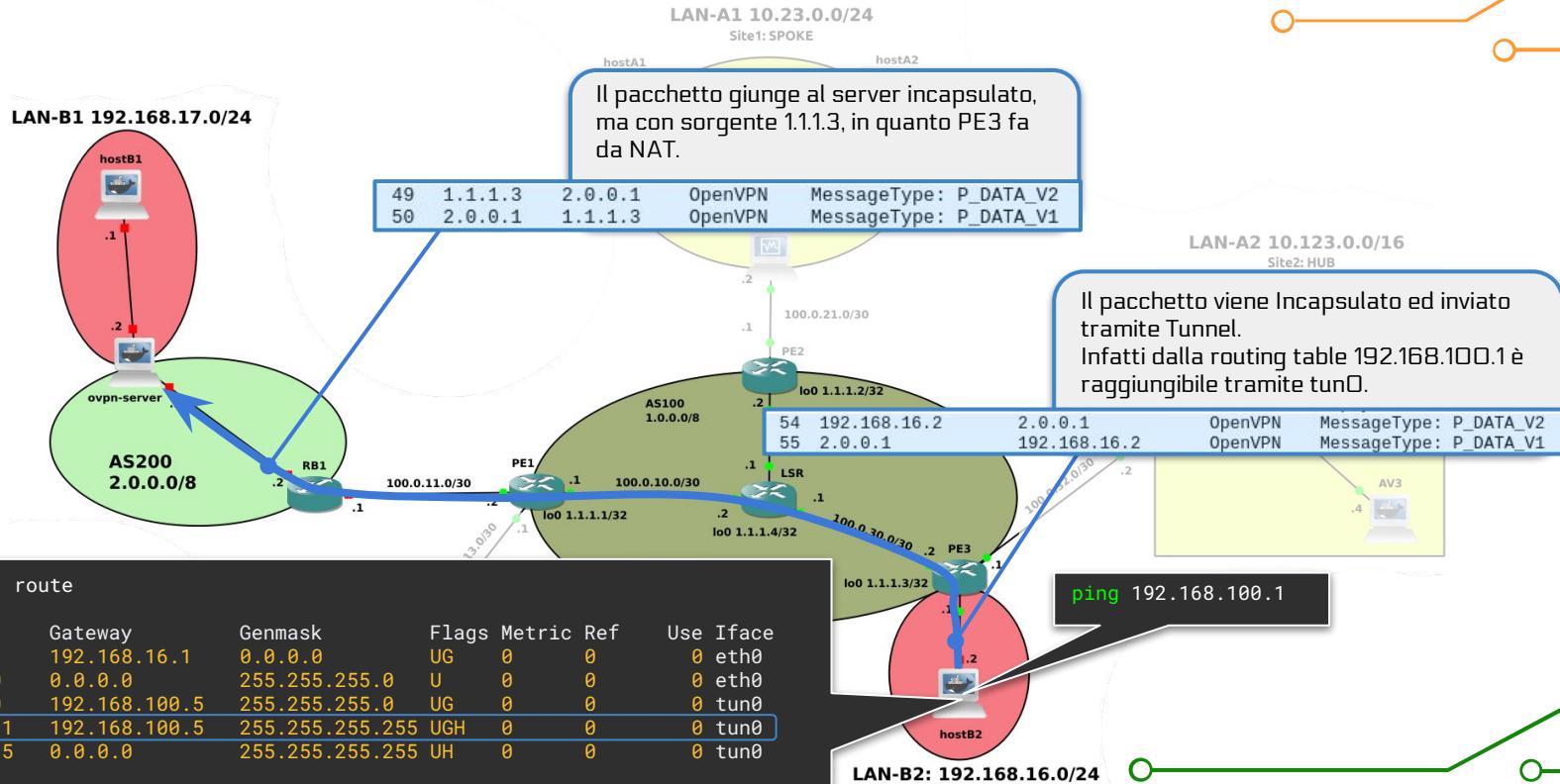


6	192.168.16.2	2.0.0.1	OpenVPN	MessageType: P_CONTROL_HARD_RESET_CLIENT_V2
7	2.0.0.1	192.168.16.2	OpenVPN	MessageType: P_CONTROL_HARD_RESET_SERVER_V2
8	192.168.16.2	2.0.0.1	OpenVPN	MessageType: P_ACK_V1
9	192.168.16.2	2.0.0.1	TLSv1.3	Client Hello
10	2.0.0.1	192.168.16.2	TLSv1.3	Server Hello, Change Cipher Spec, Application Data, Application Data
11	2.0.0.1	192.168.16.2	TLSv1.3	Continuation Data
12	2.0.0.1	192.168.16.2	TLSv1.3	Continuation Data
13	192.168.16.2	2.0.0.1	OpenVPN	MessageType: P_ACK_V1
14	192.168.16.2	2.0.0.1	OpenVPN	MessageType: P_ACK_V1
15	192.168.16.2	2.0.0.1	TLSv1.3	Change Cipher Spec
16	192.168.16.2	2.0.0.1	TLSv1.3	Continuation Data
17	192.168.16.2	2.0.0.1	TLSv1.3	Continuation Data
18	2.0.0.1	192.168.16.2	OpenVPN	MessageType: P_ACK_V1
19	2.0.0.1	192.168.16.2	OpenVPN	MessageType: P_ACK_V1
20	2.0.0.1	192.168.16.2	TLSv1.3	Application Data, Application Data
21	2.0.0.1	192.168.16.2	TLSv1.3	Application Data
22	192.168.16.2	2.0.0.1	OpenVPN	MessageType: P_ACK_V1
23	192.168.16.2	2.0.0.1	OpenVPN	MessageType: P_ACK_V1
24	192.168.16.2	2.0.0.1	TLSv1.3	Application Data

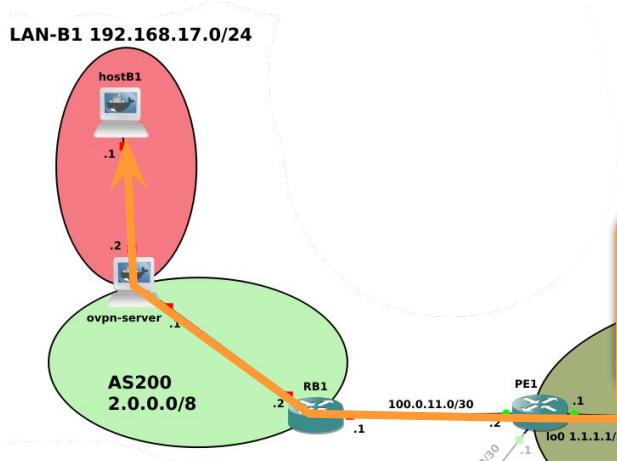
TEST 2: Ping Server 2.0.0.1



TEST 2: Ping Server 192.168.100.1

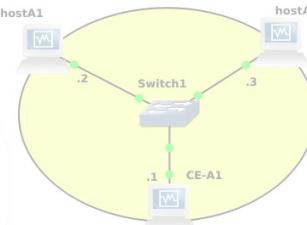


TEST 3: Ping HostB2



LAN-A1 10.23.0.0/24

Site1: SPOKE



LAN-A2 10.123.0.0/16

Site2: HUB

HostB2 vede la sua routing table. La sottorete 192.168.17.0/24 è raggiungibile tramite tun0.

Viene generato un pacchetto con sorgente ip 192.168.100.6, ovvero l'indirizzo dell'interfaccia tun0 nella Overlay Topology.

23:54:36.785428 IP 192.168.100.6 > 192.168.17.1: ICMP echo request, id 144, seq 3, length 64

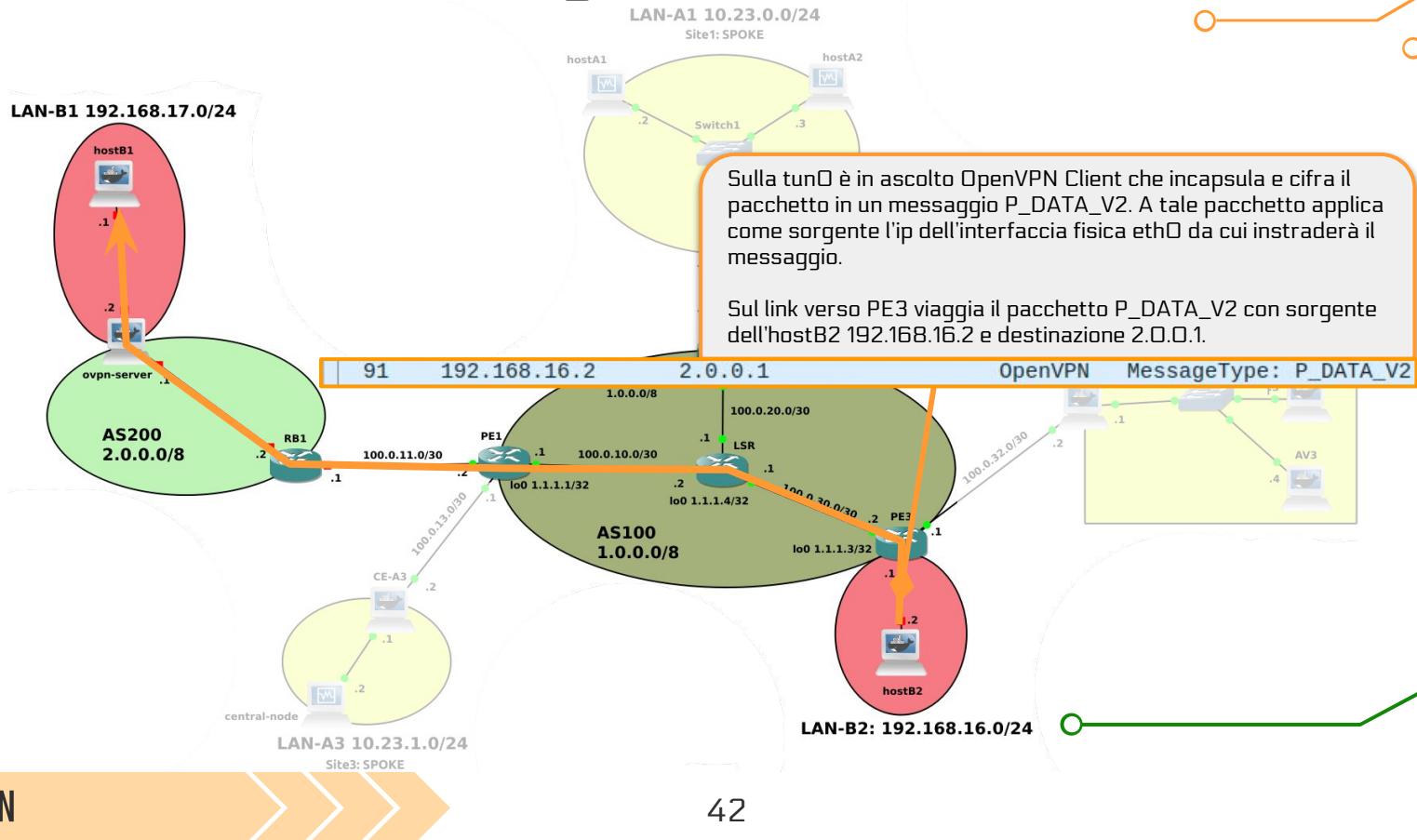
ping 192.168.17.1

root@hostB2: route

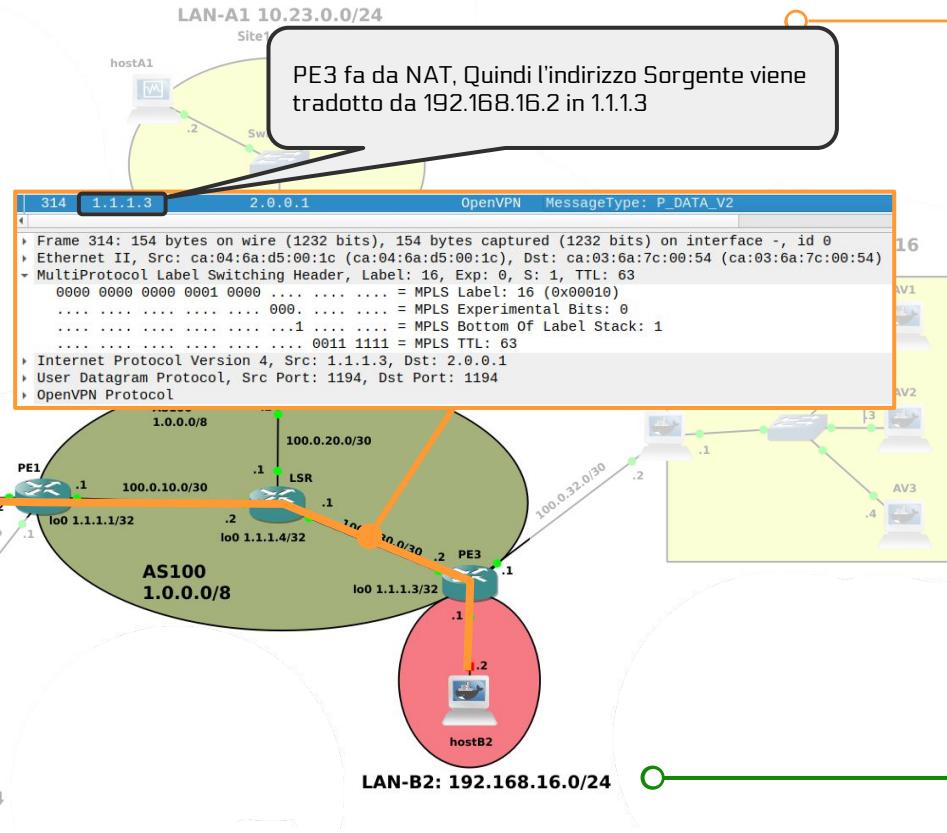
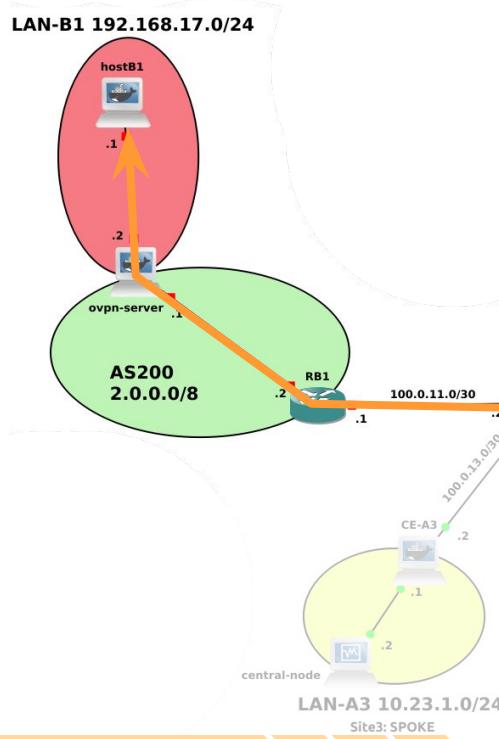
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
default	192.168.16.1	0.0.0.0	UG	0	0	0	eth0
192.168.16.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.17.0	192.168.100.5	255.255.255.0	UG	0	0	0	tun0
192.168.100.1	192.168.100.5	255.255.255.255	UGH	0	0	0	tun0
192.168.100.5	0.0.0.0	255.255.255.255	UH	0	0	0	tun0

LAN-B2: 192.168.16.0/24

TEST 3: Ping HostB2 | B2->PE3

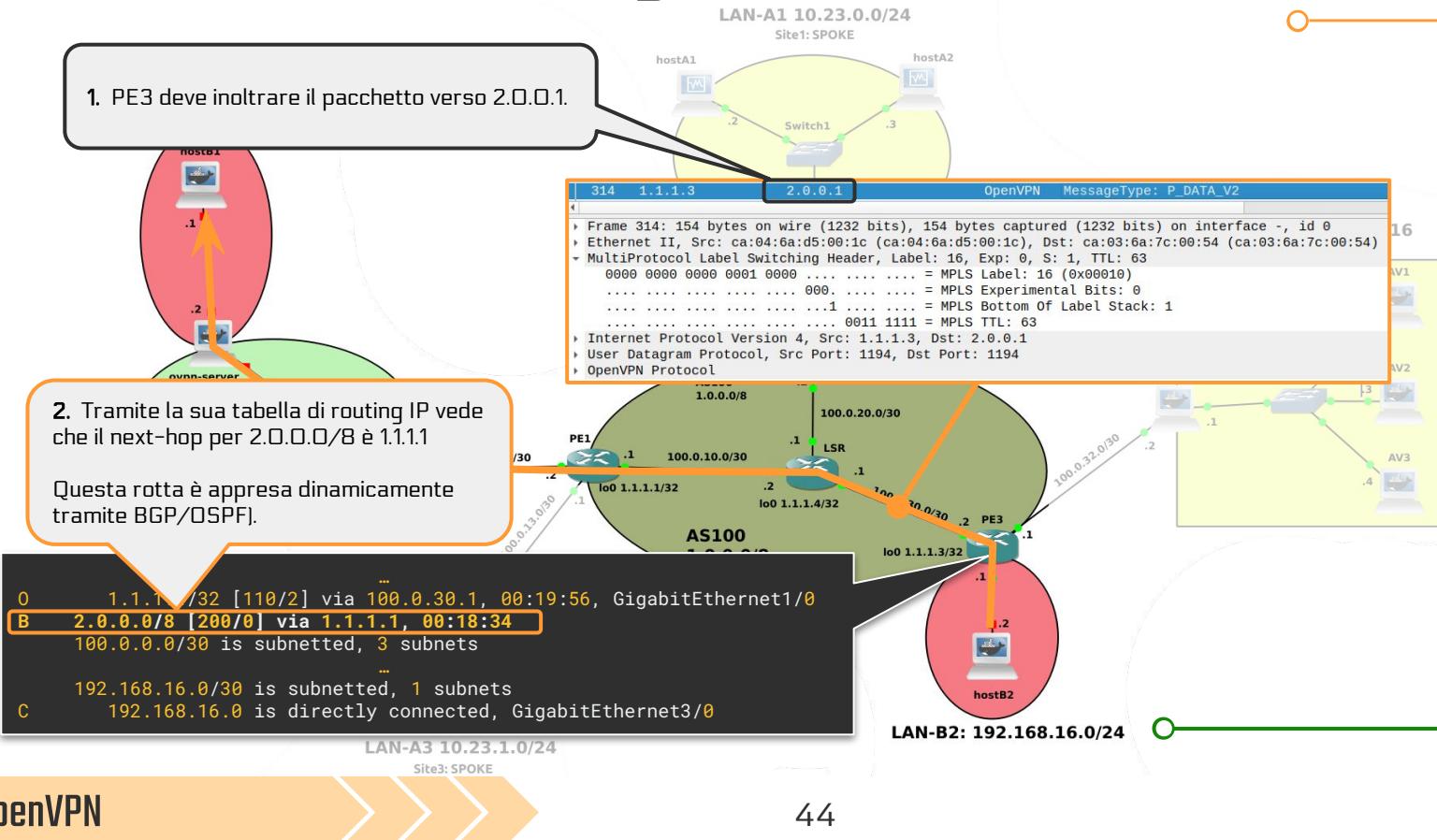


TEST 3: Ping HostB2 | PE3->LSR



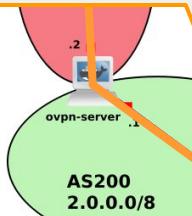
PE3 fa da NAT, Quindi l'indirizzo Sorgente viene tradotto da 192.168.16.2 in 1.1.1.3

TEST 3: Ping HostB2 | PE3->LSR



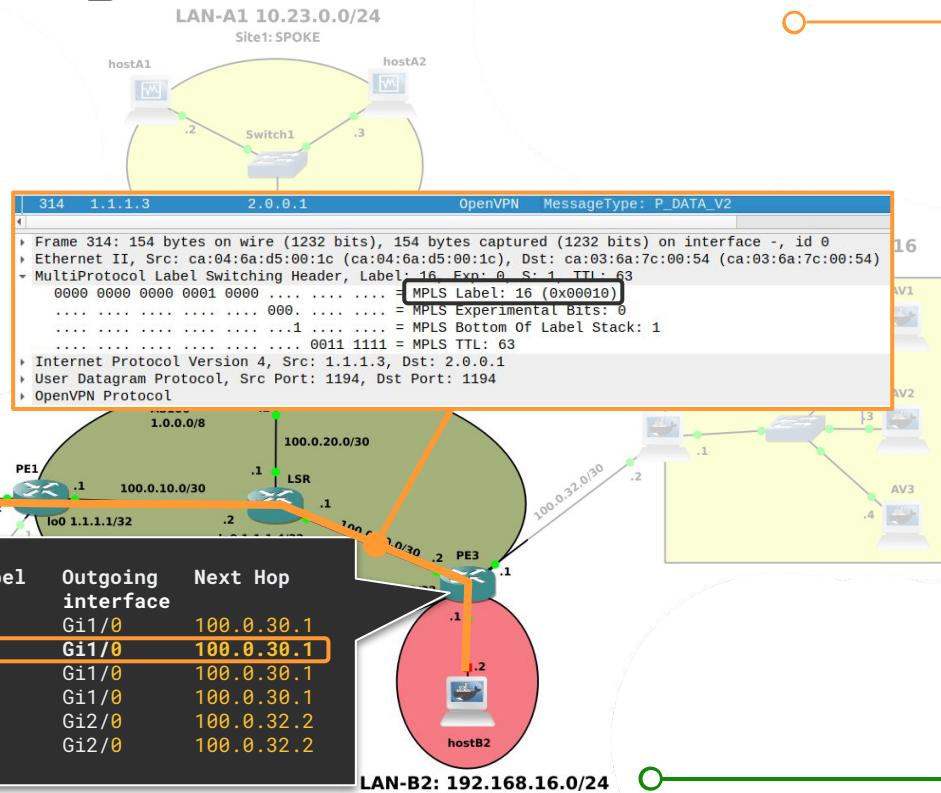
TEST 3: Ping HostB2 | PE3->LSR

Il router PE3 fa da ingress_LER per la rete MPLS. Effettua il lookup della LIB, e vede che per il next-hop 1.1.1.1 si deve pushare la label 16, ed inoltrare il pacchetto verso il next-hop 100.30.1 (LSR).

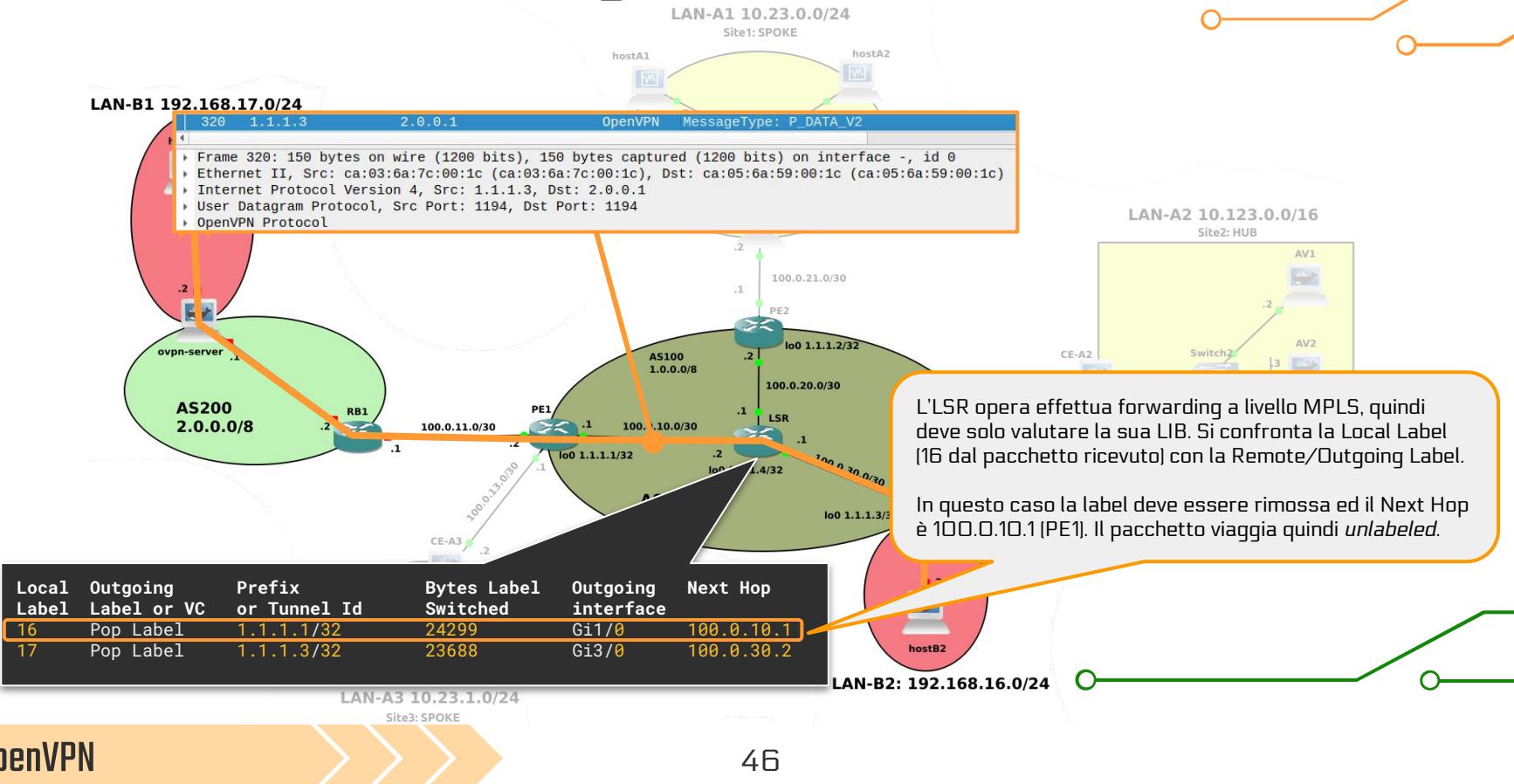


Local Label	Outgoing Label or VC	Prefix or Tunnel Id
16	Pop Label	1.1.1.4/32
17	16	1.1.1.1/32
18	Pop Label	100.0.20.0/30
19	Pop Label	100.0.10.0/30
20	No Label	10.23.0.0/16
21	No Label	10.123.0.0/16

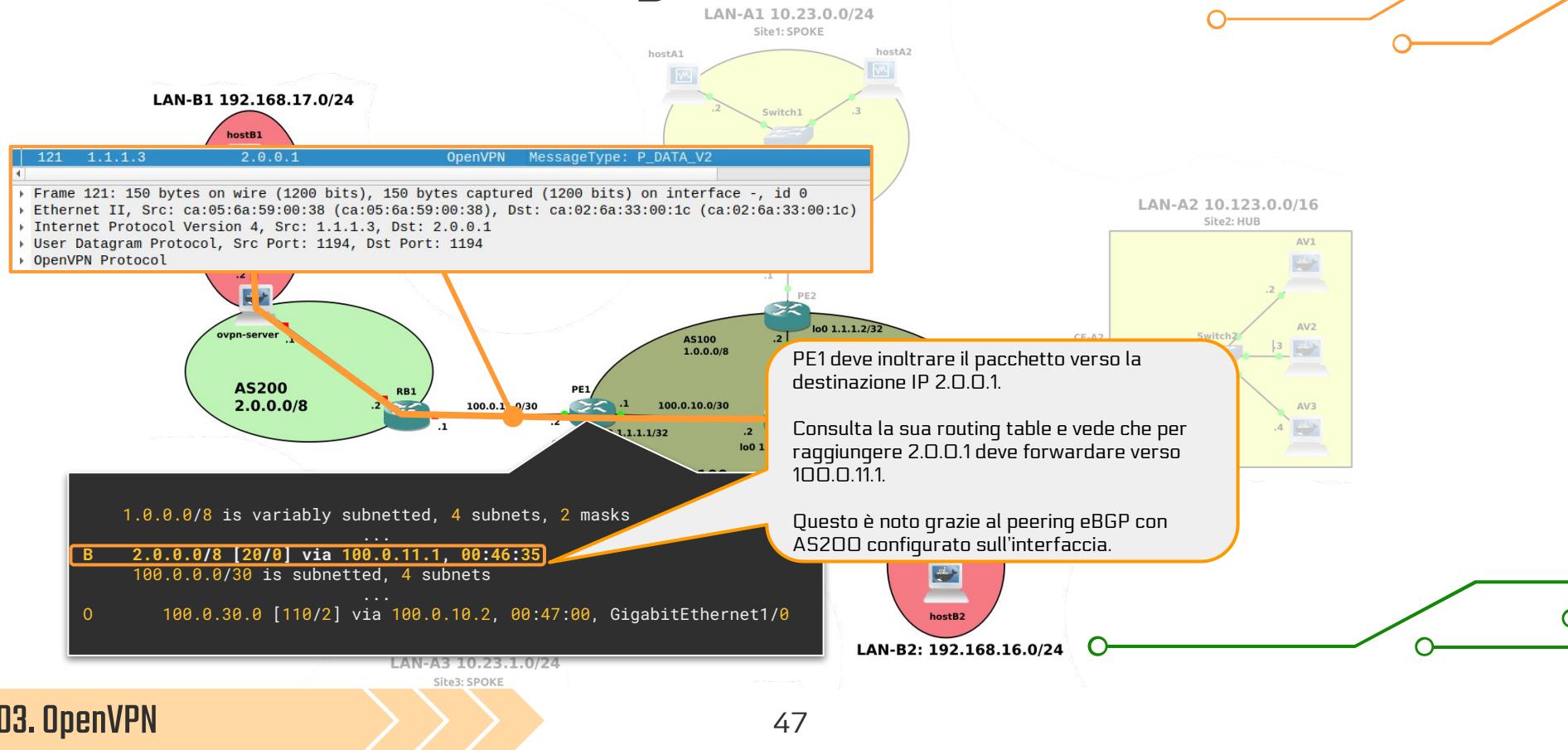
Bytes Label Switched	Outgoing interface	Next Hop
0	Gi1/0	100.0.30.1
0	Gi2/0	100.0.32.2
0	Gi2/0	100.0.32.2



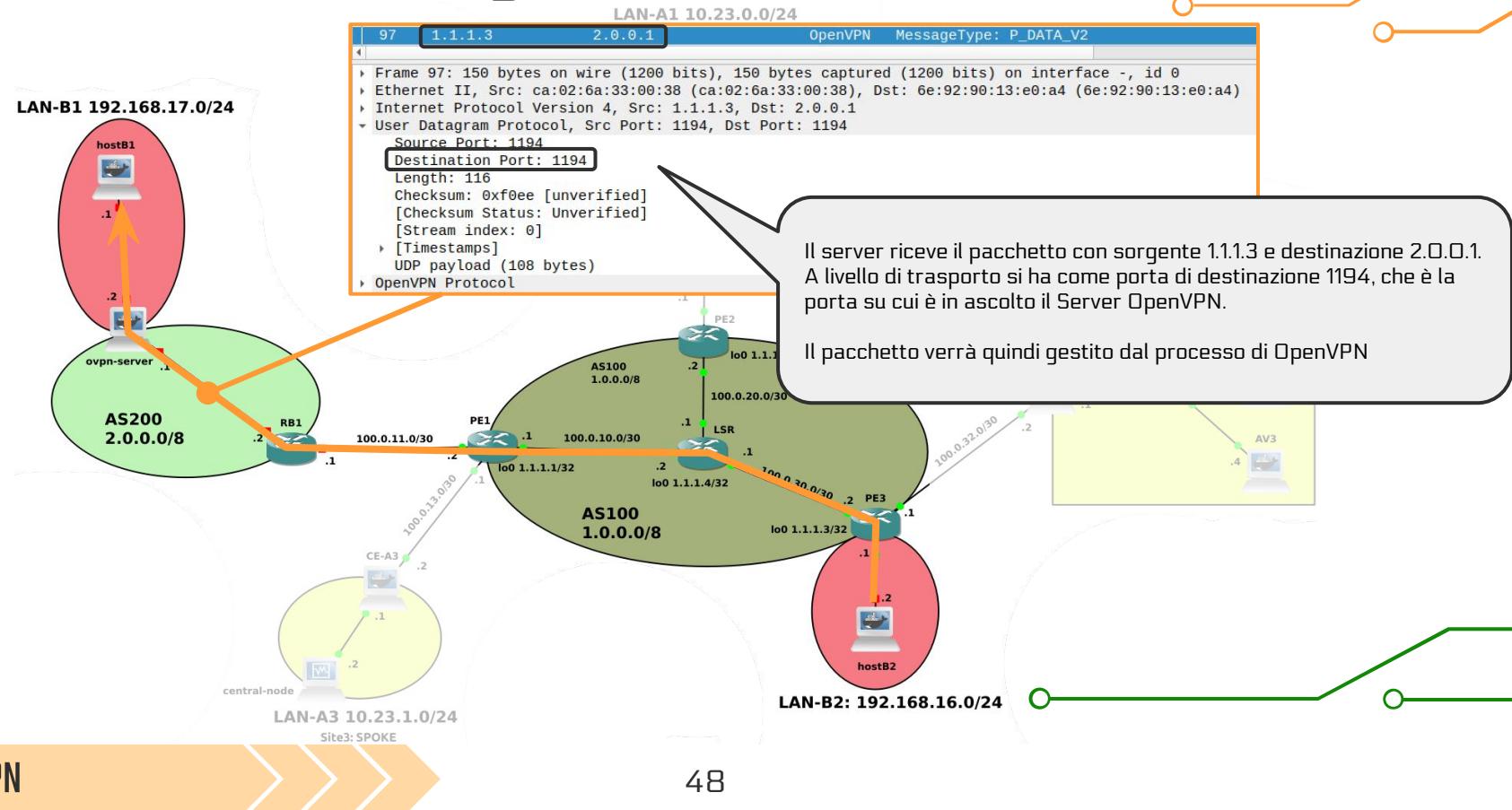
TEST 3: Ping HostB2 | LSR->PE1



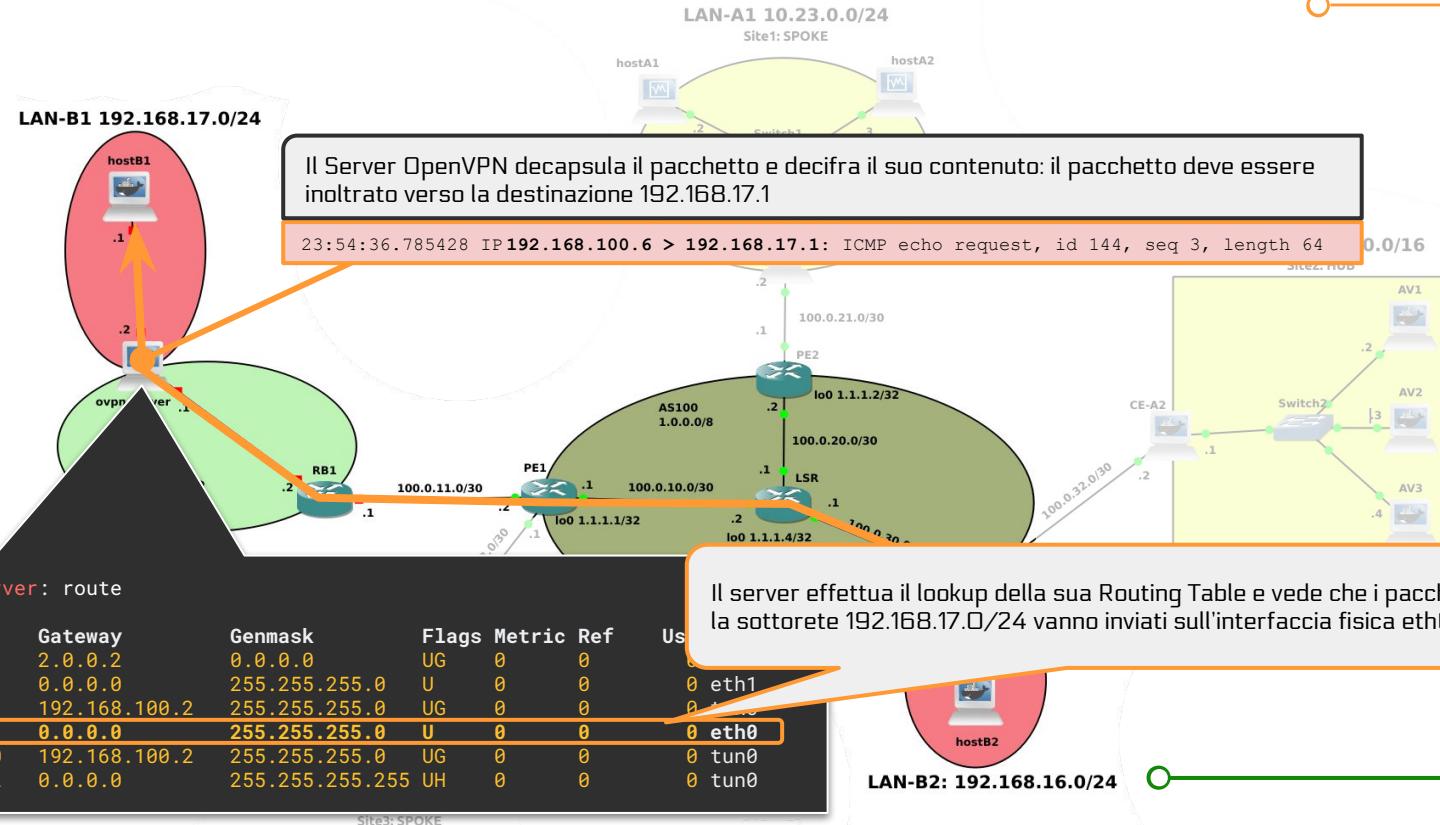
TEST 3: Ping HostB2 | PE1->RB1



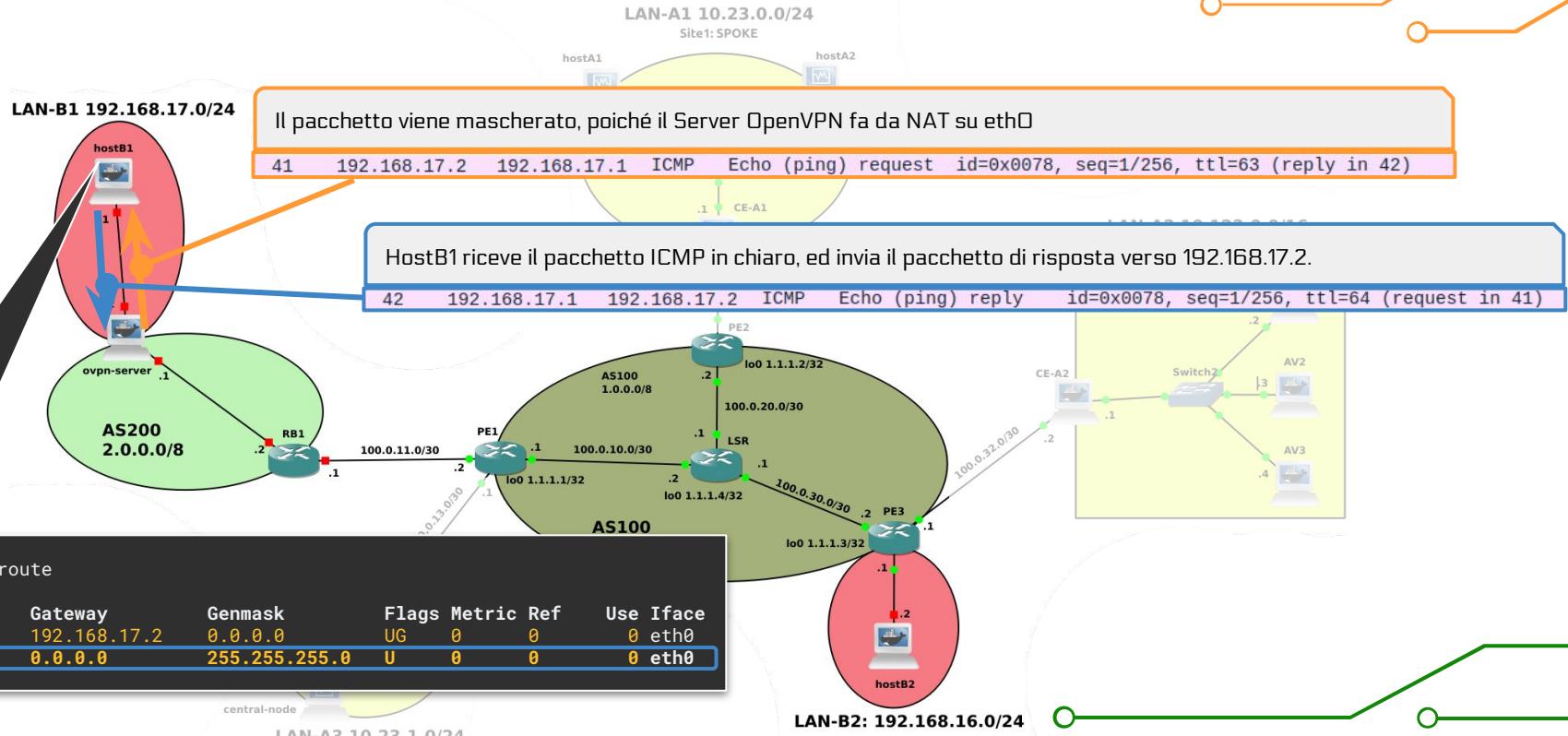
TEST 3: Ping HostB2 | RB1->Server



TEST 3: Ping HostB2 | OpenVPN Server



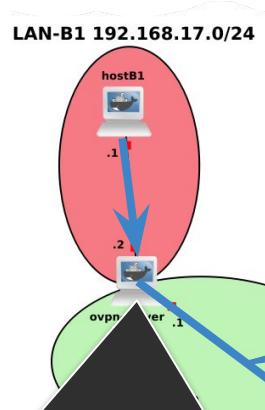
TEST 3: Ping HostB2 | Server<->HostB1



TEST 3: Ping HostB2 | Reply Trace

LAN-A1 10.23.0.0/24

Site1: SPOKE



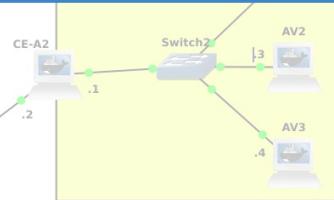
A questo punto il pacchetto segue in modo analogo la rotta precedente.

1. Il server-ovpn effettua il DE-nat e vede che deve inoltrare il pacchetto verso HostB2.tun0 (100.5).
2. Dalla sua tabella di routing vede che deve inoltrare il messaggio verso 192.168.100.0/24 tramite l'interfaccia tun0.
3. Il pacchetto viene di nuovo gestito da openvpn-server che lo inoltra tramite il tunnel con hostB2, quindi con destinazione 1.1.1.3. Il pacchetto attraversa di nuovo la rete MPLS, fa label switching, e giunge infine a PE3.

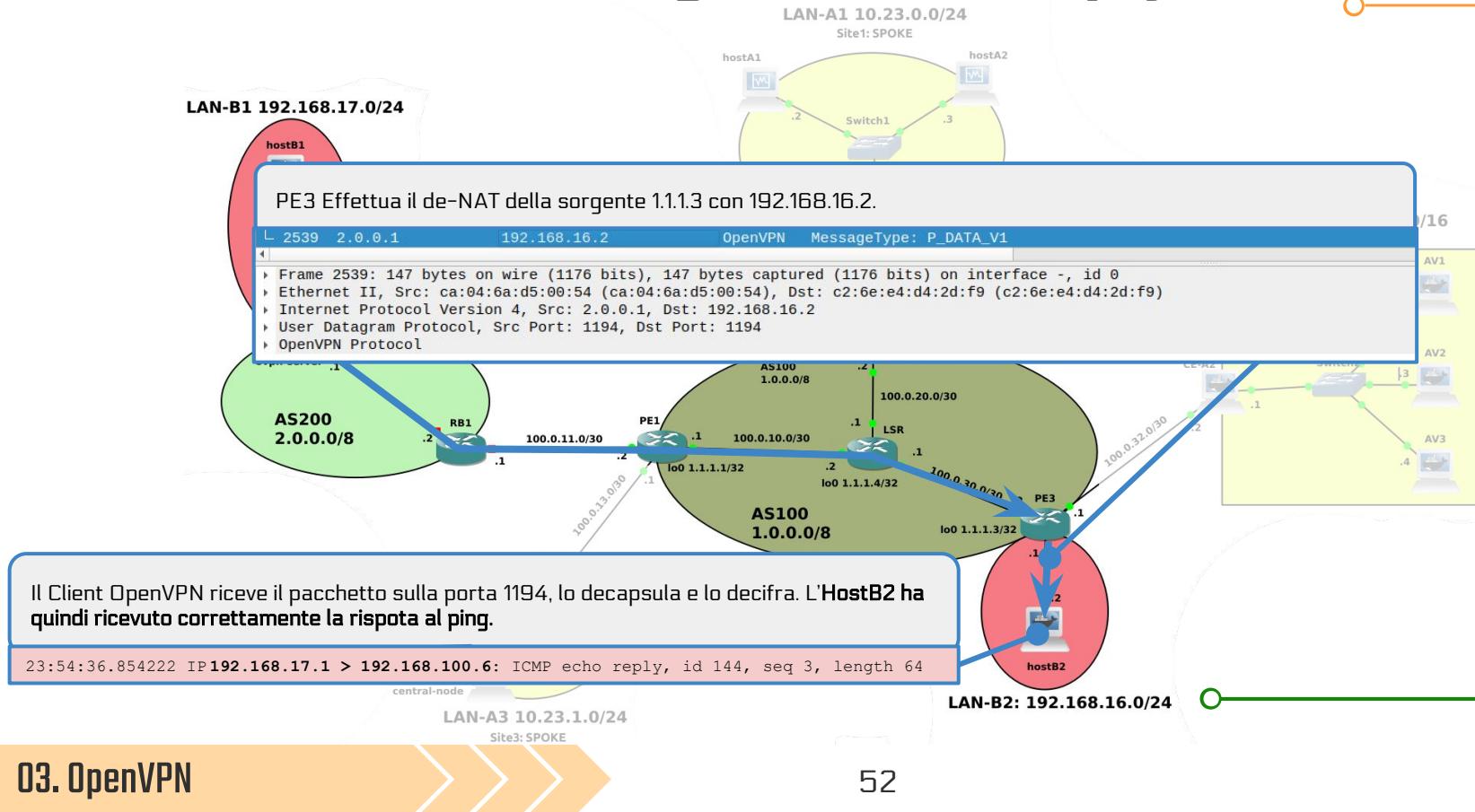
```
root@ovpn-server: route
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
default	2.0.0.2	0.0.0.0	UG	0	0	0	eth1
2.0.0.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1
192.168.16.0	192.168.100.2	255.255.255.0	UG	0	0	0	tun0
192.168.17.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.100.0	192.168.100.2	255.255.255.0	UG	0	0	0	tun0
192.168.100.2	0.0.0.0	255.255.255.255	UH	0	0	0	tun0

Site3: SPOKE



TEST 3: Ping HostB2 | Reply Trace





04

Firewalls

Configurazione Firewall sui Gateway
delle LAN A1 e A2

Specifica

L'implementazione richiesta per il **firewall sul GW di LAN-A1** è la seguente:

1. Permettere il traffico tra la LAN e l'esterno solo se iniziato dalla LAN con SNAT.
2. Negare tutto il traffico verso il GW tranne per SSH e ICMP solo se iniziato dalla LAN.
3. Permettere tutto il traffico dal GW verso ovunque (ed i relativi pacchetti di risposta).
4. Permettere il port-forwarding con DNAT verso hostA1 e hostA2 dall'esterno soltanto per il servizio HTTP.

L'implementazione richiesta per il **firewall sul GW di LAN-A2** è la seguente:

1. Permettere la comunicazione end-to-end bidirezionale tra il central node e gli AVs, e negare tutto il resto.
2. Permettere la comunicazione tra gli spokes (altrimenti non avrebbe senso la topologia).

Configurazione: CE-A1

```
export EXT=enp0s3
export LAN=macsec0

iptables -F
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT

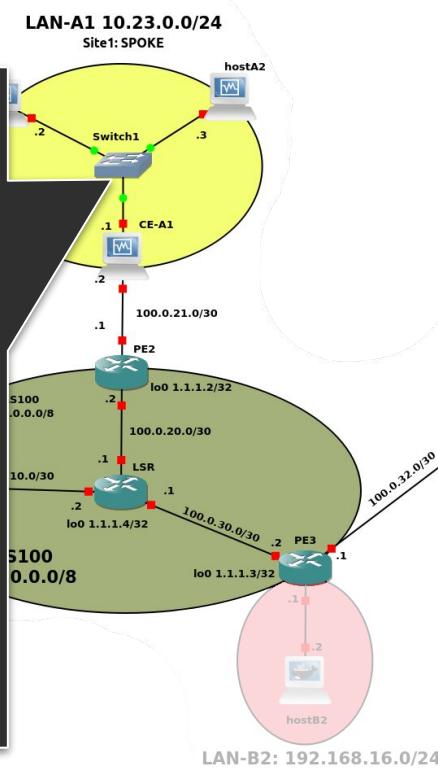
iptables -A FORWARD -i $LAN -o $EXT -j ACCEPT
iptables -A FORWARD -m state -state ESTABLISHED -j ACCEPT
iptables -A POSTROUTING -t nat -o $EXT -j SNAT -to
10.23.0.2-10.23.0.254

iptables -A INPUT-i $LAN -p tcp -dport 22 -j ACCEPT
iptables -A INPUT -i $LAN -p icmp -j ACCEPT

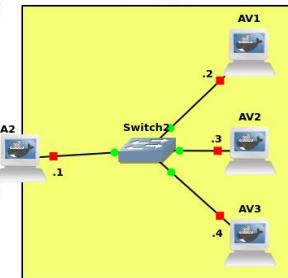
iptables -A INPUT -m state -state ESTABLISHED,RELATED -j
ACCEPT

iptables -A FORWARD -i $EXT -o $LAN -p tcp -dport 80 -j ACCEPT
iptables -A PREROUTING -t nat -i $EXT -p tcp -dport 80 -j DNAT
-to 10.23.0.3
```

LAN-A3 10.23.1.0/24
Site3: SPOKE



LAN-A2 10.123.0.0/16
Site2: HUB



1. Permettere il traffico tra la LAN e l'esterno solo se iniziato dalla LAN con SNAT

- Di default viene scartato tutto il traffico che il GW deve forwardare.
- Aggiungiamo una regola che abiliti il forwarding dei pacchetti che dall'interfaccia connessa a LAN-A1 sono diretti verso l'esterno.
 - Il pacchetto passerà per il *forward hook*
- Aggiungiamo una regola che abilita il forwarding dei pacchetti che appartengono a connessioni già stabilite.
- Aggiungiamo una regola che applichi SNAT ai pacchetti diretti verso l'esterno modificando l'IP sorgente con uno nella rete 10.23.0.0/24.
 - Il pacchetto sarà catturato dal *postrouting hook*

```
iptables -P FORWARD DROP
iptables -A FORWARD -i $LAN -o $EXT -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
iptables -A POSTROUTING -t nat -o $EXT -j SNAT -to 10.23.0.2-10.23.0.254
```

2. Negare tutto il traffico verso il GW, tranne SSH e ICMP solo se iniziato dalla LAN

- Di default si scarta tutto il traffico diretto verso il GW.
- Aggiungiamo una regola per accettare il traffico SSH e ICMP in input al GW.
 - Il pacchetto passerà per l'*input hook*
- Eventualmente si può aggiungere una regola per scartare il traffico SSH e ICMP in uscita dal GW verso l'interfaccia connessa a LAN-A1 per nuove comunicazioni.
 - Il pacchetto passerà per l'*output hook*
 - L'idea è impedire al GW di iniziare comunicazioni SSH e ICMP verso LAN-A1.
 - La regola deve scartare il traffico SSH e ICMP perché di default si accetta tutto in uscita dal GW, come vedremo nella regola successiva.

```
iptables -P INPUT DROP
iptables -A INPUT -i $LAN -p tcp -dport 22 -j ACCEPT
iptables -A INPUT -i $LAN -p icmp -j ACCEPT
#iptables -A OUTPUT -o $LAN -p tcp -dport 22 -m state --state NEW -j DROP
#iptables -A OUTPUT -o $LAN -p icmp -m state --state NEW -j DROP
```

3. Permettere tutto il traffico dal GW verso ovunque (ed i relativi pacchetti di risposta)

- Di default si accetta tutto il traffico in uscita dal GW.
- Aggiungiamo una regola per accettare il traffico in input al GW per connessioni già stabilite o in relazione con una di quelle iniziate dal GW.
 - Il pacchetto passerà per l'*input hook*
 - Abilitiamo il GW ad accettare eventuali risposte o messaggi di errore per le comunicazioni da lui iniziata.

```
iptables -P OUTPUT ACCEPT  
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

4. Permettere il port-forwarding con DNAT verso hostA2 dall'esterno solo per HTTP

- Di default si scarta tutto il traffico che il GW deve forwardare per la prima regola.
- Aggiungiamo una regola che abilita il forwarding del traffico HTTP che dalla interfaccia connessa a LAN-A1 sono diretti verso l'esterno.
 - Il pacchetto passerà per il *forward hook*
- Aggiungiamo una regola che applica il DNAT ai pacchetti diretti verso l'interfaccia connessa a LAN-A1 sostituendo l'IP di destinazione con 10.23.0.3.
 - Il pacchetto passerà per il *prerouting hook*

```
iptables -A FORWARD -i $EXT -o $LAN -p tcp -dport 80 -j ACCEPT  
iptables -A PREROUTING -t nat -i $EXT -p tcp -dport 80 -j DNAT -to 10.23.0.3
```

4. Permettere il port-forwarding con DNAT verso hostA2 dall'esterno solo per HTTP

Il **port-forwarding** con DNAT su un sito di una BGP/MPLS VPN non ha molto senso perché il suo obiettivo è quello di esporre un servizio interno verso la rete Internet pubblica.

Per questo motivo le scelte effettuate per la realizzazione del DNAT seguono l'idea di un **possibile scenario reale** in cui si utilizza questa soluzione:

- L'HostA2 nella LAN-A1 mantiene un web server che offre un servizio di consultazione dei report stilati dal CentralNode in LAN-A3 ai clients interni alla BGP/MPLS VPN.
- Qualsiasi host si contatta nella rete 10.23.0.0/24, il GW redireziona il traffico HTTP verso il web server messo a disposizione da hostA2.
- Per il firewall implementato, solamente gli hosts in LAN-A3 possono usufruire del servizio web offerto.

TEST CONFIGURAZIONE FIREWALL CE-A1

Per testare la configurazione del firewall di CE-A1 sono stati effettuati i seguenti test:

TEST 1: Connessione HostA1 <-> AV1

- Verifica di connessione hostA1 -> AV1
- Verifica di connessione AV1 -> hostA1 solo per HTTP

TEST 2: Connessione GW <-> AV1

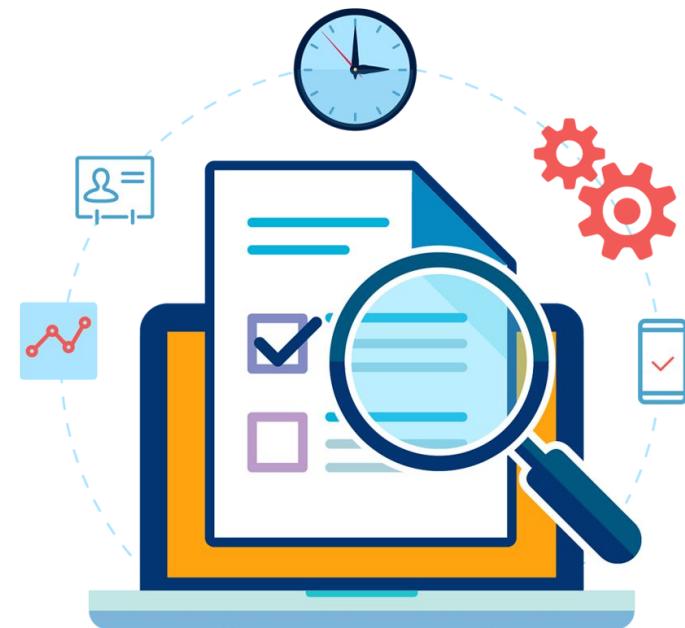
- Verifica di connessione GW -> AV1
- Verifica di connessione non possibile AV1 -> GW

TEST 3: Connessione HostA1 -> GW

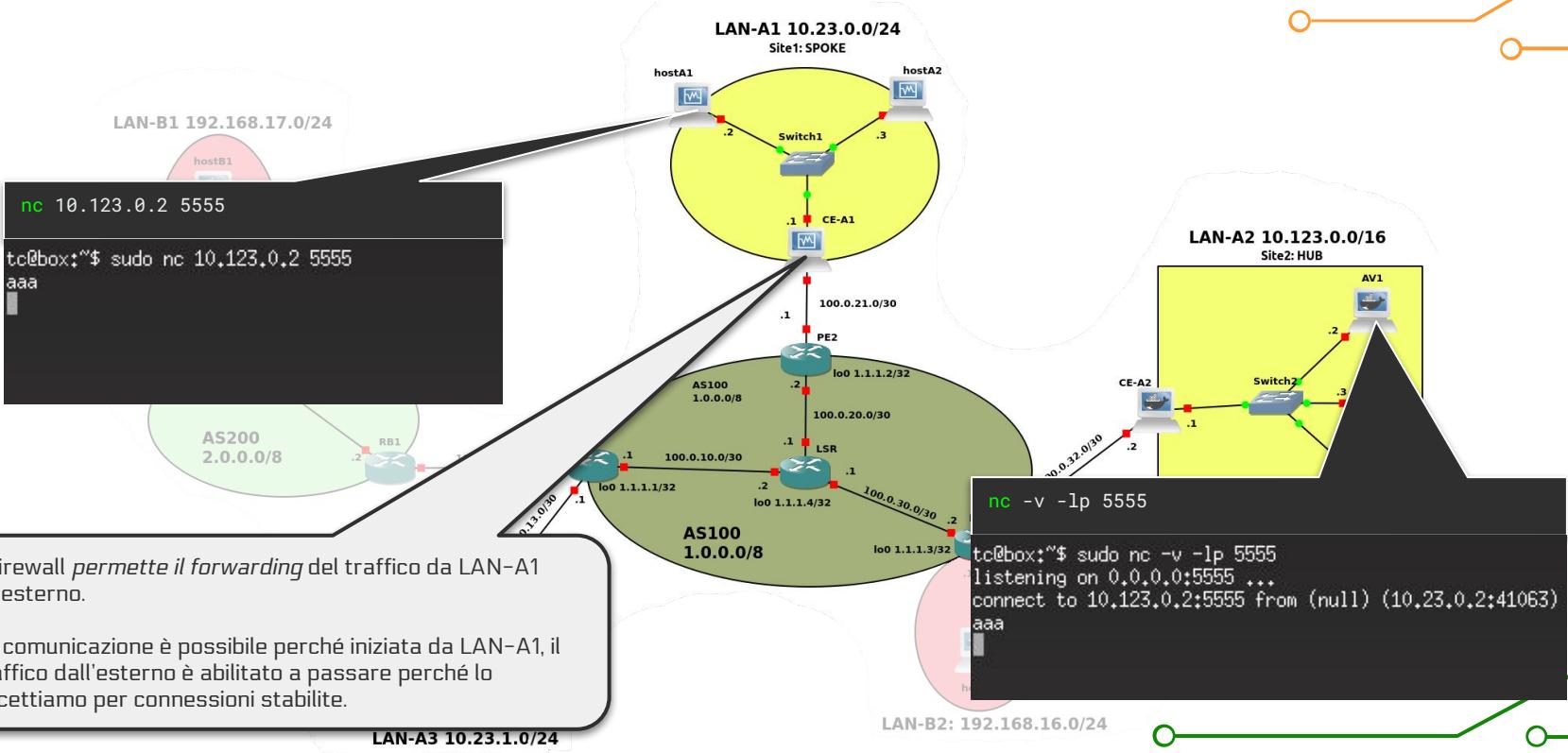
- Verifica Ping ICMP HostA1 -> GW
- Verifica di connessione hostA1 -> GW solo per SSH

TEST 4: Applicazione SNAT and DNAT

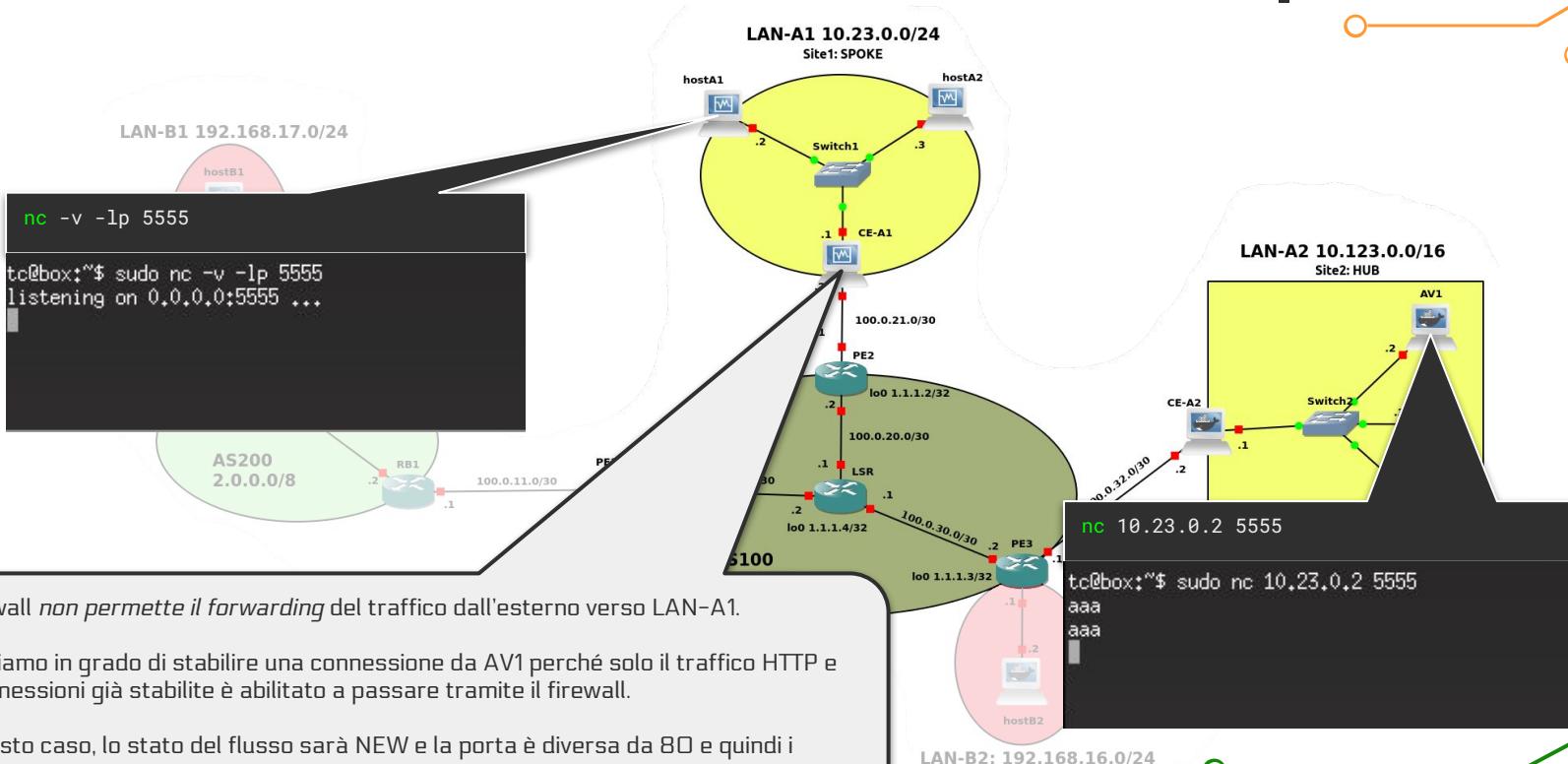
- Verifica tramite Wireshark



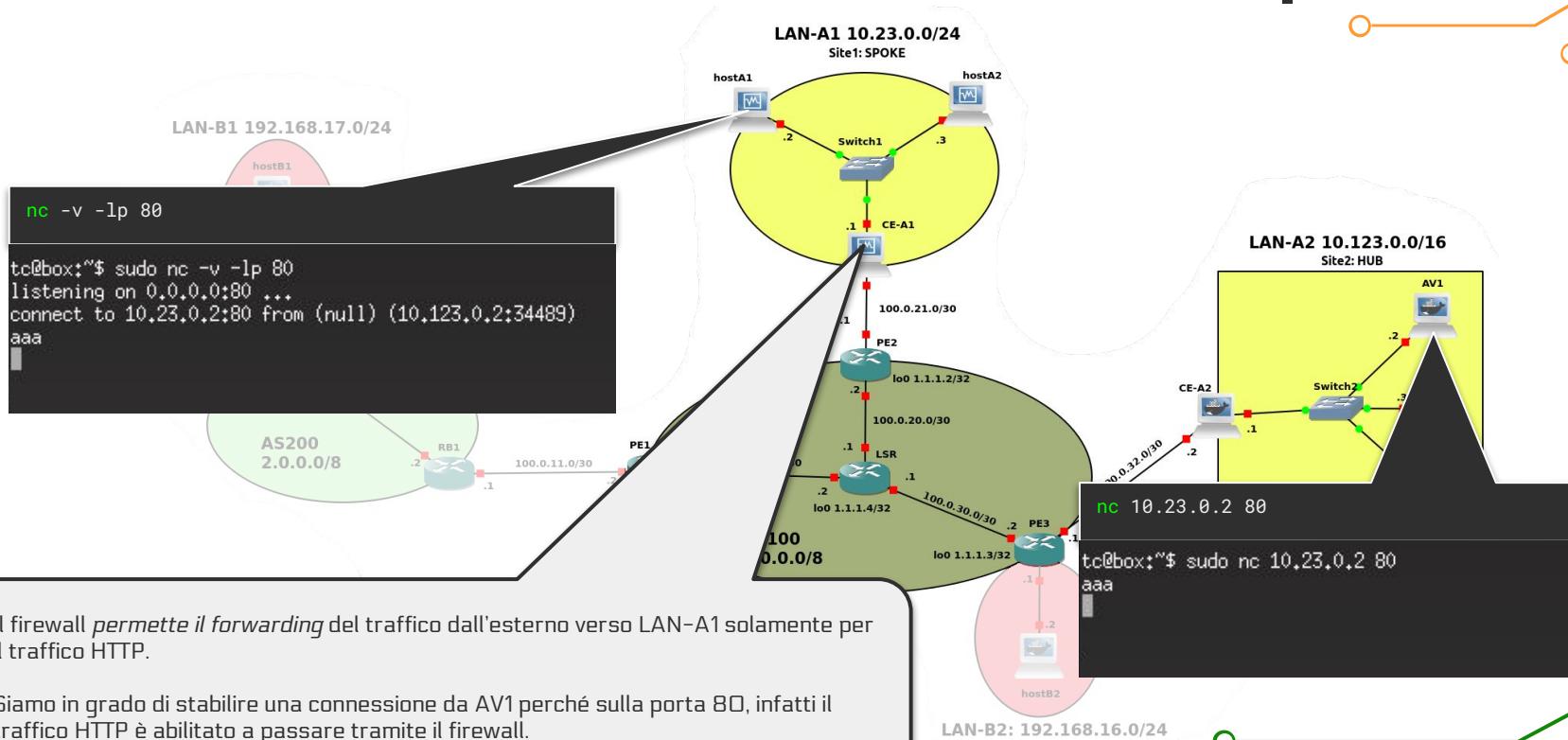
TEST 1: Connessione HostA1 -> AV1



TEST 1: Connessione AV1 -> HostA1 | REJECT



TEST 1: Connessione AV1 -> HostA1 | ACCEPT

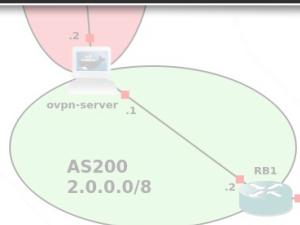


TEST 2: Connessione GW -> AV1

```
~ $ sudo nc 10.123.0.2 5555
```

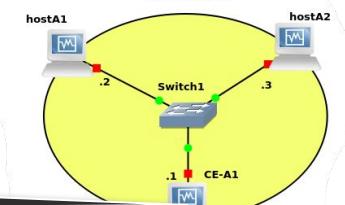
```
aaa  
aaa
```

```
nc 10.123.0.2 5555
```



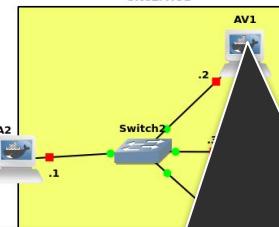
LAN-A1 10.23.0.0/24

Site1: SPOKE



LAN-A2 10.123.0.0/16

Site2: HUB



```
nc -v -lp 5555
```

```
tc@box:~$ sudo nc -v -lp 5555
```

```
listening on 0.0.0.0:5555 ...
```

```
connect to 10.123.0.2:5555 from (null) (10.23.0.186:33196)
```

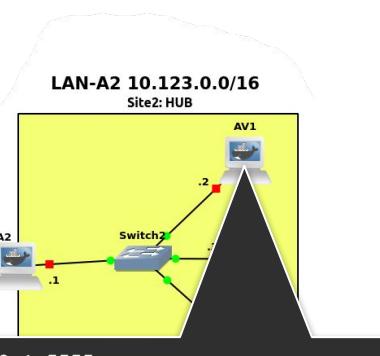
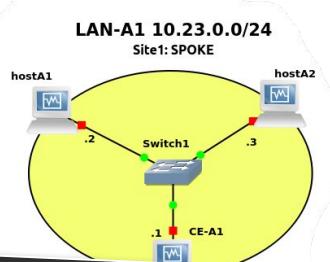
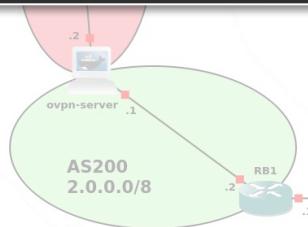
```
aaa  
aaa
```

Il firewall permette *in output* qualsiasi tipo di traffico dal GW verso ovunque.

Siamo in grado di stabilire una connessione dal GW perché il traffico di risposta che arriva dall'esterno appartiene a connessioni stabilite o in relazione con quelle iniziate dal GW, quindi è abilitato a passare tramite il firewall.

TEST 2: Connessione AV1 -> GW

```
~ $ sudo nc -l p 5555  
nc -l p 5555
```



Il firewall permette *in input* al GW dall'esterno solamente il traffico di connessioni stabilite o in relazione con quelle iniziate dal GW.

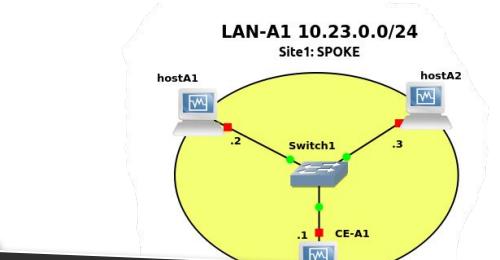
Non siamo in grado di stabilire una connessione da AV1 perché il traffico non è abilitato a passare tramite il firewall.

In questo caso, lo stato del flusso sarà NEW e quindi i pacchetti verranno scartati.

```
tc@box:~$ sudo nc 10.23.0.1 5555  
aaa  
aaa
```

TEST 3: Connessione HostA1 <-> GW

```
~ $ sudo nc -lp 22  
aaa  
aaa  
  
nc -lp 22
```



LAN-A2 10.123.0.0/16

```
ping 10.23.0.1 -c 1
```

LAN-B2: 192.168.16.0/24

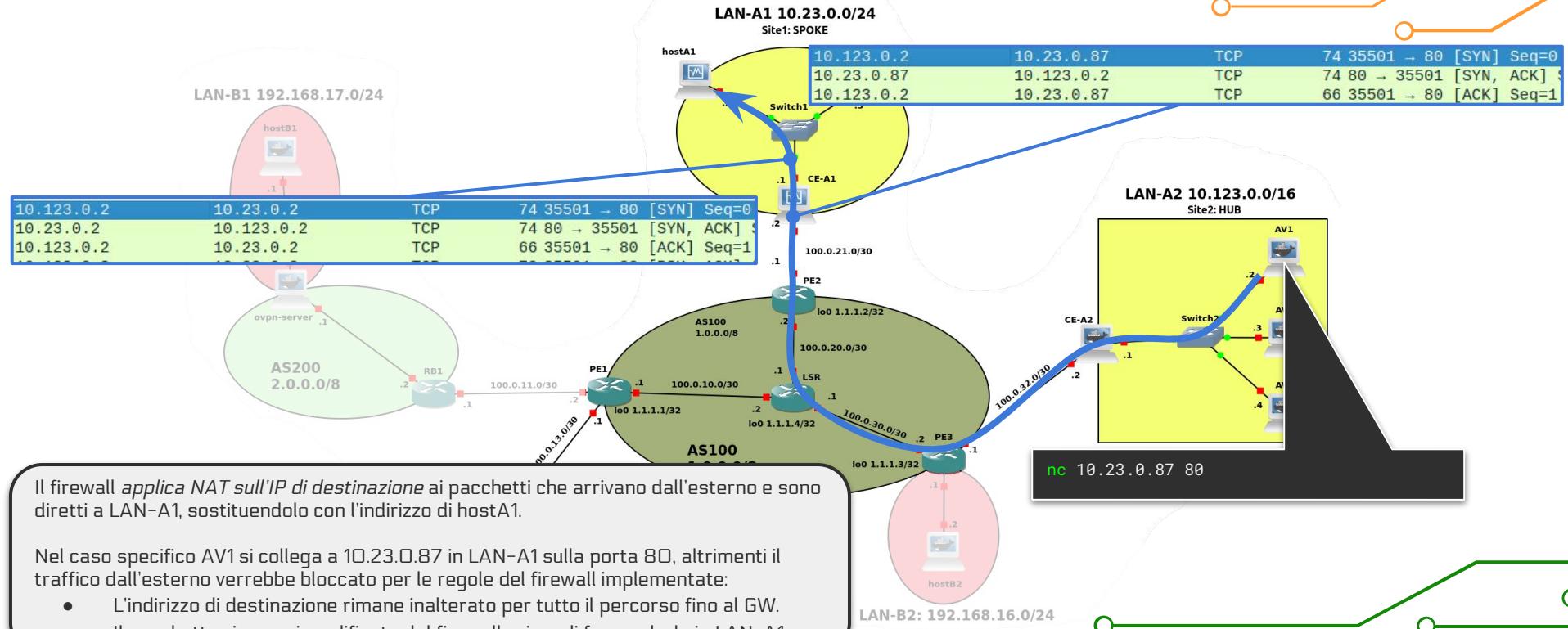
Il firewall permette in input al GW da LAN-A1 solamente il traffico ICMP ed SSH.

Siamo in grado di pingare da hostA1 il GW e ricevere le risposte perché tutto il traffico che esce dal GW è abilitato a passare tramite il firewall, lo stesso per SSH.

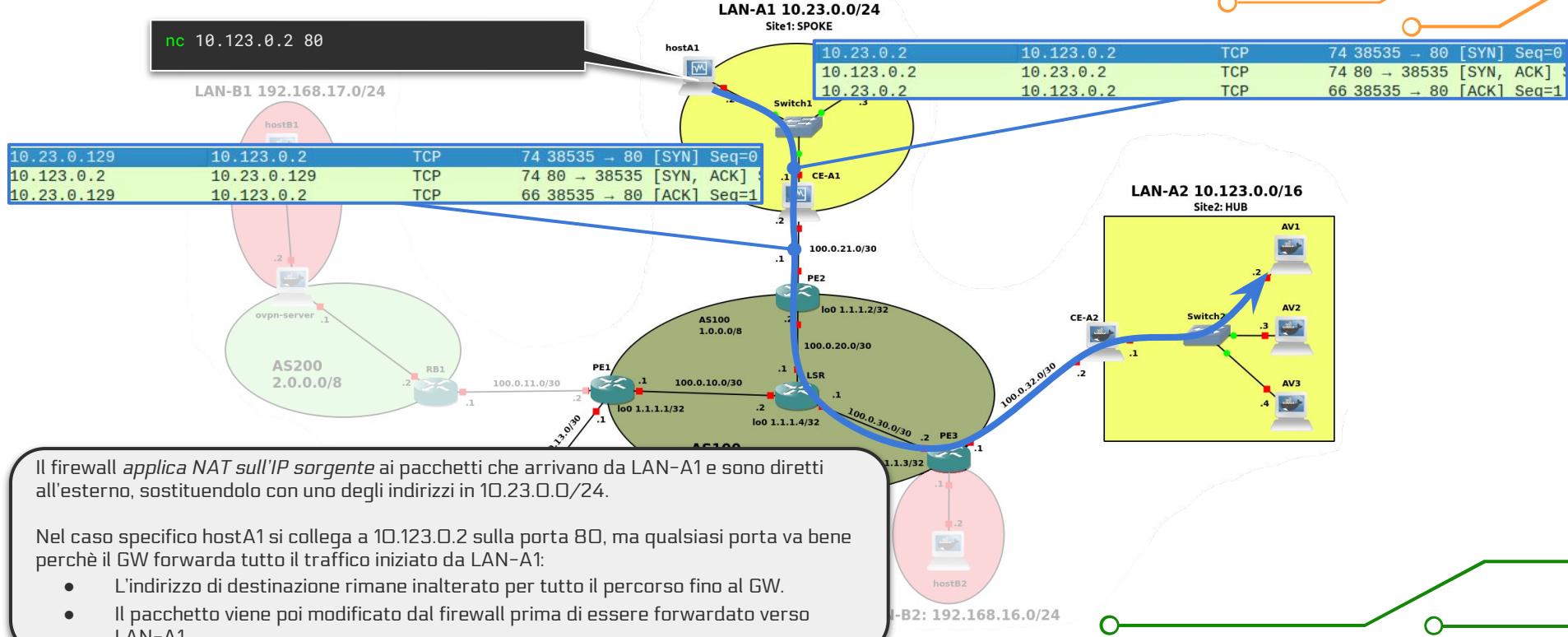
Possiamo stabilire anche la connessione GW → hostA1 perché è accettato tutto il traffico in uscita ed il traffico in entrata di connessioni ESTABLISHED o RELATED.

Per impedirla, si può aggiungere una regola che scarta il traffico SSH ed ICMP dal GW verso LAN-A1 con stato NEW.

TEST 4: Applicazione DNAT



TEST 4: Applicazione SNAT



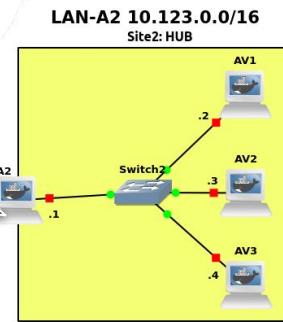
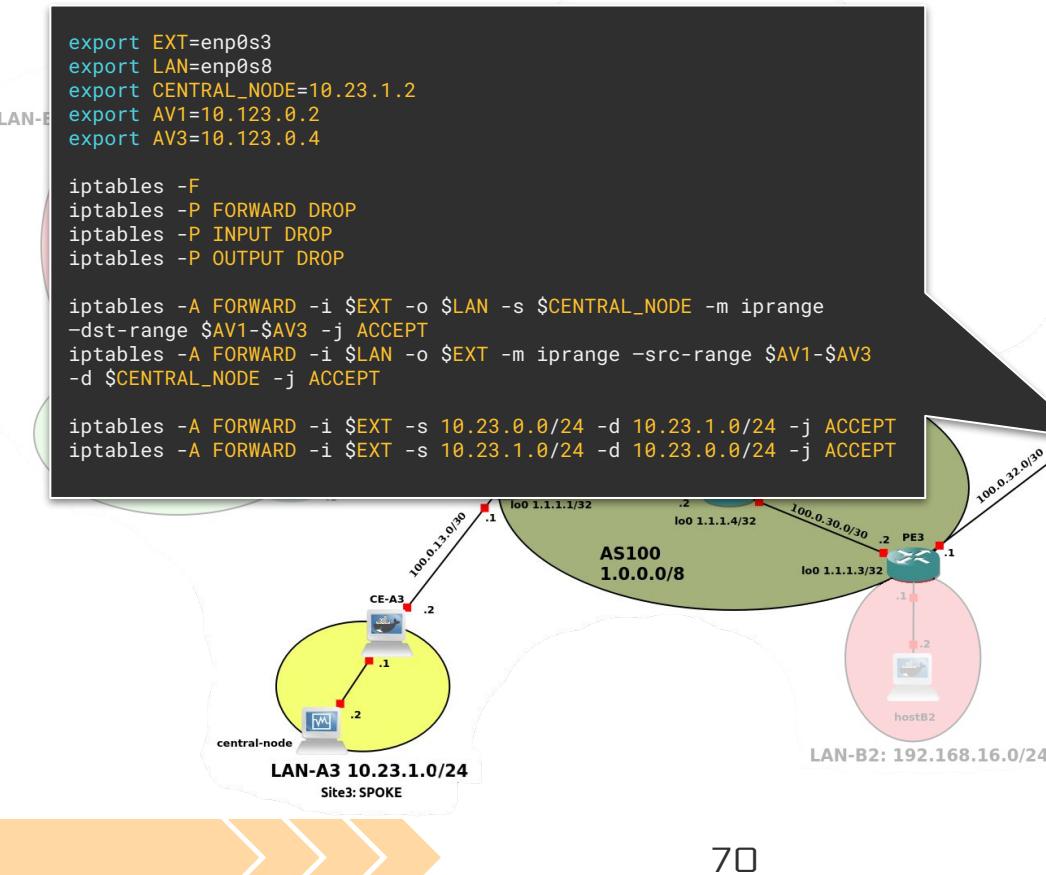
Configurazione: CE-A2

```
export EXT=enp0s3
export LAN=enp0s8
export CENTRAL_NODE=10.23.1.2
export AV1=10.123.0.2
export AV3=10.123.0.4

iptables -F
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT DROP

iptables -A FORWARD -i $EXT -o $LAN -s $CENTRAL_NODE -m iprange
-dst-range $AV1-$AV3 -j ACCEPT
iptables -A FORWARD -i $LAN -o $EXT -m iprange -src-range $AV1-$AV3
-d $CENTRAL_NODE -j ACCEPT

iptables -A FORWARD -i $EXT -s 10.23.0.0/24 -d 10.23.1.0/24 -j ACCEPT
iptables -A FORWARD -i $EXT -s 10.23.1.0/24 -d 10.23.0.0/24 -j ACCEPT
```



1. Permettere comunicazione bidirezionale tra central node e AVs e negare tutto il resto

- Di default si scarta tutto il traffico che passa per il GW.
- Aggiungiamo una regola che abilita il forwarding dei pacchetti che dalla interfaccia connessa a LAN-A2 sono diretti verso l'esterno e viceversa, con le sole macchine abilitate al forwarding che sono il central node e gli AVs.
 - Il pacchetto passerà per il *forward hook*.
 - Questa soluzione è più stringente rispetto all'uso delle intere reti, infatti se aggiungessimo macchine in LAN-A2 e LAN-A3 queste non potrebbero comunicare.

```
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -A FORWARD -i $EXT -o $LAN -s $CENTRAL_NODE -m iprange --dst-range $AV1-$AV3 -j ACCEPT
iptables -A FORWARD -i $LAN -o $EXT -m iprange --src-range $AV1-$AV3 -d $CENTRAL_NODE -j ACCEPT
```

2. Permettere la comunicazione tra gli spokes

- Di default si scarta tutto il traffico che passa per il GW per la prima regola.
- Aggiungiamo una regola che abilita il forwarding dei pacchetti che dall'interfaccia connessa all'esterno sono diretti verso LAN-A2, specificando che le macchine abilitate al forwarding sono quelle in LAN-A1 e LAN-A3, ovvero i siti degli spokes.
 - Il pacchetto passerà per l'*input hook*
 - Questa soluzione garantisce la comunicazione tra gli spokes passando per l'hub, mantenendo la caratteristica principale della topologia della VPN.

```
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -A FORWARD -i $EXT -o $LAN -s $CENTRAL_NODE -m iprange --dst-range $AV1-$AV3 -j ACCEPT
iptables -A FORWARD -i $LAN -o $EXT -m iprange --src-range $AV1-$AV3 -d $CENTRAL_NODE -j ACCEPT
```

TEST CONFIGURAZIONE FIREWALL CE-A2

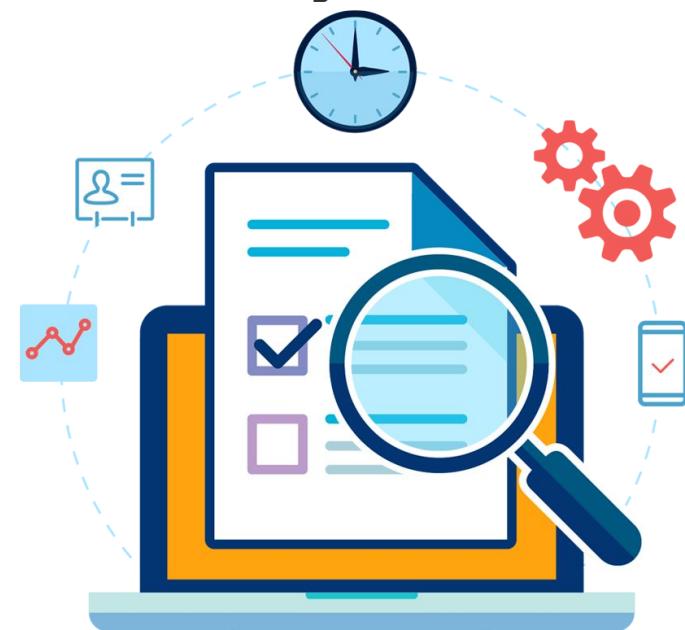
Per testare la configurazione del firewall di CE-A2 sono stati effettuati i seguenti test:

TEST 1: Connessione CentralNode <-> AV1

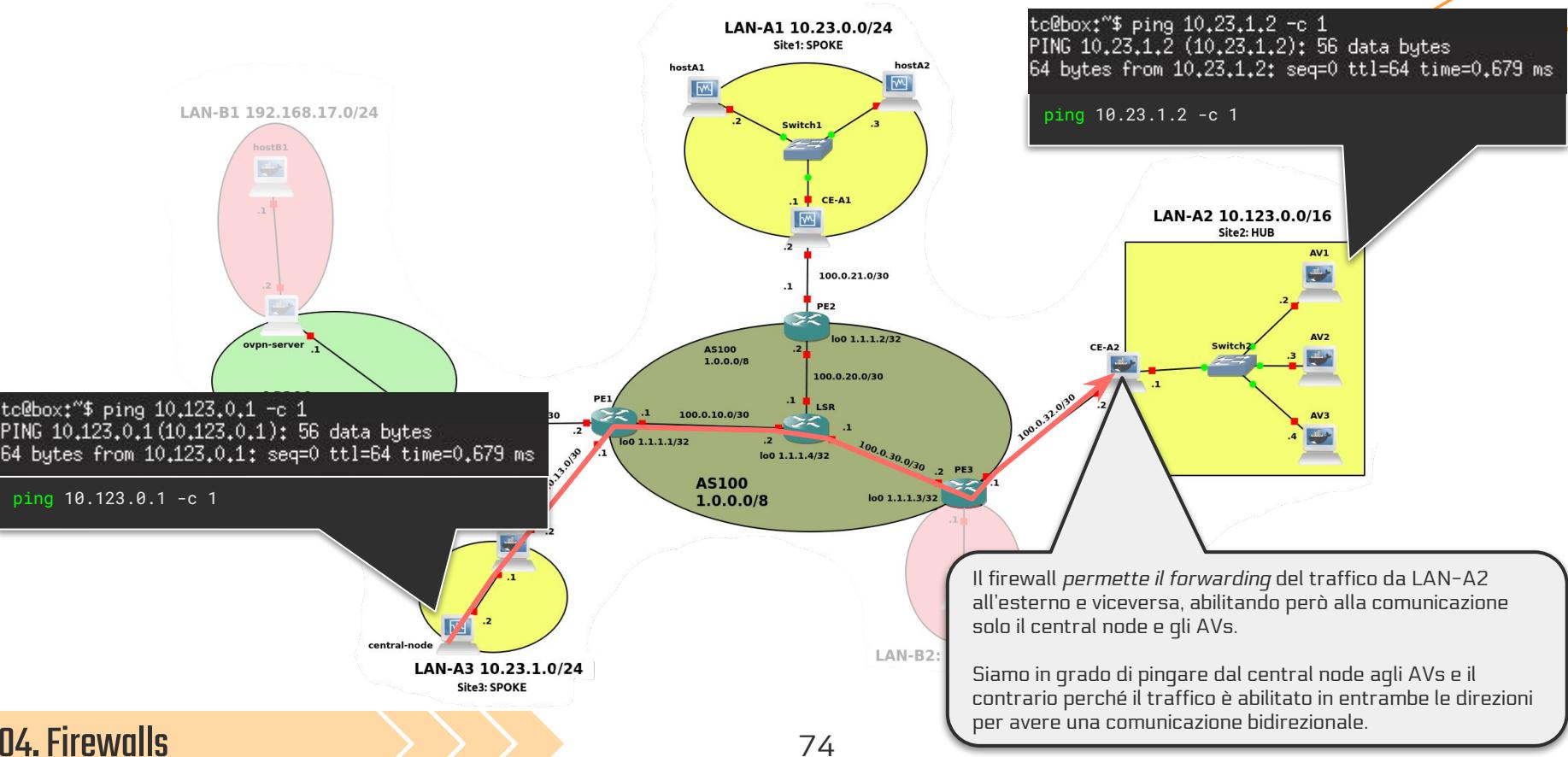
- Verifica ping ICMP CentralNode <-> AV1

TEST 2: Connessione CentralNode <-> HostA1

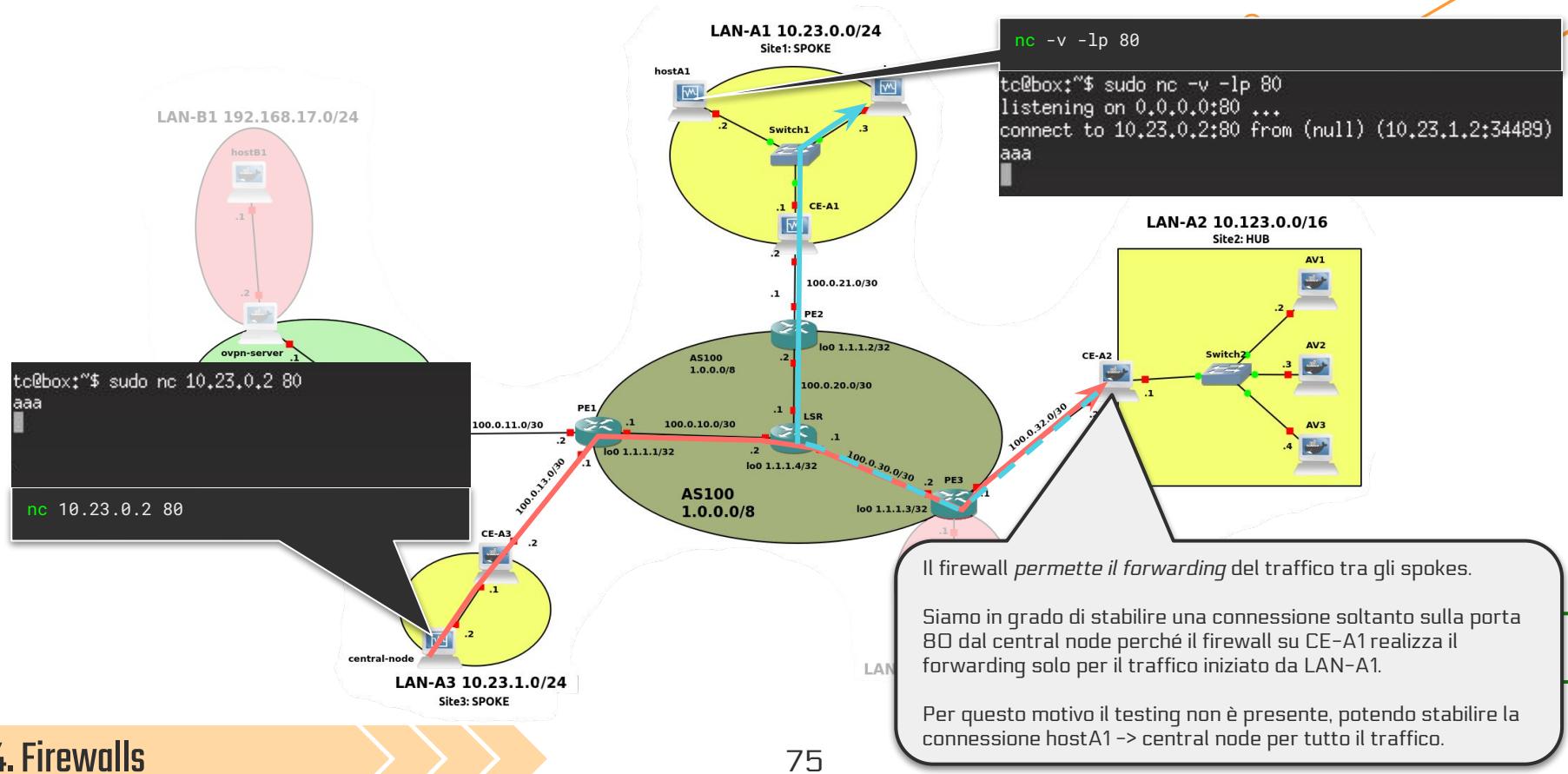
- Verifica di connessione CentralNode → HostA1
solo per HTTP

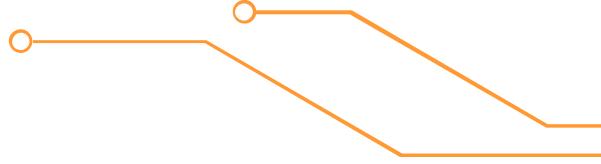


TEST 1: Connessione CentralNode <-> AV1



TEST 2: Connessione CentralNode <-> HostA1

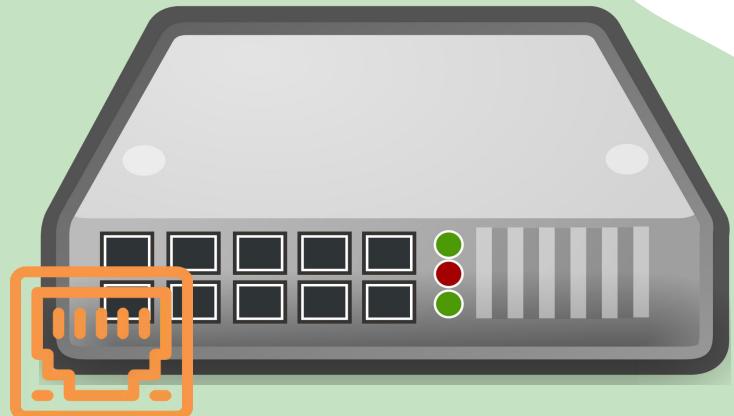




05

MACsec

Implementazione canale di
comunicazione sicuro attraverso
protocollo MACsec



Specifica

- Implementare un canale di comunicazione sicuro tramite protocollo MACsec tra i membri della LAN-A1.
- Implementare la distribuzione della Security Association Key tramite MACsec Key Agreement in CAK Static Mode.

Configurazione

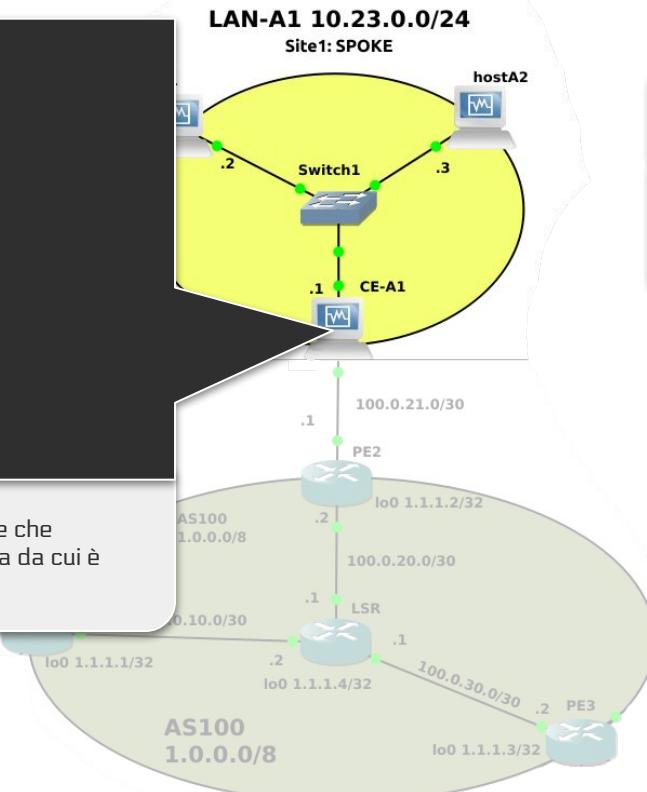
```
## MACsec

nmcli connection del macsec-connection

nmcli connection add type macsec \
con-name macsec-connection \
ifname macsec0 \
connection.autoconnect no \
macsec.parent enp0s8 \
macsec.mode psk \
macsec.mka-cak $MKA_CKA \
macsec.mka-cak-flags 0 \
macsec.mka-ckn $MKA_CKN \
ipv4.method manual \
ipv4.addresses 10.23.0.1/24

nmcli connection up macsec-connection
```

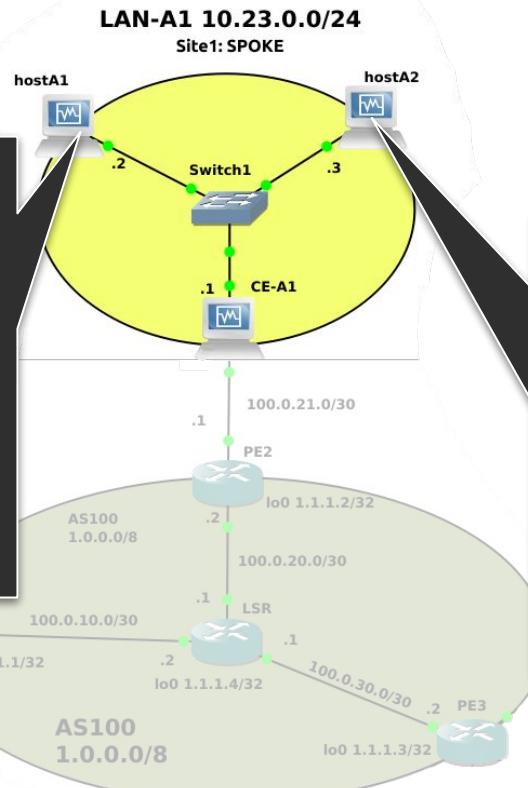
Specificiamo il nome dell'interfaccia virtuale che realizzerà il canale sicuro e l'interfaccia fisica da cui è ospitata.



```
export MKA_CKA=aaaabbbcccccddd1234567812345678...
export MKA_CKN=00001111222233334444555566667777...
```

Connectivity Association Key e Connectivity Key Name per la realizzazione del Key Agreement in CAK Static Mode

Configurazione



```
## MACsec  
nmcli connection del macsec-connection  
  
nmcli connection add type macsec \  
con-name macsec-connection \  
iface macsec0 \  
connection.autoconnect no \  
macsec.parent enp0s3 \  
macsec.mode psk \  
macsec.mka-cak $MKA_CKA \  
macsec.mka-cak-flags 0 \  
macsec.mka-ckn $MKA_CKN \  
ipv4.method manual \  
ipv4.addresses 10.23.0.2/24  
  
nmcli connection up macsec-connection
```

```
export MKA_CKA=aaaabbbcccccddd1234567812345678  
export MKA_CKN=00001111222233334444555566667777...
```

```
## MACsec  
nmcli connection del macsec-connection  
  
nmcli connection add type macsec \  
con-name macsec-connection \  
iface macsec0 \  
connection.autoconnect no \  
macsec.parent enp0s3 \  
macsec.mode psk \  
macsec.mka-cak $MKA_CKA \  
macsec.mka-cak-flags 0 \  
macsec.mka-ckn $MKA_CKN \  
ipv4.method manual \  
ipv4.addresses 10.23.0.3/24  
  
nmcli connection up macsec-connection
```

Message Exchange

1393	PcsCompu_75:a8:16	Nearest-non-TPMR-bridge	EAPOL-MKA Key Server	Live Peer List	Distributed SAK
1394	PcsCompu_75:a8:16	Nearest-non-TPMR-bridge	EAPOL-MKA Key Server, Live Peer List		MACsec SAK Use, Distributed SAK
1395	PcsCompu_9e:bb:67	Nearest-non-TPMR-bridge	EAPOL-MKA Live Peer List, MACsec SAK Use		
1396	PcsCompu_9e:bb:67	Nearest-non-TPMR-bridge	EAPOL-MKA Live Peer List, MACsec SAK Use		

```
Key Server Priority: 255
1... .... = Key Server: True
.1... .... = MACsec Desired: True
..10 .... = MACsec Capability: MACsec Integrity with no confidentiality offset (2)
.... 0000 0011 1100 = Parameter set body length: 60
SCI: 08002775a8160001
Actor Member Identifier: 151411bb3ed83164c4697cf1
Actor Message Number: 00000002
Algorithm Agility: IEEE Std 802.1X-2010 (0x0080c201)
CAK Name: 00001112223333444555566667778888999001122334455667788990123
» Live Peer List Parameter set
- Distributed SAK parameter set
    Parameter set type: Distributed SAK (4)
    00... .... = Distributed AN: 0
    ..01 .... = Confidentiality Offset: No confidentiality offset (1)
    .... 0000 0001 1100 = Parameter set body length: 28
    Key Number: 00000001
    AES Key Wrap of SAK: 7b779e655d15214097cbcfc46069be2ec104a09c96475321
    Integrity Check Value: 318330T5DTT9Tb94184D/acc16656608
```

SAK Distribuita all'interno della Security Association

```
usr@CE-A1:/# ip macsec show
```

```
macsec0: protect on validate strict sc off sa off encrypt on ...
cipher suite: GCM-AES-128, using ICV length 16
TXSC: 0800279ebb670001 on SA 0
    0: PN 35, state on, key 1644879dccd14b006a6720fa01000000
RXSC: 0800279ebb670001, state on
    0: PN 47, state on, key 1644879dccd14b006a6720fa01000000
RXSC: 0800279ebb670001, state on
    0: PN 29, state on, key 1644879dccd14b006a6720fa01000000
```

TEST CONFIGURAZIONE MACsec

Per testare la configurazione MACsec sono stati effettuati i seguenti test:

TEST 1: Ping HostA2 -> HostA1

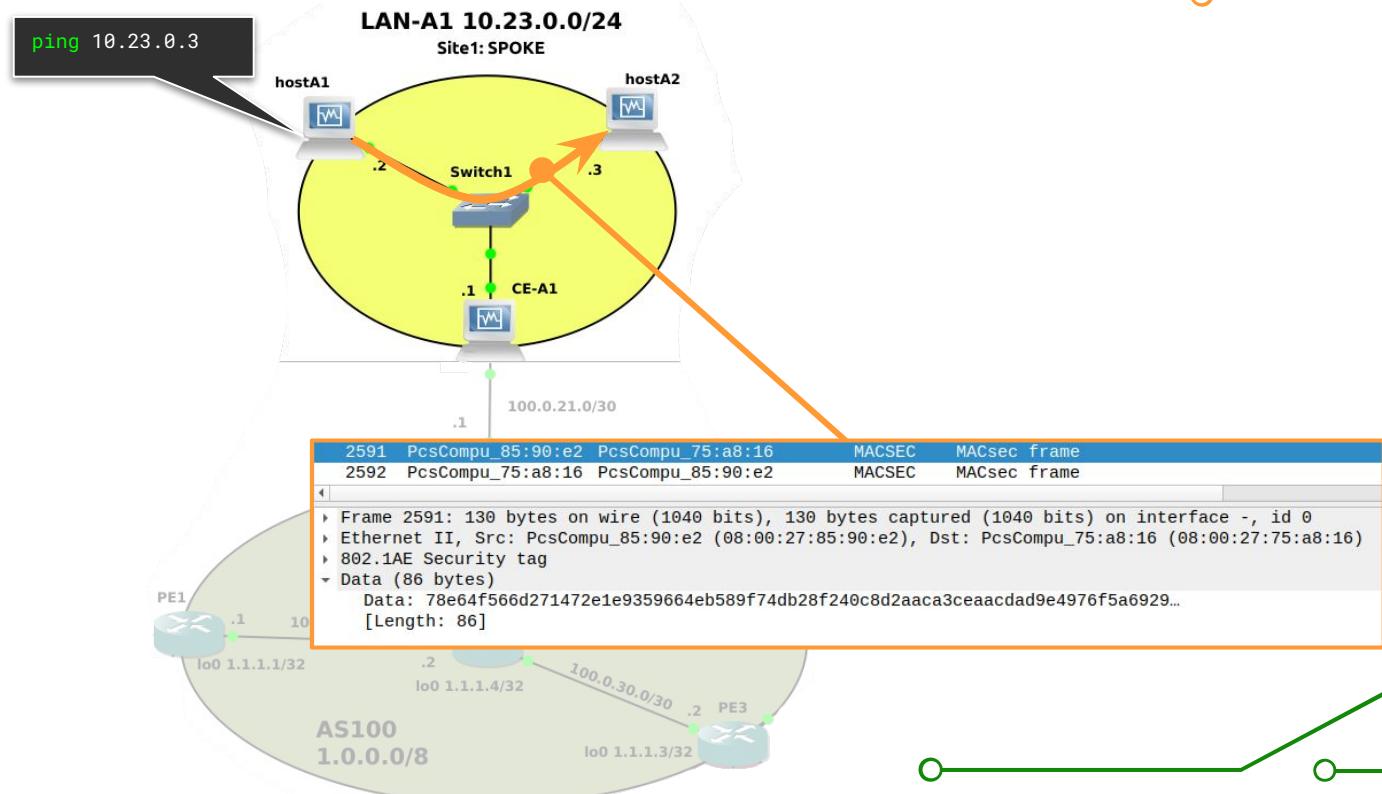
- Verifica Ping ICMP verso 10.23.0.2
- Verifica encapsulamento in MACsec Frame

TEST 2: Ping HUB -> LAN-A1 Spoke

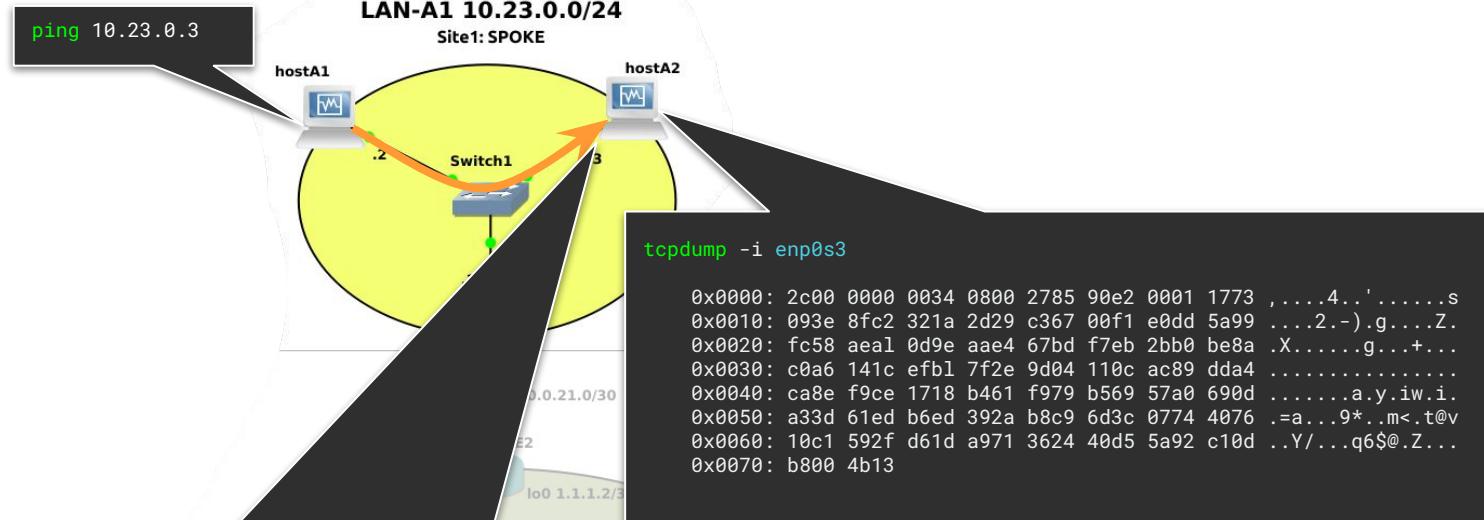
- Verifica encapsulamento in MACsec Frame all'ingresso della LAN-A1



TEST 1: Ping HostA1 -> HostA2



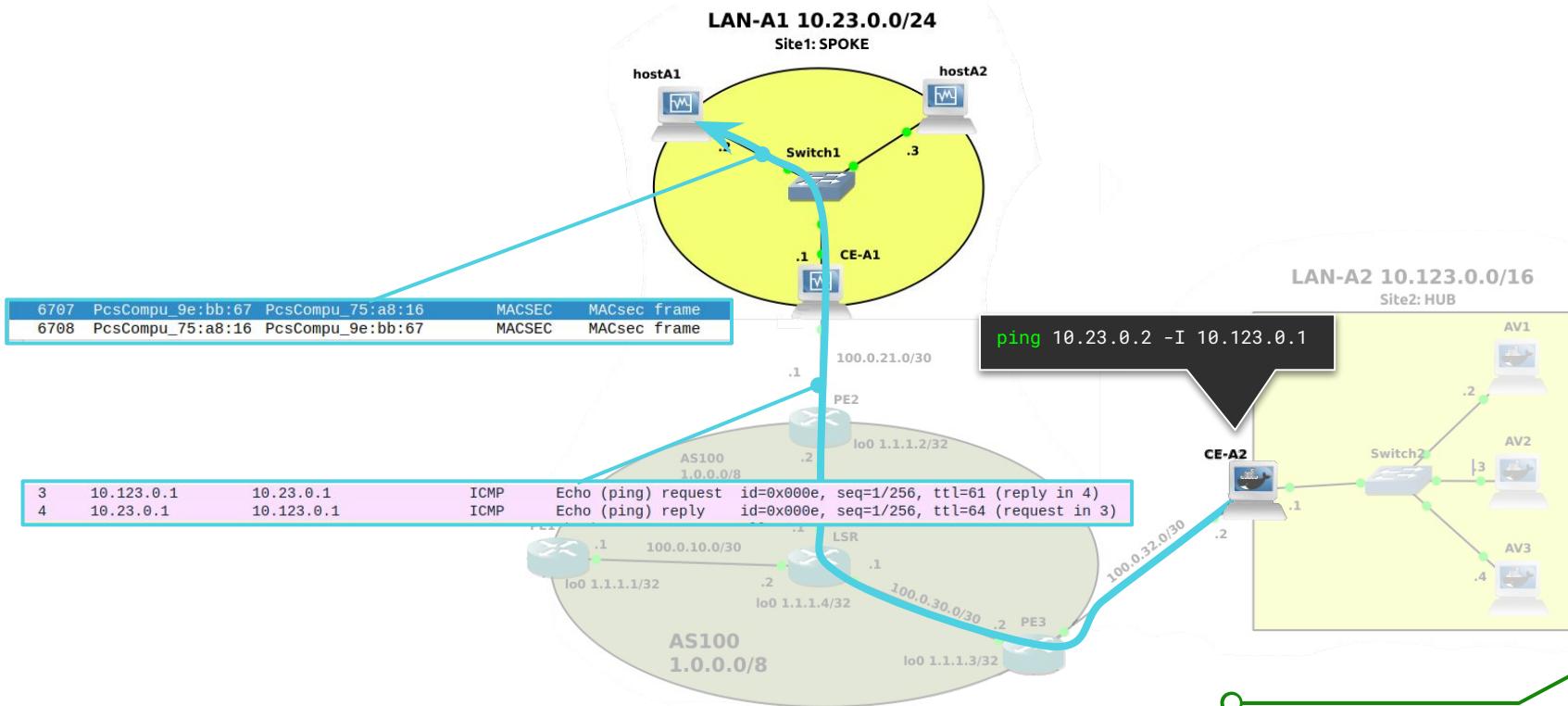
TEST 1: Ping HostA1 -> HostA2



```
tcpdump -i macsec0
00:05:30.528847 IP 10.23.0.2 > 10.23.0.3: ICMP echo request, id 9, seq 1, length 64
00:05:30.528879 IP 10.23.0.3 > 10.23.0.2: ICMP echo reply, id 9, seq 1, length 64
```

AS100
1.0.0.0/8

TEST 2: Ping HUB -> LAN-A1 Spoke





06

Antivirus

Configurazione e implementazione di ambienti di analisi per file binari



Specifica: Central Node

Il central-node nel sito 3 consente di scaricare un eseguibile e di distribuirlo ai vari nodi di test, che analizzeranno/eseguiranno gli eseguibili e forniranno i risultati al nodo centrale.

Il nodo centrale, dopo aver ricevuto i tre report dai nodi worker, crea un report finale per l'utente, mostrando quali minacce (se presenti) sono state scoperte nel binario.

Se l'eseguibile risulta pericoloso viene mostrato all'utente un prompt con le minacce rilevate, offrendo la possibilità di tenere o rimuovere l'eseguibile malevolo. Il report finale viene infine inviato verso un web server nella LAN-A1



Iubuntu

Interfaccia `enp0s3` connessa con
il customer edge

Interfaccia `enp0s10` connessa al
NAT per scaricare gli eseguibili da
internet





Specifiche: Nodi Worker



Ognuno dei tre nodi worker hosta un diverso antivirus, in modo da poter osservare caratteristiche e comportamenti differenti sullo stesso eseguibile:



AV1: sixxpain/clamav

Effettua una scansione degli eseguibili ricevuti tramite il tool **clamscan**.

Genera una signature dell'eseguibile e lo confronta con oltre 8 600 000 malware noti presenti nel suo database.

Permette di analizzare sia file PE che ELF, ritornando inoltre il tipo di minaccia riscontrata



AV2: sixxpain/jmdav

Effettua l'analisi statica su header e sezioni dell'eseguibile:

- **readelf** per i file ELF
- **lief** per i file PE

Verifica la presenza di eventuali packers

Nel caso degli ELF effettua analisi dinamica per verificare se l'eseguibile ottiene permessi sospetti.



AV3: sixxpain/rkhav

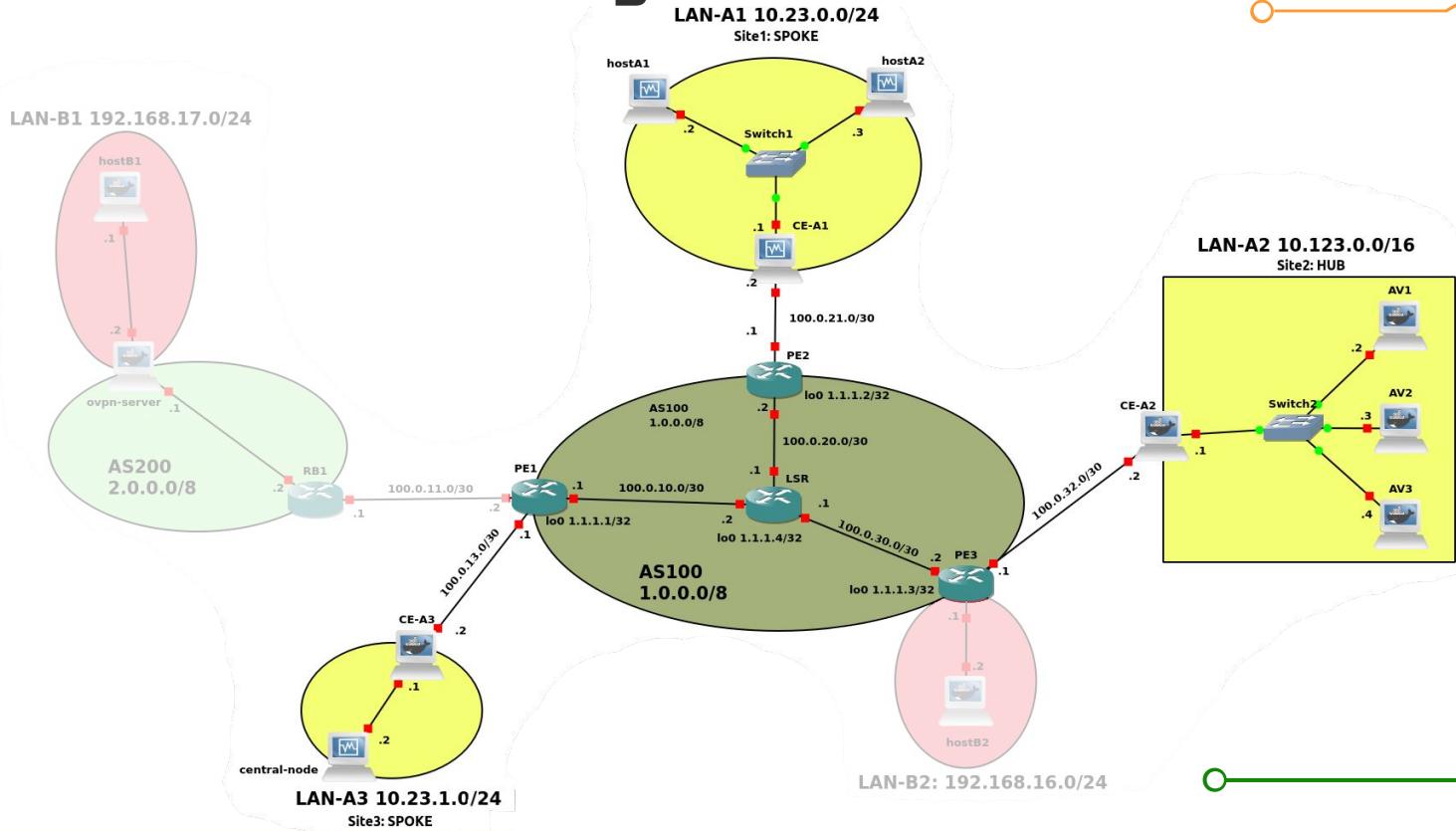
Effettua l'analisi dinamica del malware, eseguendolo sulla sandbox.

Tramite rkhunter si individua l'installazione di eventuali rootkit, ed in generale modifiche sul sistema introdotte dopo l'esecuzione del programma

Si confronta il log dopo ogni run con un log di base effettuato sulla macchina a clean-state

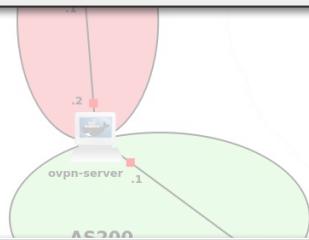


Configurazione di Rete

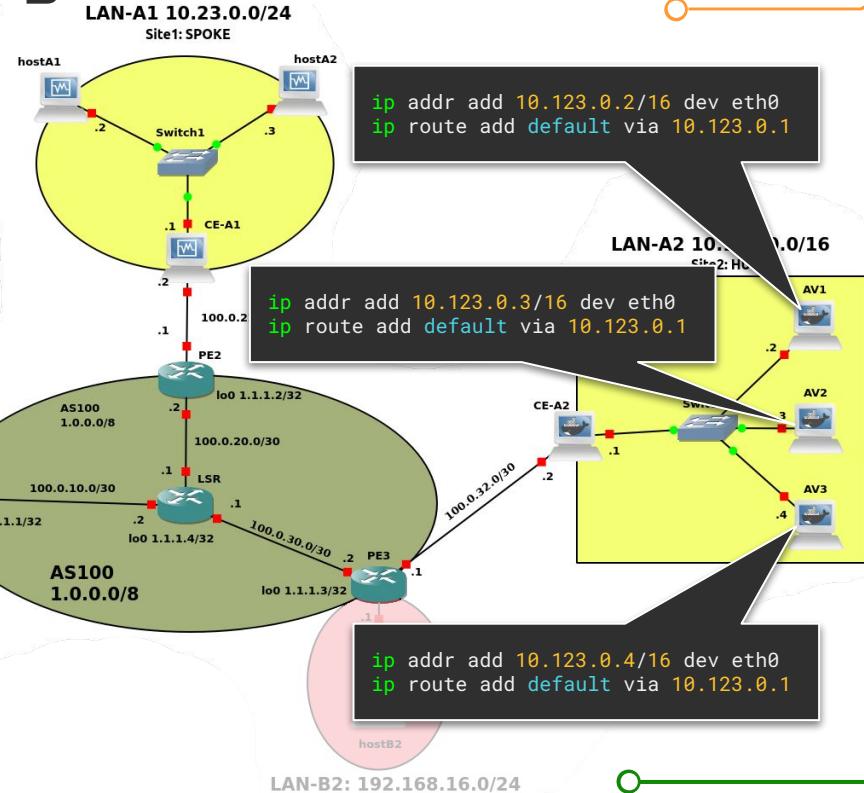
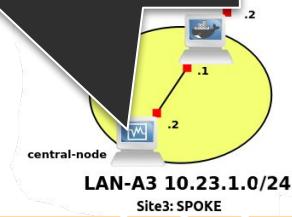


Configurazione di Rete

Ogni nodo dell'architettura funge sia da client che da server, per inviare / ricevere l'eseguibile ed il report a seguito dell'analisi



```
sudo ip route del 0/0
sudo ip addr add 10.23.1.2/24 dev enp0s3
sudo ip route add 10.123.0.0/16 via 10.23.1.1
sudo ip route add 10.23.0.0/24 via 10.23.1.1
```



Central Node: Listening Service

Il Central-Node è una macchina virtuale utilizzata da un utente comune, in cui è stato installato un **servizio** per la scansione e l'analisi remota di file sospetti.

L'*agente locale* dell'antivirus monitora le cartelle specificate nel file .conf/to_scan tramite il servizio **inotifywait**.

Ogni volta che viene trovato un nuovo file questo viene spostato nella cartella av/quarantine, e poi inviato verso i tre AV remoti, tramite lo script **remote_analysis.py**

```
File: central-node/start.sh
echo "Starting AV Listening Service.."
inotifywait -m $(cat .conf/to_scan) -e create -e moved_to |
    while read dir action file; do
        echo "The executable '$file' is received in directory '$dir' via
        '$action'"
        path=$dir$file
        q_path=../av/quarantine/$file
        mv $path $q_path
        python3 ./av/remote_analysis.py $q_path $path
    done
```

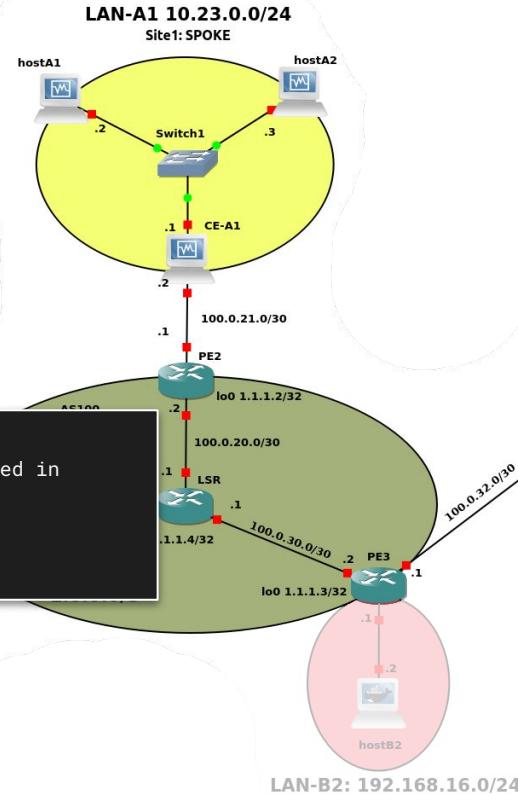
Scansione Nuovo File

Il central node scarica un file da internet sfruttando l'interfaccia NAT.

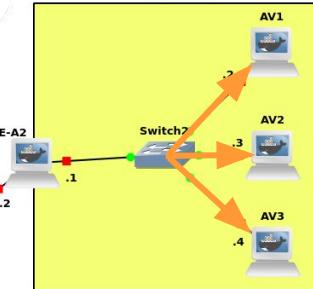
Se il file viene scaricato su una delle cartelle monitorate viene attivato l'evento di inotifywait, ed il file viene quindi processato da remote_analysis.py

```
> The Executable test_mw.exe has been received in  
directory /Scaricati  
  
> Starting Remote Analysis...
```

GET somesite.org/test_mw.exe
200: test_mw.exe
central-node



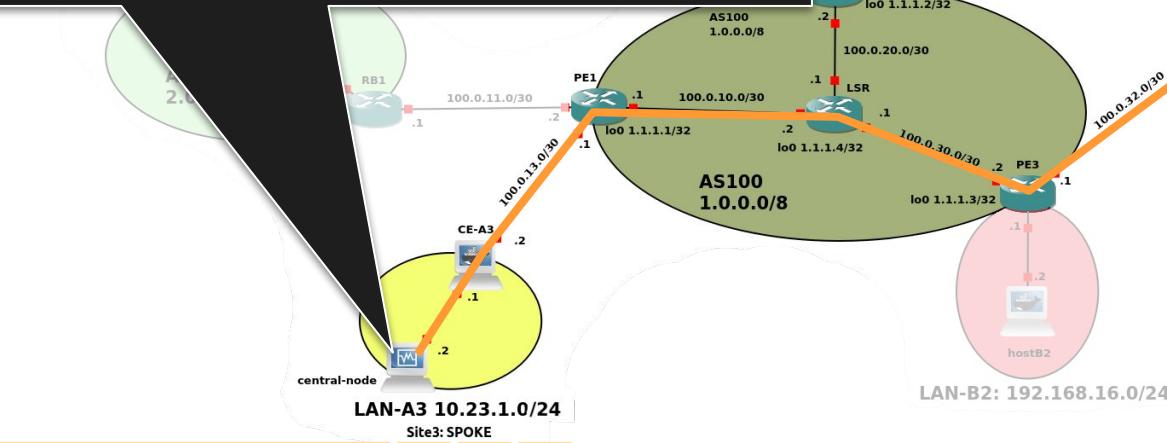
LAN-A2 10.123.0.0/16 Site2: HUB



Invio Malware

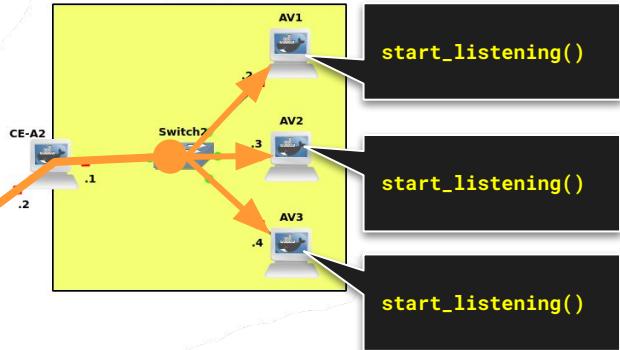
L'invio del file ai nodi worker avviene in parallelo tramite 3 thread differenti.

```
av1_thr = threading.Thread(target=send_file, args=[  
    file_path, file_name, AV1_IP, AV1_REP_PORT])  
av2_thr = threading.Thread(target=send_file, args=[  
    file_path, file_name, AV2_IP, AV2_REP_PORT])  
av3_thr = threading.Thread(target=send_file, args=[  
    file_path, file_name, AV3_IP, AV3_REP_PORT])
```



Ogni worker si mette in ascolto per la ricezione dei file su una socket dedicata.

LAN-A2 10.123.0.0/16
Site2: HUB



Invio Malware: Client/Server API

File: `central-node/remote_analysis.py`

```
def send_file(file_path, file_name, dest_ip, rep_port):  
  
    if rep_port == AV1_REPORT_PORT:  
        lock = lock_av1  
    elif rep_port == AV2_REPORT_PORT:  
        lock = lock_av2  
    elif rep_port == AV3_REPORT_PORT:  
        lock = lock_av3  
    lock.acquire()  
  
    # Initialize Socket & Connect to Host  
    sock = socket.socket()  
    sock.connect((dest_ip, rep_port))  
  
    # Send filename to the server, and wait for ACK to continue  
    sock.send(file_name.encode())  
    sock.recv(2) # Receive ACK  
  
    # Read File in binary  
    file = open(file_path, 'rb')  
    line = file.read(1024)  
  
    # Keep sending data to the server  
    while(line):  
        time.sleep(CONGESTION_SLOWDOWN)  
        sock.send(line)  
        line = file.read(1024)  
  
    file.close()  
    sock.close()  
  
    # Listening for Report after Malware it's successfully sent  
    start_listening(rep_port, lock)  
    return 0
```

Ogni thread prima di inviare il file acquisisce un apposito lock.

Il lock verrà rilasciato solo dopo che ha ricevuto il report dall'antivirus remoto.

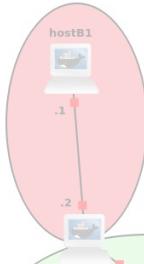
Evita la congestione dei router cisco per l'invio di grandi file.
[vedi limitazioni]

File: `av/start_av.py`

```
def start_listening():  
    # Initialize Socket Instance  
    sock = socket.socket()  
  
    # binding to the host and port  
    sock.bind((HOST_IP, REPORT_PORT))  
    sock.listen(10)  
  
    while True:  
  
        # Establish connection with the clients.  
        con, addr = sock.accept()  
  
        # Get filename from the client & Send ACK  
        data = con.recv(1024)  
        file_name = data.decode()  
        con.send("OK".encode())  
  
        # Write File in binary  
        file_path = MALWARE_DIR+file_name  
        file = open(file_path, 'wb')  
  
        # Keep receiving data from the client  
        line = con.recv(1024)  
  
        while (line):  
            file.write(line)  
            line = con.recv(1024)  
  
        file.close()  
        con.close()  
        analyze_file(file_path, file_name)
```

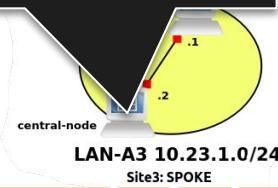
Malware Analysis

LAN-B1 192.168.17.0/24



Dopo aver inviato il malware il central-node si mette in attesa di ricevere i report da tutti i nodi worker

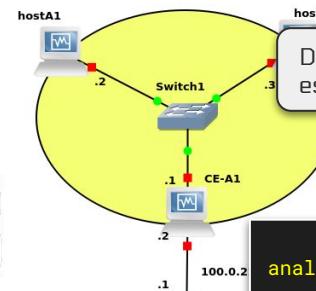
```
av1_thr: start_listening(lock_av1,AV1_REP_PORT)  
av2_thr: start_listening(lock_av2,AV2_REP_PORT)  
av3_thr: start_listening(lock_av3,AV3_REP_PORT)
```



LAN-A3 10.23.1.0/24
Site3: SPOKE

LAN-A1 10.23.0.0/24

Site1: SPOKE

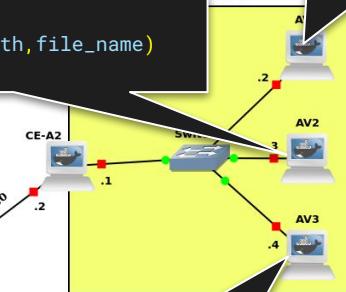


Dopo aver ricevuto il malware ogni nodo worker esegue il suo antivirus locale per analizzare il file.

```
analyze_file(file_path,file_name)
```

LAN-A2 10.123.0.0/1

Site2: HUB



```
analyze_file(file_path,file_name)
```

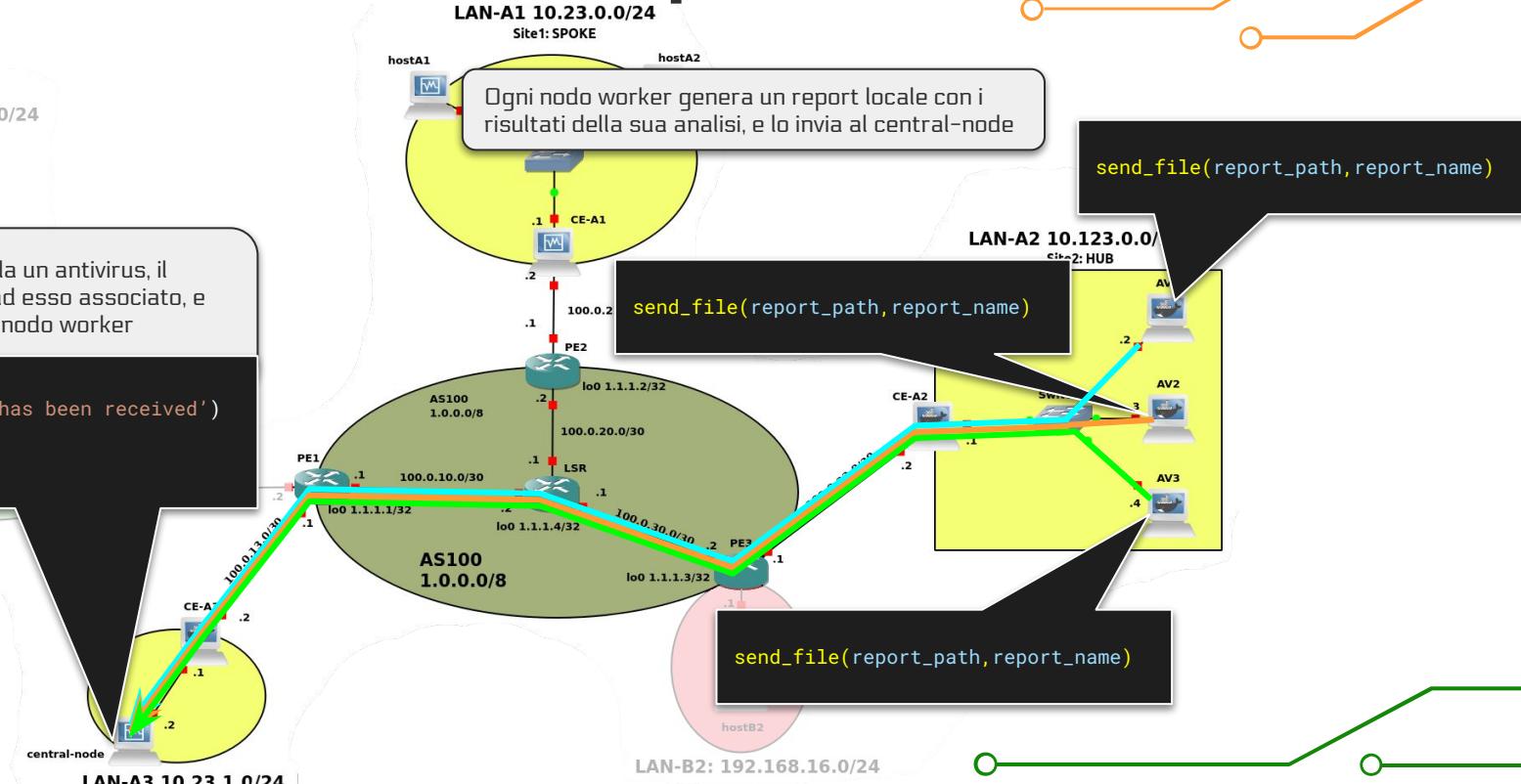
LAN-B2: 192.168.16.0/24

Invio Report

LAN-B1 192.168.17.0/24

Una volta ricevuto il report da un antivirus, il central node rilascia il lock ad esso associato, e chiude la connessione con il nodo worker

```
...  
print(f'{port}: Report has been received')  
file.close()  
con.close()  
lock.release()
```



Report e Decisione Globale

Il central-node cerca di acquisire i tre lock, in modo che solo dopo aver ricevuto tutti i report prosegua con la valutazione della minaccia.

```
# Make decision on file only after all AVs reports has been received
lock_av1.acquire()
lock_av2.acquire()
lock_av3.acquire()

# Generates the aggregate reports, end evaluates if globally the file is dangerous or not
aggregate_rep_path = aggregate_reports(file_name)
make_decision(aggregate_rep_path, MALWARE_DIR+file_name, original_path)
```

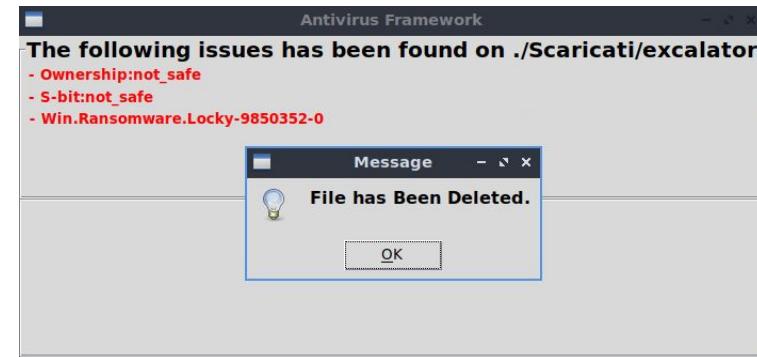
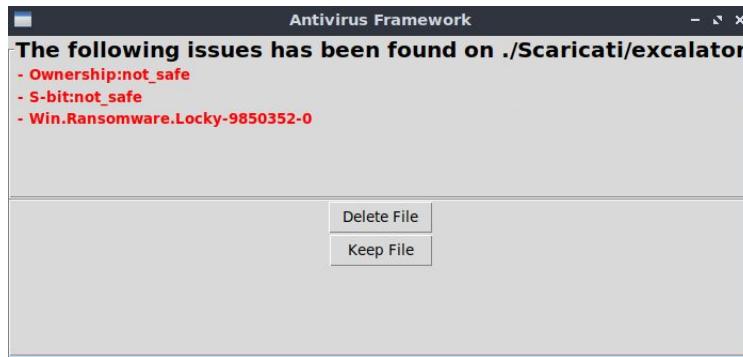
In **make_decision()** si esamina il report globale generato dalle tre analisi remote, ed in base ai risultati si stabilisce se il file è una potenziale minaccia.

```
if ("FOUND" in result_av1):
    threats_found.append(result_av1[1])
for entry in result_av2:
    if ("not_safe" in entry):
        threats_found.append(entry)
for diff in result_av3:
    if ("Found" in diff or "Warning" in diff):
        threats_found.append(diff)
if threats_found != []:
    ask_user(file_path, threats_found, original_path)
```

Report e Decisione Globale

Se il file è stato classificato come "Pericoloso" viene mostrato all'utente un prompt (realizzato con **Tkinter**) dove vengono mostrate le minacce rilevate a seguito dell'analisi dei tre AVs.

- L'utente del Central Node può quindi valutare in autonomia se mantenere o cancellare il file sospetto



Upload Report to Remote Server

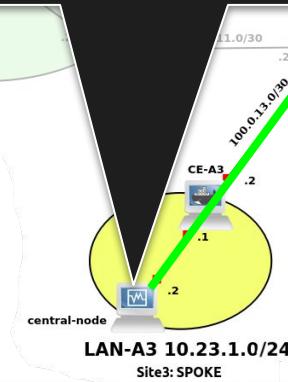
LAN-B1 192.168.17.0/24

Come ultimo step il report globale viene inviato verso il server remoto, mantenuto da HostA2 nella LAN-A1.

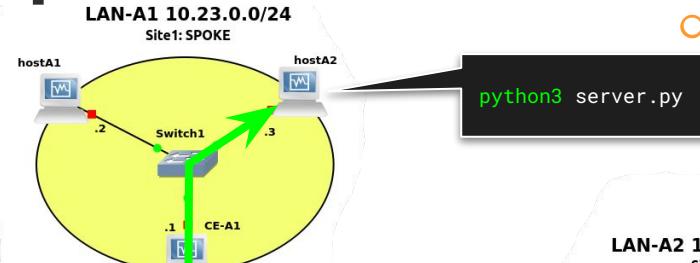
Possiamo specificare come destinazione un qualsiasi indirizzo di 10.23.0.0/24, in quanto CE-A1 utilizza il DNAT verso HostA2.

```
f = open(report, 'rb')
requests.put('http://10.23.0.69:80', data=f.read())
print("Analysis report sent to web servers.")
```

AS200
2.0.0.0/8

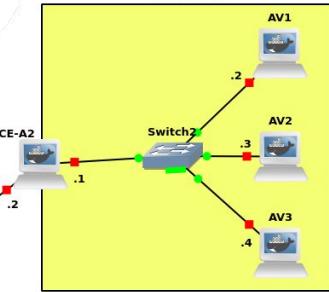


LAN-A3 10.23.1.0/24
Site3: SPOKE



LAN-A1 10.23.0.0/24
Site1: SPOKE

LAN-A2 10.123.0.0/16
Site2: HUB



LAN-B2: 192.168.16.0/24

Web Server Report Hosting

Il Web Server è implementato tramite **HttpServer** di Python, e sfrutta la libreria **jinja2** per renderizzare una pagina HTML tramite i dati ricavati dal parsing dei report ricevuti.

In questo modo, raggiungendo il web server da qualsiasi host in LAN-A3, si ha la possibilità di visualizzare lo storico di tutte le scansioni effettuate, con i relativi report dettagliati da poter scaricare.

Program Name	Security	Data scanned	Time	Start Date	End Date	Details
/av/quarantine/excalator	WARNING	0.01 MB	41.896 sec (0 m 41 s)	2023:02:21 17:06:51	2023:02:21 17:07:33	Download Analysis
/av/quarantine/LoveYou.exe	OK	0.02 MB	30.606 sec (0 m 30 s)	2023:02:21 17:05:53	2023:02:21 17:06:24	Download Analysis

CLAMAV

ClamAV è un antivirus open source per il rilevamento di trojan, virus e malware noti all'interno del sistema posto sotto analisi.

Il suo funzionamento si basa sull'utilizzo di un *signature database* di malware noti molto ampio, circa 9 milioni di signature.

L'analisi dell'eseguibile nel container AV1 viene effettuata semplicemente lanciando il comando **clamscan sul file ricevuto dal central node**, redirezionando l'output sul file di report.

```
File: clamav/clamav.py

def analyze_file(file_path, file_name):

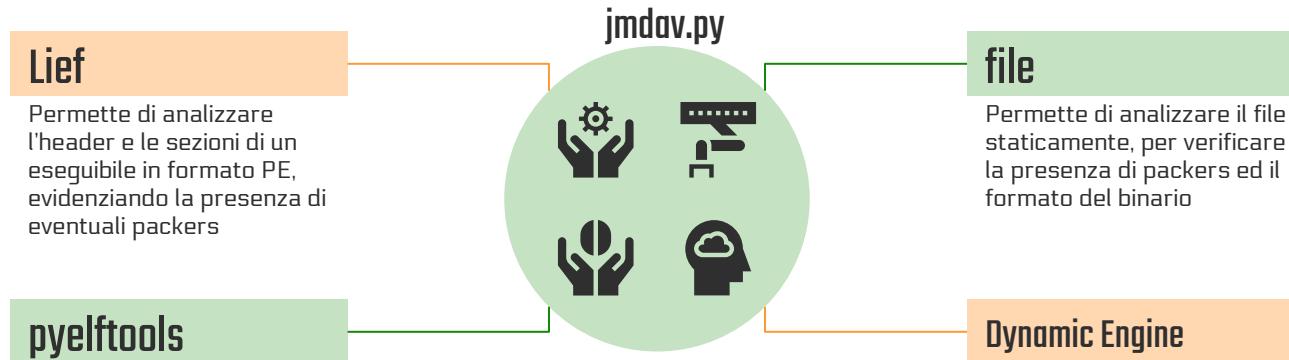
    report_path = REPORT_DIR+file_name+REPORT_SUFFIX
    report_name = file_name+REPORT_SUFFIX
    os.system(f"clamscan {file_path} > {report_path}")
    os.system(f"rm {file_path}")
    send_file(report_path, report_name)
```

```
/av/quarantine/excalator: Win.Ransomware.Locky-9850352-0
-----
SCAN SUMMARY -----
Known viruses: 8653189
Engine version: 0.103.6
Scanned directories: 0
Scanned files: 1
Infected files: 1
Data scanned: 0.01 MB
Data read: 0.01 MB (ratio 1.00:1)
Time: 54.879 sec (0 m 54 s)
Start Date: 2023:02:22 15:38:16
End Date: 2023:02:22 15:39:10
```

JMD AV

Il secondo container dedicato all'analisi dei binari utilizza la scansione definita nel file **jmdav.py**.

Questo contiene un insieme di tools per effettuare un'analisi statica (*ELF e PE*), ed un'analisi dinamica di base (*soltanto degli ELF*).



JMD AV



File: [jmdav/jmdav.py](#)

```
def analyze_file(file, report):
...
    if pattern.ELF_FORMAT in file_info:
        os.system(f"chmod 777 {file_path} ;./{file_path}")
        check_ownership(file_path)
        check_capabilities(file_path)
        check_section(file_info)
        check_sbit(file_path)

    elif pattern.EXE_FORMAT in file_info:
        check_packers(pe_binary)
        pe_binary = lief.parse(file_path)
...

```

"Ownership": "not_safe",
"S-bit": "not_safe",
"Capabilities": "safe",
"Sections": "safe",
"Packers": "safe"

ELF Header:
Magic: 7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
Class: ELF64
Data: 2's complement, LE
Version: 1 (current)
OS/ABI: UNIX - System V
ABI Version: 0
Type: DYN
Machine: Advanced Micro Devices
X86-64
Version: 0x1
Entry point address: 0x630
...



JMD AV



File: jmdav/jmdav.py

```
def check_ownership(file):
    owner = getpwuid(os.stat(file).st_uid).pw_name
    if owner == "root":
        summary_lin["Ownership"] = pattern.NOT_SAFE

def check_capabilities(file):
    caps = os.popen(f"getcap {file}").read()
    if caps != "":
        summary_lin["Capabilities"] = pattern.NOT_SAFE + " = " + caps[:-1]

def check_sbit(file):
    sbit = oct(os.stat(file).st_mode)[-4:][0]
    if int(sbit) >= 4:
        summary_lin['S-bit'] = pattern.NOT_SAFE
```

File: jmdav/jmdav.py

```
packers_sections = {
    # The packer/protector/tools section names/keywords
    '.aspack': 'Aspack packer',
    '.adata': 'Aspack packer/Armadillo packer',
    'UPX0': 'UPX packer',
    'UPX1': 'UPX packer',

def check_packers(binary):
    suspect_sections = []
    safe = True

    for section_name in binary.sections:
        if section_name.name in packers_sections.keys():
            safe = False
            suspect_sections.append(section_name.name)
```



RKHunter AV

Rootkit Hunter scansiona il sistema alla ricerca di possibili rootkit, backdoor o ulteriori vulnerabilità presenti tramite una ricerca di misconfigurazioni dei permessi, presenza di directory utilizzate da rootkit e di moduli kernel contenenti stringhe sospette.

Ad ogni startup del container dedicato, RKHunter esegue una prima scansione del sistema per creare una **baseline analysis**.

Tale analisi verrà confrontata ad ogni nuova analisi per determinare se il sistema è stato compromesso a seguito dell'esecuzione del binario sospetto.

```
File: rkhav/start.sh
echo "Creating Baseline Report.."
rm /var/log/rkhunter*
rkhunter --check --sk --nocolors > /var/log/rkhunter_baseline.log
echo "Initialization Completed"
```

RKHunter AV

Alla ricezione del binario, se possibile, esso verrà eseguito e successivamente verrà analizzato nuovamente il sistema da RKHunter.

Al termine dell'analisi verranno allegate al report stilato da RKHunter le **principali differenze** trovate con il report di **baseline_analysis** creato in fase di startup.

```
File: rkav/start_av.py
def analyze_file(file, report):

    if ELF_FORMAT not in file_info:
        report_file.write("Can't execute on this environment\n")
        send_file(report_path, report_name)
        return

    # binary execution
    os.system(f"./{file_path}")
    # rkhunter analysis
    os.system("rkhunter --check --sk --nocolors > " + LOG_REPORT)

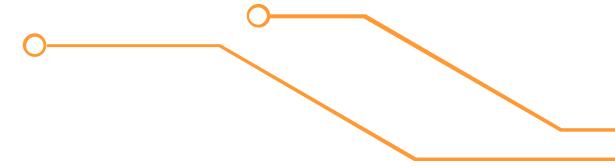
    # compare with baseline log
    with open(BASELINE_LOG) as baseline, open(LOG_REPORT) as log:
        difference = comparator.compare_report(baseline, log)

    report_file.write(summary_str)
    send_file(report_path, report_name)
```

```
<SUMMARY>
RKHunter didn't detect any system changes made by the malware
<END_SUMMARY>

[ Rootkit Hunter version 1.4.6 ]
Checking for rootkits...

Performing check of known rootkit files and directories
55808 Trojan - Variant A      [ Not found ]
ADM Worm                      [ Not found ]
AjaKit Rootkit                [ Not found ]
Adore Rootkit                 [ Not found ]
aPa Kit                        [ Not found ]
Apache Worm                    [ Not found ]
Ambient (ark) Rootkit         [ Not found ]
Balaur Rootkit                 [ Not found ]
BeastKit Rootkit               [ Not found ]
beX2 Rootkit                   [ Not found ]
cb Rootkit                     [ Not found ]
...
...
```



07

Limitazioni



Congestione della Rete

I test dei singoli sistemi analizzati hanno sempre dato esito positivo, ma durante il test finale di integrazione con la topologia completa si è riscontrato un problema con l'invio dei file.

È seguita quindi una fase di analisi e debugging, in cui si è notato che:

- Il traffico ICMP passava sempre
- Il traffico TCP veniva interrotto durante l'invio di grandi file, sia utilizzando **netcat** che il **client/server** da noi implementato
- Il traffico presentava un *elevato numero di ritrasmissioni TCP*.

1	10.23.1.2	10.23.0.2	TCP	46442 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129085860 TSecr=0 WS=128
2	10.23.1.2	10.23.0.2	TCP	46446 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129086110 TSecr=0 WS=128
3	10.23.1.2	10.23.0.2	TCP	[TCP Retransmission] [TCP Port numbers reused] 46442 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129086872 TSecr=0 WS=128
4	10.23.1.2	10.23.0.2	TCP	[TCP Retransmission] [TCP Port numbers reused] 46446 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129087128 TSecr=0 WS=128
5	10.23.1.2	10.23.0.2	TCP	[TCP Retransmission] [TCP Port numbers reused] 46442 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129088888 TSecr=0 WS=128
6	10.23.1.2	10.23.0.2	TCP	[TCP Retransmission] [TCP Port numbers reused] 46446 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129089144 TSecr=0 WS=128
7	PcsCompu...	b2:e9:2a:...	ARP	Who has 10.23.1.1? Tell 10.23.1.2
8	b2:e9:2a...	PcsCompu...	ARP	10.23.1.1 is at b2:e9:2a:ea:e5:ba
9	10.23.1.2	10.23.0.2	TCP	[TCP Retransmission] [TCP Port numbers reused] 46442 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129092920 TSecr=0 WS=128
10	10.23.1.2	10.23.0.2	TCP	[TCP Retransmission] [TCP Port numbers reused] 46446 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129093176 TSecr=0 WS=128
11	10.23.1.2	10.23.0.2	TCP	[TCP Retransmission] [TCP Port numbers reused] 46442 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129101112 TSecr=0 WS=128
12	10.23.1.2	10.23.0.2	TCP	[TCP Retransmission] [TCP Port numbers reused] 46446 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129101368 TSecr=0 WS=128
13	10.23.1.2	10.23.0.2	TCP	[TCP Retransmission] [TCP Port numbers reused] 46442 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129117240 TSecr=0 WS=128
14	10.23.1.2	10.23.0.2	TCP	[TCP Retransmission] [TCP Port numbers reused] 46446 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=129117496 TSecr=0 WS=128

Congestione della Rete



Questo situazione suggeriva uno stato di **rete congestionata**, per cui sono stati cercati indizi su una eventuale *misconfiguration* dei vari devices.

- Dall'analisi delle tracce wireshark non sono stati evidenziati problemi di ARP flooding, ed i devices conoscevano correttamente le loro rotte.

Dopo una ricerca si è visto che la probabile causa è dovuta alla **bandwidth limitata** nelle immagini utilizzate per i **cisco routers**¹. Questo succede in quanto le immagini sono distribuite in modo tale che non possano essere utilizzate in ambienti di produzione.

Per ovviare a questo problema è stato rallentato l'invio dei pacchetti nel trasferimento dei file inserendo una **time.sleep(CONGESTION_SLOWDOWN)** in modo da ridurre il rate con cui i pacchetti vengono immessi nella rete.

- Parametro configurabile, impostato nei test a **0.15s**

Si è visto sperimentalmente che questo ha permesso di risolvere i problemi di congestione evidenziati.



Ripristino Sandbox Environment

- Nei container che hostano i servizi RKHunter e JMDAV viene effettuata un'analisi dinamica del file binario ricevuto dal Central Node andandolo ad eseguire.
- In un ambiente reale di produzione sarebbe necessario effettuare uno **snapshot** dell'ambiente in **stato clean** da poter poi ripristinare prima di eseguire l'analisi successiva.
 - Questo perchè eventuali modifiche apportate da un malware sul sistema potrebbero compromettere i risultati delle analisi successivi, generando possibili falsi positivi o falsi negativi.
- Tuttavia nel nostro ambiente di sviluppo si è riscontrata l'impossibilità di effettuare snapshot da un orchestratore esterno alle sandbox utilizzate, poichè la loro virtualizzazione è gestita da GNS3.
- Quindi per utilizzare effettivamente la soluzione proposta in maniera consistente è necessario implementarla in un diverso contesto di virtualizzazione.



**GRAZIE
DELL'ATTENZIONE!**