NETWORK AND SYSTEM DEFENCE

UNIVERSITÀ DI ROMA TORVERGATA

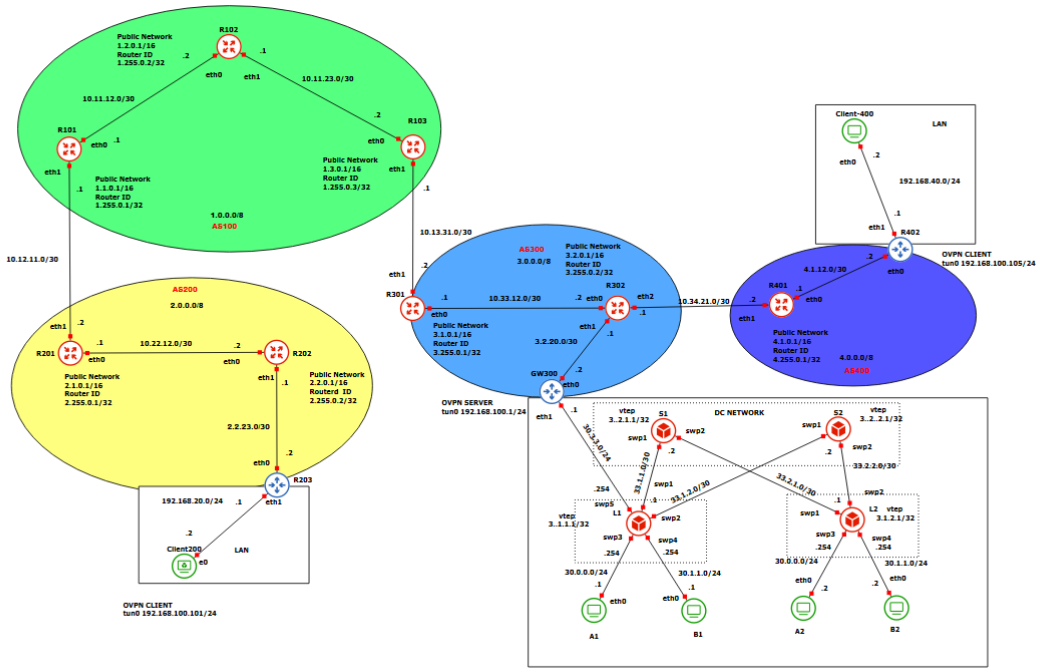
INGEGNERIA INFORMATICA

Project 1

Authors:
Luca Saverio Esposito (ID: 0334321)
Edoardo Manenti (ID: 0333574)

Date: July 3, 2024

1 Topologia



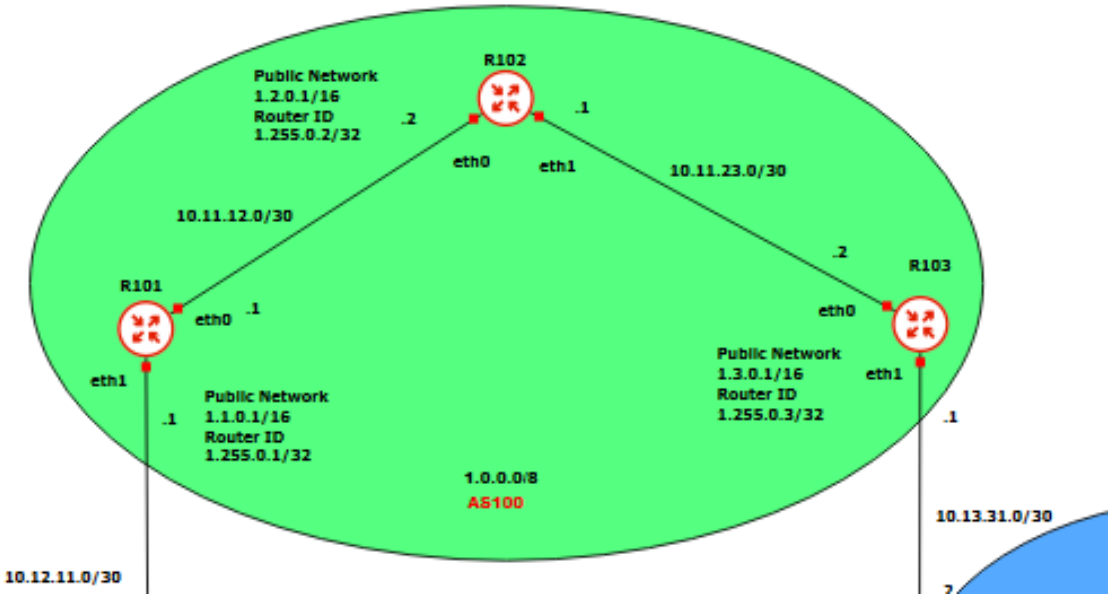
2 Configurazioni

Di seguito sono riportate le configurazioni di ogni sottorete del progetto. Sono stati creati degli script eseguiti all'avvio dei container per automatizzare il processo di configurazione. Oltre a creare un file start.sh che si occupa di configurare il container, si crea anche un file show.sh contenente i comandi principali per visualizzare la configurazione

2.1 AS100

AS100 is a transit Autonomous System providing network access to two customers: AS200 and AS300

- Configure eBGP peering with AS200 and AS300
- Configure iBGP peering between border routers
- Configure OSPF
- Configure LDP/MPLS in the core networ



2.1.1 R101

Si configurano le varie interfacce, dopodiché si procede con ospf, bgp (E-BGP verso AS200) e mpls.

```
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

# Configure sysctl for MPLS
echo 'net.mpls.conf.lo.input = 1' >> /etc/sysctl.conf
echo 'net.mpls.conf.eth0.input = 1' >> /etc/sysctl.conf
echo 'net.mpls.platform_labels = 100000' >> /etc/sysctl.conf
sysctl -p

# Start FRR services
cd /usr/lib/frr
./watchfrr zebra bgpd ldpd &

# Wait for the services to start properly
sleep 10

# Start a new shell explicitly and keep it open after execution
bash -c '
# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
```

```
configure terminal

interface lo
 ip address 1.1.0.1/16
 ip address 1.255.0.1/32
 mpls enable

interface eth0
 ip address 10.11.12.1/30
 mpls enable

interface eth1
 ip address 10.12.11.1/30

exit

router ospf
 router-id 1.255.0.1
 network 1.1.0.0/16 area 0
 network 1.255.0.1/32 area 0
 network 10.11.12.0/30 area 0
exit

router bgp 100
 network 1.1.0.0/16
 neighbor 1.255.0.3 remote-as 100
 neighbor 1.255.0.3 update-source 1.255.0.1
 neighbor 1.255.0.3 next-hop-self
 neighbor 10.12.11.2 remote-as 200
exit

mpls ldp
 router-id 1.255.0.1
 ordered-control
 address-family ipv4
  discovery transport-address 1.255.0.1
  interface eth0
  interface lo
 exit-address-family
exit

exit
VTYSH_EOF

# Start an interactive shell
```

```
cd /nsdUtils
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
show ip route connected
show ip bgp summary
show ip bgp
show ip ospf route
show mpls ldp binding
show mpls table

exit
VTYSH_EOF

SCRIPT_EOF

chmod +x show.sh
cd

echo "Node R101 configured successfully!"
```

2.1.2 R102

Su questo nodo non viene configurato BGP, presente solo tra i border routers, si procede con la configurazione delle interfacce e successivamente di OSPF e MPLS:

```
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

# Configure sysctl for MPLS
echo 'net.mpls.conf.lo.input = 1' >> /etc/sysctl.conf
echo 'net.mpls.conf.eth0.input = 1' >> /etc/sysctl.conf
echo 'net.mpls.conf.eth1.input = 1' >> /etc/sysctl.conf
echo 'net.mpls.platform_labels = 100000' >> /etc/sysctl.conf
```

```
sysctl -p

# Start FRR services
cd /usr/lib/frr
./watchfrr zebra bgpd ldpd &

# Wait for the services to start properly
sleep 10

# Start a new shell explicitly and keep it open after execution
bash -c '
# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
configure terminal

interface lo
 ip address 1.2.0.1/16
 ip address 1.255.0.2/32
 mpls enable

interface eth0
 ip address 10.11.12.2/30
 mpls enable

interface eth1
 ip address 10.11.23.1/30
 mpls enable

exit

router ospf
 router-id 1.255.0.2
 network 1.2.0.0/16 area 0
 network 1.255.0.2/32 area 0
 network 10.11.12.0/30 area 0
 network 10.11.23.0/30 area 0
exit

mpls ldp
 router-id 1.255.0.2
 ordered-control
 address-family ipv4
  discovery transport-address 1.255.0.2
  interface eth0
  interface eth1
```

```
  interface lo
 exit-address-family
exit

exit
VTYSH_EOF

# Start an interactive shell
cd /nsdUtils
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
show ip route connected
show ip ospf route
show mpls ldp binding
show mpls table

exit
VTYSH_EOF

SCRIPT_EOF

chmod +x show.sh
cd

echo "Node R102 configured successfully!"
```

2.1.3 R103

In maniera simile a come fatto su R101, si configura anche R103, in questo caso bisogna abilitare E-BGP verso l'AS300:

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash
```

```
5
6  # Configure sysctl for MPLS
7  echo 'net.mpls.conf.lo.input = 1' >> /etc/sysctl.conf
8  echo 'net.mpls.conf.eth0.input = 1' >> /etc/sysctl.conf
9  echo 'net.mpls.platform_labels = 100000' >> /etc/sysctl.conf
10 sysctl -p
11
12 # Start FRR services
13 cd /usr/lib/frr
14 ./watchfrr zebra bgpd ldpd &
15
16 # Wait for the services to start properly
17 sleep 10
18
19 # Start a new shell explicitly and keep it open after execution
20 bash -c '
21 # Configure interfaces and routing using vtysh
22 vtysh <<VTYSH_EOF
23 configure terminal
24
25 interface lo
26  ip address 1.3.0.1/16
27  ip address 1.255.0.3/32
28  mpls enable
29
30 interface eth0
31  ip address 10.11.23.2/30
32  mpls enable
33
34 interface eth1
35  ip address 10.13.31.1/30
36
37 exit
38
39 router ospf
40  router-id 1.255.0.3
41  network 1.3.0.0/16 area 0
42  network 1.255.0.3/32 area 0
43  network 10.11.23.0/30 area 0
44 exit
45
46 router bgp 100
47  network 1.3.0.0/16
48  neighbor 1.255.0.1 remote-as 100
49  neighbor 1.255.0.1 update-source 1.255.0.3
```

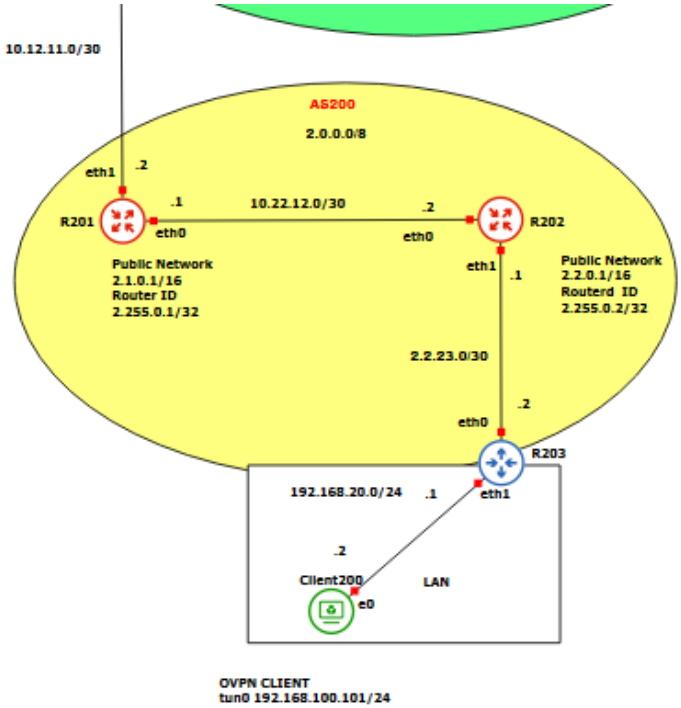
```
 neighbor 1.255.0.1 next-hop-self
 neighbor 10.13.31.2 remote-as 300
exit

mpls ldp
 router-id 1.255.0.3
 ordered-control
 address-family ipv4
 discovery transport-address 1.255.0.3
 interface eth0
 interface lo
 exit-address-family
exit

exit
VTYSH_EOF

# Start an interactive shell
cd /nsdUtils
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
show ip route connected
show ip bgp summary
show ip bgp
show ip ospf route
show mpls ldp binding
show mpls table

exit
VTYSH_EOF

SCRIPT_EOF

chmod +x show.sh
cd
```

```
95
96 echo "Node R103 configured successfully!"
```

2.2 AS200

AS 200 is a customer AS connected to AS100, which provides transit services.

- Setup eBGP peering with AS100
- Configure iBGP peering
- Configure internal routing as you wish (with or without OSPF)
- R203 is not a BGP speaker:
  - It has a default route towards R202
  - It has a public IP address from the IP address pool of AS200
  - It is the Access Gateway for the LAN attached to it:
    * Configure dynamic NAT
    * Configure a simple firewall to allow just connections initiated from the LAN



In questo sistema autonomo a differenza del precedente non è richiesta la configurazione di LDP/MPLS, è sufficiente BGP e un protocollo di routing interno.

2.2.1 R201

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

# Start FRR services
cd /usr/lib/frr
./watchfrr zebra bgpd ldpd &

# Wait for the services to start properly
sleep 10

# Start a new shell explicitly and keep it open after execution
bash -c '
# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
configure terminal

interface lo
 ip address 2.1.0.1/16
 ip address 2.255.0.1/32

interface eth0
 ip address 10.22.12.1/30

interface eth1
 ip address 10.12.11.2/30

exit

router ospf
 router-id 2.255.0.1
 network 2.1.0.0/16 area 0
 network 2.255.0.1/32 area 0
 network 10.22.12.0/30 area 0
exit

router bgp 200
 network 2.1.0.0/16
```

```
 neighbor 2.255.0.2 remote-as 200
 neighbor 2.255.0.2 update-source 2.255.0.1
 neighbor 2.255.0.2 next-hop-self
 neighbor 10.12.11.1 remote-as 100
exit

exit
VTYSH_EOF

# Start an interactive shell
cd /nsdUtils
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
show ip route connected
show ip bgp summary
show ip bgp
show ip ospf route

exit
VTYSH_EOF

SCRIPT_EOF

chmod +x show.sh
cd

echo "Node R201 configured successfully!"
```

2.2.2 R202

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash
```

```
5
6  # Start FRR services
7  cd /usr/lib/frr
8  ./watchfrr zebra bgpd ldpd &
9
10 # Wait for the services to start properly
11 sleep 10
12
13 # Start a new shell explicitly and keep it open after execution
14 bash -c '
15 # Configure interfaces and routing using vtysh
16 vtysh <<VTYSH_EOF
17 configure terminal
18
19 interface lo
20  ip address 2.2.0.1/16
21  ip address 2.255.0.2/32
22
23 interface eth0
24  ip address 10.22.12.2/30
25
26 interface eth1
27  ip address 2.2.23.1/30
28
29 exit
30
31 router ospf
32  router-id 2.255.0.2
33  network 2.2.0.0/16 area 0
34  network 2.255.0.2/32 area 0
35  network 2.2.23.0/30 area 0
36  network 10.22.12.0/30 area 0
37 exit
38
39 router bgp 200
40  network 2.2.0.0/16
41  neighbor 2.255.0.1 remote-as 200
42  neighbor 2.255.0.1 update-source 2.255.0.2
43  neighbor 2.255.0.1 next-hop-self
44  neighbor 2.2.23.2 remote-as 200
45  neighbor 2.2.23.2 update-source 2.255.0.2
46  neighbor 2.2.23.2 next-hop-self
47 exit
48
49 exit
```

```
VTYSH_EOF

# Start an interactive shell
cd /nsdUtils
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
show ip route connected
show ip bgp summary
show ip bgp
show ip ospf route

exit
VTYSH_EOF

SCRIPT_EOF

chmod +x show.sh
cd

echo "Node R202 configured successfully!"
```

2.2.3 R203

La configurazione di questo nodo è leggermente più delicata, non è un bgp speaker ma ha una default route verso R202. Deve svolgere il ruolo di gateway per la rete LAN a cui è collegato, inoltre deve implementare un meccanismo di firewall che permetta il transito del traffico che viene generato dalla LAN stessa. Di seguito è riportata la configurazione:

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

```

```
bash -c '# Flush all addresses and routes
ip addr flush dev eth0
ip addr flush dev eth1
ip route flush all

# Define all addresses and routes
ip addr add 2.2.23.2/30 dev eth0
ip addr add 192.168.20.1/24 dev eth1
ip route add default via 2.2.23.1

# Flush all rules and drop every packet not in rules
iptables -F
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT DROP

# Define all rules:

# Allow outgoing traffic from the LAN to the WAN
iptables -A FORWARD -i eth1 -o eth0 -s 192.168.20.0/24 -j ACCEPT

# Allow incoming traffic if it is part of an established or
   related connection
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j
   ACCEPT

# Configure NAT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

# Start an interactive shell
cd /nsdUtils
echo "Node R203 configured successfully!"
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

ip r
echo " "
iptables -L -v
```

```
49
50 SCRIPT_EOF
51
52 chmod +x show.sh
53 cd
54 echo "Node R203 configured successfully!"
```

2.2.4 Client 200

This device is sensitive, so it must be configured to use Mandatory Access Control.
OpenVPN → see later dedicated section.
Questo client è una macchina virtuale *Lubuntu 22.04.3*, la configurazione dell'interfaccia e del modulo MAC (AppArmor) sono le seguenti:

```
1     #Configure the interface with gateway
2 sudo ip addr add 192.168.20.2/24 dev enp0s8
3 sudo ip route add default via 192.168.20.1
```

A differenza dei container, le configurazioni sono persistenti essendo una VM, quindi è sufficiente eseguirle una sola volta.

2.2.5 App Armor

Nel client 200 è stato creato un server nginx, in ascolto sulla porta 8080, che espone due pagine web situate in:

- /data/www/safe/index.html

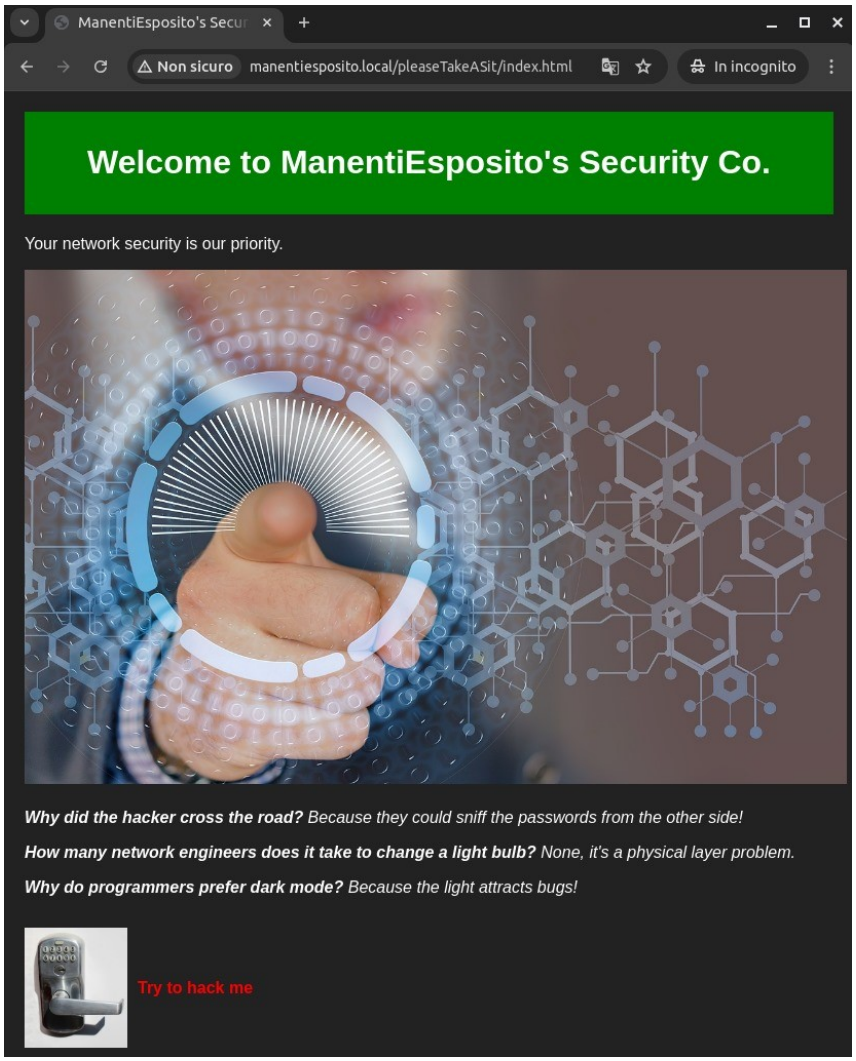


Figure 1: Legit page

- /data/www/unsafe/index.html

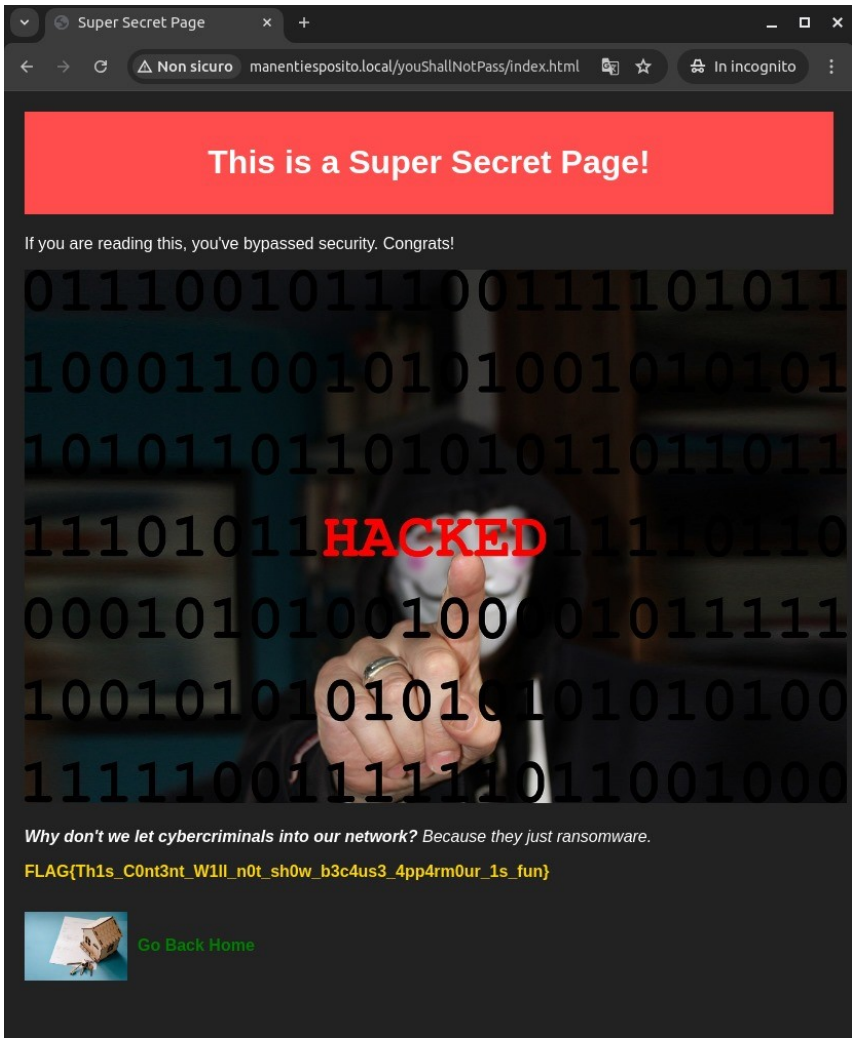


Figure 2: Not legit page

Configurando un profilo app armor per nginx viene bloccato l'accesso alla seconda pagina. I passi seguiti sono i seguenti:

```
    sudo mkdir -p /data/www/safe
    sudo mkdir -p /data/www/unsafe
    #Scrittura del codice html delle pagine ...

    #<--Configurazione nginx--
    user www-data;
    worker_processes 4;
    pid /run/nginx.pid;

    events {
        worker_connections 768;
```

```
12      }
13
14      http {
15          sendfile on;
16          tcp_nopush on;
17          tcp_nodelay on;
18          keepalive_timeout 65;
19          types_hash_max_size 2048;
20
21          include /etc/nginx/mime.types;
22          default_type application/octet-stream;
23
24          access_log /var/log/nginx/access.log;
25          error_log /var/log/nginx/error.log;
26
27          gzip on;
28          gzip_disable "msie6";
29
30          include /etc/nginx/conf.d/*.conf;
31
32          server {
33              listen 8080;
34              location / {
35                      root /data/www;
36              }
37          }
38      }
39      #--Configurazione nginx-->
40
41      #<--Profilo AA per nginx--
42      #include <tunables/global>
43
44      /usr/sbin/nginx {
45        #include <abstractions/apache2-common>
46        #include <abstractions/base>
47        #include <abstractions/nis>
48
49        capability dac_override,
50        capability dac_read_search,
51        capability net_bind_service,
52        capability setgid,
53        capability setuid,
54
55        /data/www/safe/* r,
56        deny /data/www/unsafe/* r,
```

```
57      /etc/group r,
58      /etc/nginx/conf.d/ r,
59      /etc/nginx/mime.types r,
60      /etc/nginx/nginx.conf r,
61      /etc/nsswitch.conf r,
62      /etc/passwd r,
63      /etc/ssl/openssl.cf r,
64      /run/nginx.pid rw,
65      /usr/sbin/nginx mr,
66      /var/log/nginx/access.log w,
67      /var/log/nginx/error.log w,
68    }
69     #--Profilo AA per nginx-->
```

Una volta configurato il profilo AA per nginx è possibile avviarlo tramite:

- sudo aa-enforce nginx
- sudo aa-complain nginx

Essendo la regola di tipo **deny**, verrà utilizzata anche quando il profilo è in complain mode, impedendo l'accesso alla pagina numero 2.

Figure 3: Not legit page

Per riottenere l'accesso bisogna disabilitare il profilo tramite *sudo aa-disable nginx*.
Quando si cambia la modalità di funzionamento di app armor è necessario eseguire il re-start di nginx per verificarne l'effettivo funzionamento *sudo service nginx restart*.
A causa del firewall su R203 il Client200 non può esser contattato direttamente dall'esterno, infatti R203 accetterà solo traffico originato da Client200. Fortunatamente grazie ad OVPN è possibile comunque contattare Client200, per esempio dai tenants A della DC Network, oppure dal Client400.
Tramite wget è possibile ottenere le pagina web esposte sul server nginx del Client200:

- Tramite *wget 192.168.100.101:8080/safe/index.html* si ottiene la pagina:

```
root@A1:/nsdUtils# wget 192.168.100.101:8080/safe/index.html
--2024-07-01 21:41:42--  http://192.168.100.101:8080/safe/index.html
Connecting to 192.168.100.101:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2500 (2.4K) [text/html]
Saving to: 'index.html.1'

index.html.1               100%[=====================================>]   2.44K  --.-KB/s    in 0.01s

2024-07-01 21:41:42 (233 KB/s) - 'index.html.1' saved [2500/2500]

root@A1:/nsdUtils# cat index.html
<!DOCTYPE html>
      <html>
      <head>
          <title>ManentiEsposito's Security Co.</title>
          <style>
              body {
                  font-family: Arial, sans-serif;
                  margin: 0;
                  padding: 20px;
                  background-color: #222; /* Black background */
                  color: #fff; /* White text for better contrast */
              }
              header {
                  background-color: green;
                  color: white;
                  padding: 10px 0;
                  text-align: center;
              }
              main {
                  margin-top: 20px;
```

Figure 4: WGET legit page

- Se invece si prova a richiedere la pagina unsafe mentre il profilo aa è attivo:

```
root@A1:/nsdUtils# wget 192.168.100.101:8080/unsafe/index.html
--2024-07-01 21:45:13--  http://192.168.100.101:8080/unsafe/index.html
Connecting to 192.168.100.101:8080... connected.
HTTP request sent, awaiting response... 403 Forbidden
2024-07-01 21:45:13 ERROR 403: Forbidden.
```

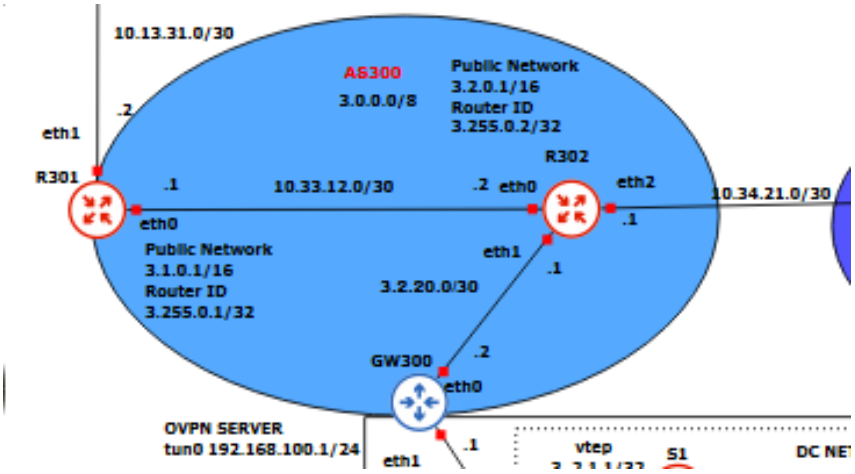
Figure 5: WGET not legit page

2.3 AS300

AS 300 is a customer AS connected to AS 100, which provides transit services. It also has a lateral peering relationship with AS 400:

- Setup eBGP peering with AS400 and AS100
- Configure iBGP peering
- Configure internal routing as you wish (with or without OSPF)
- GW300 is not a BGP speaker:
  - It has a default route towards R302
  - It has a public IP address from the IP address pool of AS300
  - It is the Access Gateway for the Data Center network attached to it
    - Configure dynamic NAT

* OpenVPN server → see later dedicated section



2.3.1 R301

La configurazione è simile a quella fatta per gli altri router, bisogna specificare E-BGP con AS100:

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

# Start FRR services
cd /usr/lib/frr
./watchfrr zebra bgpd ldpd &

# Wait for the services to start properly
sleep 10

# Start a new shell explicitly and keep it open after execution
bash -c '
# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
configure terminal

interface lo
 ip address 3.1.0.1/16
 ip address 3.255.0.1/32

interface eth0
 ip address 10.33.12.1/30
```

```
25
26 interface eth1
27  ip address 10.13.31.2/30
28
29 exit
30
31 router ospf
32  router-id 3.255.0.1
33  network 3.1.0.0/16 area 0
34  network 3.255.0.1/32 area 0
35  network 10.33.12.0/30 area 0
36 exit
37
38 router bgp 300
39  network 3.1.0.0/16
40  neighbor 3.255.0.2 remote-as 300
41  neighbor 3.255.0.2 update-source 3.255.0.1
42  neighbor 3.255.0.2 next-hop-self
43  neighbor 10.13.31.1 remote-as 100
44 exit
45
46 exit
47 VTYSH_EOF
48
49 # Start an interactive shell
50 cd /nsdUtils
51 exec bash
52 '
53 SCRIPT_EOF
54
55 chmod +x start.sh
56 cd
57 cd ../nsdUtils
58 cat << 'SCRIPT_EOF' > show.sh
59 #!/bin/bash
60
61 # Configure interfaces and routing using vtysh
62 vtysh <<VTYSH_EOF
63 show ip route connected
64 show ip bgp summary
65 show ip bgp
66 show ip ospf route
67
68 exit
69 VTYSH_EOF
```

```

SCRIPT_EOF

chmod +x show.sh
cd

echo "Node R301 configured successfully!"
```

2.3.2 R302

In maniera del tutto analoga al precedente, in questo caso E-BGP con AS400:

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

# Start FRR services
cd /usr/lib/frr
./watchfrr zebra bgpd ldpd &

# Wait for the services to start properly
sleep 10

# Start a new shell explicitly and keep it open after execution
bash -c '
# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
configure terminal

interface lo
 ip address 3.2.0.1/16
 ip address 3.255.0.2/32

interface eth0
 ip address 10.33.12.2/30

interface eth1
 ip address 3.2.20.1/30

interface eth2
 ip address 10.34.21.1/30

exit

```

```
router ospf
 router-id 3.255.0.2
 network 3.2.0.0/16 area 0
 network 3.255.0.2/32 area 0
 network 3.2.20.0/30 area 0
 network 10.33.12.0/30 area 0
exit

router bgp 300
 network 3.2.0.0/16
 neighbor 3.255.0.1 remote-as 300
 neighbor 3.255.0.1 update-source 3.255.0.2
 neighbor 3.255.0.1 next-hop-self
 neighbor 3.2.20.2 remote-as 300
 neighbor 3.2.20.2 update-source 3.2.0.1
 neighbor 3.2.20.2 next-hop-self
 neighbor 10.34.21.2 remote-as 400
exit

exit
VTYSH_EOF

# Start an interactive shell
cd /nsdUtils
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
show ip route connected
show ip bgp summary
show ip bgp
show ip ospf route

exit
VTYSH_EOF

SCRIPT_EOF
```

```
79
80 chmod +x show.sh
81 cd
82
83 echo "Node R302 configured successfully!"
```

2.3.3 GW300

Questo dispositivo gioca un ruolo chiave: fa da gateway per i nodi all'interno della DC Network. Dispone di due VLAN sulle due interfacce virtuali eth1.313 e eth.323 che consentono al traffico di entrare e uscire dai tenants del data center.
Oltre a fare da dynamic NAT implementa una regola per bloccare il traffico proveniente dal TENA verso TENB e viceversa.

```
1     cd
2 cd ../nsdUtils
3 cat << 'SCRIPT_EOF' > start.sh
4 #!/bin/bash
5
6 bash -c '
7 # Flush all addresses and routes
8 ip addr flush dev eth0
9 ip addr flush dev eth1
10 ip route flush all
11
12 # Define all addresses and routes
13 ip addr add 3.2.20.2/30 dev eth0
14 ip route add default via 3.2.20.1
15
16 sysctl -w net.ipv4.ip_forward=1
17
18 # Links with DC-Network
19 ip link add link eth1 name eth1.313 type vlan id 313
20 ip link add link eth1 name eth1.323 type vlan id 323
21 ip addr add 30.3.3.1/24 dev eth1.313
22 ip addr add 30.3.3.1/24 dev eth1.323
23 ip link set eth1.313 up
24 ip link set eth1.323 up
25 ip route add 30.0.0.0/24 via 30.3.3.254 dev eth1.313
26 ip route add 30.1.1.0/24 via 30.3.3.254 dev eth1.323
27
28 # Configure NAT
29 iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
30
```

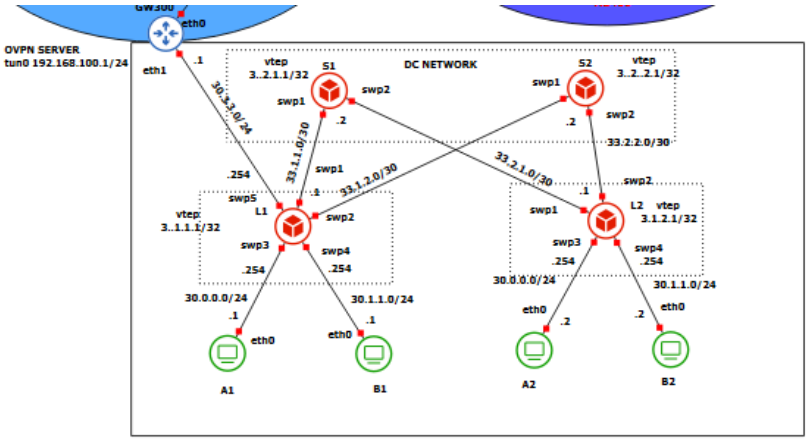
```
# Block traffic from 30.0.0.0/24 to 30.1.1.0/24
iptables -A FORWARD -s 30.0.0.0/24 -d 30.1.1.0/24 -j DROP


#TODO Configure OPENVPN

# Start an interactive shell
cd /nsdUtils/CA/GW300
openvpn GW300.ovpn &

echo "Node GW300 configured successfully!"
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

ip r

SCRIPT_EOF

chmod +x show.sh
cd
echo "Node GW300 configured successfully!"
```

2.4 DC Network

DC Network is a leaf-spine Data Center network with two leaves and two spines. There are 2 tenants (A and B) in the cloud network, each hosting two virtual machines connected to leaf1 and leaf2. The tenants are assigned one broadcast domain each.

- Realize VXLAN/EVPN forwarding in the DC network to provide L2VPNs between the tenants' machines

- In L1, enable the connectivity to the external network. In other words, both tenants' machines must reach the external network through the link between L1 and R303, including the encapsulation in OpenVPN tunnels when necessary



Le Spines e le Leaf sono delle VM *Cumulus Linux 4.10*. Si è deciso di configurarle creando un servizio ad avvio automatico.

2.4.1 S1

```
    cd
cat << 'SCRIPT_EOF' > config.sh
#!/bin/bash

net del all

# IP-ADDR
net add interface swp1 ip add 33.1.1.2/30
net add interface swp2 ip add 33.2.1.2/30
net add loopback lo ip add 3.2.1.1/32

# OSPF
net add ospf router-id 3.2.1.1
net add ospf network 0.0.0.0/0 area 0

# MP-eBGP
net add bgp autonomous-system 65000
net add bgp router-id 3.2.1.1
net add bgp neighbor swp1 remote-as external
net add bgp neighbor swp2 remote-as external
net add bgp evpn neighbor swp1 activate
net add bgp evpn neighbor swp2 activate

net commit
cd
```

```
26
27 SCRIPT_EOF
28 cd
29 chmod +x config.sh
30
31 # Write the systemd service file with sudo
32 sudo touch /etc/systemd/system/configNet.service
33 sudo tee /etc/systemd/system/configNet.service > /dev/null << '
      SCRIPT_EOF'
34 [Unit]
35 Description=Run Script To Configure Net At Startup
36 After=network.target
37
38 [Service]
39 ExecStart=/bin/bash /home/cumulus/config.sh
40
41 [Install]
42 WantedBy=multi-user.target
43 SCRIPT_EOF
44
45 # Reload systemd, enable and start the service
46 sudo systemctl daemon-reload
47 sudo systemctl enable configNet.service
48 sudo systemctl start configNet.service
49
50 echo "Node S1 configured successfully!"
```

2.4.2 S2

```
1     cd
2 cat << 'SCRIPT_EOF' > config.sh
3 #!/bin/bash
4
5 net del all
6
7 # IP-ADDR
8 net add interface swp1 ip add 33.1.2.2/30
9 net add interface swp2 ip add 33.2.2.2/30
10 net add loopback lo ip add 3.2.2.1/32
11
12 # OSPF
13 net add ospf router-id 3.2.2.1
14 net add ospf network 0.0.0.0/0 area 0
15
```

```
# MP-eBGP
net add bgp autonomous-system 65000
net add bgp router-id 3.2.2.1
net add bgp neighbor swp1 remote-as external
net add bgp neighbor swp2 remote-as external
net add bgp evpn neighbor swp1 activate
net add bgp evpn neighbor swp2 activate

net commit
cd

SCRIPT_EOF
cd
chmod +x config.sh

# Write the systemd service file with sudo
sudo touch /etc/systemd/system/configNet.service
sudo tee /etc/systemd/system/configNet.service > /dev/null << '
   SCRIPT_EOF'
[Unit]
Description=Run Script To Configure Net At Startup
After=network.target

[Service]
ExecStart=/bin/bash /home/cumulus/config.sh

[Install]
WantedBy=multi-user.target
SCRIPT_EOF

# Reload systemd, enable and start the service
sudo systemctl daemon-reload
sudo systemctl enable configNet.service
sudo systemctl start configNet.service

echo "Node S2 configured successfully!"
```

2.4.3 L1

```
    cd
cat << 'SCRIPT_EOF' > config.sh
#!/bin/bash

# BRIDGE
```

```
net add bridge bridge ports swp3,swp4,swp5
net add interface swp3 bridge access 31
net add interface swp4 bridge access 32

# IP-ADDR
net add interface swp1 ip add 33.1.1.1/30
net add interface swp2 ip add 33.1.2.1/30
net add loopback lo ip add 3.1.1.1/32

# OSPF
net add ospf router-id 3.1.1.1
net add ospf network 33.1.1.0/30 area 0
net add ospf network 33.1.2.0/30 area 0
net add ospf network 3.1.1.1/32 area 0
net add ospf passive-interface swp3,swp4


# VXLAN
net add vxlan vni31 vxlan id 31
net add vxlan vni31 vxlan remoteip 3.1.2.1
net add vxlan vni31 vxlan local-tunnelip 3.1.1.1
net add vxlan vni31 bridge access 31
net add vxlan vni32 vxlan id 32
net add vxlan vni32 vxlan remoteip 3.1.2.1
net add vxlan vni32 vxlan local-tunnelip 3.1.1.1
net add vxlan vni32 bridge access 32

# MP-eBGP
net add bgp autonomous-system 65001
net add bgp router-id 3.1.1.1
net add bgp neighbor swp1 remote-as 65000
net add bgp neighbor swp2 remote-as 65000
net add bgp evpn neighbor swp1 activate
net add bgp evpn neighbor swp2 activate
net add bgp evpn advertise-all-vni
net add bgp l2vpn evpn advertise-default-gw

# VTEPs
net add vlan 31 ip address 30.0.0.254/24
net add vlan 32 ip address 30.1.1.254/24

net add vlan 313 ip address 30.3.3.254/24
net add vlan 313 ip gateway 30.3.3.1

net add vlan 323 ip address 30.3.3.254/24
```

```
net add vlan 323 ip gateway 30.3.3.1

# VRF
net add vlan 313 vrf TENA
net add vlan 31 vrf TENA
net add vlan 323 vrf TENB
net add vlan 32 vrf TENB

net commit

cd

SCRIPT_EOF
cd
chmod +x config.sh

# Write the systemd service file with sudo
sudo touch /etc/systemd/system/configNet.service
sudo tee /etc/systemd/system/configNet.service > /dev/null << '
   SCRIPT_EOF'
[Unit]
Description=Run Script To Configure Net At Startup
After=network.target

[Service]
ExecStart=/bin/bash /home/cumulus/config.sh

[Install]
WantedBy=multi-user.target
SCRIPT_EOF

# Reload systemd, enable and start the service
sudo systemctl daemon-reload
sudo systemctl enable configNet.service
sudo systemctl start configNet.service

echo "Node L1 configured successfully!"
```

2.4.4 L2

```
    cd
cat << 'SCRIPT_EOF' > config.sh
#!/bin/bash

```

```
net del all

# BRIDGE
net add bridge bridge ports swp3,swp4
net add interface swp3 bridge access 31
net add interface swp4 bridge access 32

# IP-ADDR
net add interface swp1 ip add 33.2.1.1/30
net add interface swp2 ip add 33.2.2.1/30
net add loopback lo ip add 3.1.2.1/32

# OSPF
net add ospf router-id 3.1.2.1
net add ospf network 33.2.1.0/30 area 0
net add ospf network 33.2.2.0/30 area 0
net add ospf network 3.1.2.1/32 area 0
net add ospf passive-interface swp3,swp4

# VXLAN
net add vxlan vni31 vxlan id 31
net add vxlan vni31 vxlan remoteip 3.1.1.1
net add vxlan vni31 vxlan local-tunnelip 3.1.2.1
net add vxlan vni31 bridge access 31
net add vxlan vni32 vxlan id 32
net add vxlan vni32 vxlan remoteip 3.1.1.1
net add vxlan vni32 vxlan local-tunnelip 3.1.2.1
net add vxlan vni32 bridge access 32

# MP-eBGP
net add bgp autonomous-system 65002
net add bgp router-id 3.1.2.1
net add bgp neighbor swp1 remote-as 65000
net add bgp neighbor swp2 remote-as 65000
net add bgp evpn neighbor swp1 activate
net add bgp evpn neighbor swp2 activate
net add bgp evpn advertise-all-vni

# VTEPs
net add vlan 31 ip address 30.0.0.254/24
net add vlan 32 ip address 30.1.1.254/24

# VRF
net add vlan 31 vrf TENA
net add vlan 32 vrf TENB
```

```

net commit
cd

SCRIPT_EOF
cd
chmod +x config.sh

# Write the systemd service file with sudo
sudo touch /etc/systemd/system/configNet.service
sudo tee /etc/systemd/system/configNet.service > /dev/null << '
    SCRIPT_EOF'
[Unit]
Description=Run Script To Configure Net At Startup
After=network.target

[Service]
ExecStart=/bin/bash /home/cumulus/config.sh

[Install]
WantedBy=multi-user.target
SCRIPT_EOF

# Reload systemd, enable and start the service
sudo systemctl daemon-reload
sudo systemctl enable configNet.service
sudo systemctl start configNet.service

echo "Node L2 configured successfully!"
```

2.4.5 A1

Questi nodi (A2,B1,B2) sono invece classici container utilizzati per tutti gli altri componenti, quindi si torna al solito approccio per la persistenza: scrittura su un file sh dei comandi e avvio automatico:

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

bash -c '
# Flush all addresses and routes
ip addr flush dev eth0
ip route flush all
```

```

# Define all addresses and routes
ip addr add 30.0.0.1/24 dev eth0
ip route add default via 30.0.0.254

# Start an interactive shell
cd /nsdUtils
echo "Node A1 configured successfully!"
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

ip r

SCRIPT_EOF

chmod +x show.sh
cd
echo "Node A1 configured successfully!"
```

2.4.6 A2

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

bash -c '
# Flush all addresses and routes
ip addr flush dev eth0
ip route flush all

# Define all addresses and routes
ip addr add 30.0.0.2/24 dev eth0
ip route add default via 30.0.0.254

# Start an interactive shell
cd /nsdUtils
```

```
echo "Node A2 configured successfully!"
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

ip r

SCRIPT_EOF

chmod +x show.sh
cd
echo "Node A2 configured successfully!"
```

2.4.7 B1

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

bash -c '
# Flush all addresses and routes
ip addr flush dev eth0
ip route flush all

# Define all addresses and routes
ip addr add 30.1.1.1/24 dev eth0
ip route add default via 30.1.1.254

# Start an interactive shell
cd /nsdUtils
echo "Node B1 configured successfully!"
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
```

```
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

ip r

SCRIPT_EOF

chmod +x show.sh
cd
echo "Node B1 configured successfully!"
```

2.4.8 B2

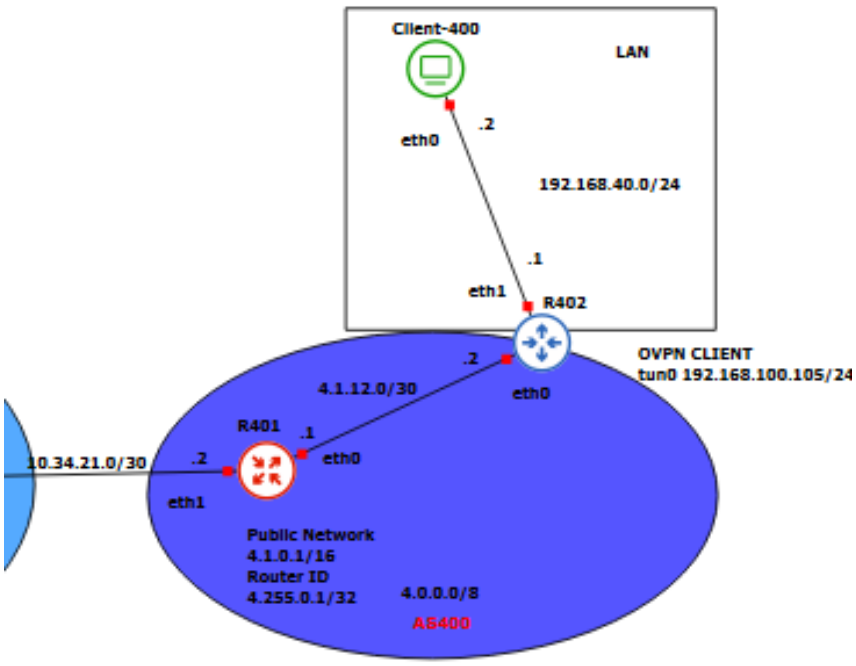
```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

bash -c '
# Flush all addresses and routes
ip addr flush dev eth0
ip route flush all

# Define all addresses and routes
ip addr add 30.1.1.2/24 dev eth0
ip route add default via 30.1.1.254

# Start an interactive shell
cd /nsdUtils
echo "Node B2 configured successfully!"
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

ip r

SCRIPT_EOF
```

```
31
32 chmod +x show.sh
33 cd
34 echo "Node B2 configured successfully!"
```

2.5 AS400

AS 400 has a lateral peering relationship with AS 300.

- Setup eBGP peering with AS400 and AS100
- R402 is not a BGP speaker
  - It has a default route towards R401
  - It has a public IP address from the IP address pool of AS400
  - It is the Access Gateway for the LAN attached to it
    - Configure dynamic NAT
    - Configure OpenVPN→ see later dedicated section



2.5.1 R401

In maniera analoga a quanto visto fin'ora sui vari router:

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

# Start FRR services
cd /usr/lib/frr
./watchfrr zebra bgpd ldpd &

# Wait for the services to start properly
sleep 10

# Start a new shell explicitly and keep it open after execution
bash -c '
# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
configure terminal

interface lo
 ip address 4.1.0.1/16
 ip address 4.255.0.1/32

interface eth0
 ip address 4.1.12.1/30

interface eth1
 ip address 10.34.21.2/30

exit

router ospf
 router-id 4.255.0.1
 network 4.1.0.0/16 area 0
 network 4.255.0.1/32 area 0
 network 4.1.12.0/30 area 0
exit

router bgp 400
 network 4.1.0.0/16
 neighbor 4.1.12.2 remote-as 400
 neighbor 4.1.12.2 update-source 4.255.0.1
 neighbor 4.1.12.2 next-hop-self
```

```
 neighbor 10.34.21.1 remote-as 300
exit

exit
VTYSH_EOF

# Start an interactive shell
cd /nsdUtils
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

# Configure interfaces and routing using vtysh
vtysh <<VTYSH_EOF
show ip route connected
show ip bgp summary
show ip bgp
show ip ospf route

exit
VTYSH_EOF

SCRIPT_EOF

chmod +x show.sh
cd

echo "Node R401 configured successfully!"
```

2.5.2 R402

Questo nodo è paragonabile a R203, infatti la configurazione è simile, eccetto per la componente legata ad OPENVPN:

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

```

```
bash -c '
# Flush all addresses and routes
ip addr flush dev eth0
ip addr flush dev eth1
ip route flush all

# Define all addresses and routes
ip addr add 4.1.12.2/30 dev eth0
ip addr add 192.168.40.1/24 dev eth1
ip route add default via 4.1.12.1

# Configure NAT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

# Start an interactive shell
cd /nsdUtils/ovpn
openvpn R402.ovpn &

echo "Node R402 configured successfully!"
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

ip r
echo " "
iptables -L -v

SCRIPT_EOF

chmod +x show.sh
cd
cd /nsdUtils

cd ..
echo "Node R402 configured successfully!"
```

2.5.3 Client 400

This is a simple LAN device with a default route through R402.
La sua configurazione:

```
    cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > start.sh
#!/bin/bash

bash -c '
# Flush all addresses and routes
ip addr flush dev eth0
ip route flush all

# Define all addresses and routes
ip addr add 192.168.40.2/24 dev eth0
ip route add default via 192.168.40.1

# Start an interactive shell
cd /nsdUtils
echo "Node Client400 configured successfully!"
exec bash
'
SCRIPT_EOF

chmod +x start.sh
cd
cd ../nsdUtils
cat << 'SCRIPT_EOF' > show.sh
#!/bin/bash

ip r

SCRIPT_EOF

chmod +x show.sh
cd
echo "Node Client400 configured successfully!"
```

2.6 OPENVPN

Setup OpenVPN to realize a VPN between client-200, R402's LAN, and the DataCenter network.

- Client-200 is an OpenVPN client
- R402 is an OpenVPN client, providing VPN access to and from the LAN attached to it.
- GW300 is the OpenVPN server, providing VPN access to and from the Data Center network. In particular, the network belonging to tenant A must be accessible through the VPN

Nelle configurazioni precedenti sono già presenti i comandi per avviare il servizio di OVPN, tuttavia sono necessari degli step preliminari prima di poterlo utilizzare.
Inizialmente va installato su Lubuntu tramite sudo apt install openvp3.
Per abilitare easy-rsa è necessario entrare nella shell della VM GNS3 ed eseguire: docker pull nsdcourse/basenet.
Ora che si ha tutto a disposizione sono stati svolti i seguenti passi per completare la configurazione:

```
    # Su GW300 (il server)
#  all'interno di /usr/share/easy-rsa:
./easyrsa init-pki
./easyrsa build-ca nopass
./easyrsa build-server-full GW300 nopass

# Dopodiche' si generano i certificati per il client-200 e per
   R402
./easyrsa build-client-full Client-200 nopass
./easyrsa build-client-full R402 nopass

# Genera la chiave da 2048 bit
./easyrsa gen-dh

# Si spostano nella directory persistente i
# certificati e la chiave generata per ogni elemento
mkdir /nsdUtils/CA
mkdir /nsdUtils/CA/GW300
mkdir /nsdUtils/CA/Client-200
mkdir /nsdUtils/CA/R402

cp pki/ca.crt /nsdUtils/CA/
cp pki/issued/GW300.crt /nsdUtils/CA/GW300/
cp pki/private/GW300.key /nsdUtils/CA/GW300/
cp pki/dh.pem /nsdUtils/CA/GW300/

cp pki/issued/Client-200.crt /nsdUtils/CA/Client-200/
cp pki/private/Client-200.key /nsdUtils/CA/Client-200/

```

```
cp pki/issued/R402.crt /nsdUtils/CA/R402/
cp pki/private/R402.key /nsdUtils/CA/R402/

#Ora si copiano all'interno di Client-200 e di R402
# i corrispettivi certificati:
# Su Client-200
cd ~/Desktop/
mkdir ovpn
cd ovpn
nano ca.crt
nano Client-200.crt
nano Client-200.key

# Su R402
cd /nsdUtils
mkdir ovpn
cd ovpn
nano ca.crt
nano R402.crt
nano R402.key

#Si procede con la configurazione del servizio

#Nel server (GW300) si copia il certificato ca
#   all'interno della dir di GW300
cp pki/ca.crt /nsdUtils/CA/GW300 # PD ma copiarlo prima

#All'interno di /CA/GW300 si crea il file di
# configurazione GW300.ovpn

port 1194
proto udp
dev tun
ca ca.crt
cert GW300.crt
key GW300.key
dh dh.pem
server 192.168.100.0 255.255.255.0
push "route 192.168.20.2 255.255.255.255"
push "route 192.168.40.0 255.255.255.0"
push "route 30.0.0.0 255.255.255.0"
route 192.168.20.2 255.255.255.255
route 192.168.40.0 255.255.255.0
client-config-dir ClientConf
client-to-client
```

```
keepalive 10 120
cipher AES-256-GCM

# Creare la cartella ClientConf in cui si mettono i file
# contenenti informazioni aggiuntive sui client (/nsdUtils/CA/
   GW300):

mkdir ClientConf

#Client-200 (e' il nome del file senza estensione)
ifconfig-push 192.168.100.101 192.168.100.102
iroute 192.168.20.2 255.255.255.255

#R402
ifconfig-push 192.168.100.105 192.168.100.106
iroute 192.168.40.0 255.255.255.0

#Per avviare in background:
openvpn GW300.ovpn &

#Nel Client-200, all'interno di ovpn, si crea il file
# Client-200.ovpn:
client
dev tun
proto udp
remote 3.2.20.2 1194
resolv-retry infinite
ca ca.crt
cert Client-200.crt
key Client-200.key
remote-cert-tls server
cipher AES-256-GCM

# Si installa openvpn:
sudo mkdir -p /etc/apt/keyrings && curl -fsSL https://packages.
   openvpn.net/packages-repo.gpg | sudo tee /etc/apt/keyrings/
   openvpn.asc
DISTRO=$(lsb_release -c | awk '{print $2}')
echo "deb [signed-by=/etc/apt/keyrings/openvpn.asc] https://
   packages.openvpn.net/openvpn3/debian $DISTRO main" | sudo tee /
   etc/apt/sources.list.d/openvpn-packages.list
sudo apt update
sudo apt install openvpn3

#Si importa la configurazione in maniera persistente con :
```

```
114 openvpn3 config-import --config /home/<nome_utente>/Desktop/ovpn/
    Client-200.ovpn --name Client-200 --persistent
115
116 #Si concede a root l'accesso:
117 openvpn3 config-acl --show --lock-down true --grant root --config
     Client-200
118
119 #Si fa partire il servizio con:
120 sudo systemctl enable --now openvpn3-session@Client-200.service
121
122 # R402 all'interno di /nsdUtils/ovpn si crea il file
123 # di configurazione R402.ovpn:
124 client
125 dev tun
126 proto udp
127 remote 3.2.20.2 1194
128 resolv-retry infinite
129 ca ca.crt
130 cert R402.crt
131 key R402.key
132 remote-cert-tls server
133 cipher AES-256-GCM
134
135 # Si avvia con
136 openvpn R402.ovpn &
```