Plateforme Cybersécurité

Préparation pour l'utilisation de docker et de la plateforme d'étude

- Installation de **docker** et ajout de votre utilisateur dans le groupe « docker » :

xterm
```
$ sudo apt install docker.io docker-compose-v2
$ sudo usermod -aG docker $USER
```

Il sera nécessaire de vous déconnecter/reconnecter de votre session graphique ou de redémarrer la machine pour activer l'appartenance à ce groupe.

- Quelques commandes utiles pour docker :

commande	description
`docker ps`	obtenir la liste des containers
`docker kill <nom>`	stopper le container
`docker rm <nom>`	effacer les données relative au container
`docker inspect -f '.State.Pid' h1`	obtenir le PID d'un container connu par son nom

- Vous installerez « openvswitch », le simulateur de switch :

xterm
```
$ sudo apt install openvswitch-common openvswitch-switch
```

- Vous récupérez le « git » de la plateforme d'étude :

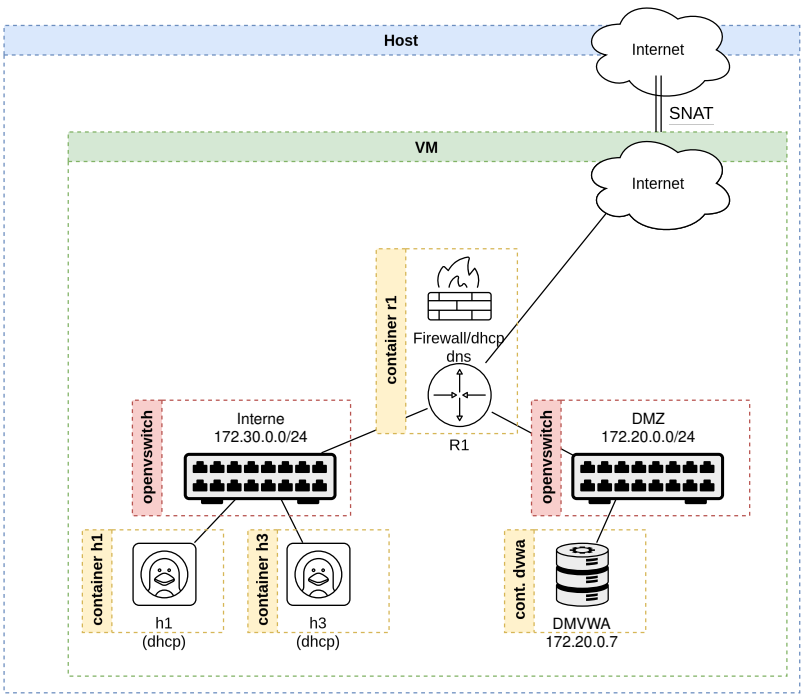
xterm
```
$ git clone https://git.p-fb.net/PeFClic/lab_cybersecu.git
$ cd lab_cybersecu
```

- Vous créerez les images des différents systèmes Linux utilisés par les différents container :

xterm
```
$ ./create_images
```

Les images sont basées sur Alpine, une petite distribution linux.

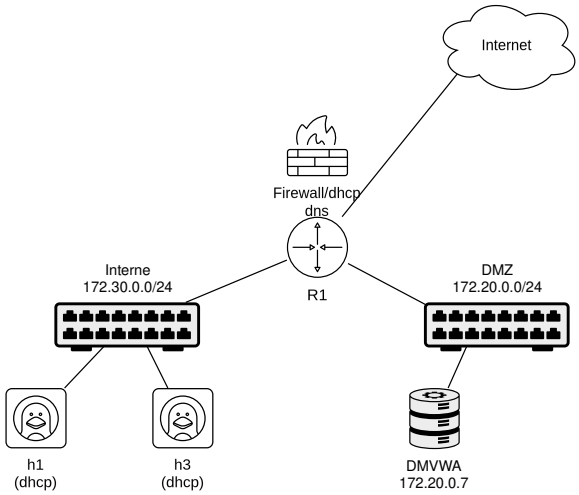
Présentation de la plateforme



Vous disposer :

- *d'une VM sur votre machine principale, hébergeant une distribution Linux Ubuntu qui bénéficie automatiquement de « SNAT » pour accéder à Internet ;*
- *de deux switches ;*
- *de plusieurs containers docker, correspondant en particulier à :*
  - *deux postes « h1 » et « h2 » ;*
  - *le routeur « r1 » qui assure les fonctions de routage, de dhcp et de dns pour les deux postes ;*

Le synoptique réseau vécu de « l'intérieur ».



Démarrage de la plateforme

xterm
```
$ sudo ./build_architecture
Ajout switch : interne
...
1f7ead3061f0: Pull complete
Digest: sha256:97a8f9fd052aaabdee76e8367d1ec985b3e44e6add0b27a28d8fee9cafe426c7
Status: Downloaded newer image for ghcr.io/mitre/caldera:latest
1a22228c808cdee71c87fab031736a986000ee298cb8f15dede8f1319bf61b91
Ajout caldera : 5791
```

Vous devez avoir les containers suivants :

xterm
```
docker ps
CONTAINER ID   IMAGE                          COMMAND                  NAMES
1a22228c808c   ghcr.io/mitre/caldera:latest   "python3 server.py"      caldera
cd3d0e8118a3   client-auth                    "/bin/sh"                client
dfdb24263467   test-auth                      "uvicorn app:app --h…"   auth
7f9369807972   containous/whoami:latest       "/whoami"                web
1b869269be9a   noeud                          "sh -c 'umount /etc/…"   h3
73c170da0640   noeud                          "sh -c 'umount /etc/…"   h1
fe45699a3e75   dns_dhcp                       "dnsmasq -k"             r1
```

Pour recommencer

Pour recommencer et nettoyer containers, switches, veth, etc. :

xterm
```
$ sudo ./clean
```

Questions :

a. Décrivez les opérations qui sont réalisées :
- Quelles sont les différents nœuds créés et où sont-ils connectés ?
- Quels sont les services présents dans le réseau ?
- Comment se passe la configuration réseau de « h1 » et « h3 » ?

Vous allez maintenant vous connecter sur « h1 » :

xterm
```
$ docker exec -it h1 bash
h1:~# ip -br link
lo               UNKNOWN        00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
h1-eth0@if22     UP             32:3c:b9:0c:54:08 <BROADCAST,MULTICAST,UP,LOWER_UP>
h1:~# ip -br address
lo               UNKNOWN        127.0.0.1/8 ::1/128
h1-eth0@if22     UP             172.30.0.185/24 fe80::303c:b9ff:fe0c:5408/64
h1:~#
```

Sur « r1 » :

xterm
```
r1:~# ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
eth0@if19        UP             172.17.0.2/16
r1-eth0@if20     UP             172.30.0.254/24 fe80::4031:27ff:fe8e:9155/64
r1:~#
```

Questions :

b. Consultez le fichier de configuration « dnsmasq.conf » du serveur « dnsmasq » présent sur « r1 » et disponible dans le répertoire « ./config_r1 » :

- Quelles sont les différentes adresses IP et nom de domaine des serveurs présents ?
  Vous testerez avec la commande « dig » pour faire des requêtes dns que vous installerez au préalable :

```
xterm
h1:~# h1:~# apk add bind-tools
(13/13) Installing bind-tools (9.18.47-r0)
Executing busybox-1.37.0-r14.trigger
OK: 25 MiB in 55 packages
h1:~# dig +short p-fb.net
198.245.60.92
```

  Essayez sur les adresses DNS que vous avez trouvées.

- Faites un ping de « h1 » vers « h3 », puis de « h1 » vers « r1 ».
  À l'aide de la commande « ip neighbor » consultez la table d'assocation « adresse MAC/adresse IP »

- Sniffez les paquets correspondants à ces pings avec « tcpdump » :

```
xterm
h1:~# tcpdump -lnvve -i h1-eth0
```

  Quel est le premier protocol que vous voyez ?
  Pouvez vous identifier les adresses MAC des interlocuteurs ?

- À l'aide de la commande faisant des requêtes http, « curl », faites des requêtes à un serveur Web identifié précédemment :

```
xterm
h1:~# curl -iv http://nom_de_domaine_identifié
```

  Pouvez vous obtenir la page d'accueil de « caldera » ?

- Comment est configuré votre routage ?

```
xterm
h1:~# ip route
```

- Testez la connectivité vers Internet avec :

```
xterm
h1:~# ping -c 1 8.8.8.8
```

c. Connectez vous sur le routeur « r1 » :

```
xterm
$ docker exec -it r1 bash
```

- Consultez la configuration du firewall « NetFilter » :

```
xterm
r1:~# iptables -t filter -nvL
r1:~# iptables -t nat -nvL
```

  Expliquez pourquoi « h1 » et « h3 » bénéficient de la connectivité vers Internet.

- À l'aide de la commande « ss », « socket state » :

```
xterm
r1:~# ss -tlnp
r1:~# ss -ulnp
```

  Identifiez les services que propose « r1 ».

- Expliquez pourquoi « h1 » a pu avoir accès à « caldera » ?

Utilisation et configuration de VLAN

Questions :

d. Pour consulter la liste des ports du switch « interne » :

xterm
```
$ sudo ovs-vsctl list-ports interne
interne-auth
interne-caldera
interne-client
interne-h1
interne-h3
interne-r1
interne-web
```

Vous lancerez un « ping » de « h1 » vers « h3 ».
Vous configurerez le port de connexion de la machine « h1 », appelé « interne-h1 » sur le VLAN 10 :

xterm
```
$ sudo ovs-vsctl set port interne-h1 tag=10
$ sudo ovs-vsctl show
74329121-5f19-4fbd-9a22-bf4d4cd315e7
    Bridge interne
        Port interne-client
            Interface interne-client
        Port interne-auth
            Interface interne-auth
        Port interne-caldera
            Interface interne-caldera
        Port interne-h1
            tag: 10
            Interface interne-h1
...
$ sudo ovs-vsctl get port interne-h1 tag
10
```

Pour supprimer l'appartenance à un VLAN :

xterm
```
$ sudo ovs-vsctl set port interne-h1 'tag=[]'
```

Qu'avez vous observé sur le « ping », pendant ces opérations ?

e. Mettez les postes « h1 » et « h3 » dans le VLAN 10.
Est-ce que le « ping » fonctionne ?

Vous ajouterez le script « build_hack » suivant :

```
#!/bin/bash

docker run --rm \
  --network=none \
  -dit --name hack --hostname hack --domainname lab \
  --entrypoint bash nicolaka/netshoot
PID_hack=$(docker inspect -f '{{.State.Pid}}' hack)
echo "Ajout hack : $PID_hack"
ip link add hack-eth0 type veth peer name interne-hack
ip link set hack-eth0 netns $PID_hack
ovs-vsctl add-port interne interne-hack
ip link set interne-hack up
nsenter -t $PID_hack -n ip address add 172.30.0.90/24 dev hack-eth0
nsenter -t $PID_hack -n ip link set hack-eth0 up
```

Questions :

f. Quelle est la configuration de la nouvelle machine que l'on ajoute au réseau et quelles sont ses capacités ?

g. En entrant sur la machine « hack », vous pourrez testez le « ping » :

xterm
```
 docker exec -it hack bash
hack:~# ping 172.30.0.107
PING 172.30.0.107 (172.30.0.107) 56(84) bytes of data.
From 172.30.0.90 icmp_seq=1 Destination Host Unreachable
From 172.30.0.90 icmp_seq=2 Destination Host Unreachable
^C
--- 172.30.0.107 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 5113ms
```

Est-ce normal ?

Si vous configurez le port « interne-hack », pour rejoindre le VLAN 10 :

xterm

```
hack:~# ping 172.30.0.107
PING 172.30.0.107 (172.30.0.107) 56(84) bytes of data.
64 bytes from 172.30.0.107: icmp_seq=1 ttl=64 time=2079 ms
64 bytes from 172.30.0.107: icmp_seq=2 ttl=64 time=1024 ms
^C
--- 172.30.0.107 ping statistics ---
2 packets transmitted, 6 received, 0% packet loss, time 5152ms
rtt min/avg/max/mdev = 0.040/517.280/2079.016/792.251 ms, pipe 3
```

Maintenant, nous allons utiliser « Scapy » pour envoyer des trames avec étiquette VLAN, « 802.1Q » :

xterm

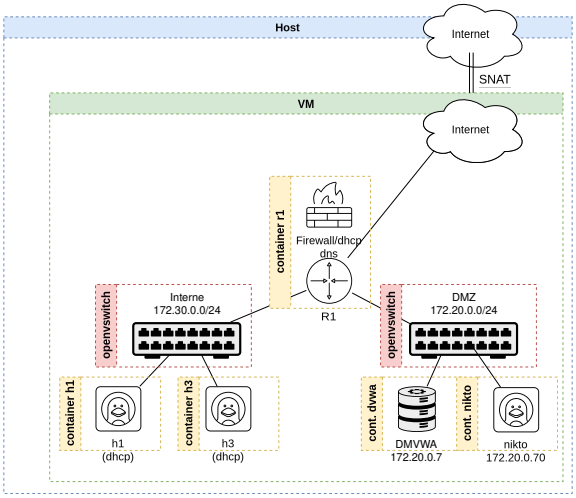
```
hack:~# scapy
                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.6.1
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | Craft me if you can.
       scccccp///pSP///p          p//Y   |                   -- IPv6 layer
      sY/////////y  caa           S//P   |
       cayCyayP//Ya              pY/Ya
        sY/PsY////YCc           aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs

>>> p = Ether()/Dot1Q(vlan=10)/IP(dst='172.30.0.107')/ICMP()
>>> p
<Ether  type=n_802_1Q |<Dot1Q  vlan=10 type=IPv4 |<IP  frag=0 proto=icmp
dst=172.30.0.107 |<ICMP  |>>>>
>>> srp(p)
Begin emission
.
Finished sending 1 packets
^C
Received 1 packets, got 0 answers, remaining 1 packets
(<Results: TCP:0 UDP:0 ICMP:0 Other:0>, <Unanswered: TCP:0 UDP:0 ICMP:1
Other:0>)
```

Suivant que « h1 », « h2 » et « hack » sont dans le même VLAN (ou sans VLAN), qu'observe-t-on ?
Pour envoyer un « ping » sans étiquette VLAN :

xterm

```
>>> p = Ether()/IP(dst='172.30.0.107')/ICMP()
>>> srp(p)
Begin emission
Finished sending 1 packets
*
Received 1 packets, got 1 answers, remaining 0 packets
(<Results: TCP:0 UDP:0 ICMP:1 Other:0>, <Unanswered: TCP:0 UDP:0 ICMP:0
Other:0>)
>>> _[0]
<Results: TCP:0 UDP:0 ICMP:1 Other:0>
>>> _[0]
QueryAnswer(query=<Ether  type=IPv4 |<IP  frag=0 proto=icmp dst=172.30.0.107
|<ICMP  |>>>, answer=<Ether  dst=92:75:db:79:25:b8 src=26:a0:6a:ff:11:0f
type=IPv4 |<IP  version=4 ihl=5 tos=0x0 len=28 id=28299 flags= frag=0 ttl=64
proto=icmp chksum=0xb354 src=172.30.0.107 dst=172.30.0.90 |<ICMP
type=echo-reply code=0 chksum=0x0 id=0x0 seq=0x0 unused=b'' |>>>)
```

On va ajouter notre « DMZ » :



DMZ

Par définition, une DMZ est un réseau hébergeant des services accessibles depuis Internet et séparé du réseau interne de l'entreprise par un firewall.

Vous exécuterez dans votre VM, le script :

xterm
```
$ ./build_services
```

Questions :

h. Vous décrirez ce que le script a fait, et quels sont les nœuds ajoutés.

i. Pouvez vous faire un ping depuis « h1 » vers « dvwa » ?

j. Vous vous connecterez au container « dvwa ».
Ce container ne dispose pas de toutes les commandes d'un linux et il va nous falloir nous y connecter avec une autre méthode qui va nous permettre d'intéragir avec sa pile tcp/ip avec les commandes présentes dans la VM :

xterm
```
$ docker inspect -f '{{.State.Pid}}' dvwa
34163
$ sudo nsenter -t 34163 -n bash
```

k. Consultez la table de routage de « dvwa » et modifiez la pour permettre à « h1 » de réussir à faire un ping vers « dvwa ».

xterm
```
# ip route
```

On va ajouter un nœud d'audit disposant de l'outil « nikto ».
Sur la VM, on exécute qui va ajouter le container « nikto » :

xterm
```
$ sudo ./build_audit
```

On va entrer dans le container et lancer l'adit de « dvwa » :

xterm
```
$ docker exec -it nikto bash
a97d3d76e85a:/# /usr/bin/nikto.pl -h
Option host requires an argument

       -config+            Use this config file
       -Display+           Turn on/off display outputs
       -dbcheck            check database and other key files for syntax errors
       -Format+            save file (-o) format
       -Help               Extended help information
       -host+              target host
       -id+                Host authentication to use, format is id:pass or id:pass:realm
...
       -Version            Print plugin and database versions
       -vhost+             Virtual host (for Host header)
         + requires a value

   Note: This is the short help output. Use -H for full help text.
```

Configuration de trunking pour répartition de VLANs entre switches

Vous ajouterez le script suivant :

```
#!/bin/bash

# ajout et connexion du trunk
ip link add interne-trunk type veth peer name dmz-trunk
ovs-vsctl add-port interne interne-trunk
ovs-vsctl add-port dmz dmz-trunk
ip link set interne-trunk up
ip link set dmz-trunk up

# configuration du trunking
ovs-vsctl set port dmz-trunk trunks=10
ovs-vsctl set port interne-trunk trunks=10
```

Questions :

l. Une fois le « trunk » établi entre les switches « interne » et « dmz », vous sniffrez le trafic sur ce trunk :

xterm
```
$ sudo tcpdump -lnvveX -i dmz-trunk
```

En faisant un « ping » depuis « h1 », configuré dans le VLAN 10, vers le routeur :

xterm
```
# ping 172.30.0.254
```

Qu'est-ce que vous observez ?

Expliquez ce qui se passe ?

Vous ajouterez le script « build_hackd » suivant :

```
#!/bin/bash

docker run --rm \
  --network=none \
  -dit --name hackd --hostname hackd --domainname lab \
  --entrypoint bash nicolaka/netshoot
PID_hackd=$(docker inspect -f '{{.State.Pid}}' hackd)
echo "Ajout hackd : $PID_hackd"
ip link add hackd-eth0 type veth peer name dmz-hackd
ip link set hackd-eth0 netns $PID_hackd
ovs-vsctl add-port dmz dmz-hackd
ip link set dmz-hackd up
nsenter -t $PID_hackd -n ip address add 172.20.0.90/24 dev hackd-eth0
nsenter -t $PID_hackd -n ip link set hackd-eth0 up
```

Questions :

m. Qu'est-ce que fait le script ?

Configurez ce poste « hackd », pour rejoindre le VLAN 10 avec « h1 », « h3 » et « hack ».

Est-ce que tout fonctionne ?

Que faut-il faire ?

Voici une documentation de la configuration des trunks :

vlan_mode	Behavior for 802.1Q frames	Typical use case
trunk	Passes only tagged frames (from the trunks list). Untagged frames treated as VLAN 0.	Inter-switch trunks, patch ports, tunnels
native-tagged	Like trunk, but untagged incoming frames get the tag= VLAN (native VLAN is also tagged on output).	Hybrid trunk with native VLAN
native-untagged	Like trunk, but native VLAN traffic exits untagged.	Connecting to devices that don't expect tags on native VLAN
access	Strips the tag on output. Drops incoming tagged frames that don't match the access VLAN.	VMs or end hosts (tag is removed)

À l'aide de Scapy, pouvez vous tester les différents modes de trunking ?

Configuration du firewall

Questions :

o. Filtrez les accès du réseau « `172.30.0.0/24` » vers le réseau « `172.20.0.0/24` » sur « r1 ». Autorisez uniquement « h1 » à avoir accès à « DVWA ».

p. Configurez du « DNAT », « port forwarding », sur « r1 », permettant à « h3 » d'atteindre « dvwa ». As-t-on besoin de « SNAT » ?

Vous observerez les états des connexions sur « r1 » avec :

```
xterm
$ sudo conntrack -L
```

Il vous faudra l'installer sur r1 :

```
xterm
# apk add conntrack-tools
```

Attention : du à la nature transitoire de docker, l'ajout de cette commande sera perdue lors de la relance du container.

Pouvez vous voir le « SNAT »/« DNAT » qui a lieu ?

Utilisation de la crypto et des tokens sécurisés

Vous diposez de deux nœuds :

- ▷ « client » qui disposent de scripts
- ▷ « auth » qui fait tourner un serveur logiciel basé Python qui sert à faire la vérification.

En vous plaçant sur « client » :

```
xterm
# ./creer_token
```

Affiche un token « JWT » créé localement.

Obtient un token depuis le serveur.

```
xterm
# ./requete_token
```

Questions :

a. Est-ce que les tokens sont identiques ?
Quelle(s) propriété(s) « D-I-C-T » cela fournit ?
Comment sont-ils construits ?
Sur quoi repose la confiance ?
Quelles sont les opérations cryptographiques utilisées et qu'est-ce qu'elles assurent ?

b. Utilisez le script « requete_protegee » pour vérifier le token auprès du serveur.
Est-ce que vous pouvez observer la transaction avec tcpdump ? (utilisez les options « lnvveX »)

c. Recommencez avec les scripts TOTP :

```
xterm
# ./requete_totp
```

Donne un code TOTP localement.

Verifie le code sur le serveur.

```
xterm
# ./verifier_totp
```

Quelles différences avec le token précédent ?

Création de certificat électronique et mise en place d'une sécurisation TLS

On va mettre en place une PKI, « Public Key Infrastructure » et émettre des certificats pour configuer un accès TLS à nos services.
On va utiliser l'outil « mkcert », en l'installant sur la VM :

xterm
```
$ sudo apt install mkcert libnss3-tools
```

On va créer le certificat ato-signé de l'AC, « Autorité de certification » et l'installer automatiquement dans le navigateur firefox :

xterm
```
$ mkcert -install
```

Puis on va créer un certificat pour un FQDN :

xterm
```
$ mkcert -key-file proxy.key -cert-file proxy.crt proxy.lab dvwa.lab '*.lab'
127.87.0.10
```

Vous obtenez deux fichiers :

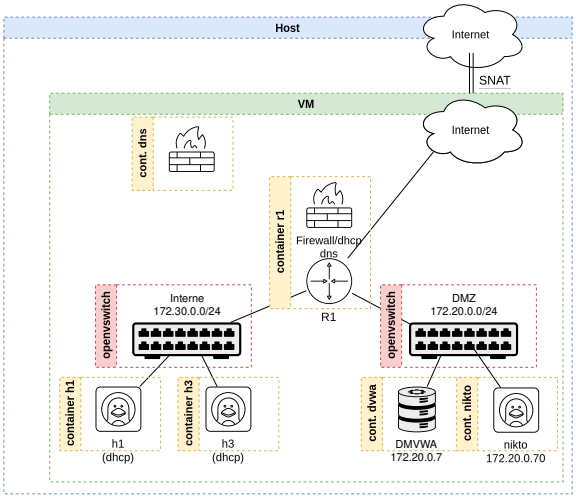
- ▷ la clé privée dans « proxy.key » ;
- ▷ le certificat contenant la clé publique et la signature de l'AC dans « proxy.crt ».

On va héberger un serveur DNS localement et tester qu'il fonctionne correctement avec la commande « dig » :

xterm
```
$ cd DOCKER_DNS
$ docker compose up -d
$ dig @127.87.0.1 +short p-fb.net
198.245.60.92
$ dig @127.87.0.1 +short lab
127.87.0.10
```

Pour remplacer le serveur DNS utilisé par la VM par celui que l'on vient de configurer :

xterm
```
$ sudo resolvectl dns enp1s0 127.87.0.1
```



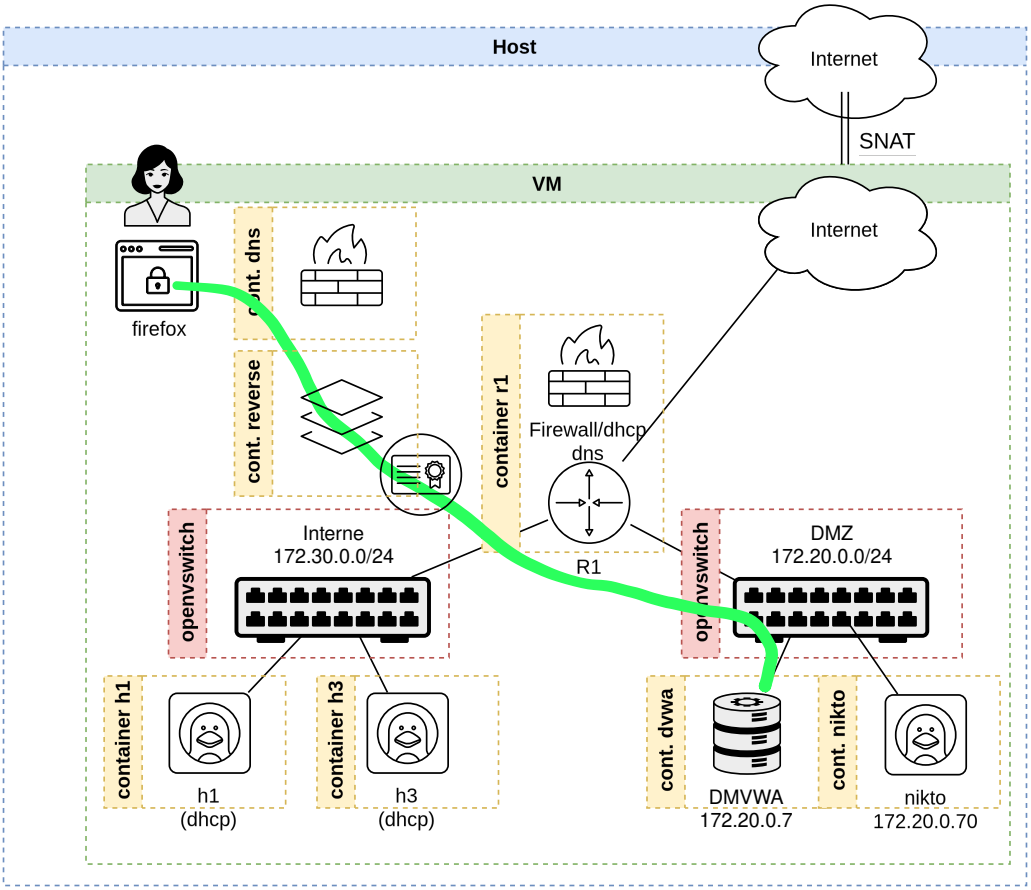
Vous pouvez tester les requêtes DNS directement :

xterm
```
$ dig +short p-fb.net
198.245.60.92
$ dig +short proxy.lab
127.87.0.10
```

Vous pouvez modifier le fichier de configuration «dnsmasq.conf» pour inclure les noms de domaines que vous voulez.
Et à chaque fois, en étant dans le répertoire correct :

xterm
```
$ docker compose down
$ docker compose up
```

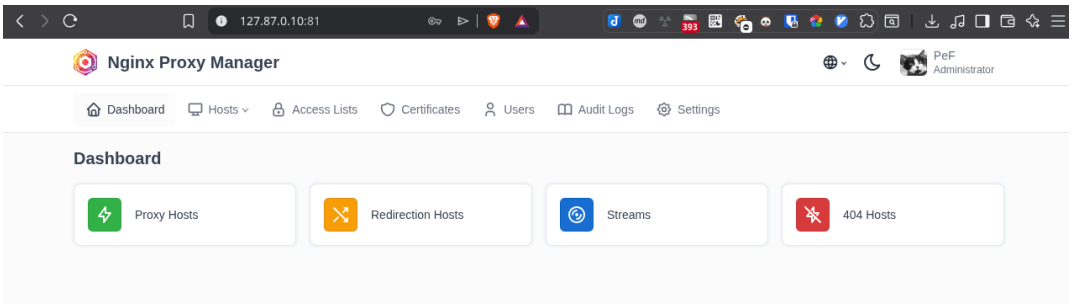
Mise en place d'un reverse proxy sécurisé par TLS



On va intégrer le container «nginx-proxy-manager» sur la VM pour faire du reverse proxy pour nos services dans la DMZ :

```
xterm
$ cd DOCKER_REVERSE_PROXY
$ ./build_reverse_proxy
```

On aura accès à l'interface du reverse proxy sur l'url « http://proxy.lab:81 » ou « http://127.87.0.10:81 » :



Par exemple pour associer notre certificat et clé à un reverse proxy.

***Questions** :*

a. *Créez un reverse proxy pour avoir un accès TLS sur DVWA.*