

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
  
```

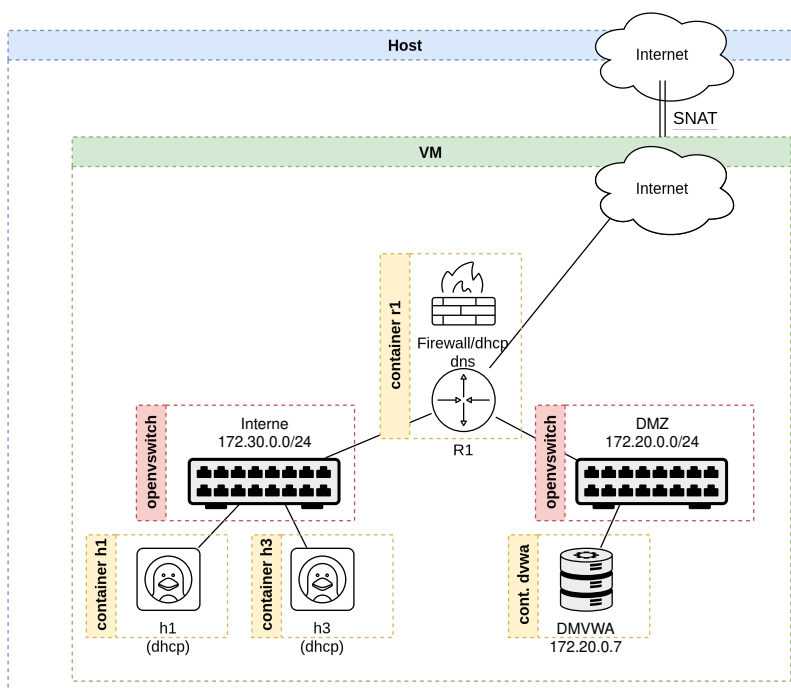
- Vous créez les images des différents systèmes Linux utilisés par les différents container :

```

xterm
$ ./create_images
  
```

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

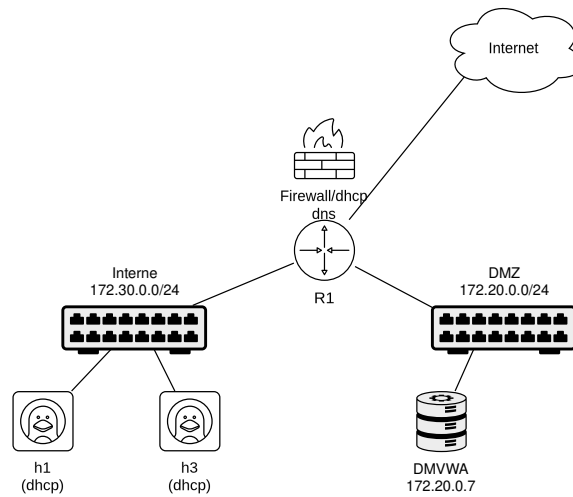
■ ■ ■ Présentation de la plateforme



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'association « adresse MAC/adresse IP »
- ◊ Sniffez les paquets correspondants à ces pings avec « tcpdump » :

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

Quel est le premier protocole 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 » :

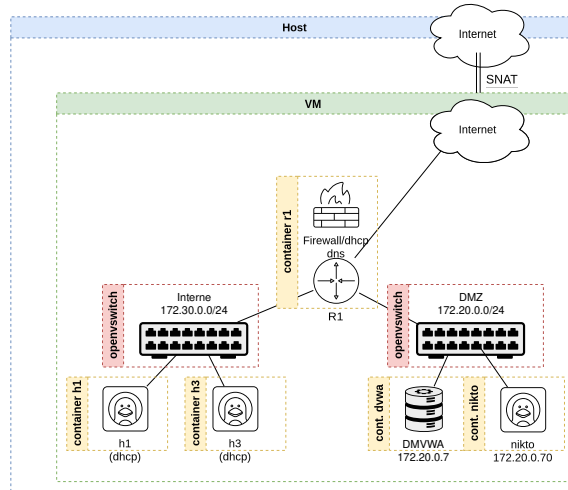
```
xterm
r1:~# ss -tlnp
r1:~# ss -ulnp
```

Identifiez les services que propose « r1 ».

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

■ ■ ■ Audit des vulnérabilités

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 :

- d. Vous décrirez ce que le script a fait, et quels sont les nœuds ajoutés.
- e. Pouvez vous faire un ping depuis « h1 » vers « dvwa » ?
- f. 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'intégrer 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
```

- g. 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'audit 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.
```

■ ■ ■ Utilisation de la crypto et des tokens sécurisés

Vous disposez de deux nœuds :

- ▷ « client » qui dispose 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 :

- 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 ?
- 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 « InvveX »)
- Recommencez avec les scripts TOTP :

```
xterm
# ./requete_totp
```

Donne un code TOTP localement.

Vérifie 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 configurer 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 auto-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 :

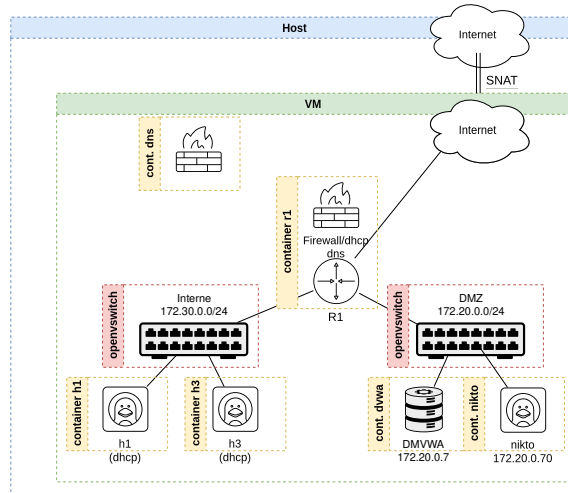
- ▷ la clé privée dans « proxy.key » ;
- ▷ le certificat contenant la clé publique et la signature de l'AC dans « proxy.crt ».

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.

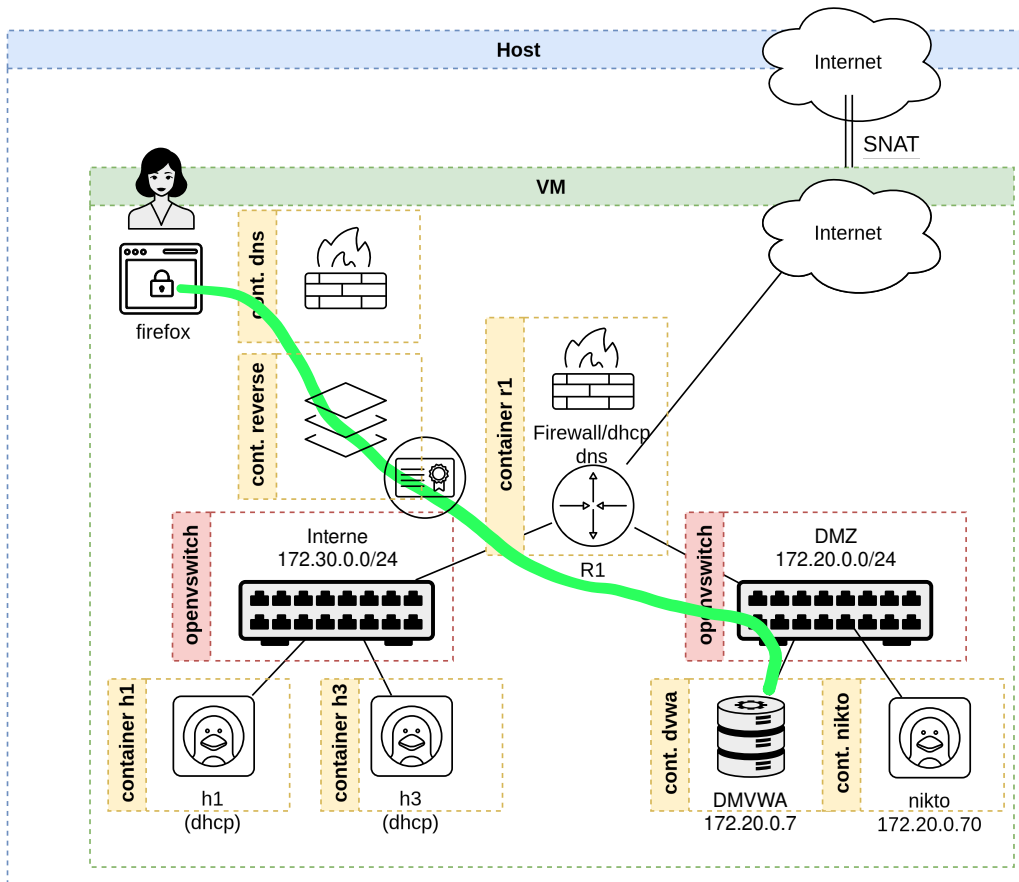
Et à chaque fois, en étant dans le répertoire correct :

```

xterm
$ docker compose down
$ docker compose up

```

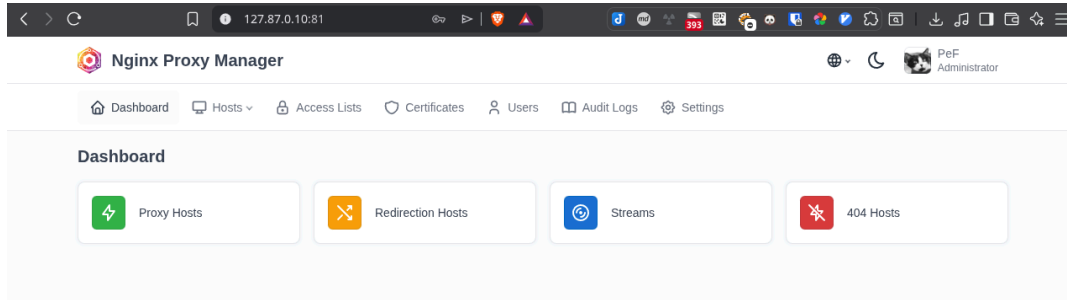
■ ■ ■ Mise en place d'un reverse proxy sécurisé par TLS



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.