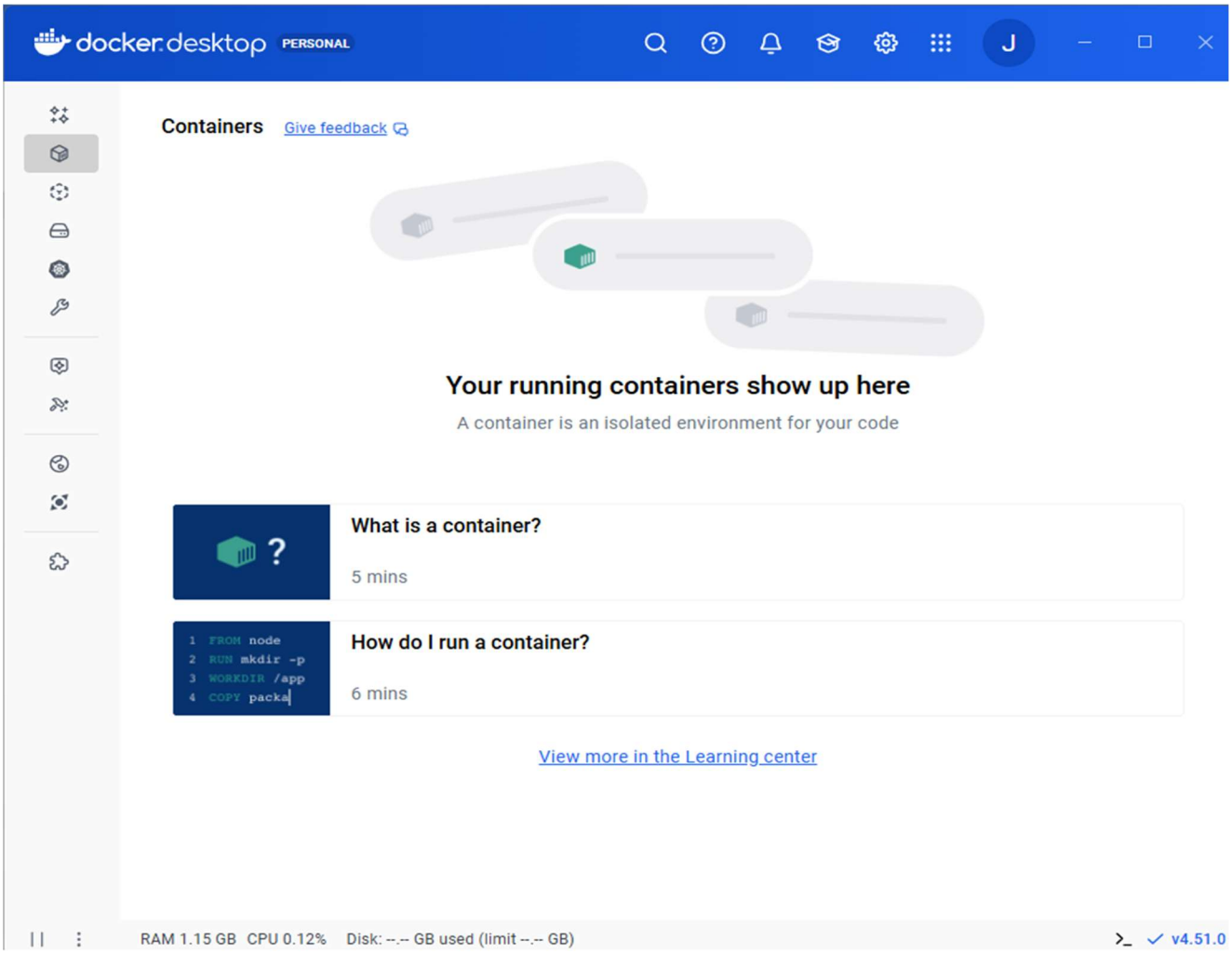
TP1

CYBERSECURITE

AGBENONZAN KOSSIVI JACQUES JUNIOR
SANOGO SOULEYMANE

1) Lancer l'environnement

- Lancez Docker DeskTop



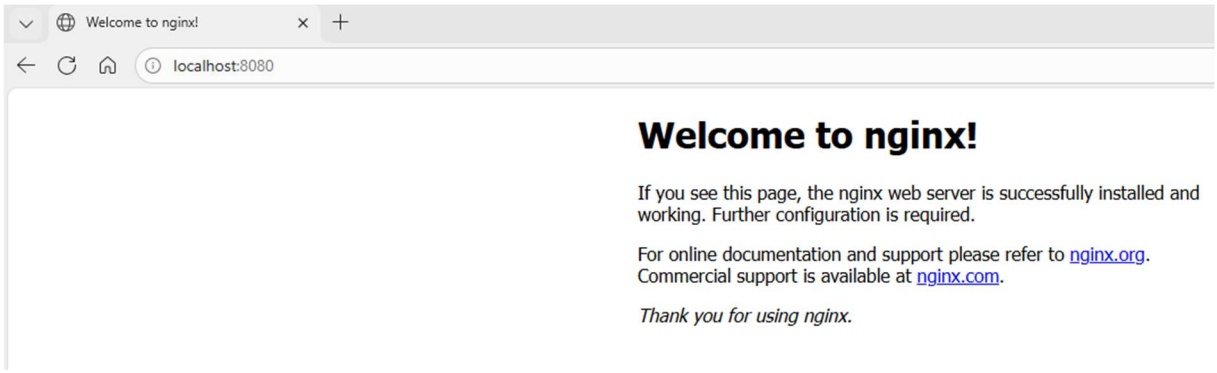
- Ouvrez un terminal dans ce dossier.
- Lancez : **`docker compose up -d`**

- Nous avons ouvert docker, puis via notre invite de commande windows cmd nous avons lancer `docker compose up -d` pour lancer le conteneur.

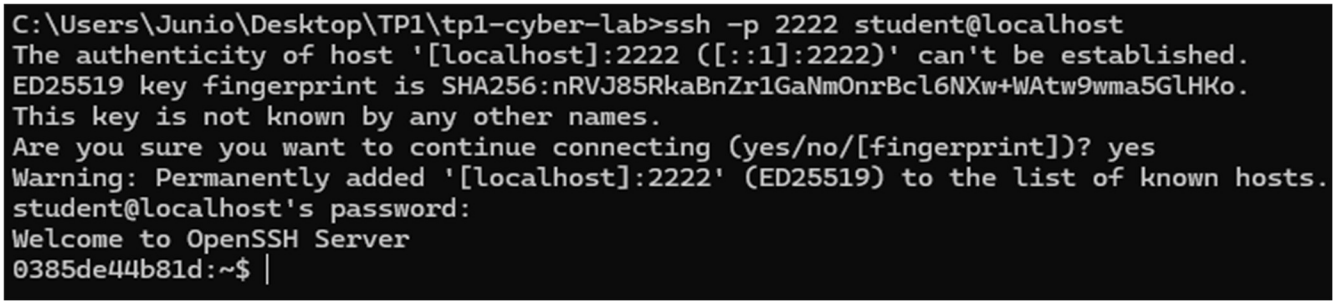
-

- Vérifiez :

- HTTP : ouvrez `http://localhost:8080`

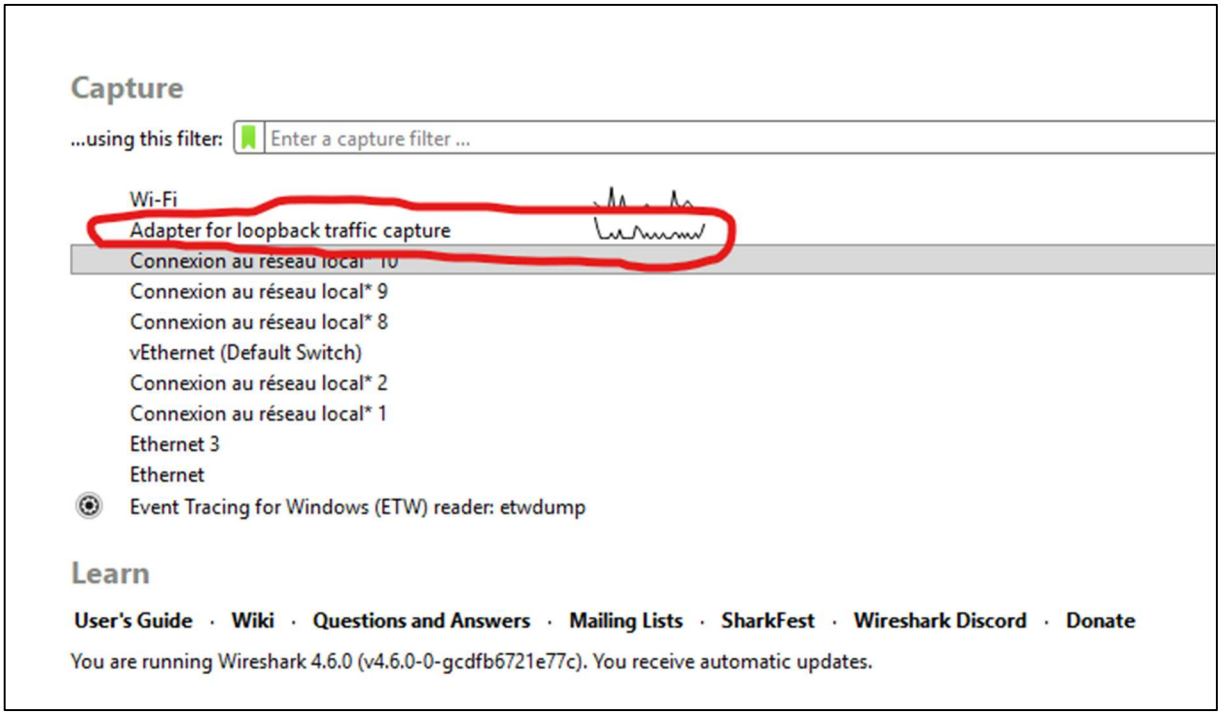


- (Optionnel) SSH : `ssh -p 2222 student@localhost` (mdp `student`)
- Nous nous sommes connecté à la machine distante via : - (Optionnel) SSH : `ssh -p 2222 student@localhost` (mdp `student`)



2. Démarrer la capture pour les échanges avec le Port 8080

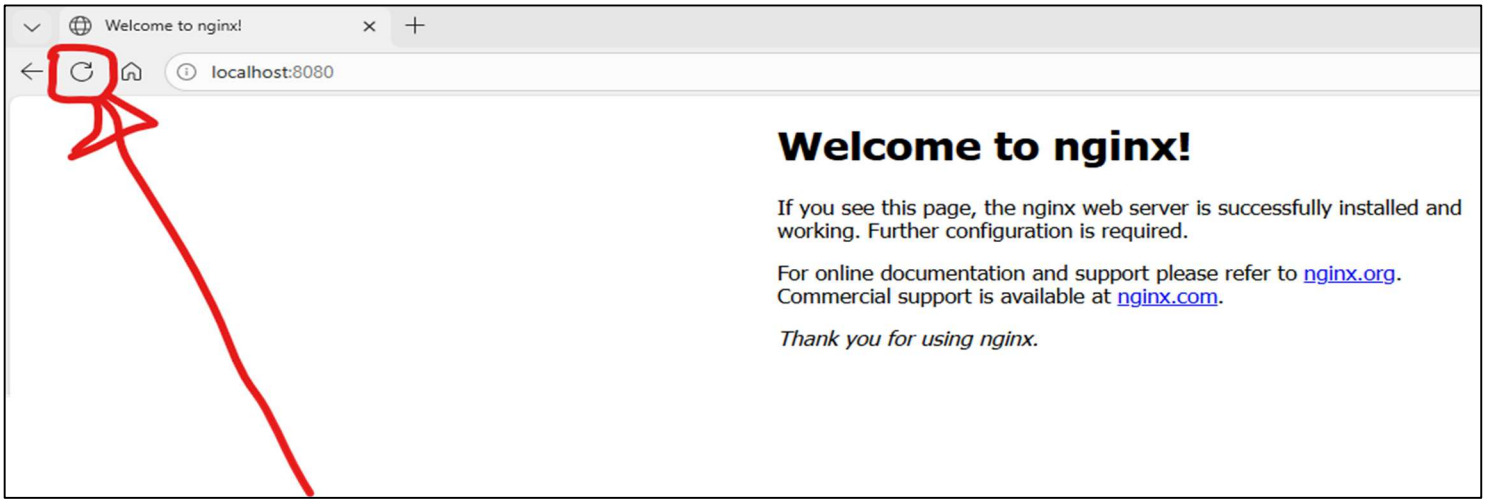
- Ouvrir WireShark
- Double-cliquez sur **"Adapter for loopback traffic capture"**



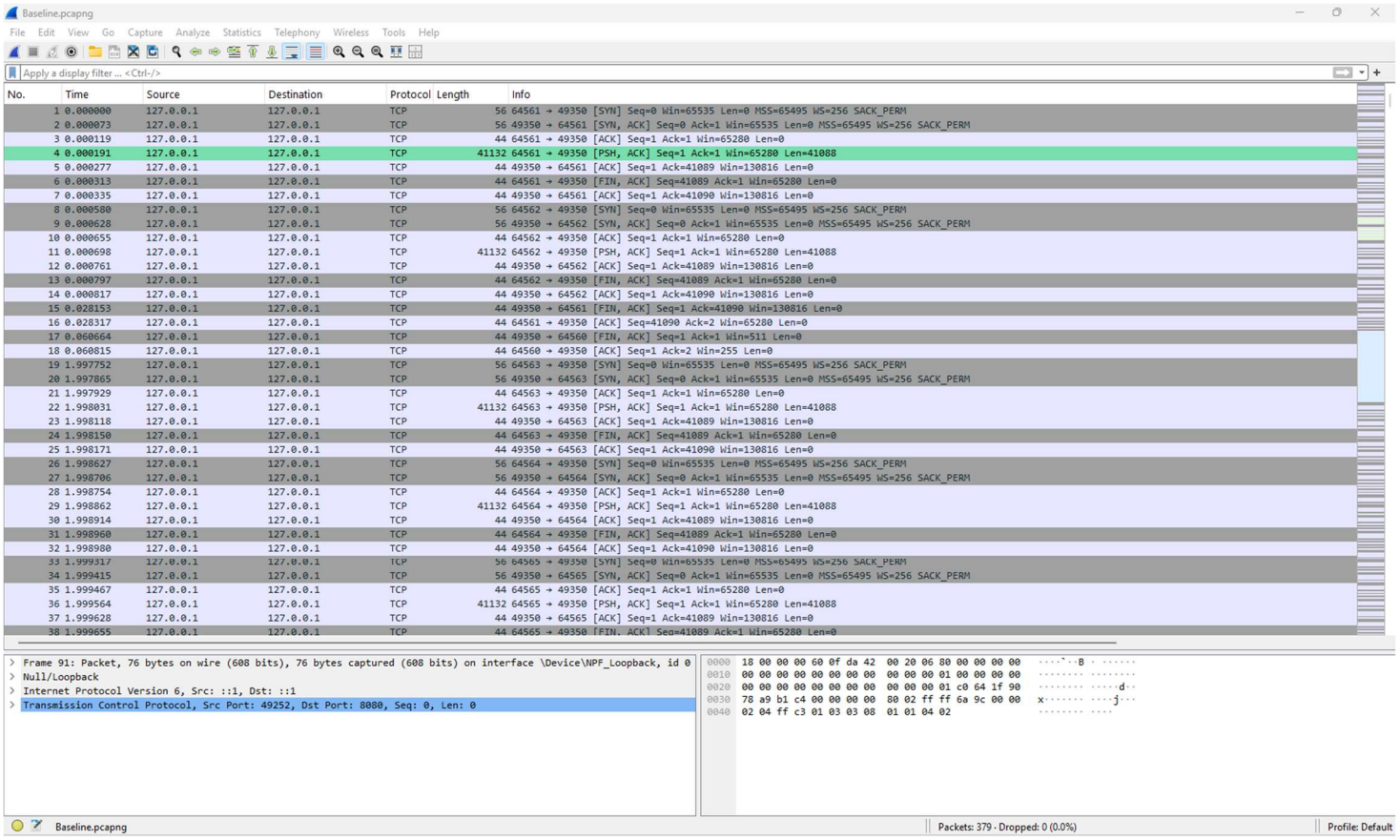
- La capture devrait démarrer automatiquement

3. Générer du trafic

- **Ouvrez votre navigateur**
- **Allez sur :** http://localhost:8080
- **Actualisez la page** 2-3 fois



3. Observer dans Wireshark



Nous voyons :

- **Paquets TCP** avec des adresses **127.0.0.1**
- **Handshakes** SYN → SYN-ACK → ACK
- **Requêtes HTTP** GET vers le port 8080
- **Patientons 30 secondes à 1 minute et arrêtons la capture**

4. Arrêter et sauvegarder

ANALYSE POUR **"ADAPTER FOR LOOPBACK TRAFFIC CAPTURE"**

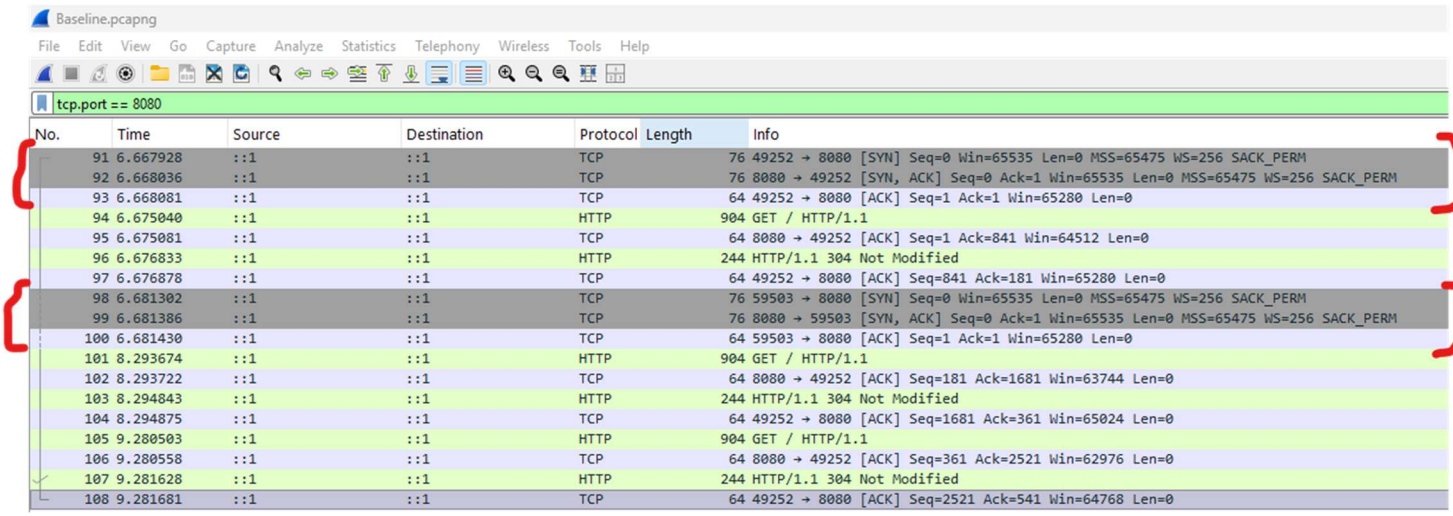
Notes à prendre pendant la capture

Observation	À noter
Nombre de paquets capturés	379
Protocoles identifiés	TCP, http, MDNS,UDP,ICMP,SSDP

1- TCP (Transmission Control Protocol)

On remarque que lorsqu'on applique un filtre « **tcp.port == 8080** »
On observe la séquence suivante entre le **Port 8080 et le Port 49252**, aussi entre le **Port 8080 et le Port 59503** :

- [SYN] → [SYN-ACK] → [ACK]



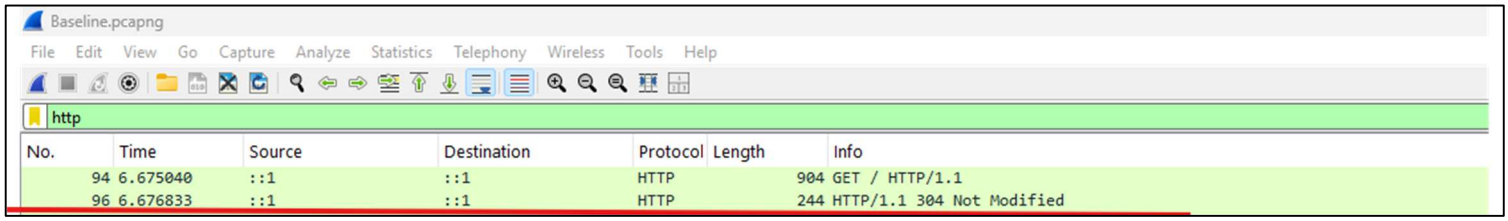
D'après le Cours quand on a la séquence :

- [SYN] → [SYN-ACK] → [ACK] alors le Port 8080 est **Ouvert** Pour TCP
- [RST] → [ACK] : Le port est **Fermé** Pour TCP

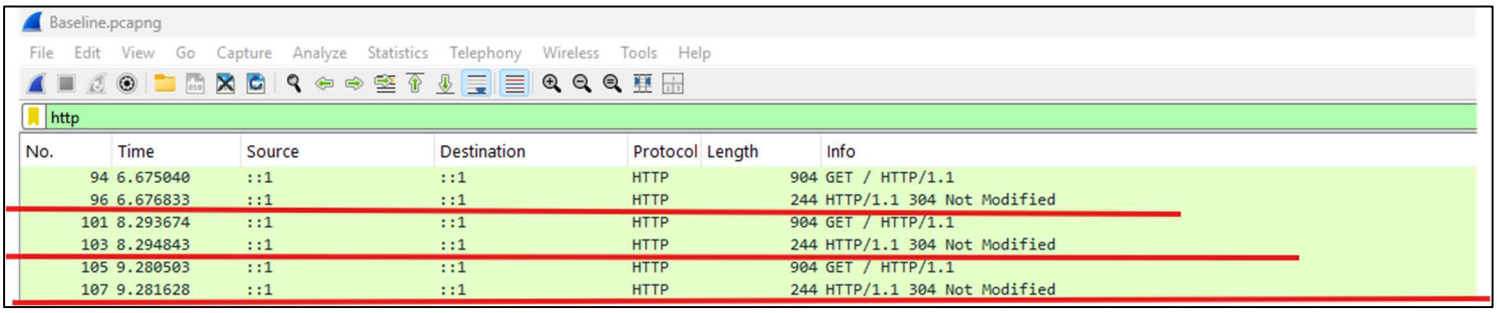
Conclusion : le port 8080 est Ouvert pour TCP dans notre cas

2- HTTP (HyperText Transport Protocol)

On appliquant le filtre à **http** On observe la séquence suivante :



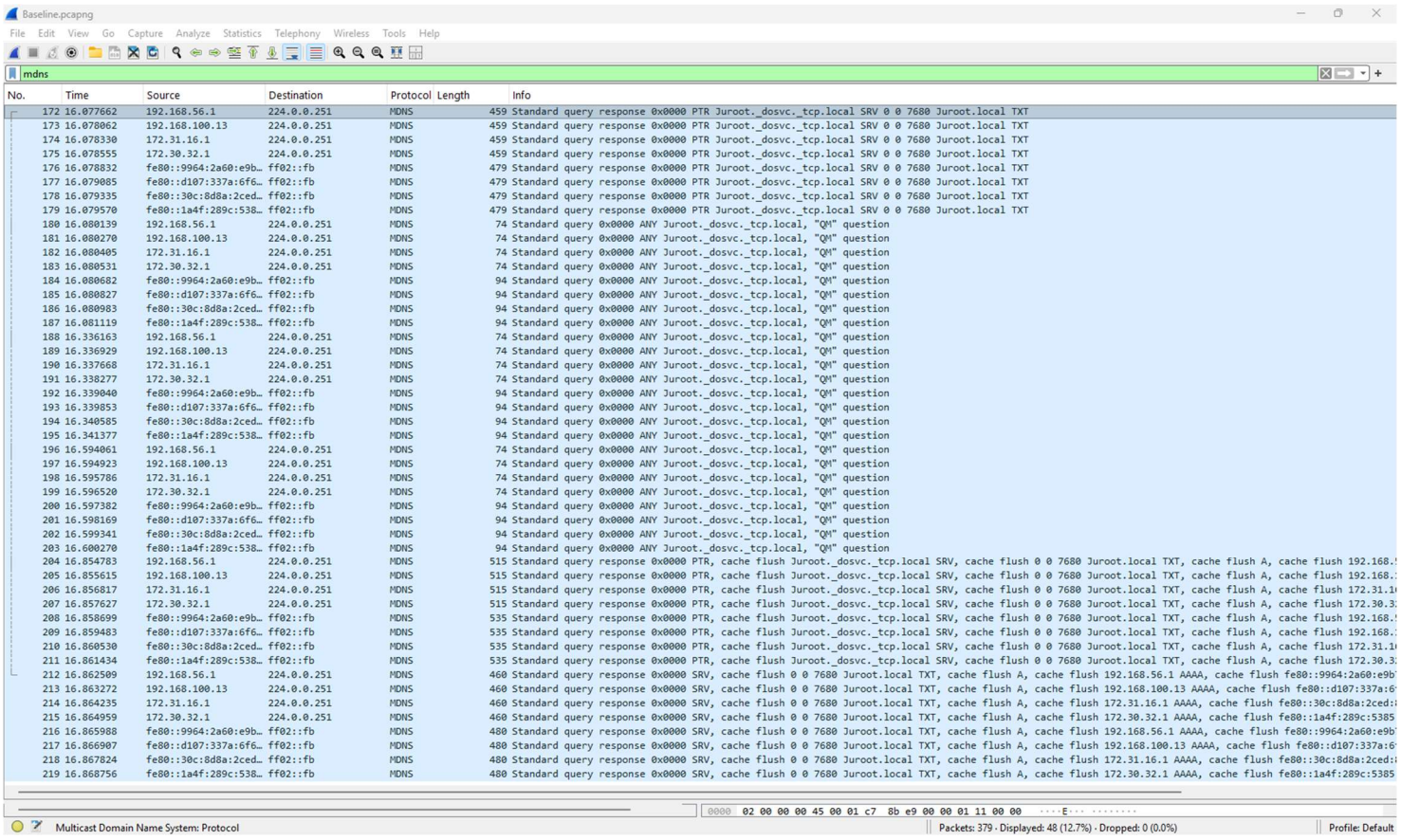
En Actualisant la page **Welcome to nginx!** Trois Fois, On observe la même séquence en boucle



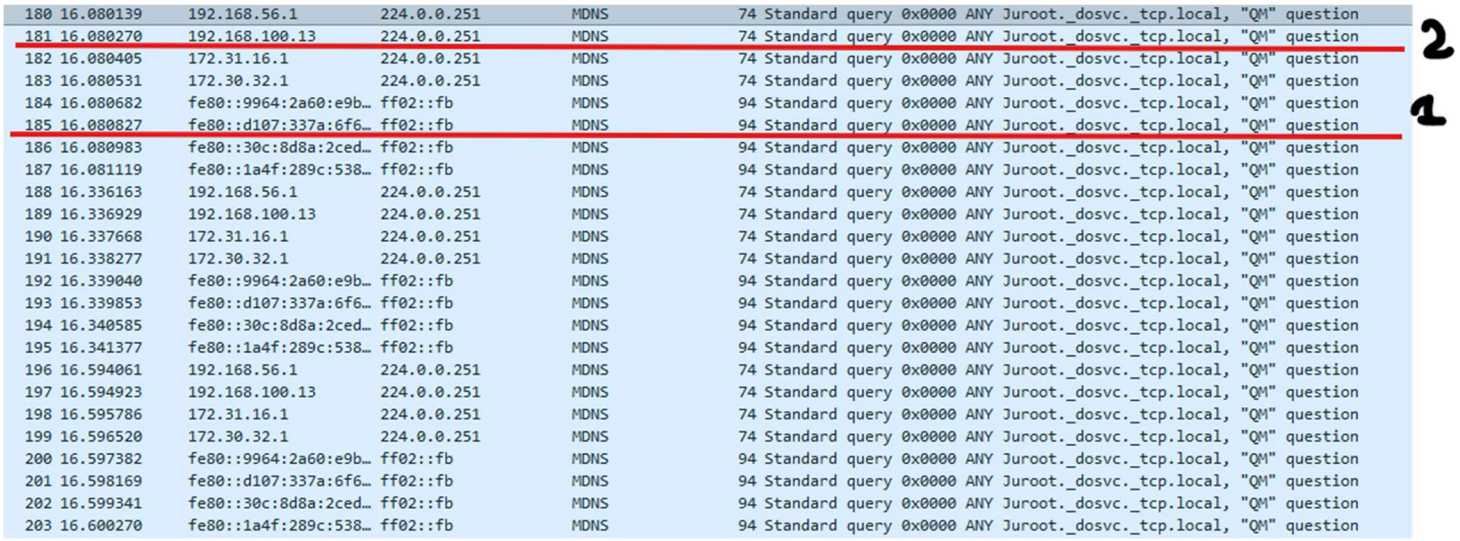
- Etant donnée que nous travaillons en local, Un client Web (nous) **d'adresse ip : :1** intérroge un serveur Web **d'adresse ip : :1** (Encore nous) pour obtenir des Informations sur la page web **HTTP/1.1 (N°96) avec GET/http/1.1**
- Le Serveur lui reponds, **http/1.1 304 Not Modified** : Qui veut dire Chers Clients Vous êtes déjà sur la page que vous demandez (**http/1.1)** pas besoin de recharger toute la page (**304 Not Modified)**
- Ce qui se Comprend vu que nous actualisons simplement la page.

3- MDNS (Multicast Domain Name System)

En appliquant le filtre pour MDNS on a les paquets suivants :



Portion 1 (Requête)

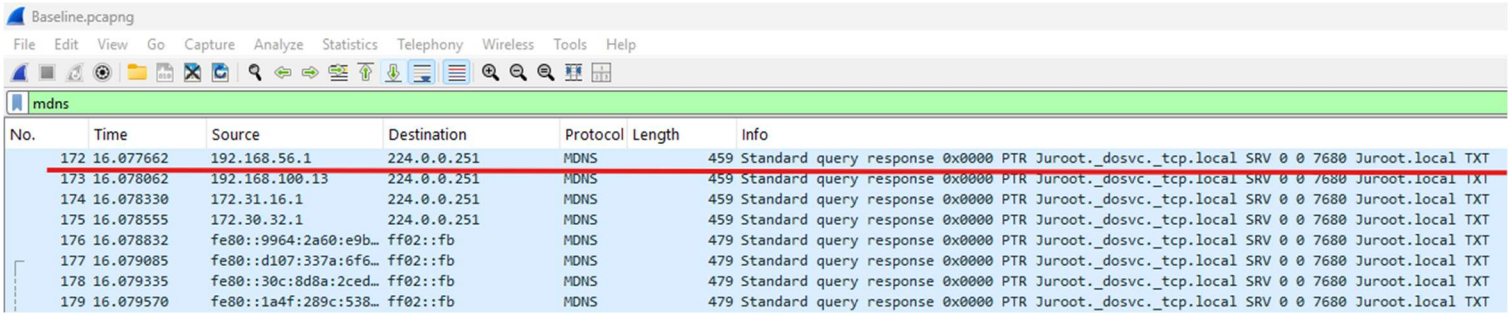


Ici (1) un appareil d'adresse IPV6 **« fe80::d107:337a:6f6d:6cb7 »** émet une requête Multicast (Requête à tout les composant sur le réseau) grâce à l'IPv6 Multicast « **ff02::fb** » via MDNS, pour demander toutes les informations sur le service « **Juroot._dosvc._tcp.local** », il en est de même pour (2) mais avec

source une adresse IPv4 source et une adresse IPv6 de diffusion ou encore dit Multicast.

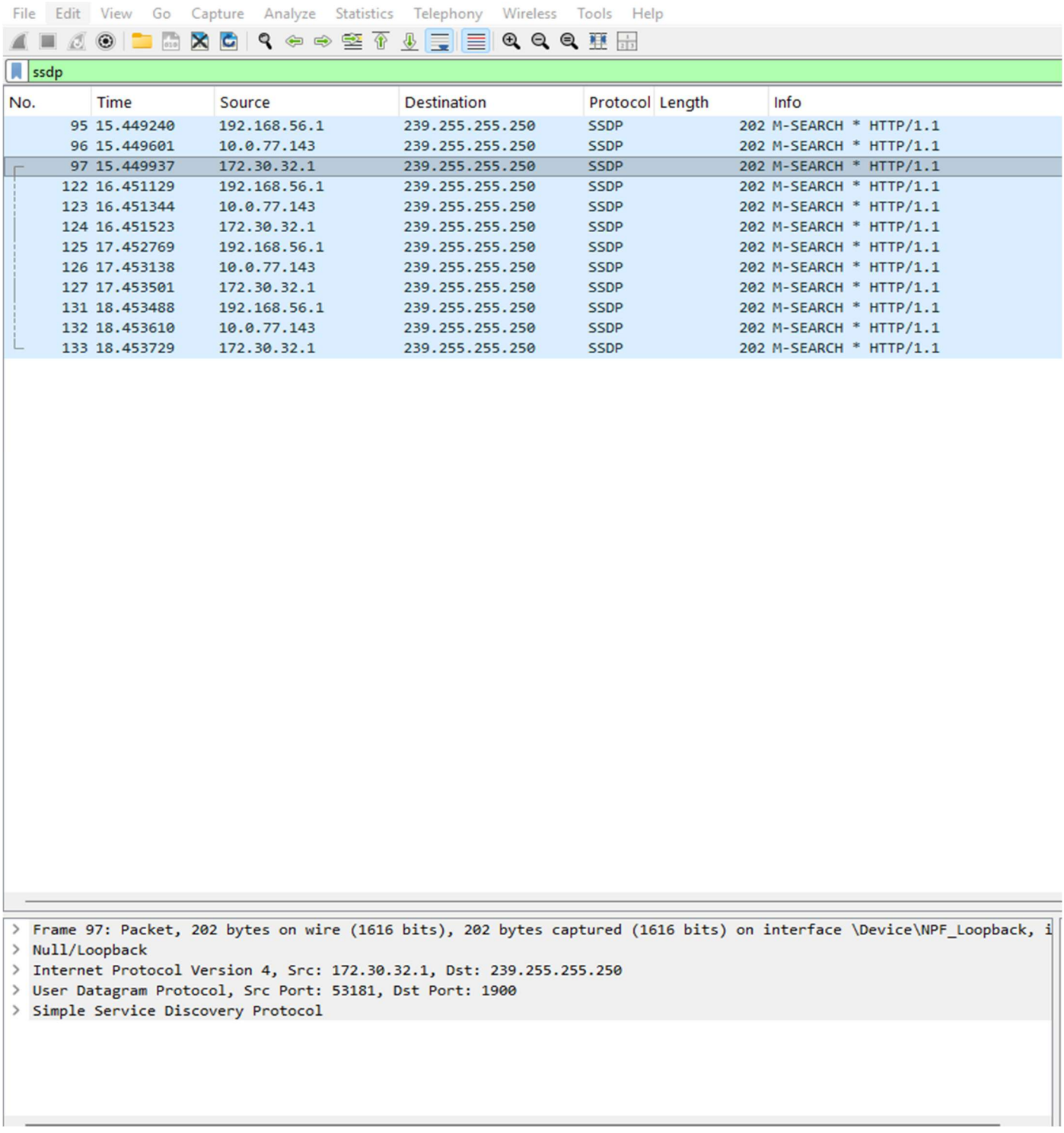
NB : MDNS se loge généralement sur le **Port** : 5353.

Portion 2 (Réponse)



Ici Plusieurs appareil d'adresse **IPv4 (192.168.56.1, 192.168.100.13, 172.31.16.1)** et **IPv6 (fe80::d107:337a:6f6d:6cb7)** émettent des Réponses vers des adresses IPv4 et IPv6 Multicast (donc vers toutes les autres machines du réseau) pour dire « j'ai des Informations sur **Juroot._dosvc._tcp.local** voici tout ce que je sais sur lui dans ce txt (Juroot.local.TXT) ».

4- SSDP (Simple Service Discovery Protocol)



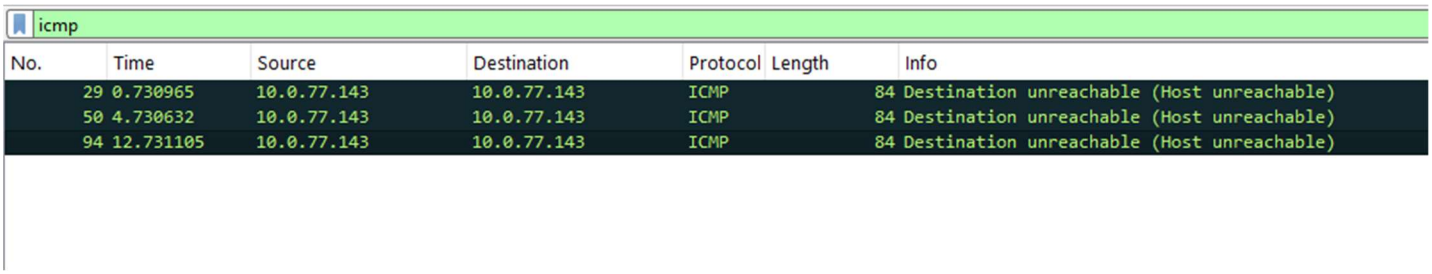
Ici plusieurs machines d'adresse IPv4 (192.168.56.1, 10.0.77.143,…) émettent des messages **Multicast ou de Diffusion** (Message à tous les appareils sur le réseau) via l'adresse **Multicast (239.255.255.250)** pour demander des services web, précisément sur **http/1.1**

NB : SDNS est principalement logé sur le port 1900

REPONSES	SIGNIFICATION
M-SEARCH	• Ce sont des requêtes envoyées par un client pour découvrir des services.
NOTIFY	• Ce sont des annonces envoyées par des périphériques pour dire "je suis là".
Si on a M-SEARCH et NOTIFY	• Le Port est actif

Conclusion : Dans notre cas le port 1900 est fermé dû à l'absence de NOTIFY

5- UDP (User Data Protocol)



REPONSES	SIGNIFICATION
IMCP → Destination unreachable (Host unreachable)	**Port UDP Fermé**
---(Silence)	**Port UDP Ouvert**

Dans notre cas le port UDP est fermé

Étape 2 : Accéder au conteneur attaquant

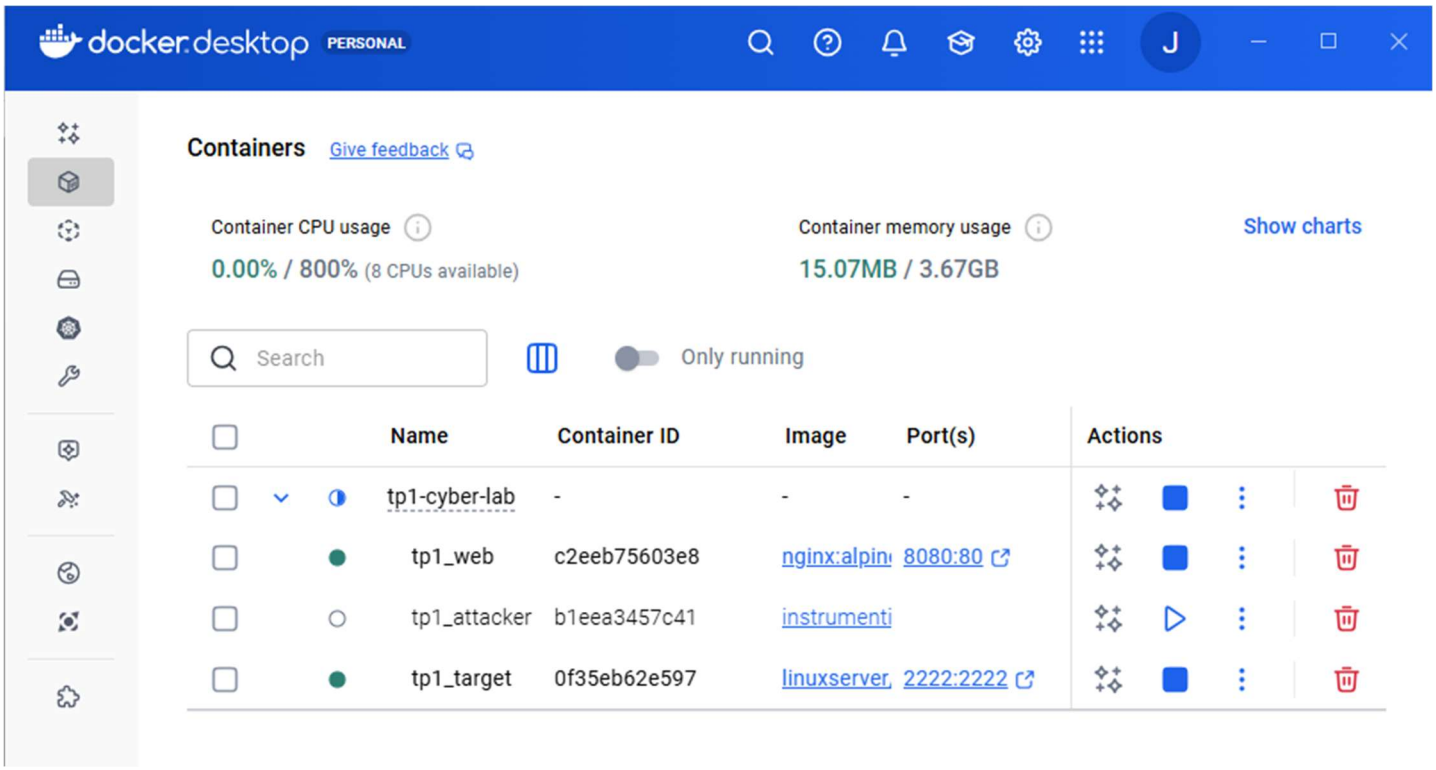
Ouvrez un nouveau terminal/CMD :

Via le Terminal Tapez : **docker compose exec attacker sh**, puis entrez **nmap -sS -p 1-1000 target**

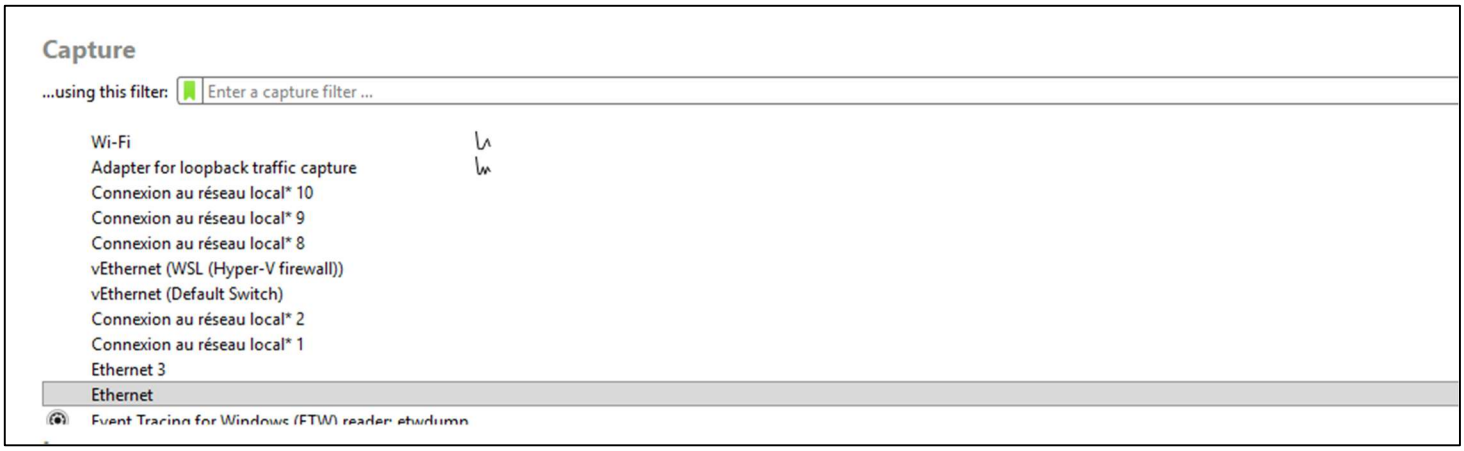
```
C:\Users\Junio\Desktop\TP1\tp1-cyber-lab>docker compose exec attacker sh
/ #
C:\Users\Junio\Desktop\TP1\tp1-cyber-lab>docker compose exec attacker sh
/ # nmap -sS -p 1-1000 target
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-15 12:36 UTC
Nmap scan report for target (172.18.0.4)
Host is up (0.000017s latency).
rDNS record for 172.18.0.4: tp1_target.tp1-cyber-lab_labnet
All 1000 scanned ports on target (172.18.0.4) are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 32:9F:7A:76:8B:5E (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
/ #
C:\Users\Junio\Desktop\TP1\tp1-cyber-lab>
```

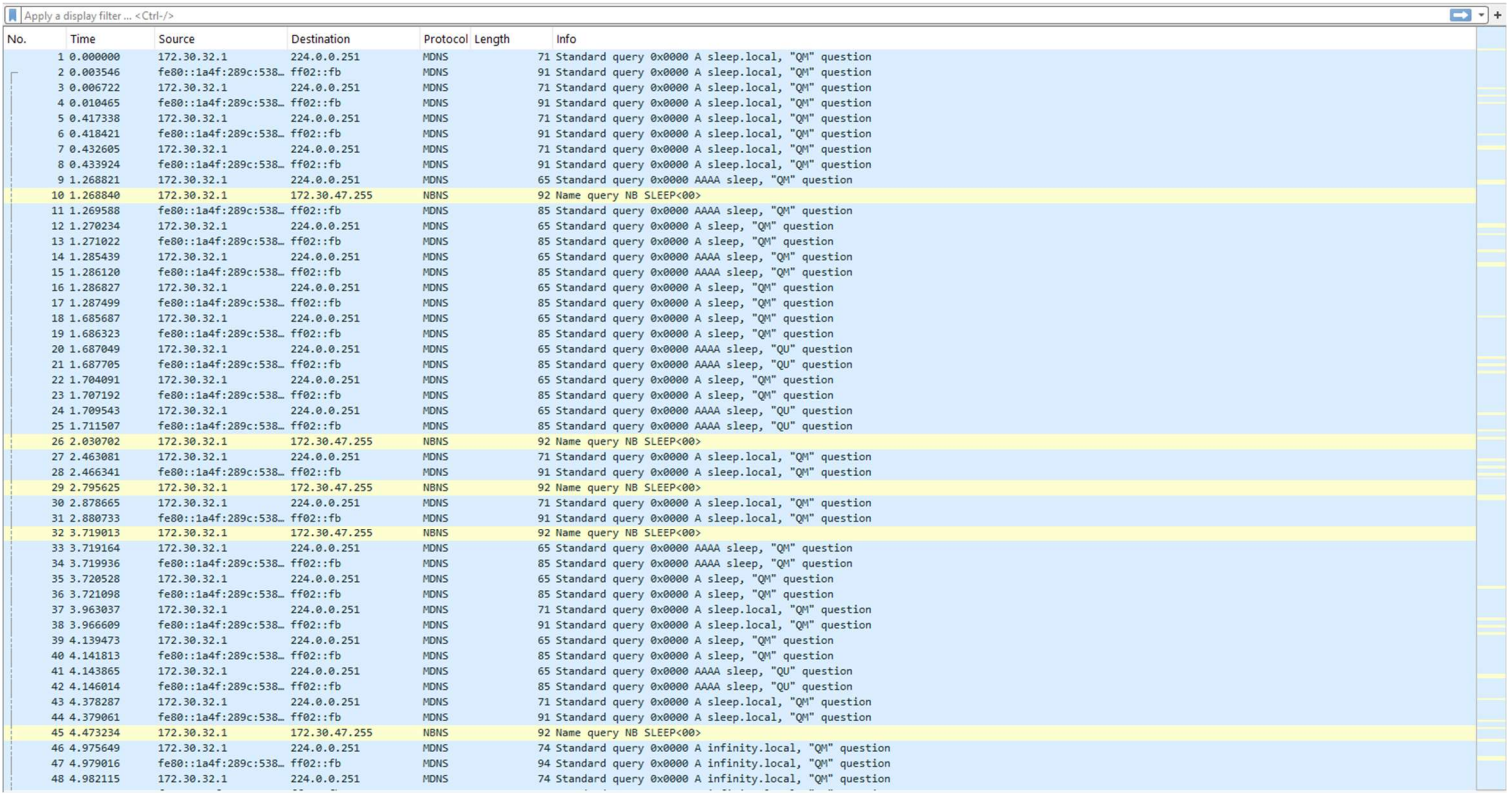
Ou encore lancer la machine via docker



Dans WireShark Double-Cliquez sur vEthernet (WSL(Hyper-V firewall))



Laisser WireShark tournez Pendant 30 secondes à une minute et stoppé la capture



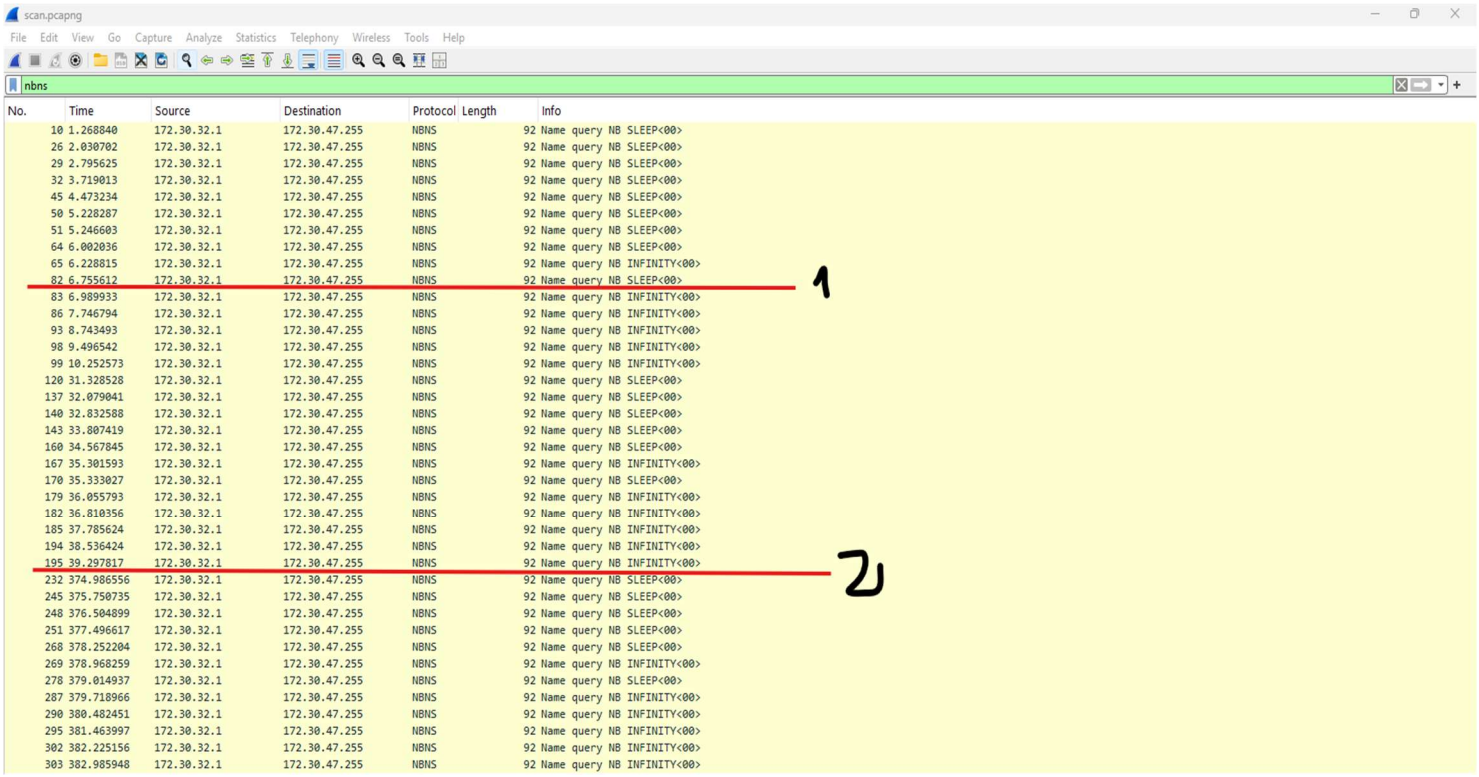
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.30.32.1	224.0.0.251	MDNS	71	Standard query 0x0000 A sleep.local, "QM" question
2	0.003546	fe80::1a4f:289c:538…	ff02::fb	MDNS	91	Standard query 0x0000 A sleep.local, "QM" question
3	0.006722	172.30.32.1	224.0.0.251	MDNS	71	Standard query 0x0000 A sleep.local, "QM" question
4	0.010465	fe80::1a4f:289c:538…	ff02::fb	MDNS	91	Standard query 0x0000 A sleep.local, "QM" question
5	0.417338	172.30.32.1	224.0.0.251	MDNS	71	Standard query 0x0000 A sleep.local, "QM" question
6	0.418421	fe80::1a4f:289c:538…	ff02::fb	MDNS	91	Standard query 0x0000 A sleep.local, "QM" question
7	0.432605	172.30.32.1	224.0.0.251	MDNS	71	Standard query 0x0000 A sleep.local, "QM" question
8	0.433924	fe80::1a4f:289c:538…	ff02::fb	MDNS	91	Standard query 0x0000 A sleep.local, "QM" question
9	1.268821	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 AAAA sleep, "QM" question
10	1.268840	172.30.32.1	172.30.47.255	NBNS	92	Name query NB SLEEP<00>
11	1.269588	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 AAAA sleep, "QM" question
12	1.270234	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 A sleep, "QM" question
13	1.271022	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 A sleep, "QM" question
14	1.285439	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 AAAA sleep, "QM" question
15	1.286120	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 AAAA sleep, "QM" question
16	1.286827	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 A sleep, "QM" question
17	1.287499	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 A sleep, "QM" question
18	1.685687	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 A sleep, "QM" question
19	1.686323	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 A sleep, "QM" question
20	1.687049	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 AAAA sleep, "QU" question
21	1.687705	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 AAAA sleep, "QU" question
22	1.704091	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 A sleep, "QM" question
23	1.707192	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 A sleep, "QM" question
24	1.709543	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 AAAA sleep, "QU" question
25	1.711507	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 AAAA sleep, "QU" question
26	2.030702	172.30.32.1	172.30.47.255	NBNS	92	Name query NB SLEEP<00>
27	2.463081	172.30.32.1	224.0.0.251	MDNS	71	Standard query 0x0000 A sleep.local, "QM" question
28	2.466341	fe80::1a4f:289c:538…	ff02::fb	MDNS	91	Standard query 0x0000 A sleep.local, "QM" question
29	2.795625	172.30.32.1	172.30.47.255	NBNS	92	Name query NB SLEEP<00>
30	2.878665	172.30.32.1	224.0.0.251	MDNS	71	Standard query 0x0000 A sleep.local, "QM" question
31	2.880733	fe80::1a4f:289c:538…	ff02::fb	MDNS	91	Standard query 0x0000 A sleep.local, "QM" question
32	3.719013	172.30.32.1	172.30.47.255	NBNS	92	Name query NB SLEEP<00>
33	3.719164	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 AAAA sleep, "QM" question
34	3.719936	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 AAAA sleep, "QM" question
35	3.720528	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 A sleep, "QM" question
36	3.721098	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 A sleep, "QM" question
37	3.963037	172.30.32.1	224.0.0.251	MDNS	71	Standard query 0x0000 A sleep.local, "QM" question
38	3.966609	fe80::1a4f:289c:538…	ff02::fb	MDNS	91	Standard query 0x0000 A sleep.local, "QM" question
39	4.139473	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 A sleep, "QM" question
40	4.141813	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 A sleep, "QM" question
41	4.143865	172.30.32.1	224.0.0.251	MDNS	65	Standard query 0x0000 AAAA sleep, "QU" question
42	4.146014	fe80::1a4f:289c:538…	ff02::fb	MDNS	85	Standard query 0x0000 AAAA sleep, "QU" question
43	4.378287	172.30.32.1	224.0.0.251	MDNS	71	Standard query 0x0000 A sleep.local, "QM" question
44	4.379061	fe80::1a4f:289c:538…	ff02::fb	MDNS	91	Standard query 0x0000 A sleep.local, "QM" question
45	4.473234	172.30.32.1	172.30.47.255	NBNS	92	Name query NB SLEEP<00>
46	4.975649	172.30.32.1	224.0.0.251	MDNS	74	Standard query 0x0000 A infinity.local, "QM" question
47	4.979016	fe80::1a4f:289c:538…	ff02::fb	MDNS	94	Standard query 0x0000 A infinity.local, "QM" question
48	4.982115	172.30.32.1	224.0.0.251	MDNS	74	Standard query 0x0000 A infinity.local, "QM" question

Notes à prendre pendant la capture

Observation	À noter
Nombre de paquets capturés	315
Protocoles identifiés	MDNS, NBNS

1- **NBNS (NetBIOS Name Service)**.

C'est un protocole ancien utilisé par Windows pour résoudre des noms d'hôtes sur un réseau local, avant que DNS ne soit généralisé.



En (1) une machine avec pour adresse IPv4 « **172.30.32.1 » en un message BroadCast** (Message à tous les appareils du sous réseau) via **l'adresse BroadCast « 172.30.47.255** » pour obtenir des infos sur SLEEP, Pour faire simple il demande « Qui est SLEEP ? », si une machine avec ce nom est sur le sous reseaux elle repondra avec son adresse IP.

Il en est de même en (2) avec la machine INFINITY

NB : NBNS est généralement sur Port **UDP 137**

Tableau d'analyse des ports

Port	Protocole	État	Justification (Séquence observée)
80	TCP	**OUVERT**	SYN trame 45 → SYN-ACK trame 48
2222	TCP	**OUVERT**	SYN trame 52 → SYN-ACK trame 55
8080	TCP	**OUVERT**	SYN trame 60 → SYN-ACK trame 63
22	TCP	**FERMÉ**	SYN trame 120 → RST-ACK trame 121
443	TCP	**FERMÉ**	SYN trame 125 → RST-ACK trame 126
9999	TCP	**FILTRÉ**	SYN trame 200 → Pas de réponse
53	UDP	**FERMÉ**	UDP trame 305 → ICMP Port Unreachable tram
128	UDP	**FERMÉ**	UDP trame 305 → ICMP Port Unreachable tram
1900	SSDP	**FERME**	Absence de NOTIFY

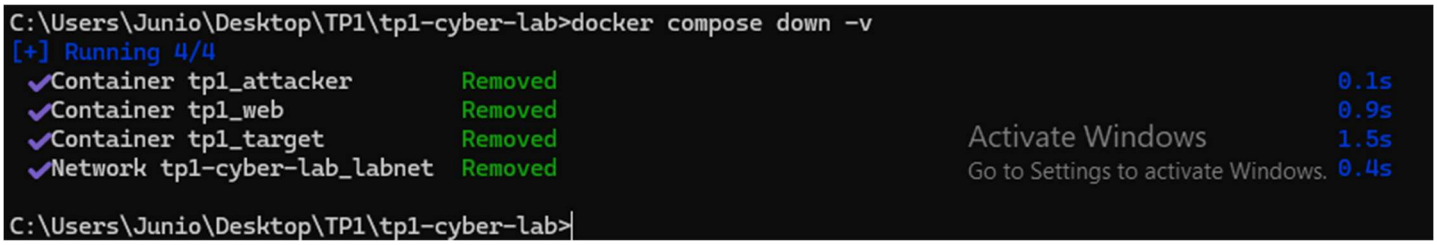
Nettoyage de l'environnement

Étape 1 : Arrêt des conteneurs

```bash
docker compose down
```

Étape 2 (optionnel) : Nettoyage complet

```bash
docker compose down -v
```



📋 Checklist de fin de TP

✅ Ce que nous devons avoir :

- **Capture baseline** (baseline.pcapng)
- **Capture scan** (scan.pcapng)
- **Analyses Wireshark** complétées
- **Tableau des ports** avec états déduits
- **Notes** sur les séquences TCP,UDP,... observées

🔒 Aspect éthique - RAPPEL IMPORTANT

- ✅ Toutes les manipulations sont restées **dans le lab fourni**
- ✅ **Aucun test** en dehors du périmètre autorisé
- ✅ Environnement maintenant **nettoyé**