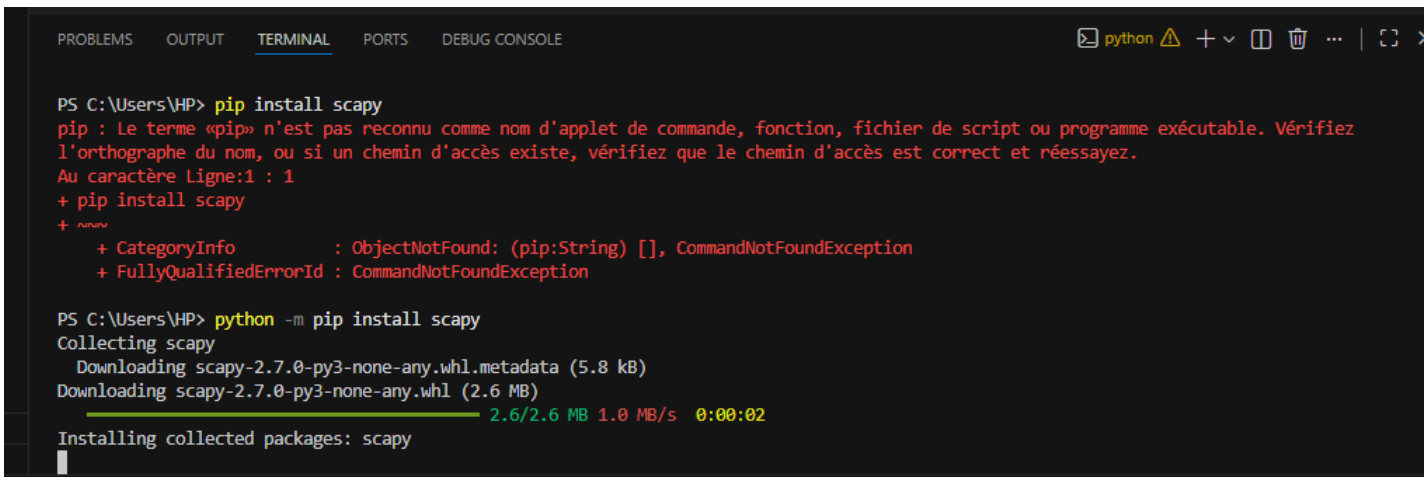
RAPPORT LSAAM

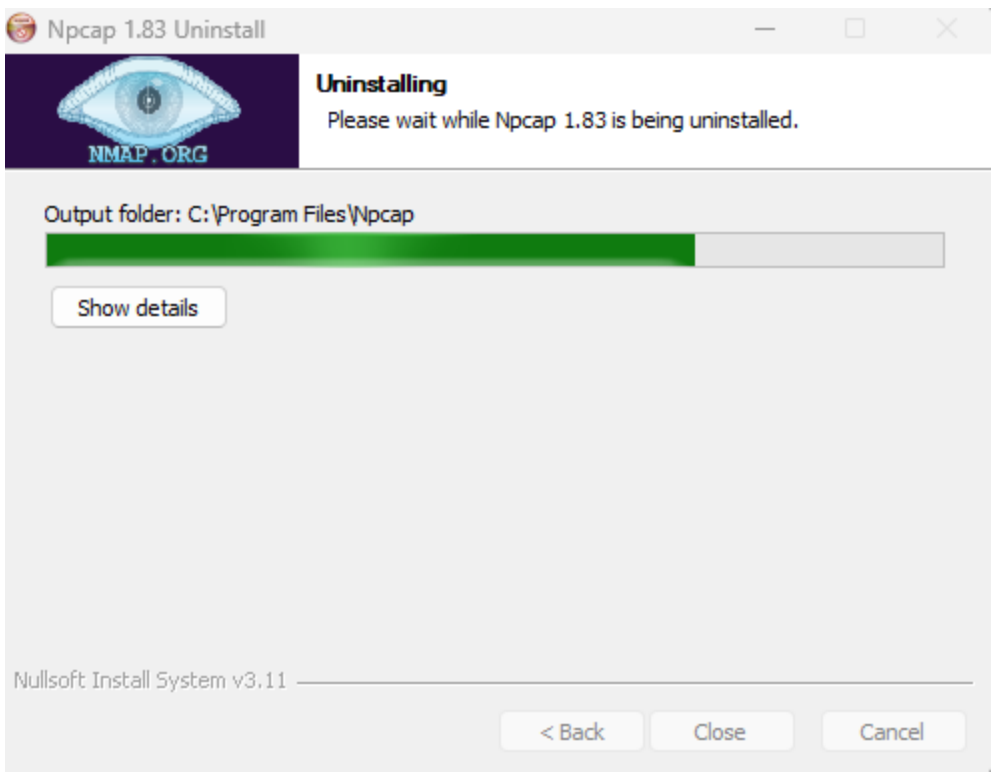
1 / Préparation de l'environnement : machines ciblés et environnement de coding :

1. ***INSTALLATION DE BIBLIOTHEQUES SCAPY*** **: (biblio responsable sur l'envoi des paquets)**



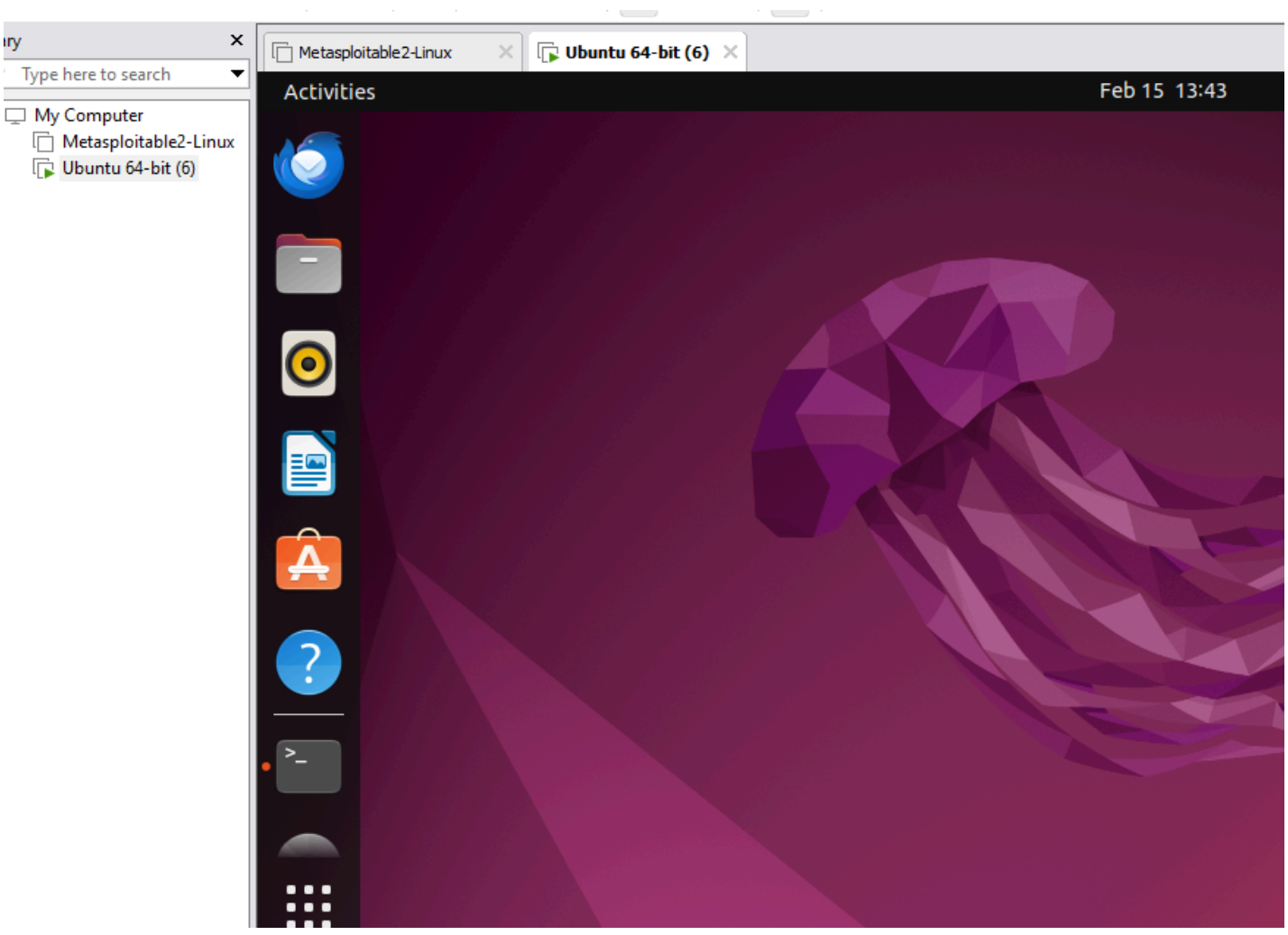
(pip seulement ne fonctionne pas , problème de chemin cad que le système ne connais pas ou python est installé (il faut modifier les variable d environnement de Path)

2. ***INSTALLATION DE Npcap : Sous Windows , Scapy ne peut pas envoyer de paquets sans un petit pilote appelé Npcap.***



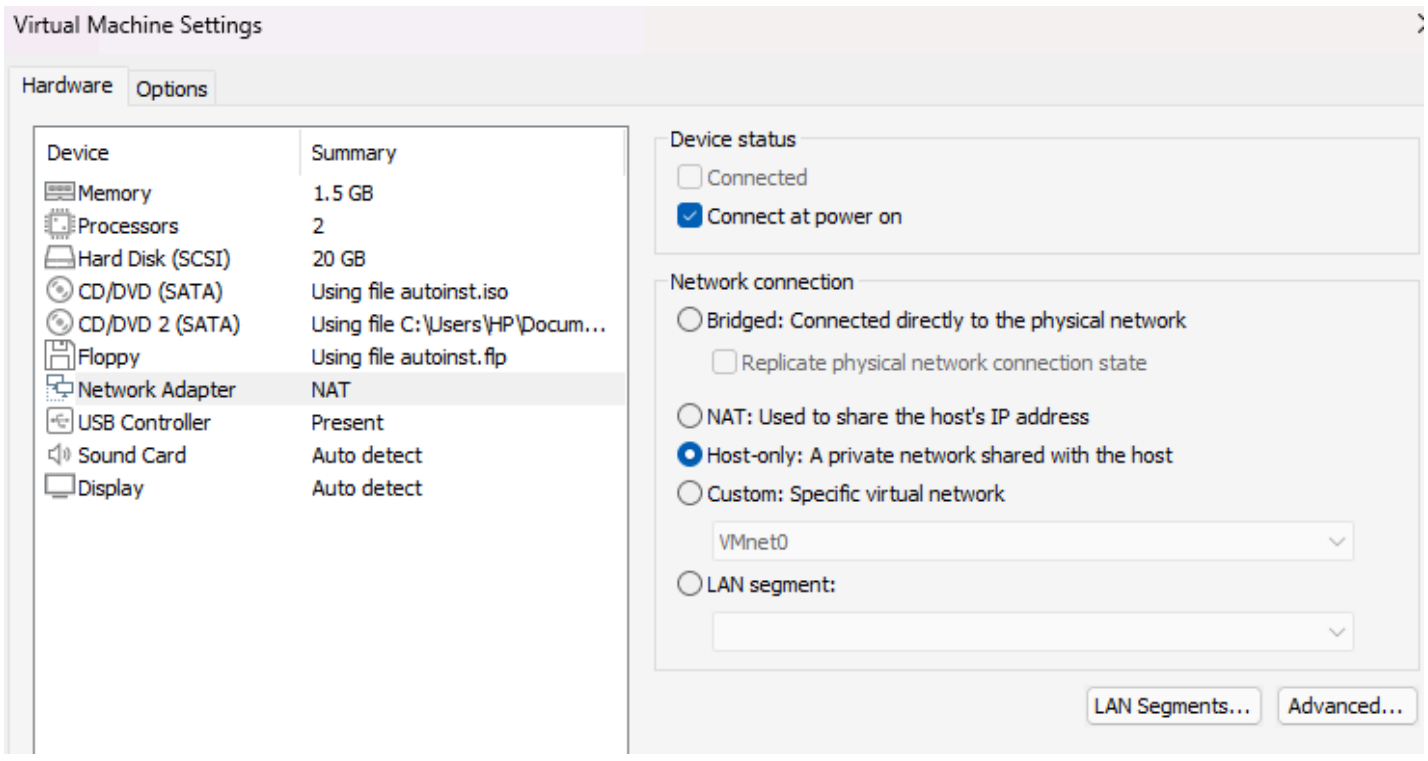
3. ***INSTALLATION DE METASPLOITABLE : (machine dédiée a etre vulnérabilisé par sa spécification de ports ouverts) la faire une mise a jour pour qu il soit compatible avec la version de workstation vm 17 et apres on fait le log-in par defaut msfadmin / msfadmin***

4. *Une deuxieme cible mais cette fois ci une machine vm avec le OS UBUNTU :*



5. ***NB:***

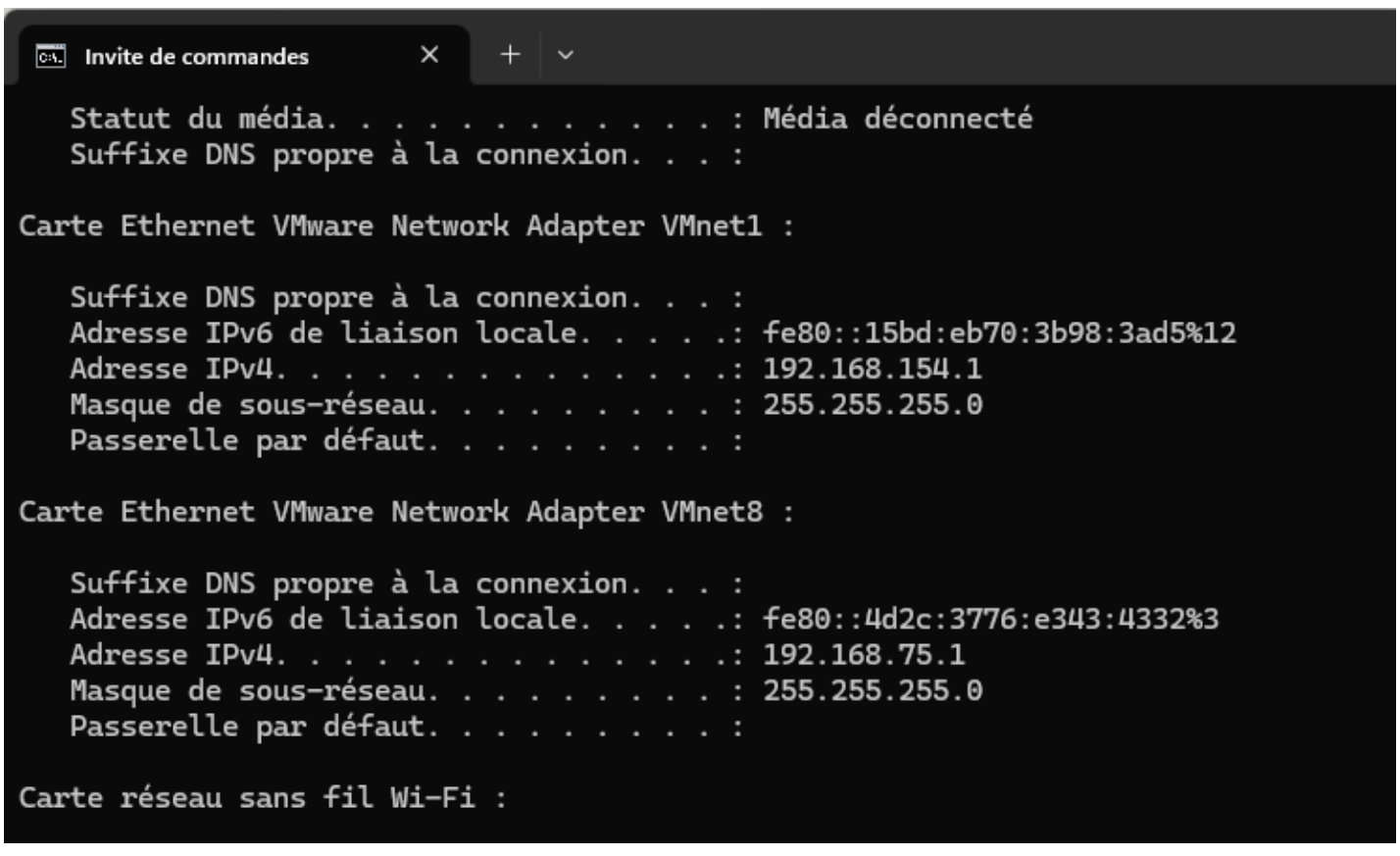
Pour avoir un réseau local et privé et sécurisé et en respectant les mesures légales (ethical) j utilise alors ma machine et les deux machines vm , on configure les deux machines en mode "HOST-ONLY "



6. *En vue d'avoir l'addresse de mon reseau prive (reseau entre ma machine et les deux machines vm) on utilise la commande* ***IPCONFIG*** :

c est exactement celle de ***VMNET 1: addr reseau*** *, c'est alors :* "*192.168.154.0/24*"

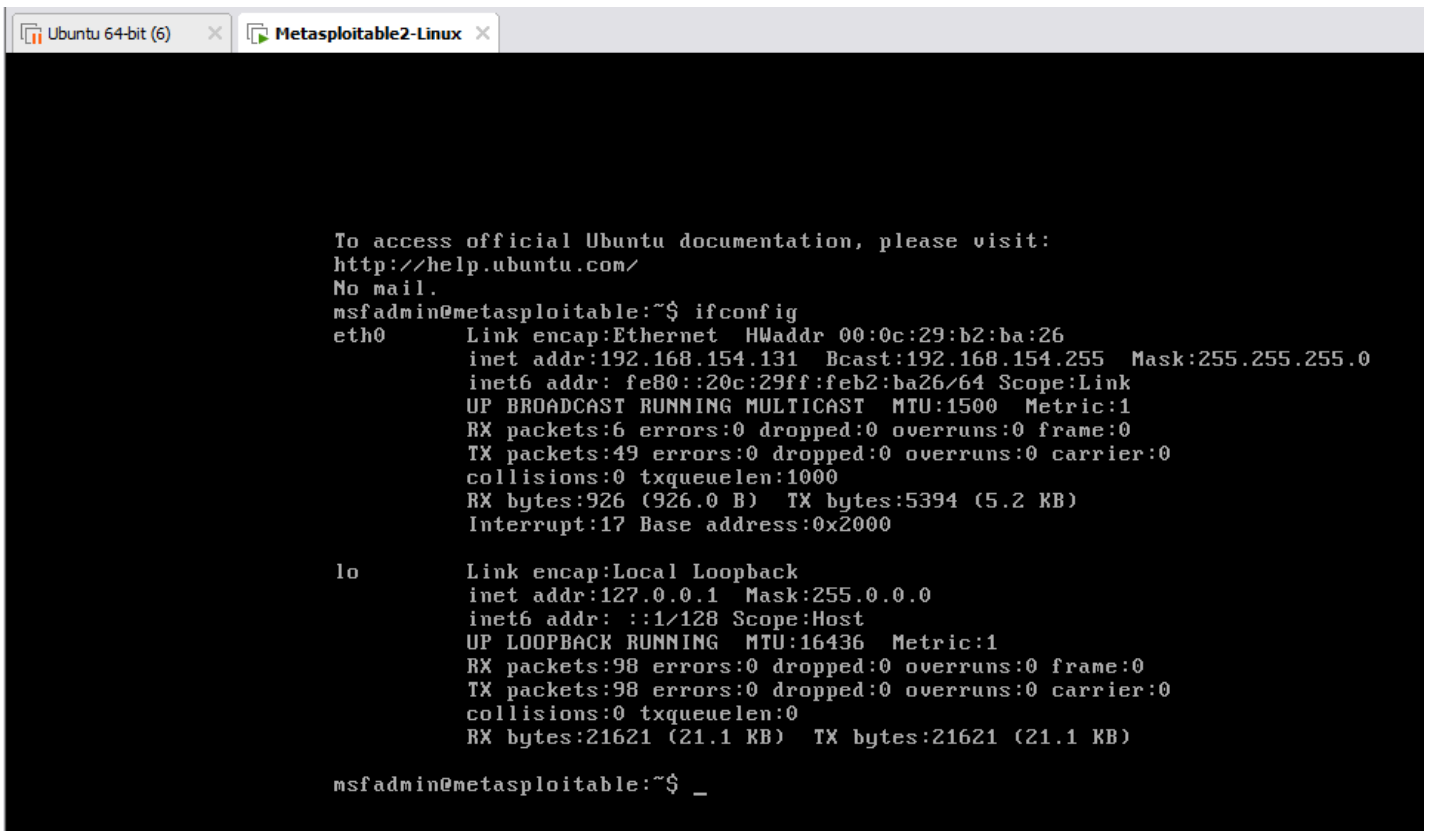
(VMnet1 *c est la carte réseau virtuelle des machines configurées en mode HOST-ONLY et* ***VMnet8*** *celles configurées en mode NAT)*



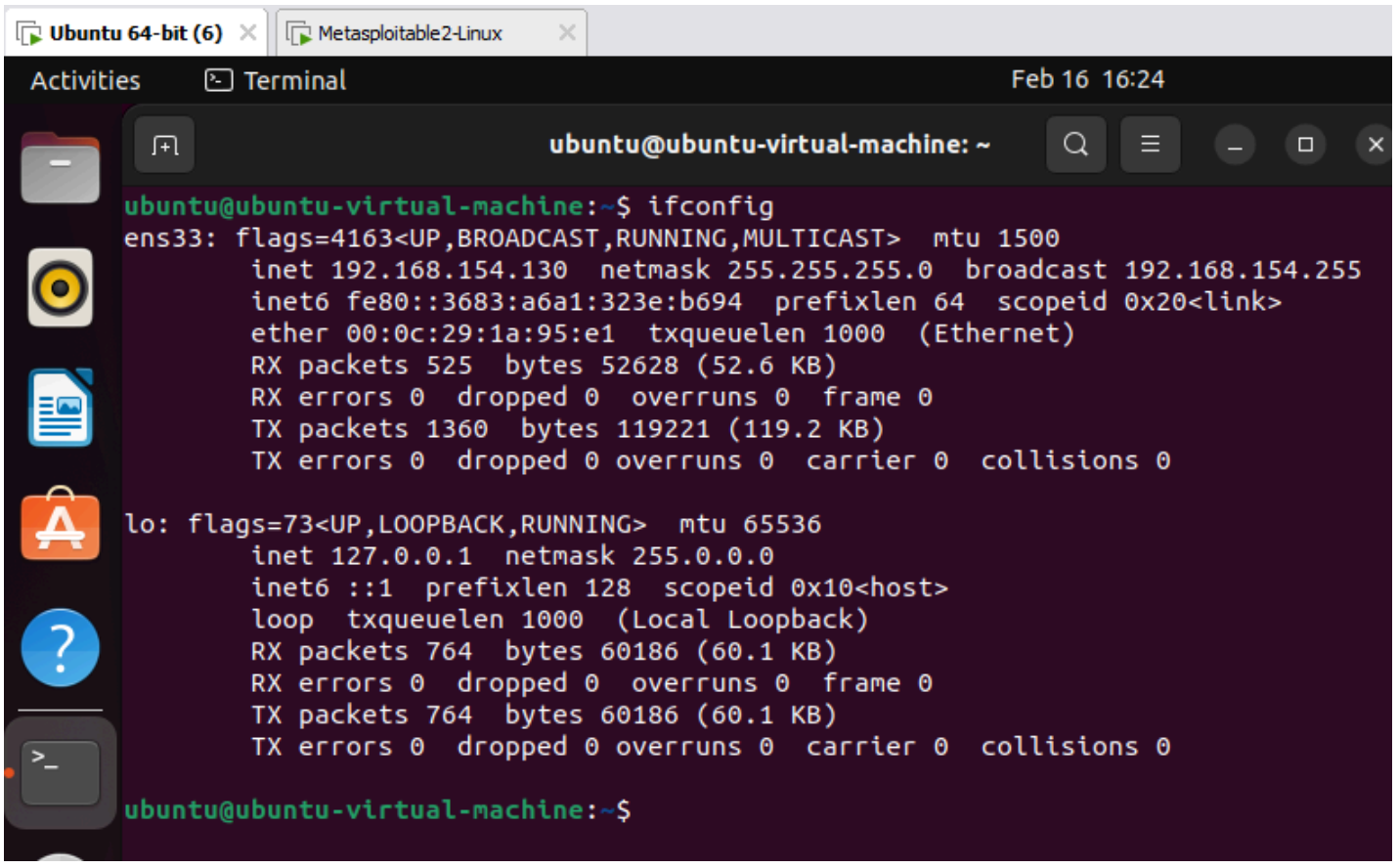
7. *Vérification de l'appartenenace des machines au meme reseau privé :*

METHODE 1 :

A l'aide de la commande ***IFCONFIG*** , on vérifie bien l'adresse IP de la première cible métasploitable ***est-ce qu il appartient a mon réseau prive ?***
c est exactement celle mentionnée dans la carte reseau eth0 : *inet addr 192.168.154.131*



de meme pour la deuxieme cible , est ce qu il appartient bien a mon addresse reseau prive



METHODE 2 :

OU bien on vérifie à l'aide d'un *PING* entre ma machine et la machine metasploitable ciblé à savoir si le canal est bien établi .
>>>>>PING bien établie

```
C:\Users\HP>ping 192.168.154.131

Envoi d'une requête 'Ping'  192.168.154.131 avec 32 octets de données :
Réponse de 192.168.154.131 : octets=32 temps=16 ms TTL=64
Réponse de 192.168.154.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.154.131 : octets=32 temps<1ms TTL=64
Réponse de 192.168.154.131 : octets=32 temps<1ms TTL=64

Statistiques Ping pour 192.168.154.131:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 0ms, Maximum = 16ms, Moyenne = 4ms

C:\Users\HP>
```

>>>>>>>>>> *ON REMARQUE bien que tous les machines appartiennent au meme reseau privé*

2/ Scripts en Phyton : Découverte du reseauvia SCAPY et scan des ports TCP via SOCKET :

1 >>> Etablir deux fonctions en phyton , une indispensable de la découverte du réseau et l'autre indispensable du scan des ports ouverts

main1.py

```python
def decouverte_reseau(ip_range):
    print("[1] Recherche des machines sur ",ip_range)
    requete_arp = ARP(pdst=ip_range)
    broadcast = Ether(dst="ff:ff:ff:ff:ff:ff")
    paquet_complet = broadcast / requete_arp

    reponses = srp(paquet_complet, timeout=2, verbose=False)[0]

    machines_trouvees = []

    print(len(reponses)," machine(s) détectée(s).")

    for envoye, recu in reponses:
        machine = {"ip": recu.psrc, "mac": recu.hwsrc}
        machines_trouvees.append(machine)
        print("* Trouvé :",recu.psrc, "| MAC : ",recu.hwsrc)

    return machines_trouvees
```

main1.py

C:\Users\HP\OneDrive\Desktop\PROJET LSAAM\main1.py

```python
    ports_communs = {
        80: "HTTP",
        443: "HTTPS",
        3306: "MySQL",
        25: "SMTP",
        53: "DNS",
    }
    for port, service in ports_communs.items():
        soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        soc.settimeout(0.5)
        resultat = soc.connect_ex((ip_cible, port))

        if resultat == 0:
            info = f" Port {port} {service} OUVERT"
            print(info ,"\n")
            resultats_machine.append(info)
            if port in [21, 23]:
                alerte = f"    !ALERTE : {service} n'est pas chiffré (Risque d'écoute) ! \n"
                print("  ",alerte)
                resultats_machine.append(alerte)
        soc.close()

    return resultats_machine
```

et apres on regroupe tout ceci dans un seul fichier <u>main.py</u>

main1.py

```python
from scapy.all import ARP, Ether, srp, conf
import socket
import json
import os
from datetime import datetime


conf.iface = "VMware Virtual Ethernet Adapter for VMnet1"
plage_reseau = "192.168.154.0/24"
nom_fichier = "inventaire.json"



def decouverte_reseau(ip_range):
    print("[1] Recherche des machines sur ",ip_range)
    requete_arp = ARP(pdst=ip_range)
    broadcast = Ether(dst="ff:ff:ff:ff:ff:ff")
    paquet_complet = broadcast / requete_arp

    reponses = srp(paquet_complet, timeout=2, verbose=False)[0]

    machines_trouvees = []

    print(len(reponses)," machine(s) détectée(s).")

    for envoye, recu in reponses:
        machine = {"ip": recu.psrc, "mac": recu.hwsrc}
        machines_trouvees.append(machine)
        print("* Trouvé :",recu.psrc, "| MAC : ",recu.hwsrc)

    return machines_trouvees

\ 0
```

```python
def audit_ports(ip_cible):
    print("\n[2] Analyse des services sur ",ip_cible)
    resultats_machine = []

    ports_communs = {
        21: "FTP",
        22: "SSH",
        23: "Telnet",
        80: "HTTP",
        443: "HTTPS",
        3306: "MySQL",
        25: "SMTP",
        53: "DNS",
    }
    for port, service in ports_communs.items():
        soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        soc.settimeout(0.5)
        resultat = soc.connect_ex((ip_cible, port))

        if resultat == 0:
            info = f" Port {port} {service} OUVERT"
            print(info ,"\n")
            resultats_machine.append(info)
            if port in [21, 23]:
                alerte = f"    !ALERTE : {service} n'est pas chiffré (Risque d'écoute) ! \n"
                print("  ",alerte)
                resultats_machine.append(alerte)
        soc.close()

    return resultats_machine
```

>>>>>>> Difficultés rencontrées :

lors du running de code global , j'avais toujours le message suivant !!!!

```
PS C:\Users\HP\OneDrive\Desktop\PROJET LSAAM> py main.py
--- Recherche des machines sur  192.168.154.0/24 ---
Machines trouvées :
Aucune machine trouvée. Vérifie ta configuration VMware !
PS C:\Users\HP\OneDrive\Desktop\PROJET LSAAM>
```

et tandis que la vérification des address ip manuellement montre bien qu'ils sont sur meme reseau mais aussi le passage du ping entre les machines qui force bien la configuration vm , alors c'etait probablement un problème des biblio et on dirait en global probleme sur mes scripts

C'est pourquoi j'ai essayé plusiuers tests :

>>Premier test : la désactivation du pare-feu windows

On dirait que le pare feu bloque le passage des requets ARP pour qu'ils circulent vers les cartes reseau des deux machines vm ,

mais ce n'était pas ca exactement .

>>Deusieme test : Quelle interface reseau utilise la biblio scapy ?

On vérifie alors quelle interface reseau utilise la biblio scapy , est-ce celle pas défaut (WIFI) ?

```
PROJET LSAAM > test_scapy.py
1
2
3   from scapy.all import show_interfaces, conf
4   print("--- Liste des interfaces détectées ---")
5   show_interfaces()
6
7   print(f"\nInterface par défaut actuelle : {conf.iface}")
8   from scapy.all import conf
9   print(conf.use_pcap)
```

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE                                  Python

--- Liste des interfaces détectées ---
Source   Index  Name                                    MAC                IPv4             IPv6
libpcap  1      Software Loopback Interface 1           00:00:00:00:00:00  127.0.0.1        ::1
libpcap  10     WAN Miniport (Network Monitor)
libpcap  11     Microsoft Wi-Fi Direct Virtual Adapter  82:30:49:84:2b:cd  169.254.110.133  fe80::86d3:9262:ea96:bce3
libpcap  12     VMware Virtual Ethernet Adapter for VMnet1  VMware:c0:00:01  192.168.154.1  fe80::15bd:eb70:3b98:3ad5
libpcap  13     WAN Miniport (IP)
libpcap  16     WAN Miniport (IPv6)
libpcap  17     Qualcomm Atheros QCA9377 Wireless Network_  LiteonTechno:84:2b:cd  192.168.8.146  fda2:f851:ca69:a800:5cd4:aab3:c414:6167
                                                                                                  fe80::16d4:560b:feff:1f2f
                                                                                                  fda2:f851:ca69:a800:af2f:ba77:5b55:308c
libpcap  18     Microsoft Wi-Fi Direct Virtual Adapter #2  92:30:49:84:2b:cd  169.254.6.192  fe80::4615:c6f3:7639:b82c
libpcap  19     Realtek(R) PCI(e) Ethernet Controller   CompalInform:44:e1:57  169.254.82.218  fe80::aea9:c91a:1e1e:2128
libpcap  3      VMware Virtual Ethernet Adapter for VMnet8  VMware:c0:00:08  192.168.75.1  fe80::4d2c:3776:e343:4332

Interface par défaut actuelle : \Device\NPF_{EAE8B657-E9F0-433C-ADDB-DE77FD15EDA0}
PS C:\Users\HP\OneDrive\Desktop\PROJET LSAAM> py main.py
```


On remarque bien que la biblio sacpy en fait utilise par défaut l'interface WIFI , donc les requetes ARP ne seront jamais envoyés vers mon reseau privé local qui utilise l ' interface Vmware ce qui explique la non découverte des machines sur mon reseau

donc le problème se référé a la non déclaration de l'interface reseau que la biblio SCAPY va utiliser et c es exactement celle qui a l index 12 intitulé "VMware Virtual Ethernet Adapter for VMnet1

>>Troisième test : Essayer par la fonction arping ()

c 'est une fonction naïve responsable sur l'envoie des requêtes ARP on lui passant la plage adresse

PROJET LSAAM > der_test.py

```python

from scapy.all import arping, conf

conf.iface = 12
print("Tentative avec la fonction native arping...")
arping("192.168.154.0/24")
```

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE

PS C:\Users\HP\OneDrive\Desktop\PROJET LSAAM> py der_test.py
Tentative avec la fonction native arping...
Begin emission
****
Finished sending 256 packets

Received 4 packets, got 4 answers, remaining 252 packets
src                manuf   psrc
00:50:56:c0:00:01  VMware  192.168.154.1
00:0c:29:1a:95:e1  VMware  192.168.154.130
00:0c:29:b2:ba:26  VMware  192.168.154.131
00:50:56:e4:6e:14  VMware  192.168.154.254
PS C:\Users\HP\OneDrive\Desktop\PROJET LSAAM>
```

ceci alors montre que tous se passe bien si jamais on utilise la fonction arping()avec l 'ajout de conf.iface ce qui montre alors que le problème etait bien le non ajout de l interface réseau grâce a conf.iface

>>>>>>>>>>>>>> La soltution du problème :

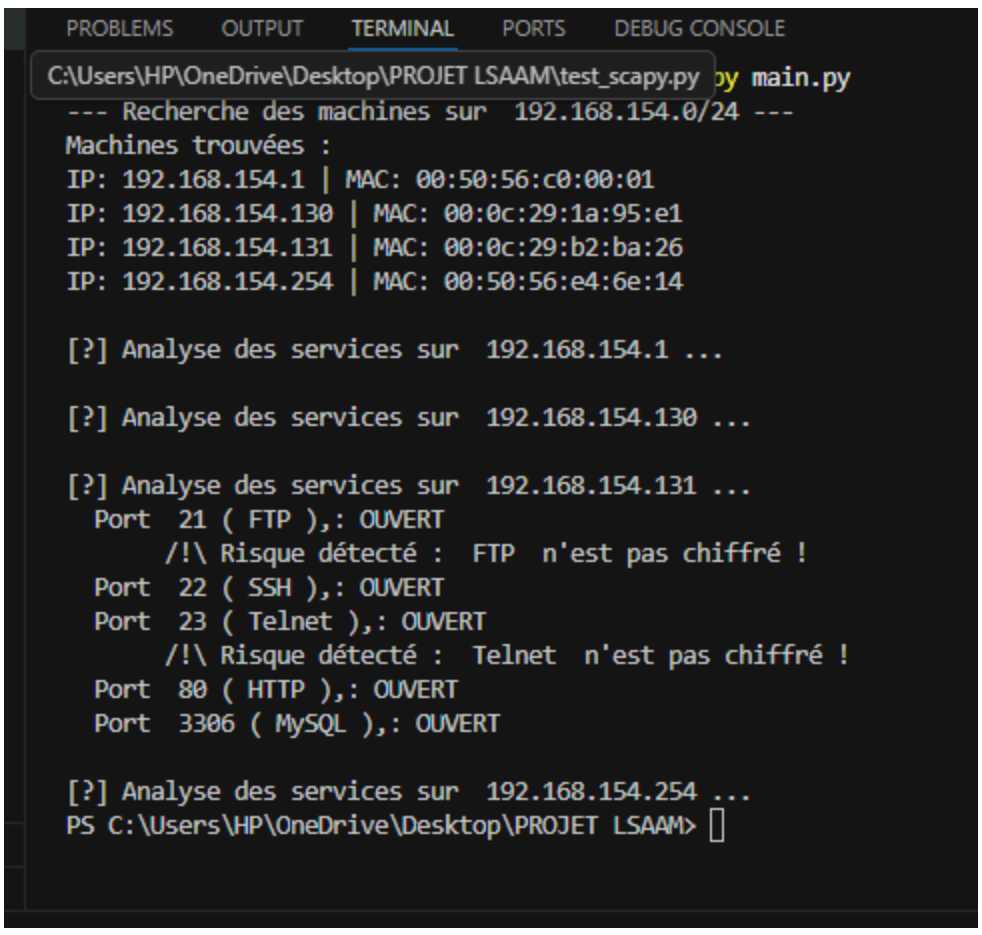
c'est exactement l'ajout de l'interface pour que notre biblio scapy sait ou adresser les paquets ARP , on ajout la variable conf.iface et on l'infecte le bon chemin

```python
from scapy.all import ARP, Ether, srp, conf
import socket
import json
import os
from datetime import datetime


conf.iface = "VMware Virtual Ethernet Adapter for VMnet1"
```

3/ Runing du code :

Aprés avoir regler les problémes rencontrés on essaye alors de faire run a nnotre code main.py



On remarque bien que on detecte tout les machines du reseau privé mais aussi des differentes ports ouverts des machines apres avoir les scanner , et spéciallement la machine metasploitable qui l addresse IP 169.168.175.131 , le scann a detecter effectivement plusieurs ports ouvets

donc on peut dire alors que ca marche bien

3/ Gestion de la mémoire de l'outil JSON (gestion d'Assets) et gener les fichiers d'audit texte

dans le but non seulement de faire un scan comme ca et partir cad on vérifie sans avoir stocker les résultats de notre scan c est pourquoi on essaye de stocker ces résultats a l'aide d'une base de données , et a chaque fois qu on cherche a les retourner il se fait de la consulter et comme ca on peut faire des comparaisons et des analyses si jamais on avait des attaques

la base de donnes utilise est celle intégrée dans le langage de programmation Python , c est bien les fichier avec l extension .JSON , sa caractéristique de garder les résultats seulement du dernier scan réalisé nous aidera après dans la génération des fichiers d'audit textes , cela en fait aidera beaucoup en terme de gestion administratif et plutôt gestion audit , car au lieu d'avoir des lignes énormes devant ses yeux il aura alors un fichier distinct qui porte le nom du jour et heure ou le scan a été réalise .

pour ce but , on crie une fonction sauvegarder_inventaire_json() qui va générer le fichier .JSON qui est notre base de donnés instantanée et au meme temps il va nous generer l'affectation a notre fichier d'audit qui portera en faite l extension .txt

```python
def sauvegarder_inventaire_json(ip, mac, ports):
    nom_fichier = "inventaire.json"

    if os.path.exists(nom_fichier) and os.path.getsize(nom_fichier) > 0:
        try:
            with open(nom_fichier, "r", encoding="utf-8") as f:
                inventaire = json.load(f)
        except json.JSONDecodeError:
            inventaire = {}
    else:
        inventaire = {}

    if mac not in inventaire:
        print("\n[!] NOUVEL ASSET DÉTECTÉ : ",ip)

    inventaire[mac] = {
        "ip": ip,
        "mac": mac,
        "derniere_vue": datetime.now().strftime("%d/%m/%Y %H:%M:%S"),
        "services": ports
    }

    with open(nom_fichier, "w", encoding="utf-8") as f:
        json.dump(inventaire, f, indent=4, ensure_ascii=False)



  print("================================================================================")
```

ain1.py

```python
  print( ================================================================================ )
  print("      LSAAM - LOCAL SECURITY AUDITOR & ASSETS MONITORING        ")
  print("================================================================================")


  liste_machines = decouverte_reseau(plage_reseau)

  if not liste_machines:
      print("\n Aucune machine n'a répondu. Fin de l'audit.")
  else:
      date_heure = datetime.now().strftime("%d-%m-%Y %H_%M_%S")
      nom_fichier_unique =f"rapport_audit_{date_heure}.txt"

  with open(nom_fichier_unique, "w", encoding="utf-8") as f:
      f.write("RAPPORT D'AUDIT DU "+date_heure+"\n")
      f.write("="*50 + "\n\n")

      for machine in liste_machines:
              liste_ports_ouverts = audit_ports(machine["ip"])

              sauvegarder_inventaire_json(machine["ip"], machine["mac"], liste_ports_ouverts)

              f.write("CIBLE : " + machine['ip'] + "(MAC: "+machine['mac']+") \n")
              if  liste_ports_ouverts:
                  for ligne in liste_ports_ouverts:
                      f.write(ligne+"\n")
              else:
                  f.write("  Aucun service critique détecté.\n")
              f.write("-" * 30 + "n\n")
```

```
            sauvegarder_inventaire_json(machine["ip"], machine["mac"], liste_ports_ouverts)

            f.write("CIBLE : " + machine['ip'] + "(MAC: "+machine['mac']+") \n")
            if  liste_ports_ouverts:
                for ligne in liste_ports_ouverts:
                    f.write(ligne+"\n")
            else:
                f.write("  Aucun service critique détecté.\n")
            f.write("-" * 30 + "n\n")

    print("\n" + "="*50)
    print(" AUDIT TERMINÉ !")
    print("   -> Mémoire mise à jour : ",nom_fichier)
    print("   -> Rapport généré : ",nom_fichier_unique)
    print("="*50)
```

>>>>>>>>>>>>>>> Exemple du fichier .JSON :

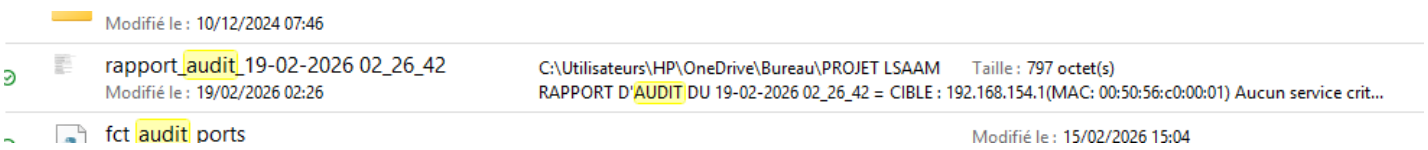
inventaire
sai : personnel\Bureau\PROJET LSAAM 23/02/2026 12:40

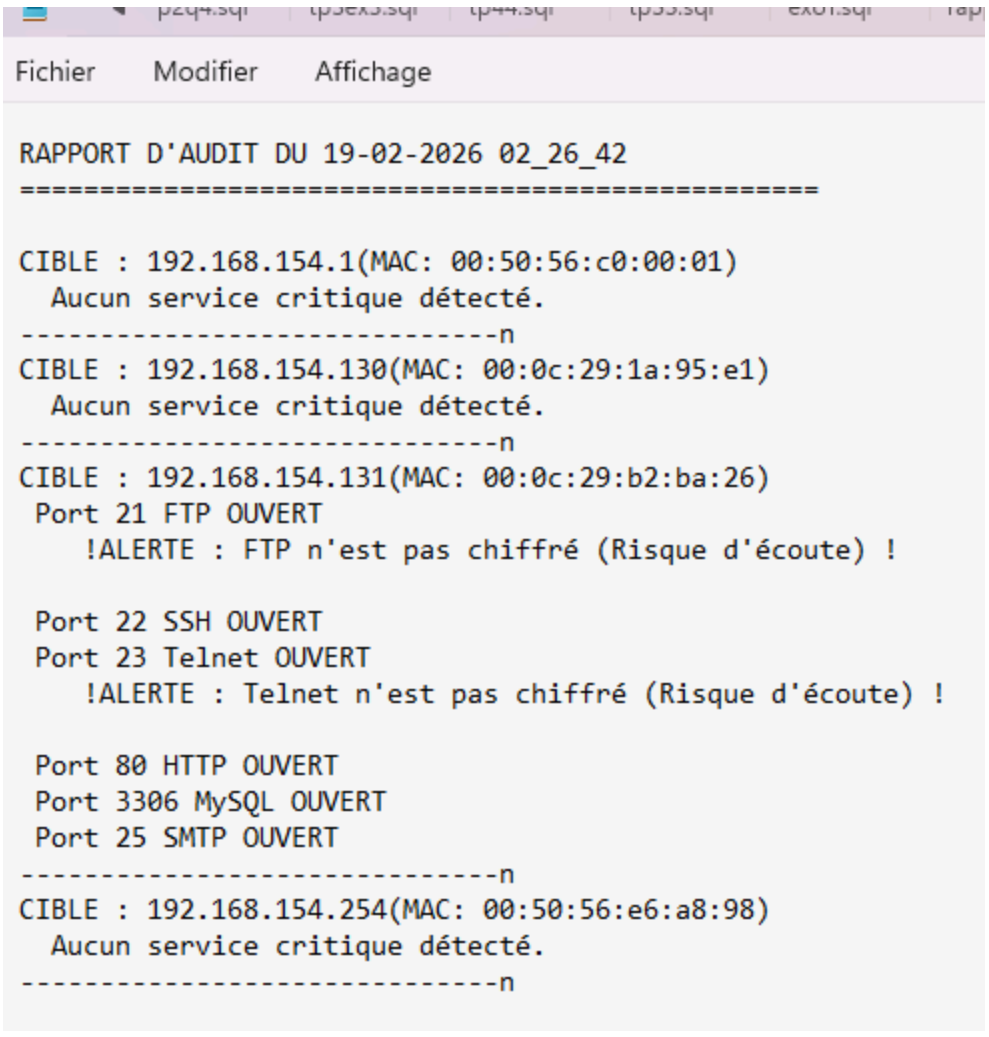
main1.py ● main.py inventaire.json ×

inventaire.json

```
{
    "00:0c:29:1a:95:e1": {
        "ip": "192.168.154.130",
        "mac": "00:0c:29:1a:95:e1",
        "derniere_vue": "19/02/2026 02:26:51",
        "services": []
    },
    "00:0c:29:b2:ba:26": {
        "ip": "192.168.154.131",
        "mac": "00:0c:29:b2:ba:26",
        "derniere_vue": "19/02/2026 02:26:52",
        "services": [
            " Port 21 FTP OUVERT",
            "    !ALERTE : FTP n'est pas chiffré (Risque d'écoute) ! \n",
            " Port 22 SSH OUVERT",
            " Port 23 Telnet OUVERT",
            "    !ALERTE : Telnet n'est pas chiffré (Risque d'écoute) ! \n",
            " Port 80 HTTP OUVERT",
            " Port 3306 MySQL OUVERT",
            " Port 25 SMTP OUVERT"
        ]
    },
    "00:50:56:f9:10:50": {
        "ip": "192.168.154.254",
        "mac": "00:50:56:f9:10:50",
        "derniere_vue": "17/02/2026 23:35:20",
        "services": []
    },
    "00:50:56:f4:ce:64": {
        "ip": "192.168.154.254",
        "mac": "00:50:56:f4:ce:64",
        "derniere_vue": "19/02/2026 00:09:17",
```

>>>>>>>>>>>>>>> Exemple d’un fichier d’audit .txt :





Fichier Modifier Affichage

```
RAPPORT D'AUDIT DU 19-02-2026 02_26_42
================================================

CIBLE : 192.168.154.1(MAC: 00:50:56:c0:00:01)
  Aucun service critique détecté.
------------------------------n
CIBLE : 192.168.154.130(MAC: 00:0c:29:1a:95:e1)
  Aucun service critique détecté.
------------------------------n
CIBLE : 192.168.154.131(MAC: 00:0c:29:b2:ba:26)
 Port 21 FTP OUVERT
    !ALERTE : FTP n'est pas chiffré (Risque d'écoute) !

 Port 22 SSH OUVERT
 Port 23 Telnet OUVERT
    !ALERTE : Telnet n'est pas chiffré (Risque d'écoute) !
 Port 80 HTTP OUVERT
 Port 3306 MySQL OUVERT
 Port 25 SMTP OUVERT
------------------------------n
CIBLE : 192.168.154.254(MAC: 00:50:56:e6:a8:98)
  Aucun service critique détecté.
------------------------------n
```

Ce travail permettra au personne chargé d’audit de bien organiser son travail mais aussi un enregistrement ciblé .

>> Réalisé par :

El BAHRI Saida