Ghost in the heap

First we check for flags in in strings using grep for the flag format

```
┌──(kali㉿kali)-[~]
└─$ strings ghost_in_the_heap | grep 'SGCTF'
SGCTF{memory_never_lies}
```

We get a flag SGCTF{memory_never_lies} but we find out it is a fake flag

So we load the binary file into gdb

```
┌──(kali㉿kali)-[~]
└─$ gdb ./ghost_in_the_heap
GNU gdb (Debian 16.3-1) 16.3
Copyright (C) 2024 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/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./ghost_in_the_heap...
(No debugging symbols found in ./ghost_in_the_heap)
(gdb)
```

We break at main and run

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./ghost_in_the_heap...
(No debugging symbols found in ./ghost_in_the_heap)
(gdb) break main
Breakpoint 1 at 0x118d
(gdb) run
Starting program: /home/kali/ghost_in_the_heap
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x000055555555518d in main ()
(gdb)
```

We continue to see the contents of the program

```
Breakpoint 1, 0×000055555555518d in main ()
(gdb) continue
Continuing.
Welcome.
A whisper points here: 0×5555555592f0
Some echoes hide; seek 0×30 steps beyond the whispers.
[Inferior 1 (process 7484) exited normally]
(gdb)
```

We see a whisper and an offset of 0x30 saying that the flag is 0x30 after the whisper

So we restart gdb again and and break at exit also so the program doesn't exit

```
Breakpoint 1, 0×000055555555518d in main ()
(gdb) break exit
Breakpoint 2 at 0×7ffff7df1340
(gdb) continue
Continuing.
Welcome.
A whisper points here: 0×5555555592f0
Some echoes hide; seek 0×30 steps beyond the whispers.

Breakpoint 2, 0×00007ffff7df1340 in exit () from /lib/x86_64-linux-gnu/libc.so.6
(gdb)
```

So the whisper points to the address (**0x5555555592f0**) and the hint says seek 0x30 steps beyond the whispers saying that the flag is located at an offset of 0x30 after the whisper

```
Breakpoint 2, 0×00007ffff7df1340 in exit () from /lib/x86_64-linux-gnu/libc.so.6
(gdb) x/s 0×5555555592f0 + 0×30
0×555555559320: "SGCTF{memory_never_lies}"
(gdb)
```

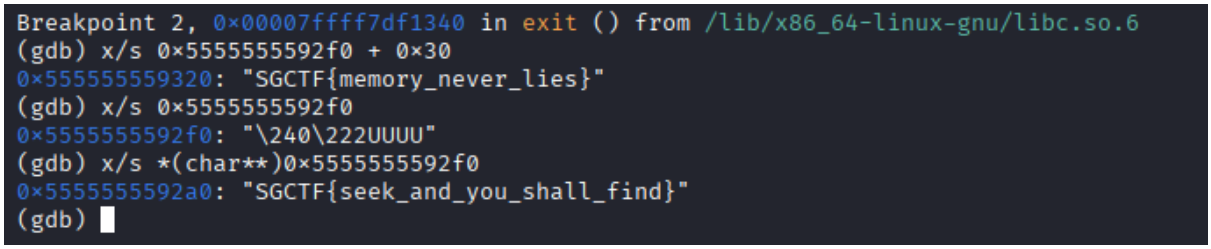
Which shows that it is the same fake flag

From the overview we saw that 'Some whispers are hidden, some lies are told'

Here it may hint that the 0x30 offset is a lie and since we didn't consider the whisper it could suggest that the whisper itself might be the flag

```
Breakpoint 2, 0×00007ffff7df1340 in exit () from /lib/x86_64-linux-gnu/libc.so.6
(gdb) x/s 0×5555555592f0 + 0×30
0×555555559320: "SGCTF{memory_never_lies}"
(gdb) x/s 0×5555555592f0
0×5555555592f0: "\240\222UUUU"
(gdb)
```

The whisper holds some garbage value since the whisper is a pointer we first dereference the addres and read whatever might be the flag



Which gives us our flag **SGCTF{seek_and_you_shall_find}**