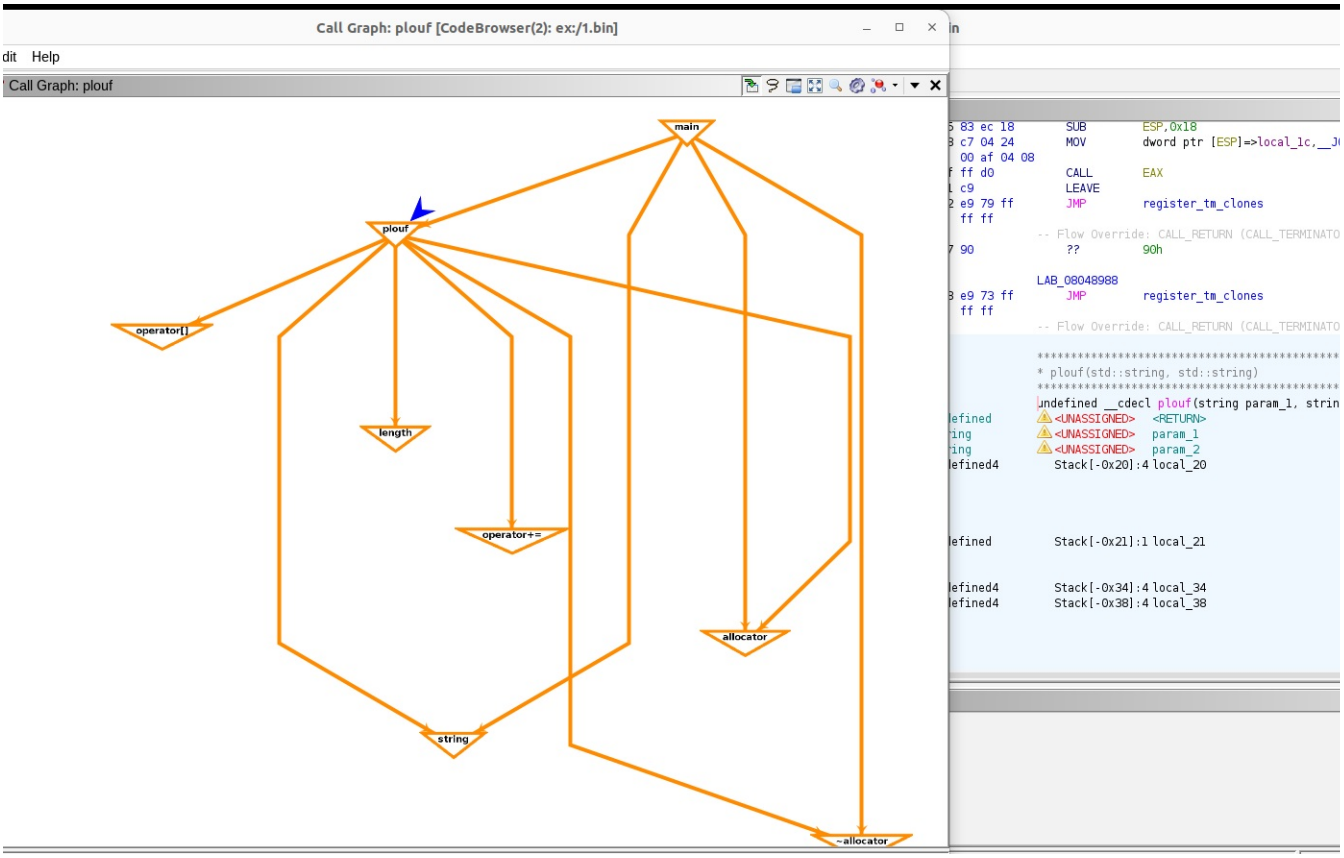
1.bin password

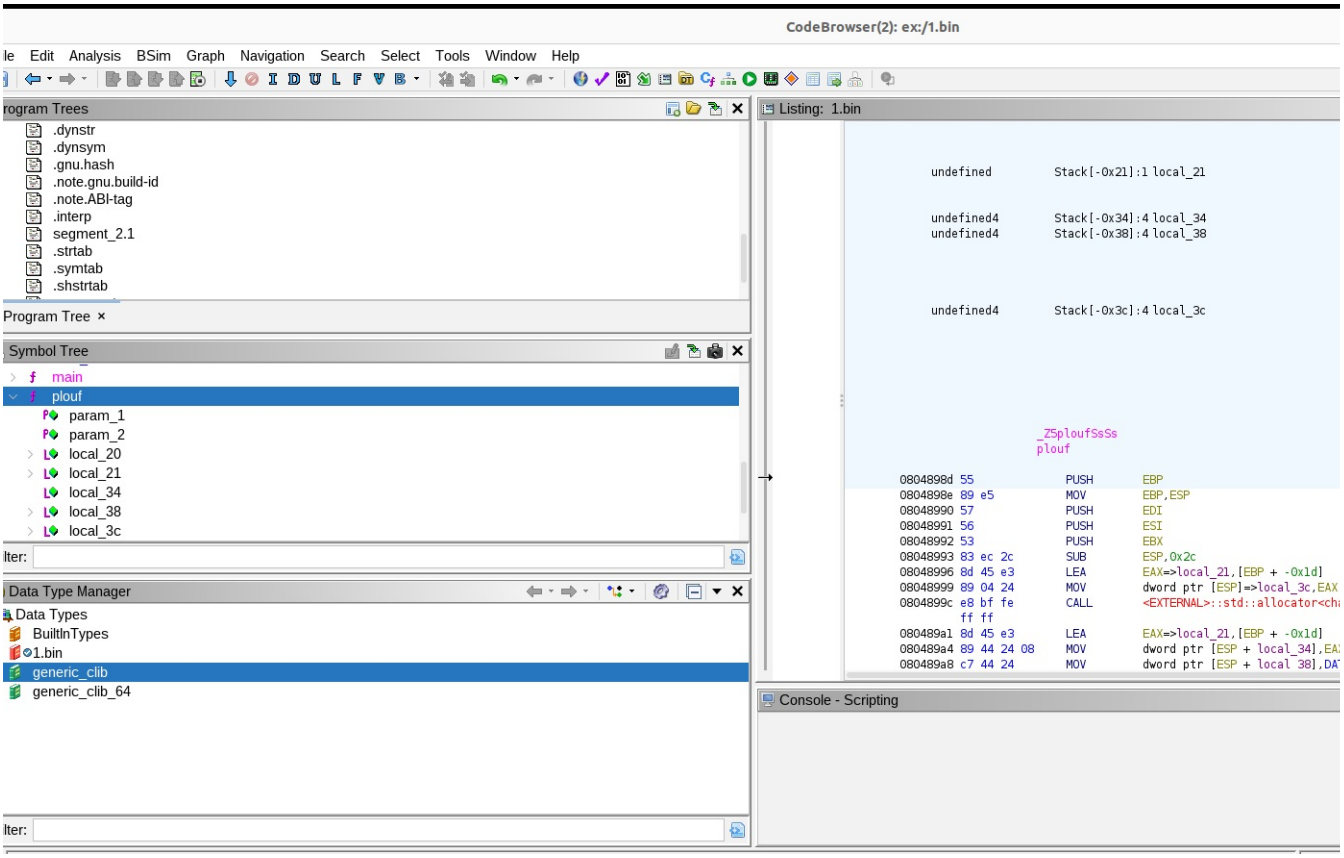
(gdb) break *0x08048b51

gdb) x/4wx $eax 0xffffcfd4: 0x08050ccc 0x08050bdc 0x08050bbc 0xffffd000 (gdb) 0xffffcfe4: 0xf7d3f000 0xf7ffd020 0xf7b36519 0xf7ef4a70 (gdb) set $p = *(unsigned int)*$eax (gdb) x/s $p 0x8050ccc: "Here_you_have_to_understand_a_little_C++_stuffs" (gdb)

I chose 0x08048b51 because it's the instruction immediately after the call to plouf in the main() function. Here's why that matters:

Why that address is key When you call a function in x86 (32-bit), the return value is placed in the EAX register. But you can only inspect EAX after the function has returned. So we need to break at the instruction right after the call plouf.





Post-call address for plouf from the provided image

From the XREF line in your image: `_Z5ploufSsSs XREF[6]: Entry Point(*), main:08048b4c(c), 08048e94, 08048f30(*), 08049059(*), 0804905e(*)` - **Key detail:** `main:08048b4c(c)` indicates a call to plouf at address `0x08048b4c` inside main.

On 32-bit x86, a near call (`call rel32`) is 5 bytes long, so the instruction immediately after the call (the post-call "return address" you break on to read EAX) is: `0x08048b4c + 5 = 0x08048b51`

That's why breaking at `0x08048b51` lets you inspect the return value from plouf in EAX.