GDB baby step 3

Medium | Reverse Engineering | picoGym Exclusive | x86_64

AUTHOR: LT 'SYREAL' JONES

Description

Now for something a little different. `0x2262c96b` is loaded into memory in the `main` function. Examine byte-wise the memory that the constant is loaded in by using the GDB command `x/4xb addr`.
The flag is the four bytes as they are stored in memory. If you find the bytes `0x11 0x22 0x33 0x44` in the memory location, your flag would be: `picoCTF{0x11223344}`.
Debug this.

Hints

1 2 3 4 5

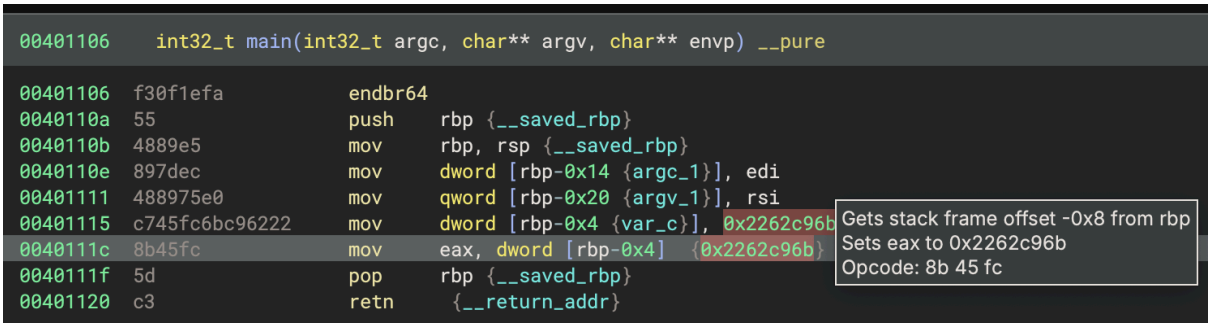
Solusi dari write up lain :

Download binary and run gdb with 'gdb debugger0_c'.
Dissasemble main() function with 'disassemble main' and you will get the dump:

```
0x0000000000401106 <+0>:     endbr64
0x000000000040110a <+4>:     push   %rbp
0x000000000040110b <+5>:     mov    %rsp,%rbp
0x000000000040110e <+8>:     mov    %edi,-0x14(%rbp)
0x0000000000401111 <+11>:    mov    %rsi,-0x20(%rbp)
0x0000000000401115 <+15>:    movl   $0x2262c96b,-0x4(%rbp)
0x000000000040111c <+22>:    mov    -0x4(%rbp),%eax
0x000000000040111f <+25>:    pop    %rbp
0x0000000000401120 <+26>:    ret
```

Our memory load is at <+15>, meaning we have to set a breakpoint on next instruction:
b *(main+22)
and run the program with 'run'. Now we can inspect the address of $rbp-0x4:
x/4xb $rbp-0x4
which outputs our bytes as they are stored in memory:
0x7fffffffddec: 0x6b 0xc9 0x62 0x22
We got our flag:
picoCTF{0x6bc96222}

Solusiku :
Buka dengan binary ninja dan langsung pergi ke fungsi main.



Disini kita bisa melihat bahwa fungsi main mereturn nilai rbp yang menyimpan 0x2262c96b. Karena dalam hints menyinggung endianness, jadi aku coba balik nilainya menjadi 0x6bc96222 sehingga flagnya adalah picoCTF{0x6bc96222}