Muthra

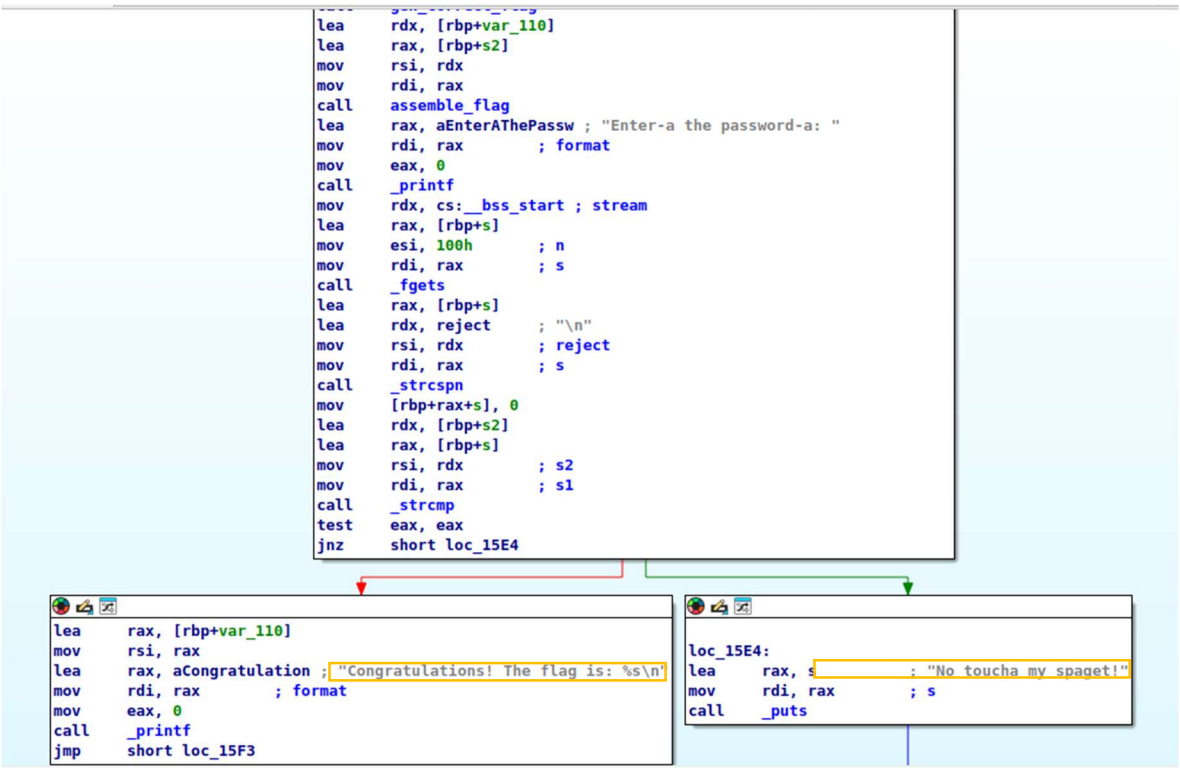
Patriot CTF 2024, Challenge: **" Revioli, Revioli, give me the formeoli",**Category**:rev**

CTF Date :09/20/24-09/22/2024

Step 1: Disassembly with IDA Free

I opened the executable file in **IDA Free** to disassemble the binary and analyze its logic. The program has a password check where it compares the entered password with a hardcoded string using strcmp.

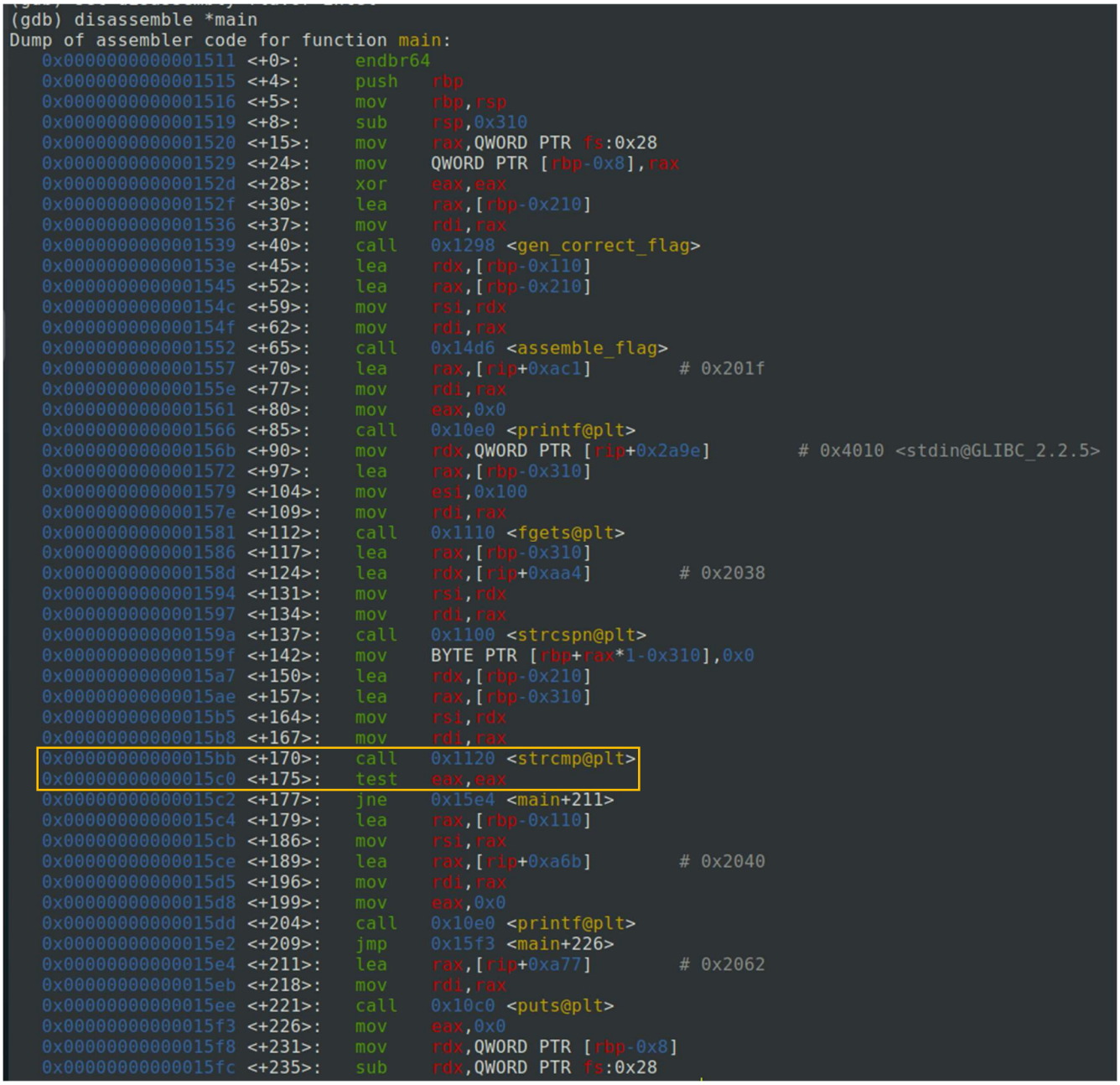
- If the strings match, the program provides the flag.
- If the strings don't match, it prints "No toucha my spaget!" and exits.



Step 2: Dynamic Analysis with GDB

I proceeded to debug the program using **GDB** to manipulate its behavior and bypass the password check.

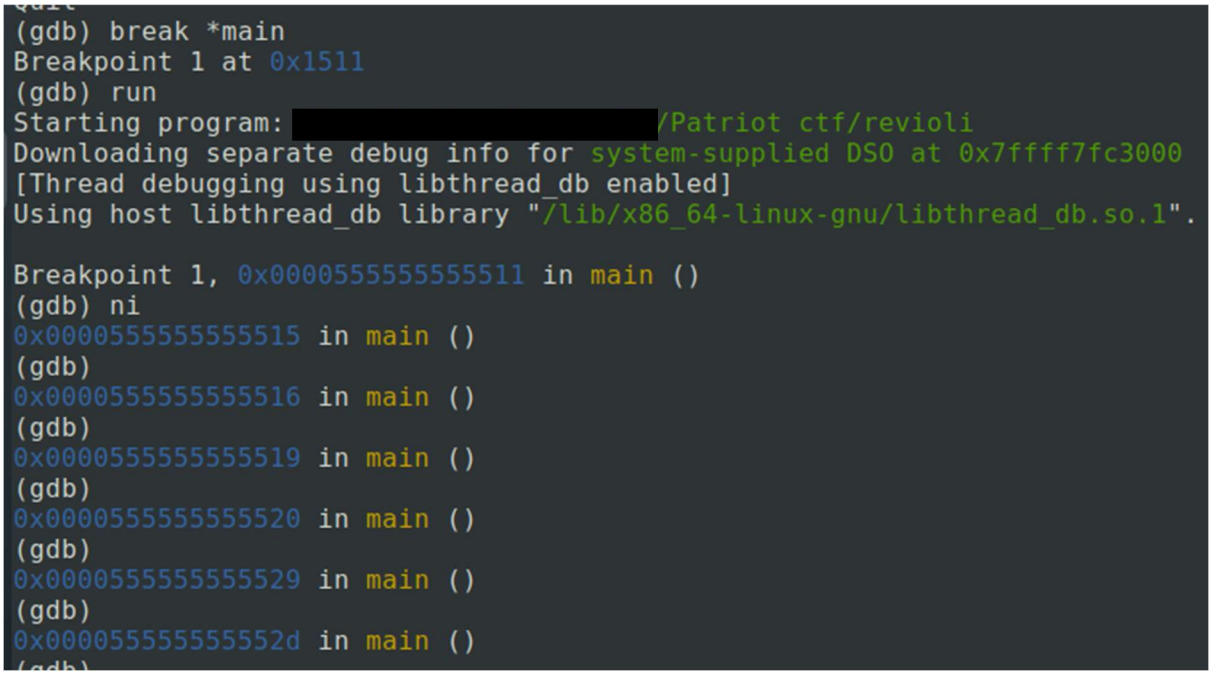
- First, I disassembled the main function by running:
- Next, I identified the strcmp call, which compares the entered password with the correct one.

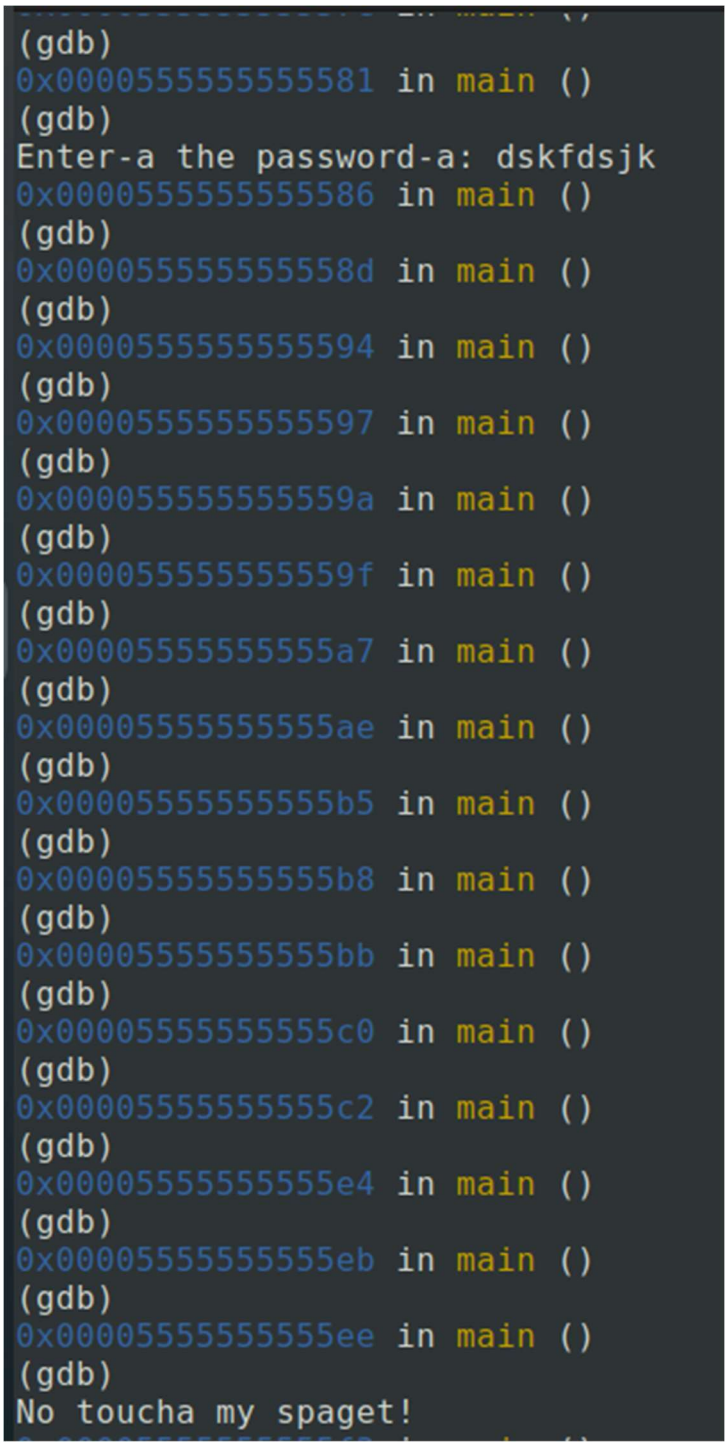


Step 3: Breakpoint at main

I set a breakpoint at the start of the main function to step through the program and understand its flow:

I then ran the program and used the ni (next instruction) command to step through it line by line.



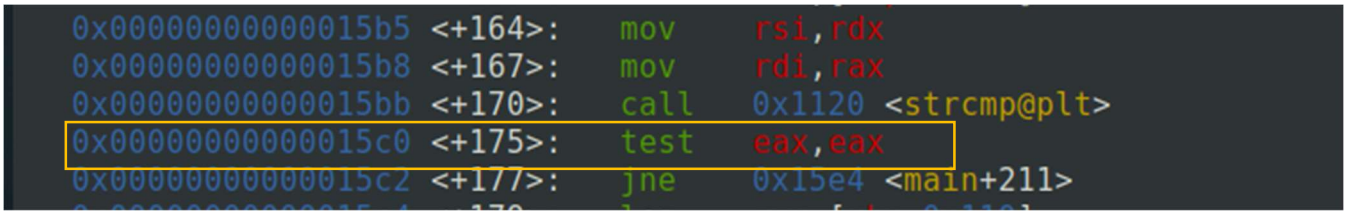

(gdb)
0x0000555555555581 in main ()
(gdb)
Enter-a the password-a: dskfdsjk
0x0000555555555586 in main ()
(gdb)
0x000055555555558d in main ()
(gdb)
0x0000555555555594 in main ()
(gdb)
0x0000555555555597 in main ()
(gdb)
0x000055555555559a in main ()
(gdb)
0x000055555555559f in main ()
(gdb)
0x00005555555555a7 in main ()
(gdb)
0x00005555555555ae in main ()
(gdb)
0x00005555555555b5 in main ()
(gdb)
0x00005555555555b8 in main ()
(gdb)
0x00005555555555bb in main ()
(gdb)
0x00005555555555c0 in main ()
(gdb)
0x00005555555555c2 in main ()
(gdb)
0x00005555555555e4 in main ()
(gdb)
0x00005555555555eb in main ()
(gdb)
0x00005555555555ee in main ()
(gdb)
No toucha my spaget!

Step 4: Manipulating the eax Register

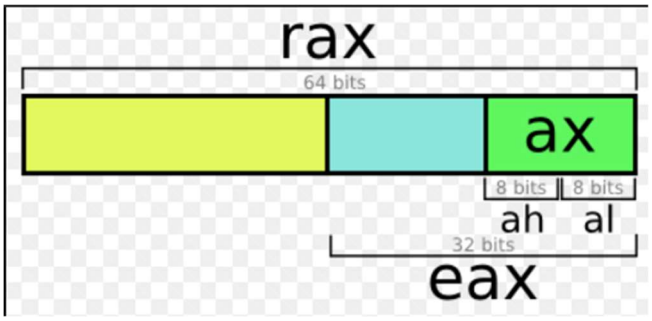
After the strcmp call, I found a test eax, eax instruction. This is where the result of the comparison (strcmp) is checked. If the eax register contains 0 (indicating the strings matched), the program jumps to the section that prints the flag.

So , I set a breakpoint at this instruction.

Then, I continued execution until the breakpoint was hit. At this point, I modified the value of the eax register to 0, forcing the program to believe that the password was correct.



Note : Eax is a part of rax



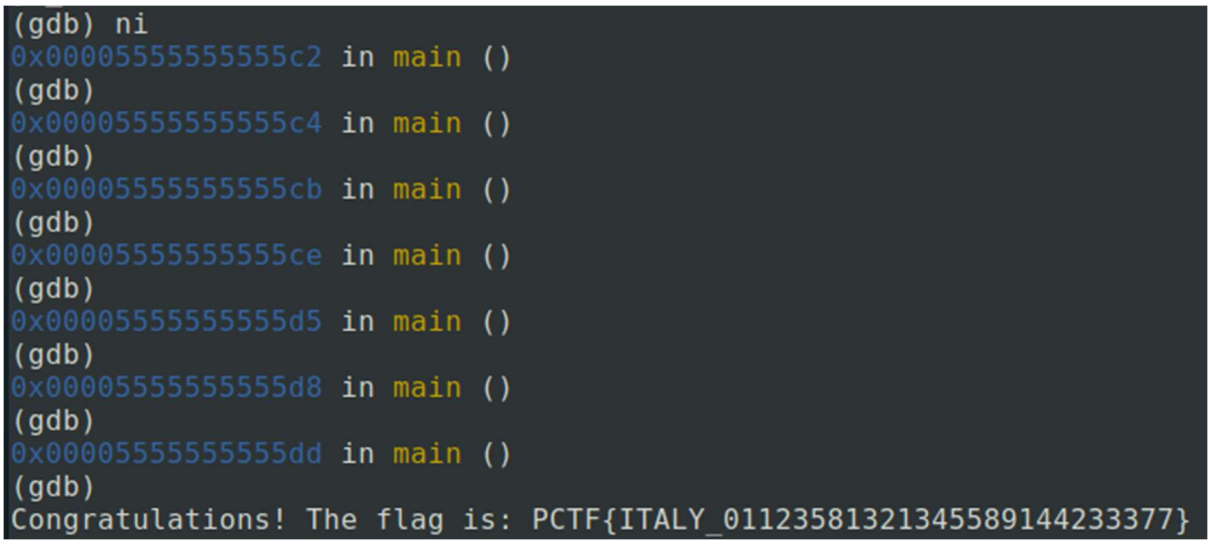
Note: I used info register command to find the register values

```
(gdb) break *0x00005555555555c0
Breakpoint 2 at 0x5555555555c0
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: [redacted]/Patriot ctf/revioli
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x0000555555555511 in main ()
(gdb) continue
Continuing.
Enter-a the password-a: dfdsf

Breakpoint 2, 0x00005555555555c0 in main ()
(gdb) info register
rax            0x1b                27
rbx            0x7fffffffdec8      140737488346824
rcx            0x49                73
rdx            0x7fffffffdb90      140737488346000
rsi            0x7fffffffdb90      140737488346000
rdi            0x7fffffffda90      140737488345744
rbp            0x7fffffffdda0      0x7fffffffdda0
rsp            0x7fffffffda90      0x7fffffffda90
r8             0x5555555596b6      93824992253622
```

```
(gdb) set $eax=0
(gdb) info register
rax            0x0                 0
rbx            0x7fffffffdec8      140737488346824
rcx            0x49                73
rdx            0x7fffffffdb90      140737488346000
rsi            0x7fffffffdb90      140737488346000
rdi            0x7fffffffda90      140737488345744
rbp            0x7fffffffdda0      0x7fffffffdda0
rsp            0x7fffffffda90      0x7fffffffda90
r8             0x5555555596b6      93824992253622
```

Step 5: Continuing Execution to Retrieve the Flag

After setting eax to 0, I used the ni command to continue stepping through the program, which now jumped to the flag-printing section. As a result, the program printed the flag, successfully bypassing the password check.



Conclusion

By analyzing the logic in IDA and using GDB to manipulate the execution flow, I was able to bypass the password check and retrieve the flag without knowing the correct password.