Writeup(2020.7.1-2020.7.7)

原创

BIAUTUMN 于 2020-07-07 16:11:41 发布 135 收藏

文章标签： CTF 逆向 C++ 逆向工程

BUUCTF不一样的flag

查壳，无壳，32位
打开ida静态分析，main函数f5查看伪代码

```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v3; // [esp+17h] [ebp-35h]
  int v4; // [esp+30h] [ebp-1Ch]
  int v5; // [esp+34h] [ebp-18h]
  signed int v6; // [esp+38h] [ebp-14h]
  int i; // [esp+3Ch] [ebp-10h]
  int v8; // [esp+40h] [ebp-Ch]

  __main();
  v4 = 0;
  v5 = 0;
  qmemcpy(&v3, _data_start__, 0x19u);
  while ( 1 )
  {
    puts("you can choose one action to execute");
    puts("1 up");
    puts("2 down");
    puts("3 left");
    printf("4 right\n:");
    scanf("%d", &v6);
    if ( v6 == 2 )
    {
      ++v4;
    }
    else if ( v6 > 2 )
    {
      if ( v6 == 3 )
      {
        --v5;
      }
      else
      {
        if ( v6 != 4 )
LABEL_13:
          exit(1);
        ++v5;
      }
    }
    else
    {
      if ( v6 != 1 )
        goto LABEL_13;
      --v4;
    }
    for ( i = 0; i <= 1; ++i )
    {
      if ( *(&v4 + i) < 0 || *(&v4 + i) > 4 )
        exit(1);
    }
    if ( *((_BYTE *)&v8 + 5 * v4 + v5 - 41) == '1' )
      exit(1);
    if ( *((_BYTE *)&v8 + 5 * v4 + v5 - 41) == '#' )
    {
      puts("\nok, the order you enter is the flag!");
      exit(0);
    }
  }
}
```

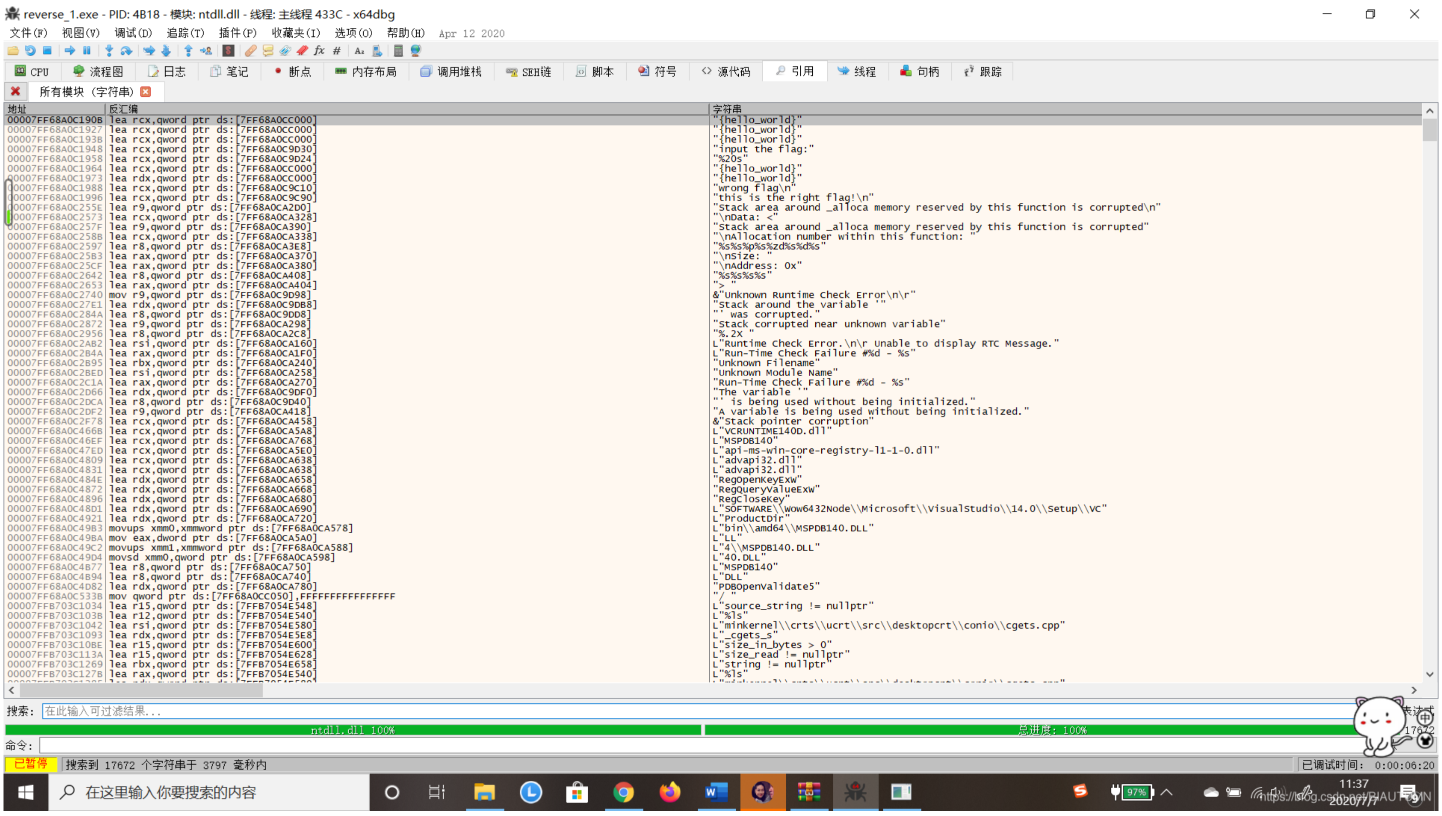
对13行的datastart进行追踪，并结合后面16-21行，51-55行，可以发现是一道5*5的迷宫移动题

```
*1111
01000
01010
00010
1111#
```

其中从*开始移动到#，0为可移动路径
在结合程序运行
得到flag{222441144222}

reverse1

查壳，64位，64位od被win10吃了好几次，改用x64dbg
打开直接查找全部模块字符串

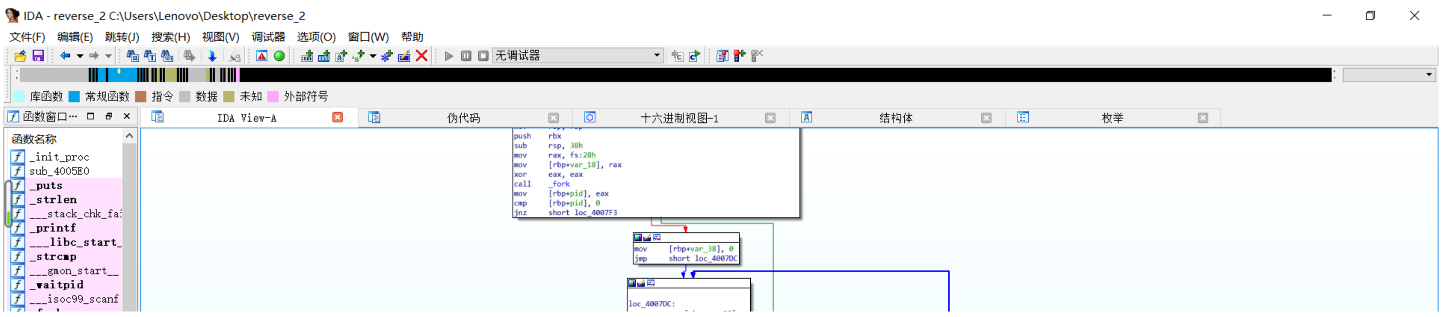


```
reverse_1.exe - PID: 4B18 - 模块: ntdll.dll - 线程: 主线程 433C - x64dbg
所有模块 (字符串)
地址              反汇编                                                  字符串
00007FF68A0C190B  lea rcx,qword ptr ds:[7FF68A0CC000]                     "{hello_world}"
00007FF68A0C1927  lea rcx,qword ptr ds:[7FF68A0CC000]                     "{hello_world}"
00007FF68A0C193B  lea rcx,qword ptr ds:[7FF68A0CC000]                     "{hello_world}"
00007FF68A0C1948  lea rcx,qword ptr ds:[7FF68A0C9D30]                     "input the flag:"
00007FF68A0C1958  lea rcx,qword ptr ds:[7FF68A0C9D24]                     "%20s"
00007FF68A0C1964  lea rcx,qword ptr ds:[7FF68A0CC000]                     "{hello_world}"
00007FF68A0C1973  lea rdx,qword ptr ds:[7FF68A0CC000]                     "{hello_world}"
00007FF68A0C1988  lea rcx,qword ptr ds:[7FF68A0C9C10]                     "wrong flag\n"
00007FF68A0C1996  lea rcx,qword ptr ds:[7FF68A0C9C90]                     "this is the right flag!\n"
00007FF68A0C255E  lea r9,qword ptr ds:[7FF68A0CA2D0]                      "Stack area around _alloca memory reserved by this function is corrupted\n"
00007FF68A0C2573  lea rcx,qword ptr ds:[7FF68A0CA328]                     "\nData: <"
00007FF68A0C257F  lea r9,qword ptr ds:[7FF68A0CA390]                      "Stack area around _alloca memory reserved by this function is corrupted"
00007FF68A0C258B  lea rcx,qword ptr ds:[7FF68A0CA338]                     "\nAllocation number within this function: "
00007FF68A0C2597  lea r8,qword ptr ds:[7FF68A0CA3E8]                      "%s%s%p%s%zd%s%d%s"
00007FF68A0C25B3  lea rax,qword ptr ds:[7FF68A0CA370]                     "\nSize: "
00007FF68A0C25CF  lea rax,qword ptr ds:[7FF68A0CA380]                     "\nAddress: 0x"
00007FF68A0C2642  lea r8,qword ptr ds:[7FF68A0CA408]                      "%s%s%s%s"
00007FF68A0C2653  lea rax,qword ptr ds:[7FF68A0CA404]                     "> "
00007FF68A0C2740  mov r9,qword ptr ds:[7FF68A0C9D98]                      &"Unknown Runtime Check Error\n\r"
00007FF68A0C27E1  lea rdx,qword ptr ds:[7FF68A0C9DB8]                     "Stack around the variable '"
00007FF68A0C284A  lea r8,qword ptr ds:[7FF68A0C9DD8]                      "' was corrupted."
00007FF68A0C2872  lea r9,qword ptr ds:[7FF68A0CA298]                      "Stack corrupted near unknown variable"
00007FF68A0C2956  lea r8,qword ptr ds:[7FF68A0CA2C8]                      "%.2X "
00007FF68A0C2AB2  lea rsi,qword ptr ds:[7FF68A0CA160]                     L"Runtime Check Error.\n\r Unable to display RTC Message."
00007FF68A0C2B4A  lea rax,qword ptr ds:[7FF68A0CA1F0]                     L"Run-Time Check Failure #%d - %s"
00007FF68A0C2B95  lea rbx,qword ptr ds:[7FF68A0CA240]                     "Unknown Filename"
00007FF68A0C2BED  lea rsi,qword ptr ds:[7FF68A0CA258]                     "Unknown Module Name"
00007FF68A0C2C1A  lea rax,qword ptr ds:[7FF68A0CA270]                     "Run-Time Check Failure #%d - %s"
00007FF68A0C2D66  lea rdx,qword ptr ds:[7FF68A0C9DF0]                     "The variable '"
00007FF68A0C2DCA  lea r8,qword ptr ds:[7FF68A0C9D40]                      "' is being used without being initialized."
00007FF68A0C2DF2  lea r9,qword ptr ds:[7FF68A0CA418]                      "A variable is being used without being initialized."
00007FF68A0C2F78  lea rcx,qword ptr ds:[7FF68A0CA458]                     &"Stack pointer corruption"
00007FF68A0C466B  lea rcx,qword ptr ds:[7FF68A0CA5A8]                     L"VCRUNTIME140D.dll"
00007FF68A0C46EF  lea rcx,qword ptr ds:[7FF68A0CA768]                     L"MSPDB140"
00007FF68A0C47ED  lea rcx,qword ptr ds:[7FF68A0CA5E0]                     L"api-ms-win-core-registry-l1-1-0.dll"
00007FF68A0C4809  lea rcx,qword ptr ds:[7FF68A0CA638]                     L"advapi32.dll"
00007FF68A0C4831  lea rcx,qword ptr ds:[7FF68A0CA638]                     L"advapi32.dll"
00007FF68A0C484E  lea rdx,qword ptr ds:[7FF68A0CA658]                     "RegOpenKeyExW"
00007FF68A0C4872  lea rdx,qword ptr ds:[7FF68A0CA668]                     "RegQueryValueExW"
00007FF68A0C4896  lea rdx,qword ptr ds:[7FF68A0CA680]                     "RegCloseKey"
00007FF68A0C48D1  lea rdx,qword ptr ds:[7FF68A0CA690]                     L"SOFTWARE\\Wow6432Node\\Microsoft\\VisualStudio\\14.0\\Setup\\VC"
00007FF68A0C4921  lea rdx,qword ptr ds:[7FF68A0CA720]                     L"ProductDir"
00007FF68A0C49B3  movups xmm0,xmmword ptr ds:[7FF68A0CA578]               L"bin\\amd64\\MSPDB140.DLL"
00007FF68A0C49BA  mov eax,dword ptr ds:[7FF68A0CA5A0]                     L"LL"
00007FF68A0C49C2  movups xmm1,xmmword ptr ds:[7FF68A0CA588]               L"4\\MSPDB140.DLL"
00007FF68A0C49D4  movsd xmm0,qword ptr ds:[7FF68A0CA598]                  L"40.DLL"
00007FF68A0C4B77  lea r8,qword ptr ds:[7FF68A0CA750]                      L"MSPDB140"
00007FF68A0C4B94  lea r8,qword ptr ds:[7FF68A0CA740]                      L"DLL"
00007FF68A0C4D82  lea rdx,qword ptr ds:[7FF68A0CA780]                     "PDBOpenValidate5"
00007FF68A0C533B  mov qword ptr ds:[7FF68A0CC050],FFFFFFFFFFFFFFFF        "/ "
00007FFB703C1034  lea r15,qword ptr ds:[7FFB7054E548]                     L"source_string != nullptr"
00007FFB703C103B  lea r12,qword ptr ds:[7FFB7054E540]                     L"%ls"
00007FFB703C1042  lea rsi,qword ptr ds:[7FFB7054E580]                     L"minkernel\\crts\\ucrt\\src\\desktopcrt\\conio\\cgets.cpp"
00007FFB703C1093  lea rdx,qword ptr ds:[7FFB7054E5E8]                     L"_cgets_s"
00007FFB703C10BE  lea r15,qword ptr ds:[7FFB7054E600]                     L"size_in_bytes > 0"
00007FFB703C113A  lea r15,qword ptr ds:[7FFB7054E628]                     L"size_read != nullptr"
00007FFB703C1269  lea rbx,qword ptr ds:[7FFB7054E658]                     L"string != nullptr"
00007FFB703C127B  lea rax,qword ptr ds:[7FFB7054E540]                     L"%ls"
搜索: 在此输入可过滤结果...
ntdll.dll 100%                                                             总进度: 100%
命令:
已暂停  搜索到 17672 个字符串于 3797 毫秒内                                  已调试时间: 0:00:06:20
```

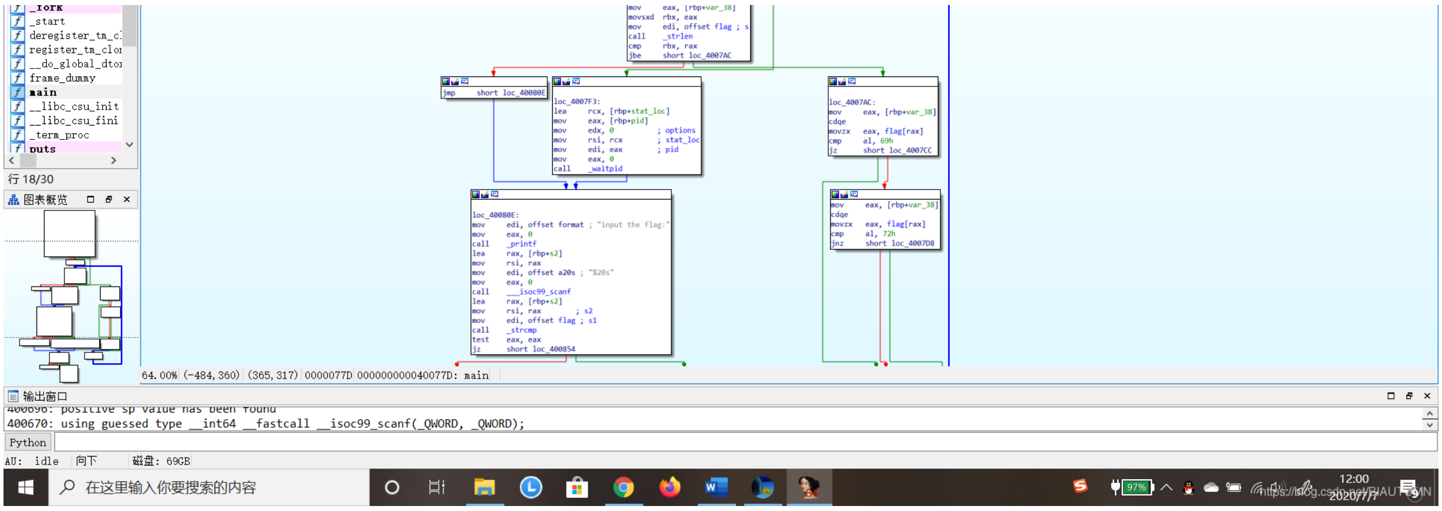
在代码最上方可以看到{hello_world},似乎就是flag，但是在input flag中，flag中的o全部被替换成0，所以最后的flag应该是{hell0_w0rld}

reverse2

查壳，64位，ida打开
main函数观察流程图



```
IDA - reverse_2 C:\Users\Lenovo\Desktop\reverse_2

函数名称
_init_proc
sub_4005E0
_puts
_strlen
___stack_chk_fai
_printf
___libc_start_
_strcmp
___gmon_start__
_waitpid
___isoc99_scanf

push    rbx
sub     rsp, 38h
mov     rax, fs:28h
mov     [rbp+var_18], rax
xor     eax, eax
call    _fork
mov     [rbp+pid], eax
cmp     [rbp+pid], 0
jnz     short loc_4007F3

mov     [rbp+var_38], 0
jmp     short loc_4007DC

loc_4007DC:
```

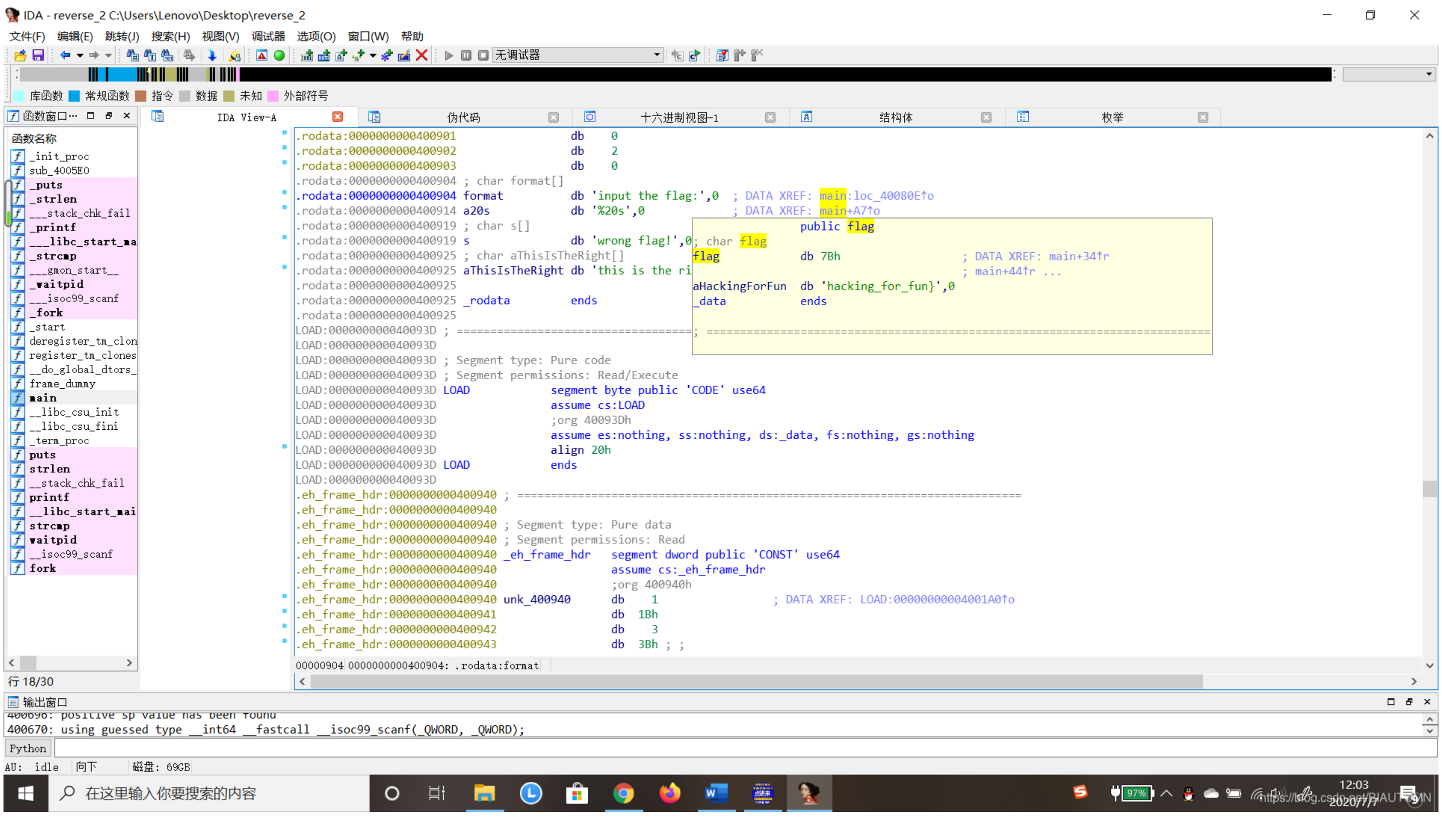


```
mov     eax, [rbp+var_38]
movsxd  rbx, eax
mov     edi, offset flag ; s
call    _strlen
cmp     rbx, rax
jbe     short loc_4007AC

jmp     short loc_40080E

loc_4007F3:
lea     rcx, [rbp+stat_loc]
mov     eax, [rbp+pid]
mov     edx, 0          ; options
mov     rsi, rcx        ; stat_loc
mov     edi, eax        ; pid
mov     eax, 0
call    _waitpid

loc_4007AC:
mov     eax, [rbp+var_38]
cdqe
movzx   eax, flag[rax]
cmp     al, 69h
jz      short loc_4007CC

mov     eax, [rbp+var_38]
cdqe
movzx   eax, flag[rax]
cmp     al, 72h
jnz     short loc_4007D8

loc_40080E:
mov     edi, offset format ; "input the flag:"
mov     eax, 0
call    _printf
lea     rax, [rbp+s2]
mov     rsi, rax
mov     edi, offset a20s ; "%20s"
mov     eax, 0
call    ___isoc99_scanf
lea     rax, [rbp+s2]
mov     rsi, rax        ; s2
mov     edi, offset flag ; s1
call    _strcmp
test    eax, eax
jz      short loc_400854
```

```
64.00% (-484,360) (365,317) 0000077D 000000000040077D: main
400690: positive sp value has been found
400670: using guessed type __int64 __fastcall __isoc99_scanf(_QWORD, _QWORD);
```

从左边的input可以得出，左边是判断，右边是对字符串进行变换

在main函数中对flag进行查看



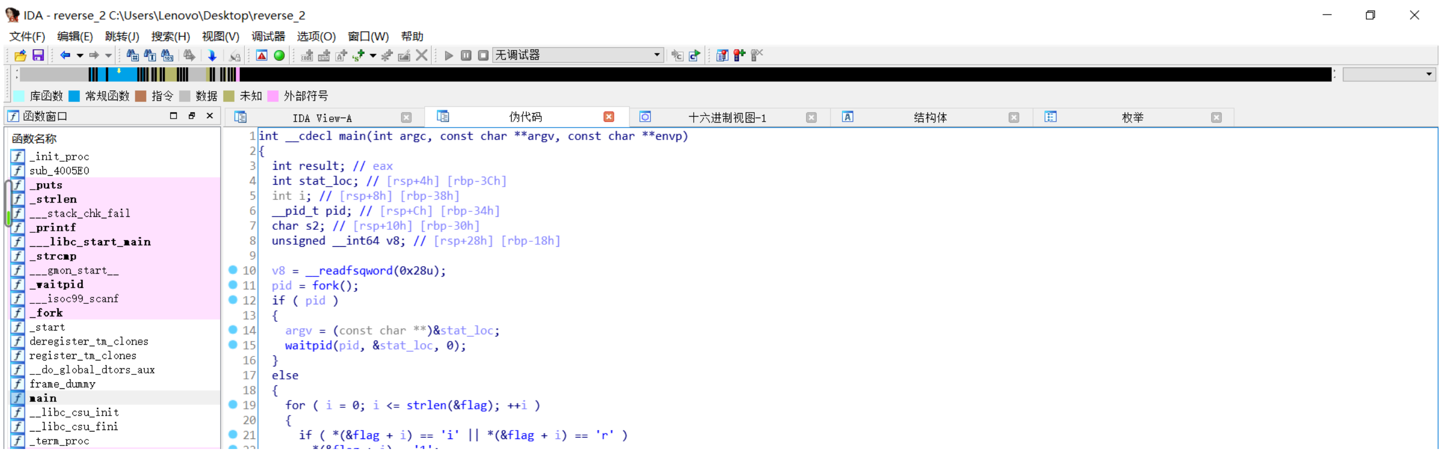
```
.rodata:0000000000400901                 db    0
.rodata:0000000000400902                 db    2
.rodata:0000000000400903                 db    0
.rodata:0000000000400904 ; char format[]
.rodata:0000000000400904 format          db 'input the flag:',0  ; DATA XREF: main:loc_40080E↑o
.rodata:0000000000400914 a20s            db '%20s',0             ; DATA XREF: main+A7↑o
.rodata:0000000000400919 ; char s[]
.rodata:0000000000400919 s               db 'wrong flag!',0
.rodata:0000000000400925 ; char aThisIsTheRight[]
.rodata:0000000000400925 aThisIsTheRight db 'this is the ri
.rodata:0000000000400925
.rodata:0000000000400925 _rodata         ends
.rodata:0000000000400925
LOAD:000000000040093D ; ================================
LOAD:000000000040093D
LOAD:000000000040093D ; Segment type: Pure code
LOAD:000000000040093D ; Segment permissions: Read/Execute
LOAD:000000000040093D LOAD            segment byte public 'CODE' use64
LOAD:000000000040093D                 assume cs:LOAD
LOAD:000000000040093D                 ;org 40093Dh
LOAD:000000000040093D                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
LOAD:000000000040093D                 align 20h
LOAD:000000000040093D LOAD            ends
LOAD:000000000040093D
.eh_frame_hdr:0000000000400940 ; ==============================================
.eh_frame_hdr:0000000000400940
.eh_frame_hdr:0000000000400940 ; Segment type: Pure data
.eh_frame_hdr:0000000000400940 ; Segment permissions: Read
.eh_frame_hdr:0000000000400940 _eh_frame_hdr   segment dword public 'CONST' use64
.eh_frame_hdr:0000000000400940                 assume cs:_eh_frame_hdr
.eh_frame_hdr:0000000000400940                 ;org 400940h
.eh_frame_hdr:0000000000400940 unk_400940      db    1               ; DATA XREF: LOAD:00000000004001A0↑o
.eh_frame_hdr:0000000000400941                 db  1Bh
.eh_frame_hdr:0000000000400942                 db    3
.eh_frame_hdr:0000000000400943                 db  3Bh ; ;
```

```
                public flag
; char flag
flag            db 7Bh                  ; DATA XREF: main+34↑r
                                        ; main+44↑r ...
aHackingForFun  db 'hacking_for_fun}',0
_data           ends

; ==================================================================
```

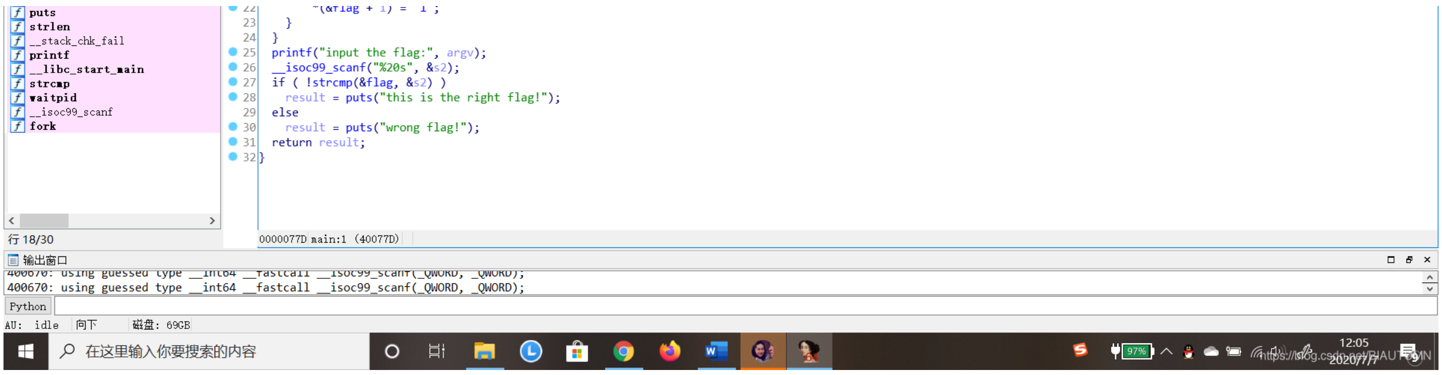
```
00000904 0000000000400904: .rodata:format
400690: positive sp value has been found
400670: using guessed type __int64 __fastcall __isoc99_scanf(_QWORD, _QWORD);
```

发现有{hacking_for_fun}

观察main函数伪代码



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax
  int stat_loc; // [rsp+4h] [rbp-3Ch]
  int i; // [rsp+8h] [rbp-38h]
  __pid_t pid; // [rsp+Ch] [rbp-34h]
  char s2; // [rsp+10h] [rbp-30h]
  unsigned __int64 v8; // [rsp+28h] [rbp-18h]

  v8 = __readfsqword(0x28u);
  pid = fork();
  if ( pid )
  {
    argv = (const char **)&stat_loc;
    waitpid(pid, &stat_loc, 0);
  }
  else
  {
    for ( i = 0; i <= strlen(&flag); ++i )
    {
      if ( *(&flag + i) == 'i' || *(&flag + i) == 'r' )
```


```
puts
strlen
__stack_chk_fail
printf
__libc_start_main
strcmp
waitpid
__isoc99_scanf
fork
```
```
        *(&flag + 1) = '1';
    }
  }
  printf("input the flag:", argv);
  __isoc99_scanf("%20s", &s2);
  if ( !strcmp(&flag, &s2) )
    result = puts("this is the right flag!");
  else
    result = puts("wrong flag!");
  return result;
}
```
行 18/30

0000077D main:1 (40077D)

输出窗口
```
400670: using guessed type __int64 __fastcall __isoc99_scanf(_QWORD, _QWORD);
400670: using guessed type __int64 __fastcall __isoc99_scanf(_QWORD, _QWORD);
```
Python

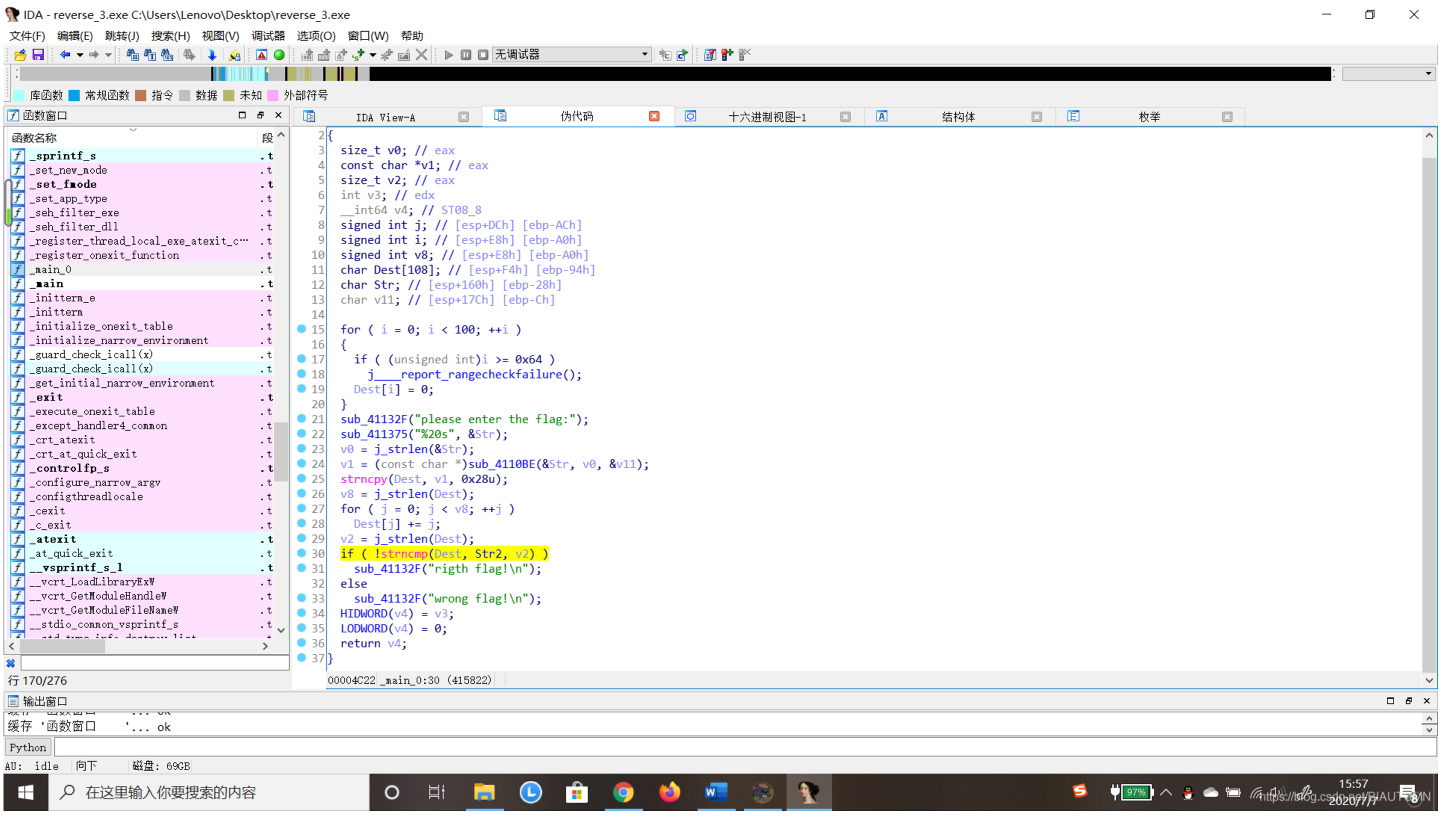
AU: idle 向下 磁盘: 69GB

发现将字符串中的i，r变换成1

所以得到flag为flag{hack1ng_fo1_fun}

reverse3

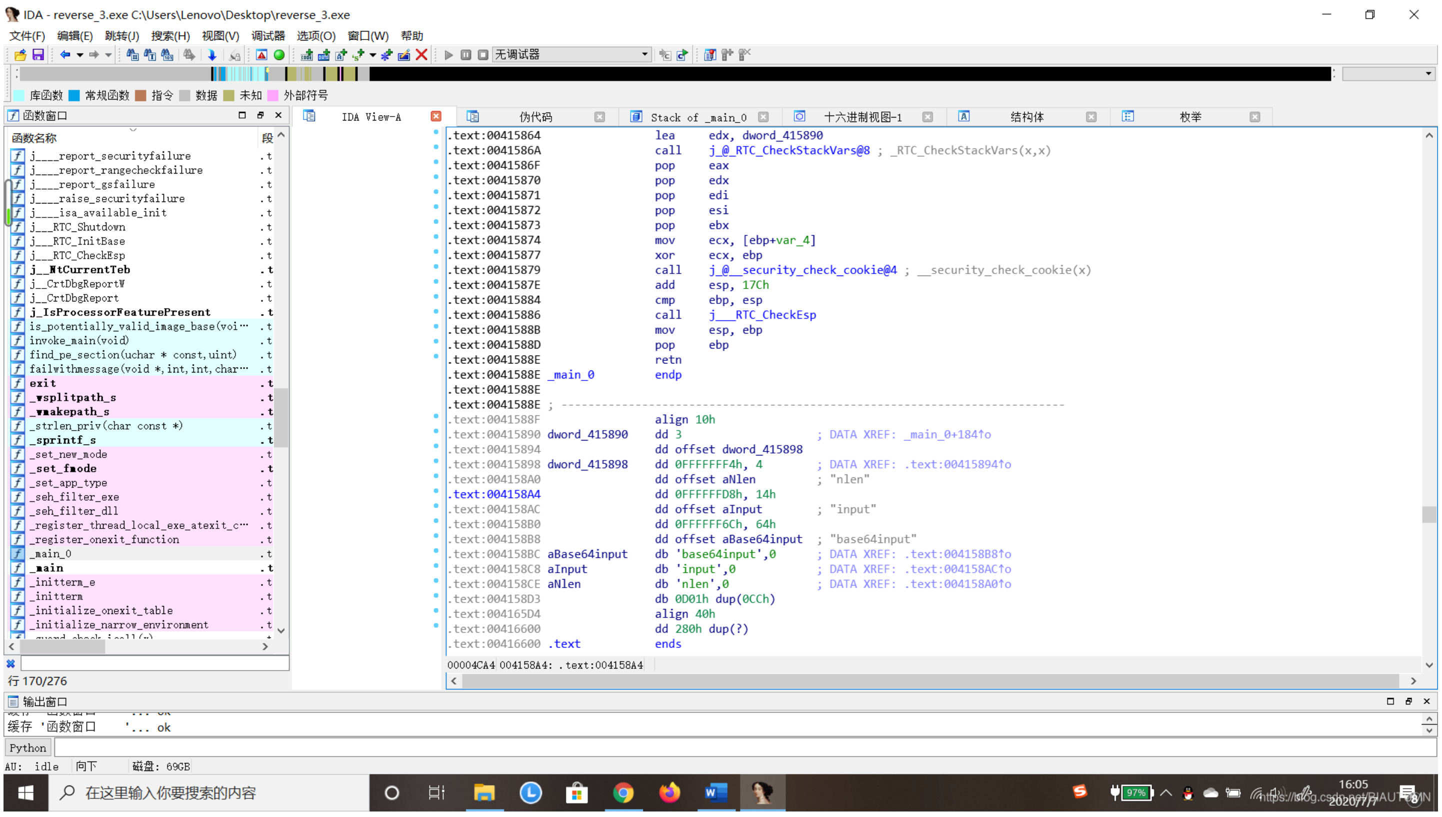
查壳，32位，ida打开，查看main0伪代码



```
 2 {
 3   size_t v0; // eax
 4   const char *v1; // eax
 5   size_t v2; // eax
 6   int v3; // edx
 7   __int64 v4; // ST08_8
 8   signed int j; // [esp+DCh] [ebp-ACh]
 9   signed int i; // [esp+E8h] [ebp-A0h]
10   signed int v8; // [esp+E8h] [ebp-A0h]
11   char Dest[108]; // [esp+F4h] [ebp-94h]
12   char Str; // [esp+160h] [ebp-28h]
13   char v11; // [esp+17Ch] [ebp-Ch]
14
15   for ( i = 0; i < 100; ++i )
16   {
17     if ( (unsigned int)i >= 0x64 )
18       j____report_rangecheckfailure();
19     Dest[i] = 0;
20   }
21   sub_41132F("please enter the flag:");
22   sub_411375("%20s", &Str);
23   v0 = j_strlen(&Str);
24   v1 = (const char *)sub_4110BE(&Str, v0, &v11);
25   strncpy(Dest, v1, 0x28u);
26   v8 = j_strlen(Dest);
27   for ( j = 0; j < v8; ++j )
28     Dest[j] += j;
29   v2 = j_strlen(Dest);
30   if ( !strncmp(Dest, Str2, v2) )
31     sub_41132F("rigth flag!\n");
32   else
33     sub_41132F("wrong flag!\n");
34   HIDWORD(v4) = v3;
35   LODWORD(v4) = 0;
36   return v4;
37 }
```

00004C22 _main_0:30 (415822)

第30行str2为变换后的flag，只要找出算法将str2逆运算即可



```
.text:00415864                 lea     edx, dword_415890
.text:0041586A                 call    j_@_RTC_CheckStackVars@8 ; _RTC_CheckStackVars(x,x)
.text:0041586F                 pop     eax
.text:00415870                 pop     edx
.text:00415871                 pop     edi
.text:00415872                 pop     esi
.text:00415873                 pop     ebx
.text:00415874                 mov     ecx, [ebp+var_4]
.text:00415877                 xor     ecx, ebp
.text:00415879                 call    j_@__security_check_cookie@4 ; __security_check_cookie(x)
.text:0041587E                 add     esp, 17Ch
.text:00415884                 cmp     ebp, esp
.text:00415886                 call    j___RTC_CheckEsp
.text:0041588B                 mov     esp, ebp
.text:0041588D                 pop     ebp
.text:0041588E                 retn
.text:0041588E _main_0         endp
.text:0041588E
.text:0041588E ; ---------------------------------------------------------------------------
.text:0041588F                 align 10h
.text:00415890 dword_415890    dd 3                    ; DATA XREF: _main_0+184↑o
.text:00415894                 dd offset dword_415898
.text:00415898 dword_415898    dd 0FFFFFFF4h, 4        ; DATA XREF: .text:00415894↑o
.text:004158A0                 dd offset aNlen         ; "nlen"
.text:004158A4                 dd 0FFFFFFD8h, 14h
.text:004158AC                 dd offset aInput        ; "input"
.text:004158B0                 dd 0FFFFFF6Ch, 64h
.text:004158B8                 dd offset aBase64input  ; "base64input"
.text:004158BC aBase64input    db 'base64input',0      ; DATA XREF: .text:004158B8↑o
.text:004158C8 aInput          db 'input',0            ; DATA XREF: .text:004158AC↑o
.text:004158CE aNlen           db 'nlen',0             ; DATA XREF: .text:004158A0↑o
.text:004158D3                 db 0D01h dup(0CCh)
.text:004165D4                 align 40h
.text:00416600                 dd 280h dup(?)
.text:00416600 .text           ends
```

00004CA4 004158A4: .text:004158A4

仔细观察，发现input后有base64
对str2中的字符串进行base64解密，得到flag
flag{i_l0ve_you}