逆向入门笔记（一）

原创

Wust-Mo0n5ea丶 于 2020-11-23 19:16:12 发布 98 收藏

分类专栏： 笔记

本文链接：https://blog.csdn.net/qq_33590156/article/details/108954979

writeup

逆向静态：bugku-love

逆向动态：bugku-游戏过关

逆向静态：bugku-love

下载文件，是一个exe可执行文件，ida看看先，找到main，双击main0

```
__int64 __cdecl main_0()
{
  size_t v0; // eax
  const char *v1; // eax
  size_t v2; // eax
  int v3; // edx
  __int64 v4; // ST08_8
  signed int j; // [esp+DCh] [ebp-ACh]
  signed int i; // [esp+E8h] [ebp-A0h]
  signed int v8; // [esp+E8h] [ebp-A0h]
  char Dest[108]; // [esp+F4h] [ebp-94h]
  char Str; // [esp+160h] [ebp-28h]
  char v11; // [esp+17Ch] [ebp-Ch]

  for ( i = 0; i < 100; ++i )
  {
    if ( (unsigned int)i >= 0x64 )
      j____report_rangecheckfailure();
    Dest[i] = 0;
  }
  sub_41132F("please enter the flag:");
  sub_411375("%20s", &Str);
  v0 = j_strlen(&Str);
  v1 = (const char *)sub_4110BE(&Str, v0, &v11);
  strncpy(Dest, v1, 0x28u);
  v8 = j_strlen(Dest);
  for ( j = 0; j < v8; ++j )
    Dest[j] += j;
  v2 = j_strlen(Dest);
  if ( !strncmp(Dest, Str2, v2) )
    sub_41132F("rigth flag!\n");
  else
    sub_41132F("wrong flag!\n");
```

```
34  HIDWORD(v4) = v3;
35  LODWORD(v4) = 0;
36  return v4;
37}
```


先直接看最下面，如果Dest和Str2相等，则为flag，所以双击Str2，导出后

```
char Str2[] =
"e3niflH9b_C@n@dH";
```

再看Dest，第一个for循环中，0x64就是100，所以这个循环不执行，无视掉，重点就是sub_4110BE函数，双击看看

```
4  int v5; // STE0_4
5  int v6; // STE0_4
6  int v7; // [esp+D4h] [ebp-38h]
7  signed int i; // [esp+E0h] [ebp-2Ch]
8  unsigned int v9; // [esp+ECh] [ebp-20h]
9  int v10; // [esp+ECh] [ebp-20h]
10 signed int v11; // [esp+ECh] [ebp-20h]
11 void *Dst; // [esp+F8h] [ebp-14h]
12 char *v13; // [esp+104h] [ebp-8h]
13
14 if ( !a1 || !a2 )
15   return 0;
16 v9 = a2 / 3;
17 if ( (signed int)(a2 / 3) % 3 )
18   ++v9;
19 v10 = 4 * v9;
20 *a3 = v10;
21 Dst = malloc(v10 + 1);
22 if ( !Dst )
23   return 0;
24 j_memset(Dst, 0, v10 + 1);
25 v13 = a1;
26 v11 = a2;
27 v7 = 0;
28 while ( v11 > 0 )
29 {
30   byte_41A144[2] = 0;
31   byte_41A144[1] = 0;
32   byte_41A144[0] = 0;
33   for ( i = 0; i < 3 && v11 >= 1; ++i )
34   {
35     byte_41A144[i] = *v13;
36     --v11;
37     ++v13;
38   }
39   if ( !i )
40     break;
41   switch ( i )
42   {
43     case 1:
44       *((_BYTE *)Dst + v7) = aAbcdefghijklmn[(signed int)(unsigned __int8)byte_41A144[0] >> 2];
45       v4 = v7 + 1;
46       *((_BYTE *)Dst + v4++) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | 16 * (byte_41A144[0] & 3)];
47       *((_BYTE *)Dst + v4++) = aAbcdefghijklmn[64];
48       *((_BYTE *)Dst + v4) = aAbcdefghijklmn[64];
49       v7 = v4 + 1;
50       break;
51     case 2:
52       *((_BYTE *)Dst + v7) = aAbcdefghijklmn[(signed int)(unsigned __int8)byte_41A144[0] >> 2];
53       v5 = v7 + 1;
54       *((_BYTE *)Dst + v5++) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | 16 * (byte_41A144[0] & 3)];
55       *((_BYTE *)Dst + v5++) = aAbcdefghijklmn[((byte_41A144[2] & 0xC0) >> 6) | 4 * (byte_41A144[1] & 0xF)];
56       *((_BYTE *)Dst + v5) = aAbcdefghijklmn[64];
57       v7 = v5 + 1;
58       break;
59     case 3:
60       *((_BYTE *)Dst + v7) = aAbcdefghijklmn[(signed int)(unsigned __int8)byte_41A144[0] >> 2];
61       v6 = v7 + 1;
62       *((_BYTE *)Dst + v6++) = aAbcdefghijklmn[((byte_41A144[1] & 0xF0) >> 4) | 16 * (byte_41A144[0] & 3)];
63       *((_BYTE *)Dst + v6++) = aAbcdefghijklmn[((byte_41A144[2] & 0xC0) >> 6) | 4 * (byte_41A144[1] & 0xF)];
64       *((_BYTE *)Dst + v6) = aAbcdefghijklmn[byte_41A144[2] & 0x3F];
65       v7 = v6 + 1;
66       break;
67   }
68 }
69 *((_BYTE *)Dst + v7) = 0;
70 return Dst;
71}
```

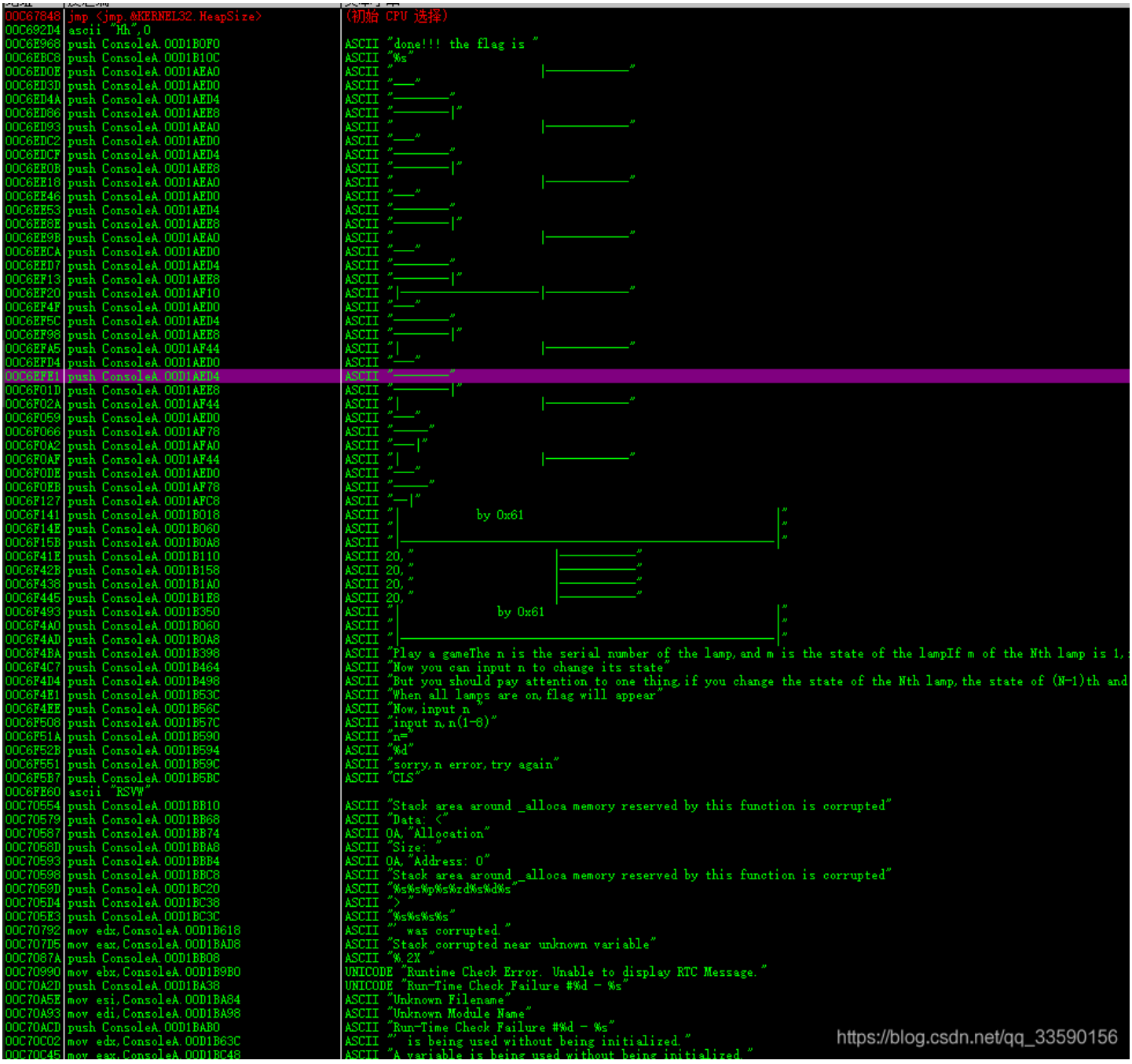
很长的一堆啊，但仔细分析完发现就是将它扩展然后进行base64加密，之后回到main0进行+j，所以写个脚本，将之前导出的Str2拿过来：

```
a ='e3nifIH9b_C@n@dH'
flagbase64=''
for i in range(len(a)):
    flagbase64+=chr(ord(a[i])-i)
print(flagbase64)
```

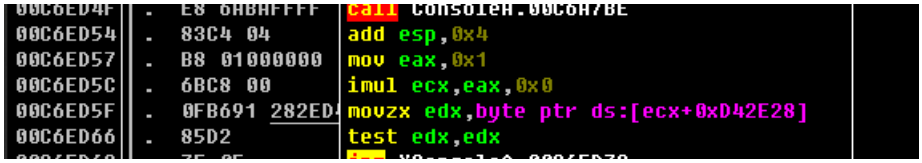
输出得到base64加密后的flag：e2lfbDB2ZV95b3V9，之后进行base64解密后：{i_l0ve_you}
所以flag为flag{i_l0ve_you}

逆向动态：bugku-游戏过关

打开之后，确实和名字一样，是个游戏，ida打开，发现连main函数都没有，只能打开od了，搜索一下字符串flag



双击点开发现，每一步运行完的都指向一个地方啊



```
00C6ED68  .v 75 0F        jnz XConsoleA.00C6ED79
00C6ED6A  .  68 E0AED100  push ConsoleA.00D1AEE0
00C6ED6F  .  E8 4ABAFFFF  call ConsoleA.00C6A7BE
00C6ED74  .  83C4 04      add esp,0x4
00C6ED77  .v EB 0D        jmp XConsoleA.00C6ED86
00C6ED79  >  68 E4AED100  push ConsoleA.00D1AEE4
00C6ED7E  .  E8 3BBAFFFF  call ConsoleA.00C6A7BE
00C6ED83  .  83C4 04      add esp,0x4
00C6ED86  >  68 E8AED100  push ConsoleA.00D1AEE8                ASCI
00C6ED8B  .  E8 2EBAFFFF  call ConsoleA.00C6A7BE
00C6ED90  .  83C4 04      add esp,0x4
00C6ED93  .  68 A0AED100  push ConsoleA.00D1AEA0                ASCI
00C6ED98  .  E8 21BAFFFF  call ConsoleA.00C6A7BE
00C6ED9D  .  83C4 04      add esp,0x4
00C6EDA0  .  B8 01000000  mov eax,0x1
00C6EDA5  .  C1E0 00      shl eax,0x0
00C6EDA8  .  0FB688 282ED movzx ecx,byte ptr ds:[eax+0xD42E28]
00C6EDAF  .  85C9         test ecx,ecx
00C6EDB1  .v 75 0F        jnz XConsoleA.00C6EDC2
00C6EDB3  .  68 CCAED100  push ConsoleA.00D1AECC
00C6EDB8  .  E8 01BAFFFF  call ConsoleA.00C6A7BE
00C6EDBD  .  83C4 04      add esp,0x4
00C6EDC0  .v EB 0D        jmp XConsoleA.00C6EDCF
00C6EDC2  >  68 D0AED100  push ConsoleA.00D1AED0                ASCI
00C6EDC7  .  E8 F2B9FFFF  call ConsoleA.00C6A7BE
00C6EDCC  .  83C4 04      add esp,0x4
00C6EDCF  >  68 D4AED100  push ConsoleA.00D1AED4                ASCI
00C6EDD4  .  E8 E5B9FFFF  call ConsoleA.00C6A7BE
00C6EDD9  .  83C4 04      add esp,0x4
00C6EDDC  .  B8 01000000  mov eax,0x1
00C6EDE1     C1E0 00      shl eax,0x0
```


就是00C6A7BE，去找这个地址，发现是个jmp，指向00C6F770,那就去这里，

```
00C6F76E     CC           int3
00C6F76F     CC           int3
00C6F770  r> 55           push ebp
00C6F771  .  8BEC         mov ebp,esp
00C6F773  .  81EC D800000 sub esp,0xD8
00C6F779  .  53           push ebx
00C6F77A  .  56           push esi
00C6F77B  .  57           push edi
00C6F77C  .  8DBD 28FFFFF lea edi,[local.54]
00C6F782  .  B9 36000000  mov ecx,0x36
00C6F787  .  B8 CCCCCCCC  mov eax,0xCCCCCCCC
00C6F78C  .  F3:AB        rep stos dword ptr es:[edi]
00C6F78E  .  E8 9F9BFFFF  call ConsoleA.00C69332
00C6F793  .  8D45 0C      lea eax,[arg.2]
00C6F796  .  8945 EC      mov [local.5],eax
00C6F799  .  8B45 EC      mov eax,[local.5]
00C6F79C  .  50           push eax
00C6F79D  .  6A 00        push 0x0
00C6F79F  .  8B4D 08      mov ecx,[arg.1]
00C6F7A2  .  51           push ecx
00C6F7A3  .  6A 01        push 0x1
00C6F7A5  .  E8 BD84FFFF  call ConsoleA.00C67C67
00C6F7AA  .  83C4 04      add esp,0x4
00C6F7AD  .  50           push eax
00C6F7AE  .  E8 4198FFFF  call ConsoleA.00C68FF4
```


下一个断点，运行发现，果然每一步都会跳转到此处，但我们的目的是跳转到done！flag那一段。我们记下done！flag的地址，来到

```
00C6F4B2  .  E8 07B3FFFF  call ConsoleA.00C6A7BE
00C6F4B7  .  83C4 04      add esp,0x4
00C6F4BA  .  68 98B3D100  push ConsoleA.00D1B398
00C6F4BF     E8 FAB2FFFF  call ConsoleA.00C6A7BE
00C6F4C4  .  83C4 04      add esp,0x4
00C6F4C7  .  68 64B4D100  push ConsoleA.00D1B464
00C6F4CC     E8 EDB2FFFF  call ConsoleA.00C6A7BE
00C6F4D1  .  83C4 04      add esp,0x4
00C6F4D4  .  68 98B4D100  push ConsoleA.00D1B498
00C6F4D9     E8 E0B2FFFF  call ConsoleA.00C6A7BE
00C6F4DE  .  83C4 04      add esp,0x4
00C6F4E1  .  68 3CB5D100  push ConsoleA.00D1B53C
00C6F4E6     E8 D3B2FFFF  call ConsoleA.00C6A7BE
00C6F4EB     83C4 04      add esp,0x4
00C6F4EE     68 6CB5D100  push ConsoleA.00D1B56C
```

汇编于此处：00C6F4D9

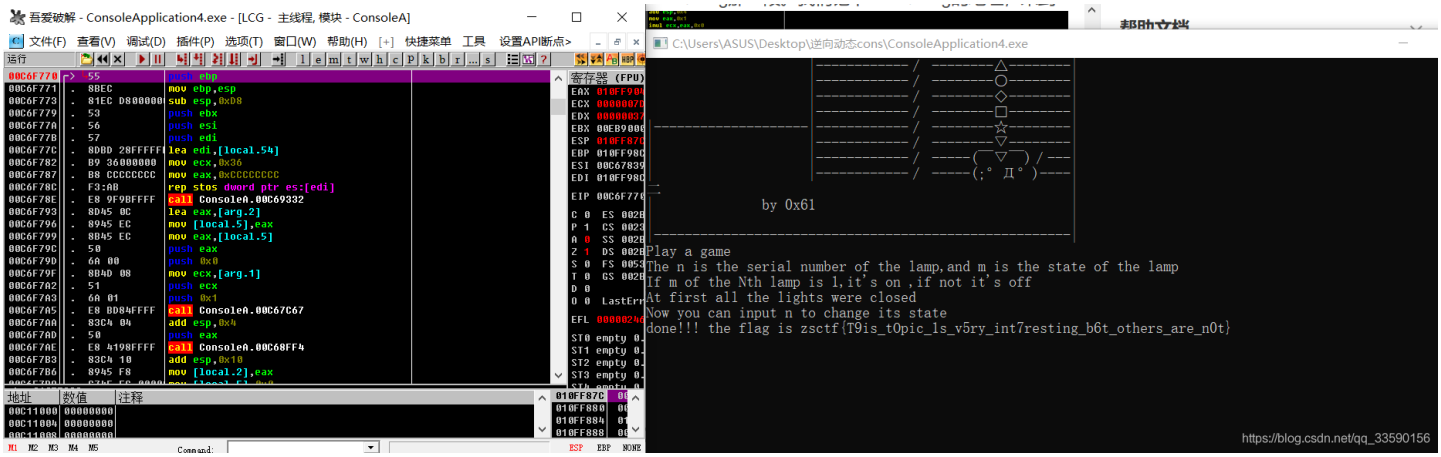
call 00C6E968

使用 NOP 填充　汇编　取消

ASCII "When all lamps are on,flag will appear"

ASCII "Now,input n "

双击进行修改，开始运行程序



拿到flag