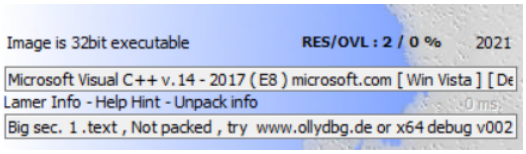
crackme

这题主要考察选手动态调试。扔进exeinfope，32位exe，无壳



main程序，点进sub_4017D0看

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [esp+0h] [ebp-44h]

  sub_4022E0("Please close ida, x32dbg and ollydbg first", v4);
  system("pause");
  if ( !(unsigned __int8)sub_4017D0() )
    sub_401930();
  return 0;
}
```

sub_4017D0，可以看出是检测进程名称。找到for之前的汇编，尝试在动调的时候修改eip直接jmp到CloseHandle，又或者是自己patch一下，防止return 1，从而反反调试。patch教程参考链接（留意这里的dword_406120，之后会讲）

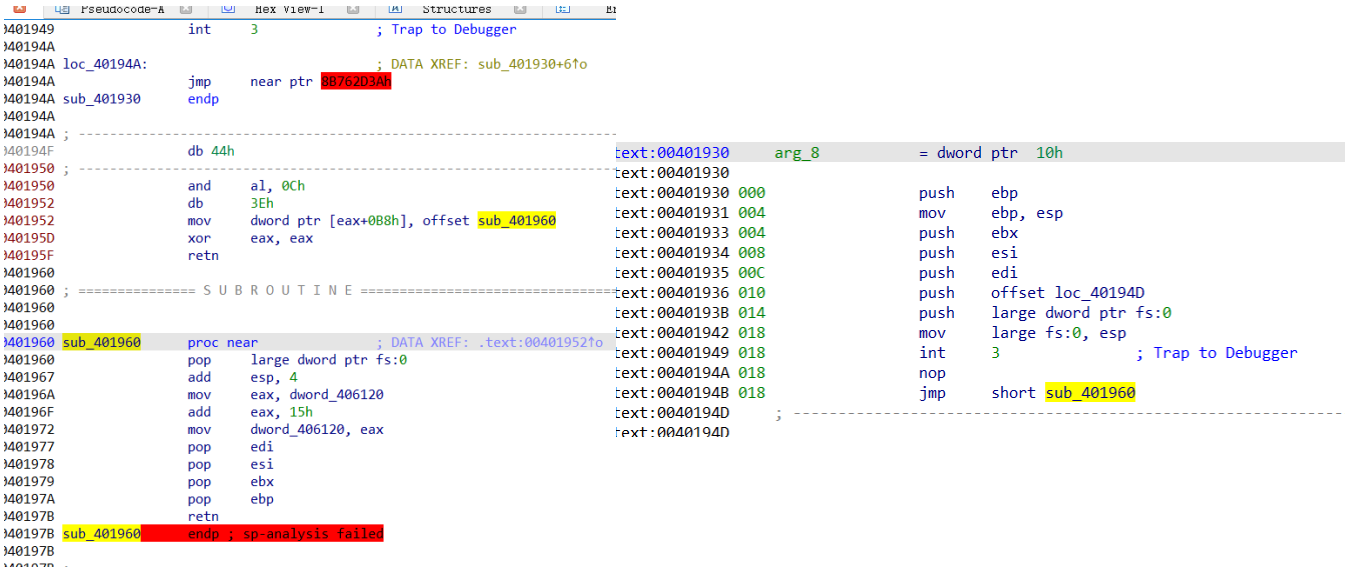
```
char sub_4017D0()
{
  char result; // al
  BOOL i; // [esp+4h] [ebp-134h]
  HANDLE hSnapshot; // [esp+8h] [ebp-130h]
  PROCESSENTRY32 pe; // [esp+Ch] [ebp-12Ch] BYREF

  pe.dwSize = 296;
  hSnapshot = CreateToolhelp32Snapshot(2u, 0);
  if ( hSnapshot == (HANDLE)-1 )
  {
    dword_406120 += 21;
    result = 0;
  }
  else
  {
    for ( i = Process32First(hSnapshot, &pe); i; i = Process32Next(hSnapshot, &pe) )
    {
      if ( !stricmp(pe.szExeFile, "OllyDBG.EXE")
        || !stricmp(pe.szExeFile, "ida64.exe")
        || !stricmp(pe.szExeFile, "x32dbg.exe")
        || !stricmp(pe.szExeFile, "x64dbg.exe")
        || !stricmp(pe.szExeFile, "ida.exe")
        || !stricmp(pe.szExeFile, &byte_4041B8) )
      {
        return 1;
      }
    }
    CloseHandle(hSnapshot);
    dword_406120 += 21;
    result = 0;
  }
```

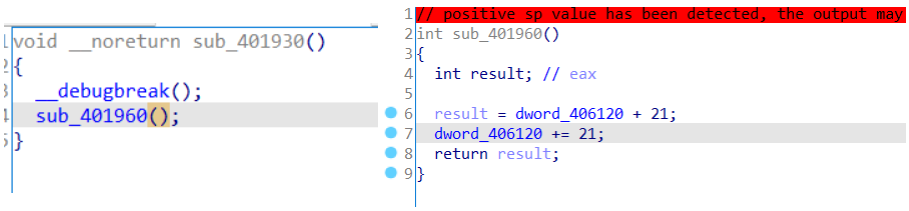
sub_401930是个SEH异常反调试，可以参考源码

```
void __noreturn sub_401930()
{
  __debugbreak();
  JUMPOUT(0x8B762D3A);
}
```

直接查看汇编，参考刚刚上面的源码，反调试的E9刚好起到了花指令的作用，变成了jmp near ptr 8B762D3Ah。在jmp按一下U，在E9下面按c转为代码，然后再把E9 nop掉。



没红色了那就F5一下，按两下进sub_401960（再留意一下dword_406120）



返回到主函数的汇编，发现藏起来了一个函数sub_401FB0

```
.text:00402250 044                pusn    eax             ; cnar
.text:00402251 048                lea     eax, [ebp+var_C]
.text:00402254 048                mov     large fs:0, eax
.text:0040225A 048                push    offset aPleaseCloseIda ; "Please close ida, x32dbg and ollydbg fi"...
.text:0040225F 04C                call    sub_4022F0
.text:00402264 04C                add     esp, 4
.text:00402267 048                push    offset Command  ; "pause"
.text:0040226C 04C                call    ds:system
.text:00402272 04C                add     esp, 4
.text:00402275 048                call    sub_4017D0
.text:0040227A 048                movzx   eax, al
.text:0040227D 048                test    eax, eax
.text:0040227F 048                jz      short loc_402285
.text:00402281 048                xor     eax, eax
.text:00402283 048                jmp     short loc_4022D0
.text:00402285     ; ---------------------------------------------------------------------------
.text:00402285
.text:00402285     loc_402285:                            ; CODE XREF: _main+4F↑j
.text:00402285 048                call    sub_401930
.text:0040228A     ; ---------------------------------------------------------------------------
.text:0040228A 048                mov     ecx, 6
.text:0040228F 048                mov     esi, offset aMocsctfIAmAFak ; "MOCSCTF{I_am_a_fake_flag}"
.text:00402294 048                lea     edi, [ebp+var_38]
.text:00402297 048                rep movsd
.text:00402299 048                movsw
.text:0040229B 048                push    offset sub_401FB0
.text:004022A0 04C                lea     ecx, [ebp+var_18]
.text:004022A3 04C                call    sub_401000
.text:004022A8     ;   try {
```

又或者用shift+f12找到关键字符，对着Format按一下x键，找到调用的地方，然后再F5一下

```
                db  78h ; x
                db    0
; const char Format[]
Format          db 'Welcome to MOCSCTF, please input your flag:',0
                                        ; DATA XREF: sub_401FB0+B↑o
; const char aS[]
aS              db '%s',0               ; DATA XREF: sub_401FB0+2F↑o
```

找到了验证函数，先来看_Initialize_parallel_init_info函数

```
  `anonymous namespace'::_Initialize_parallel_init_info(v6, v9);
  sub_402310("Welcome to MOCSCTF, please input your flag:", v7);
  v10 = (const char *)unknown_libname_4(40);
  sub_402350("%s", (char)v10);
  v8 = strlen(v10);
  if ( v8 != 31 )
    return sub_402310("Failed. Please try again.", v8);
  srand(0);
  for ( i = 0; i < 31; ++i )
  {
    v1 = rand();
    v10[i] = sub_401CD0(v10[i], v1 % 100 + 1, 2);
    v2 = rand();
    v10[i] = sub_401CD0(v10[i], v2 % 100 + 1, 5);
    v3 = rand();
    v10[i] = sub_401CD0(v10[i], v3 % 100 + 1, 3);
    v4 = rand();
    v10[i] = sub_401CD0(v10[i], v4 % 100 + 1, 1);
    v5 = rand();
    v10[i] = sub_401CD0(v10[i], v5 % 100 + 1, 4);
    v10[i] = sub_401CD0(v10[i], 1, 0);
    if ( v10[i] != byte_404294[i] )
    {
      sub_402310("Failed. Please try again.", 31);
      return sub_402310("Congratulations! You cracked this software.", v8);
    }
  }
  return sub_402310("Congratulations! You cracked this software.", v8);
}
```

发现了NtSetInformationThread，这整个函数是隐藏线程防止调试用的，参考链接，直接eip修改过就行（再留意一下dword_406120）

```
int __cdecl `anonymous namespace'::_Initialize_parallel_init_info()
{
  HANDLE v0; // eax
  int result; // eax
  FARPROC NtSetInformationThread; // [esp+4h] [ebp-8h]
  HMODULE hModule; // [esp+8h] [ebp-4h]

  hModule = LoadLibraryA("ntdll.dll");
  NtSetInformationThread = GetProcAddress(hModule, "NtSetInformationThread");
  v0 = GetCurrentThread();
  result = ((int (__stdcall *)(HANDLE, int, _DWORD, _DWORD))NtSetInformationThread)(v0, 17, 0, 0);
  dword_406120 += 14;
  return result;
}
```

再看主函数里面的加密函数sub_401CD0，可看出来应该是做了加密的函数，解密后动态调用

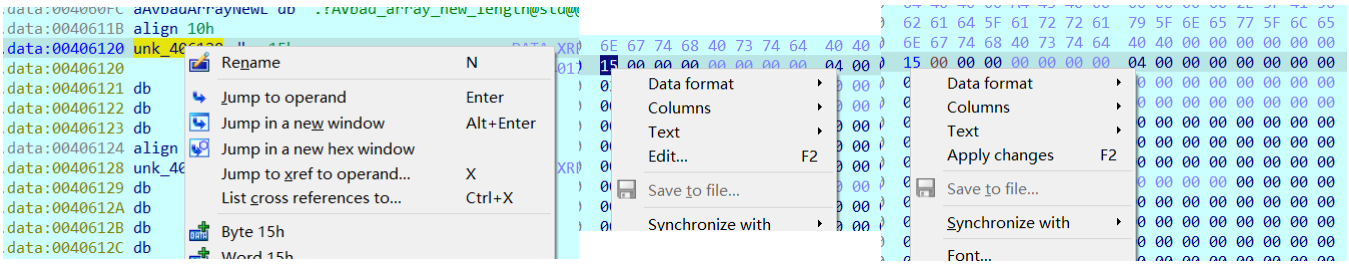
```
  switch ( a3 )
  {
    case 1:
      v5 = (int (__stdcall *)(int, int))sub_401DA0(&unk_404208, 23);
      result = v5(a1, a2);
      break;
    case 2:
      v6 = (int (__stdcall *)(int, int))sub_401DA0(&unk_404250, 43);
      result = v6(a1, a2);
      break;
    case 3:
      v7 = (int (__stdcall *)(int, int))sub_401DA0(&unk_404220, 24);
      result = v7(a1, a2);
      break;
    case 4:
      v8 = (int (__stdcall *)(int, int))sub_401DA0(&unk_404274, 30);
      result = v8(a1, a2);
      break;
    case 5:
      v9 = (int (__stdcall *)(int, int))sub_401DA0(&unk_4041F0, 23);
      result = v9(a1, a2);
      break;
    default:
      result = -1;
      break;
  }
}
else
{
  v4 = (int (__stdcall *)(int, int))sub_401DA0(&unk_404238, 24);
  result = v4(a1, a2);
}
```

```
int __cdecl sub_401DA0(int a1, int a2)
{
  int v3; // [esp+4h] [ebp-8h]
  int i; // [esp+8h] [ebp-4h]

  v3 = unknown_libname_4(a2);
  for ( i = 0; i < a2; ++i )
    *(_BYTE *)(i + v3) = dword_406120 ^ *(_BYTE *)(i + a1);
  return v3;
}
```

发现是用之前看到的dword_406120解密的，当反调试函数都过了，让dword_406120+=了21,21,14，即dword_406120(下面改成了unk_406120)为56时才解密成功，从而调用函数。

右键Jump in a new hex window –>找到15h右键Edit–>改成38（即十进制56）再右键Apply changes。



成功解码之后再慢慢步入进加密函数，当中加了花指令，花指令可参考安全客

```
debug046:004C73BE db    0
debug046:004C73BF db  25h ; %
debug046:004C73C0 ; ---------------------------------------------------------------------------
debug046:004C73C0 push    ebp
debug046:004C73C1 mov     ebp, esp
debug046:004C73C3 push    ebx
debug046:004C73C4 push    esi
debug046:004C73C5 push    edi
debug046:004C73C6 test    eax, eax
debug046:004C73C8 jnz     short near ptr loc_4C73CA+2
debug046:004C73CA
debug046:004C73CA loc_4C73CA:                             ; CODE XREF: debug046:004C73C8↑j
debug046:004C73CA jmp     near ptr 891FFBCh
debug046:004C73CA ; ---------------------------------------------------------------------------
debug046:004C73CF db  0Bh
debug046:004C73D0 db  45h ; E
debug046:004C73D1 db  0Ch
debug046:004C73D2 db  8Bh
debug046:004C73D3 db  4Dh ; M
debug046:004C73D4 db    8
debug046:004C73D5 db  23h ; #
debug046:004C73D6 db  4Dh ; M
```

我出题的时候发现有时候jnz没跳转，所以又加了test eax, eax
所以整个花指令模式如下

```
__asm{
    test eax, eax
    __emit(0x75) //jnz $+4
    __emit(0x02)
    __emit(0xE9) //干扰IDA
    __emit(0xED)
}
```

这个花指令和上面是同一个，用的E9，参考上面的做法patch一下，把E9，ED nop掉

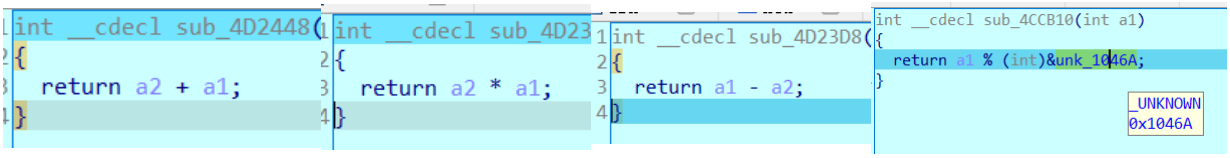
```
debug046:004C73C5 push    edi
debug046:004C73C6 test    eax, eax
debug046:004C73C8 jnz     short loc_4C73CC
debug046:004C73C8 ; ---------------------------
debug046:004C73CA db 0E9h
debug046:004C73CB db 0EDh
debug046:004C73CC ; ---------------------------
debug046:004C73CC
debug046:004C73CC loc_4C73CC:
debug046:004C73CC mov     eax, [ebp+8]
debug046:004C73CF or      eax, [ebp+0Ch]
```

```
debug046:004C73C5 push    edi
debug046:004C73C6 test    eax, eax
debug046:004C73C8 jnz     short loc_4C73CC
debug046:004C73CA nop
debug046:004C73CB nop
debug046:004C73CC
debug046:004C73CC loc_4C73CC:
debug046:004C73CC mov     eax, [ebp+8]
debug046:004C73CF or      eax, [ebp+0Ch]
debug046:004C73D2 mov     ecx, [ebp+8]
```

在最上面的push ebp右键 create function，成功建立函数后就可以F5了，图片这个实际上就是异或运算，都是按位运算，试一下就会发现是异或

```
int __cdecl sub_4C73C0(int a1, int a2)
{
  return ~(a2 & a1) & (a2 | a1);
}
```

以此类推，发现运算分别为加减乘除异或和取余66666（十六进制0x1046A，小彩蛋），其中除1等于没效果，取余66666还是调用了rand，所以还原加密过程的时候还是要保留



可以看出可以单字节爆破，还原一次加密过程，再爆破就得到flag了（导出加密的数据不细说了，shift+e）

```
#include<stdlib.h>
#include<stdio.h>
#include<iostream>

#define rand100 rand()%100+1

const char enflag[] = {
  0x0C,0x17,0x80,0x40,0x29,0x34,0x0C,0x29,0x28,0xA1,
  0x3A,0x80,0x82,0x1D,0x00,0x18,0xC3,0xCA,0x10,0x2E,
  0xD3,0x21,0x48,0xA5,0x3A,0x99,0xFB,0x46,0x0F,0xC6,
  0x78
};

void gen_rand(){
    srand(0);
    for(int i=0;i<31;i++){
        if(i%2==0)printf("\n");
        for(int j=0;j<5;j++){
            printf("0x%02X,",rand100);
        }
    }
}

const char random_list[] = {
  0x27,0x14,0x27,0x26,0x38,0x62,0x42,0x56,0x33,0x0D,
  0x36,0x01,0x2B,0x52,0x26,0x16,0x2E,0x56,0x62,0x51,
  0x4D,0x5C,0x38,0x07,0x3A,0x18,0x52,0x29,0x1A,0x4F,
  0x2F,0x5B,0x29,0x58,0x08,0x26,0x0C,0x12,0x39,0x44,
  0x22,0x4F,0x18,0x58,0x62,0x55,0x0D,0x0C,0x4F,0x43,
  0x1E,0x05,0x50,0x06,0x59,0x32,0x1E,0x4D,0x20,0x41,
  0x0F,0x25,0x1D,0x03,0x35,0x05,0x26,0x39,0x63,0x49,
  0x62,0x0E,0x54,0x04,0x3D,0x2B,0x30,0x4C,0x48,0x05,
```

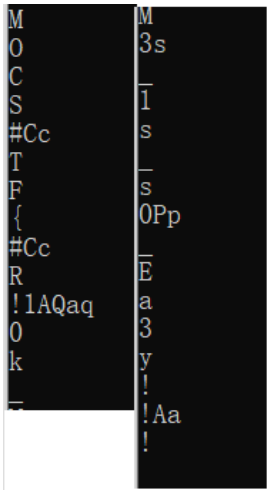
```
  0x4A,0x35,0x14,0x05,0x28,0x57,0x05,0x26,0x18,0x24,
  0x22,0x5E,0x15,0x4B,0x54,0x3E,0x19,0x42,0x46,0x1F,
  0x44,0x25,0x32,0x25,0x14,0x1C,0x01,0x18,0x17,0x4B,
  0x0C,0x3F,0x42,0x5C,0x14,0x30,0x33,0x15,0x23,0x45,
  0x19,0x4E,0x2F,0x20,0x3B,0x49,0x1F,0x23,0x52,0x24,
  0x44,0x3D,0x0F,0x2B,0x4D,0x1C,0x18,0x5F,0x45,0x2D,
  0x19,0x16,0x08,0x61,0x1B,0x40,0x29,0x3F,0x30,0x51,
  0x30,0x1D,0x0E,0x54,0x3C
};

char encode(int input,int i){
    input ^= random_list[i*5];
    input += random_list[i*5+1];
    input *= random_list[i*5+2];
    input -= random_list[i*5+3];
    input %= 66666;
    return input&0xFF;
}

void brute(){
    for(int i=0;i<31;i++){
        for(int j=33;j<125;j++){
            if(encode(j,i) == enflag[i]){
                printf("%c",j);
                //break;
            }
        }
        printf("x");
    }
}

int main(){
    brute();
}
```

我出题出的比较烂，直接多解算了



MOCSCTF{cRa0k_M3_1s_s0_Ea3y!!!}

MOCSCTF{CRa0k_M3_1s_s0_Ea3y!!!}

MOCSCTF{CRA0k_M3_1s_s0_Ea3y!!!}

MOCSCTF{cRA0k_M3_1s_s0_Ea3y!!!}

```
Please close ida, x32dbg and ollydbg first请按任意键继续. . .
Welcome to MOCSCTF, please input your flag:MOCSCTF{cRa0k_M3_1s_s0_Ea3y!!!}
Congratulations! You cracked this software.
```