高校战“疫”网络安全分享赛Writeup——天津垓

原创

[A_whiteCat](#) 于 2020-03-10 17:51:03 发布 714 收藏

分类专栏： [CTF](#) [逆向](#)

本文链接：https://blog.csdn.net/qq_43557177/article/details/104778520

版权

[CTF 同时被 2 个专栏收录](#)

6 篇文章 0 订阅

订阅专栏

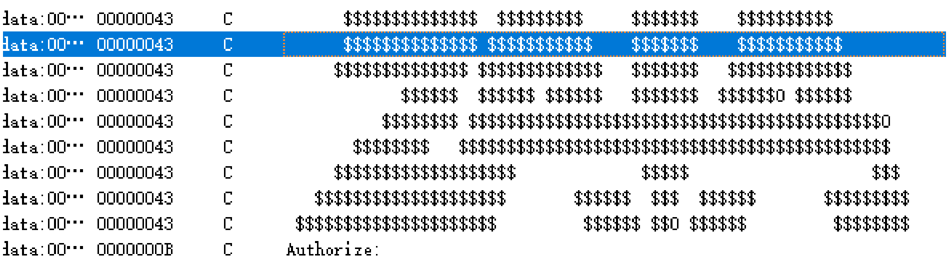
[逆向](#)

5 篇文章 1 订阅

订阅专栏

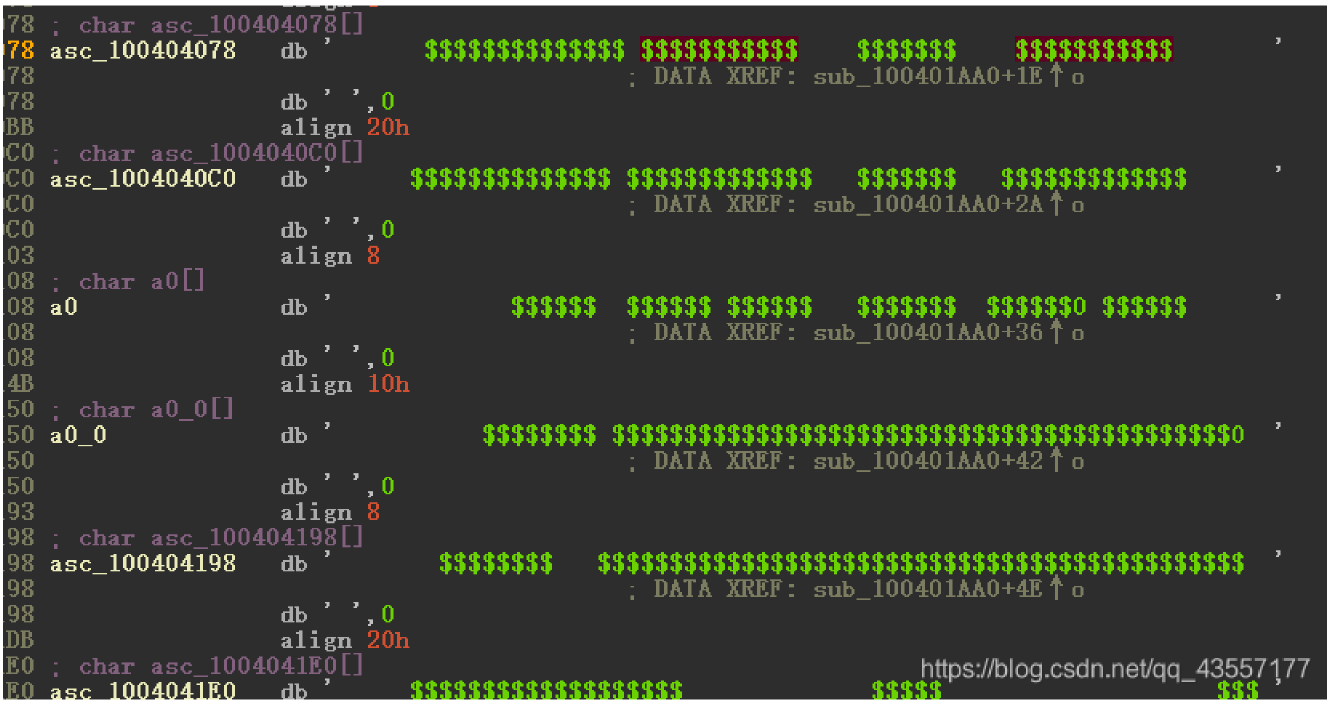
拿到题目，发现由于缺少cygwin1.dll本机并没有办法运行，但是题还是要做的

直接拖进IDA里，进行分析，查找字符串



看起来是有点东西，

双击其中一个



然后按住

x查找引用

Directi	Ty	Address	Text
Up	o	sub_100401AA0+2A	lea rcx, asc_1004040C0; " $$$$$$$$$$$$$$ $$$$$$$$$$$$$ $…

但是目测这个函数并不能帮到我们，继续对函数名称查找引用

```
int sub_100401AA0()
{
  putchar(10);
  puts("      $$$$$$$$$$$$$$  $$$$$$$$$     $$$$$$$     $$$$$$$$$$         ");
  puts("      $$$$$$$$$$$$$$ $$$$$$$$$$$    $$$$$$$     $$$$$$$$$$$        ");
  puts("     $$$$$$$$$$$$$$ $$$$$$$$$$$$$   $$$$$$$    $$$$$$$$$$$$$       ");
  puts("            $$$$$$  $$$$$$ $$$$$$   $$$$$$$   $$$$$$0 $$$$$$       ");
  puts("          $$$$$$$$ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$0   ");
  puts("       $$$$$$$$   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$   ");
  puts("     $$$$$$$$$$$$$$$$$$$                $$$$$                $$$ ");
  puts("   $$$$$$$$$$$$$$$$$$$$       $$$$$$  $$$  $$$$$$       $$$$$$$$$ ");
  puts(" $$$$$$$$$$$$$$$$$$$$          $$$$$$ $$0 $$$$$$         $$$$$$$$ ");
  return putchar(10);
}
```

https://blog.csdn.net/qq_43557177

```
__int64 __fastcall f(__int64 a1,
{
  _main(a1, a2, a3);
  printf("Authorize:");
  sub_1004011F6();
  sub_100401AA0();
  return 0i64;
}
```

进入第一个函数，有点东西了

```
v39 = 'H_gnisiR';
v40 = 'eppo';
v41 = '!r';
v42 = '\0';
v31 = 'eht neh▼';
v32 = 'oh evif ';
v33 = 'sorc snr';
v34 = 'g eht ,s';
v35 = 'os nedlo';
v36 = 'HT reidl';
v37 = 'si RESUO';
v38 = '\n.nrob ';
v25 = 't pmuj A';
v26 = 'ks eht o';
v27 = ' snrut y';
v28 = 'dir a ot';
v29 = '.kcik re';
v30 = '\n';
v21 = 'etneserP';
v22 = 'IAZ yb d';
v23 = '\nA';
v24 = 0;
```

```
9 = 47;
10 = 3;
11 = 47;
12 = 4;
13 = 16;
14 = 72;
15 = 62;
16 = 0;
17 = 7;
18 = 16;
canf(Format, Str);
```

```
if ( strlen(Str) != 18 )

 printf(v20, &v25);
 exit(1);

for ( i = 0; i <= 17; ++i )

 v43 = ~(Str[i] & *((_BYTE *)&v39 + i % 14)) & (Str[i] | *((_BYTE *)&v39 + i % 14));
 if ( v43 != *(&v1 + i) )
 {
   printf(v20, &v25);
   exit(1);
 }

printf(v20, &v31);
```


大体逻辑是将字符串“Rising_Hopper!”作为key，v1到v18作为数组Number，遍历输入Str，满足一个简单逻辑，Python解密核心代码如下

```
Key=list('Rising_Hopper!')
Number=[17,8,6,10,15,20,42,59,47, 3,47, 4,16,72,62,0,7,16]
Result=[]
for i in range(18):
  for n in range(255):
    if Number[i]==~((n & ord(Key[i % 14]))) & (n |ord(Key[i % 14])):
      Result.append(chr(n))
      break
print("".join(Result))
```

运行结果：`Caucasus@s_ability`

~~然后就愉快的提交flag！发现长得这样像flag的东西居然不是flag…~~

后面的参考星盟安全的大佬们发布的Writeup自己复现了一下

查找Str的引用：

```
Up  o  sub_1004011F6+1C4   lea  rdx, Str
Up  o  sub_1004011F6+1D3   lea  rax, Str
Up  o  sub_1004011F6:loc_1… lea  rdx, Str
Up  o  sub_1004011F6+258   lea  rdx, Str
D…  o  sub_100401506+13    lea  rax, Str
D…  o  sub_100401A6C+8     lea  rax, Str
```

```
if ( strlen(Str) != 18 )
  exit(1);
if ( !VirtualProtect(lpAddress, v7, 0x40u, &f101dProtect) )
  exit(1);
for ( i = 0; i < v7; ++i )
  *((_BYTE *)lpAddress + i) ^= *(_BYTE *)(i % 18 + v8);
result = VirtualProtect(lpAddress, v7, f101dProtect, &f101dProtect);
if ( !result )
  exit(1);
```

发现有个函数没见过？？？

```
VirtualProtect([要修改内容的起始地址],[修改的长度],0x40(表示修改为可读写),....)
```

大概意思是要对某块数据要开始修改，以输入的Str作为key，让地址指向的内容与key的[i%18]进行异或

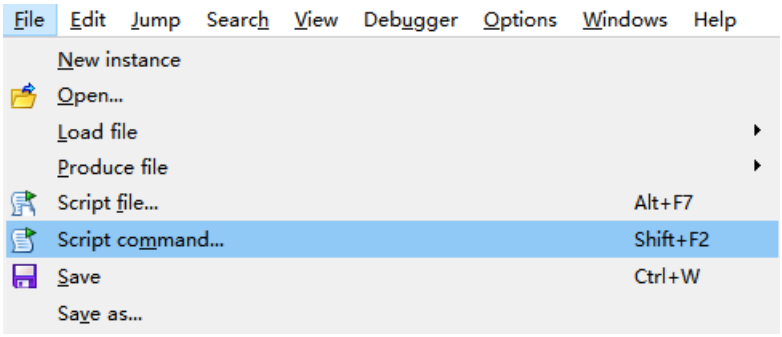
```
.text:000000010040164D byte_10040164D   db 16h, 29h, 0F4h       ; CODE XREF: sub_100401A6C+28↓p
.text:000000010040164D                                          ; DATA XREF: sub_100401A6C+17↓o ...
.text:0000000100401650                  dq 0F3FE08737572918Fh, 0D3FCB3696C69E245h, 87736BD113637561h
.text:0000000100401650                  dq 5406A8696261CBF6h, 76736163EDE48479h, 6C69FEE498735D51h
.text:0000000100401650                  dq 0C1E6B2615D962169h, 0A5617B8251737573h, 6C5B3F797469C8ECh
.text:0000000100401650                  dq 67734073DDF6A663h, 4379D8ECAB697D5Dh, 0F0F6B27340351861h
.text:0000000100401650                  dq 0B3694F4ADD615F73h, 57FA98637561F7FCh, 6F696261E7F68773h
.text:0000000100401650                  dq 6163C9E48479694Bh, 0A2E49873617AFF73h, 0B2615D71D8696C69h
.text:0000000100401650                  dq 7D4E56737573A5E6h, 58797469A4ECA561h, 4073B9F6A6636AE8h
.text:0000000100401650                  dq 0A4ECAB694111FD73h, 0B2737F36FA614379h, 4E5474615F7394F6h
.text:0000000100401650                  dq 5C6375619BFCB369h, 626183F6877369FBh, 95E484796BE07769h
.text:0000000100401650                  dq 987362FA8C736163h, 5FF149696C6986E4h, 3373757389E6B261h
.text:0000000100401650                  dq 746980ECA56151C4h, 85F6A6637B0BD379h, 0AB696C0BCF734073h
.text:0000000100401650                  dq 6F09E561437980ECh, 0AD615F73B8F6B273h, 7561BFFCB3696775h
.text:0000000100401650                  dq 5FF6877369FB5C63h, 847956E095696260h, 62A5A973616271E4h
.text:0000000100401650                  dq 62696C686AE49873h, 75726DE6B2616144h, 7CECA5617E252D73h
.text:0000000100401650                  dq 0A6635468C9797468h, 7C8E0A73407261F6h, 0FA6143786CECAB69h
.text:0000000100401650                  dq 5F725CF6B2737F36h, 63FCB3694E547461h, 877369FB5C637560h
.text:0000000100401650                  dq 56BFB06962607BF6h, 4D7361625DE48479h, 6C684EE498735F4Fh
.text:0000000100401650                  dq 51E6B2615E5B7769h, 0A5617E7ACA737572h, 69E97E79746858ECh
.text:0000000100401650                  dq 717340724DF6A663h, 437848ECAB6946C5h, 0F6B2737F36FA61h
.text:0000000100401650                  dq 0B3694E5474615F72h, 54250C63756007FCh, 8B69626017F68773h
.text:0000000100401650                  dq 616239E4847979EAh, 32E4987361251873h, 0B2616270FE696C68h
.text:0000000100401650                  dq 4126CF73757235E6h, 9479746834ECA561h, 301D3CCB296350EBh
.text:0000000100401650                  dq 11D32401164120B06h, 3D074104140D2559h, 243137E8172305FAh
.text:0000000100401650                  dq 2021F082B591BD1h, 4204370760C93D18h, 3D0106F03C000E08h
.text:0000000100401650                  dq 2B1A2CCB3D1B34EAh, 0BC33C490A064218h, 81D3C530F061108h
.text:0000000100401650                  dq 3C1139E02A111AFAh, 1014080F190437C1h, 6C6962E1DAFA081Dh
.text:0000000100401650                  dq 400756143F1F1AEh, 50E2D23F83B7549h, 3E41F931111A051Bh
.text:0000000100401650                  dq 0D63B2E1C16531806h, 0FB315C3CE5214224h, 605D11160C111C07h
.text:0000000100401650                  dq 5410080080733E53Bh, 0FC3B5126FC292C0Dh, 91B0041E73B7826h
.text:0000000100401650                  dq 2126FC2949571F08h, 32BE73B403B30B5h, 0CF29631E1A00070Ah
.text:0000000100401650                  dq 553611814161321h, 4B2CFD216C2CEB29h, 2E3655120802CD29h
.text:0000000100401650                  dq 1D1B1C1BD8293A07h, 6536E82B754F260Ah, 499227A64726C93Bh
.text:0000000100401650                  dq 7638A24850A4D5Ch, 98249900658B30B4h, 34137163AD12469h
.text:0000000100401650                  dq 989905FA3D1A1306h, 841F100C050F9024h, 411BF0B461428324h
.text:0000000100401650                  dq 0F921EC69626A5F73h, 776D89A2FC29133Ch, 0E121C234D23B4073h
.text:0000000100401650                  dq 635D9DA0CA318F2Ch, 0EB297F36CD3B7573h, 0F8294379766B84A8h
.text:0000000100401650                  dq 5D45A8B2FC3BC126h, 550D4791EF216261h, 0A8B2FC3B8B26F829h
.text:0000000100401650                  dq 74696CD062615E97h, 0F0B4616374DBAB79h, 6C692E825F734117h
.text:0000000100401650                  dq 616375614215F1AEh, 630DDAF818987573h, 7AC1463DC2666C69h
.text:0000000100401650                  dq 5F7224F6DA7CA1D5h, 0F68E74696C69D861h, 4113E0FA61637409h
.text:0000000100401650                  dq 74696D05E7EA5F73h, 4C736163E5E4C7F2h, 247F16615F7220F6h
.text:0000000100401650                  dq 1C8BB4E80B9331E4h, 62615F73F9737572h, 1[illegible]
.text:0000000100401650                  dq 5E1FFDE074736162h, 0ABF03CE61A5B6261h, 4073742B89A2EC29h
```

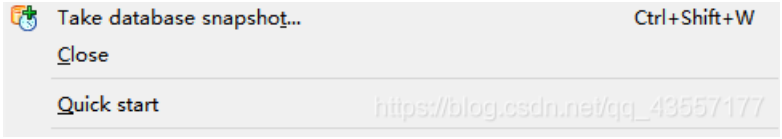

~~这就是所谓的代码加密吧，瑟瑟发抖~~

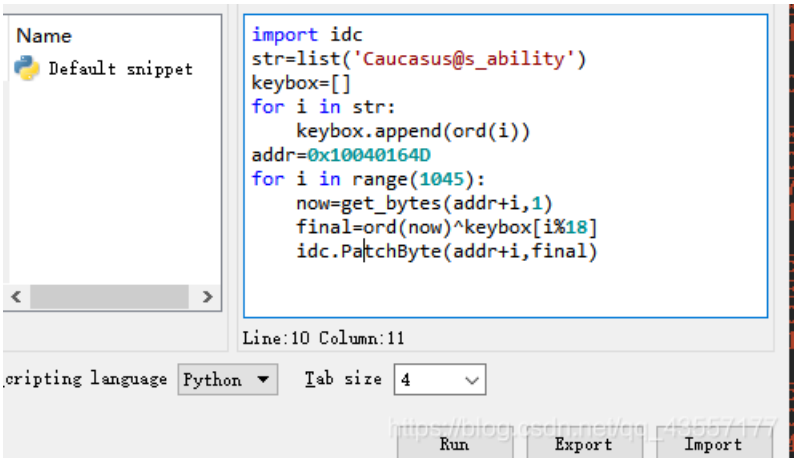
执行Python脚本，

```
import idc
str=list('Caucasus@s_ability')
keybox=[]
for i in str:
    keybox.append(ord(i))
addr=0x10040164D
for i in range(1045):
    now=get_bytes(addr+i,1)
    final=ord(now)^keybox[i%18]
    idc.PatchByte(addr+i,final)
```

新手引导：



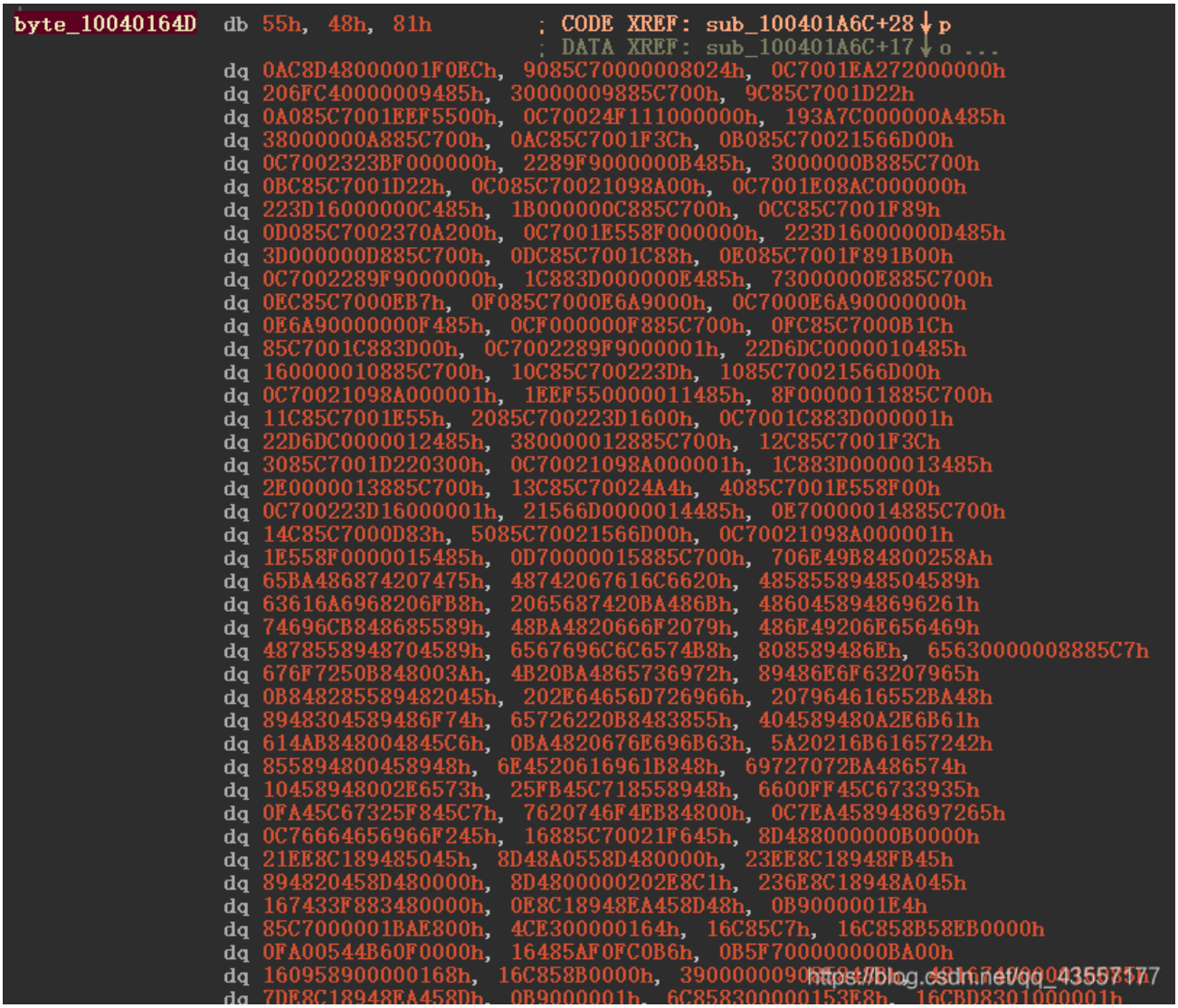




选择Python，放到里面运行即可，IDAPython的常见用法

可参看最近写的另一篇博客

解密后：



依然看不懂，对吧？

选中0x10040164D至0x100401A68,右键，选择Analyese Selected area

```
00100040164D
00100040164D loc_10040164D:                            ; CODE XREF: sub_100401A6C+28
00100040164D                                           ; DATA XREF: sub_100401A6C+17
00100040164D                push    rbp
00100040164E                sub     rsp, 1F0h
00100401655                 lea     rbp, [rsp+80h]
00100040165D                mov     dword ptr [rbp+90h], 1EA272h
00100401667                 mov     dword ptr [rbp+94h], 206FC4h
00100401671                 mov     dword ptr [rbp+98h], 1D2203h
0010040167B                 mov     dword ptr [rbp+9Ch], 1EEF55h
00100401685                 mov     dword ptr [rbp+0A0h], 24F111h
0010040168F                 mov     dword ptr [rbp+0A4h], 193A7Ch
00100401699                 mov     dword ptr [rbp+0A8h], 1F3C38h
001004016A3                 mov     dword ptr [rbp+0ACh], 21566Dh
001004016AD                 mov     dword ptr [rbp+0B0h], 2323BFh
001004016B7                 mov     dword ptr [rbp+0B4h], 2289F9h
001004016C1                 mov     dword ptr [rbp+0B8h], 1D2203h
001004016CB                 mov     dword ptr [rbp+0BCh], 21098Ah
001004016D5                 mov     dword ptr [rbp+0C0h], 1E08ACh
001004016DF                 mov     dword ptr [rbp+0C4h], 223D16h
001004016E9                 mov     dword ptr [rbp+0C8h], 1F891Bh
001004016F3                 mov     dword ptr [rbp+0CCh], 2370A2h
001004016FD                 mov     dword ptr [rbp+0D0h], 1E558Fh
00100401707                 mov     dword ptr [rbp+0D4h], 223D16h
00100401711                 mov     dword ptr [rbp+0D8h], 1C883Dh
0010040171B                 mov     dword ptr [rbp+0DCh], 1F891Bh
00100401725                 mov     dword ptr [rbp+0E0h], 2289F9h
0010040172F                 mov     dword ptr [rbp+0E4h], 1C883Dh
00100401739                 mov     dword ptr [rbp+0E8h], 0D8A90h
00100401743                 mov     dword ptr [rbp+0ECh], 0E6A90h
```


选中函数名Create Function，

```
0164D sub_10040164D   proc near                ; CODE XREF:
0164D                                          ; DATA XREF:
0164D
0164D Str             = byte ptr -1D0h
0164D var_186         = byte ptr -186h
0164D var_17E         = dword ptr -17Eh
0164D var_17A         = word ptr -17Ah
0164D var_178         = word ptr -178h
0164D var_176         = byte ptr -176h
0164D var_175         = byte ptr -175h
0164D var_171         = byte ptr -171h
0164D var_170         = byte ptr -170h
0164D var_168         = qword ptr -168h
0164D var_160         = qword ptr -160h
0164D var_158         = qword ptr -158h
0164D var_150         = byte ptr -150h
0164D var_148         = qword ptr -148h
0164D var_140         = qword ptr -140h
0164D var_138         = qword ptr -138h
0164D var_130         = qword ptr -130h
0164D var_128         = byte ptr -128h
0164D Format          = byte ptr -120h
```


按下Tab获得伪代码

```
v58 = 1987983;
v59 = 2460375;
strcpy(Format, "Input the flag to hijack the ability of Hiden Intelligence:");
strcpy(v7, "Progrise Key confirmed. Ready to break.\n");
strcpy(v6, "Jacking Break! Zaia Enterprise.");
strcpy(v5, "%59s");
v3 = 29477;
v4 = 0;
strcpy(v2, "Not verified!");
v62 = -2147483637;
printf(Format);
scanf(v5, Str);
printf(v7);
if ( strlen(Str) != 51 )
{
  printf(v2);
  exit(0);
```

```
}
v61 = 19683;
for ( i = 0; i <= 0x32; ++i )
{
  v60 = v61 * (unsigned int)(unsigned __int8)Str[i] % v62;
  if ( v60 != *(&v9 + i) )
  {
    printf(v2);
```


这个逻辑...比获得Key还要简单...

以v9开头的数组为Number，

```
for i in range(0x33):
        for n in range(255):
            if Number[i] == 19683 * n % 0x8000000B:
                Result.append(c)
                break
```

得到flag：`flag{Thousandriver_is_1000%_stronger_than_zero-one}`
果真还是自己资历尚浅，这次比赛和大佬们学习了