Xctf进阶刷题-1

0x01shuffle

IDA分析即可看到flag

```
IDA View-A    Pseudocode-A    Hex View-1    Structures
52  s = 'S';
53  v11 = 'E';
54  v12 = 'C';
55  v13 = 'C';
56  v14 = 'O';
57  v15 = 'N';
58  v16 = '{';
59  v17 = 'W';
60  v18 = 'e';
61  v19 = 'l';
62  v20 = 'c';
63  v21 = 'o';
64  v22 = 'm';
65  v23 = 'e';
66  v24 = ' ';
67  v25 = 't';
68  v26 = 'o';
69  v27 = ' ';
70  v28 = 't';
71  v29 = 'h';
72  v30 = 'e';
73  v31 = ' ';
74  v32 = 'S';
75  v33 = 'E';
76  v34 = 'C';
77  v35 = 'C';
78  v36 = 'O';
79  v37 = 'N';
80  v38 = ' ';
81  v39 = '2';
```

0x02 reversing-x64elf（python二维数组）

IDA分析关键函数

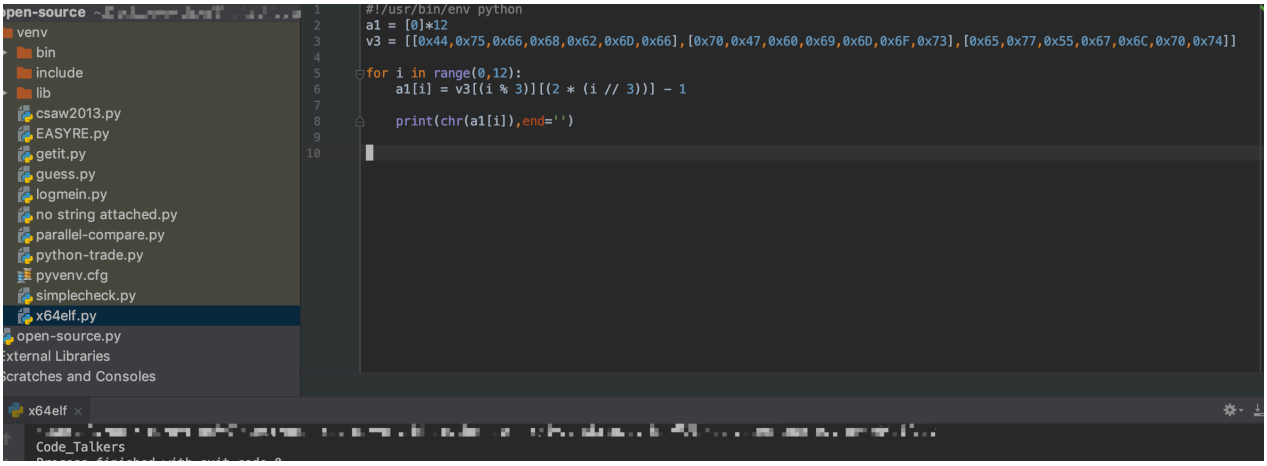
可以看到是对输入的值进行了sub_4006FD函数操作然后进行判定

```
signed __int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  signed __int64 result; // rax
  char s; // [rsp+0h] [rbp-110h]
  unsigned __int64 v5; // [rsp+108h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  printf("Enter the password: ", a2, a3);
  if ( !fgets(&s, 255, stdin) )
    return 0LL;
  if ( (unsigned int)sub_4006FD(&s, 255LL) )
  {
    puts("Incorrect password!");
    result = 1LL;
  }
  else
  {
    puts("Nice!");
    result = 0LL;
  }
  return result;
}
```

进入函数查看，可以看到是一个计算操作

```
signed __int64 __fastcall sub_4006FD(__int64 a1)
{
  signed int i; // [rsp+14h] [rbp-24h]
  const char *v3; // [rsp+18h] [rbp-20h]
  const char *v4; // [rsp+20h] [rbp-18h]
  const char *v5; // [rsp+28h] [rbp-10h]

  v3 = "Dufhbmf";
  v4 = "pG`imos";
  v5 = "ewUglpt";
  for ( i = 0; i <= 11; ++i )
  {
    if ( (&v3)[i % 3][2 * (i / 3)] - *(char *)(i + a1) != 1 )
      return 1LL;
  }
  return 0LL;
}
```

编写脚本即可得flag



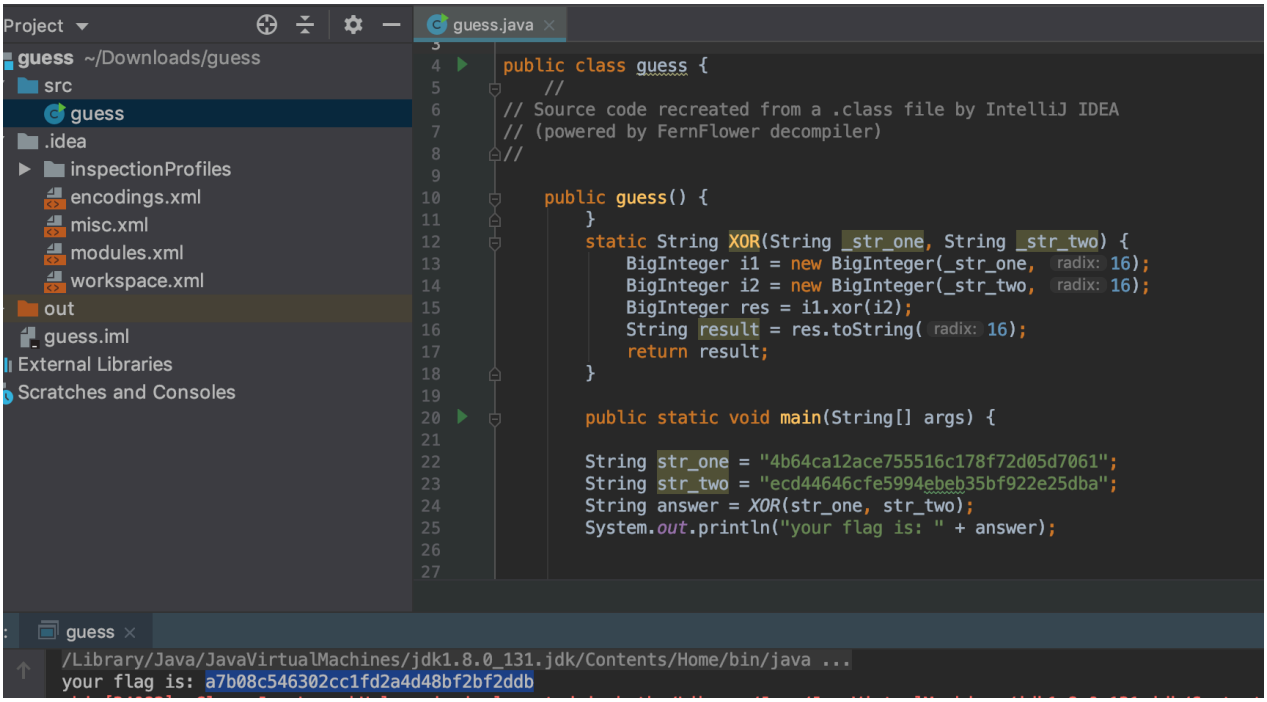
注意：python中二维数组的编写

0x03 guess-the-number（jar包逆向）

是一个java程序，直接解压jar包，查看.class源码，可以看到是对两个字符串进行操作

```java
public static void main(String[] args) {
    int guess_number = false;
    int my_num = 349763335;
    int my_number = 1545686892;
    int flag = 345736730;
    if (args.length > 0) {
        try {
            int guess_number = Integer.parseInt(args[0]);
            if (my_number / 5 == guess_number) {
                String str_one = "4b64ca12ace755516c178f72d05d7061";
                String str_two = "ecd44646cfe5994ebeb35bf922e25dba";
                int var10000 = my_num + flag;
                String answer = XOR(str_one, str_two);
                System.out.println("your flag is: " + answer);
            } else {
                System.err.println("wrong guess!");
                System.exit(1);
            }
        } catch (NumberFormatException var8) {
            System.err.println("please enter an integer \nexample: java -jar guess 12");
            System.exit(1);
        }
    } else {
        System.err.println("wrong guess!");
        int num = 1000000;
```

一开始想要用python编写脚本的，但是xor函数涉及到了Biginter，所以直接改下代码用Java跑



0x04 easyre（python异或）

IDA分析关键函数，输入字符串为v7，长度为24，并且做了以下操作，其中unk_402158是已知的数据

```
sub_401020((int)&unk_402150);
v9 = 0;
v10 = 0;
v7 = '\0';
v8 = '\0';
sub_401050((const char *)&unk_402158, &v7);
v0 = strlen((const char *)&v7);
if ( v0 >= 16 && v0 == 24 )                    // v7长度为24
{
  v1 = 0;
  v2 = (char *)&v8 + 7;
  do
  {
    v3 = *v2--;
    byte_40336C[v1++] = v3;
  }
  while ( v1 < 24 );
  v4 = 0;
  do
  {
    byte_40336C[v4] = (byte_40336C[v4] + 1) ^ 6;
    ++v4;
  }
  while ( v4 < 24 );
  v5 = strcmp(byte_40336C, (const char *)&unk_402124);
  if ( v5 )
    v5 = -(v5 < 0) | 1;
  if ( !v5 )
  {
```

```
sub_401050((const char *)&unk_402158, (unsigned int)&v7);
v0 = strlen((const char *)&v7);
if ( v0 >= 0x10 && v0 == 24 )
{
  v1 = 0;
  v2 = (char *)&v8 + 7;
  do
  {
    v3 = *v2--;
    byte_40336C[v1++] = v3;
  }
  while ( v1 < 24 );
  v4 = 0;
  do
  {
    byte_40336C[v4] = (byte_40336C[v4] + 1) ^ 6;
    ++v4;
  }
  while ( v4 < 0x18 );
  v5 = strcmp(byte_40336C, (const char *)&unk_402124);
  if ( v5 )
    v5 = -(v5 < 0) | 1;
  if ( !v5 )
  {
    sub_401020("right\n", v7);
    system("pause");
  }
}
return 0;
```

编写脚本即可跑出flag

```
#!/usr/bin/env python
unk = [0x78,0x49,0x72,0x43,0x6A,0x7E,0x3C,0x72,0x7C,0x32,0x74,0x57,0x73,0x76,0x33,0x50,0x74,0x49,0x7F
       ,0x7A,0x6E,0x64,0x6B,0x61]
unky = [0]*24
unkyy = [0]*24

for i in range(24):
    unky[i] = (unk[i] ^ 6) - 1
    #print(unky[i])
    unkyy[23-i] = unky[i]

for n in range(24):
    print(chr(unkyy[n]),end='')
```

```
flag{xNqU4otPq3ys9wkDsN}
Process finished with exit code 0
```

注意：Python中异或要加个括号，不加会先运行6-1

0x05 re-for-50-piz（mips）

IDA分析，发现和以往的代码都不太一样，载入的时候IDA也提示是MIPS

参考：https://blog.csdn.net/qq_42967398/article/details/94845416

```
                j       loc_401434
                move    $at, $at
# ---------------------------------------------------------------------------

loc_4013C8:                             # CODE XREF: main+A4↓j
                lui     $v0, 0x4A
                addiu   $v1, $v0, (meow - 0x4A0000)  # "cbtcqLUBChERV[[Nh@_X^D]X_YPV[CJ"
                lw      $v0, 0x28+var_10($fp)
                addu    $v0, $v1, $v0
                lb      $v1, 0($v0)
                lw      $v0, 0x28+arg_4($fp)
                addiu   $v0, 4
                lw      $a0, 0($v0)
                lw      $v0, 0x28+var_10($fp)
                addu    $v0, $a0, $v0
                lb      $v0, 0($v0)
                xori    $v0, 0x37
                sll     $v0, 24
                sra     $v0, 24
                beq     $v1, $v0, loc_401428  # 判定条件: v1 = v0
                move    $at, $at
                lui     $v0, 0x47
                lw      $gp, 0x28+var_18($fp)

loc_401428:                             # CODE XREF: main+68↑j
                lw      $v0, 0x28+var_10($fp)
                addiu   $v0, 1
                sw      $v0, 0x28+var_10($fp)

loc_401434:                             # CODE XREF: main+28↑j
                lw      $v0, 0x28+var_10($fp)
                slti    $v0, 0x1F
                bnez    $v0, loc_4013C8
                move    $at, $at
                lui     $v0, 0x47
                addiu   $a0, $v0, (aC0ngr4ssulatio - 0x470000)  # "C0ngr4ssulations!! U did it."
                la      $v0, puts
                move    $t9, $v0
                jalr    $t9 ; puts
                move    $at, $at
                lw      $gp, 0x28+var_18($fp)
                jal     exit_funct
                move    $at, $at
                lw      $gp, 0x28+var_18($fp)
                li      $v0, 1
                move    $sp, $fp
```

0x06 dmd-50

IDA分析关键函数，要求输入Key的值

```
v43 = __readfsqword(0x28u);
std::operator<<<std::char_traits<char>>(&std::cout, "Enter the valid key!\n", envp);
std::operator>>>char,std::char_traits<char>>(&edata, &v42);
std::allocator<char>::allocator(&v38);
std::string::string(&v39, &v42, &v38);
md5(&v40, &v39);
v41 = (_BYTE *)std::string::c_str((std::string *)&v40);
std::string::~string((std::string *)&v40);
std::string::~string((std::string *)&v39);
std::allocator<char>::~allocator(&v38);
if ( *v41 != '7'
  || v41[1] != '8'
  || v41[2] != '0'
  || v41[3] != '4'
  || v41[4] != '3'
  || v41[5] != '8'
  || v41[6] != 'd'
```

可以看到输出，所以需要进下面的判定

```
  v23 = std::operator<<<std::char_traits<char>>(&std::cout, 'I');
  v24 = std::operator<<<std::char_traits<char>>(v23, 'n');
  v25 = std::operator<<<std::char_traits<char>>(v24, 'v');
  v26 = std::operator<<<std::char_traits<char>>(v25, 'a');
  v27 = std::operator<<<std::char_traits<char>>(v26, 'l');
  v28 = std::operator<<<std::char_traits<char>>(v27, 'i');
  v29 = std::operator<<<std::char_traits<char>>(v28, 'd');
  v30 = std::operator<<<std::char_traits<char>>(v29, ' ');
  v31 = std::operator<<<std::char_traits<char>>(v30, 'K');
  v32 = std::operator<<<std::char_traits<char>>(v31, 'e');
  v33 = std::operator<<<std::char_traits<char>>(v32, 'y');
  v34 = std::operator<<<std::char_traits<char>>(v33, '!');
  v35 = std::operator<<<std::char_traits<char>>(v34, ' ');
  v36 = std::operator<<<std::char_traits<char>>(v35, ':');
  v37 = std::operator<<<std::char_traits<char>>(v36, '(');
  std::ostream::operator<<(v37, &std::endl<char,std::char_traits<char>>);
  result = 0;
}
else
{
  v3 = std::operator<<<std::char_traits<char>>(&std::cout, 'T');
  v4 = std::operator<<<std::char_traits<char>>(v3, 'h');
  v5 = std::operator<<<std::char_traits<char>>(v4, 'e');
  v6 = std::operator<<<std::char_traits<char>>(v5, ' ');
  v7 = std::operator<<<std::char_traits<char>>(v6, 'k');
  v8 = std::operator<<<std::char_traits<char>>(v7, 'e');
  v9 = std::operator<<<std::char_traits<char>>(v8, 'y');
  v10 = std::operator<<<std::char_traits<char>>(v9, ' ');
  v11 = std::operator<<<std::char_traits<char>>(v10, 'i');
```

所以输入要等于41里的值，将41的值拖出来用Md5解密即可

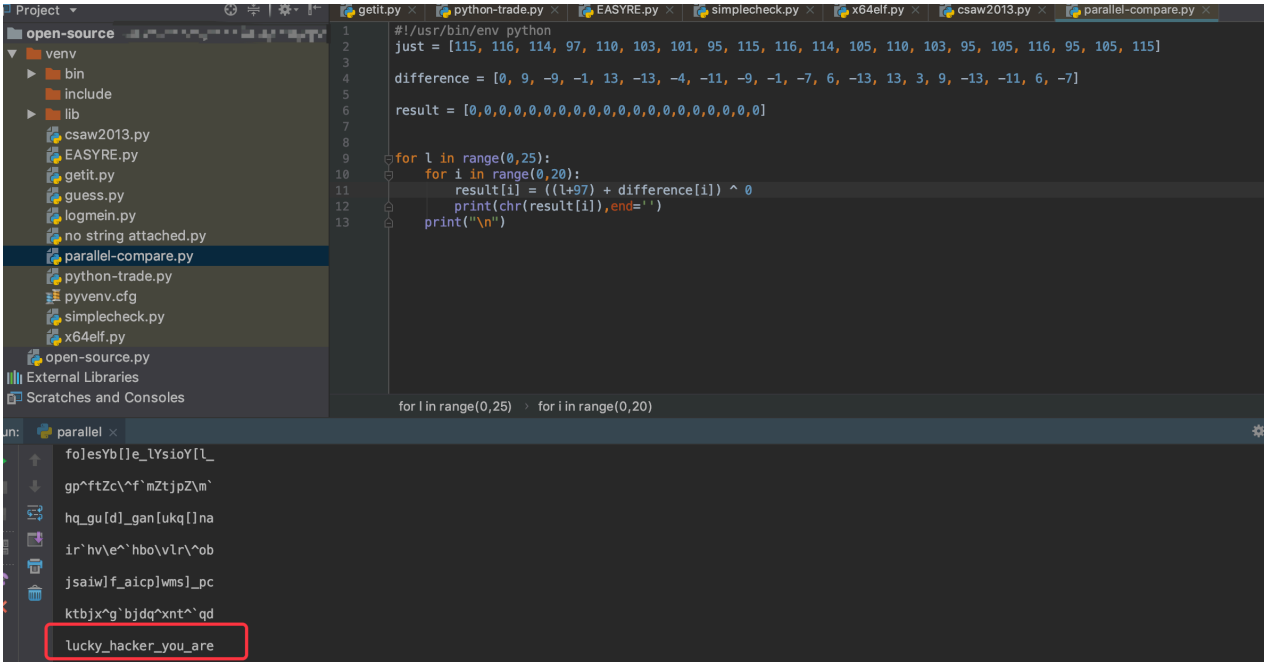
0x07 parallel-comparator-200（爆破）

直接可以看源码分析：

```c
#define FLAG_LEN 20

void * checking(void *arg) {
    char *result = malloc(sizeof(char));
    char *argument = (char *)arg;
    *result = (argument[0]+argument[1]) ^ argument[2];//第一列+第二列和第三列（输入的字符串）异或
    return result;
}

int highly_optimized_parallel_comparsion(char *user_string)
{
```

```c
int highly_optimized_parallel_comparsion(char *user_string)
{
    int initialization_number;
    int i;
    char generated_string[FLAG_LEN + 1];
    generated_string[FLAG_LEN] = '\0';

    while ((initialization_number = random()) >= 64);

    int first_letter;
    first_letter = (initialization_number % 26) + 97;//任意一个大于64的随机数%26+97给first

    pthread_t thread[FLAG_LEN];
    char differences[FLAG_LEN] = {0, 9, -9, -1, 13, -13, -4, -11, -9, -1, -7, 6, -13, 13, 3, 9, -13, -11, 6, -7};
    char *arguments[20];
    for (i = 0; i < FLAG_LEN; i++) {
        arguments[i] = (char *)malloc(3*sizeof(char));
        arguments[i][0] = first_letter;
        arguments[i][1] = differences[i];
        arguments[i][2] = user_string[i];

        arguments[i][2] = user_string[i];

        pthread_create((pthread_t*)(thread+i), NULL, checking, arguments[i]);
    }

    void *result;
    int just_a_string[FLAG_LEN] = {115, 116, 114, 97, 110, 103, 101, 95, 115, 116, 114, 105, 110, 103, 95, 105, 116, 95, 105, 115};
    for (i = 0; i < FLAG_LEN; i++) {
        pthread_join(*(thread+i), &result);
        generated_string[i] = *(char *)result + just_a_string[i];
        free(result);
        free(arguments[i]);
    }

    int is_ok = 1;
    for (i = 0; i < FLAG_LEN; i++) {
        if (generated_string[i] != just_a_string[i])//判定条件generate = just
            return 0;
    }
```

所以编写一个脚本就可以了



注意：由于生成的是随机数，所以需要爆破一下

0x08 secret-galaxy-300（运行时堆栈中）

这个题比较神奇

主函数：看到调用了fill和print

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __main();
  fill_starbase(&starbase);
  print_starbase((int)&starbase);
  return 0;
}
```

fill函数:给一堆变量赋值了

```
void __cdecl fill_starbase(int a1)
{
  signed int i; // [esp+8h] [ebp-10h]
  int v2; // [esp+Ch] [ebp-Ch]

  v2 = 0;
  for ( i = 0; i <= 4; ++i )
  {
    *(_DWORD *)(a1 + 24 * i) = galaxy_name[i];
    *(_DWORD *)(24 * i + a1 + 4) = rand();
    *(_DWORD *)(24 * i + a1 + 8) = 0;
    *(_DWORD *)(24 * i + a1 + 12) = 0;
    *(_DWORD *)(24 * i + a1 + 16) = 24 * (i + 1) + a1;
    *(_DWORD *)(a1 + 24 * i + 20) = v2;
    v2 = 24 * i + a1;
  }
}
```

打印出这些值

```
int __cdecl print_starbase(int a1)
{
  int result; // eax
  const char *v2; // edx
  signed int i; // [esp+1Ch] [ebp-Ch]

  puts("--------------GALAXY DATABASE-------------");
  printf("%10s | %s | %s\n", "Galaxy name", "Existence of life", "Distance from Earth");
  result = puts("-----------------------------------------");
  for ( i = 0; i <= 4; ++i )
  {
    if ( *(_DWORD *)(24 * i + a1 + 8) == 1 )
      v2 = "INHABITED";
    else
      v2 = "IS NOT INHABITED";
    result = printf("%11s | %17s | %d\n", *(_DWORD *)(24 * i + a1), v2, *(_DWORD *)(24 * i + a1 + 4));
  }
  return result;
}
```

动态运行看看：

```
C:\Documents and Settings\Administrator\桌面\task10_x86_89ec72d2e52ccd6a8fef8a
--------------GALAXY DATABASE-------------
Galaxy name | Existence of life | Distance from Earth
-----------------------------------------
   NGS 2366 |  IS NOT INHABITED | 41
  Andromeda |  IS NOT INHABITED | 18467
    Messier |  IS NOT INHABITED | 6334
   Sombrero |  IS NOT INHABITED | 26500
 Triangulum |  IS NOT INHABITED | 19169
```

也没有什么输出也没有什么flag的信息

但是在动态调试的时候在堆栈中意外发现了flag

```
4016F6 call    _fill_starbase
4016FB mov     [esp+10h+var_10], offset _starbase
401702 call    _print_starbase
401707 mov     eax, 0
40170C leave
40170D retn
40170D _main endp
40170D
40170D ; ---------------------------------------------------------------------------
40170E align 10h
401710
401710 ; =============== S U B R O U T I N E =======================================
401710
401710 ; Attributes: bp-based frame
401710
401710 public __setargv
0040170C: _main+2B (Synchronized with EIP)
```

```
EAX 00000000
EBX 7FFDD000  debug007:7FFDD000
ECX 77C118BF  msvcrt.dll:msvcrt_pri
EDX 77C31B78  msvcrt.dll:msvcrt__wp
ESI 00000000
EDI 00000000
EBP 0022FF78  Stack[000009C0]:0022F
ESP 0022FF60  Stack[000009C0]:0022F
EIP 0040170C  _main+2B
EFL 00000202
```

```
84 67 00 00 00 00 00 00 00 00 00 00  00 80 DA 40 00 50 DA 40 00 88 A0 40 00  ..g..........■.@.P.@...@.
4A 00 00 00 00 00 00 00 00 00 00 00  98 DA 40 00 68 DA 40 00 93 A0 40 00 69  .J............@.h.@...@.i
00 00 01 00 00 00 C0 DA 40 00 00 00  00 00 00 00 00 00 00 00 00 00 00 00 00  z........@..............
00 00 00 00 00 00 00 00 00 00 61 6C  69 65 6E 73 5F 61 72 65 5F 61 72 6F 75  ..........aliens_are_arou
64 5F 75 73 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 00 00 00 00 00  nd_us...................
```

```
Stack view
0022FF60  0040DA20  .bss:_starbase
0022FF64  00000000
0022FF68  00020688  debug002:0002068
0022FF6C  8055A4C8
0022FF70  00000018
0022FF74  7FFDD000  debug007:7FFDD000
```

0x09 srm-50（十六进制转字符串）

IDA分析关键函数

```
memset(&Text, 0, 0x100u);
GetDlgItemTextA(hDlg, 1001, &String, 256);
GetDlgItemTextA(hDlg, 1002, v11, 256);
if ( strstr(&String, "@") && strstr(&String, ".") && strstr(&String, ".")[1] && strstr(&String, "@")[1] != 46 )
{
  v28 = xmmword_410AA0;                    // fail
  v29 = 'erul';
  *(_OWORD *)Src = xmmword_410A90;         // suc
  v30 = 46;
  v26 = xmmword_410A80;                    // cess,flag
  v27 = 3830633;
  if ( strlen(v11) != 16                   // 序列号长度为16位
    || v11[0] != 'C'                       // 第一位为C
    || v23 != 'X'                          // 16 X
    || v11[1] != 90                        // 第二位为Z
    || v11[1] + v22 != 155                 // 15 155-90 = 65（A）
    || v11[2] != 57                        // 第三位为9
    || v11[2] + v21 != 155                 // 14 155-57 = 98（b）
    || v11[3] != 'd'                       // 第四位为d
    || v20 != '7'                          // 13 7
    || v12 != 'm'                          // 第五位为m
    || v19 != 'G'                          // 12 G
    || v13 != 113                          // 第六位为q
    || v13 + v18 != 170                    // 11 170-113 = 57（9）
    || v14 != '4'                          // 第七位 4
    || v17 != 'g'                          // 第10位 g
    || v15 != 'c'                          // 第8位 c
    || v16 != '8' )                        // 第9位 8
  {
```

```
  v30 = 46;
  v26 = xmmword_410A80;                    // cess,flag
  v27 = 3830633;
  if ( strlen(v11) != 16                   // 序列号长度为16位
    || v11[0] != 'C'                       // 第一位为C
    || v23 != 'X'                          // 16 X
    || v11[1] != 90                        // 第二位为Z
    || v11[1] + v22 != 155                 // 15 155-90 = 65（A）
    || v11[2] != 57                        // 第三位为9
    || v11[2] + v21 != 155                 // 14 155-57 = 98（b）
    || v11[3] != 'd'                       // 第四位为d
    || v20 != '7'                          // 13 7
    || v12 != 'm'                          // 第五位为m
    || v19 != 'G'                          // 12 G
    || v13 != 113                          // 第六位为q
    || v13 + v18 != 170                    // 11 170-113 = 57（9）
    || v14 != '4'                          // 第七位 4
    || v17 != 'g'                          // 第10位 g
    || v15 != 'c'                          // 第8位 c
    || v16 != '8' )                        // 第9位 8
  {
    strcpy_s(&Text, 0x100u, (const char *)&v28);
  }
  else
  {
    strcpy_s(&Text, 0x100u, Src);
    strcat_s(&Text, 0x100u, v11);          // v11 = CZ9dmq4c8g9G7bAX
  }
```

可以查看到410A80等的值，十六进制转字符串即可，最终可直接看出flag

16进制转换文本 / 文本转16进制

637553206E6F69746172747369676552

字符串转16进制 >>

16进制转字符串 >>

结果互换

全部清空

cuS noitartsigeR

```
.rdata:00410A64 Caption          db 'Registeration',0     ; DATA XREF: DialogFunc+2B8↑o
.rdata:00410A72                  align 10h
.rdata:00410A80 xmmword_410A80   xmmword 2067616C662072756F590A2173736563h
.rdata:00410A80                                           ; DATA XREF: DialogFunc+15C↑r
.rdata:00410A90 xmmword_410A90   xmmword 637553206E6F69746172747369676552h
.rdata:00410A90                                           ; DATA XREF: DialogFunc+14B↑r
.rdata:00410AA0 xmmword_410AA0   xmmword 696166206E6F69746172747369676552h
.rdata:00410AA0                                           ; DATA XREF: DialogFunc+130↑r
```

0x0A simple-check-100（linux动态调试）

IDA分析关键函数，是对输入的Key进行一个check操作，然后会对v8运行interesting_function

```
  v4 = alloca(32);
  v9 = &v7;
  printf("Key: ");
  v6 = v9;
  scanf("%s", v9);
  if ( check_key(v9) )
    interesting_function(&v8);
  else
    puts("Wrong");
  return 0;
}
```

check函数：

```
BOOL __cdecl check_key(int a1)
{
  signed int i; // [esp+8h] [ebp-8h]
  int v3; // [esp+Ch] [ebp-4h]

  v3 = 0;
  for ( i = 0; i <= 4; ++i )
    v3 += *(_DWORD *)(4 * i + a1);
  return v3 == -559038737;
}
```

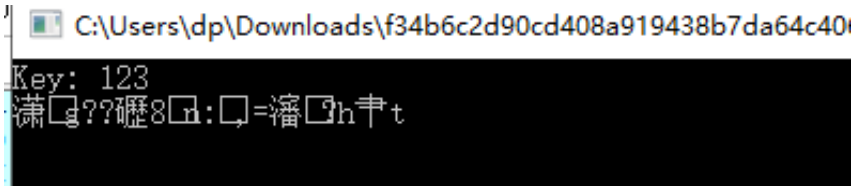
interesting函数：可以看到是v3和flag_data做异或

```
int *__cdecl interesting_function(int a1)
{
  int *result; // eax
  unsigned int v2; // [esp+1Ch] [ebp-1Ch]
  int *v3; // [esp+20h] [ebp-18h]
  int v4; // [esp+24h] [ebp-14h]
  int j; // [esp+28h] [ebp-10h]
  int i; // [esp+2Ch] [ebp-Ch]

  result = (int *)a1;
  v4 = a1;
  for ( i = 0; i <= 6; ++i )
  {
    v2 = *(_DWORD *)(4 * i + v4) ^ 0xDEADBEEF;
    result = (int *)&v2;
    v3 = (int *)&v2;
    for ( j = 3; j >= 0; --j )
      result = (int *)putchar((char)(*((_BYTE *)v3 + j) ^ flag_data[4 * i + j]));
  }
  return result;
}
```

Flagdata：

```
0040A080                 public _flag_data
0040A080 ; char flag_data[28]
0040A080 _flag_data      db 0DCh                 ; DATA XREF: _interesting_function+59↑r
0040A081                 db  17h
0040A082                 db 0BFh
0040A083                 db  5Bh ; [
0040A084                 db 0D4h
0040A085                 db  0Ah
0040A086                 db 0D2h
0040A087                 db  1Bh
0040A088                 db  7Dh ; }
0040A089                 db 0DAh
0040A08A                 db 0A7h
0040A08B                 db  95h
0040A08C                 db 0B5h
```

有点复杂，选择动态调试，想看看直接绕过输出，结果是一堆乱码



最后得知需要在Linux下进行动态调试，即可得到flag

0x0B Mysterious（atoi函数）

IDA分析关键函数，可以看到只要知道v5的内容就可以了

```
  else if ( a2 == 273 )
  {
    if ( a3 == 1000 )
    {
      GetDlgItemTextA(hWnd, 1002, &String, 260);
      strlen(&String);
      if ( strlen(&String) > 6 )                // 输入的字符串长度小于等于6
        ExitProcess(0);
      v10 = atoi(&String) + 1;                  // atoi(输入字符串)=122
      if ( v10 == 123 && v12 == 'x' && v14 == 'z' && v13 == 'y' )
      {
        strcpy(Text, "flag");
        memset(&v7, 0, 0xFCu);
        v8 = 0;
        v9 = 0;
        _itoa(v10, &v5, 10);
        strcat(Text, "{");
        strcat(Text, &v5);
        strcat(Text, "_");
        strcat(Text, "Buff3r_0v3rf|0w");
        strcat(Text, "}");
        MessageBoxA(0, Text, "well done", 0);
      }
      SetTimer(hWnd, 1u, 0x3E8u, TimerFunc);
    }
    if ( a3 == 1001 )
      KillTimer(hWnd, 1u);
  }
  return 0;
}
```

一开始去分析atoi函数了，但是分析了半天没有弄懂，后来发现这是固定的函数，和itoa对应，是字符串如'123'转整数123

char *itoa (int value, char *str, int base);

返回值：返回指向str的指针，无错误返回。

int value 被转换的整数，char *string 转换后储存的字符数组，int radix 转换进制数，如2,8,10,16 进制等，大小应在2-36之间。

所以输入的字符串为'122xyz'，即可得到flag

0x0C newbie_calculations（计算器思路）

看题目知道这是个计算器题

IDA分析，函数调用很复杂

参考：https://www.cnblogs.com/DirWang/p/11586159.html

思路就是：因为没有输入，所以只是对固定数值进行的计算操作，对出现的401100，401000,401220函数进行分析，然后自己用熟悉的语言重写一下，然后复制这段代码运行即可得出flag

```
  v120[1] = 1;
  v121 = 0;
  puts("Your flag is:");
  v3 = sub_401100(v120, 1000000000);
  v4 = (_DWORD *)sub_401220(v3, 999999950);
  sub_401100(v4, 2);
  v5 = sub_401000(&v120[1], 5000000);
  v6 = (int *)sub_401220(v5, 6666666);
  v7 = sub_401000(v6, 1666666);
  v8 = sub_401000(v7, 45);
  v9 = sub_401100(v8, 2);
  sub_401000(v9, 5);
  v10 = sub_401100(&v120[2], 1000000000);
  v11 = (_DWORD *)sub_401220(v10, 999999950);
  v12 = sub_401100(v11, 2);
  sub_401000(v12, 2);
  v13 = sub_401000(&v120[3], 55);
  v14 = (int *)sub_401220(v13, 3);
  v15 = sub_401000(v14, 4);
  sub_401220(v15, 1);
  v16 = sub_401100(&v120[4], 100000000);
  v17 = (_DWORD *)sub_401220(v16, 99999950);
  v18 = sub_401100(v17, 2);
  sub_401000(v18, 2);
  v19 = (_DWORD *)sub_401220(&v120[5], 1);
  v20 = sub_401100(v19, 1000000000);
  v21 = sub_401000(v20, 55);
  sub_401220(v21, 3);
  v22 = sub_401100(&v120[6], 1000000);
  sub_401100(v116, v115);
  sub_401000(_88, _84[0]);
  ((void (__cdecl *)(const char *, signed int))sub_401C7F)("CTF{", 1);
  for ( j = 0; j < 32; ++j )
    sub_401C7F("%c", SLOBYTE(v120[j]));
  sub_401C7F("}\n");
  return 0;
}
```

0x0D re1-100

IDA分析，关键函数：

```
while ( 1 )
{
  printf("Input key : ", argva);
  memset(bufWrite, 0, 0xC8uLL);
  gets(bufWrite, 0LL);
  v4 = strlen(bufWrite);
  v5 = write(pParentWrite[1], bufWrite, v4);
  if ( v5 != strlen(bufWrite) )
    printf("parent - partial/failed write", bufWrite);
  do
  {
    memset(bufParentRead, 0, 0xC8uLL);
    numReada = read(pParentRead[0], bufParentRead, 0xC8uLL);
    v6 = bCheckPtrace || checkDebuggerProcessRunning();
    if ( v6 )
    {
      puts("Wrong !!!\n");
    }
    else if ( !checkStringIsNumber(bufParentRead) )
    {
      puts("Wrong !!!\n");
    }
    else
    {
      if ( atoi(bufParentRead) )
      {
        puts("True");
        if ( close(pParentWrite[1]) == -1 )
          exit(1);
        exit(0);
```

可以看出输入就是bufWrite

```
    if ( numRead == -1 )
      break;
    if ( numRead )
    {
      if ( childCheckDebugResult() )
      {
        responseFalse();
      }
      else if ( bufParentRead[0] == '{' )
      {
        if ( strlen(bufParentRead) == 42 )
        {
          if ( !strncmp(&bufParentRead[1], "53fc275d81", 10uLL) )
          {
            if ( bufParentRead[strlen(bufParentRead) - 1] == '}' )
            {
              if ( !strncmp(&bufParentRead[31], "4938ae4efd", 10uLL) )
              {
                if ( !confuseKey(bufParentRead, 42) )
                {
                  responseFalse();
                }
                else if ( !strncmp(bufParentRead, "{daf29f59034938ae4efd53fc275d81053ed5be8c}", 42uLL) )
                {
                  responseTrue();
                }
                else
                {
```

所以输入字符串的长度为42，第一位和最后一位分别是{}，前十位是53fc275d81,后十位是4938ae4efd，最后经过一个混淆，要等于下面那个字符串

查看混淆的代码，就是分块处理了下各个字符串，换了下位置

```
szPart2[14] = 0;
*(_QWORD *)szPart3 = 0LL;
*(_DWORD *)&szPart3[8] = 0;
*(_WORD *)&szPart3[12] = 0;
szPart3[14] = 0;
*(_QWORD *)szPart4 = 0LL;
*(_DWORD *)&szPart4[8] = 0;
*(_WORD *)&szPart4[12] = 0;
szPart4[14] = 0;
if ( iKeyLength != 42 )
  return 0;
if ( !szKey )
  return 0;
if ( strlen(szKey) != 42 )
  return 0;
if ( *szKey != '{' )
  return 0;
strncpy(szPart1, szKey + 1, 0xAuLL);
strncpy(szPart2, szKey + 11, 0xAuLL);
strncpy(szPart3, szKey + 21, 0xAuLL);
strncpy(szPart4, szKey + 31, 0xAuLL);
memset(szKey, 0, iKeyLength);
*szKey = '{';
strcat(szKey, szPart3);
strcat(szKey, szPart4);
strcat(szKey, szPart1);
strcat(szKey, szPart2);
```

最终换回来就得到flag了

0x0E answer_to_everything

IDA分析关键函数

所以只要输入42，就可以得到一串字符：Cipher from Bill \nSubmit without any tags\n#kdudpeh

```
__int64 __fastcall not_the_flag(int a1)
{
  if ( a1 == 42 )
    puts("Cipher from Bill \nSubmit without any tags\n#kdudpeh");
  else
    puts("YOUSUCK");
  return 0LL;
}
```

根据题目提示sha1加密和不需要加tag，变化一下就可以得到flag了

0x0F elrond32

0x10 tt3441810

0x11 re2-cpp-is-awesome（dword/dd）

IDA看看

```
__int64 v6; // rax
__int64 v7; // rdx
_BYTE *v8; // rax
__int64 i; // [rsp+10h] [rbp-60h]
char v11; // [rsp+20h] [rbp-50h]
char v12; // [rsp+4Fh] [rbp-21h]
__int64 v13; // [rsp+50h] [rbp-20h]
int v14; // [rsp+5Ch] [rbp-14h]

if ( a1 != 2 )
{
  v3 = *a2;
  v4 = std::operator<<<std::char_traits<char>>(&std::cout, "Usage: ", a3);
  v6 = std::operator<<<std::char_traits<char>>(v4, v3, v5);
  std::operator<<<std::char_traits<char>>(v6, " flag\n", v7);
  exit(0);
}
std::allocator<char>::allocator(&v12, a2, a3);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&v11, a2[1], &v12);
std::allocator<char>::~allocator(&v12);
v14 = 0;
for ( i = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::begin(&v11); ; sub_400D7A(&i) )
{
  v13 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::end(&v11);
  if ( !(unsigned __int8)sub_400D3D(&i, &v13) )
    break;
  v8 = (_BYTE *)sub_400D9A(&i);
  if ( *v8 != off_6020A0[dword_6020C0[v14]] )
```

```
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(
  (__int64)&v12,
  (__int64)a2[1],
  (__int64)&v13);
std::allocator<char>::~allocator(&v13);
v15 = 0;
for ( i = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::begin(&v12); ; sub_400D7A(&i) )
{
  v14 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::end((__int64)&v12);
  if ( !sub_400D3D((__int64)&i, (__int64)&v14) )
    break;
  v9 = *(unsigned __int8 *)sub_400D9A((__int64)&i);
  if ( (_BYTE)v9 != off_6020A0[word_6020C0[v15]] )// 看看off_6020A0和dword_6020C0是什么，判定条件
    sub_400B56((__int64)&i, (__int64)&v14, v9);// 输出的是“better luck next time”，说明不是
  ++v15;
}
sub_400B73((__int64)&i, (__int64)&v14, v8);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string((__int64)&v12);
return 0LL;
}
```

这个是主函数

可以看到sub_400B56函数：

```
void __fastcall __noreturn sub_400B56(__int64 a1, __int64 a2, __int64 a3)
{
  std::operator<<<std::char_traits<char>>(&std::cout, "Better luck next time\n", a3);
  exit(0);
}
```

和sub_400B73函数

```
__int64 __fastcall sub_400B73(__int64 a1, __int64 a2, __int64 a3)
{
  return std::operator<<<std::char_traits<char>>(&std::cout, "You should have the flag by now\n", a3);
}
```

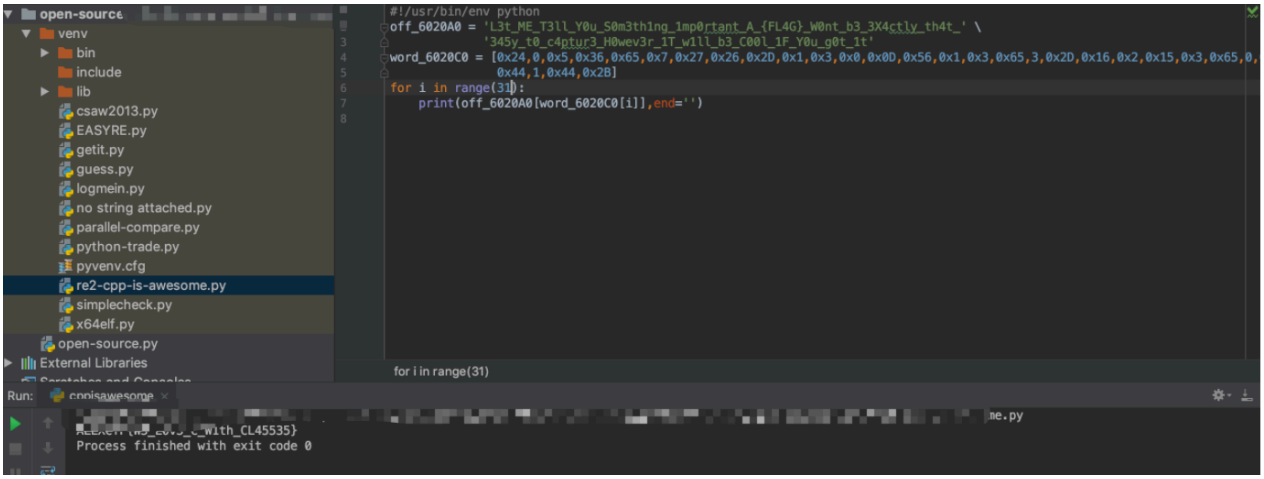
所以我们得知要满足判定条件

我们可以看到off_6020A0和dword_6020C0

```
E                   db    0
F                   db    0
0 off_6020A0        dq offset aL3tMeT3llY0uS0
0                                           ; DATA XREF: main+D1↑r
0                                           ; "L3t_ME_T3ll_Y0u_S0m3th1ng_1mp0rtant_A_{"...
8                   align 20h
0 ; int dword_6020C0[]
0 dword_6020C0      dd 24h                  ; DATA XREF: main+DD↑r
4                   align 8
8                   db    5
9                   db    0
```

```
400E57                  db    0
400E58 aL3tMeT3llY0uS0 db 'L3t_ME_T3ll_Y0u_S0m3th1ng_1mp0rtant_A_{FL4G}_W0nt_b3_3X4ctly_th4t'
400E58                                      ; DATA XREF: .data:off_6020A0↓o
400E58                  db '_345y_t0_c4ptur3_H0wev3r_1T_w1ll_b3_C00l_1F_Y0u_g0t_1t',0
```

那么我们只要编写脚本，满足判定条件即可



注意：dword代表是4个字节一位，align8代表8字节对齐，所以在0x24和5之间是有一个0的，汇编中dd代表4字节，dw代表2字节，db代表1字节

0x12 re4-unvm-me（python3的int()函数和md5加解密）

是个后缀pyc的文件

直接拖进软件解出源码

```
# Embedded file name: unvm_me.py
import md5
md5s = [174282896860968005525213562254350376167L,
 137092044126081477479435678296496849608L,
 126300127609096051658061491018211963916L,
 314989972419727999226545215739316729360L,
 256525866025901597224592941642385934114L,
 115141138810151571209618282728408211053L,
 870597347094265257792933699383906158 2L,
 256697681645515528548061291580728800189L,
 398185526521702743408511442959130915 99L,
 653135619778120180462009978989043133 50L,
 230909080238053318105407334248228870753L,
 196125799557195268866757688147870815374L,
 748741451323455030953072766147279158 85L]
print 'Can you turn me back to python ? ...'
flag = raw_input('well as you wish.. what is the flag: ')
if len(flag) > 69:
    print 'nice try'
    exit()
if len(flag) % 5 != 0:
    print 'nice try'
    exit()
for i in range(0, len(flag), 5):
    s = flag[i:i + 5]
    if int('0x' + md5.new(s).hexdigest(), 16) != md5s[i / 5]:
        print 'nice try'
        exit()

print 'Congratz now you have the flag'
```

只要将md5s中的数字转成十六进制后找md5解密网站解密即可，其中有一个少一位的在首位补上0

注意：#int('0x'+n.hexdigest(),16))是将一个十六进制字符串（添加了'0x'让其变成了十六进制）转成十进制

0x13 流浪者

解开压缩包，是一个exe，运行，发现需要输入字符串然后点击验证按钮

放入IDA分析，在import里寻找getwindowstext函数，果然有

```
004030E0  5163   CWnd::OnWndMsg(uint, uint, long, long *)          MFC42
0040310C  5065   CWnd::OnToolHitTest(CPoint, tagTOOLINFOA *)       MFC42
004030F4  4837   CWnd::OnNotify(uint, long, long *)                MFC42
00403124  4627   CWnd::OnFinalRelease(void)                        MFC42
004030F8  4441   CWnd::OnCommand(uint, long)                       MFC42
004030D4  4407   CWnd::OnChildNotify(uint, uint, long, long *)     MFC42
004030E8  4353   CWnd::OnAmbientProperty(COleControlSite *, ...   MFC42
00403150  4224   CWnd::MessageBoxA(char const *, char const ...   MFC42
004030CC  4078   CWnd::IsFrameWnd(void)                            MFC42
00403158  3874   CWnd::GetWindowTextA(CString &)                   MFC42
004030F0  3798   CWnd::GetSuperWndProcAddr(void)                   MFC42
00403108  3749   CWnd::GetScrollBarCtrl(int)                      MFC42
0040315C  3092   CWnd::GetDlgItem(int)                            MFC42
004030FC  2648   CWnd::EndModalLoop(int)                          MFC42
00403118  2446   CWnd::DestroyWindow(void)                         MFC42
```

找到对应调用的函数，可以看到对输入的字符串进行了一些操作

```
  v1 = (CWnd *)((char *)this + 100);
  v2 = CWnd::GetDlgItem(this, 1002);
  CWnd::GetWindowTextA(v2, v1);
  v3 = sub_401A30((char *)v8 + 100);
  Str = CString::GetBuffer((CWnd *)((char *)v8 + 100), v3);
  if ( !strlen(Str) )
    return CWnd::MessageBoxA(v8, &byte_4035DC, 0, 0);
  for ( i = 0; Str[i]; ++i )                    // 对输入的字符串进行分类操作，转换成其他字符
  {
    if ( Str[i] > '9' || Str[i] < '0' )
    {
      if ( Str[i] > 'z' || Str[i] < 'a' )
      {
        if ( Str[i] > 'Z' || Str[i] < 'A' )
          sub_4017B0();
        else
          v5[i] = Str[i] - 29;
      }
      else
      {
        v5[i] = Str[i] - 87;
      }
    }
    else
    {
      v5[i] = Str[i] - 48;
    }
  }
  return sub_4017F0(v5);
}
```

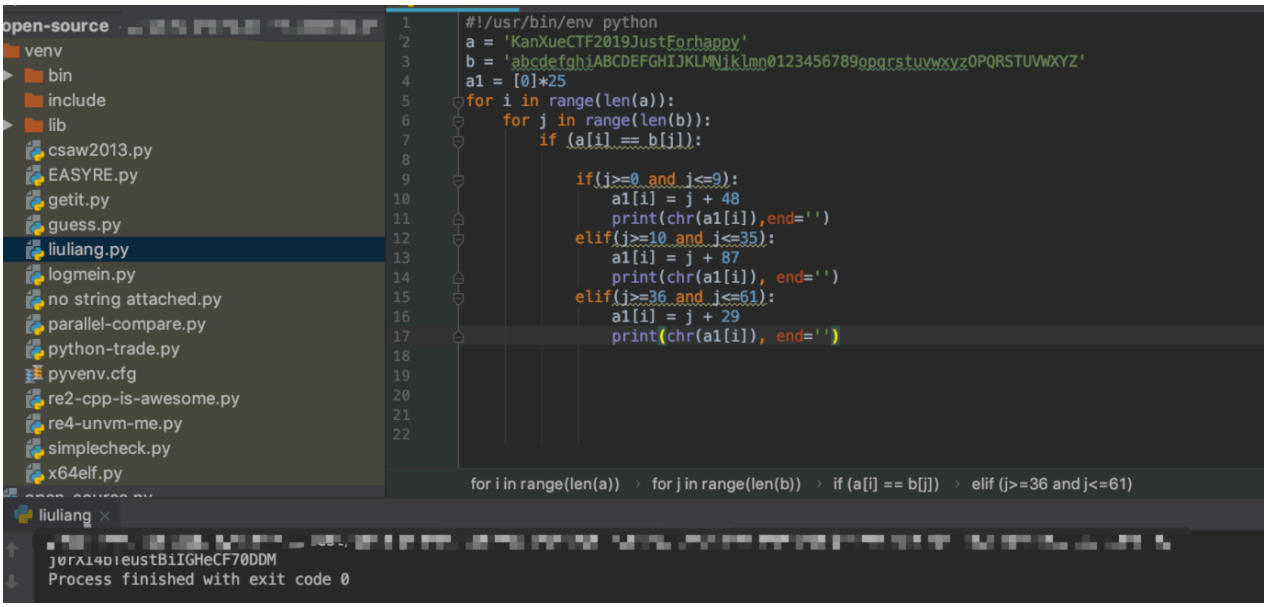
跟下去看sub_4017F0,看到是根据变换后的字符串，然后再和aA这个数组进行变换，因为是dword类型，是四字节一位的，所以a1+4*v4就相当于a1[v4]，变换完后和对应字符串比较

```
int __cdecl sub_4017F0(int a1)
{
  int result; // eax
  char Str1[28]; // [esp+D8h] [ebp-24h]
  int v3; // [esp+F4h] [ebp-8h]
  int v4; // [esp+F8h] [ebp-4h]

  v4 = 0;
  v3 = 0;
  while ( *(_DWORD *)(a1 + 4 * v4) < 62 && *(_DWORD *)(a1 + 4 * v4) >= 0 )
  {
    Str1[v4] = aAbcdefghiabcde[*(_DWORD *)(a1 + 4 * v4)];
    ++v4;
  }
  Str1[v4] = 0;
  if ( !strcmp(Str1, "KanXueCTF2019JustForhappy") )
    result = sub_401770();
  else
    result = sub_4017B0();
  return result;
}
```

aA这个数组如下：

```
.:00403570 byte_403570     db 0BCh                 ; DATA XREF: sub_4017B0+10↑o
.:00403571                 db 0D3h
.:00403572                 db 0D3h
.:00403573                 db 0CDh
.:00403574                 db  21h ; !
.:00403575                 db    0
.:00403576                 db    0
.:00403577                 db    0
.:00403578 ; const CHAR byte_403578
.:00403578 byte_403578     db 0B4h                 ; DATA XREF: sub_4017B0+B↑o
.:00403579                 db 0EDh
.:0040357A                 db 0C1h
.:0040357B                 db 0CBh
.:0040357C                 db  21h ; !
.:0040357D                 db    0
.:0040357E                 db    0
.:0040357F                 db    0
.:00403580 aAbcdefghiabcde db 'abcdefghiABCDEFGHIJKLMNjklmn0123456789opqrstuvwxyzOPQRSTUVWXYZ',0
.:00403580                                         ; DATA XREF: sub_4017F0+21↑o
.:004035BF                 align 10h
.:004035C0 aKanxuectf2019j db 'KanXueCTF2019JustForhappy',0
.:004035C0                                         ; DATA XREF: sub_4017F0+1A↑o
.:004035DA                 align 4
.:004035DC ; char byte_4035DC
.:004035DC byte_4035DC     db 0C7h                 ; DATA XREF: sub_401890+58↑o
```

那么我们就可以编写脚本，最终可得到flag，其中对j判断的0，9，10等数字就是从字符串第一次变换里得到的范围



第一页题目完成: