xctf新手逆向训练刷题

原创

mukami0621 于 2020-10-23 17:21:28 发布 296 收藏

分类专栏： 逆向 基础

本文链接：https://blog.csdn.net/mukami0621/article/details/109245867


逆向 同时被 2 个专栏收录

5 篇文章 0 订阅

订阅专栏


基础

7 篇文章 0 订阅

订阅专栏

xctf新手逆向训练题

这两天在工作之余刷了下xctf的逆向新手题

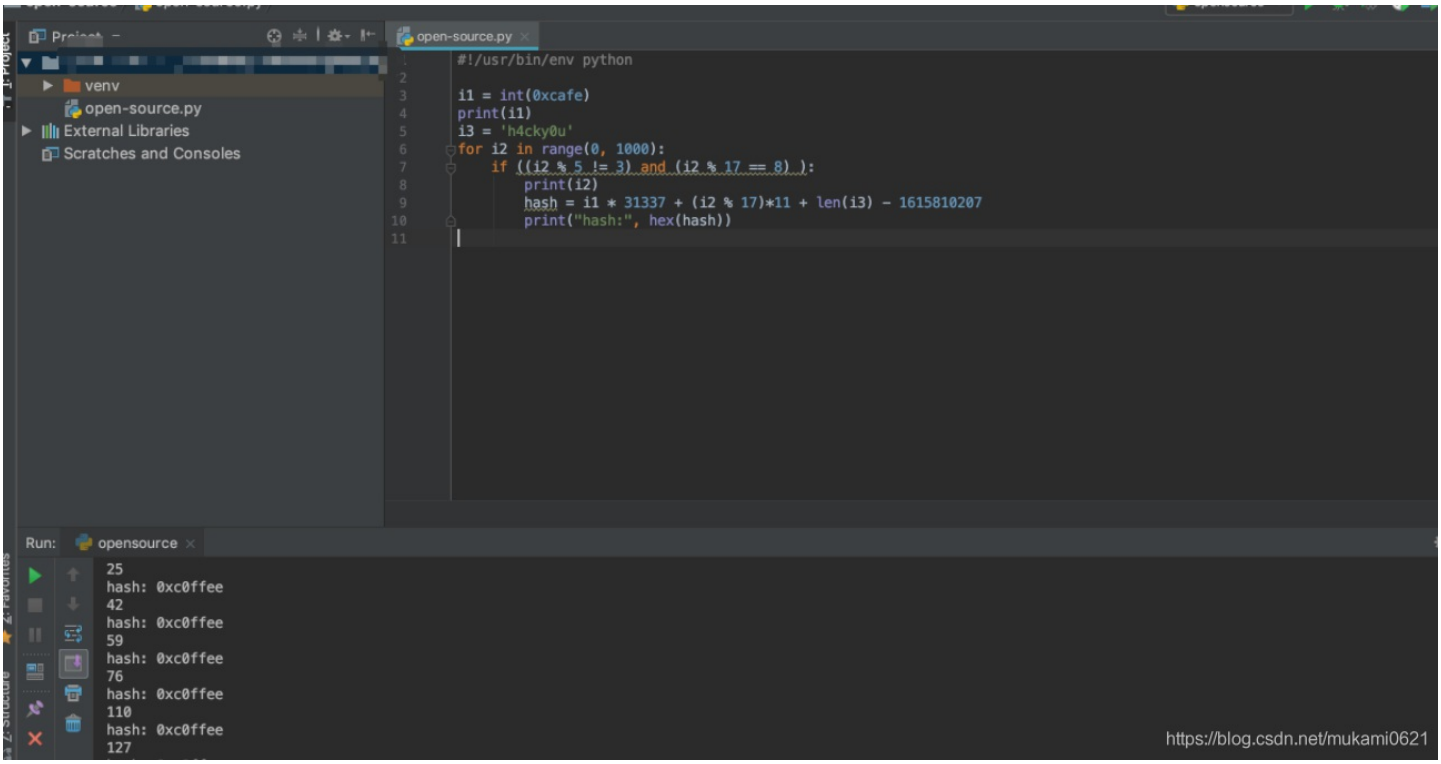
0x1 open-source

下载文件后可以直接看源码

```
int main(int argc, char *argv[]) {
    if (argc != 4) {
        printf("what?\n");
        exit(1);
    }

    unsigned int first = atoi(argv[1]);
    if (first != 0xcafe) {
        printf("you are wrong, sorry.\n");
        exit(2);
    }

    unsigned int second = atoi(argv[2]);
    if (second % 5 == 3 || second % 17 != 8) {
        printf("ha, you won't get it!\n");
        exit(3);
    }

    if (strcmp("h4cky0u", argv[3])) {
        printf("so close, dude!\n");
        exit(4);
    }

    printf("Brr wrrr grr\n");

    unsigned int hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207;

    printf("Get your key: ");
    printf("%x\n", hash);
    return 0;
}
```

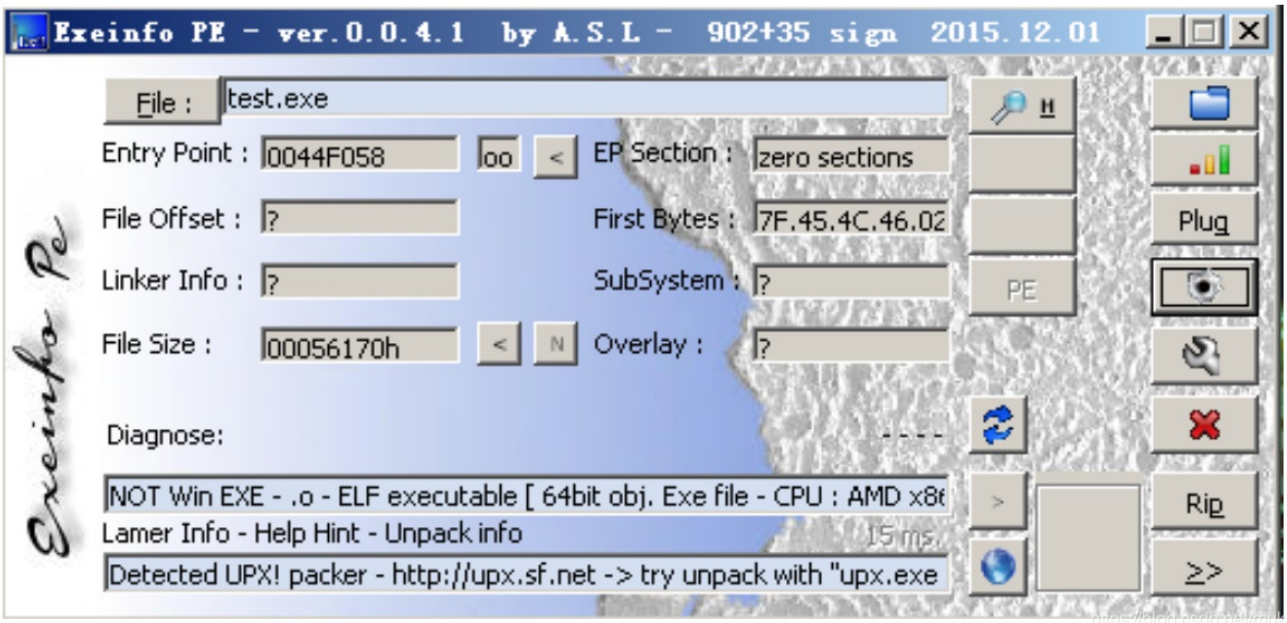

要求是输入4个参数，第一个参数为'0xcafe'，第二个参数是符合判定条件的int值，第三个参数是'h4cky0u'，然后进行一些数据操作得到hash，并用十六进制方式输出（%x）

编写脚本实现，得到flag为0xc0ffee



0x2 simple-unpack

提示是加了壳的二进制文件，查壳



一开始不知道怎么脱二进制文件的壳，参考：https://bbs.pediy.com/thread-157645.htm

到软件中提示的地址下载脱壳程序进行脱壳（https://github.com/upx/upx/releases）

```
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96w       Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    912808 <-    352624   38.63%   linux/amd64   847be14b3e724782b658f2dda2e8045b

Unpacked 1 file.
```


脱壳成功后用IDA查看，即得到flag

```
; __unwind {
                push    rbp
                mov     rbp, rsp
                sub     rsp, 70h
                mov     rax, fs:28h
                mov     [rbp+var_8], rax
                xor     eax, eax
                lea     rax, [rbp+s1]
                mov     rsi, rax
                mov     edi, offset a96s ; "%96s"
                mov     eax, 0
                call    __isoc99_scanf
                lea     rax, [rbp+s1]
                mov     esi, offset flag ; "flag{Upx_1s_n0t_a_d3liv3r_c0mp4ny}"
                mov     rdi, rax        ; s1
                call    _strcmp
                test    eax, eax
                jnz     short loc_4009FC
                mov     edi, offset aCongratulation ; "Congratulations!"
                call    puts
                jmp     short loc_400A06
; ---------------------------------------------------------------------------
```


0x3 logmein

查看下载的文件

```
ELF STX SOH SOH NUL NUL NUL NUL NUL NUL NUL NUL NUL STX NUL > NUL SOH NUL NUL NUL @ ENQ @ N
NUL NUL ETX NUL N NUL NUL NUL DLE NUL NUL NUL u SUB i NUL NUL STX NUL X NUL NUL NUL NUL NUL
 NUL f x9D NUL NUL NUL NUL NUL NUL NUL NUL l 鞘壯^H夤H稞餚TI抢 x9D BS @ NUL H橈 BS @ NUL H
 NUL NUL u DC1 UH夂鑞 xFF xFF xFF ] xC6 ENQ V
 NUL SOH竺 STUS @ NUL xBF SO ` NUL H x83 ? NUL u ENQ屑 STUS NUL xB8 NUL NUL NUL NUL H咿t馬
NUL To continue, you must enter the correct password.
NUL Enter your guess: NUL %32s NUL Incorrect password!
NUL You entered the correct password!
Great job!
NUL NUL SOH ESC ETX ; @ NUL NUL NUL BEL NUL NUL NUL , xFB xFF xFF x8C NUL NUL NUL滬 xFF xFF
ACK NUL NUL NUL NUL NUL NUL NUL FS NUL NUL NUL d NUL NUL NUL P xFD xFF xFF # NUL NUL NUL NU
ACK NUL NUL NUL NUL NUL NUL NUL FS NUL NUL NUL x84 NUL NUL NUL ` xFD xFF xFF # NUL NUL NUL
ACK NUL NUL NUL NUL NUL NUL NUL D NUL NUL NUL xA4 NUL NUL NUL p xFD xFF xFF e NUL NUL NUL N
NUL NUL NUL NUL NUL NUL NUL x94 BS @ NUL NUL NUL NUL NUL EM NUL NUL NUL NUL NUL NUL NUL DI
NUL NUL NUL NUL NUL NUL NUL d NUL NUL NUL NUL NUL NUL NUL VT NUL NUL NUL NUL NUL NUL NUL C
```


用IDA找到对应关键字进行分析

```
0 sub_400760    proc near               ; CODE XREF: main:loc_400ACFp
0
0 var_4         = dword ptr -4
```

```
; __unwind {
                push    rbp
                mov     rbp, rsp
                sub     rsp, 10h
                mov     rdi, offset aYouEnteredTheC ; "You entered the correct password!\nGrea"...
                mov     al, 0
                call    _printf
                xor     edi, edi        ; status
                mov     [rbp+var_4], eax
                call    _exit
; } // starts at 4007F0
sub_4007F0      endp

; ---------------------------------------------------------------------------
                align 20h
```


重点函数：

```
int v9; // [rsp+8Ch] [rbp-4h]

v9 = 0;
strcpy(v8, ":\"AL_RT^L*.?+6/46");
v7 = 'ebmarah';
v6 = 7;
printf("Welcome to the RC3 secure password guesser.\n", a2, a3);
printf("To continue, you must enter the correct password.\n");
printf("Enter your guess: ");
__isoc99_scanf("%32s", s);
v3 = strlen(s);
if ( v3 < strlen(v8) )
  sub_4007C0(v8);
for ( i = 0; i < strlen(s); ++i )
{
  if ( i >= strlen(v8) )
    ((void (*)(void))sub_4007C0)();
  if ( s[i] != (char)(*((_BYTE *)&v7 + i % v6) ^ v8[i]) )
    ((void (*)(void))sub_4007C0)();
}
sub_4007F0();
}
```


编写脚本即可

```
#!/usr/bin/env python
v7 = ['h','a','r','a','m','b','e']
char = ['A', 'a', 'B', 'b', 'C', 'c', 'D', 'd', 'E', 'e', 'F', 'f', 'G', 'g',
        'H', 'h', 'I', 'i', 'J', 'j', 'K', 'k', 'L', 'l', 'M', 'm', 'N', 'n', 'O', 'o', 'P', 'p', 'Q', 'q'
        'R', 'r', 'S', 's', 'T', 't', 'U', 'u', 'V', 'v', 'W', 'w', 'X', 'x', 'Y',
        'y', 'Z', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '-', '_', '(', ')', '{', '}']
v8 = [':', '"', 'A', 'L', '_', 'R','T','^','L','*','.','?','+','6','/','4','6' ]

v9 = ":\"AL_RT^L*.?+6/64"

flag = ''
for i in range(0, 17):
    flag += chr(ord(v7[(i % 7)]) ^ ord(v8[i]))

print(flag)
```


注意：小端模式

0x4 insanity

下载文件后直接编辑器就能查看到flag

0x5 getit

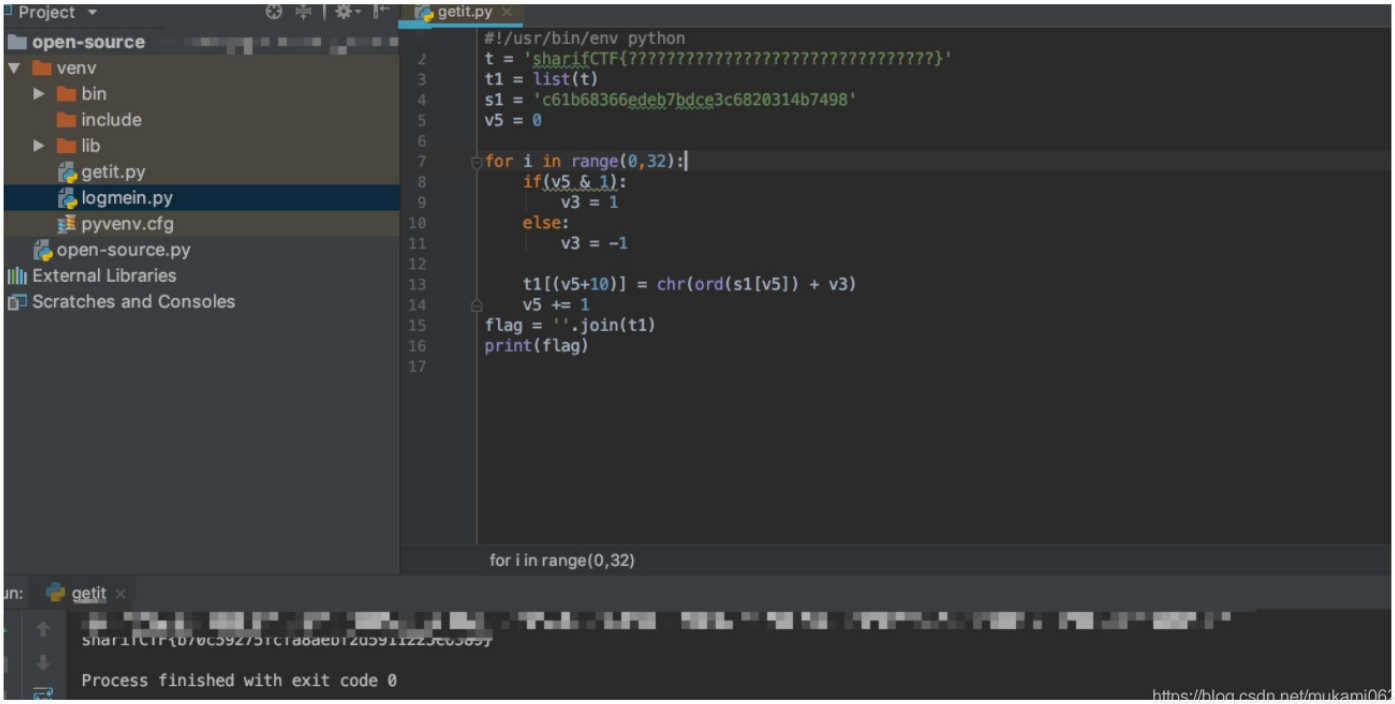
IDA中反编译

```
while ( (signed int)v5 < strlen(s) )
{
  if ( v5 & 1 )
    v3 = 1;
  else
    v3 = -1;
  *(&t + (signed int)v5 + 10) = s[(signed int)v5] + v3;
```

```
19    LODWORD(v5) = v5 + 1;
20  }
21  strcpy(filename, "/tmp/flag.txt");
22  stream = fopen(filename, "w");
23  fprintf(stream, "%s\n", u, v5);
24  for ( i = 0; i < strlen(&t); ++i )
25  {
26    fseek(stream, p[i], 0);
27    fputc(*(&t + p[i]), stream);
28    fseek(stream, 0LL, 0);
29    fprintf(stream, "%s\n", u);
30  }
```


可以看到t就是flag，我们需要知道？的数值，编写脚本即可得到flag

```
6010A0                  public s
6010A0 ; char s[]
6010A0 s                db 'c61b68366edeb7bdce3c6820314b7498',0
6010A0                                  ; DATA XREF: main+25↑o
6010A0                                  ; main+3F↑r
6010C1                  align 20h
6010E0                  public t
6010E0 ; char t
6010E0 t                db 53h          ; DATA XREF: main+65↑w
6010E0                                  ; main+C9↑o ...
6010E1 aHarifctf        db 'harifCTF{??????????????????????????????}',0
60110C                  align 20h
601120                  public u
601120 u                db '*****************************************',0
601120                                  ; DATA XREF: main+A5↑o
601120                                  ; main+13F↑o
60114C                  align 20h
601160                  public p
```

此题注意：python中字符串是个可以直接操作修改的，可以先改为列表，然后重新生成

0x6 python-trade

.pyc后缀的文件，直接用EasyPythonDecompiler解了，获得源码

```
# Embedded file name: 1.py
import base64

def encode(message):
    s = ''
    for i in message:
        x = ord(i) ^ 32
        x = x + 16
        s += chr(x)

    return base64.b64encode(s)


correct = 'XlNkVmtUI1MgXWBZXCFeKY+AaXNt'
flag = ''
print 'Input flag:'
flag = raw_input()
if encode(flag) == correct:
    print 'correct'
else:
    print 'wrong'
```



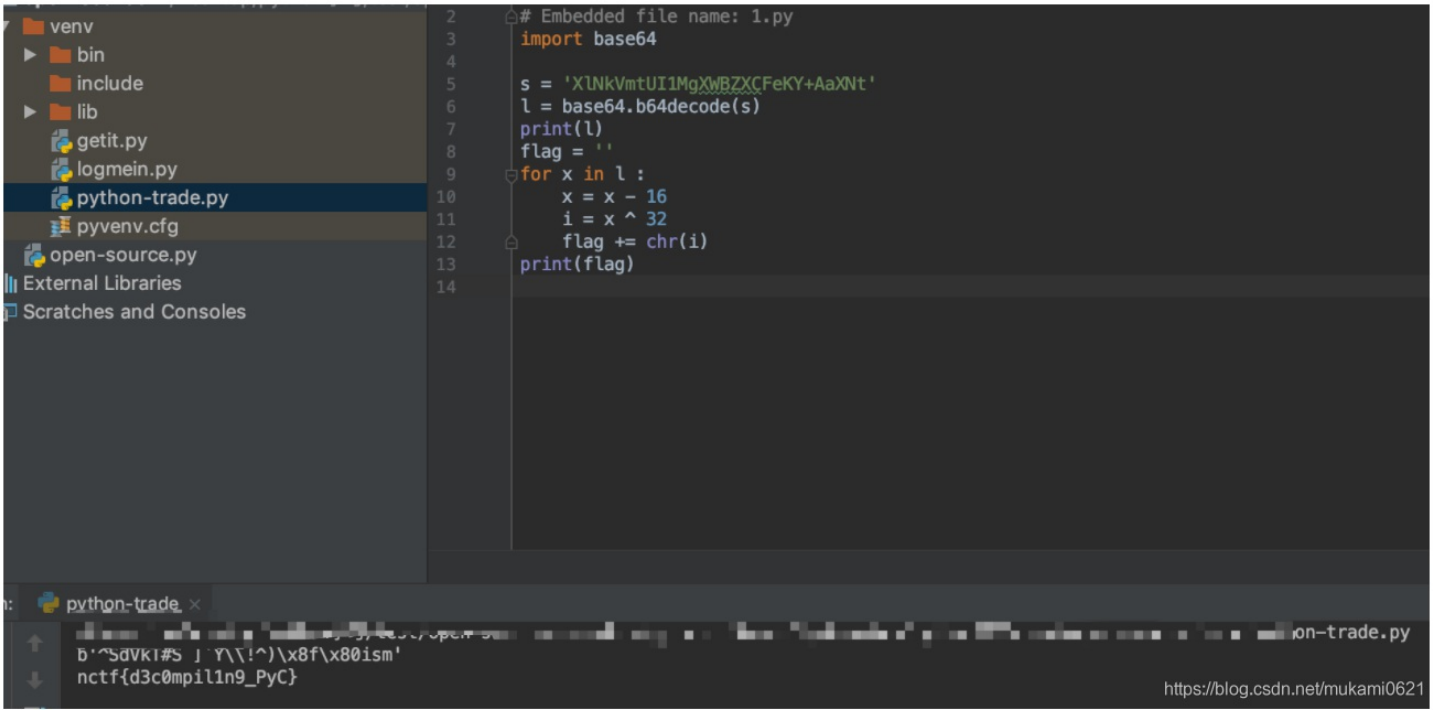
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  __int128 v5; // [esp+0h] [ebp-44h]
  __int64 v6; // [esp+10h] [ebp-34h]
  int v7; // [esp+18h] [ebp-2Ch]
  __int16 v8; // [esp+1Ch] [ebp-28h]
  char v9; // [esp+20h] [ebp-24h]

  _mm_storeu_si128((__m128i *)&v5, _mm_loadu_si128((const __m128i *)&xmmword_413E34));
  v7 = 0;
  v6 = qword_413E44;
  v8 = 0;
  printf(&byte_413E4C);
  printf(&byte_413E60);
  printf(&byte_413E80);
  scanf("%s", &v9);
  v3 = strcmp((const char *)&v5, &v9);
  if ( v3 )
    v3 = -(v3 < 0) | 1;
  if ( v3 )
    printf(aFlag);
```

```
  else
    printf((const char *)&unk_413E90);
  system("pause");
  return 0;
}
```

```
.rdata:00413E2C ; char a1Qnan[]
.rdata:00413E2C a1Qnan          db '1#QNAN',0           ; DATA XREF: _$I10_OUTP
.rdata:00413E33                 align 4
.rdata:00413E34 xmmword_413E34  xmmword '0tem0c1eW{FTCTUD'
.rdata:00413E34                                         ; DATA XREF: _main+10↑r
.rdata:00413E44 qword_413E44    dq '}FTCTUD'            ; DATA XREF: _main+27↑r
.rdata:00413E4C ; char byte_413E4C
.rdata:00413E4C byte_413E4C     db 0BBh                 ; DATA XREF: _main+1A↑o
.rdata:00413E4D                 db 0B6h
```


可以看到操作：对输入的字符串进行自定义的encode操作然后用base64加密

编写脚本解码即得flag



0x7 re1

放进IDA查看

查看主函数，可以看到只要输入的字符串和v5相同就可以了，找到对应v5的值即可得flag

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  __int128 v5; // [esp+0h] [ebp-44h]
  __int64 v6; // [esp+10h] [ebp-34h]
  int v7; // [esp+18h] [ebp-2Ch]
  __int16 v8; // [esp+1Ch] [ebp-28h]
  char v9; // [esp+20h] [ebp-24h]

  _mm_storeu_si128((__m128i *)&v5, _mm_loadu_si128((const __m128i *)&xmmword_413E34));
  v7 = 0;
  v6 = qword_413E44;
  v8 = 0;
  printf(&byte_413E4C);
  printf(&byte_413E60);
  printf(&byte_413E80);
  scanf("%s", &v9);
  v3 = strcmp((const char *)&v5, &v9);
  if ( v3 )
    v3 = -(v3 < 0) | 1;
  if ( v3 )
    printf(aFlag);
  else
    printf((const char *)&unk_413E90);
  system("pause");
  return 0;
}
```

```
:00413E0C aE000           db 'e+000',0          ; DATA XREF: __cftoe2_l:loc_40DEAA↑o
:00413E12                 align 4
:00413E14 a1Snan          db '1#SNAN',0         ; DATA XREF: _$I10_OUTPUT+C7↑o
:00413E1B                 align 4
:00413E1C a1Ind           db '1#IND',0          ; DATA XREF: _$I10_OUTPUT+E0↑o
:00413E22                 align 4
:00413E24 ; char a1Inf[]
:00413E24 a1Inf           db '1#INF',0          ; DATA XREF: _$I10_OUTPUT+EF↑o
:00413E2A                 align 4
:00413E2C ; char a1Qnan[]
:00413E2C a1Qnan          db '1#QNAN',0         ; DATA XREF: _$I10_OUTPUT:loc_40F13C↑o
:00413E33                 align 4
:00413E34 xmmword_413E34  xmmword '0tem0c1eW{FTCTUD'
:00413E34                                       ; DATA XREF: _main+10↑r
:00413E44 qword_413E44    dq '}FTCTUD'          ; DATA XREF: _main+27↑r
:00413E4C ; char byte_413E4C
:00413E4C byte_413E4C     db 0BBh               ; DATA XREF: _main+1A↑o
:00413E4D                 dd 0C0ADD3B6h
:00413E51                 dd 44BD85B4h
:00413E55                 dd 54435455h
:00413E59                 dd 0ACFDF46h
```


0x8 Hello,CTF

用IDA打开，找到主要函数

```
13  char v13; // [esp+4Ch] [ebp-24h]
14
15  strcpy(&v13, "437261636b4d654a757374466f7246756e");
16  while ( 1 )
17  {
18    memset(&v10, 0, 32u);
19    v11 = 0;
20    v12 = 0;
21    sub_40134B(aPleaseInputYou, v6);            // v9 17位
22    scanf(aS, v9);
23    if ( strlen(v9) > 0x11 )
24      break;
25    v3 = 0;
26    do
27    {
28      v4 = v9[v3];
29      if ( !v4 )
30        break;
31      sprintf(&v8, asc_408044, v4);
32      strcat(&v10, &v8);
33      ++v3;
34    }
35    while ( v3 < 17 );
36    if ( !strcmp(&v10, &v13) )
37      sub_40134B(aSuccess, v7);
38    else
```


从下面的循环判定值为17和上面v13的长度为34猜测出两位代表一个字符

十六进制转ASCII即可得flag

0x9 no-string-attached

IDA中查看主函数

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3   setlocale(6, &locale);
4   banner();
5   prompt_authentication();
6   authenticate();
7   return 0;
8 }
```


进入每个函数看看

先看banner()

```
1 int banner()
2 {
3   unsigned int v0; // eax
4
5   v0 = time(0);
6   srand(v0);
7   wprintf(&unk_80488B0);
8   rand();
9   return wprintf(&unk_8048960);
10 }
```


就是打印一些东西，查看对应打印的数值（unk_80488B0和8048960）

```
.rodata:080488AF                db    0
.rodata:080488B0 unk_80488B0    db  57h ; W             ; DATA XREF: banner+1A↑o
.rodata:080488B1                db    0
.rodata:080488B2                db    0
.rodata:080488B3                db    0
.rodata:080488B4                db  65h ; e
.rodata:080488B5                db    0
.rodata:080488B6                db    0
.rodata:080488B7                db    0
.rodata:080488B8                db  6Ch ; l
.rodata:080488B9                db    0
.rodata:080488BA                db    0
.rodata:080488BB                db    0
.rodata:080488BC                db  63h ; c
.rodata:080488BD                db    0
.rodata:080488BE                db    0
.rodata:080488BF                db    0
.rodata:080488C0                db  6Fh ; o
.rodata:080488C1                db    0
.rodata:080488C2                db    0
.rodata:080488C3                db    0
.rodata:080488C4                db  6Dh ; m
.rodata:080488C5                db    0
.rodata:080488C6                db    0
.rodata:080488C7                db    0
.rodata:080488C8                db  65h ; e
.rodata:080488C9                db    0
.rodata:080488CA                db    0
.rodata:080488CB                db    0
```


可以看到是两个字符串，就是一些欢迎的信息之类的

查看下一个，也是打印一些信息，需要你输入字符串

IDA View-A | Pseudocode-B | Pseudocode-A | Hex View-1

```
1 int prompt_authentication()
2 {
3   return wprintf(&unk_80489F8);
4 }
```

```
.rodata:080489F7                db    0
.rodata:080489F8 unk_80489F8    db  50h ; P             ; DATA XREF: prompt_authentication+6↑o
.rodata:080489F9                db    0
.rodata:080489FA                db    0
.rodata:080489FB                db    0
.rodata:080489FC                db  6Ch ; l
.rodata:080489FD                db    0
.rodata:080489FE                db    0
.rodata:080489FF                db    0
.rodata:08048A00                db  65h ; e
.rodata:08048A01                db    0
.rodata:08048A02                db    0
.rodata:08048A03                db    0
.rodata:08048A04                db  61h ; a
.rodata:08048A05                db    0
.rodata:08048A06                db    0
.rodata:08048A07                db    0
.rodata:08048A08                db  73h ; s
.rodata:08048A09                db    0
.rodata:08048A0A                db    0
.rodata:08048A0B                db    0
.rodata:08048A0C                db  65h ; e
.rodata:08048A0D                db    0
.rodata:08048A0E                db    0
.rodata:08048A0F                db    0
.rodata:08048A10                db  20h
```

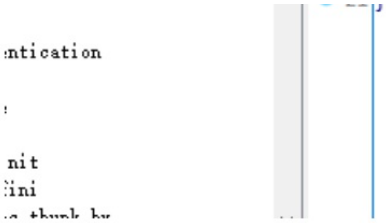

再看下一个，是主函数了，只要输入的函数和s2相同即可，s2是两个输入进行decrypt操作后得到的值

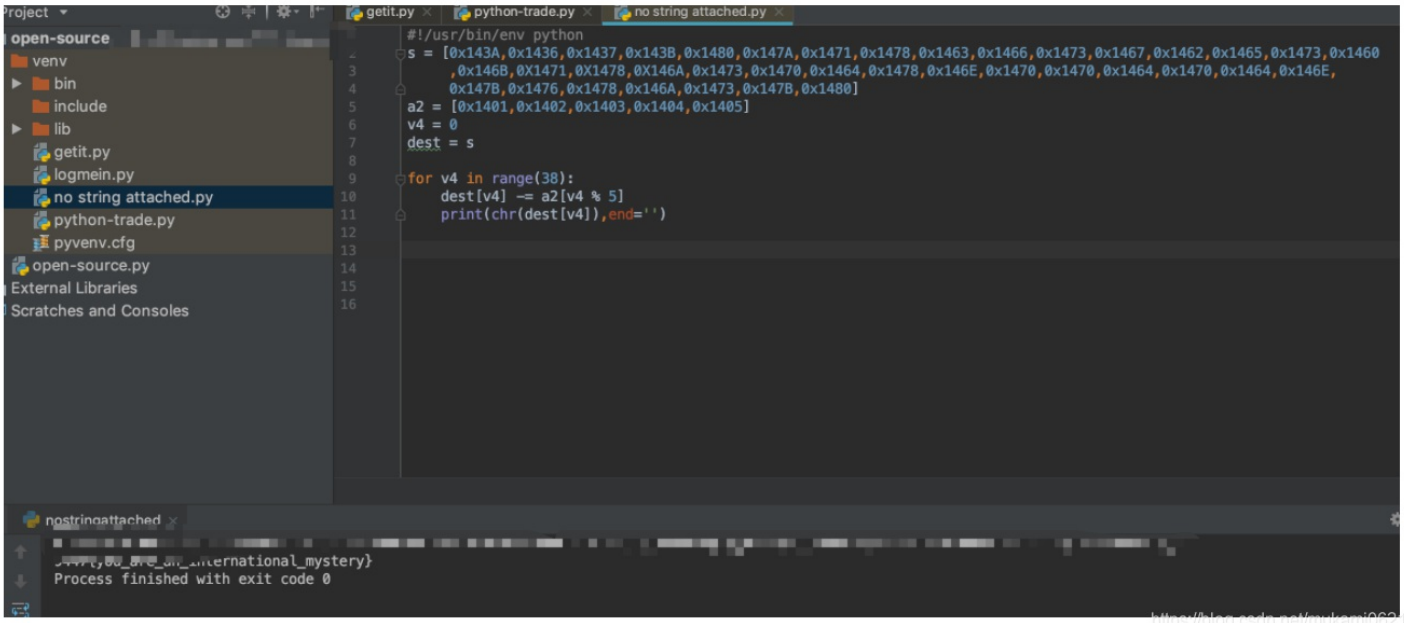
IDA View-A | Pseudocode-B | Pseudocode-A | Hex View-1 | Struc

```
1  void authenticate()
2  {
3    int ws[8192]; // [esp+1Ch] [ebp-800Ch]
4    wchar_t *s2; // [esp+801Ch] [ebp-Ch]
5
6    s2 = decrypt(&word_8048AA8, &word_8048A90);
7    if ( fgetws(ws, 0x2000, stdin) )
8    {
9      ws[wcslen(ws) - 1] = 0;
10     if ( !wcscmp(ws, s2) )
11       wprintf(&unk_8048B44);
12     else
13       wprintf(&unk_8048BA4);
14   }
15   free(s2);
16 }
```


看decrypt,对两个宽字节进行了操作。编写脚本实现功能即可得flag

```
1  wchar_t *__cdecl decrypt(wchar_t *s, wchar_t *a2)
2  {
3    size_t v2; // eax
4    signed int v4; // [esp+1Ch] [ebp-1Ch]
5    signed int i; // [esp+20h] [ebp-18h]
6    signed int v6; // [esp+24h] [ebp-14h]
7    signed int v7; // [esp+28h] [ebp-10h]
8    wchar_t *dest; // [esp+2Ch] [ebp-Ch]
9
10   v6 = wcslen(s);
11   v7 = wcslen(a2);
12   v2 = wcslen(s);
13   dest = (wchar_t *)malloc(v2 + 1);
14   wcscpy(dest, s);
15   while ( v4 < v6 )
16   {
17     for ( i = 0; i < v7 && v4 < v6; ++i )
18       dest[v4++] -= a2[i];
19   }
20   return dest;
21 }
```





0xA csaw2013reversing2

IDA中看主函数

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  int v3; // ecx
  CHAR *lpMem; // [esp+8h] [ebp-Ch]
  HANDLE hHeap; // [esp+10h] [ebp-4h]

  hHeap = HeapCreate(0x40000u, 0, 0);
  lpMem = (CHAR *)HeapAlloc(hHeap, 8u, MaxCount + 1);
  memcpy_s(lpMem, MaxCount, &unk_409B10, MaxCount);
  if ( sub_40102A() || IsDebuggerPresent() )
  {
    __debugbreak();
    sub_401000(v3 + 4, (int)lpMem);
    ExitProcess(0xFFFFFFFF);
  }
  MessageBoxA(0, lpMem + 1, "Flag", 2u);
  HeapFree(hHeap, 0, lpMem);
  HeapDestroy(hHeap);
  ExitProcess(0);
}
```

通过运行程序，发现并不会进入if的判定。

lpMem+1就是我们需要知道的乱码消息

2020/04/02 • 2分钟阅读 • +2

在缓冲区之间复制字节。这些是memcpy，wmemcpy的版本，具有CRT的“安全功能”中所述的安全性增强功能。

句法

C 复制

```
errno_t memcpy_s(
   void *dest,
   size_t destSize,
   const void *src,
   size_t count
);
errno_t wmemcpy_s(
   wchar_t *dest,
   size_t destSize,
   const wchar_t *src,
   size_t count
);
```


可以知道通过memcpy_s操作将unk_409B10的数据给了lpMem

```
a:00409B10 byte_409B10     db 0BBh                 ; DATA XREF: _main
a:00409B11                 db 0CCh
a:00409B12                 db 0A0h
a:00409B13                 db 0BCh
a:00409B14                 db 0DCh
a:00409B15                 db 0D1h
a:00409B16                 db 0BEh
a:00409B17                 db 0B8h
a:00409B18                 db 0CDh
a:00409B19                 db 0CFh
a:00409B1A                 db 0BEh
a:00409B1B                 db 0AEh
a:00409B1C                 db 0D2h
a:00409B1D                 db 0C4h
a:00409B1E                 db 0ABh
a:00409B1F                 db  82h
a:00409B20                 db 0D2h
a:00409B21                 db 0D9h
a:00409B22                 db 93h
a:00409B23                 db 0B3h
a:00409B24                 db 0D4h
a:00409B25                 db 0DEh
a:00409B26                 db  93h
a:00409B27                 db 0A9h
a:00409B28                 db 0D3h
a:00409B29                 db 0CBh
a:00409B2A                 db 0B8h
a:00409B2B                 db  82h
```

查看sub_401000

```
1  unsigned int __fastcall sub_401000(int a1, int a2)
2  {
3    int v2; // esi
4    unsigned int v3; // eax
5    unsigned int v4; // ecx
6    unsigned int result; // eax
7
8    v2 = dword_409B38;
9    v3 = a2 + 1 + strlen((const char *)(a2 + 1)) + 1;
10   v4 = 0;
11   result = ((v3 - (a2 + 2)) >> 2) + 1;
12   if ( result )
13   {
14     do
15       *(_DWORD *)(a2 + 4 * v4++) ^= v2;
16     while ( v4 < result );
17   }
18   return result;
19 }
```


是对lpMem进行了一些操作的，猜测是由于并没有运行，所以出现乱码

用IDA动态调试（debugger），下断点

```
.text:00401094                 jz      short loc_4010B9
.text:00401096
.text:00401096 loc_401096:                             ; CODE XREF: _main+50↑j
.text:00401096                 inc     ecx
.text:00401097                 inc     ecx
.text:00401098                 inc     ecx
.text:00401099                 inc     ecx
.text:0040109A                 int     3               ; Trap to Debugger
.text:0040109B                 mov     edx, [ebp+lpMem]
.text:0040109E                 call    sub_401000
.text:004010A3                 jmp     short loc_4010EF
.text:004010A5 ; ---------------------------------------------------------------------------
.text:004010A5                 push    2               ; uType
.text:004010A7                 push    offset Caption  ; "Flag"
.text:004010AC                 push    [ebp+lpMem]     ; lpText
.text:004010AF                 push    0               ; hWnd
.text:004010B1                 call    ds:MessageBoxA
.text:004010B7                 jmp     short loc_4010CD
.text:004010B9 ; ---------------------------------------------------------------------------
.text:004010B9
.text:004010B9 loc_4010B9:                             ; CODE XREF: _main+5A↑j
.text:004010B9                 push    2               ; uType
.text:004010BB                 push    offset Caption  ; "Flag"
.text:004010C0                 mov     eax, [ebp+lpMem]
.text:004010C3                 inc     eax
.text:004010C4                 push    eax             ; lpText
.text:004010C5                 push    0               ; hWnd
.text:004010C7                 call    ds:MessageBoxA
```


跳过进入判定后的ExitProcess()，直接跳到下面弹出messageboxA的地方，就能得到flag了

```
.text:004010A3 jmp     short loc_4010EF
.text:004010A5 ; ---------------------------------------------------------------------------
.text:004010A5 push    2                               ; uType
.text:004010A7 push    offset Caption                  ; "Flag"
.text:004010AC push    [ebp+lpMem]                     ; lpText
.text:004010AF push    0                               ; hWnd
.text:004010B1 call    ds:MessageBoxA
.text:004010B7 jmp     short loc_4010CD
.text:004010B9 ; ---------------------------------------------------------------------------
.text:004010B9
.text:004010B9 loc_4010B9:                             ; CODE XREF: _main+5A↑j
.text:004010B9 push    2                               ; uType
.text:004010BB push    offset Caption                  ; "Flag"
.text:004010C0 mov     eax, [ebp+lpMem]
.text:004010C3 inc     eax
```

0xB maze

看题目就知道是一个迷宫题

IDA中找主函数

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  const char *v3; // rsi
  signed __int64 v4; // rbx
  signed int v5; // eax
  char v6; // bp
  char v7; // al
  const char *v8; // rdi
  __int64 v10; // [rsp+0h] [rbp-28h]

  v10 = 0LL;
  puts("Input flag:");
  scanf("%s", &s1, 0LL);
  if ( strlen(&s1) != 24 || (v3 = "nctf{", strncmp(&s1, "nctf{", 5uLL)) || *(&byte_6010BF + 24) != 125 )
  {
LABEL_22:
    puts("Wrong flag!");
    exit(-1);
  }
  v4 = 5LL;
  if ( strlen(&s1) - 1 > 5 )
  {
    while ( 1 )
    {
      v5 = *(&s1 + v4);
      v6 = 0;
```

```
    if ( v5 > 78 )
    {
      v5 = (unsigned __int8)v5;
      if ( (unsigned __int8)v5 == 79 )
```


看到这四个不同函数的判断，猜测应该就是上下左右了

```
      v5 = (unsigned __int8)v5;                 // v5转成Int值
      if ( (unsigned __int8)v5 == 'O' )
      {
        v7 = sub_400650((char *)&v10 + 4, v3);
        goto LABEL_14;
      }
      if ( v5 == 'o' )
      {
        v7 = sub_400660((char *)&v10 + 4, v3);
        goto LABEL_14;
      }
    }
    else
    {
      v5 = (unsigned __int8)v5;
      if ( (unsigned __int8)v5 == '.' )
      {
        v7 = sub_400670(&v10, v3);
        goto LABEL_14;
      }
      if ( v5 == '0' )
      {
        v7 = sub_400680(&v10, v3);
LABEL_14:
        v6 = v7;
```


分别点进去查看，得知'O'为左，'o'为右，'.'为上，'0'为下，并且在函数中有规定了8这个边界值

IDA View-A | Pseudocode-A | Hex V

```
1 bool __fastcall sub_400650(_DWORD *a1)
2 {
3   int v1; // eax
4
5   v1 = (*a1)--;
6   return v1 > 0;
7 }
```


接下来找迷宫和终点

```
    if ( ++v3 >= strlen(&s1) - 1 )
    {
      if ( v5 )
        break;
LABEL_20:
      v7 = "Wrong flag!";
      goto LABEL_21;
    }
```

```
    }
  }
  if ( asc_601060[8 * (signed int)v9 + SHIDWORD(v9)] != '#' )// 终点是'#'
    goto LABEL_20;
  v7 = "Congratulations!";
LABEL_21:
  puts(v7);
  return 0LL;
}
```


通过最终判定的条件，可以知道终点是'#'

```
0000601050 ; Segment type: Pure data
0000601050 ; Segment permissions: Read/Write
0000601050 _data           segment para public 'DATA' use64
0000601050                 assume cs:_data
0000601050                 ;org 601050h
0000601050                 align 20h
0000601060 asc_601060      db '  *******   *  **** * ****  * ***  *#  *** *** ***     *********',0
0000601060                                       ; DATA XREF: main+112↑o
0000601060                                       ; main+147↑r
0000601060 _data           ends
0000601060
0006010A1 ; ============================================================================
0006010A1
0006010A1 ; Segment type: Pure data
```

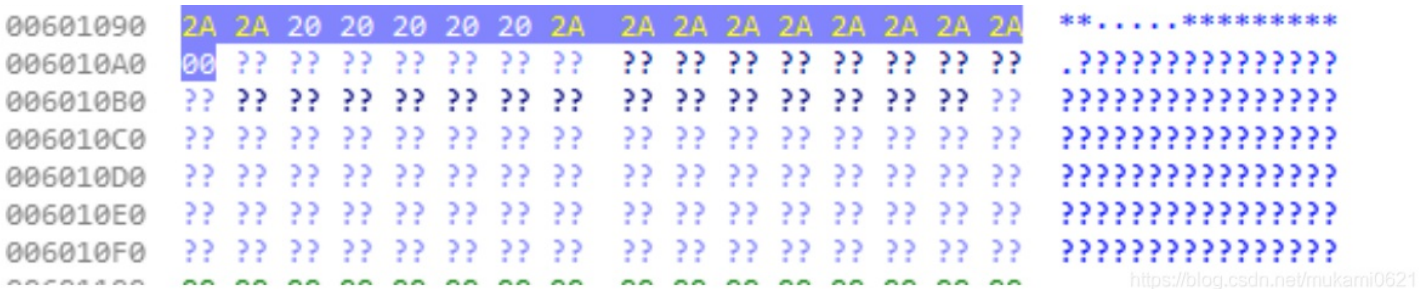

601060就是需要找的迷宫

```
      }
    }
LABEL_15:
    v3 = (const char *)HIDWORD(v10);
    if ( !(unsigned __int8)sub_400690((__int64)asc_601060, SHIDWORD(v10), v10) )
      goto LABEL_22;
    if ( ++v4 >= strlen(&s1) - 1 )
    {
      if ( v6 )
        break;
LABEL_20:
      v8 = "Wrong flag!";
```

```
1  __int64 __fastcall sub_400690(__int64 a1, int a2, int a3)
2  {
3    __int64 result; // rax
4
5    result = *(unsigned __int8 *)(a1 + a2 + 8LL * a3);
6    LOBYTE(result) = (_DWORD)result == 32 || (_DWORD)result == 35;
7    return result;
8  }
```


在HEX界面查看

```
00601040  28 11 60 00 00 00 00 00  00 00 00 00 00 00 00 00  (.`.............
00601050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00601060  20 20 2A 2A 2A 2A 2A 2A  2A 20 20 20 2A 20 20 2A  ..*******...*..*
00601070  2A 2A 2A 20 2A 20 2A 2A  2A 2A 20 20 2A 20 2A 2A  ***.*.****..*.**
00601080  2A 20 20 2A 23 20 20 2A  2A 2A 20 2A 2A 2A 20 2A  *..*#..***.***.*
```



根据边界值8重新排列，23就是终点，只走20即可得flag