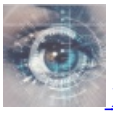
XCTF Hello CTF

原创

YenKoc 于 2020-01-15 15:13:18 发布 778 收藏 1

分类专栏： XCTF

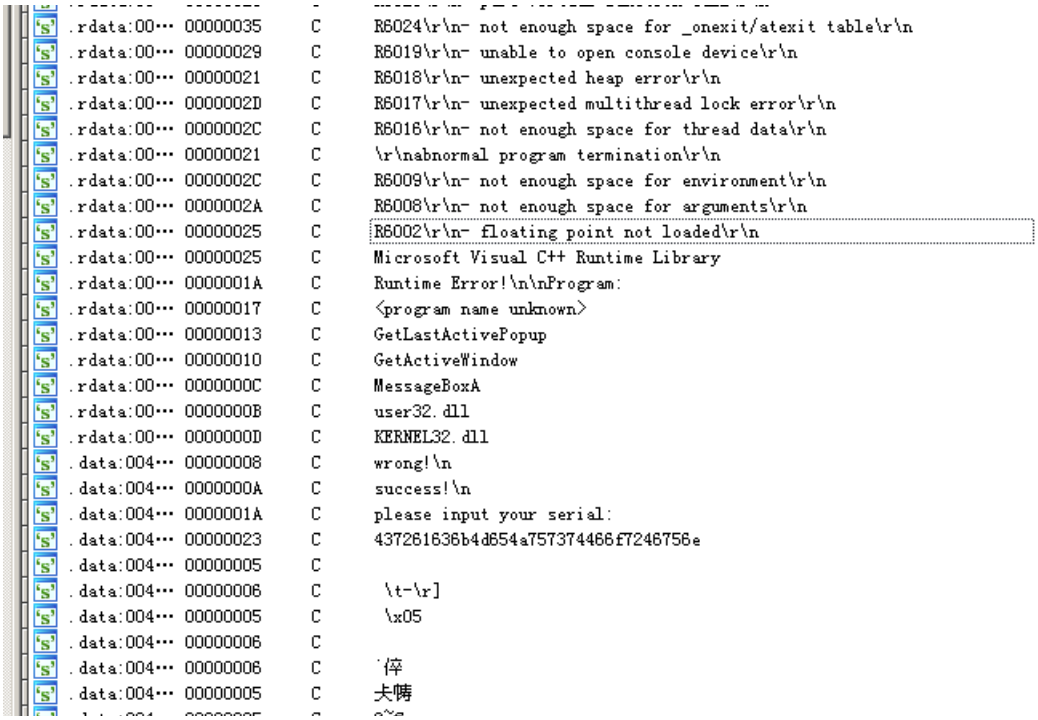
本文链接：https://blog.csdn.net/YenKoc/article/details/103989710



一.查壳

二.拖入**ida x86**静态分析

shift +F12找到字符串。



```
.data:004… 00000005  C  @ ₹
.data:004… 00000006  C  粒冢
.data:004… 00000006  C  粒冢
```


发现关键字please input your serial

点击进入

```
.text:0040101A                                  ; _main+108↓j
.text:0040101A                  mov     ecx, 8
.text:0040101F                  xor     eax, eax
.text:00401021                  lea     edi, [esp+70h+var_48]
.text:00401025                  push    offset aPleaseInputYou ; "please input your serial:"
.text:0040102A                  rep stosd
.text:0040102C                  stosw
.text:0040102E                  stosb
.text:0040102F                  call    sub_40134B
.text:00401034                  lea     eax, [esp+74h+var_5C]
.text:00401038                  push    eax
.text:00401039                  push    offset aS       ; "%s"
.text:0040103E                  call    _scanf
.text:00401043                  lea     edi, [esp+7Ch+var_5C]
.text:00401047                  or      ecx, 0FFFFFFFFh
.text:0040104A                  xor     eax, eax
.text:0040104C                  add     esp, 0Ch
.text:0040104F                  repne scasb
.text:00401051                  not     ecx
.text:00401053                  dec     ecx
.text:00401054                  cmp     ecx, 11h
.text:00401057                  ja      loc_40110D
.text:0040105D                  xor     ebx, ebx
.text:0040105F
.text:0040105F loc_40105F:                      ; CODE XREF:  main+AE↓j
```


这是我们的主函数，F5反编译一下，看一下逻辑。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  signed int v3; // ebx@3
  char v4; // al@4
  int result; // eax@9
  int v6; // [sp+0h] [bp-70h]@0
  int v7; // [sp+0h] [bp-70h]@2
  char v8; // [sp+12h] [bp-5Eh]@5
  char v9[20]; // [sp+14h] [bp-5Ch]@2
  char v10; // [sp+28h] [bp-48h]@2
  __int16 v11; // [sp+48h] [bp-28h]@2
  char v12; // [sp+4Ah] [bp-26h]@2
  char v13; // [sp+4Ch] [bp-24h]@1

  qmemcpy(&v13, a437261636b4d65, 0x23u);
  while ( 1 )
  {
    memset(&v10, 0, 0x20u);
    v11 = 0;
    v12 = 0;
    sub_40134B((int)aPleaseInputYou, v6);
    scanf(aS, v9);
    if ( strlen(v9) > 0x11 )          char[26]
      break;
    v3 = 0;
    do
    {
      v4 = v9[v3];
      if ( !v4 )
        break;
      sprintf(&v8, asc_408044, v4);
      strcat(&v10, &v8);
      ++v3;
    }
    while ( v3 < 17 );
    if ( !strcmp(&v10, &v13) )
      sub_40134B((int)aSuccess, v7);
    else
      sub_40134B((int)aWrong, v7);
  }
  sub_40134B((int)aWrong, v7);
  result = stru_408090._cnt-- - 1;
  if ( stru_408090._cnt < 0 )
    result = _filbuf(&stru_408090);
  else
    ++stru_408090._ptr;
```

```
return result;
```


这里有几个函数，我也不懂啥意思，百度的.

1.qmemcpy(void *destin ,void *source)：将source指针所指的值拷贝到destin中去

2.sprintf（char *str, char * format [, argument, …])，在本文的意思是将v4的字符串以十六进制的形式存入v8中。

三.

那么逻辑已经很明显了，V13就是我们的flag，只不过转成了十六进制，只需要将十六进制转成字符串就好了。