XCTF EasyRE

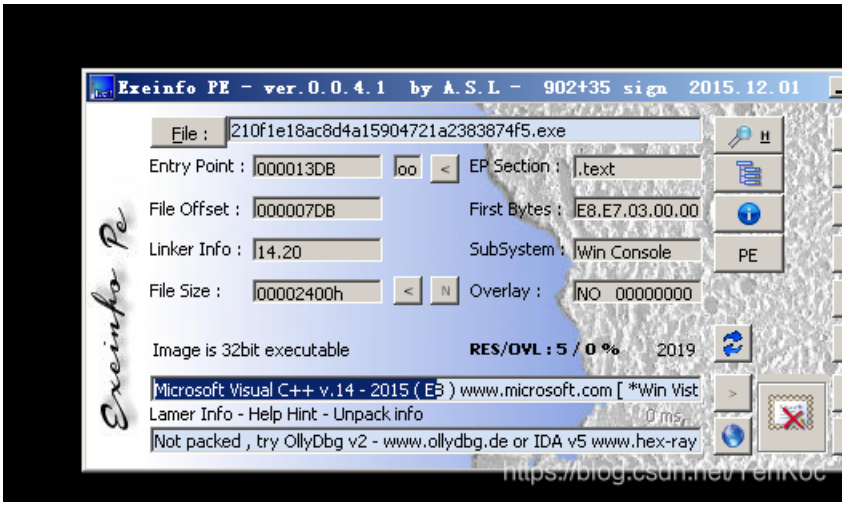
原创

YenKoc 于 2020-03-29 19:04:50 发布 372 收藏

分类专栏： XCTF

本文链接：https://blog.csdn.net/YenKoc/article/details/105183873

一.查壳



无壳，并且发现是vc++编译的

二.拖入ida，来静态分析，这题让我深刻感觉到汇编的nb。

Pseudocode-A

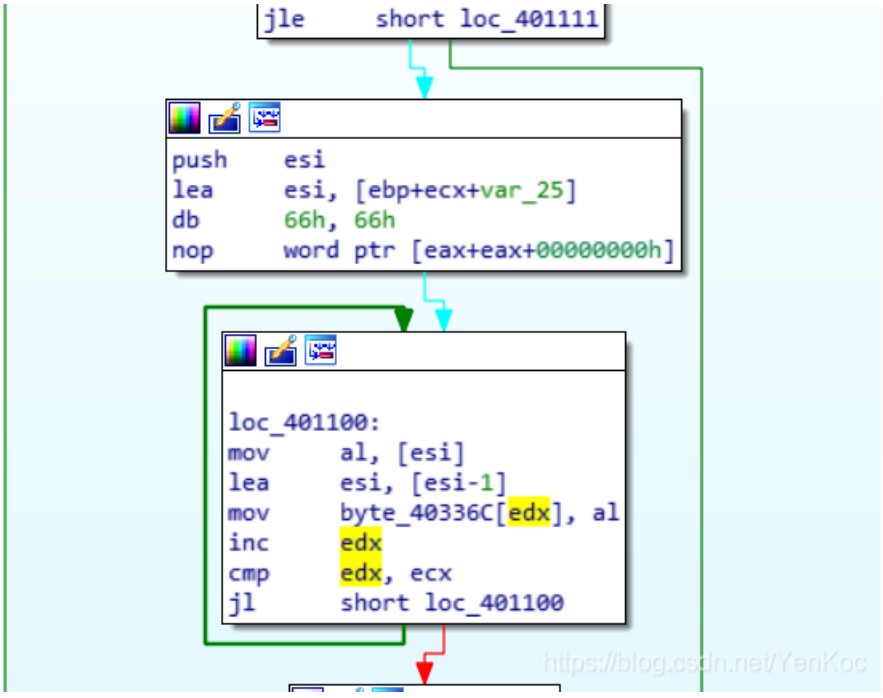
```
1  int sub_401080()
2  {
3    unsigned int v0; // kr00_4
4    signed int v1; // edx
5    char *v2; // esi
6    char v3; // al
7    unsigned int v4; // edx
8    int v5; // eax
9    __int128 v7; // [esp+2h] [ebp-24h]
10   __int64 v8; // [esp+12h] [ebp-14h]
11   int v9; // [esp+1Ah] [ebp-Ch]
12   __int16 v10; // [esp+1Eh] [ebp-8h]
13
14   sub_401020((int)&unk_402150);
15   v9 = 0;
16   v10 = 0;
17   v7 = 0i64;
18   v8 = 0i64;
19   sub_401050((const char *)&unk_402158, &v7);
20   v0 = strlen((const char *)&v7);
21   if ( v0 >= 16 && v0 == 24 )
22   {
23     v1 = 0;
24     v2 = (char *)&v8 + 7;
25     do
26     {
```

```
    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_401020((int)"right\n");
    system("pause");
  }
 }
 return 0;
}
```



```
if ( v0 >= 16 && v0 == 24 )
{
  v1 = 0;
  v2 = (char *)&v8 + 7;
  do
  {
    v3 = *v2--;
    byte_40336C[v1++] = v3;
  }
  while ( v1 < 24 );
  v4 = 0;
```

这段算是灵性的一段了，单从静态语句来看，发现分析不出啥，只能靠猜一下，我当时猜的是将输入的字符串又赋值到一个新的字符数组里了，后面，写脚本就出现了问题，一直乱码，又返回分析，结合**wp**才知道，这玩意，要看汇编，我又把汇编看了一遍，才知道原来是倒序**2333**，



就是这段，认真分析下来，就能知道，然后写出脚本**flag**就出来了。

三.脚本

```
res=[0x78, 0x49, 0x72, 0x43, 0x6A, 0x7E, 0x3C, 0x72, 0x7C, 0x32, 0x74, 0x57, 0x73, 0x76, 0x33, 0x50, 0x74, 0x49,
 0x7F, 0x7A, 0x6E, 0x64, 0x6B, 0x61]
print(len(res))
flag=""
for i in range(len(res)):
    flag+=chr((res[23-i]^6)-1)
print(flag)
```

四.总结

这题本身加密算法是不难的，一个异或和加法，算入门，感觉考的就是会不会看汇编，这也是我的思维定式了，发现一上来就**F5**，然后不去看汇编，以后要改进。