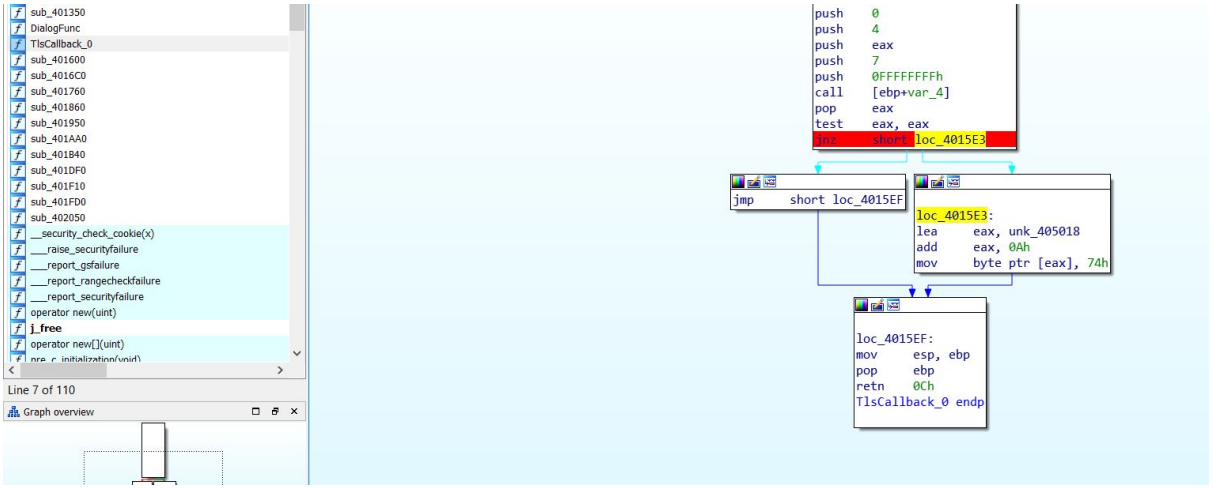
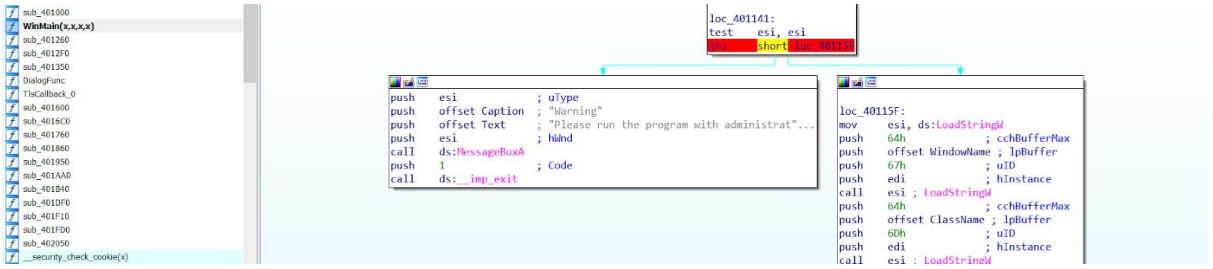
ANTIDEBUG

1. tlscallback



=> sửa jnz short loc_4015E3 thành jmp loc_4015EF

2. check admin



=> sửa jnz short loc_40115F thành jmp short loc_40115F

3. Hàm check

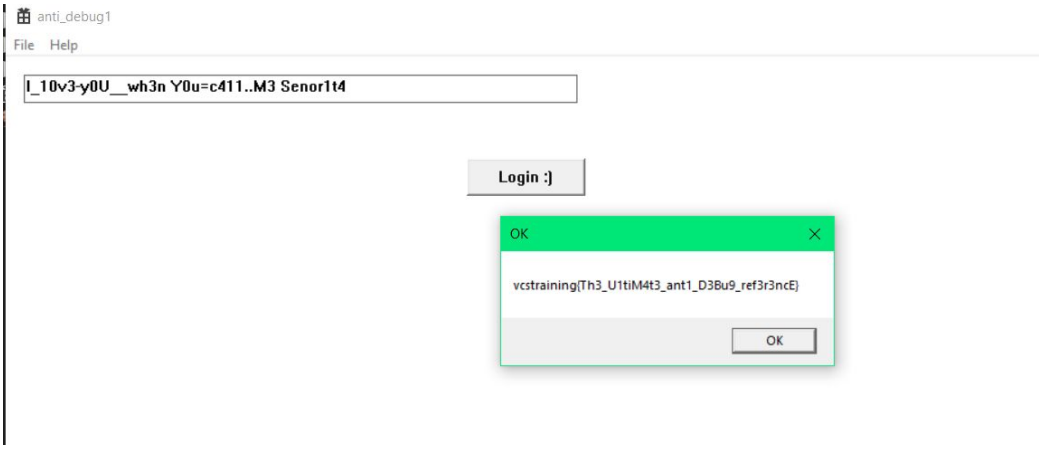
Có 7 case check debug:

- Case 0: Hàm check **NtGlobalFlag**, nếu như nó bằng 0x70 thì là chạy trên debug.
- Case 1: Check **heap flag**, nếu nó bằng 0x40000062 thì nó chạy trên debug
- Case 2: Check **heap force flag**, nếu nó bằng 0x40000060 thì nó chạy trên debug
- Case 3: Check **heap protection**, nếu HEAP TAIL CHECKING ENABLED bằng 0xABABABAB thì có debug.
- Case 4: Check **parent process** tên có bằng với cái của nó không. Nếu đúng thì đang chạy debug.
- Case 5: **BlockInput**
- Case 6: **NtQueryProcessInformation**

=> Sau khi sửa các anti debug ở trên, chạy debug để tìm ra các byte kí tự mà input xor với, ta tìm được pass cần nhập:

I_10v3-y0U__wh3n Y0u=c411..M3 Senor1t4

=> Flag: ***vcstraining{Th3_U1tiM4t3_ant1_D3Bu9_ref3r3ncE}***



Python script:

```
# test string: I_10v3-y0U__wh3n Y0u=c411..M3 Senor1t4
list_str = ['_'] * 0x26

dump_hex = [91, 219, 157, 198, 167, 90, 138, 246, 13, 165, 218, 116, 233, 207, 88, 150, 91, 90, 208, 252, 37, 246, 84, 184, 110, 204, 122, 63, 164, 30, 115, 63, 16, 231, 241, 33, 182, 232]

check_dump = bv.read(0x4032c8, 0x130)
check = [struct.unpack('<I', check_dump[x:x+4])[0] for x in range(0, len(check_dump), 4)]
check = map(lambda x: x-1, check)

location_dump = bv.read(0x4033f8, 0x98)
location = [struct.unpack('<I', location_dump[x:x+4])[0] for x in range(0, len(location_dump), 4)]

xorred = bv.read(0x4032a0, 0x26)
dump_xor = map(ord, xorred)

result = map(lambda x: x[0] ^ x[1], zip(dump_xor, dump_hex))
result = map(chr, result)

final_result = [''] * 0x26
for i, _ in enumerate(result):
    final_result[location[i]] += result[i]

''.join(final_result)
```