[HW] trace

一開始我用 ida pro 打開 trace 發現 0x1288 有 analysis false 的問題，disassemble 後找不出哪裡”事有蹊俏”，因為我對 ida 也不熟，所以就用 ghidra 跑了。用 ghidra 看了 0x1288 後知道他會寫入東西，然後我發現寫入的 function 裡有另一個 binary，因此我用

binwalk --dd=".*" trace_63ae7693f94d1f0b

後出現了 3020 的檔案

再放進 ghidra，看到了 INT3 中斷點，disassemble 下面的問號後可以看到下面的 function

每個點進去看一下，最後在 FUN_004011ca 看到了 0x04011d9 的 SUB_013c57a5，然後好好看 code 後知道他會固定從 0x404050 取值並 xor 0x71 因此我寫了個 script 推回去就得出了 flag。

```python
from ast import Bytes


n=[0x37,0x3d,0x30,0x36,0x0a,0x25,0x03,0x30,0x12,0x42,0x2e,0x3c,0x42,0x2e,0x40,0x37,0x2e,0x24,0x2e,0x12,0x30,0x3f,0x0c,0x00]
# for k in [0,24]:
for i in range(len(n)):
    n[i]=n[i]^0x71

print(bytes(n))
```

```
/* WARNING: Instruction at (ram,0x004011e3) overlaps instruction at (ram,0x00401
    */

void UndefinedFunction_004011d9(void)

{
  char *pcVar1;
  code *pcVar2;
  char *pcVar3;
  undefined2 uVar4;
  uint in_ECX;
  char unaff_BL;
  undefined7 unaff_00000019;
  long unaff_RBP;

  do {
    uVar4 = (undefined2)(in_ECX >> 8);
    pcVar3 = (char *)func_0x013c57a5();
    *pcVar3 = *pcVar3 + (char)pcVar3;
    unaff_BL = unaff_BL + (char)((ushort)uVar4 >> 8);
    pcVar1 = (char *)(CONCAT71(unaff_00000019,unaff_BL) + -0x67b703bb);
    *pcVar1 = *pcVar1 - (char)uVar4;
    in_ECX = (byte)pcVar3[0x404050] ^ 0x71;
    (&DAT_00404050)[*(int *)(unaff_RBP + -4)] = (char)in_ECX;
    *(int *)(unaff_RBP + -4) = *(int *)(unaff_RBP + -4) + 1;
  } while (*(uint *)(unaff_RBP + -4) < 0x17);
  func_0xffffffffdeedd107();
  pcVar2 = (code *)swi(3);
  (*pcVar2)();
  return;
}
```

上圖為 SUB_013c57a5 的內容

```
❯ python3 solve.py
b'FLAG{TrAc3_M3_1F_U_cAN}q'
```

最後的 flag

[LAB] Sacred Arts

這題前面跟著講師做了很多，最後在這裡知道 flag 被加密的操作

```
1
1 loc_4010D1:                           ; CODE XREF: .text:00000000004010EC↓j
1                 lea     rdx, ds:0FFFFFFFFFFFFFFF8h[rcx*8]
9                 mov     rax, [rsp+rdx]
D                 neg     rax
0                 xchg    al, ah
2                 cmp     rax, [rbx+rdx]
6                 jnz     near ptr unk_401035
C                 loop    loc_4010D1
E                 jmp     short loc_4010FD
```

將前面 disassemble 後變 array 的值 export 出來

```
arr             dq 8D909984B8BEBAB3h
                dq 8D9A929E98D18B92h
                dq 0D0888BD19290D29Ch
                dq 8C9DC08F978FBDD1h
                dq 0D9C7C7CCCDCB92C2h
                dq 0C8CFC7CEC2BE8D91h
                dq 0FFFFFFFFFFFFCF82h
```

將前面加密的操作否著做回去就得出了 flag。

```
ea=0x40108b
enc_flag = idc.get_bytes(ea,8*7)
enc_list = [enc_flag[i:i+8] for i in range(0,len(enc_flag),8)]

mask= (1<< 64)-1
for el in enc_list:
    # print(el)
    el=el[0:2][::-1]+el[2:]
    # print(el)

    el=int.from_bytes(el,'little')
    el^=mask
    el=(el+1) & mask
    print(int.to_bytes(el,8,'little').decode())
```

```
FLAG{for
um.gamer
.com.tw/
C.php?bs
n=42388&
snA=1807
1}
```