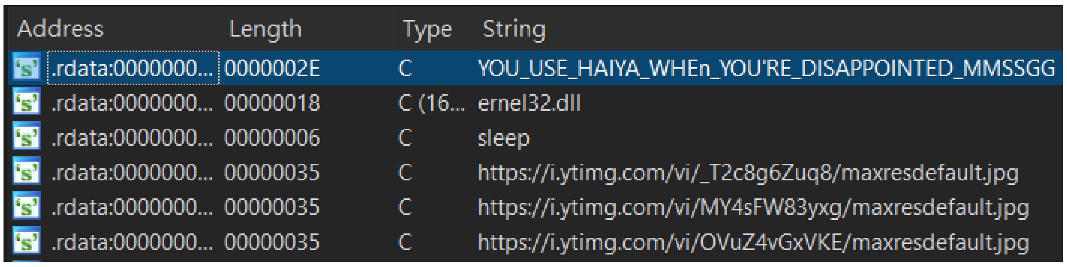
giveUflag

- 直接執行, 沒有任何反應, 但也沒有馬上退出
- 觀察一下字串、Import functions, 有幾個比較特別的字串:



Address	Length	Type	String
.rdata:0000000...	0000002E	C	YOU_USE_HAIYA_WHEn_YOU'RE_DISAPPOINTED_MMSSGG
.rdata:0000000...	00000018	C (16...	ernel32.dll
.rdata:0000000...	00000006	C	sleep
.rdata:0000000...	00000035	C	https://i.ytimg.com/vi/_T2c8g6Zuq8/maxresdefault.jpg
.rdata:0000000...	00000035	C	https://i.ytimg.com/vi/MY4sFW83yxg/maxresdefault.jpg
.rdata:0000000...	00000035	C	https://i.ytimg.com/vi/OVuZ4vGxVKE/maxresdefault.jpg

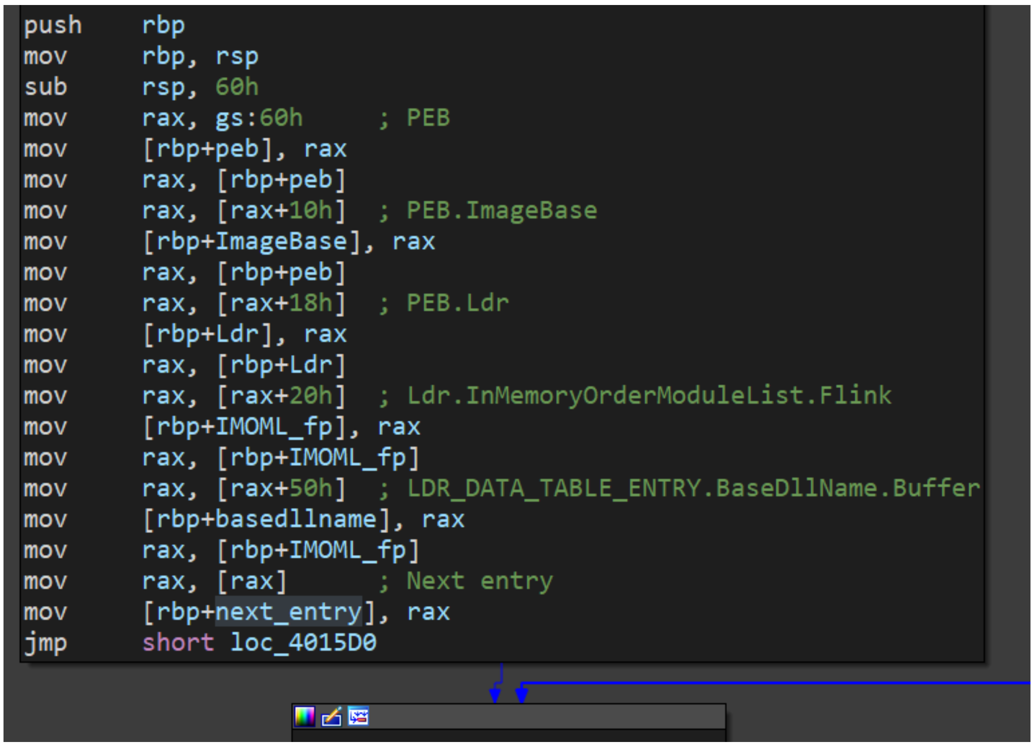
- 0x401870 為 main:

```
main proc near
push    rbp
mov     rbp, rsp
sub     rsp, 20h
call    sub_401940
call    sub_40184C
mov     eax, 0
add     rsp, 20h
pop     rbp
retn
main endp
```

- `sub_40184C` :

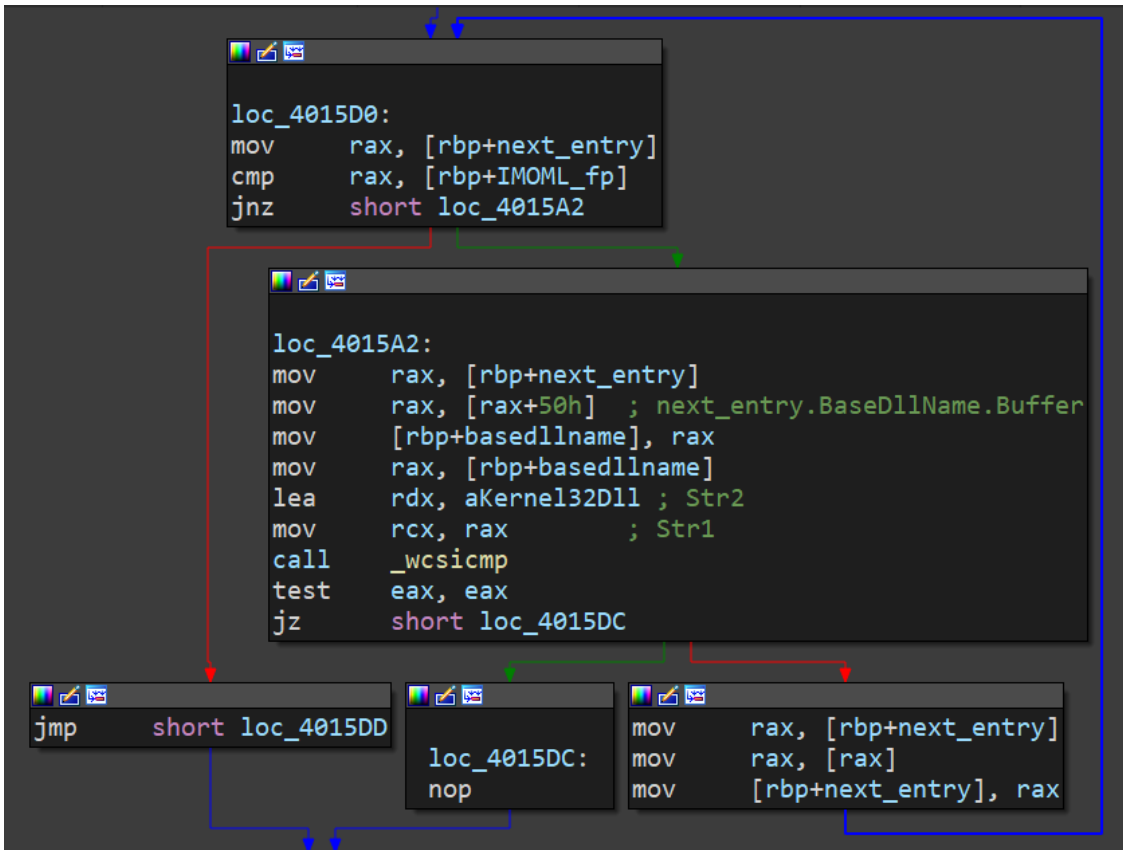
```
sub_40184C proc near
push    rbp
mov     rbp, rsp
sub     rsp, 30h
call    find_kernel32_dll_base
mov     [rbp+kernel32_base], rax
mov     rax, [rbp+kernel32_base]
mov     rcx, rax
call    main_logic
nop
add     rsp, 30h
pop     rbp
retn
sub_40184C endp
```

- `find_kernel32_dll_base` 從 PEB 爬出 kernel32.dll ImageBase:



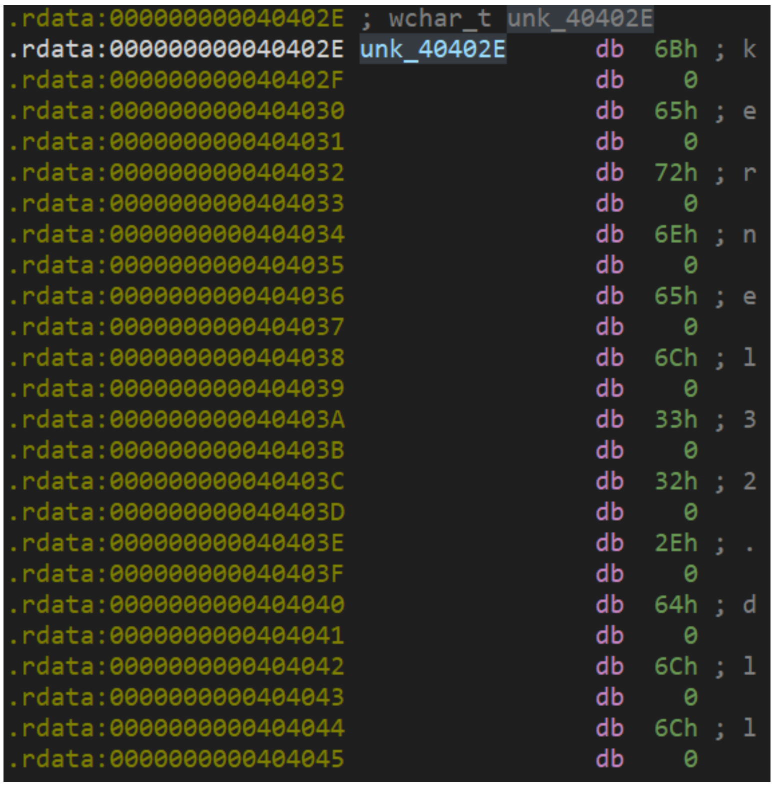
```
push    rbp
mov     rbp, rsp
sub     rsp, 60h
mov     rax, gs:60h     ; PEB
mov     [rbp+peb], rax
mov     rax, [rbp+peb]
mov     rax, [rax+10h]  ; PEB.ImageBase
mov     [rbp+ImageBase], rax
mov     rax, [rbp+peb]
mov     rax, [rax+18h]  ; PEB.Ldr
mov     [rbp+Ldr], rax
mov     rax, [rbp+Ldr]
mov     rax, [rax+20h]  ; Ldr.InMemoryOrderModuleList.Flink
mov     [rbp+IMOML_fp], rax
mov     rax, [rbp+IMOML_fp]
mov     rax, [rax+50h]  ; LDR_DATA_TABLE_ENTRY.BaseDllName.Buffer
mov     [rbp+basedllname], rax
mov     rax, [rbp+IMOML_fp]
mov     rax, [rax]      ; Next entry
mov     [rbp+next_entry], rax
jmp     short loc_4015D0
```

- 上圖為爬出 PEB, PEB.Ldr ...



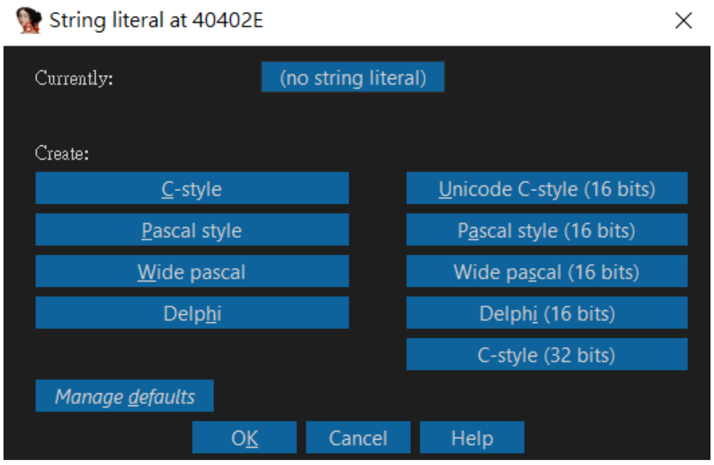
```
loc_4015D0:
mov     rax, [rbp+next_entry]
cmp     rax, [rbp+IMOML_fp]
jnz     short loc_4015A2

loc_4015A2:
mov     rax, [rbp+next_entry]
mov     rax, [rax+50h]  ; next_entry.BaseDllName.Buffer
mov     [rbp+basedllname], rax
mov     rax, [rbp+basedllname]
lea     rdx, aKernel32Dll ; Str2
mov     rcx, rax        ; Str1
call    _wcsicmp
test    eax, eax
jz      short loc_4015DC

jmp     short loc_4015DD

loc_4015DC:
nop

mov     rax, [rbp+next_entry]
mov     rax, [rax]
mov     [rbp+next_entry], rax
```

- 上圖為遍尋 `_LDR_DATA_TABLE_ENTRY` 鏈表, 直到找到 `BaseDllName.Buffer` 字串為 `Kernel32.dll` 的 entry 才跳出迴圈
- `aKernel32Dll` 是 wide char array, 在 IDA 中, 將其 undefine 後如下圖:

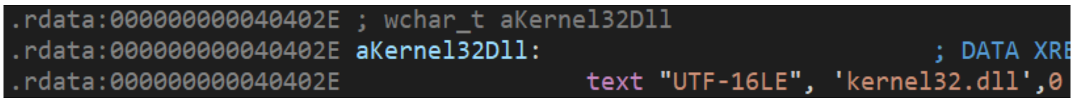


```
.rdata:000000000040402E ; wchar_t unk_40402E
.rdata:000000000040402E unk_40402E      db  6Bh ; k
.rdata:000000000040402F                 db    0
.rdata:0000000000404030                 db  65h ; e
.rdata:0000000000404031                 db    0
.rdata:0000000000404032                 db  72h ; r
.rdata:0000000000404033                 db    0
.rdata:0000000000404034                 db  6Eh ; n
.rdata:0000000000404035                 db    0
.rdata:0000000000404036                 db  65h ; e
.rdata:0000000000404037                 db    0
.rdata:0000000000404038                 db  6Ch ; l
.rdata:0000000000404039                 db    0
.rdata:000000000040403A                 db  33h ; 3
.rdata:000000000040403B                 db    0
.rdata:000000000040403C                 db  32h ; 2
.rdata:000000000040403D                 db    0
.rdata:000000000040403E                 db  2Eh ; .
.rdata:000000000040403F                 db    0
.rdata:0000000000404040                 db  64h ; d
.rdata:0000000000404041                 db    0
.rdata:0000000000404042                 db  6Ch ; l
.rdata:0000000000404043                 db    0
.rdata:0000000000404044                 db  6Ch ; l
.rdata:0000000000404045                 db    0
```

- 對其位址按下 `alt` + `a` 叫出以下畫面:



將其解析成 16 bits unicode:

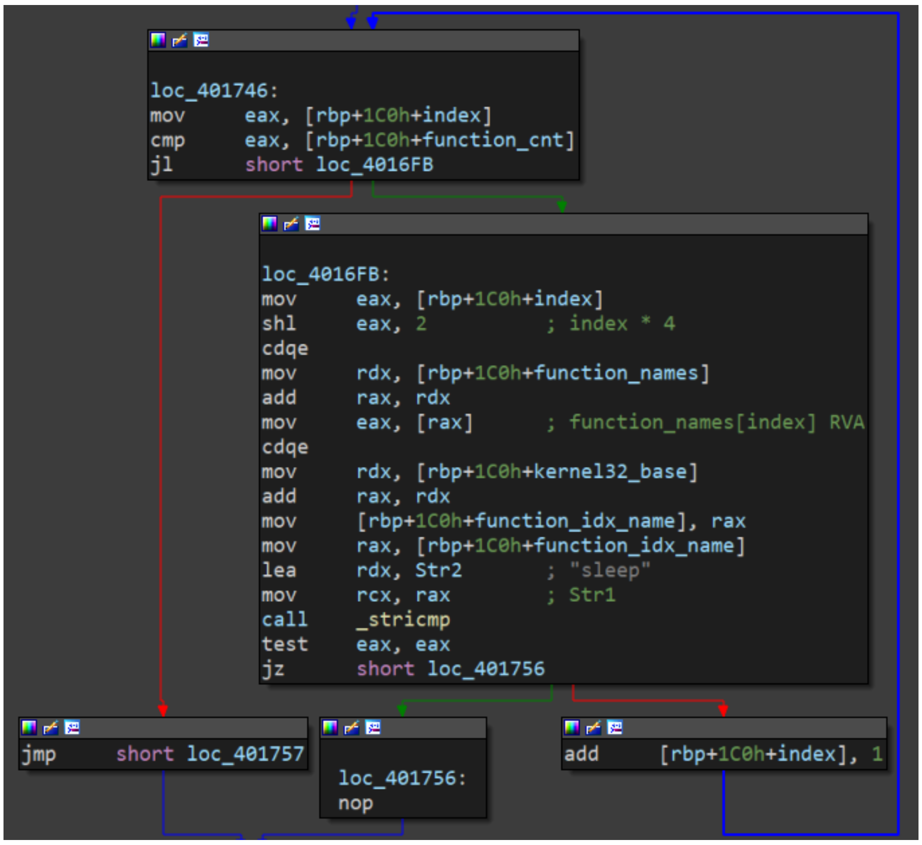


```
.rdata:000000000040402E ; wchar_t aKernel32Dll
.rdata:000000000040402E aKernel32Dll:                              ; DATA XRE
.rdata:000000000040402E                   text "UTF-16LE", 'kernel32.dll',0
```

- `main_logic`:
  - 第一塊 basic block 由於 code 蠻長的, 故不用截圖的, 如下:

```
push    rbp
push    rdi
sub     rsp, 238h
lea     rbp, [rsp+80h]
mov     [rbp+1C0h+kernel32_base], rcx
lea     rax, [rbp+1C0h+array_b4]
lea     rdx, array_iv   ; Src
mov     ecx, 0B4h
mov     r8, rcx         ; Size
mov     rcx, rax        ; Dst
call    memcpy
lea     rdx, [rbp+1C0h+str_20]
mov     eax, 0
mov     ecx, 20h ; ' '
mov     rdi, rdx
rep stosq
mov     rax, [rbp+1C0h+kernel32_base]
add     rax, 3Ch ; '<'
mov     eax, [rax]      ; file address of new exe header
cdqe
mov     rdx, [rbp+1C0h+kernel32_base]
add     rax, rdx
mov     [rbp+1C0h+pe_hdr], rax ; PE_Hdr
mov     rax, [rbp+1C0h+pe_hdr]
add     rax, 88h
mov     eax, [rax]      ; Export Dir
cdqe
mov     rdx, [rbp+1C0h+kernel32_base]
add     rax, rdx
mov     [rbp+1C0h+export_dir], rax
mov     rax, [rbp+1C0h+export_dir]
add     rax, 0Ch        ; export_dir.Name
mov     eax, [rax]
cdqe
mov     rdx, [rbp+1C0h+kernel32_base]
add     rax, rdx
mov     [rbp+1C0h+export_dir_name], rax
mov     rax, [rbp+1C0h+export_dir]
mov     eax, [rax+14h]  ; NumberOfFunctions
mov     [rbp+1C0h+function_cnt], eax
mov     rax, [rbp+1C0h+export_dir]
add     rax, 1Ch        ; AddressOfFunctions
mov     eax, [rax]
cdqe
mov     rdx, [rbp+1C0h+kernel32_base]
add     rax, rdx
mov     [rbp+1C0h+function_addrs], rax
mov     rax, [rbp+1C0h+export_dir]
add     rax, 20h ; ' '  ; AddressOfNames
mov     eax, [rax]
cdqe
mov     rdx, [rbp+1C0h+kernel32_base]
add     rax, rdx
mov     [rbp+1C0h+function_names], rax
mov     [rbp+1C0h+index], 0
jmp     short loc_401746
```

    - 從第一個參數 rcx 取得剛剛爬到的 kernel32.dll base, 接著爬其 Export Directory, 取得各種資訊 e.g. NumberOfFunctions、AddressOfNames ...
    - 前面有初始化一個陣列
- 繼續往下看



```
loc_401746:
mov     eax, [rbp+1C0h+index]
cmp     eax, [rbp+1C0h+function_cnt]
jl      short loc_4016FB
```

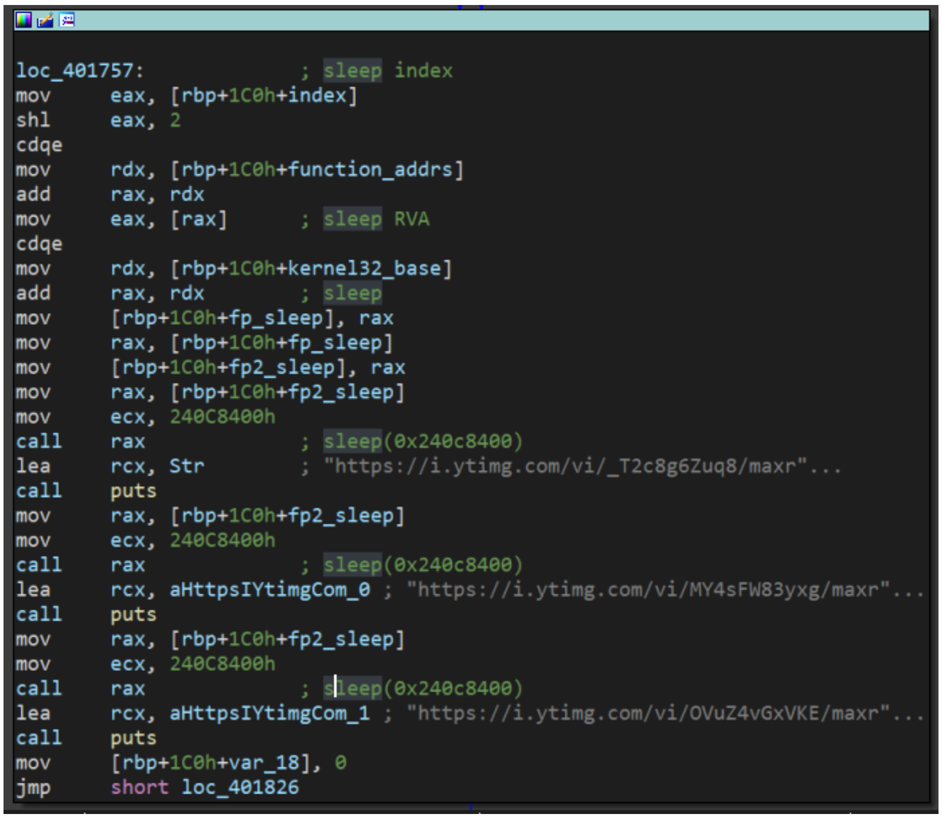
```
loc_4016FB:
mov     eax, [rbp+1C0h+index]
shl     eax, 2          ; index * 4
cdqe
mov     rdx, [rbp+1C0h+function_names]
add     rax, rdx
mov     eax, [rax]      ; function_names[index] RVA
cdqe
mov     rdx, [rbp+1C0h+kernel32_base]
add     rax, rdx
mov     [rbp+1C0h+function_idx_name], rax
mov     rax, [rbp+1C0h+function_idx_name]
lea     rdx, Str2       ; "sleep"
mov     rcx, rax        ; Str1
call    _stricmp
test    eax, eax
jz      short loc_401756
```

```
jmp     short loc_401757
```

```
loc_401756:
nop
```

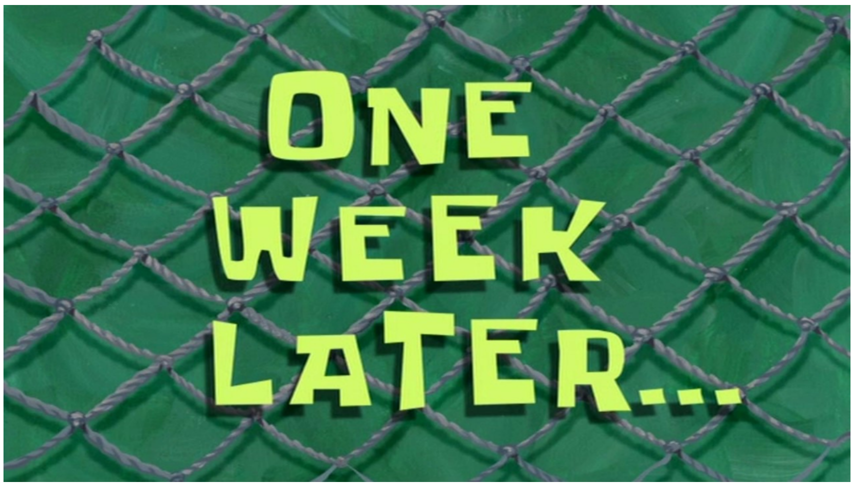
```
add     [rbp+1C0h+index], 1
```

- 上圖為爬 AddressOfNames, 取得指向的 Function Name 字串是否為 `sleep` (stricmp 不區分大小寫), 爬到則跳出迴圈
- 因此 index 為 `sleep` 的 index, 繼續往下看



```
loc_401757:             ; sleep index
mov     eax, [rbp+1C0h+index]
shl     eax, 2
cdqe
mov     rdx, [rbp+1C0h+function_addrs]
add     rax, rdx
mov     eax, [rax]      ; sleep RVA
cdqe
mov     rdx, [rbp+1C0h+kernel32_base]
add     rax, rdx        ; sleep
mov     [rbp+1C0h+fp_sleep], rax
mov     rax, [rbp+1C0h+fp_sleep]
mov     [rbp+1C0h+fp2_sleep], rax
mov     rax, [rbp+1C0h+fp2_sleep]
mov     ecx, 240C8400h
call    rax             ; sleep(0x240c8400)
lea     rcx, Str        ; "https://i.ytimg.com/vi/_T2c8g6Zuq8/maxr"...
call    puts
mov     rax, [rbp+1C0h+fp2_sleep]
mov     ecx, 240C8400h
call    rax             ; sleep(0x240c8400)
lea     rcx, aHttpsIYtimgCom_0 ; "https://i.ytimg.com/vi/MY4sFW83yxg/maxr"...
call    puts
mov     rax, [rbp+1C0h+fp2_sleep]
mov     ecx, 240C8400h
call    rax             ; sleep(0x240c8400)
lea     rcx, aHttpsIYtimgCom_1 ; "https://i.ytimg.com/vi/OVuZ4vGxVKE/maxr"...
call    puts
mov     [rbp+1C0h+var_18], 0
jmp     short loc_401826
```

- 爬出 sleep, 並且呼叫, sleep(0x240c8400) 會睡一周, 查看字串連結:
  - https://i.ytimg.com/vi/_T2c8g6Zuq8/maxresdefault.jpg

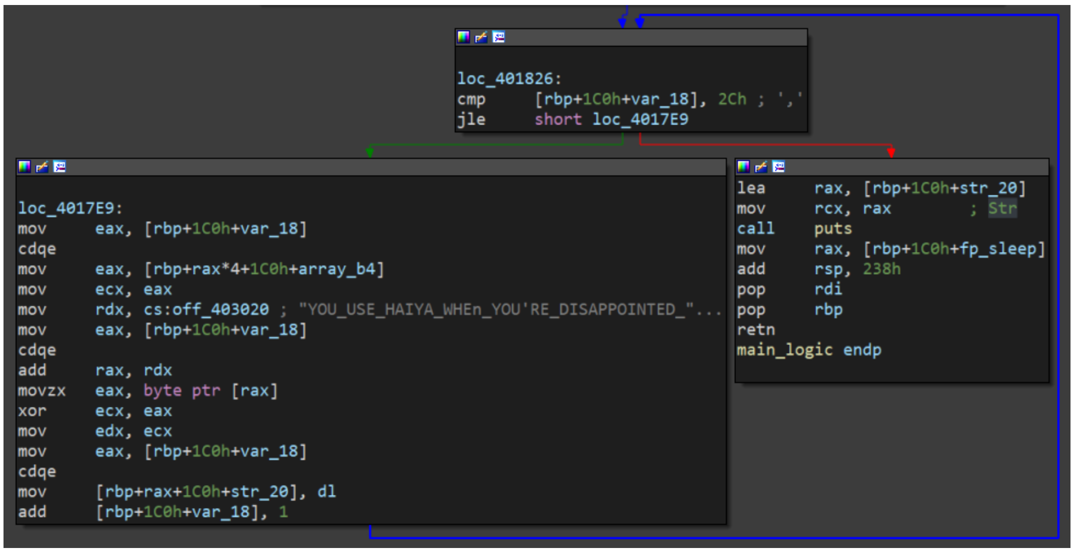


    - https://i.ytimg.com/vi/MY4sFW83yxg/maxresdefault.jpg

    - https://i.ytimg.com/vi/OVuZ4vGxVKE/maxresdefault.jpg

    - ~~睡完三周作業 deadline 就過了~~

- 接著看



```
loc_401826:
cmp     [rbp+1C0h+var_18], 2Ch ; ','
jle     short loc_4017E9
```

```
loc_4017E9:
mov     eax, [rbp+1C0h+var_18]
cdqe
mov     eax, [rbp+rax*4+1C0h+array_b4]
mov     ecx, eax
mov     rdx, cs:off_403020 ; "YOU_USE_HAIYA_WHEn_YOU'RE_DISAPPOINTED_"...
mov     eax, [rbp+1C0h+var_18]
cdqe
add     rax, rdx
movzx   eax, byte ptr [rax]
xor     ecx, eax
mov     edx, ecx
mov     eax, [rbp+1C0h+var_18]
cdqe
mov     [rbp+rax+1C0h+str_20], dl
add     [rbp+1C0h+var_18], 1
```

```
lea     rax, [rbp+1C0h+str_20]
mov     rcx, rax        ; Str
call    puts
mov     rax, [rbp+1C0h+fp_sleep]
add     rsp, 238h
pop     rdi
pop     rbp
retn
main_logic endp
```

  - 這邊用到前面初始化的陣列, 並與一段 key 做 xor, 解出 flag 明文
- 利用動態分析, 將 sleep patch 成 ret, 並在解完 flag 的地方設斷點, 即可得到 flag:
- `FLAG{PaRs1N6_PE_aNd_D11_1S_50_C00111!!!!!111}`